Zapple(tm) Resident Monitor & I/O System for Epson QX-10 with TPM-II

The Zapple(tm) I/O Monitor system was first developed in 1976, and represents years of refinement. This Z-80-based version was derived from the "Apple" I/O Monitor, an 8080-based program.
Apple, in turn, was derived from Intel's Intellec 8008 monitor.

The following list of commands are used in the current Epson QX-10 implementation of Zapple. Each version of Zapple has a slightly different repertoire of commands, due to different hardware and software demands, but anyone who has used Apple or Zapple will be comfortable with the QX-10 version.

In most cases, commands are a single letter, followed by a numeric value. These values are *always* expressed in HEXadecimal notation. Note the following valid HEX values:

0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f (or capitals A,B,C,etc.)

An incorrect entry will abort the command and cause an asterisk (*) to be printed. In most cases, no harm is done--just repeat the command from the beginning. If an error is made while inputting a number, simply continue entering until the last four digits are correct, or in the case of a byte value, the last two digits. For example: 123456 will be taken as a word value of 3456 (Hex), or a byte value of 56 (Hex). This feature is especially useful when entering the third value of a command.

Zapple is a "machine level" tool as well as an I/O handler. It contains the tools needed to insure system integrity and to "debug" both hardware and software problems. Although Zapple is available at all times, access is restricted in order to prevent someone without the understanding of machine-level programming from becoming confused or damaging data.

To access Zapple, type a control-\ (the key with the degree symbol) any time the running program is testing for console input. Zapple will "TRAP", printing the address that it will return to when you allow the program to resume. This is done with a "G<ret>" command, and since this usually occurs when the running program is expecting a character, you then type a character to cause the program to continue. Often "G<ret><ret>" is sufficient.

Zapple handles all system level input and output: it contains drivers for the video CRT, the interrupt driven keyboard, the parallel and serial ports, and the initialization code during cold boot, and it retains the status of the last I/O assignment. The entire system can be run via an external terminal (such as an ADM3, Televideo, Hazeltine, etc.), by setting the rear panel switch #1 to "UP", and the baud rate to the correct value. Use the "A" command to assign "Console=Teletype (serial port)". The system will remain this way until you assign "Console=Crt" from the external terminal. To force the *internal* console, set switch #1 *down*. This procedure is used when you have disconnected the external terminal before performing the reassignment.

The commands appearing below are listed alphabetically, and include a brief description and an example of usage. For further details of the effects of some of the hardware-related commands, refer to the *QX-10 Hardware Reference Manual*.

One final note. Zapple can be your best friend, your ally, your "man on the inside." Read the list of commands and then experiment. You can *always* hit RESET.

COMMAND SYNTAX

A - Assign I/O Devices

On the command line:

```
>A[d]=[u]
```

Assigns a device to be a particular unit.
First letter of specifier is all that is required.

[d]evice:=Console, Reader, Punch, List

[u]nit:=

if Console:	Tty, Char Video, Batch mode, Bit-mapped Video
if Reader:	Tty, Keyboard, User #1, User #2
if Punch:	Tty, Crt display, User #1, User #2
if List:	Tty, Crt display, Line printer, User #1

```
EXAMPLE:  AC=T;assigns Console to RS-232 Serial port
          AL=L;assigns list to Centronics port
```

B - Boot TPM-II

On the command line:
```
  >B
```

Only valid *once*. The command is declared void as soon as the TPM-II system has been initialized. In normal configuration, TPM-II will be "booted up" when the system is reset (and a valid boot disk is in the *left* drive). For debugging purposes, however, if the switch on the rear panel of the QX-10 mainframe is set so that switch #2 is down, Zapple will sign-on and the disk operating system will *not* be booted up. The command "B" is then used to boot the system up.

C - C-MOS Clock & Ram I/O

On the command line:
```
 >CO[addr],[val]<ret>
 >CI[addr]<ret>
```

The hardware of the QX-10 contains a battery-operated C-MOS clock chip with 50 bytes of battery backed-up RAM, which is read and written via two normal "Z-80 style" I/O ports. The "CO" (C-MOS Output) and "CI" (C-MOS Input) commands were added to make it easier to examine and/or write the values in these C-MOS RAM locations.

```
EXAMPLE:  >CO32,55;writes the value 55H to RAM loc 32H
          >CI32 01010101;shows the bit pattern at RAM loc 32H
```

D - Display Memory (in HEX)

On the command line:
```
  >D[addr1],[addr2],<byte>
```

Dumps memory from addr1 through addr2, where <byte> is optional depending on line width. Defaults to 16 bytes per line.

The ASCII representation is also displayed to the right of the HEX values. (See "T" command.)

EXAMPLE:

```
  >D0,1F
0000  C3 07 F7 C3 24 F7 C3 32 F5 C3 84 F5 C3 53 F5 C3  C.wC$wC2uC.uCSuC
0010  65 F6 DB 76 C9 C3 CD F1 C3 DC F0 C3 38 F0 C3 38  evKvICMqCL'C8'C8
```

E - Unused

F - Fill Memory With [byte]

On the command line:
```
 >F[addr1],[addr2],[byte]
```

Fills from addr1 through addr2 with byte.

```
EXAMPLE:F0,17FF,0
```

G - Goto - Execute A Program, With Optional Breakpoints

On the command line:
```
 >G[addr1],<addr2>,<addr3>
```

Goes to addr1, and optionally sets breakpoints at addr2 and addr3. If continuing from a breakpoint, the first parameter may be omitted, which will execute whatever addr.is contained in the "P" register.

```
EXAMPLE:  >G,1234;continue, with breakpoint at 1234H
          >G1600,163E;goto 1600 with breakpoint at 163EH
```

H - Hexadecimal Math

On the command line:
```
 >H[val1],[val2]
```

Hex math of: val1+val2 and val1-val2 is displayed.
Useful for calculating offsets for Z-80 relative jumps.

```
EXAMPLE:  >H2000,102A
           302A 0FD6;302A is SUM; 0FD6 is difference
```

I - Unused
J - Unused
K - Unused

L - Line Printer Enable

On the command line:
```
 >LE
 >LD
```

Enables/disables line printer echo. Typing 'E' enables the echo, and 'D' disables it. Useful for getting hard copy during Zapple dumps.

