

SYSTEM PROGRAMMER'S GUIDE

for the

TRS-80[®] Model 4/4P

using

Montezuma Micro CP/M[®] 2.2
Version 2.2x



**MONTEZUMA
MICRO**

TABLE OF CONTENTS

1. INTRODUCTION	1
2. THE SYSTEM PARAMETER BLOCK	3
3. I/O USING THE IOBYTE.....	5
4. THE KEYBOARD DRIVER	9
5. THE VIDEO DISPLAY DRIVER	11
6. THE PARALLEL PRINTER DRIVER	13
7. THE SERIAL PORT DRIVER	15
8. THE MEMORY DISK DRIVE	17
9. DISK I/O.....	19
9.1. The Floppy Disk Driver.....	19
9.2. DPB Extensions	19
9.3. DCB Definitions	21
9.4. Using the Disk Driver	22
9.5. EXBIOS - EXtending the BIOS	23
9.6. The Hard Disk Driver	23
10. CP/M BOOTS.....	25
10.1. The Cold Boot	25
10.2. The Warm Boot	25
11. PITFALLS AND TRAPS	27
11.1. INTERRUPTS	27
11.2. FEATURES UNIQUE TO THE Z80	27
11.3. RAM USAGE	28
12. INDEX	29
13. THE LISTING	31

1. INTRODUCTION

All of the information in this manual is copyrighted by Montezuma Micro, including the source code listings. The purpose of this manual is to provide you with the information necessary to modify the BIOS of your own copy of CP/M. Much like the service manual for a car, this document will show you how the BIOS works, and how you can use your own options with it. It will NOT teach you how to use CP/M and it will NOT teach you how to write assembly language programs for use with CP/M. If you bought it for that purpose you will be frustrated and confused. Let us state from the outset that this manual is for experienced programmers! We absolutely cannot and will not provide any telephone or written support for any modifications made to CP/M. In short, you are ON YOUR OWN and if the information you need can't be found in this manual it simply is not available from us.

Furthermore this manual is applicable to Montezuma Micro CP/M BIOS version 2.2x, where x is the patch level. When future versions of our CP/M are released the listings and possibly some of the information will no longer be valid. We make no promises of any kind as to the availability of a similar manual for future versions, and offer no "upgrades" of any kind on this document.

The BIOS listing which accompanies this manual was created using the **2500 A.D. Z80 Macro Assembler**. This assembler uses Z80 mnemonics, unlike the ASM assembler provided with CP/M which uses only 8080 codes. Without apology the author admits to a strong bias for the Zilog Z80 mnemonics, and a strong distaste for the Intel 8080 mnemonics. Only Zilog mnemonics will be used in this manual. The 2500 A.D. assembler is available from Montezuma Micro, and is highly recommended for Z80 programming.

Well, now that we have the preliminaries out of the way let's proceed!

2. THE SYSTEM PARAMETER BLOCK

MOVCPM is a very handy utility which makes it possible to change the size of CP/M so as to reserve space at the top of memory. Unfortunately this creates a major headache for the programmer who wants to write utilities to run under CP/M, since it is not possible to know exactly where in memory each individual copy of CP/M resides. Furthermore some parts of the BIOS may be relocated in the event of an update, necessitating the update of all related utility programs.

To solve these problems we have collected all of the "need to know" information into a section of memory called the System Parameter Block (SPB). The relative location of this block within the BIOS is guaranteed not to change from one version of the BIOS to the next. Further, any additions to it will be made to the end so that relative offsets within the SPB will be good in future versions.

Location of the SPB within CP/M is very simple. It is always 48 bytes (0030H) past the Warm Boot vector in the BIOS. Since the address of the Warm Boot vector always follows the JMP instruction at memory location 0 the SPB can be found using this simple routine:

```
LD    HL,(0001H)    ;Get Warm Boot vector address
LD    BC,0030H      ;Set up offset to SPB
ADD   HL,BC         ;HL now points to SPB
```

The remainder of this chapter will deal with the various fields of the SPB. All offsets are given in decimal. Conversion to hex is left as an exercise to the reader.

Offset 0

The first field of the SPB is a single byte which contains the standard system IOBYTE value set by the CONFIG utility. At each warm boot the contents of this byte are copied to location 3.

Offset 1

Acting as a flag, the contents of this byte tell the BIOS whether to display the CP/M banner after booting. Any non-zero value will cause the banner to be displayed, while zero suppresses it.

Offset 2

In this byte is stored the total number of disk drives, as set by CONFIG. It is never used by the BIOS, but may be of use to external utilities.

Offset 3

The current version of the CP/M BIOS is stored here as two BCD digits packed in a single byte. The first digit is the release number, which changes only upon a complete rewrite. The second is the revision level, which changes upon reassembly of the BIOS. In a fit of optimism the BIOS programmer set this byte to 20H, meaning 2.00. By the time release 2 was ready the revision level had crept up to 2, but the byte was left at 20H. Thus a value of 20H in this byte should be treated as being synonymous with 22H.

Offset 4

Access to disk drives in the BIOS is done using a data structure known as the Disk Parameter Header. It is discussed fully in the manual provided with your CP/M. To allow for the maximum 16 drives possible within CP/M (A: through P:) we have built a 32 byte table of DPH addresses within the BIOS. Whenever a drive is selected via BIOS call XX1BH its corresponding address in this table is

returned. Unused entries are set to 0000H. The two byte address contained in this offset of the SPB is the actual base address of the DPH table, i.e. the address of the address of the DPH for A:. By using relative offsets to this address utility programs may add logical drives to the DPH table. Simply store the address of the DPH of your logical drive at the corresponding entry in the table. Use extreme caution with drive M:, however. The BIOS disk read/write routines test for drive M: and transfer control to special driver coding for that case. Storing the DPH for any other drive in the M: slot will cause problems.

At boot time the DPH table is filled with zeroes and the slots for A:, B:, C:, and D: are filled with the addresses of the four DPHs resident within the BIOS. If the system has 128k of RAM the DPH for drive M: is also added to the table. When the system is booted from a hard disk a patch in the disk boot causes this table to be overwritten with the configuration specified when the hard disk driver was installed.

Offsets 6, 8, 10, and 12

These four offsets contain the two-byte addresses of the disk Device Control Blocks (DCB) for each of four possible floppy disk drives 0 through 3 respectively. The disk DCB, which will be discussed in depth in the DISK I/O section, is used to access a particular floppy drive and contains all the physical characteristics of that drive.

Offset 14

At this offset in the SPB is a two-byte address which points to the base of a table of device driver addresses. This table is used by the BIOS for all I/O except for disk and has been designed to simplify the installation of custom drivers. See *I/O USING THE IOBYTE* for full details.

Offset 16

The base address of the Keyboard Device Control Block (DCB) is found at this offset. See *THE KEYBOARD DRIVER* for information regarding the Keyboard DCB.

Offset 18

Here is the base address of the Video Display DCB. This data structure is fully explained in *THE VIDEO DISPLAY DRIVER*. One item of interest in this DCB is the current cursor location.

Offset 20

The base address of the Parallel Printer Port DCB is stored here. For a full explanation of the DCB see *THE PARALLEL PRINTER DRIVER*.

Offset 22

This offset contains the base address of the Serial Port DCB. See *THE SERIAL PORT DRIVER* for full details.

This is the end of the SPB, at least for now. Any extensions made in future versions of our CP/M BIOS will be made starting at offset 24, thereby retaining compatibility with programs written for earlier versions. You are encouraged to use this structure whenever you must "peek", "poke", or otherwise fiddle with the BIOS. Doing so will make your life easier and could help to remove unsightly warts!

3. I/O USING THE IOBYTE

One of the optional features of CP/M 2.2 that we have implemented is the IOBYTE. The IOBYTE is a set of four two-bit (literally!) fields that can be manipulated to change the assignment of real devices (Keyboard, Serial Port, etc.) to logical devices (CONsole, Line PrinTer, etc.). By convention it resides in memory at location 3 (0003H for hard-core hexaphiles) and is structured like this:

Logical Device --->		LST:	PUN:	RDR:	CON:
Bits Used --->		7, 6	5, 4	3, 2	1, 0
Decimal	Binary	Physical Device			
0	00	TTY:	TTY:	TTY:	TTY:
1	01	CRT:	PTP:	PTR:	CRT:
2	10	LPT:	UP1:	UR1:	BAT:
3	11	UL1:	UP2:	UR2:	UC1:

The logical devices referenced above are as follows:

LST: The LiST device, typically used in CP/M for hardcopy, is output only. Usually it is assigned to a printer.

PUN: The PUNch device reveals the origins of CP/M, back when a paper tape reader/punch was the norm for microcomputers. While such hardware is now relegated mainly to museums and junk yards, we still have this output (only) device to use as we see fit.

RDR: Like PUN:, the ReaDeR device is a throwback to those golden days of 1k RAM boards and Tiny BASIC. Use it as you wish for input (only) operations.

CON: Perhaps the most important device from an operational standpoint is the CONsole. It is the device from which CP/M gets its commands and to which it sends its output. You must be very careful in making assignments to this device, since mistakes can cause you to be unable to communicate with CP/M at all!

Now let's look at the case of physical devices, as defined within our CP/M:

TTY: Another relic from the Neanderthal age of computing is the TeleTYpe. This was a large, noisy machine consisting of a keyboard and a printer. It was able to send and receive data at the blazing speed of 10 characters per second. We have assigned this device to the Serial Port of the Model 4, since the original TTY was serial and you could actually connect one to this device if you really wanted to.

CRT: CRT stands for Cathode Ray Tube and as come to stand for just about any kind of terminal using video output. We have defined the CRT to be the Keyboard of the Model 4 on input and the Video Display on output.

PTP: Since there is no Paper Tape Punch on the Model 4 we have assigned this device to the Video Display. You could, of course attach a true paper tape punch to that port, but if you think about this sort of thing often you really ought to get professional help.

PTR:The Paper Tape Reader falls in the same category as the PTP. Since there (thankfully) is no such device on the Model 4 we use the Keyboard.

LPT:This device is the Line PrinTer, which could be either a serial or a parallel device. Since we already have TTY: for serial output LPT: is assigned to the parallel port. It is, by nature, output only.

UP1:UP2: UP1 and UP2 are user-defined punch devices. The BIOS maps both of these devices to a null driver which does nothing, but is never busy and will not hang up the system if used. Both of these devices are available for user-written drivers.

UR1:UR2: Like UP1 and 2 the UR1 and UR2 drivers are user-defined, but these are reader devices. The BIOS, as shipped, has both devices assigned to a null driver which provides only the end-of-file character Z as input, but will not hang up the system. Both are available for use with user-written drivers.

UL1: The UL1 device is a user-defined line printer. It is, by nature, output only and is assignable only to the LST: logical device. Interfacing of special output devices, e.g. a plotter, can be done using this device. As shipped this device is assigned to a null driver.

UC1: UC1 is the user-defined console device. If implemented it must provide both input and output capability. Unlike other user-defined devices, this one is preassigned to the Keyboard driver for input and the Video Display driver for output. This was done so that there would be no nasty loss of control should the device be accidentally assigned.

BAT:In olden times when computers were slow and I/O devices were even slower it was common to set up a "batch" of instructions for the machine, start it up, and leave for a two-week vacation. We have implemented the BATch device faithful to its original use. When input is taken from BAT: it actually comes from whatever is assigned to the RDR: device. Output to BAT: actually goes to the LST: device. As you can see BAT: is not a physical device, but merely a switch to change the assignment of CON:.

Now that we have seen just what devices are available, how do we go about actually installing a driver for one? Obviously you must first write the driver. What it does is up to you, but there are some conventions that must be followed if the driver is to work properly in CP/M. At minimum it must provide the following four functions:

Input Status

No parameters are required. If the device has input available it should return 0FFH in the A register, otherwise return 0.

Input Data

A byte of input is read from the device and returned in the A register. If no input is available this routine must wait until it is.

Output Data

The C register contains the byte of data to be output to the device. No value is returned to the caller after output.

Output Busy

The output device is tested for a busy condition. If it is busy a value of 0 is returned in the A register. Ready is indicated by a return of 0FFH in the A register.

Of course your device may not actually need all of the above routines, especially if it is input-only or output-only. Unused routines should be replaced with null driver code, so as to prevent problems should a calling routine do something stupid. A sample null device driver can be found in the BIOS listing.

Installation of a driver is not difficult. First you need to find a place in RAM to hold the driver. If it replaces an existing CP/M driver you may want to load it over the existing driver. Be sure that you don't overwrite other code if you do this! The safest method is to locate the driver either above or below CP/M. To load it above CP/M use MOVCPM to create a smaller system. This is preferable to loading it below CP/M, since that method makes the driver vulnerable to being destroyed by programs which use all of available memory. Once you have located the driver in RAM it is necessary only to install the addresses of the above four routines in the proper place within the Device Driver Address Table in the BIOS. The base address of this table can be obtained from offset 14 in the System Parameter Block (SPB) discussed previously. Each entry in this table consists of four two-byte addresses arranged in the following order:

Address of Input Status routine
Address of Input Data routine
Address of Output Data routine
Address of Output Busy routine

Each device holds a particular entry in the table. The devices and their respective offsets are as follows:

Device	Offset	Device	Offset	Device	Offset
TTY:	0	CRT:	8	UC1:	16
LPT:	24	UL1:	32	PTR:	40
UR1:	48	UR2:	56	PTP:	64
UP1:	72	UP2:	80		

These offsets will be held constant in future versions of CP/M. It is unlikely that any new devices will be added, but should that happen the new devices would be added after all existing ones.

4. THE KEYBOARD DRIVER

The console device CON: is CP/M's main source of input other than disk. Most of the time the Model 4 keyboard serves as the primary input device for CON:.

At first glance the keyboard looks like a fairly simple device to interface with. Just scan the rows and see which key is being pressed, then return the ASCII value for that key. Unfortunately the problem is not that simple since you have to contend with SHIFT, CTRL, rollover, repeating keys, function keys, debounce, and a whole lot of other annoying stuff. Our methods for dealing with this chaos may be found in the keyboard driver portion of the BIOS listing. The purpose of this section of the manual is not to explain how the driver works, but rather to discuss the Keyboard Device Control Block (DCB).

The various fields of the Keyboard DCB will be discussed using the field name as shown on the BIOS listing, as well as the decimal offset that the field has from the base of the DCB.

KBDBUF - Offset 0

When the Keyboard driver is called to check for pending input it must scan the keyboard to see if a key is depressed. Since the status routine does not return the actual key value, but only a flag, it is possible that the key will no longer be depressed when the input routine is called to read it. To prevent that from happening any key that is found will be stored in this buffer. On future calls to the status routine if anything is in the buffer a 0FFH will be returned without doing another keyboard scan. Likewise the input routine always checks the buffer before scanning the keyboard for input. It is important to note that this buffer is for only one character and is not in any sense a type-ahead buffer.

KBDFKP - Offset 1

The function keys can be set up to deliver from 0 to 8 characters when pushed. Since only one character is transmitted on each call to the driver a way is needed for the driver to "remember" what the rest of the characters are. This is done using this two-byte pointer, which contains the address of the next character to be used as input.

On every call to the keyboard input routine KBDFKP is checked for a non-zero value. If it is found to be non-zero then a character is loaded from the address in KBDFKP. As long as the loaded character is not zero it is returned to the calling program just as though it had been typed. When a zero value is read KBDFKP is set to zero and the driver resumes keyboard scanning.

Although the function keys are located within the Keyboard Driver in the BIOS, it is possible to set KBDFKP to point to strings of "key input" in other parts of memory. This is exactly the technique used by *Monte's Window* to return Calculator results as keyboard input. There is no limit to the length of the pseudo keyboard input, but the last byte must be 0.

KBDHST - Offset 3

This 8 byte field is used to hold the results of previous scans of each of the 8 keyboard row lines. The information is used to lock out keys that were depressed on previous calls to the driver. Since these fields are necessary for the correct implementation of rollover and repeat it is strongly recommended that you don't mess with them!

KBDPKR - Offset 11

KBDPKI - Offset 12

KBDDLY - Offset 14

KBDRPT - Offset 16

These fields are all used in controlling the automatic repeat of a key held down more than a few seconds. Tweaking and other perverse manipulation is likely to have bad results!

KBDCLF - Offset 18

The Caps Lock Flag is used by the driver to remember whether the keyboard is locked into all upper case or not. Only bit 0 of this byte is actually used but care should be used to keep the other 7 bits set to 0. A value of 0 in this byte indicates that the keyboard is operating in normal upper/lower case, while a value of 1 says that it is locked into all-caps. This flag may be changed so long as the only values used are 0 and 1.

KBDCOD - Offset 19

All of the alpha keys are decoded "on the fly" using the scan information to generate ASCII codes immediately. This technique is not easily applied to the numeric and special function keys, so those are decoded using a table. The Keyboard Decode Table is divided into three entries of 24 bytes each. Each entry defines the following keys in this order:

0	1	2	3	4	5	6	7
8	9	:	;	,	-	.	/
ENTER	CLEAR	BREAK	UP	DOWN	LEFT	RIGHT	SPACE

The first entry defines the above keys used alone, i.e. without either SHIFT key or the CTRL key. Definitions for SHIFT in combination with these keys are in the second entry, and the last entry is for CTRL definitions. Obviously you can change these key definitions to virtually anything you want. We do offer a few recommendations. The ENTER key should produce a carriage return whether SHIFTed, CTRLed, or used alone. That's why our utilities don't allow it to be changed. It would probably also be a good idea for the normal and SHIFTed keys to produce the character inscribed on the keytop. Beyond that, have fun!

KBFKD - Offset 91

The Function Key Definition table contains the strings to be issued when any of the nine possible function key combinations is selected. This table has nine entries of nine bytes each. The first three correspond to unshifted F1, F2, and F3, while the next three are used for SHIFT/ F1, SHIFT/F2, and SHIFT/F3. CTRL/F1, CTRL/F2, and CTRL/F3 are defined in the last three entries. Each definition string may be from 0 to 8 characters long, but the last byte in the string MUST be 0. That's the signal for the Keyboard driver to stop.

This concludes the Keyboard DCB. Since this data structure is used only by the BIOS Keyboard driver it need not be present for Keyboard drivers that you may write yourself. However the last two fields (KBDCOD and KBFKD) must be present at those offsets if the CONFIG utility is to be used with your driver. Field KBDFKP must be present for Monte's Window to work, and that product may not work at all if your Keyboard driver is not physically located in the BIOS memory space.

5. THE VIDEO DISPLAY DRIVER

The Video Display driver uses the Model 4 memory-mapped Video Display to emulate a Lear-Siegler ADM-3A terminal. Most CP/M compatible software expects the CON: device to be an ASCII terminal and the ADM-3A is one of the more common terminals in use, so this arrangement works well. At all times except when it is actually being updated the Video Display RAM is kept switched out of the memory map. This allows BIOS code to occupy the same space and reduces the overall memory overhead on the system.

Operation of the Video Display Driver is uncomplicated and should be easy to follow in the BIOS listing. The main purpose of this section is to explain the Video Display Device Control Block (DCB). This discussion will use the field names from the listing and their offsets from the base of the DCB. As mentioned earlier the address of the Video Display DCB can be obtained from the System Parameter Block (SPB).

VDDCHR - Offset 0

Depending on the state of the Video Display the cursor character may be either an inverse-video form of the character under it or a wedge-shaped character. This field keeps the character under the cursor so that the driver may easily replace it when moving the cursor.

VDDROW - Offset 1

In this byte is stored the current cursor row address, which will have a value varying from 0 to 23. The top line of the screen is row 0. Programs may interrogate this field to "find" the cursor on the screen, but should not write to it as a means of moving the cursor. If you do the Video Display driver will not be able to erase the old cursor and you will soon have a display full of cursors with only one of them being real.

VDDCOL - Offset 2

The current cursor column address is stored in this byte. It will range from 0 to 79 with 0 being the leftmost column on the screen. As with VDDROW this field should be considered "read only".

VDDINV - Offset 3

Data may be displayed in normal video (white characters on black background) or inverse video (black characters on white background). The current video mode is kept in bit 7 of this byte. A value of 0 means normal video, while 1 (80 hex) means inverse. No other value should ever be put in this byte except 0 and 80H.

VDDESC - Offset 4

VDDESX - Offset 5

These fields are used in processing the ESCape sequence used for cursor positioning. They provide no useful information to external programs and should never be modified.

VDCXAT - Offset 6

This table is a series of 32 byte entries that correspond to the control codes from 0 to 31 (00H to 1FH). Originally the table consisted of two-byte addresses of service routines to process the various control codes. During the final stages of implementing release 2.00 of CP/M, though, the bytes got very scarce and this table was converted to 32 one-byte offsets in a desperate attempt to recover space. This makes it somewhat unstable for future versions, since it cannot span a 256 byte RAM boundary, and it should not be counted on in subsequent releases.

The reason for using the table was so that any unused video control codes could be assigned as duplicates of existing codes to possibly provide compatibility with other computers. In particular the Kaypro series can be closely emulated in this way. When using this table the offsets which are defined should be considered read-only, while the others may be modified as needed. Don't bank on this table working the same in future generations.

This concludes the Video Display DCB. You may replace the entire Video Display driver with one of your own design, but be advised that Monte's Window may not function correctly if the driver is not located in BIOS RAM space. Since part of the BIOS code resides in the Video RAM space you must also switch the Video RAM in and out of the map as the existing driver does.

6. THE PARALLEL PRINTER DRIVER

The Model 4 computer is equipped with a Centronics-compatible parallel printer port. Our CP/M BIOS provides a driver for this port with some options that are contained in a Device Control Block (DCB). Operation of the driver is quite simple and should be apparent from the BIOS listing. The purpose of this section is to describe the DCB used for this port. Each field of the DCB and its offset follows.

PPDPRV - Offset 0

The character last output to the printer is kept here. This is needed whenever a linefeed is output, since we may not wish to output it at all depending on what went before. Manipulating this byte probably serves no useful purpose to an external program.

PPDOPT - Offset 1

All options set by the CONFIG utility are stored here. At present there are only two options. Bit 0, if set to 1, indicates that linefeeds which follow carriage returns are not to be output. Setting this bit to 0 causes all linefeeds to be output normally. Bit 1 is used to control the simulation of the formfeed character (0CH). For printers which do not recognize that code setting this bit to 1 will cause it to be simulated by repeated linefeeds. To do this the page length in lines must be known, along with a count of how many lines have been printed already. That information is kept in the following two fields. Bits 2 through 7 of this byte are not, as yet, used but are reserved for future use.

PPDLCT - Offset 2

This contains the count of the number of lines left to print on the current page. It is decremented each time a linefeed is sent to the driver, even if the linefeed was not actually performed. Unless bit 1 of PPDOPT is set this field serves no purpose. It is reset to the value in the next field after each Warm Boot.

PPDPGL - Offset 3

The number of lines on one page is kept in this byte. Unless changed by CONFIG the default is 66, which is a standard eleven inch page at six lines per inch. At Warm Boot and end of page time this value is loaded into PPDLCT.

There are no more fields in the Parallel Printer DCB. If you choose to replace this driver you may choose to use the existing DCB just so you can have access to the CONFIG settings.

7. THE SERIAL PORT DRIVER

The Serial Port on the Model 4 provides a standard RS-232C interface for external devices. Unfortunately the only thing standard about RS-232C seems to be which lines are used for data and ground. All others tend to change from one manufacturer to another. We have attempted to create a driver that will deal with as many different configurations as possible, but it is by no means comprehensive. In this section we will describe how the DCB is used. Operation of the driver, which is quite short, should be apparent from the BIOS listing. As with all the other drivers in this manual, each field is listed by name and offset.

SPDINT - Offset 0

This three-byte field is actually an executable Z-80 instruction. It contains a JuMP (opcode C3H) to a routine which will initialize the Serial Port and return. We had to provide for this capability since a change in Serial Port parameters almost always requires that the port be reinitialized. At this time this routine is used by CONFIG and MODEM7 (version 7.31). The initialization routine must end with a RET instruction (C9H).

SPDOPT - Offset 3

All of the options for the serial port driver are contained in this byte as bit flags. Bit 0, if set, tells the driver not to output data until the CTS (Clear To Send) line goes true. Similarly bit 1 is used to suspend output until DSR (Data Set Ready) becomes true. Bits 2-7 are reserved, but not in use at this time. The CTS and DSR lines of the Serial Port have presented somewhat of a problem for many users. On output the RTS (Request To Send) and DTR (Data Terminal Ready) lines must be inverted by the software. Since CTS and DSR are essentially the same signals coming from the other end of the RS-232 link we also inverted them before testing. This caused problems for many users, especially those with serial printers that use the lines for handshaking. To correct the problem we came out with version 2.22. The only change in this version from 2.21 was the removal of the XOR 80H following SPBSY in the Serial Port driver, as well as the XOR 40H following SPBSY1. These two instructions were removed by storing two bytes of zero (the NOP instruction) over the XORs.

SPBBDR - Offset 4

A single byte containing the code to be output to the baud rate generator is stored here. It is actually output when SPDINT is called. See the *Model 4 Technical Reference Manual* (any version) for a list of the codes.

SPDCFG - Offset 5

When SPDINT is called the byte in this field is output to the UART control register to configure it. As with SPBBDR you can get the codes from the *Model 4 Technical Reference Manual*.

That's all there is to the Serial Port DCB. Although we do have the hardware handshake problem addressed with this driver, it does not handle the common XON-XOFF software protocol used by many devices. In any future revision this option would have a high priority. It was omitted due to RAM constraints.

8. THE MEMORY DISK DRIVE

One of the features added in release 2 of the CP/M BIOS was the enabling of the Memory Disk (Drive M:) automatically at boot time. In release 1 the drivers for the Memory Disk were not part of the BIOS, but were loaded externally below the CCP. This was a messy implementation since the CCP then had to remain resident, resulting in a smaller TPA, and the system configuration could not be saved using CONFIG. By moving the drivers into the BIOS both problems were solved.

During a CP/M Cold Boot (RESET button) a test is made to see if the system has 128k of memory installed. If it does the 64k expansion RAM is filled with E5H bytes to make it appear to CP/M as a blank disk, and the DPH for drive M: is installed in the DPH table. When no extra RAM is present the entry in the DPH table for drive M: is simply left zero-filled.

Several users have requested the ability to leave drive M: in an uninitialized state, so that data could be preserved from one boot to another, or even from one DOS to another. Of course this can only happen if power has been constantly turned on between boots. By patching a RET instruction (C9H) into BOOT2 in the BIOS you can prevent drive M: from being initialized. However this leaves no way to ever get it initialized, which must be done when the system is first powered up. In future revisions we will probably provide for drive M: to be preserved if we can establish that it contains a flag showing that it was set up correctly.

Other users have asked that drive M: be permanently disabled and not initialized at all. This can be done simply by changing the JR NZ (opcode 20H) at location EA8BH on the BIOS listing to JR (opcode 18H). Here, too, we will consider implementing this capability on future versions.

Although the actual drivers for reading and writing to drive M: are quite short they may not be readily understandable. The problem is the "magic" that occurs when the expansion RAM is switched in and out of the memory map. We used a special 128 byte buffer in the BIOS to hold records on their way to or from drive M:. After calculating the expansion bank and RAM address of the desired "sector" the required bank is switched into the map at location 0000H. The transfer is made using the BIOS buffer, and the bank is switched back out. Because the first 32k is involved in this the BIOS can NEVER reside in any part below address 8000H. In other words it must always be in the top 32k of RAM.

This section on the RAM disk has been included mainly for the curious. It is not recommended that any tinkering be done on this driver. Those who choose to do so anyway should be warned that some products, notably Monte's Window, may fail miserably if the RAM disk is farked.

9. DISK I/O

The code to control the disk drives is one of the main parts of the BIOS. CP/M is a disk-based operating system and makes frequent use of disk drives. Floppy disks are standard on the Model 4/4P, and you may also optionally attach a hard disk drive. Code for the floppy disk is an integral part of the BIOS, but hard disk drive require the installation of a separate driver. Since this driver takes 1k of space it is necessary to reduce the Temporary Program Area (TPA) of CP/M by 1k with MOVCPM to make room for the additional code.

Although both types of disk drivers will be discussed in this section primary emphasis will be on the floppy driver. No listing is provided for the hard disk driver since the code is very much unique to the type of hard disk drive used. The hard disk drive DCB will be explained in full.

9.1. The Floppy Disk Driver

The standard disk Model 4/4P comes with two 40 track, single- sided disk drives, and can be equipped with two more external drives. One of the primary goals of the BIOS was to support any combination of disk drives and as many CP/M disk formats as possible. In this section we will describe the data structures used for disk I/O as well as the operation of the Floppy Disk driver itself.

Access to disk drives in CP/M is made using the Disk Parameter Header (DPH) and the Disk Parameter Block (DPB). Both of these structures are explained fully in David Cortesi's book *Inside CP/M*, which you got with your copy of CP/M. Our version of CP/M adds a few bytes to the end of the DPB and creates a new data structure called the Disk Device Control Block (DCB).

9.2. DPB Extensions

The standard CP/M DPB is 15 bytes long. Our additions fields follow the standard ones, preserving the original offsets. The added fields are:

DPBSPT - Offset 15

This byte contains the number of real sectors per track, NOT the number of 128 byte CP/M sectors per track. It is used mainly by external programs such as DUP who need to know this value for formatting purposes.

DPBSSZ - Offset 16

In this byte is stored a code which indicates the true size of a physical disk sector. Only the four IBM standard sizes are supported, using the following values:

- 00 = 128 bytes per sector
- 01 = 256 bytes per sector
- 02 = 512 bytes per sector
- 03 = 1024 bytes per sector

The value in this field is used both externally, mainly for formatting, and internally, in sector deblocking.

DPBDCB - Offset 17

This two-byte address points to the Device Control Block (DCB) of the physical disk drive assigned to this logical disk drive. By using a pointer, instead of putting the actual drive parameters in the DPB, we can assign two or more to the same physical drive. This saves trips to CONFIG when you need to use several different disk formats at the same time.

DPBOPT - Offset 19

A collection of bit flags is stored in this byte. These flags are normally set by CONFIG when establishing a format. Bit assignments are as follows:

- 7: Drive density
0=Single, 1=Double
- 6: Drive sides
0=Single sided, 1=Double sided
- 5: Drive stepping
0=Normal, 1=Double step (on 80 track drive)
- 4: Data status
0=Normal, 1=Inverted (Superbrain)
- 3: Sector numbering on double-sided drive
0=Same numbers on each side of disk
1=Side 1 continues where side 0 left off
- 2: Track numbering on double-sided drive
0=Track numbers same on each side
1=Even tracks on side 0, odd tracks on side 1
- 1: Side selection on double-sided drives
0=Tracks map on alternating sides
1=Tracks map first on side 0, then on side 1
- 0: Track usage on side 1 if bit 1 is set to 1
0=Tracks run from track 0 to innermost track
1=Tracks run from innermost track back to track 0

It is our hope that these bits provide the means to access any format, no matter how strange. Bits 0 and 1 were initially unused, due to a naive belief on the part of the BIOS programmer that all double-sided formats alternated from side to side before stepping to a higher track. This was particularly embarrassing since one of our earlier versions of CP/M was among the ones that didn't follow this pattern. As a result the necessary code to handle this condition was left out of the BIOS and had to be added later using a patch program called PATCHIOS.

Most of the bits in the byte are used by the disk driver in the BIOS. Use extreme care in setting them, since the driver does not blindly do I/O from selected tracks and sectors, but rather makes decisions based on the parameters passed to it and the bits set in this byte.