M - Move Memory Blocks

On the command line:
```
 >M[addr1],[addr2],[addr3]
```

Moves a block of memory starting at addr1, ending at addr2, to the block starting at addr3.

```
EXAMPLE: >M8000,87FF,100;would copy the 2K from 8000 to 100H
         >M0,7FF,1000
```

CAUTION: You could destroy the data you are attempting to move. When moving *up*, you should move up *beyond* the block you are moving. The best way to move is *down*.

N - Unused
O - Unused

P - Put ASCII From Keyboard To Memory

On the command line:
```
 >P[addr]
```

Puts keyboard input directly into memory starting at addr. Inputting is terminated with a control-D. The address of the byte that would have been loaded next is displayed on the console.

```
EXAMPLE: >P1000
         The quick brown fox jumped over a byte.
         Boy was he surprised.<control-D>
         103D  <-address of NEXT byte to use>
```

Q - Query Z-80 I/O Ports

On the command line:
```
 >QI[port]
 >QO[port],[byte]
```

May be used both to display (QI) and send to (QO) any of the 256 I/O ports. When inputting, the results are displayed in binary: 00001101 with bit zero on the right. When outputting, [port] will be sent [byte].

```
EXAMPLE: >QI70  00000010
         >QO71,7
```

R - Unused

S - Substitutes Memory

On the command line:

```
 S - >S[addr]
```

Substitutes memory, starting at addr. Used to alter a byte in main memory. The current value at the selected address is displayed; followed by the option of changing the current address, or skipping to the next address (by hitting the space-bar). A change is in effect as soon as the space-bar has been hit; a command can be terminated by hitting a return.

Note that the address currently being worked on is printed on every 8-byte boundary. A backspace will cause a re-examination of the previous byte. Used *only* when a space-bar or CR would be used.

```
EXAMPLE: >S1000 54- 68- 65-79 20- 71- 75- 69- 63-
          1008 6B- 20- 62- 72-
```

T - Type Out Memory (in ASCII)

On the command line:

```
 >T[addr1],[addr2],[byte]
```

Types out memory from addr1 through addr2. The number of bytes per line is given in the third parameter. The default is 40H (64) bytes per line.

More useful than the "D" command when just looking for ASCII text in memory.

```
EXAMPLE: >T1000,100F

          1000  Thy quick brown
         >T0,1F

          0000  C.2pC.vC6vC..tCDtC+tC...qC,uC.
```

U - Unused

V - Verify Blocks of Memory

On the command line:

```
>V[addr1],[addr2],[addr3]
```

Verifies the contents of memory from addr1 through addr2 with memory starting at addr3. If a difference is found, the address of the lower block is printed, followed by the byte found at that address, and the byte found at the address that would correspond relative to [addr3].

```
EXAMPLE: >V0,402,F000

          0400 FF ED
          0401 FF 52
          0402 FF 20
```

W - Unused

X - Examine Z-80 Processor Registers

On the command line:

```
>X<'><r>
```

Examines the "main" registers or, optionally, the "prime" set. X or X' followed by a carriage return displays the entire set, where X<r> or X'<r>, followed by a space bar, will examine the contents of a single register, with the option of altering its contents. The technique is similar to the 'S' command.

```
EXAMPLE: >X
      A=18 B=AA C=28 D=A9 E=FA F=44 H=AC L=41 M=00 P=ADC2 S=AC96 I=00
         >X'
      A=01 B=00 C=06 D=00 E=04 F=94 H=04 L=2A M=3E X=0219 Y=01FB R=45
         >XA 18- AA- 28- A9-00 FA- 44- AC- 41- 00- ADC2-F000
         >X
      A=18 B=AA C=28 D=00 E=FA F=44 H=AC L=41 M=00 P=ADC2 S=F000 I=00
```

Y - Search For Byte Strings in Memory

On the command line:

```
 >Y[byte],<byte>,<byte>,<byte>........
```

Searches all of memory for a match on the series of <byte>s. The starting address of each occurrence is displayed on the console. Search string limit is 255 characters.

```
EXAMPLE: >YCD,1E,F0
                     0836
                     0979
                     1703
                     231C
```

Z - Z End

On the command line:

```
 >Z
```

Z alone displays the last R/W memory address under Zapple on the console.

On the QX-10, the result returned will *always* be 0EFFFH.

```
EXAMPLE: >Z
         EFFF
```

(The Z command was more useful in the days when systems had from 1K to 48K of main memory, with all amounts in between. It is now used more as a simple way of ascertaining that Zapple is "alive and well," since hitting 'Z' causes it to respond in a known fashion.)

COMPARISON OF TPM-II and CP/M FUNCTION CALLS

CALL # DEC.	FUNCTION	HEX
0.	Reset System	00
1.	Read Console Character (echo)	01
2.	Write Console Character	02
3.	Read Character from Reader	03
4.	Write Character to Punch	04
5.	Write Character to List Device	05
6.	Get TPM Serial # and Console I/O	06
7.	Read I/O Assignment Byte	07
8.	Modify I/O Assignment Byte	08
9.	Write Buffer to Console	09
10.	Read Console Buffer	0A
11.	Interrogate Console Status	0B
12.	TPM-Return System I.D./ Set Mode-CP/M	0C
13.	Reset Disk System	0D
14.	Log in Disk Drive	0E
15.	Open Disk File	0F
16.	Close Disk File	10
17.	Search for Disk File	11
18.	Search for Next Disk File	12
19.	Erase Disk File	13
20.	Read Disk File Record	14
21.	Write Disk File Record	15
22.	Create New Disk File	16
23.	Rename Disk File	17
24.	Return Login Vector	18
25.	Identify Logged-In Disk Drive	19
26.	Set Disk I / O Address	1A
27.	Get Disk Information	1B

HEX	TPM-II MODE	CALL# DEC.	CP/M MODE	HEX
1C	Read Console Char.(no echo)	28.	Write Protect Disk	1C
1D	Get System Date	29.	Get Read/Only Vector	1D
1E	Get Time	30.	Set/Reset File Attributes	1E
1F	Trap Control	31.	Get Disk Param Block Address	1F
20	Set Date/Time	32.	Set/Get User Value Curr.Unit	20
21	Chain Program	33.	Read Random	21
22	Get TPM Version Number	34.	Write Random	22
23	Do Direct Disk I/O	35.	Compute File Size	23
24	Create File Control Block	36.	Set Random Record	24
25	Return Time in Seconds	37.	Reserved for Future Use	25
26	Set/Reset File Attributes *	38.	Unused	26
27	Graphics Driver Support	39.	Unused	27
28	Multibank&Interrupt Function	40.	Write Random with Zero Fill	28

* Call #38 in TPM mode is identical to a CP/M call # 30

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 0: RESET SYSTEM

```
MVI  C,0     ( 00H )
CALL 5
```

DESCRIPTION OF FUNCTION

Terminates program execution and re-initializes the system. Performs same function as a jump to location 0.