DPBDID - Offset 20

This last byte of the DPB is used to keep track of what format has been assigned to a drive. A value of zero is used to signal that the drive is not a floppy disk, although the floppy disk driver pays no attention to this field. Values from 1 to 128 refer to entries in the DISK.FDF file. Since that file is subject to change this byte may become unexpectedly obsolete and point to the wrong format definition. However BIOS space is very tight, and this byte is used by DUP and CONFIG only to extract the format description for the drive, so the danger is small.

9.3. DCB Definitions

There are four Disk DCBs, one for each of four possible physical drives on the Model 4. Conceivably one could add more, but it would serve no practical purpose. Since the DCB is assigned to a logical drive by a pointer in the DPB it is possible for one DCB to serve multiple logical drives.

DKDDVR - Offset 0

This first field in the DCB is actually a 3 byte Z80 JP instruction. The first byte is 0C3H, the opcode for JP, followed by the address of the disk driver. We used this technique so that it would be relatively easy to add other drives to the system, such as hard disk drives.

DKDSEL - Offset 3

Each of the drives on the Model 4 has a unique select address, indicated by setting one of the four low order bits. Only one bit should be set in this byte, corresponding to the physical address of the drive.

DKDATT - Offset 4

In this byte we use bit flags to keep track of the physical attributes of the drive. Bit assignments are:

- 7: 0=Single sided drive
1=Double sided drive
- 6: 0=5 1/4 inch drive
1=8 inch drive
- 5: Reserved
- 4: Reserved
- 3: Reserved
- 2: Reserved
- 1,0: Drive step rate code
0=6 ms, 1=12 ms, 2=20 ms, 3=30 ms

These bits are normally set by the CONFIG utility.

DKDSTD - Offset 5

This byte contains the start-up delay time for the drive in quarter seconds. It is arbitrarily set at 2, which gives 1/2 second of delay. This byte cannot be set by CONFIG, but must be set using DDT.

DKDSTL - Offset 6

After a seek operation it is necessary to give the head time to settle before attempting to read or write. The required settle time in milliseconds is stored here, initially set at 15. This byte cannot be changed except by direct patch.

DKDNTK - Offset 7

The number of tracks that a drive can physically access is stored here. The driver will not step to any track beyond this limit.

DKDPTO - Offset 8

Since the disk controller in the Model 4/4P has write precompensation controlled by software the driver must know when to turn it on. The value is not arrived at scientifically, but more by divine inspiration. In his experience the author has not had any problems using half the number of tracks plus 2, so that "magic" number is used here.

DKDCTK - Offset 9

With the potential for up to four drives in the system it is necessary for the driver to reset the current track in the disk controller when changing drives. The current track value is stored here, and is normally not written to. One exception is at cold boot time when 0FFH is written in this field for each DCB. A value of 0FFH forces the driver to restore the drive prior to doing any disk I/O.

DKDCSL - Offset 10

In addition to selecting a disk drive the hardware drive select register also sets the density, write precompensation, and side for the drive being accessed. Once all this information has been collected it is stored here for "refreshing" the drive select register.

DKDLTK - Offset 11

Most of the time the track number recorded on the disk will correspond to the track number that the drive read/write head is positioned over. On those occasions where this is not the case, e.g. an 80 track drive used with a 40 track format, this byte holds the logical track that is expected on the disk.

Although this DCB is used only by the BIOS Floppy Disk Driver and the CONFIG utility, it should not be changed. For other disk drivers you may want to create a DCB which meets the requirements of the drive. This was done when adding hard disk drives to CP/M.

9.4. Using the Disk Driver

It is quite possible to use the BIOS Floppy Disk Driver in external programs. The driver is capable of reading or writing a sector, but does not contain code for more exotic functions such as format. The calling setup is as follows:

- A Contains function code
 - 1 = Read sector
 - 2 = Write sector
- BC Contains track number (B should always be 0)
- DE Contains sector number (D should always be 0)
 - This is the actual sector, i.e. interleave must already be figured before calling the driver.
- HL Contains address of data buffer
- IX Contains address of DCB for selected drive
- IY Contains address of DPB for selected drive

On return the A register contains the status read from the 17xx disk controller status register. All non-error bits are masked off so only error conditions need be checked for.

When using this driver on double-sided disk drives you must be aware that the controller will determine for itself what track, sector, and side to access. This decision will be based on the rules established in field DPBOPT of the DPB pointed to by the IY register. Study this field carefully and set your track and sector numbers accordingly.

9.5. EXBIOS - EXtending the BIOS

As mentioned earlier all possible methods of accessing a disk in CP/M were not handled in the original BIOS. When this fact became known it was not possible to fix the problem with a simple patch due to the fact that the BIOS filled all but 3 bytes of the two tracks originally reserved for the system. The idea of adding another reserved track was rejected, due to the potential problems that it could cause for existing users. Therefore we decided to create a small program that would install the fix into BIOS memory after the system was booted, since it would remain in place until the next full reset.

The instruction where EXBIOS is "hooked" in is shown in the listing of the Floppy Disk Driver, after the label FDBEGN. At the end of the listing the code for the BIOS extension is given. Note that the CONFIG utility knows about EXBIOS and will remove it before saving the configuration to disk. This must be done since the portion of memory that the extension resides in will never be saved in the two reserved tracks. In the next revision this code will be absorbed back into the BIOS where it belongs.

9.6. The Hard Disk Driver

As good as CP/M is, the use of a hard disk drive makes it even better. With disk space in the millions of bytes instead of thousands you can have all of your favorite software available at once. Even better, programs load many times faster and all disk I/O is generally faster.

Like all good things, this convenience carries a price. Since a hard disk drive is not a part of the standard Model 4/4P computer it must be purchased separately. There are dozens of possible configurations, so the code to access the hard disk cannot be economically contained in the BIOS. MOVCPM is used to make a smaller CP/M (no larger than 63k) so that the extra space at the top of memory can be used for the hard disk driver code.

Montezuma Micro offers hard disk drivers for a broad range of hard disk drives, and the code varies according to the type of controller and drive used. For the convenience of external programs we have kept the DPB for hard drives the same as for floppies. The only real difference is in the DPBDID field, which is always set to 0 for a hard disk. Only the memory disk, drive M:, can have a format ID of 0.

The fields of the hard disk drive DCB are unique to that device, although the first field must be the same as that of the floppy disk drive. This is the only link that the BIOS has to the device driver. Here are the fields:

DKDDVR - Offset 0

Like the floppy disk driver, this field is actually a 3 byte JP instruction to the driver routine. The first byte contains 0C3H, and the last two contain the address of the entry point of the hard disk driver.

DKDSEL - Offset 3

This byte contains the drive select bits for the hard disk drive. The actual bit usage will vary from one controller to another, and is of importance only to the driver itself.

DKDCYL - Offset 4

This is a two-byte field containing the 16-bit count of cylinders on the hard disk. CP/M is oblivious to the cylinder concept and works only with tracks. Head positioning on hard disk drives, however, is done by cylinders. Some drivers use this field, others don't. We recommend that it be kept at this location in the DCB for the convenience of external utilities that may need such information.

DKDOFF - Offset 6

In this two-byte field is kept the 16-bit count of CP/M tracks that precede this logical drive on the physical hard drive. At first glance this might appear to be the same thing as field DPBOFF, but that field cannot be used if drive A: is to be located on any track but the very first track of the hard drive. To avoid such restrictions we let CP/M think that each logical hard drive is an entity all to itself, and use the DCB to sort out who lies where.

The above fields are more or less standard for our drivers. Some drivers may have additional fields in the DCB, but these are for driver use only. Any drivers of your own creation should include these standard fields as a minimum. Also your driver **MUST** use the same calling sequence as the floppy driver, with regard to registers, etc. If it doesn't the BIOS will not be able to call it correctly.

10. CP/M BOOTS

In addition to providing I/O drivers for CP/M the BIOS performs one other critical function, that of bootstrap loading the operating system. The term "boot" comes from the phrase "pulling yourself up by your own bootstraps." When the Model 4/4P is first powered up there is no software in RAM. The first level of boot is the ROM, which reads track 0, sector 1, of disk drive 0 into RAM beginning at 4300H. This sector must be in double density. It may be any length, although the 4P will insist on first loading the ROM image for any size other than 256 bytes.

In CP/M there are two boot processes generally referred to as the Cold Boot and the Warm Boot. The Cold Boot is so named because it is activated when the machine is first turned on, i.e. is still "cold." At other times, usually between programs, a Warm Boot is performed. While the Cold Boot loads the entire CP/M operating system, including the BIOS, the Warm Boot loads only the CCP and the BDOS. The remainder of this section will discuss each of these two routines.

10.1. The Cold Boot

Once the boot sector has been loaded it will proceed to load the CCP, BDOS, and BIOS into their designated locations. Control is then given to the first BIOS vector, which in turn transfers to the label BOOT in the listing. The system stack is established at address 0000H. At first glance this may appear strange, but in fact when the stack is written to it is first decremented. This results in the stack pointer "wrapping around" to the top of memory so that the data is actually written to 0FFFFH downward.

The first order of business is to do a complete reset of all I/O devices, mainly to initialize the associated DCBs, as well as to clear the RAM work areas used by the BIOS. Next drive A: is made the current drive, and the current track number is set invalid on all floppy drives. Drives A:, B:, C:, and D: are set up in the DPH table.

The next routine has been subject to some criticism by CP/M users. It first tests for the presence of the 128k RAM option. If found all 64k of the expansion RAM is set to 0E5H and drive M: is entered into the DPH table. Many users have requested that the RAM drive be formatted (i.e. filled with 0E5H bytes) ONLY if it is found to be corrupt. At present such a test would require a major change in the BIOS and more memory than is currently available. However we will consider this in the next revision.

Finally, if configured for it, the Cold Boot displays the opening "banner" announcing the CP/M size and version. Control then passes to the CCP, which issues the "A>" prompt and begins CP/M operation. If the system was booted from a hard drive (Model 4P with Radio Shack hard disk only) the jump to the CCP is patched to return to the hard drive boot. There the DPH table is modified to reflect the drive configuration, both floppy and hard, and then the jump is made to the CCP.

10.2. The Warm Boot

Like the Cold Boot, the first thing the Warm Boot must do is to establish a stack. However this stack runs from 0080H to 00FFH. It cannot reside in high memory due to the use of the last 128 bytes by the disk I/O routines for internal stack space.

Next it clears the BIOS disk buffer. This step is very important, since different sizes of disk sectors are used. It is possible for a program to terminate with a write operation pending on the sector currently in the disk buffer. By making this call the Warm Boot can be sure that all pending writes have been serviced.

Next a "warm" reset is done. Essentially this just resets the device drivers.

The remainder of the Warm Boot code reloads the CCP and the BDOS from drive A:. Loading begins at track 0, CP/M sector 2, using the assumption that the DPB for A: contains valid information for deblocking. Some CP/Ms access the system tracks in a different manner from the rest of the disk (including version 1.xx of ours), but we have chosen to be consistent throughout the disk. It makes utility writing so much easier.

11. PITFALLS AND TRAPS

After examining what we have done in the BIOS you may be filled with an urge to improve on it. Before you go wading in with a flailing text editor we'd like to tip you off to a few things.

11.1. INTERRUPTS

Interrupts are truly wonderful things. With interrupts one computer can be made to appear to be doing several things at once. Then again they can also cause disasters of truly epic proportions, as well as introduce bugs that are tougher to kill than a New York Cockroach.

We have chosen not to implement interrupts in the BIOS. Why not? Well the first reason is the Model /4P hardware. It dictates that maskable interrupts will generate a ReSTart to location 38H. Coincidentally this is also RST 7 on the 8080, and it is used by some CP/M software, most notably DDT and other debug utilities. While we could release our CP/M with a modified version of DDT it is certain that other programs out there also use RST 7. One of the main goals for our CP/M was to be compatible with the rest of the world, and a conflict over a ReSTart address would make that goal unattainable.

A second reason for not interrupting has to do with memory management. The Model 4/4P has all sorts of memory map possibilities. Interrupting when the wrong map was in could create disaster, so elaborate locking schemes would be necessary to keep them turned off at critical times. This would mean a much larger BIOS, one which was not as robust.

What do we lose by not having interrupts? A keyboard type-ahead buffer is more difficult to do (but not impossible). A steady blinking cursor is real tough, and background I/O such as serial communications is all but impossible. This is only a partial list. There may be other things, too. Systems software is a constant trade-off situation. We are happy with the choices we have made.

11.2. FEATURES UNIQUE TO THE Z80

While thumbing through the listing you Z80 gurus may begin thinking "Hey, I could shorten this code up by using the IX register here, and the alternate registers there." No doubt you are right, but we have tried to avoid using Z80 features just because they are there.

In the early days of CP/M all programs were written for the 8080 and it was perfectly safe to use Z80 features without fear of overlap. This is no longer true. Newer programs, such as Turbo Pascal, use ALL of the Z80 features. Using them in the BIOS could lead to surprise crashes on a massive scale. Help stamp out unscheduled Cold Boots and be very conservative in your code.

11.3. RAM USAGE

While perusing CP/M literature you may notice the odd bit of RAM that is reserved for, but not used by, our BIOS. One example of this is locations 0040H through 004FH. It is reserved for the BIOS, but not referenced at all by ours.

Using this area would be unwise, however, because Monte's Window uses the fool out of it. There are other little cracks and crevices in the RAM above and below the BIOS. Enhancement architects are always looking for little crannies to stick their constructions into. Be very, very careful about using these since we sometimes need RAM, too, and we don't know or care what you have used a RAM "hole" for.

12. INDEX

A

Assembler: 1

B

BIOS: 1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 15, 17, 19, 20, 22, 23, 24, 25, 26, 27, 28, 31

Boot: 3, 4, 13, 17, 22, 23, 25, 26, 27

 Cold: 17, 22, 25, 26, 27

 Warm: 3, 13, 25, 26

D

DCB: 4, 9, 10, 11, 12, 13, 15, 19, 20, 21, 22, 23, 24, 25

Device control block: 4, 9, 13, 19, 20

Device driver: 4, 7, 23

 Device driver address table: 4

Device Parameter Block: 19

 DPB: 19, 20, 21, 22, 23, 24, 26

Device Parameter Header: 3

 DPH: 3, 4, 17, 25

Disk I/O: 4, 19, 23, 26

E

EXBIOS: 20, 23

F

Floppy disk driver: 19, 22, 23

Function keys: 9, 10

H

Hard disk driver: 4, 19, 23

I

Initialization: 15

Installation: 4, 7, 19

Interrupts: 27

IOBYTE: 3, 4, 5

K

Keyboard driver: 4, 6, 9, 10

 CAPS lock: 10

 Key definitions: 10

L

Logical device: 5, 6

M

Memory disk: 17, 23

P

Parallel printer driver: 13

Physical device: 5, 6

S

Serial port driver: 15

SPB: 3, 4, 7, 11, 15

System Parameter Block: 3

 BIOS version: 1

 Boot display: 25

 Disk DCB: 4, 21

 DPH table: 4, 17, 25

V

Video Display DCB: 11

 Control codes: 11, 12

 Cursor: 4, 11, 27

Z

Z80 features: 27

Z80 mnemonics: 1

13. THE LISTING

If you're a true System Programmer you've probably already been through the listing of the BIOS before reading the text. You may have noticed that there are some routines missing. For the most part these are mundane little service routines that are not significant to the operation of the BIOS as a whole. Our intent with this manual was not to give you every last byte of source code, but rather a tool with which you could interact with the BIOS.

The bottom line is "This is it!". Please don't write or call us with sad stories about why you need this unpublished routine or that undocumented code. You won't get it. We have a lot of time and money invested in bringing our CP/M this far. Modesty tells us that it could be improved, but good business sense tells us that we should reserve the first option for making those improvements.

INPUT FILENAME : BIOS.ASM
OUTPUT FILENAME : BIOS.OBJ

TRS-80 Model 4 BIOS Version 2.00+ General Definitions

```

;
; Copyright (c) (p) 1984
; Montezuma Micro
; P. O. Box 763009
; Dallas, TX 75376-3009
;
; All rights reserved
;
; This BIOS is written for Montezuma Micro CP/M 2.2.
;
; *****
; * CP/M address constants
; *****
00 D4 BASE EQU 0D400H ;Base for 64K system
00 D4 CCP EQU BASE ;Base of CCP
06 DC BDOS EQU CCP+806H ;Base of BDOS
00 EA BIOS EQU CCP+1600H ;Base of BIOS
40 00 MSIZE EQU (BIOS+1200H)/1024+1 ;Memory size in K bytes
;
; >>---> WARNING: BIOS must be 8000H or higher!
;
00 00 WBJP EQU 0000H ;BIOS Warm boot vector
03 00 IOBYTE EQU 0003H ;System I/O byte
04 00 CDISK EQU 0004H ;System current disk drive
80 00 DEFBUF EQU 0080H ;Default disk buffer
2C 00 NRECS EQU (BIOS-CCP)/128 ;Number of warm boot recs
;
; *****
; * Model 4 port addresses
; *****
84 00 MEMCTL EQU 84H ;Memory mapping port
90 00 SOUND EQU 90H ;Sound control port
E0 00 INTCTL EQU 0E0H ;Interrupt control port
E4 00 NMICTL EQU 0E4H ;Non-maskable interrupt ctrl
E8 00 SERRST EQU 0E8H ;Serial port reset
E9 00 SERBRG EQU 0E9H ;Serial port baud rate gen.
EA 00 SERURT EQU 0EAH ;Serial port UART ctl/status
EB 00 SERDAT EQU 0EBH ;Serial port data
EC 00 MISCTL EQU 0ECH ;Miscellaneous function port
F0 00 FDCCTL EQU 0F0H ;Disk command/status
F1 00 FDCTRK EQU 0F1H ;Disk track
F2 00 FDCSEC EQU 0F2H ;Disk sector
F3 00 FDCDAT EQU 0F3H ;Disk data
F4 00 FDCSEL EQU 0F4H ;Disk select
F8 00 PARSDT EQU 0F8H ;Parallel port status/data
;
; *****
; * Model 4 data constants
; *****
8E 00 KVMIN EQU 8EH ;Keyboard/Video mapped in
8F 00 KVMOUT EQU 8FH ;Keyboard/Video mapped out

```

TRS-80 Model 4 BIOS Version 2.00+ Entry vectors & configuration data

```

;
; ORG BIOS ;Start BIOS code
;
; *****
; * Standard BIOS jump vectors
; *****

```

EA00	C3 4B EA	JP	BOOT	;Cold start
EA03	C3 61 EB	JP	WBOOT	;Warm start
EA06	C3 D0 EB	JP	CONST	;Console status
EA09	C3 F2 EB	JP	CONIN	;Console character in
EA0C	C3 02 EC	JP	CONOUT	;Console character out
EA0F	C3 1A EC	JP	LIST	;List character out
EA12	C3 3E EC	JP	PUNCH	;Punch character out
EA15	C3 52 EC	JP	READER	;Reader character in
EA18	C3 E5 F1	JP	HOME	;Restore disk drive
EA1B	C3 78 F1	JP	SELDSK	;Select disk drive
EA1E	C3 93 F1	JP	SETTRK	;Set track number
EA21	C3 9C F1	JP	SETSEC	;Set sector number
EA24	C3 DD F1	JP	SETDMA	;Set DMA address
EA27	C3 ED F1	JP	READ	;Read disk
EA2A	C3 24 F2	JP	WRITE	;Write disk
EA2D	C3 2C EC	JP	LISTST	;List status
EA30	C3 E2 F1	JP	SECTRN	;Sector translation

```

;
; *****
; * System Parameter Block *
; * This block is used to contain configuration data of *
; * a general nature that is required by the BIOS and *
; * external routines that may need to modify it. *
; *****
EA33 33 EA SPB EQU $
EA33 81 SPBIOB DEFB 81H ;IOBYTE: LPT,TTY,TTY,CRT +0
EA34 FF SPBSOM DEFB 0FFH ;Display sign-on at boot +1
EA35 02 DEFB 2 ;Total # of disk drives +2
EA36 22 DEFB 22H ;BIOS version number +3
EA37 FD F6 DEFW DPHTBL ;DPH table address +4
EA39 55 F6 DEFW D0DCB ;Disk DCB 0 address +6
EA3B 61 F6 DEFW D1DCB ;Disk DCB 1 address +8
EA3D 6D F6 DEFW D2DCB ;Disk DCB 2 address +10
EA3F 79 F6 DEFW D3DCB ;Disk DCB 3 address +12
EA41 73 EC DEFW DDATBL ;Device Driver Address +14
EA43 9A EE DEFW KBDCB ;Keyboard DCB +16
EA45 8D F0 DEFW VDDCB ;Video Display DCB +18
EA47 24 F1 DEFW PPDCB ;Parallel Port DCB +20
EA49 72 F1 DEFW SPDCB ;Serial Port DCB +22

```

TRS-80 Model 4 BIOS Version 2.00+ Boot routines

```

;
; *****
; * BIOS Cold Start entry *
; * Input: None *
; * Output: None - System loaded into RAM *
; *****
EA4B 31 00 00 BOOT LD SP,0000H ;Set stack at top of RAM
EA4E CD D3 EC CALL CRESET ;Do a complete reset
;
EA51 AF XOR A ;Current drive/USER=A/0
EA52 32 04 00 LD (CDISK),A
EA55 32 3F EB LD (BANNRM),A ;Kill drive M message
;
EA58 3D DEC A ;Reset drive track history
EA59 32 5E F6 LD (D0DCB+DKDCTK),A
EA5C 32 6A F6 LD (D1DCB+DKDCTK),A
EA5F 32 76 F6 LD (D2DCB+DKDCTK),A
EA62 32 82 F6 LD (D3DCB+DKDCTK),A
;
EA65 21 9C F5 LD HL,DPHA ;Point HL at first DPH
EA68 01 10 00 LD BC,16 ;BC=length of DPH
EA6B 22 FD F6 LD (DPHTBL),HL ;Set Drive A in DPHTBL
EA6E 09 ADD HL,BC
EA6F 22 FF F6 LD (DPHTBL+2),HL ; Drive B
EA72 09 ADD HL,BC

```

EA73	22 01 F7	LD	(DPHTBL+4),HL	; Drive C
EA76	09	ADD	HL,BC	
EA77	22 03 F7	LD	(DPHTBL+6),HL	; Drive D
;				
EA7A	21 00 00	LD	HL,0000H	;Point to start of RAM
EA7D	3E EF	LD	A,KVMOUT+60H	;Switch in expansion bank 0
EA7F	D3 84	OUT	(MEMCTL),A	
EA81	36 3C	LD	(HL),3CH	;Plug with inversion of C3H
EA83	3E 8F	LD	A,KVMOUT	;Switch back to main RAM
EA85	D3 84	OUT	(MEMCTL),A	
EA87	7E	LD	A,(HL)	;Get test byte
EA88	36 C3	LD	(HL),0C3H	;Replace in case it changed
EA8A	BE	CP	(HL)	;Is it unchanged?
EA8B	20 1E	JR	NZ,BOOT1	;Go if changed - not 128K
EA8D	3E EF	LD	A,KVMOUT+60H	;Switch in expansion bank 0
EA8F	D3 84	OUT	(MEMCTL),A	
EA91	CD BA EA	CALL	BOOT2	;Fill 32K with E5 bytes
EA94	29	ADD	HL,HL	;Set HL back to 0000H
EA95	3E FF	LD	A,KVMOUT+70H	;Switch in expansion bank 1
EA97	D3 84	OUT	(MEMCTL),A	
EA99	CD BA EA	CALL	BOOT2	;Fill 32K with E5 bytes
EA9C	3E 0D	LD	A,0DH	;Enable drive M message
EA9E	32 3F EB	LD	(BANNRM),A	
EAA1	21 DC F5	LD	HL,DPHM	;Set up DPH for M:
EAA4	22 15 F7	LD	(DPHTBL+24),HL	
EAA7	3E 8F	LD	A,KVMOUT	;Restore lower RAM map
EAA9	D3 84	OUT	(MEMCTL),A	
;				
EAAB	21 C2 EA	BOOT1 LD	HL,BANNER	;Point to opening banner
EAAE	3A 34 EA	LD	A,(SPBSOM)	;Check the signon flag
EAB1	B7	OR	A	
EAB2	C4 08 ED	CALL	NZ,DISPLY	;Display if requested
EAB5	0E 00	LD	C,0	;Set default drive to A:
EAB7	C3 00 D4	JP	CCP	;Go to CP/M
;				
EABA	36 E5	BOOT2 LD	(HL),0E5H	;Store an E5 byte
EABC	23	INC	HL	;Advance pointer
EABD	CB 7C	BIT	7,H	;Check bit 7 of address
EABF	C0	RET	NZ	;Exit if at 32K
EAC0	18 F8	JR	BOOT2	;Keep filling
;				
; CP/M signon banner				
; -----				
EAC2	1A 07 16	BANNER DEFB	1AH,07H,16H	
EAC5	54 52 53 2D	DEFB	'TRS-80 Model 4 '	
EAC9	38 30 20 4D			
EACD	6F 64 65 6C			
EAD1	20 34 20			
EAD4	36 34	DEFB	MSIZE/10+'0',MSIZE.MOD.10+'0'	
EAD6	6B 20 43 50	DEFB	'k CP/M vers 2.2 '	
EADA	2F 4D 20 76			
EADE	65 72 73 20			
EAE2	32 2E 32 20			
EAE6	28 63 29 20	DEFB	'(c) (p) 1982 Digital Research Inc.'	
EAEA	28 70 29 20			
EAAE	31 39 38 32			
EAF2	20 44 69 67			
EAF6	69 74 61 6C			
EAFA	20 52 65 73			
EAFE	65 61 72 63			
EB02	68 20 49 6E			
EB06	63 2E			
EB08	15 0D 0A	DEFB	15H,0DH,0AH	
EB0B	42 49 4F 53	DEFB	'BIOS vers 2.20 '	
EB0F	20 76 65 72			
EB13	73 20 32 2E			

EB1A	28 63 29 20	DEFB	'(c) (p)1984 Montezuma Micro/JBO'
EB1E	28 70 29 20		
EB22	31 39 38 34		
EB26	20 4D 6F 6E		
EB2A	74 65 7A 75		
EB2E	6D 61 20 4D		
EB32	69 63 72 6F		
EB36	2F 4A 42 4F		
EB3A	15 16 00 0A	DEFB	15H,16H,0DH,0AH,0AH
EB3E	0A		
EB3F	00	BANNRM DEFB	0
EB40	3E 3E 3E 20	DEFB	'>>> Memory Drive M: '
EB44	4D 65 6D 6F		
EB48	72 79 20 44		
EB4C	72 69 76 65		
EB50	20 4D 3A 20		
EB54	16 45 4E 41	DEFB	16H,'ENABLED',16H
EB58	42 4C 45 44		
EB5C	16		
EB5D	00 0A 0A 00	DEFB	0DH,0AH,0AH,0
; *****			
; * BIOS Warm Start entry *			
; * Input: None *			
; * Output: None - System reloaded into RAM *			
; *****			
EB61	31 00 01	WBOOT LD	SP,DEFBUF+128 ;Use buffer for stack
EB64	CD 07 F3	CALL	CLBUF ;Clear the BIOS disk buffer
EB67	CD E0 EC	CALL	WRESET ;Do a warm reset
;			
EB6A	0E 00	LD	C,0 ;Select drive A:
EB6C	CD 78 F1	CALL	SELDISK
EB6F	01 0A 00	LD	BC,DPHDPB ;Point HL at DPB
EB72	09	ADD	HL,BC
EB73	7E	LD	A,(HL)
EB74	23	INC	HL
EB75	66	LD	H,(HL)
EB76	6F	LD	L,A
EB77	7E	LD	A,(HL) ;Records/track to HL
EB78	23	INC	HL
EB79	66	LD	H,(HL)
EB7A	6F	LD	L,A
EB7B	22 34 F7	LD	(DSBRPT),HL ;Save it
EB7E	21 00 00	LD	HL,0 ;Set starting track
EB81	22 30 F7	LD	(DSBNTK),HL
EB84	2E 02	LD	L,2 ;Set up starting sector
EB86	22 32 F7	LD	(DSBNSC),HL
EB89	21 00 D4	LD	HL,CCP ;Set beginning DMA
EB8C	22 25 F7	LD	(DSBDMA),HL
EB8F	06 2C	LD	B,NRECS ;Set record counter
EB91	C5	WBOOT1 PUSH	BC ;Save record counter
EB92	ED 4B 30 F7	LD	BC,(DSBNTK) ;Set the track
EB96	CD 93 F1	CALL	SETTRK
EB99	ED 4B 32 F7	LD	BC,(DSBNSC) ;Set the sector
EB9D	CD 9C F1	CALL	SETSEC
EBA0	CD ED F1	CALL	READ ;Read the record
EBA3	B7	OR	A ;Any error?
EBA4	20 BB	JR	NZ,WBOOT ;If so start all over
EBA6	21 30 F7	LD	HL,DSBNTK ;Update sector #
EBA9	CD 90 F2	CALL	NXTSEC
EBAC	2A 25 F7	LD	HL,(DSBDMA) ;Update DMA
EBAF	01 80 00	LD	BC,128
EBB2	09	ADD	HL,BC
EBB3	22 25 F7	LD	(DSBDMA),HL

EBB6	C1	POP	BC	;Restore record counter
EBB7	10 D8	DJNZ	WBOOT1	;Loop until complete
EBB9	3A 04 00	LD	A,(CDISK)	;Get current drive #
EBBC	E6 0F	AND	0FH	;Mask off user code
EBBE	4F	LD	C,A	;Drive # to C
EBBF	CD 78 F1	CALL	SELDSK	;Select it (validate)
EBC2	7C	LD	A,H	;Check for validity
EBC3	B5	OR	L	
EBC4	20 03	JR	NZ,WBOOT3	;Go if valid drive
EBC6	32 04 00	LD	(CDISK),A	;Reset to USER 0, A:
EBC9	3A 04 00	LD	A,(CDISK)	;Set User/Default Drive
EBCC	4F	LD	C,A	
EBCD	C3 03 D4	JP	CCP+3	;Go to CCP

TRS-80 Model 4 BIOS Version 2.00+ I/O routines for CON: device

```

;
; *****
; * Report console status
; * Input: None
; * Output: A=FFH if input present
; *          00H if no input present
; *****
EBD0  CD 12 EC  CONST  CALL  CONIOB      ;Get CON IOBYTE
EBD3  28 0B      JR      Z,CONST1    ;Go if BAT status
EBD5  CD 64 EC  CALL  IODSP      ;Call I/O dispatcher
EBD8  73 EC      DEFW  TTYSTS      ; TTY status
EBDA  7B EC      DEFW  CRTSTS      ; CRT status
EBDC  CB EC      DEFW  NULSTS      ; BAT status (Dummy entry)
EBDE  83 EC      DEFW  UC1STS      ; UC1 status
EBE0  3A 03 00  CONST1 LD  A,(IOBYTE) ;Get the IOBYTE
EBE3  0F          RRCA             ;Isolate RDR bits
EBE4  0F          RRCA
EBE5  E6 03      AND  03H
EBE7  CD 64 EC  CALL  IODSP      ;Call I/O dispatcher
EBEA  73 EC      DEFW  TTYSTS      ; TTY status
EBEC  9B EC      DEFW  PTRSTS      ; PTR status
EBEE  A3 EC      DEFW  UR1STS      ; UR1 status
EBF0  AB EC      DEFW  UR2STS      ; UR2 status

;
; *****
; * Console input
; * Input: None
; * Output: A=Character input from console
; *****
EBF2  CD 12 EC  CONIN  CALL  CONIOB      ;Get CON IOBYTE
EBF5  28 5B      JR      Z,READER     ;Go if BAT
EBF7  CD 64 EC  CALL  IODSP      ;Call I/O dispatcher
EBFA  75 EC      DEFW  TTYINP      ; TTY input
EBFC  7D EC      DEFW  CRTINP      ; CRT input
EBFE  CD EC      DEFW  NULINP      ; BAT input (Dummy entry)
EC00  85 EC      DEFW  UC1INP      ; UC1 input

;
; *****
; * Console output
; * Input: C=Character to be output to console
; * Output: None
; *****
EC02  CD 12 EC  CONOUT CALL  CONIOB      ;Get CON IOBYTE
EC05  28 13      JR      Z,LIST       ;Go if BAT
EC07  CD 64 EC  CALL  IODSP      ;Call I/O dispatcher
EC0A  77 EC      DEFW  TTYOUT      ; TTY output
EC0C  7F EC      DEFW  CRTOUT      ; CRT output
EC0E  CC EC      DEFW  NULOUT      ; BAT output (Dummy entry)
EC10  87 EC      DEFW  UC1OUT      ; UC1 output