**

FUNCTION 1: READ CONSOLE CHARACTER (Echo)

```
MVI  C,1     ( 01H )
CALL 5
```

DESCRIPTION OF FUNCTION

Reads the input character from the console key-board and returns it in the A register. Each character entered on the keyboard is echoed on the console screen. Characters listed are trapped by TPM and not passed to the program (^C, ^P, ^Q, ^S, ^X). Default tabs are set in every eighth column.

**

FUNCTION 2: WRITE CONSOLE CHARACTER

```
MVI  C,2     ( 02H )
CALL 5
```

DESCRIPTION OF FUNCTION

Writes the character in the E register to the console. Tab characters are expanded to fill the spaces to the next default tab. (Default tabs are set every eight columns. Tab characters are represented by a 09H.) All other characters are written directly, making the program responsible for initiating any special control functions by the terminal in response to keyboard entries.

**

FUNCTION 3: READ CHARACTER FROM READER

```
MVI  C,3     ( 03H )
CALL 5
```

DESCRIPTION OF FUNCTION

Reads a single character from the device currently defined as the logical reader. The character is returned in the A register.

**

FUNCTION 4: WRITE CHARACTER TO PUNCH

```
MVI  C,4     ( 04H )
MVI  E, < character to be written >
CALL 5
```

DESCRIPTION OF FUNCTION

Writes the E register contents to the logical punch device.

**

FUNCTION 5: WRITE CHARACTER TO LIST DEVICE

```
MVI  C,5     ( 05H )
MVI  E, < character to be written >
CALL 5
```

DESCRIPTION OF FUNCTION

Writes the E register character to the list device. No result is returned.

**

FUNCTION 6: GET TPM SERIAL NUMBER
(Direct Console I/O-for CP/M Compatibility)

```
MVI  E,0FEH
MVI  C,6     ( 06H )
CALL 5
```

TO DIRECT CONSOLE I/O:

```
MVI C,6
MVI  E <Input/Output> (See table)
```

DESCRIPTION OF FUNCTION

Returns the address of the 6-byte area containing the TPM serial number. The address in the BC register pair points to the serial number, in an ASCII value. Application programs should not write to this area.

In CP/M mode, this operation performs direct I/O to the logical console. Function 6 requires one of four values in the E register. Care must be exercised, as Function 6 bypasses the control characters ordinarily used to manipulate the console display (^C, ^P, ^Q, ^S, ^X). All results are returned in the A register (see table at left).

E REGISTER ENTRY VALUES

E Reg. HEX	MEANING
0FF	Console Input/Status Command (Returns input character;if no character ready, 00 returns.)
0FE	Console Status Command (Returns 00 if no character ready; otherwise, contains FFH.)
0FD	Console Input Command (Returns an input character; will suspend the calling process until character is ready.)
ASCII Char.	ASCII value contained in E Register is sent to the console.

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 7: READ I/O ASSIGNMENT BYTE

```
MVI  C,7     ( 07H )
CALL 5
```

DESCRIPTION OF FUNCTION

Returns the I/O assignment byte in the A register. The assignment byte consists of four fields of two bits each. Each field describes the current assignment of one of four logical I/O devices. (See the Assignments Table that appears below.)

ZAPPLE is the resident monitor and handles the I/O manipulation of the console, disk drives, printer, and RS232 serial port. Other devices need I/O drivers built if used.

The ZAPPLE MONITOR program is explained in detail in this manual--see Chapter 3 for specific byte assignments and how they are represented.

TABLE OF BYTE ASSIGNMENTS

```
[              MSB                 LSB     ]
[ I/O byte: ] 11] pp] rr] cc]              ]
```

**

FUNCTION 8: MODIFY I/O ASSIGNMENT BYTE

```
MVI  C,8     ( 08H )
MVI  E, < new I/O byte >
CALL 5
```

DESCRIPTION OF FUNCTION

Replaces the I/O assignment byte with the value placed in the E register. Programs should not have to modify the logical I/O assignment, as ZAPPLE handles I/O via PIOS.

**

FUNCTION 9: WRITE BUFFER TO CONSOLE

```
MVI  C,9     ( 09H )
LXI  D, < buffer address >
CALL 5
```

DESCRIPTION OF FUNCTION

Writes the buffer pointed to by the DE register to the console. The buffer is terminated by a 00H (null), or a 24H ($). Neither is displayed. The buffer can also be terminated by setting the high order bit of the last byte displayed. Programs are responsible for console control characters (ie: carriage return and line feed).

**

FUNCTION 10: READ CONSOLE BUFFER

```
MVI  C,10    ( 0AH )
LXI  D, < buffer address >
CALL 5
```

DESCRIPTION OF FUNCTION

Used to read console commands. Characters input at the keyboard are buffered and echoed on the console screen until a carriage return is typed. The control commands available to edit the line are listed at left.

Characters are stored in a buffer which begins at the address pointed to by the DE register. The first byte of the buffer must state the maximum buffer length (up to 255 bytes). On Return, the second byte contains the current buffer length. The first two bytes are not included in the count.
Input characters start in the third buffer position. The carriage return does not appear in the buffer, but if a line-feed (0AH) is entered to perform a carriage return without terminating the input, the line-feed character appears in the buffer and is counted. If the buffer is full before a carriage return, no additional characters will be accepted.