```



```

; *****
; * Return CON IOBYTE value
; * Input: None
; * Output: A=CON iobyte value, Z flag set if BAT
; *****
EC12 3A 03 00 CONIOB LD A,(IOBYTE) ;Get the IOBYTE
EC15 E6 03 AND 03H ;Isolate CON bits
EC17 FE 02 CP 02H ;Check for BAT
EC19 C9 RET

```

TRS-80 Model 4 BIOS Version 2.00+ I/O routines for LST: device

```

; *****
; * Output character to LST device
; * Input: C=character to be output
; * Output: None
; *****
EC1A 3A 03 00 LIST LD A,(IOBYTE) ;Get the IOBYTE
EC1D 07 RLCA ;Isolate LST bits
EC1E 07 RLCA
EC1F E6 03 AND 03H
EC21 CD 64 EC CALL IODSP ;Call I/O dispatcher
EC24 77 EC DEFW TTYOUT ; TTY output
EC26 7F EC DEFW CRTOUT ; CRT output
EC28 8F EC DEFW LPTOUT ; LPT output
EC2A 97 EC DEFW UL1OUT ; UL1 output

```

```

; *****
; * Return LST status
; * Input: None
; * Output: A=LST status
; *****
EC2C 3A 03 00 LISTST LD A,(IOBYTE) ;Get the IOBYTE
EC2F 07 RLCA ;Isolate LST bits
EC30 07 RLCA
EC31 E6 03 AND 03H
EC33 CD 64 EC CALL IODSP ;Call I/O dispatcher
EC36 79 EC DEFW TTYBSY ; TTY busy
EC38 81 EC DEFW CRTBSY ; CRT busy
EC3A 91 EC DEFW LPTBSY ; LPT busy
EC3C 99 EC DEFW UL1BSY ; UL1 busy

```

TRS-80 Model 4 BIOS Version 2.00+ I/O routines for PUN: device

```

; *****
; * Output character to PUN device
; * Input: C=character to output
; * Output: None
; *****
EC3E 3A 03 00 PUNCH LD A,(IOBYTE) ;Get the IOBYTE
EC41 07 RLCA ;Isolate PUN bits
EC42 07 RLCA
EC43 07 RLCA
EC44 07 RLCA
EC45 E6 03 AND 03H
EC47 CD 64 EC CALL IODSP ;Call I/O dispatcher
EC4A 77 EC DEFW TTYOUT ; TTY output
EC4C B7 EC DEFW PTPOUT ; PTP output
EC4E BF EC DEFW UP1OUT ; UP1 output
EC50 C7 EC DEFW UP2OUT ; UP2 output

```

EC52 3A 03 00
EC55 0F
EC56 0F
EC57 E6 03
EC59 CD 64 EC
EC5C 75 EC
EC5E 9D EC
EC60 A5 EC
EC62 AD EC

```

;
; *****
; * Input from RDR device
; * Input: None
; * Output: A=character input
; *****
READER LD A,(IOBYTE) ;Get the IOBYTE
RRCA ;Isolate RDR bits
RRCA
AND 03H
CALL IODSP ;Call I/O dispatcher
DEFW TTYINP ; TTY input
DEFW PTRINP ; PTR input
DEFW UR1INP ; UR1 input
DEFW UR2INP ; UR2 input

```

TRS-80 Model 4 BIOS Version 2.00+ General BIOS subroutines

EC64 E1
EC65 87
EC66 5F
EC67 16 00
EC69 19
EC6A 5E
EC6B 23
EC6C 56
EC6D EB
EC6E 5E
EC6F 23
EC70 56
EC71 EB
EC72 E9

```

;
; *****
; * I/O dispatch routine
; * Input: A=Device code (0-3)
; * (SP)=pointer to address table
; * Output: None - goes to device routine
; *****
IODSP POP HL ;Table pointer to HL
ADD A,A ;Compute offset
LD E,A ;Move offset to DE
LD D,0
ADD HL,DE ;Point to address
LD E,(HL) ;DE=vector pointer
INC HL
LD D,(HL)
EX DE,HL ;HL=vector pointer
LD E,(HL) ;Vector to DE
INC HL
LD D,(HL)
EX DE,HL ;HL=driver address
JP (HL) ;Exit to device driver

```

73 EC

```

;
; *****
; * Device Driver Address Table
; *****
DDATBL EQU $

```

; TTY definitions

EC73 28 F1
EC75 30 F1
EC77 3B F1
EC79 44 F1

```

; -----
TTYSTS DEFW SPSTS ;Serial port status
TTYINP DEFW SPINP ;Serial port input
TTYOUT DEFW SPOUT ;Serial port output
TTYBSY DEFW SPBSY ;Serial port busy

```

; CRT definitions

EC7B 51 ED
EC7D 61 ED
EC7F 46 EF
EC81 D0 EC

```

; -----
CRTSTS DEFW KBSTS ;Keyboard status
CRTINP DEFW KBINP ;Keyboard input
CRTOUT DEFW VDOUT ;Video output
CRTBSY DEFW NULBSY ;Null busy

```

; UC1 definitions

EC83 51 ED
EC85 61 ED
EC87 46 EF
EC89 D0 EC

```

; -----
UC1STS DEFW KBSTS ;Keyboard status
UC1INP DEFW KBINP ;Keyboard input
UC1OUT DEFW VDOUT ;Video output
UC1BSY DEFW NULBSY ;Null busy

```

```

; LPT definitions
; -----
EC8B  CB EC  LPTSTS  DEFW  NULSTS  ;Null status
EC8D  CD EC  LPTINP  DEFW  NULINP  ;Null input
EC8F  BF F0  LPTOUT  DEFW  PPOUT   ;Parallel port output
EC91  B3 F0  LPTBSY  DEFW  PPBSY   ;Parallel port busy

; UL1 definitions
; -----
EC93  CB EC  UL1STS  DEFW  NULSTS  ;Null status
EC95  CD EC  UL1INP  DEFW  NULINP  ;Null input
EC97  BF F0  UL1OUT  DEFW  PPOUT   ;Parallel port output
EC99  B3 F0  UL1BSY  DEFW  PPBSY   ;Parallel port busy

; PTR definitions
; -----
EC9B  51 ED  PTRSTS  DEFW  KBSTS   ;Keyboard status
EC9D  61 ED  PTRINP  DEFW  KBINP   ;Keyboard input
EC9F  CC EC  PTROUT  DEFW  NULOUT   ;Null output
ECA1  D0 EC  PTRBSY  DEFW  NULBSY   ;Null busy

; UR1 definitions
; -----
ECA3  28 F1  UR1STS  DEFW  SPSTS   ;Serial port status
ECA5  30 F1  UR1INP  DEFW  SPINP   ;Serial port input
ECA7  3B F1  UR1OUT  DEFW  SPOUT   ;Serial port output
ECA9  44 F1  UR1BSY  DEFW  SPBSY   ;Serial port busy

; UR2 definitions
; -----
ECAB  28 F1  UR2STS  DEFW  SPSTS   ;Serial port status
ECAD  30 F1  UR2INP  DEFW  SPINP   ;Serial port input
ECAF  3B F1  UR2OUT  DEFW  SPOUT   ;Serial port output
ECB1  44 F1  UR2BSY  DEFW  SPBSY   ;Serial port busy

; PTP definitions
; -----
ECB3  CB EC  PTPSTS  DEFW  NULSTS  ;Null status
ECB5  CD EC  PTPINP  DEFW  NULINP  ;Null input
ECB7  46 EF  PTPOUT  DEFW  VDOUT   ;Video output
ECB9  D0 EC  PTPBSY  DEFW  NULBSY   ;Null busy

; UP1 definitions
; -----
ECBB  28 F1  UP1STS  DEFW  SPSTS   ;Serial port status
ECBD  30 F1  UP1INP  DEFW  SPINP   ;Serial port input
ECBF  3B F1  UP1OUT  DEFW  SPOUT   ;Serial port output
ECC1  44 F1  UP1BSY  DEFW  SPBSY   ;Serial port busy

; UP2 definitions
; -----
ECC3  28 F1  UP2STS  DEFW  SPSTS   ;Serial port status
ECC5  30 F1  UP2INP  DEFW  SPINP   ;Serial port input
ECC7  3B F1  UP2OUT  DEFW  SPOUT   ;Serial port output
ECC9  44 F1  UP2BSY  DEFW  SPBSY   ;Serial port busy

; *****
; * Null device drivers
; *   Input:  None expected
; *   Output: None
; *****

ECCB  AF     NULSTS  XOR    A        ;Null status
ECCC  C9     NULOUT  RET                     ;Null output
ECCD  3E 1A  NULINP  LD     A,1AH    ;Null input
ECCF  C9     RET
ECD0  3E FF  NULBSY  LD     A,0FFH    ;Null busy

```

TRS-80 Model 4 BIOS Version 2.00+ Device driver for Keyboard

ED51 3A 9A EE
ED54 B7
ED55 20 07
ED57 CD 6F ED
ED5A C8
ED5B 32 9A EE
ED5E F6 FF
ED60 C9

```

; *****
; * Keyboard device drivers
; * Input: None
; * Output: Dependent on function
; *****
; Return keyboard status in A
; -----
KBSTS LD A,(KBDBUF) ;Check key buffer
      OR A
      JR NZ,KBSTS1 ;Go if key there
      CALL KBSCAN ;Scan the keyboard
      RET Z ;Exit if no key
      LD (KBDBUF),A ;Save the key found
KBSTS1 OR 0FFH ;Set status
      RET

```

ED61 21 9A EE
ED64 7E
ED65 36 00
ED67 B7
ED68 C0
ED69 CD 6F ED
ED6C 28 FB
ED6E C9

```

; Input from keyboard & return key in A
; -----
KBINP LD HL,KBDBUF ;Point to key buffer
      LD A,(HL) ;Empty it
      LD (HL),0
      OR A ;Check for key
      RET NZ ;Exit if found
KBINP1 CALL KBSCAN ;Scan the keyboard
      JR Z,KBINP1 ;Loop if no key
      RET

```

ED6F F3
ED70 3E 8E
ED72 D3 84
ED74 CD 24 EE
ED77 C2 1C EE
ED7A 11 01 F4
ED7D 06 00
ED7F 21 9D EE
ED82 1A
ED83 4F
ED84 AE
ED85 71
ED86 A1
ED87 20 3D
ED89 04
ED8A 23
ED8B CB 03
ED8D F2 82 ED
ED90 3A A5 EE
ED93 5F
ED94 1A
ED95 4F
ED96 2A A6 EE
ED99 7E
ED9A A1
ED9B 20 0D
ED9D ED 62
ED9F 22 AA EE
EDA2 21 00 08
EDA5 22 A8 EE
EDA8 18 72
EDAA AF

```

; General keyboard scan - key returned in A if found
; -----
KBSCAN DI ;No interrupts!
      LD A,KVMIN ;Switch Keyboard into RAM
      OUT (MEMCTL),A
      CALL KBFKC ;Check function keys
      JP NZ,KBSCNX ;Go if key found
KBSCN1 LD DE,0F401H ;Point to first row
      LD B,0 ;Initialize row #
      LD HL,KBHIST ;Point DE at history table
KBSCN2 LD A,(DE) ;Strobe the keyboard
      LD C,A ;Save strobe in C
      XOR (DE) ;Mask off prior keys
      LD (DE),C ;Save current scan
      AND C ;Mask released keys
      JR NZ,KBSCN4 ;Go if any key pressed
      INC B ;Update row #
      INC HL ;Update history pointer
      RLC E ;Move to next key row
      JP P,KBSCN2 ;Loop if any rows left
      LD A,(KBDPKR) ;Point DE at Prv Key Row
      LD E,A
      LD A,(DE) ;Scan the row again
      LD C,A ;Save the scan
      LD HL,(KBDPKI) ;Point HL at Prv Key Image
      LD A,(HL) ;Get previous image
      AND C ;Key still down?
      JR NZ,KBSCN3 ;Go if yes
      SBC HL,HL ;Clear Repeat Counter
      LD (KBDRPT),HL
      LD HL,0800H ;Reset Delay Counter
      LD (KBDDLY),HL
      JR KBSCNX ;Exit with no key
KBSCN3 XOR A ;Clear carry & A