CONTROL KEY		FUNCTION
^STYLE	=	Caps Lock.
TAB	=	8 spaces each Tab.
RETURN	=	Carry out command.
ENTER	=	Same as Return.
<x	=	Backspace / Delete.
^R	=	Retype line.
^U	=	Cancel line / do CR/LF
^X	=	Erase line / no CR/LF

CONSOLE BUFFER

```
Byte:       1.   2.   3.    4.     L
           [ M ] L ] c1 ] c2 ]---] cL ]
DE Reg = Location of 1st byte in buffer
```

M	=	Maximum Buffer Length (00-FF)
L	=	Current Buffer Length (00-FF)
c1	=	Characters (in Hex)
c2	=	Characters (in Hex)
cL	=	Last Character (Current Length)

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 11: INTERROGATE CONSOLE STATUS

```
MVI  C,11    ( 0BH )
CALL 5
```

DESCRIPTION OF FUNCTION

Used to determine if the console input is waiting to be read by a running program. If a character *has* been typed, but not read, the A register contains a non-zero. If *no* characters are waiting in the input buffer, the A register contains 00H.

**

FUNCTION 12: GET SYSTEM IDENTIFICATION
(SET MODE-for CP/M compatibility)

```
LXI  D,0
MVI  C,12    ( 0CH )
CALL 5
```

SET	HEX
TPM Mode Pass:	0AAAA
CP/M Mode Pass:	0CCCC

(Values are passed to the DE register. Other values leave the mode unchanged.)

DESCRIPTION OF FUNCTION

Returns the address of the current TPM version number. The address is returned in the BC register pair and the version number is stored as an ASCII value delimited by a 00H (null). Programs should not write to this area.
To remain compatible with CP/M, Function 12 returns zero in the HL register pair if the system is in the TPM mode. If the system is in CP/M, 0022H is returned.

The DE register pair can be examined from an application program to determine if the system is running in TPM or CP/M mode. If 0054H (ASCII "T") is returned, the system is in *TPM regardless* of the mode returned in the HL register.

Shuttling from TPM to CP/M between extended calls requires that a unique value be passed to the DE register before calling Function 12. This allows a program to take advantage of TPM's extended capability, and CPM features.

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 13: RESET DISK SYSTEM

```
MVI   C,13    ( 0DH )
CALL 5
```

DESCRIPTION OF FUNCTION

Resets the system disk, logs in drive A, and sets the disk I/O the default of 80H. The ^C trap is reset to the default value and any waiting console input is flushed.

**

FUNCTION 14: LOG IN DISK DRIVE

```
MVI   C,14    ( 0EH )
MVI   E, < # of Drive >
CALL 5
```

DESCRIPTION OF FUNCTION

Logs in a particular disk drive by placing the hexadecimal value in the E register. Zero will log in Drive A, "1" logs in Drive B, etc. Note: this is not the same convention for drive numbers as that explained in the FCB (File Control Block) section.

**

FUNCTION 15: OPEN DISK FILE

```
MVI   C,15    ( 0FH )
LXI   D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

All disk files must be opened with either Function 15 or Function 22.

Function 15 opens the file specified in the FCB and prepares the file for Read and Write operations. If the file specified does not exist, then an FFH is returned in the A register.

**
FUNCTION 16: CLOSE DISK FILE

```
MVI  C,16    ( 10H )
LXI  D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Closes a disk file from within a program by specifying the file name in the FCB. Files not written to need not be closed, but all files written to must be closed to update the disk directory space allocation for the file.

If the specified file is not present in the disk directory, FFH is returned in the A register. This error usually indicates that a disk or disk file is write protected, but it can also mean that a disk was changed without resetting the disk map.

**
FUNCTION 17: SEARCH FOR DISK FILE

```
MVI  C,17    ( 11H )
LXI  D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Determines whether or not a specified file exists in the logged-in disk directory. The address of the disk directory entry for the first FCB matching the specifications is returned in the A register. If no match is found, then FFH is returned.

Each FCB occupies 32 bytes. TPM reads a 128-byte directory block (four FCB entries) into the console buffer at the disk I/O address. If a match is found in the block, its number within the block is determined by the low order two bits of the A register.
Function 17 can be used to search for ambiguous file names, but Function 18 must be used to get subsequent entries. Note that a "?" placed in the FCB extent field returns directory entries for all extents.

To Place an Address of a Directory entry into the HL Register:

```
LXI  H, < disk I/O address >
ANI  03 ,mod 4
RRC     ,multiply A by 32
RRC
RRC
MOV  E,A
MVI  D,0
DAD  D
```

**

FUNCTION 18: SEARCH FOR NEXT DISK FILE

```
MVI   C,18    ( 12H )
LXI   D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Returns the byte address of the next directory on the disk that matches the specified file name attributes. No intermediate TPM disk I/O calls are allowed between a Function 17 and 18 call, or between one Function 18 call followed by another Function 18 call.

**

FUNCTION 19: ERASE DISK FILE

```
MVI   C,19    ( 13H )
LXI   D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

All files with the file name, extension, user code, and protection level specified in the FCB will be deleted with this function. Ambiguous file names, extensions, and user codes may be placed in the FCB to be erased. No result is returned, unless no matching file is found. (Protection level rules must be observed.)

**

FUNCTION 20: READ DISK FILE RECORD

```
MVI   C,20    ( 14H )
LXI   D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Reads the next 128-byte record from the specified disk file into the disk I/O address. The file must have been previously opened or created (see Function 26). A byte with the result is returned in the A register.

A Reg. Result		Meaning
00	=	Successful Read
01	=	End of File
02	=	Attempted to Read Unwritten Record

Note: If the file was created using TPM Random I/O Functions, there can be holes in it where no records were written. This can cause file size calculation errors (see Function 35).

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 21: WRITE DISK FILE RECORD

```
MVI  C,21    ( 15H )
LXI  D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Writes the record stored in the 128-byte disk I/O area, to the indicated disk file at the next sequential location. The file must be named in the FCB and must have been opened or created previously. Result is returned in the A register (see left).

A Register Result		Meaning
00	=	Successful Write
01	=	File Exceeds Maximum Size
02	=	No More Space on Disk for File
03	=	Write Protection Violation
FF	=	No More Space in Disk Directory

**

FUNCTION 22: CREATE NEW DISK FILE

```
MVI  C,22    ( 16H )
LXI  D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Used to create a new disk file having the name stored in the FCB at 05CH. The program terminates if the file name is invalid. If the file name is valid, the file is created and left open so that write operations can begin. Any pre-existing file with the same name, extension, user code, and protection level is erased.
If there is not enough room for the file on the disk, an FFH is returned in the A register.

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 23: RENAME DISK FILE

```
MVI  C,23    ( 17H )
LXI  D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Renames a file from within a program. The program must load the old name into the first 16 bytes of the FCB (at 5CH), and the new name into the second 16 bytes (at 6CH).
Four bytes following the first file name are not used. No result is returned.
Files with ambiguous file names can be renamed, but they must be ambiguous in the same positions to avoid conflicts during the renaming process.

An application program can also set a new protection level on the renamed file. The disk drive ID of the first file name is assumed for both, but if the low order bit of the first byte of the second file (which is the disk drive ID) is set to one, the protection code is set to the high order three bits of that byte.

**

FUNCTION 24: RETURN LOGIN VECTOR

```
MVI  C,24    ( 18H )
CALL 5
```

DESCRIPTION OF FUNCTION

Returns the log-in vector in the HL register.
The vector is a 16-bit value, with the least significant bit of the L register corresponding to drive A, and the high order bit of H corresponding to the 16th drive. A zero bit means that the drive is not on line, and a one bit means the drive is active.

(** Indicates Function Calls for both TPM & CP/M)

**

FUNCTION 25: IDENTIFY LOGGED-IN DISK DRIVE

```
MVI  C,25    ( 19H )
CALL 5
```

DESCRIPTION OF FUNCTION

Function 13, RESET DISK SYSTEM, has returned a hexadecimal value in the A register. 01H is drive A, 02H is drive B, etc. Note that the drive number conventions are different than the conventions in the FCB.

**

FUNCTION 26: SET DISK I/O ADDRESS

```
MVI  C,26    ( 1AH )
LXI  D, < New Disk I/O address >
CALL 5
```

DESCRIPTION OF FUNCTION

Allows the disk I/O buffer to be set. Disk reads and writes require the transfer of 128 bytes to or from the disk I/O buffer. The address of this buffer is set to 80H, whether it is a cold start or a warm boot.
If a program requires that several disk files be opened at the same time, a separate 128-byte buffer should be allocated for each file, and a Function 26 call made for every read or write, making a positive buffer selection for every operation.

TPM uses the default I/O buffer for all file opens and closes, except for program chaining (see Function 33 CHAIN PROGRAM).
A new I/O buffer address should be set if a program will need to access the information in it after it has been written out.

**
FUNCTION 27: GET DISK INFORMATION

```
MVI  C,27    ( 1BH )
CALL 5
```

DESCRIPTION OF FUNCTION

Returns a set of addresses which point to the information TPM maintains concerning disk drives. Access to any needed information is available. The addresses returned are in the tables that follow.

DISK PARAMETER BLOCK (DPB) MAP

```
                    ( Usage same as CP/M 2.2 )
{ 00   01]    02]    03]    04] 05   06] 07   08] 09   0A] 0B   0C] 0D   0E]
  SPT        BSH    BLM    EXM      DSM      DRM      ALV        CKS     OFF

                    ( Extended usage only with TPM-II )
] 0F    10]   11     12]    13]            14]     15]     16] 17  18]      19]
#Tracks  Allocation  Bytes/  Error Retry Track 0  S-Flag  Handler   X-Flags
          Map Size   Block    Count      Sector Count     Address

        ] 1A           1B]          1C]  1D...          }
        Physical Media  Block/Deblock  Scramble Table
         Sector Size        Mask         SPT bytes
```

GENERIC DISK INFORMATION DEFINITIONS

OFFSET HEX	WORD OR BYTE	CP/M NAME	CONTENT
0	Word	SPT	Sectors Per Track
2	Byte	BSH	Block Shift Factor
3	Byte	BLM	Block Mask
4	Byte	EXM	Extent Mask
5	Word	DSM	# of Blocks-1
7	Word	DRM	# of Directory Entries-1
9	Word	ALV	Directory Allocation Mask
0B	Word	CKS	Directory Check Size
0D	Word	OFF	Directory Track
0F	Word		Total Tracks this Unit
11	Word		Total Bytes for Allocation Map
13	Byte		K Bytes/Block
14	Byte		Error Retry Count
15	Byte		Sector Count in Track 0
16	Byte		S-Flag (If bit 0=1,Scramble Table is present)
17	Word		Address of Physical Handler
19	Byte		X-Flags
1A	Word		Physical Media Sector Size
1C	Byte		Blocking / Deblocking Mask
1D	N Bytes (N=SPT)		Scramble Table Size (used to map logical to physical sector numbers)

FUNCTION 27: (Continued)

DISK PARAMETER HEADER (DPH) MAP

```
                     ( Extended usage only with TPM-II )
{ 00   01]  02   03]        04]   05]       06]  07 08         26]     27      28]
 Current   Current   On Line   PIOS    Current   Directory Check   Translation
 Track #   Sector    Flag    System   User #    Vector (CP/M)    Table Address

                                  <Usage same as CP/M 2.2>          <CP/M    2.2>
]29    2A] 2B     2C]  2D   2E]  2F   30]31 32] 33 34]    35 36   ]37 38   SPT}
Scratch#1 Scratch#2 Scratch#3  DIRBUF   DPB    CSV    Allocation      ALV
   CP/M      CP/M      CP/M                           Map Address
```

CURRENT INFORMATION

OFFSET HEX	WORD OR BYTE	CP/M NAME	CONTENT
0	Word		Current Track Number
2	Word		Current Sector Number
4	Byte		Online Flag (If bit 0=1, diskette is mounted in drive and ready to use)
5	Byte		PIOS System Flag Byte
6	Byte		Current User Number
7	32 Bytes		Directory Check Vector-for CP/M 2.2
27	Word		Translation Table Address
29	Word		CP/M Scratch #1
2B	Word		CP/M Scratch #2
2D	Word		CP/M Scratch #3
2F	Word	DIRBUF	Directory Buffer Address
31	Word	DPB	Address of DPB (X Index Value)
33	Word	CSV	Address of Check Vector
35	Word		Address of Allocation Map
37	Variable	ALV	Allocation Map Vector One bit per available block on disk For total number of bits assigned, see Generic Table. If bit = 1, then block is in use. Bit zero of first byte represents first block. Bit one of second block...etc. Last byte in table is checksum.

TPM-II

FUNCTION 28: READ CONSOLE CHARACTER (NO ECHO)