```

EDAB	EB		LD	HL,(KBDRPT)	;History pointer to DE
EDAC	2A AA EE		INC	HL	;Repeat counter to HL
EDAF	23		LD	(KBDRPT),HL	;Increment the count
EDB0	22 AA EE		LD	BC,(KBDDLY)	;Save the counter
EDB3	ED 4B A8 EE		SBC	HL,BC	;Get the delay value
EDB7	ED 42		JR	C,KBSCNX	;Delay long enough?
EDB9	38 61		LD	(DE),A	;Exit if no time-out
EDBB	12		LD	(KBDRPT),HL	;Clear history for rescan
EDBC	22 AA EE		LD	L,80H	;Save zeroed counter
EDBF	2E 80		LD	(KBDDLY),HL	;Set short delay
EDC1	22 A8 EE		JR	KBSCN1	;Go scan again
EDC4	18 B4		LD	C,A	;True scan to C
EDC6	4F	KBSCN4	CALL	KBDBN	;Do debounce delay
EDC7	CD 84 EE		JR	Z,KBSCNX	;Exit if no key
EDCA	28 50		LD	A,E	;Save row bit
EDCC	7B		LD	(KBDPKR),A	
EDCD	32 A5 EE		LD	(KBDPKI),HL	;Save image pointer
EDD0	22 A6 EE		SLA	B	;Multiply row # by 8
EDD3	CB 20		SLA	B	
EDD5	CB 20		SLA	B	
EDD7	CB 20		DEC	B	;Precomp for shift
EDD9	05		INC	B	;Update char position
EDDA	04	KBSCN5	SRL	C	;Shift strobe bit left 1
EDDB	CB 39		JR	NC,KBSCN5	;Loop till it falls off
EDDD	30 FB		LD	HL,KBDHST+7	;Point HL at control image
EDDF	21 A4 EE		LD	A,B	;Get table offset
EDE2	78		CP	32	;In alpha keys?
EDE3	FE 20		JR	NC,KBSCN6	;Go if not
EDE5	30 1B		BIT	2,(HL)	;Control pressed?
EDE7	CB 56		JR	NZ,KBSCNX	;Exit if yes
EDE9	20 31		SET	6,B	;Convert offset to ASCII
EDEB	CB F0		OR	A	;Is this '@' key?
EDED	B7		JR	Z,KBSCN9	;Exit if yes
EDEE	28 2B		LD	A,(KBDCLF)	;Get CAPS Lock Flag
EDF0	3A AC EE		OR	A	;CAPS locked?
EDF3	B7		JR	NZ,KBSCN9	;Go if yes
EDF4	20 25		SET	5,B	;Make lower case
EDF6	CB E8		LD	A,3	;Check SHIFT keys
EDF8	3E 03		AND	(HL)	;Is either one down?
EDFA	A6		JR	Z,KBSCN9	;Go if not
EDFB	28 1E		LD	A,20H	;Invert bit 5
EDFD	3E 20		XOR	B	
EDFF	A8		JR	KBSCNX	;Exit with key
EE00	18 1A		SUB	32	;Calculate offset
EE02	D6 20	KBSCN6	LD	C,A	;Put in BC
EE04	4F		LD	B,0	
EE05	06 00		LD	DE,KBDCOD	;Decode table base to DE
EE07	11 AD EE		EX	DE,HL	;Move to HL, KBDHST to DE
EE0A	EB		ADD	HL,BC	;Point to standard table
EE0B	09		LD	BC,24	;Table length to BC
EE0C	01 18 00		LD	A,(DE)	;Isolate CTRL,SHIFT keys
EE0F	1A		AND	07H	
EE10	E6 07		JR	Z,KBSCN8	;Go if neither down
EE12	28 06		ADD	HL,BC	;Move to SHIFT table
EE14	09		AND	4	;Isolate CTRL key
EE15	E6 04		JR	Z,KBSCN8	;Go if only SHIFT
EE17	28 01		ADD	HL,BC	;Move to CTRL table
EE19	09		LD	B,(HL)	;Get decoded key in B
EE1A	46	KBSCN8	LD	A,B	;Return key to A
EE1B	78	KBSCN9	LD	C,A	;Store character in C
EE1C	4F	KBSCNX	LD	A,KVMOUT	;Switch out keyboard
EE1D	3E 8F		OUT	(MEMCTL),A	
EE1F	D3 84		LD	A,C	;Restore the key, if any
EE21	79		OR	A	;Set Z if no key found
EE22	B7		RET		
EE23	C9				

```

EE24 2A 9B EE
EE27 7C
EE28 B5
EE29 20 4C
EE2B 11 7F F4
EE2E 4B
EE2F 1A
EE30 B7
EE31 28 02
EE33 0E 07
EE35 21 A4 EE
EE38 1C
EE39 1A
EE3A A1
EE3B 4F
EE3C AE
EE3D 71
EE3E A1
EE3F C8
EE40 4F
EE41 CD 84 EE
EE44 C8
EE45 CB 59
EE47 28 0F
EE49 3A AC EE
EE4C EE 01
EE4E 32 AC EE
EE51 C5
EE52 0E 28
EE54 C4 EE EF
EE57 C1
EE58 3E 70
EE5A A1
EE5B C8
EE5C 07
EE5D EB
EE5E 21 10 EF
EE61 01 09 00
EE64 ED 42
EE66 07
EE67 30 FB
EE69 0E 1B
EE6B 1A
EE6C E6 03
EE6E 20 06
EE70 1A
EE71 E6 04
EE73 28 02
EE75 09
EE76 09
EE77 7E
EE78 23
EE79 B7
EE7A 22 9B EE
EE7D C0
EE7E 67
EE7F 6F
EE80 22 9B EE
EE83 C9

```

; Scan function keys

```

KBFKC LD HL,(KBDFKP)
LD A,H
OR L
JR NZ,KBFKC5
LD DE,0F47FH
LD C,E
LD A,(DE)
OR A
JR Z,KBFKC1
LD C,07H
KBFKC1 LD HL,KBDHST+7
INC E
LD A,(DE)
AND C
LD C,A
XOR (HL)
LD (HL),C
AND C
RET Z
LD C,A
CALL KBDBN
RET Z
BIT 3,C
JR Z,KBFKC2
LD A,(KBDCLF)
XOR 01H
LD (KBDCLF),A
PUSH BC
LD C,40
CALL NZ,VDBEL1
POP BC
KBFKC2 LD A,70H
AND C
RET Z
RLCA
EX DE,HL
LD HL,KBDCOD+99
LD BC,9
KBFKC3 SBC HL,BC
RLCA
JR NC,KBFKC3
LD C,27
LD A,(DE)
AND 03H
JR NZ,KBFKC4
LD A,(DE)
AND 4
JR Z,KBFKC5
ADD HL,BC
KBFKC4 ADD HL,BC
KBFKC5 LD A,(HL)
INC HL
OR A
LD (KBDFKP),HL
RET NZ
LD H,A
LD L,A
LD (KBDFKP),HL
RET

```

```

;Get Function Key Pointer
;Is a key active?
;Go if yes
;Set DE for rows 0-6
;Preset key mask
;Strobe rows 0-6
;Anything down?
;Go if not
;Must ignore F1,F2,F3,CAPS
;Point HL at row 7 image
;Set DE for row 7
;Strobe row 7
;Mask off if necessary
;Result in C
;Set changed bits
;Save current scan
;Mask released keys
;Exit if no key down
;Corrected scan to C
;Do debounce delay
;Exit if no key down
;CAPS key down?
;Go if not
;Toggle the flag
;Save registers
;Set counter for short beep
;Beep if locking
;Check function keys
;Exit if none down
;Prepare to position
;History pointer to DE
;Point HL at Decode table
;Set BC to 1 entry length
;Back up table pointer
;Check next F key bit
;Loop until found
;Preload for next round
;Get key scan from KBDHST
;Check the SHIFT keys
;Go if either down
;Get key scan again
;Check the CTRL key
;Go if not pressed
;Move down one group
;Move down one group
;Get next keystroke
;Update pointer
;End of definition?
;Save def pointer
;Exit if valid key
;Clear the pointer
;Exit with key

```

; Debounce a key

```

EE84 3E 0F
EE86 CD 14 ED

```

```

KBDBN LD A,15
CALL MSDELY

```

```

;Set time (app. 15ms)
;Do the delay

```



```

EE89 1A LD A,(DE) ;Scan keyboard again
EE8A A1 AND C ;Mask off released keys
EE8B C9 RET

;
; Initialize Keyboard DCB
; -----
EE8C CD 23 ED KBINIT CALL CLRMEM ;Clear DCB fields
EE8F 9A EE 12 00 DEFW KBDCB,KBDCLEF-KBDCB
EE93 21 00 08 LD HL,0800H ;Reset repeat counter
EE96 22 A8 EE LD (KBDDLY),HL
EE99 C9 RET

;
; Keyboard Device Control Block
; -----
EE9A 9A EE KBDCB EQU $
EE9B 00 00 KBDBUF DEFB 0 ;Character buffer
EE9D 00 00 00 00 KBDFKP DEFW 0 ;Function Key Pointer
EEA1 00 00 00 00 KBDBST DEFB 0,0,0,0,0,0,0,0 ;History for 8 rows
EEA5 00 KBDBPKR DEFB 0 ;Previous Key Row bit
EEA6 00 00 KBDBPKI DEFW 0 ;Previous Key Image pointer
EEA8 00 08 KBDDLY DEFW 0800H ;Delay before repeating
EEAA 00 00 KBDBRPT DEFW 0 ;Delay between repeats
EEAC 01 00 KBDBCLF DEFB 1 ;CAPS Lock Flag
AD EE KBDBCOD EQU $ ;Keyboard Decode table

;
; Unshifted keys
EEAD 30 31 32 33 DEFB '01234567' ;0 1 2 3 4 5 6 7
EEB1 34 35 36 37
EEB5 38 39 3A 3B DEFB '89:;,-./' ;8 9 : ; , - . /
EEB9 2C 2D 2E 2F
EEBD 0D 18 03 0B DEFB 0DH,18H,03H,0BH ;ENTER CLEAR BREAK UP
EEC1 0A 08 09 20 DEFB 0AH,08H,09H,20H ;DOWN LEFT RIGHT SPACE

;
; Shifted keys
EEC5 30 21 22 23 DEFB '0!"#$%&' ;0 1 2 3 4 5 6 7
EEC9 24 25 26 27
EECD 28 29 2A 2B DEFB '()*+<=>?' ;8 9 : ; , - . /
EED1 3C 3D 3E 3F
EED5 0D 1B 03 0B DEFB 0DH,1BH,03H,0BH ;ENTER CLEAR BREAK UP
EED9 0A 08 09 20 DEFB 0AH,08H,09H,20H ;DOWN LEFT RIGHT SPACE

;
; Control keys
EEDD 30 7C 32 7E DEFB '0>241/6d' ;0 1 2 3 4 5 6 7
EEE1 34 5E 36 60
EEE5 5B 5D 3A 3B DEFB '[';e_4e' ;8 9 : ; , - . /
EEE9 7B 5F 7D 5C
EEED 0D 7F 03 0B DEFB 0DH,7FH,03H,0BH ;ENTER CLEAR BREAK UP
EEF1 0A 08 09 20 DEFB 0AH,08H,09H,20H ;DOWN LEFT RIGHT SPACE

;
; Function Key Definition table (9 bytes per entry)
; -----
EEF5 46 31 20 20 KBFKD DEFB 'F1' ',0'
EEF9 20 20 20 20
EEFD 00
EEFE 46 32 20 20 DEFB 'F2' ',0'
EF02 20 20 20 20
EF06 00
EF07 46 33 20 20 DEFB 'F3' ',0'
EF0B 20 20 20 20
EF0F 00

;
EF10 53 48 49 46 DEFB 'SHIFT/F1',0
EF14 54 2F 46 31
EF18 00
EF19 53 48 49 46 DEFB 'SHIFT/F2',0
EF1D 54 2F 46 32

```

```

EF21 00
EF22 53 48 49 46
EF26 54 2F 46 33
EF2A 00

EF2B 43 54 52 4C
EF2F 2F 46 31 20
EF33 00
EF34 43 54 52 4C
EF38 2F 46 32 20
EF3C 00
EF3D 43 54 52 4C
EF41 2F 46 33 20
EF45 00

```

```

DEFB 'SHIFT/F3',0

```

```

DEFB 'CTRL/F1 ',0

```

```

DEFB 'CTRL/F2 ',0

```

```

DEFB 'CTRL/F3 ',0

```

TRS-80 Model 4 BIOS Version 2.00+ Device driver for Video Display

```

; *****
; * Video Display drivers *
; * Input: Dependent on function *
; * Output: None returned to caller *
; *****

```

```

; Output character in C to Video Display
; -----

```

```

EF46 F3
EF47 3E 8E
EF49 D3 84
EF4B 3A 8D F0
EF4E CD 6F EF
EF51 CD 8B EF
EF54 CD 6A EF
EF57 32 8D F0
EF5A CB 7F
EF5C 28 02
EF5E 3E 9B
EF60 F6 80
EF62 CD 6F EF
EF65 3E 8F
EF67 D3 84
EF69 C9

```

```

VDOUT DI ;No interrupts
LD A,KVMIN ;Switch Video into RAM
OUT (MEMCTL),A
LD A,(VDDCHR) ;Get character at cursor
CALL VDPUT ;Replace it in Video RAM
CALL VDPROC ;Process input character
CALL VDGET ;Get character at cursor
LD (VDDCHR),A ;Save in DCB
BIT 7,A ;Is character inverted?
JR Z,VDOUT1 ;Go if not
LD A,9BH ;Set in alternate cursor
VDOUT1 OR 80H ;Insure reverse video
CALL VDPUT ;Output cursor
LD A,KVMOUT ;Switch out Video
OUT (MEMCTL),A
RET

```

```

; Get a character from Video RAM at cursor
; -----

```

```

EF6A CD 74 EF
EF6D 7E
EF6E C9

```

```

VDGET CALL VDCSR ;Point HL at cursor position
LD A,(HL) ;Get the character
RET

```

```

; Put a character into Video RAM at cursor
; -----

```

```

EF6F CD 74 EF
EF72 77
EF73 C9

```

```

VDPUT CALL VDCSR ;Point HL at cursor position
LD (HL),A ;Output the character
RET

```

```

; Point HL at cursor position in Video RAM
; -----

```

```

EF74 2A 8E F0

```

```

VDCSR LD HL,(VDDROW) ;Get cursor column & row

```

```

; Compute RAM Address for position (L=Row, H=Col)
; -----

```

```

EF77 C5
EF78 D5
EF79 01 00 F8
EF7C 51
EF7D 5D
EF7E 4C

```

```

VDCRA PUSH BC ;Save work registers
PUSH DE
LD BC,0F800H ;Video RAM base to BC
LD D,C ;Row # to DE
LD E,L
LD C,H ;Column # to C

```

EF7F	62	LD	H,D	;Row # also in HL
EF80	29	ADD	HL,HL	;HL=Row # * 4
EF81	29	ADD	HL,HL	
EF82	19	ADD	HL,DE	;HL=Row # * 5 (4+1)
EF83	29	ADD	HL,HL	;HL=Row # * 80 (80=5*16)
EF84	29	ADD	HL,HL	
EF85	29	ADD	HL,HL	
EF86	29	ADD	HL,HL	
EF87	09	ADD	HL,BC	;Add video base, Column #
EF88	D1	POP	DE	;Restore registers
EF89	C1	POP	BC	
EF8A	C9	RET		

; Process Video output characters

EF8B	3A 91 F0	VDPROC	LD	A,(VDDESC)	;Get ESC sequence control
EF8E	B7		OR	A	;In ESC sequence?
EF8F	20 12		JR	NZ,VDESH	;Go if yes
EF91	79		LD	A,C	;Get the character
EF92	FE 20		CP	20H	;Control code?
EF94	38 43		JR	C,VDCTL	;Go if yes
EF96	3A 90 F0		LD	A,(VDDINV)	;Get inverse video mask
EF99	B1		OR	C	;Combine with character
EF9A	CD 6F EF		CALL	VDPUT	;Output to Video Display
EF9D	2A 8E F0		LD	HL,(VDDROW)	;Cursor Column,Row to HL
EFA0	C3 24 F0		JP	VDCRT	;Cursor right & exit

; Video Display ESC Sequence Handler

EFA3	FE 02	VDESH	CP	2	;Check state of ESC
EFA5	38 0C		JR	C,VDESH1	;Go if state 1
EFA7	28 1E		JR	Z,VDESH2	;Go if state 2
EFA9	79		LD	A,C	;Get input character
EFAA	FE 3D		CP	'='	;Must be '='
EFAC	3E 02		LD	A,2	;Set next state in A
EFAE	28 25		JR	Z,VDESHX	;Go if valid
EFB0	AF		XOR	A	;Clear state variable
EFB1	18 22		JR	VDESHX	; and exit
EFB3	3A 92 F0	VDESH1	LD	A,(VDDEX)	;Get saved Row
EFB6	6F		LD	L,A	;Put it in L
EFB7	79		LD	A,C	;Get input Column
EFB8	D6 20		SUB	20H	;Convert to actual
EFBA	FE 50		CP	80	;Is column # valid?
EFBC	38 02		JR	C,\$+4	;Skip next if so
EFBE	3E 4F		LD	A,79	;Move to last column
EFC0	67		LD	H,A	;Put it in H
EFC1	22 8E F0		LD	(VDDROW),HL	;Store as new cursor
EFC4	AF		XOR	A	;Clear state variable
EFC5	18 0E		JR	VDESHX	; and exit
EFC7	79	VDESH2	LD	A,C	;Get the input character
EFC8	D6 20		SUB	20H	;Convert to actual Row
EFCA	FE 18		CP	24	;Is it valid?
EFCC	38 02		JR	C,\$+4	;Skip next if it is
EFCE	3E 17		LD	A,23	;Move to last row
EFD0	32 92 F0		LD	(VDDEX),A	;Store in DCB
EFD3	3E 01		LD	A,1	;Set next state in A
EFD5	32 91 F0	VDESHX	LD	(VDDESC),A	;Save state variable
EFD8	C9		RET		; and exit

; Video Display Control Code processing

EFD9	21 93 F0	VDCTL	LD	HL,VDCXAT	;HL=Code Address Table
EFDC	06 00		LD	B,0	;Table offset in BC
EFDE	09		ADD	HL,BC	;Index to routine offset
EFDF	7E		LD	A,(HL)	;Pick up routine offset
EFE0	B7		OR	A	;Is the code defined?

```

EFE1  C8
EFE2  21 E2 EF
EFE5  4F
EFE6  09
EFE7  E5
EFE8  2A 8E F0
EFEB  C9