```
MVI  C,28   ( 1CH )
CALL 5
```

DESCRIPTION OF FUNCTION

Identical to Function 1, except characters are not echoed on the console screen.

CP/M 2.2

FUNCTION 28: WRITE PROTECT DISK

```
MVI  C,28   ( 1DH )
CALL 5
```

DESCRIPTION OF FUNCTION

Sets the write protect bit on the default drive. Once this bit is set, any attempt to write to the drive will generate the write protection message: 'The Diskette is Write Protected'. The operator can override this message with a ^C. The write protection remains until the next warm boot. If Function 28 is called a second time, it resets the write protection bit and removes the protection.

TPM-II

FUNCTION 29: GET SYSTEM DATE

```
MVI  C,29   ( 1DH )
LXI  D, < 8-byte result area >
CALL 5
```

There are two ways to set the date:
1. The SET-TIME utility.
2. TPM Function call 32.

DESCRIPTION OF FUNCTION

Returns the current date, in ASCII, to the 8-byte area pointed to by the DE register. Formatting is MM/DD/YY (standard calendar).

CP/M 2.2

FUNCTION 29: GET READ/ONLY VECTOR

```
MVI  C,29   ( 1DH )
CALL 5
```

DESCRIPTION OF FUNCTION

Identifies protected files and reads from the disk drive in which the write protect bit has been set. The drive is returned in the HL register. It will also read files set to TPM protection levels 6 and 7.

TPM-II

FUNCTION 30: GET TIME

```
MVI  C,30    ( 1EH )
LXI  D, < Address 8 byte result area >
CALL 5
```

DESCRIPTION OF FUNCTION

Returns the current time, in ASCII, to the 8-byte area pointed to by the DE register. Formatting is a 24-hour clock (military time) HH/MM/SS .

There are two ways to set the time:
1. The SET-TIME utility.
2. TPM Function call 32.

CP/M 2.2

FUNCTION 30: SET/RESET FILE ATTRIBUTES

```
MVI  C,30    ( 1EH )
LXI  D, < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Searches the directory of the selected drive for all matches to the file name stored at the FCB address. When a match is found, bytes 01H through 0BH are replaced by those in the FCB. This different bit pattern is updated in the disk file directory to the new attribute status.

TPM-II

FUNCTION 31: TRAP CONTROL

```
MVI  C,31    ( 1FH )
LXI  D, < Program trap address >
CALL 5
```

DESCRIPTION OF FUNCTION

TPM provides ^C to abort the program that periodically interrogates console status, or incorporates console read/write functions. ^C also resets the system.
By using Function 31, instead of ^C, a program can trap to a predefined error handling routine. TPM jumps to the address given in the DE register. Each warm boot resets the ^C trap address to the TPM system initialization routine.

Function 31 does not affect the ^\ (degree key) trap to ZAPPLE. ZAPPLE is provided by the PIOS not the LIOS.
To avoid a software conflict, the ^\ trap to ZAPPLE can be disabled or changed.

CP/M 2.2

FUNCTION 31: GET PARAMETER BLOCK ADDRESS (DPB)

```
MVI  C,31    ( 1FH )
CALL 5
```

DESCRIPTION OF FUNCTION

Returns the address of the active DPB in the HL register. The calling program can use the address to determine the disk parameter values for display or to compute the space on a drive. Errors will result in a OFFFFH in the HL register.

TPM-II

FUNCTION 32: SET SYSTEM TIME AND DATE

```
MVI   C,32    ( 20H )
LXI   D, <Address of 4-byte input area>
CALL  5
```

DESCRIPTION OF FUNCTION

Generally, the time and date are set with the SET-TIME utility.
Function 32 permits the time and date to be set or reset under the control of a program.

Each call changes either the date or time, depending upon the first hexadecimal byte of the input area: 0=Date, 1=Time.
The remaining three bytes give the three fields of the date or time in binary coded decimal (BCD).
Example: To set the date 01/16/47, the four input bytes would be: 0,1,16,47H.

CP/M 2.2

FUNCTION 32: GET/SET CURRENT USER CODE

```
MVI   C,32    ( 20H )
MVI   E < FFH > or < User number >
CALL  5
```

DESCRIPTION OF FUNCTION

Determines the current user number by entering FFH in the E register. The result is returned in the A register. The user number can be changed as well, by placing the new number in the E register.
There are 256 possible numerical user codes, 0 through 255.

TPM-II

FUNCTION 33: CHAIN PROGRAM

```
MVI   C,33    ( 21H )
MVI   B, < 0=Return when done, or
           1=Load and execute >

LXI   D, < Address of program FCB >
LXI   H, < Load address >
CALL 5
```

DESCRIPTION OF FUNCTION

Loads a second program and conditionally executes it. The DE register must point to an FCB that gives the disk file name of the program. When the new program is loaded, control is returned to the caller (B=0), or the program will be executed at the load address. If TPM encounters an error, it returns control to the caller, regardless of the value in the B register. The < load address >+< file length > must always be below the LIOS, (the value at address 6, D800H), or a MEMORY OVFL error will occur.

The caller can be forced to pass data to the called program. Place the data to be passed above the highest address used by either program (the default buffer could be used at address 80H).
If the Load fails because the called program is either not on the disk, or is too large, TPM terminates both programs.

If a read of a random access file with "holes" in it is attempted, a one is returned in the A register.