```

```

RET      Z
VDCTL1  LD      HL,VDCTL1
LD      C,A
ADD      HL,BC
PUSH     HL
LD      HL,(VDDROW)
RET

```

```

;Ignore if not
;Point HL at base address
;Add offset for this code

;Routine address to stack
;Cursor Column,Row to HL
;Go to it

```

```

;
; Sound the built-in speaker
;

```

```

EFEC  0E 00
EFEE  3E 01
EFF0  06 64
EFF2  D3 90
EFF4  10 FC
EFF6  AF
EFF7  06 64
EFF9  D3 90
EFFB  10 FC
EFFD  0D
EFEE  20 EE
F000  C9

```

```

VDBEL  LD      C,0
VDBEL1 LD      A,1
LD      B,100
VDBEL2 OUT     (SOUND),A
DJNZ    VDBEL2
XOR     A
LD      B,100
VDBEL3 OUT     (SOUND),A
DJNZ    VDBEL3
DEC     C
JR      NZ,VDBEL1
RET

```

```

;Set duration counter
;Set bit 0 on
;Set pitch counter
;Crank up a wave

;Turn bit 0 off
;Reset pitch counter
;Let the wave die

;Count down duration
;Loop until timeout

```

```

;
; Move the cursor left 1 position
;

```

```

F001  7C
F002  B5
F003  28 2D
F005  25
F006  F2 32 F0
F009  26 4F
F00B  18 0F

```

```

VDCLT  LD      A,H
OR      L
JR      Z,VDCSCK
DEC     H
JP      P,VDCSCK
LD      H,79
JR      VDTV

```

```

;At top of screen?
;Go if yes
;Back up 1 position
;Exit if no wrap
;Move to end of line
; and back up 1 row

```

```

;
; Move cursor to next tab stop
;

```

```

F00D  7C
F00E  E6 F8
F010  C6 08
F012  67
F013  FE 50
F015  38 1B
F017  26 00

```

```

VDTAB  LD      A,H
AND     0F8H
ADD     A,8
LD      H,A
CP      80
JR      C,VDCSCK
LD      H,0

```

```

;Column # to A
;Make it 0 mod 8
;Move to next tab stop

;Line overflow?
;Exit if not
;Move down 1 line

```

```

;
; Move cursor down 1 line
;

```

```

F019  2C
F01A  18 16

```

```

VDLF  INC      L
JR      VDCSCK

```

```

;Increment the row #

```

```

;
; Move cursor up 1 line
;

```

```

F01C  2D
F01D  F2 32 F0
F020  2E 00
F022  18 0E

```

```

VDVT  DEC      L
JP      P,VDCSCK
LD      L,0
JR      VDCSCK

```

```

;Back up 1 row
;Go if not negative
;Hold on top line

```

```

;
; Move cursor right 1 position
;

```

```

F024  24
F025  7C
F026  FE 50
F028  38 08
F02A  26 00
F02C  18 EB

```

```

VDCRT  INC      H
LD      A,H
CP      80
JR      C,VDCSCK
LD      H,0
JR      VDLF

```

```

;Advance 1 column
;Get the new column
;Still on line?
;Go if yes
;Move to next line

```

```

; Perform Cursor Home
; -----
F02E 2E 00 VDHOM LD L,0 ;Set row # to 0
;
; Perform Carriage Return
; -----
F030 26 00 VDCR LD H,0 ;Set column # to 0
;
; Check cursor position & scroll if necessary
; -----
F032 7D VDCSCK LD A,L ;Get the cursor Row #
F033 FE 18 CP 24 ;Is it on-screen?
F035 22 8E F0 LD (VDDR0W),HL ;Save the cursor
F038 D8 RET C ;Exit if on screen
F039 2E 17 LD L,23 ;Stay on line 23
F03B 22 8E F0 LD (VDDR0W),HL ;Save the cursor
F03E 21 50 F8 LD HL,0F800H+80 ;Point HL at second line
F041 11 00 F8 LD DE,0F800H ;Point DE at top of screen
F044 01 30 07 LD BC,80*23 ;Move 23 lines of video
F047 ED B0 LDIR ;Scroll Video RAM
F049 21 17 00 LD HL,23 ;Set Row=23, Column=0
F04C 18 10 JR VDEOS ;Clear new line & exit
;
; Erase to end of current line
; -----
F04E E5 VDEOL PUSH HL ;Save cursor position
F04F 26 50 LD H,80 ;Set to end of line + 1
F051 CD 77 EF CALL VDCRA ;Calculate RAM address
F054 EB EX DE,HL ;Put in DE
F055 E1 POP HL ;Restore cursor
F056 18 09 JR VDEOS1 ;Go clear
;
; Home the cursor and clear the screen
; -----
F058 21 00 00 VDCLS LD HL,0000H ;Set cursor at 0,0
F05B 22 8E F0 LD (VDDR0W),HL ;Save it in DCB
;
; Erase to end of screen
; -----
F05E 11 80 FF VDEOS LD DE,0FF80H ;Set end address
F061 CD 77 EF VDEOS1 CALL VDCRA ;Calc start address
F064 EB EX DE,HL ;Start to DE, end to HL
F065 3A 90 F0 LD A,(VDDINV) ;Get inverse video mask
F068 F6 20 OR 20H ;Create a blank
F06A ED 52 SBC HL,DE ;Compute clear length
F06C 44 LD B,H ;Move length to BC
F06D 4D LD C,L
F06E C3 2E ED JP MFILL ;Fill memory & exit
;
; Turn inverse video OFF
; -----
F071 AF VDIV0 XOR A ;Clear the flag
F072 18 08 JR VDIV2 ;Go store it
;
; Turn inverse video ON
; -----
F074 AF VDIV1 XOR A ;Clear the flag
F075 18 03 JR VDIV1 ;Toggle & store it
;
; Toggle state of inverse video
; -----
F077 3A 90 F0 VDIV LD A,(VDDINV) ;Get inverse video mask
F07A EE 80 VDIV1 XOR 80H ;Reverse it
F07C 32 90 F0 VDIV2 LD (VDDINV),A ;Replace in DCB
F07F C9 RET

```

F080 3E 03
F082 C3 D5 EF

VDESC LD A,3 ;Set ESC state variable
JP VDESHX ; & exit

; Initialize Video Display DCB fields

F085 CD 23 ED
F088 8E F0 05 00
F08C C9

VDINIT CALL CLRMEM ;Clear DCB fields
DEFW VDDROW,VDCXAT-VDDROW
RET

; Video Display Device Control Block

8D F0

F08D 20
F08E 00
F08F 00
F090 00
F091 00
F092 00

93 F0

F093 00
F094 00
F095 00
F096 00
F097 00
F098 00
F099 00
F09A 0A
F09B 1F
F09C 2B
F09D 37
F09E 3A
F09F 42
F0A0 4E
F0A1 8F
F0A2 92
F0A3 00
F0A4 00
F0A5 00
F0A6 00
F0A7 00
F0A8 6C
F0A9 95
F0AA 00
F0AB 00
F0AC 7C
F0AD 76
F0AE 9E
F0AF 00
F0B0 00
F0B1 4C
F0B2 00

VDDCB EQU \$
VDDCHR DEFB ' ' ;Character under cursor
VDDROW DEFB 0 ;Cursor row (0-23)
VDDCOL DEFB 0 ;Cursor column (0-79)
VDDINV DEFB 0 ;Inverse video mask
VDDESC DEFB 0 ;Escape Sequence Control
VDDSEX DEFB 0 ;Escape Sequence Storage
VDCXAT EQU \$;Control code address table
DEFB 00H ;Code 00 - ignored
DEFB 01H ;Code 01 - ignored
DEFB 02H ;Code 02 - ignored
DEFB 03H ;Code 03 - ignored
DEFB 04H ;Code 04 - ignored
DEFB 05H ;Code 05 - ignored
DEFB 06H ;Code 06 - ignored
DEFB VDBEL-VDCTL1 ;Code 07 - Sound bell
DEFB VDCLT-VDCTL1 ;Code 08 - Backspace
DEFB VDTAB-VDCTL1 ;Code 09 - Tab
DEFB VDLF-VDCTL1 ;Code 0A - Linefeed
DEFB VDDVT-VDCTL1 ;Code 0B - Vertical tab
DEFB VDCRT-VDCTL1 ;Code 0C - Cursor right
DEFB VDCR-VDCTL1 ;Code 0D - Carriage return
DEFB VDIV0-VDCTL1 ;Code 0E - Inverse video OFF
DEFB VDIV1-VDCTL1 ;Code 0F - Inverse video ON
DEFB 10H ;Code 10 - ignored
DEFB 11H ;Code 11 - ignored
DEFB 12H ;Code 12 - ignored
DEFB 13H ;Code 13 - ignored
DEFB 14H ;Code 14 - ignored
DEFB VDEOL-VDCTL1 ;Code 15 - Erase to EOL
DEFB VDINV-VDCTL1 ;Code 16 - Toggle inverse
DEFB 17H ;Code 17 - ignored
DEFB 18H ;Code 18 - ignored
DEFB VDEOS-VDCTL1 ;Code 19 - Erase to EOS
DEFB VDCLS-VDCTL1 ;Code 1A - Home & clear
DEFB VDESC-VDCTL1 ;Code 1B - Start ESC
DEFB 1CH ;Code 1C - ignored
DEFB 1DH ;Code 1D - ignored
DEFB VDHOM-VDCTL1 ;Code 1E - Home cursor
DEFB 1FH ;Code 1F - ignored

TRS-80 Model 4 BIOS Version 2.00+ Parallel Printer Port device driver

* Parallel Port device drivers *
* Input: Dependent on function *
* Output: Dependent on function *

; Check port for busy &/or error - return in A

F0B3 DB F8
F0B5 E6 F0

PPBSY IN A,(PARSDT) ;Read port status
AND 0F0H ;Isolate status bits

F0B7	EE 30		XOR	30H	;Invert negative logic bits
F0B9	28 02		JR	Z,PPBSY1	;Go if ready
F0BB	3E 01		LD	A,1	;Preset for zero return
F0BD	3D	PPBSY1	DEC	A	;Set A to 00H or FFH
F0BE	C9		RET		
; Output C to parallel port					

F0BF	21 25 F1	PPOUT	LD	HL,PPDOPT	;Point at option bits
F0C2	79		LD	A,C	;Get character to print
F0C3	FE 20		CP	20H	;Is it a control code?
F0C5	30 2D		JR	NC,PRINT	;Go if not
F0C7	FE 0A		CP	0AH	;Is it linefeed?
F0C9	20 0D		JR	NZ,PPOUT1	;Go if not
F0CB	CB 46		BIT	0,(HL)	;Suppress linefeeds?
F0CD	28 25		JR	Z,PRINT	;Go if not
F0CF	3A 24 F1		LD	A,(PPDPRV)	;Check previous character
F0D2	FE 0D		CP	0DH	;Was it carriage return?
F0D4	28 26		JR	Z,PPCLF	;Exit if so
F0D6	18 1C		JR	PRINT	;Go print the linefeed
F0D8	FE 0C	PPOUT1	CP	0CH	;Is this a formfeed?
F0DA	20 18		JR	NZ,PRINT	;Go if not
F0DC	CB 4E		BIT	1,(HL)	;Simulate formfeeds?
F0DE	28 14		JR	Z,PRINT	;Go if not
F0E0	3A 26 F1		LD	A,(PPDLCT)	;Get line counter
F0E3	B7		OR	A	;Anything left on page?
F0E4	28 22		JR	Z,PPRLC	;Exit if not
F0E6	47		LD	B,A	;Set up loop counter
F0E7	CD B3 F0	PPOUT2	CALL	PPBSY	;Wait for printer ready
F0EA	28 FB		JR	Z,PPOUT2	
F0EC	3E 0A		LD	A,0AH	;Output a linefeed
F0EE	D3 F8		OUT	(PARSDT),A	
F0F0	10 F5		DJNZ	PPOUT2	;Loop through the page
F0F2	18 1A		JR	PPRLC1	;Exit
; Print the character in C					

F0F4	CD B3 F0	PRINT	CALL	PPBSY	;Wait for printer ready
F0F7	28 FB		JR	Z,PRINT	
F0F9	79		LD	A,C	;Print the character
F0FA	D3 F8		OUT	(PARSDT),A	
; Check for linefeed, count down if so					

F0FC	3E 0A	PPCLF	LD	A,0AH	;Set A to linefeed
F0FE	B9		CP	C	;Did we just do one?
F0FF	20 07		JR	NZ,PPRLC	;Exit if not
F101	3A 26 F1		LD	A,(PPDLCT)	;Decrement line counter
F104	3D		DEC	A	
F105	32 26 F1		LD	(PPDLCT),A	
; Reset line counter if zero, exit					

F108	3A 26 F1	PPRLC	LD	A,(PPDLCT)	;Get line counter
F10B	B7		OR	A	;Is it zero?
F10C	20 06		JR	NZ,PPOUTX	;Exit if not
F10E	3A 27 F1	PPRLC1	LD	A,(PPDPGL)	;Reset line counter
F111	32 26 F1		LD	(PPDLCT),A	
; Save character and exit					

F114	79	PPOUTX	LD	A,C	;Save character in DCB
F115	32 24 F1		LD	(PPDPRV),A	
F118	C9		RET		

F119	3A 27 F1	PPINIT	LD	A,(PPDPGL)	;Reset line counter
F11C	32 26 F1		LD	(PPDLCT),A	
F11F	AF		XOR	A	;Kill previous character
F120	32 24 F1		LD	(PPDPRV),A	
F123	C9		RET		

; Parallel port DCB

	24 F1	PPDCB	EQU	\$	
F124	00	PPDPRV	DEFB	0	;Previous character
F125	01	PPDOPT	DEFB	1	;Option bits
					; 0=Suppress LF after CR
					; 1=Simulate formfeeds
					; 2-7=Reserved
F126	00	PPDLCT	DEFB	0	;Line counter
F127	42	PPDPGL	DEFB	66	;Page length

TRS-80 Model 4 BIOS Version 2.00+ Serial Port device driver

```

; *****
; * Serial Port device drivers *
; * Input: Dependent on function *
; * Output: Dependent on function *
; *****

```

; Check for input at Serial Port, return status in A

F128	DB EA	SPSTS	IN	A,(SERURT)	;Get UART status
F12A	E6 80		AND	80H	;Isolate data received bit
F12C	C8		RET	Z	;Exit if nothing
F12D	F6 FF		OR	0FFH	;Set status to show input
F12F	C9		RET		

; Input a byte from the Serial Port

F130	DB EA	SPINP	IN	A,(SERURT)	;Get UART status
F132	E6 80		AND	80H	;Anything received?
F134	28 FA		JR	Z,SPINP	;Loop if not
F136	DB EB		IN	A,(SERDAT)	;Read data byte
F138	E6 7F		AND	7FH	;Mask off parity bit
F13A	C9		RET		

; Output a byte to the Serial Port

F13B	CD 44 F1	SPOUT	CALL	SPBSY	;Is the port busy?
F13E	28 FB		JR	Z,SPOUT	;Loop until ready
F140	79		LD	A,C	;Get output byte
F141	D3 EB		OUT	(SERDAT),A	;Output it
F143	C9		RET		

; Check Serial Port for busy

F144	DB EA	SPBSY	IN	A,(SERURT)	;Get UART status
F146	E6 40		AND	40H	;Ready to Xmit?
F148	C8		RET	Z	;Exit if not
F149	21 75 F1		LD	HL,SPDOPT	;Point to options byte
F14C	CB 46		BIT	0,(HL)	;Wait for CTS enabled?
F14E	28 07		JR	Z,SPBSY1	;Go if not
F150	DB E8		IN	A,(SERRST)	;Get secondary status
F152	E6 80		AND	80H	;Check CTS input bit
F154	EE 80		XOR	80H	;Invert state of CTS *
					; Above changed to NOP in 2.22
F156	C8		RET	Z	;Exit if no CTS
F157	CB 4E	SPBSY1	BIT	1,(HL)	;Wait for DSR enabled?