CP/M 2.2

FUNCTION 33: READ RANDOM RECORD

```
MVI   C,33    ( 21H )
LXI   D, < FCB address >
CALL  5
```

Return Results in the A Register:

HEX	MEANING
00=	Successful Read Completed
01=	Attempted to Read Unallocated File
03=	Cannot Close/Update Current FCB
04=	Attempted to Read Unallocated Extent
06=	Direct Address Larger than Allowed

DESCRIPTION OF FUNCTION

Reads directly from an open file. Specify the record in bytes 21H, 22H, and 23H of the FCB. LIOS will determine the required extent number, calculate the correct allocation block within the extent, and look up the entry in the extent data map. Once the record is found on the disk, it is read into the current file buffer. A 00 is returned in the A register, indicating a successful read.

If the current extent is not the one required, LIOS updates the directory entry for the current extent (if it has been changed) and copies the correct extent number into the FCB.

Random access permits alternate reads and writes to a file, but problems can develop if these random access files have "holes." If the block being read has ever been part of another file, it could be filled with garbage.

TPM-II
FUNCTION 34: GET TPM VERSION NUMBER

```
MVI  C,34    ( 22H )
CALL 5
```

DESCRIPTION OF FUNCTION

Same as Function 12. The TPM version number is returned as an ASCII value delimited by a null (00H) in the BC register pair.

CP/M 2.2
FUNCTION 34: WRITE RANDOM RECORD

```
MVI  C,34    ( 22H )
LXI  D < FCB address >
CALL 5
```

Return Results in the A Register:

HEX	MEANING
05 =	Directory Overflow, New Extent not Created

DESCRIPTION OF FUNCTION

Writes directly to an open file. Specify the record in bytes 21H, 22H, and 23H of the FCB. The LIOS then determines the extent number, calculates the allocation block within the extent, and writes the record.
If the record exists, then the LIOS will overwrite it with the new data. If no block has been allocated in the extent, LIOS will assign one. And if the current extent is not the one required, LIOS updates the directory entry for the current extent with any modifications. LIOS then copies the correct extent number to the FCB. The result code is returned in the A register. Codes have the same meaning as in Function 33 CP/M, except for the 05H addition:

Random access files can easily be created with "holes," which contain no useful data. If the block has ever been written to, the directory program will view it as full.

TPM-II

FUNCTION 35: DO DIRECT DISK I/O

```
MVI   C,35     ( 23H )
MVI   E, < Function Code >
MVI   L, < Parameter >-or-
           LXI  H,< Parameter >
CALL  5
```

The table of codes defines the operations to be performed.

Parameters Specify Register Usage:

DIRECT DISK I/O FUNCTIONS

FUNC. CODE	OPERATION PERFORMED	PARAMETERS
0 =	Home Head	NONE
1 =	Select Drive	L= Drive # (A=0, B=1,etc..)
2 =	Set Track #	HL= Track # (zero is first)
3 =	Set Sector #	L= Sector # (zero is first)
4 =	Set I/O Address (DMA)	HL= New Address
5 =	Read Disk Record	L= Retry Count
6 =	Write Disk Record	L= Retry Count

DESCRIPTION OF FUNCTION

Performs disk I/O at any track and sector. The operation bypasses the standard TPM disk file interface, so care *must* be exercised.

Set the address where the I/O is to take place, select the desired drive, track, and sector, and initiate the read or write. (Function 27 will determine the characteristics of the drive in use, if needed.)

If the read or write fails, TPM will repeat the operation for the number of times specified in the L register. Error codes are returned only when the retry count has been exhausted. These codes are returned in the A register, and if a zero is returned, the operation was successful.

TPM DIRECT DISK I/O ERROR CODES

ERROR CODE HEX	MEANING
00 =	blank unformatted disk
04 =	lost data (speed wrong)
08 =	CRC error (data) encountered
10 =	requested record (header) not found
18 =	CRC error (header) encountered
C0 =	disk physically write-protected
FF =	drive empty

CP/M 2.2

FUNCTION 35: COMPUTE FILE SIZE

```
MVI   C,35     ( 23H )
LXI   D < FCB address >
CALL  5
```

DESCRIPTION OF FUNCTION

Computes file size of file pointed to by the DE register, so data may be added sequentially to the end. If the file being computed was written directly, only the identity of the highest numbered extent in the directory FCB is returned.

TPM-II

FUNCTION 36: PARSE FILE CONTROL BLOCK

```
MVI  C,36   ( 24H )
LXI  H, <Address of file name string>
LXI  D,<Address of 12 byte result area>
CALL 5
```

DESCRIPTION OF FUNCTION

Parses disk file names for programs from the string pointed to by the HL register. If a valid file name is found, it is placed in the 12 byte area pointed to by the DE register, in the format of the first 12 bytes of an FCB. The file name string is terminated by the first invalid character used for the TPM file name.

The BC register returns the address of the byte that follows the last byte of a valid file name.
The A register returns a byte with the coded result.

When ambiguous file names are used (with an *, or ?), TPM will automatically produce the proper FCB format, by expanding with question marks.
If the string terminator is a colon (:), for drive ID, the file name and file type are left blank.

INVALID CHARACTERS for TPM FILE NAMES

([ ] = : ; < > ,)

RESULT CODES for FILE NAME PARSING

BIT	STATUS	MEANING
0	True	Error, No File Name Found
1	True	Only Disk Drive ID Found
2	True	No File Type Supplied
6	True	File Name Found :*.* (all files)
7	True	File Name Ambiguous

(Bits 3,4,5, not used at this time)

CP/M 2.2

FUNCTION 36: SET RANDOM RECORD

```
MVI  C,36
LXI  D < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Calculates the direct address of each record read from a sequential file. The record numbers and compiled addresses are like an index.
Function 33 can read from any individual record in the sequential file. If the records contain a key field, the table converts an ordinary sequential file to an indexed sequential file.

NOTE: If the records are of varying size, this operation will still record the buffer byte position with the record number.

TPM-II
FUNCTION 37: GET SYSTEM TIME

DESCRIPTION OF FUNCTION

```
MVI  C,37    ( 25H )
CALL 5
```

Returns the current time in seconds to the HL register (least significant digits) and the BC register (most significant digits). When the BC and HL registers are concatenated, they create a 4-byte hexadecimal number that expresses the total number of seconds since midnight, up to a maximum of 86,400 seconds: 24:00:00 hours.

CP/M 2.2
FUNCTION 37:
(RESERVED FOR FUTURE EXPANSION)

DESCRIPTION OF FUNCTION

TPM-II
FUNCTION 38: SET RESET FILE ATTRIBUTES

DESCRIPTION OF FUNCTION

```
MVI  C,38    ( 26H )
LXI  D, < FCB address >
CALL 5
```

Same as a FUNCTION 30 in CP/M mode. The directory of the selected drive is searched for all matches to the file name stored at the FCB address. When a match is located, bytes 01H through 0BH are replaced by those in the FCB. The different bit pattern is updated in the disk file directory to the new attribute status.

CP/M 2.2
FUNCTION 38:
(NOT USED)

DESCRIPTION OF FUNCTION

TPM-II
FUNCTION 39: GRAPHICS DRIVER SUPPORT

```
MVI  C,39    ( 27H )
MVI  B, < D-OPCODE >
LXI  H, < Argument 1 >
LXI  D, < Argument 2 >
CALL 5
```

DESCRIPTION OF FUNCTION

The Graphics linkage function. If the graphic drivers are not loaded, any Function 39 calls will return an error code of OFFH in the A register. To ensure that the driver is loaded (in an application which needs to use the drivers, this should be done early in the program) call the Function with B=0. If, on return, A=0, the drivers are available for use.
(See Video Driver Program Document on the actual Drawing Operation Code Commands.)

CP/M 2.2
FUNCTION 39:
(NOT USED)

DESCRIPTION OF FUNCTION

TPM-II
FUNCTION 40: MULTIBANK & INTERRUPT FUNCTIONS

```
MVI  C,40    ( 28H )
LXI  H, < Argument 1 >
LXI  D, < Argument 2 >
CALL 5
```

DESCRIPTION OF FUNCTION

The linkage to the Multi-Bank Functions such as spooler, clock display control, serial port buffered reads, etc.
If the support routines are not available, any Function 40 call will return a OFFH in the A register.
(See SYSINIT for the functions available for CALL 40.)

CP/M 2.2
FUNCTION 40: WRITE RANDOM WITH ZERO FILL

```
MVI  C,40    ( 28H )
LXI  DE < FCB address >
CALL 5
```

DESCRIPTION OF FUNCTION

Similar to Function 34, except that the block is filled with zeros before the record is written. If this function has been used to create a file, records accessed by a Read Random operation containing all zeros will identify unwritten random record numbers.

PIOS JUMP TABLE VECTORS

The following table of jump vectors can be used to bypass some LIOS I/O processing. These vectors are compatible with CP/M 1.4 and 2.2 as well as all versions of TPM. To determine the beginning of the table, pick up the 16-bit word value at location 1 in memory, then modify the low 8-bit value to the offset into the table you desire.

For example, if you wish to get a character from the console, the following routine would bypass the LIOS processing:

```
CHARIN: PUSH    H
        LHLD    1
        MVI     L,9       ;OFFSET
        XTHL              ;RESTORE HL, PLACE TO GO IN STACK
        RET
```

The above routine could be called whenever a character was needed from the console, and the character would be returned in the "A" register.

The register conventions are:

Pass an 8-bit value TO Pios:	place value in "C"
Pass a 16-bit value TO Pios:	place value in "BC"
Get an 8-bit value FROM Pios:	value returned in "A"
Get a 16-bit value FROM Pios:	value returned in "HL"

There *are* some exceptions to these conventions. The "SELDK" vector also returns the disk definition tables in "IX" and "IY". The "TINMS" vector returns the time, in "Seconds past midnight", in two register pairs, "HL-BC".
(See TPM-II manual for further details.)

```
PIOS
Offset     LABEL                                      Description

0000       %BOOT:      JMP       $BOOT               ;COLD START
0003       %WBOOT:     JMP       $WBOOT              ;WARM START
0006       %CONST:     JMP       $CONST              ;CONSOLE STATUS
0009       %CONIN:     JMP       $CONIN              ;CONSOLE CHARACTER IN
000C       %CONOT:     JMP       $CONOT              ;CONSOLE CHARACTER OUT
000F       %LIST:      JMP       $LIST               ;LIST CHARACTER OUT
0012       %PUNCH:     JMP       $PUNCH              ;PUNCH CHARACTER OUT
0015       %READR:     JMP       $READR              ;READER CHARACTER IN
        ;
        ; DISK-ORIENTED VECTORS
        ;
0018       %HOME:      JMP       $HOME    ;HOME THE HEADS
001B       %SELDK:     JMP       $SELDK   ;SELECT A DISK
001E       %STTRK:     JMP       $STTRK   ;SELECT A TRACK
```

```
PIOS
Offset       LABEL                                   Description

0021         %STSEC:    JMP     $STSEC   ;SELECT A SECTOR
0024         %STDMA:    JMP     $STDMA   ;SET THE DMA ADDRESS
0027         %READ:     JMP     $READ    ;DO THE READ
002A         %WRITE:    JMP     $WRITE   ;DO THE WRITE
       ;
       ; CP/M 2.X ADDITIONS
       ;
002D         %LISTS:    JMP     $LISTS   ;LIST DEVICE STATUS
0030         %SECTR:    JMP     $SECTR   ;SECTOR XLATE FOR CP/M GUYS
       ;
       ; TPM-II ADDITIONS
       ;
0033         %IIOS:     JMP     $IIOS          ;GET IOBYTE
0036         %AIOS:     JMP     $AIOS          ;SET IOBYTE
0039         %DATE:     JMP     $DATE          ;DATE ROUTINES
003C         %TIME:     JMP     $TIME          ;TIME ROUTINES
003F         %TINMS:    JMP     $TINMS         ;GET TIME IN SECONDS
0042         %INITD:    JMP     $INITD         ;INITIALIZE SYSTEM FLAGS
0045         %SETUP:    JMP     $SETUP         ;SET X & Y INDEX VECTORS
0048         %CALCK:    JMP     $CALCK         ;LIOS INTERCEPT
       ;
004B         %ERROR:    JMP     $ERROR         ;ERROR PRINTING ROUTINE
       ;
004E         %TRAP:     JMP     $TRAP          ;ERROR TRAP HANDLER
       ;
```