F159	28 07	JR	Z,SPBSY2	;Go if not
F15B	DB E8	IN	A,(SERRST)	;Get secondary status
F15D	E6 40	AND	40H	;Isolate DSR bit
F15F	EE 40	XOR	40H	;Invert state of DSR *
				; Above changed to NOP in 2.22
F161	C8	RET	Z	;Exit if no DSR
F162	F6 FF	SPBSY2 OR	0FFH	;Indicate ready state
F164	C9	RET		

```

;
; Initialize Serial Port
; -----

```

F165	3A 76 F1	SPINIT LD	A,(SPDBDR)	;Set the baud rate
F168	D3 E9	OUT	(SERBRG),A	
F16A	D3 E8	OUT	(SERRST),A	;Reset the UART
F16C	3A 77 F1	LD	A,(SPDCFG)	;Configure primary UART reg
F16F	D3 EA	OUT	(SERURT),A	
F171	C9	RET		

```

;
; Serial Port Device Control Block
; -----

```

	72 F1	SPDCB EQU	\$	
F172	C3 65 F1	SPDINT JP	SPINIT	;Initialization vector
F175	00	SPDOPT DEFB	0	;Serial Port Options
				; Bit 0=Wait for CTS
				; Bit 1=Wait for DSR
F176	55	SPDBDR DEFB	55H	;Baud rate code
F177	6C	SPDCFG DEFB	6CH	;UART configuration

TRS-80 Model 4 BIOS Version 2.00+ I/O routines for drive M:

```

;
; *****
; * Memory drive read routine *
; * Input: Select parameters in Select Control Block *
; * Output: Record moved to (DSBDMA) *
; *****
F3B5 CD D7 F3 MDREAD CALL MDADDR ;Set up addresses
F3B8 CD ED F3 CALL MDMOVE ;Move data to work buffer
F3BB ED 5B 25 F7 LD DE,(DSBDMA) ;Point DE at destination
F3BF 21 DB F9 LD HL,WKBUF ; & HL at source

```

```

;
; *****
; * Move a record *
; * Input: HL=Source record address *
; * DE=Destination record address *
; * Output: None - record moved to new location *
; *****
F3C2 01 80 00 MOVREC LD BC,128 ; for 1 record length
F3C5 ED B0 LDIR ;Move the record
F3C7 C9 RET

```

```

;
; *****
; * Memory drive write routine *
; * Input: Select parameters in Select Control Block *
; * Output: Record moved from (DSBDMA) *
; *****
F3C8 2A 25 F7 MDWRIT LD HL,(DSBDMA) ;Point HL at record
F3CB 11 DB F9 LD DE,WKBUF ; & DE at work buffer
F3CE CD C2 F3 CALL MOVREC ;Move record to work buffer
F3D1 CD D7 F3 CALL MDADDR ;Set up addresses
F3D4 EB EX DE,HL ;Switch for write
F3D5 13 16 JR MDMOVE ;Write record & exit

```

```

F3D7 21 23 F7
F3DA 7E
F3DB 2B
F3DC 66
F3DD 2E 00
F3DF CB 3C
F3E1 CB 1D
F3E3 F6 06
F3E5 07
F3E6 07
F3E7 07
F3E8 07
F3E9 11 DB F9
F3EC C9

```

```

; *****
; * Memory drive address setup routine *
; * Input: Information in Select Control Block *
; * Output: A=Map address select bits *
; * DE=Internal record buffer address *
; * HL=Record address in alternate memory map *
; *****
MDADDR LD HL,DSBSEC+1 ;Point HL at sector #
LD A,(HL) ;Page # to A (0 or 1)
DEC HL ;Point to 1st byte of sector
LD H,(HL) ;Memory address * 256 to HL
LD L,0
SRL H ;Divide by 2 to get true
RR L ; record address
OR 6 ;Set FXUPMEM, MBIT1
RLCA ;Rotate into bits 6-4
RLCA
RLCA
RLCA
LD DE,WKBUF ;Point to internal buffer
RET

```

```

F3ED F3
F3EE F6 8F
F3F0 D3 84
F3F2 CD C2 F3
F3F5 3E 8F
F3F7 D3 84
F3F9 AF
F3FA C9

```

```

; *****
; * Memory drive data move routine *
; * Input: A=Address select bits for move *
; * HL=Source address for move *
; * DE=Destination address for move *
; * Output: 128 bytes moved as requested *
; *****
MDMOVE DI ;No interrupts now!
OR KVMOUT ;Set mapping bits
OUT (MEMCTL),A ;Select alternate map
CALL MOVREC ;Move the record
LD A,KVMOUT ;Set normal map bits
OUT (MEMCTL),A ;Restore normal map
XOR A ;Clear status for good I/O
RET

```

TRS-80 Model 4 BIOS Version 2.00+ I/O routines for Floppy Drives

```

F3FB F3
F3FC 3D
F3FD 28 07
F3FF 3D
F400 28 3A

```

```

; *****
; * Floppy Disk I/O Driver *
; * Input: A=Function code *
; * 1 - Read a sector *
; * 2 - Write a sector *
; * BC=Track number (B should always be 0) *
; * DE=Sector number (D should always be 0) *
; * HL=Buffer address *
; * IX=DCB for selected drive *
; * IY=DPB for selected drive *
; * Output: A=Status of operation *
; * Bits match WD 1791 FDC conventions *
; *****
FDD DI ;No interrupts
DEC A ;Check function code
JR Z,FDREAD ;1 = Read
DEC A
JR Z,FDWRIT ;2 = Write

```

```

F402 3E 10
F404 B7
F405 C9

```

```

; Return INOP status for Floppy Disk Drive
; -----
FDINOP LD A,10H ;Return RNF error
OR A ;Clear Z to set error status
RET

```

F406	CD 83 F4	FDREAD	CALL	FDBEGN	;Start the I/O operation
F409	C0		RET	NZ	;Exit if error
F40A	CD 19 F4		CALL	FDRD3	;Try to read 3 times
F40D	C8		RET	Z	;Exit if successful
F40E	F8		RET	M	;Exit if inoperative
F40F	CD 16 F5		CALL	FDJOG	;Jog the head
F412	CD 19 F4		CALL	FDRD3	;Try 3 more times
F415	C8		RET	Z	;Exit if read OK
F416	CD 0D F5		CALL	FDRST	;Restore the drive

; Read a sector with 3 attempts

F419	CD 22 F4	FDRD3	CALL	FDRDSC	;Try to read the sector
F41C	C8		RET	Z	;Exit if it worked
F41D	F8		RET	M	;Exit if inoperative
F41E	CD 22 F4		CALL	FDRDSC	;Try again
F421	C8		RET	Z	;Exit if OK

; Read a sector

F422	E5	FDRDSC	PUSH	HL	;Save buffer address
F423	06 80		LD	B,80H	;Set up read command
F425	CD 4A F5		CALL	FDSET	;Start the command
F428	38 F4		DEFW	FDRDS3	; Termination address
F42A	DB F0	FDRDS1	IN	A,(FDCCTL)	;Read the status
F42C	A3		AND	E	;Got a DRQ yet?
F42D	28 FB		JR	Z,FDRDS1	;Loop if not
F42F	ED A2		INI		;Read first byte
F431	7A		LD	A,D	;Establish wait states
F432	D3 F4	FDRDS2	OUT	(FDCSEL),A	;Go into wait state
F434	ED A2		INI		;Read a byte
F436	18 FA		JR	FDRDS2	;Keep reading
F438	E1	FDRDS3	POP	HL	;Restore buffer address
F439	E6 9C		AND	9CH	;Any errors?
F43B	C9		RET		;Exit with status

; Write a sector to disk

F43C	CD 83 F4	FDWRIT	CALL	FDBEGN	;Start the I/O operation
F43F	C0		RET	NZ	;Exit if error
F440	CD 51 F4		CALL	FDWT3	;Try to write 3 times
F443	C8		RET	Z	;Exit if successful
F444	E6 C0		AND	0C0H	;Exit if inop or w/p
F446	C0		RET	NZ	
F447	CD 16 F5		CALL	FDJOG	;Jog the head
F44A	CD 51 F4		CALL	FDWT3	;Try 3 more times
F44D	C8		RET	Z	;Exit if write OK
F44E	CD 0D F5		CALL	FDRST	;Restore the drive

; Write a sector with 3 attempts

F451	CD 5C F4	FDWT3	CALL	FDWTSC	;Try to write the sector
F454	C8		RET	Z	;Exit if it worked
F455	E6 C0		AND	0C0H	;Exit if inop or w/p
F457	C0		RET	NZ	
F458	CD 5C F4		CALL	FDWTSC	;Try again
F45B	C8		RET	Z	;Exit if OK

; Write a sector

F45C	E5	FDWTSC	PUSH	HL	;Save buffer address
F45D	06 A0		LD	B,A0H	;Set up write command
F45F	CD 4A F5		CALL	FDSET	;Start the command
F462	7F F4		DEFW	FDWTS4	; Termination address

F464	DB F0	FDWTS1	IN	A,(FDCCTL)	;Read the status
F466	A3		AND	E	;Got a DRQ yet?
F467	28 FB		JR	Z,FDWTS1	;Loop if not
F469	ED A3		OUTI		;Output first byte
F46B	7E		LD	A,(HL)	;Get the next byte in A
F46C	23		INC	HL	
F46D	0E F0		LD	C,FDCCTL	;Point C at status reg
F46F	ED 58	FDWTS2	IN	E,(C)	;Loop for second DRQ
F471	E2 6F F4		JP	PO,FDWTS2	
F474	D3 F3		OUT	(FDCDAT),A	;Output the byte
F476	0E F3		LD	C,FDCDAT	;Restore C to data port
F478	7A		LD	A,D	;Establish wait states
F479	D3 F4	FDWTS3	OUT	(FDCSEL),A	;Go into wait state
F47B	ED A3		OUTI		;Write a byte
F47D	18 FA		JR	FDWTS3	;Keep writing
F47F	E1	FDWTS4	POP	HL	;Restore buffer address
F480	E6 FC		AND	0FCH	;Any errors?
F482	C9		RET		;Exit with status
; Select the disk & wait for speed					

F483	FD 7E 13	FDBEGN	LD	A,(IY+DPBOPT)	;Get drive option bits
F486	E6 80		AND	80H	;Isolate density
F488	DD B6 03		OR	(IX+DKDSEL)	;Combine with select bits
F48B	DD 71 0B		LD	(IX+DKDLTK),C	;Save logical track #
F48E	FD CB 13 76		BIT	6,(IY+DPBOPT)	;Double-sided disk?
>> NOTE	1/2---- EXBIOS replaces the above instruction with this:				
>> NOTE	CD 80 FE 00		CALL	BIOSEX	;Call BIOS patch
F492	28 1C		JR	Z,FDBEG2	;Go if not
F494	CB 39		SRL	C	;Divide track # by 2
F496	FD CB 13 56		BIT	2,(IY+DPBOPT)	;Side 1 same track #?
F49A	20 03		JR	NZ,FDBEG1	;Go if not
F49C	DD 71 0B		LD	(IX+DKDLTK),C	;Save new track #
F49F	30 0F	FDBEG1	JR	NC,FDBEG2	;Go if on side 0
F4A1	F6 10		OR	10H	;Turn on side 1 select
F4A3	FD CB 13 5E		BIT	3,(IY+DPBOPT)	;Side 1 sectors biased?
F4A7	28 07		JR	Z,FDBEG2	;Go if not
F4A9	F5		PUSH	AF	;Save select bits
F4AA	7B		LD	A,E	;Get sector #
F4AB	FD 86 0F		ADD	A,(IY+DPBSPT)	;Add side 1 bias
F4AE	5F		LD	E,A	;Restore sector #
F4AF	F1		POP	AF	;Restore select bits
F4B0	DD 77 0A	FDBEG2	LD	(IX+DKDCSL),A	;Save select bits
F4B3	79		LD	A,C	;Get track #
F4B4	FD CB 13 6E		BIT	5,(IY+DPBOPT)	;Double stepping drive?
F4B8	28 01		JR	Z,FDBEG3	;Go if not
F4BA	87		ADD	A,A	;Compute true track #
F4BB	DD BE 08	FDBEG3	CP	(IX+DKDPTO)	;Precomp needed yet?
F4BE	38 04		JR	C,FDBEG4	;Go if not
F4C0	DD CB 0A EE		SET	5,(IX+DKDCSL)	;Turn it on
F4C4	57	FDBEG4	LD	D,A	;True track # to D
F4C5	DB F0		IN	A,(FDCCTL)	;Get controller status
F4C7	07		RLCA		;Ready bit to C flag
F4C8	CD 3C F5		CALL	FDSEL	;Select the drive
F4CB	3E D0		LD	A,0D0H	;Reset the FDC
F4CD	D3 F0		OUT	(FDCCTL),A	
F4CF	30 0D		JR	NC,FDBEG6	;Go if drive running
F4D1	DD 46 05		LD	B,(IX+DKDSTD)	;Start-up delay to B
F4D4	3E FA	FDBEG5	LD	A,250	;Delay for 1/4 second
F4D6	CD 14 ED		CALL	MSDELY	
F4D9	CD 3C F5		CALL	FDSEL	;Select again
F4DC	10 F6		DJNZ	FDBEG5	;Wait for speed
F4DE	7B	FDBEG6	LD	A,E	;Get the sector #
F4DF	D3 F2		OUT	(FDCSEC),A	;Give to controller


```

; Seek the proper track
; -----
F4E1 DD 7E 09
F4E4 D3 F1
F4E6 3C
F4E7 CC 1D F5
F4EA 7A
F4EB DD BE 07
F4EE D2 02 F4
F4F1 D3 F3
F4F3 DD 77 09
F4F6 CD 3C F5
F4F9 DB F1
F4FB 92
F4FC 28 09
F4FE 7A
F4FF B7
F500 28 02
F502 3E 10
F504 CD 1D F5
F507 DD 7E 0B
F50A D3 F1
F50C C9

FDSEEK LD A,(IX+DKDCTK) ;Get current track
OUT (FDCTRK),A ;Give to controller
INC A ;First access (=FFH)?
CALL Z,FDSTEP ;Restore the drive if so
LD A,D ;Get desired track
CP (IX+DKDNTK) ;Is it legal?
JP NC,FDINOP ;Return INOP if so
OUT (FDCDAT),A ;Output track to FDC
LD (IX+DKDCTK),A ;Save also in DCB
CALL FDSEL ;Re-select the drive
IN A,(FDCTRK) ;Get the track #
SUB D ;Any seek required?
JR Z,FDSEK2 ;Go if not
LD A,D ;Target track # to A
OR A ;Is it zero?
JR Z,FDSEK1 ;Go if yes
LD A,10H ;Set up seek command
CALL FDSTEP ;Seek the track
FDSEK1 LD A,(IX+DKDLTK) ;Get logical track #
FDSEK2 OUT (FDCTRK),A ;Give it to controller
RET

;
; Restore the head for I/O retry
; -----
F50D DD 56 09
F510 DD 36 09 FF
F514 18 CB

FDRST LD D,(IX+DKDCTK) ;Current track # to D
LD (IX+DKDCTK),0FFH ;Force restore
JR FDSEEK ;Restore, seek & exit

;
; Jog the head for I/O retry
; -----
F516 3E 58
F518 CD 1D F5
F51B 3E 68

FDJOG LD A,58H ;Step the head in 1 track
CALL FDSTEP
LD A,68H ;Now step out 1 track

;
; Perform a step operation
; -----
F51D C5
F51E 4F
F51F 3E 02
F521 CD 14 ED
F524 DD 7E 04
F527 E6 03
F529 B1
F52A C1
F52B CD 42 F5
F52E CD 3C F5
F531 DB F0
F533 1F
F534 38 F8
F536 DD 7E 06
F539 C3 14 ED

FDSTEP PUSH BC ;Save BC
LD C,A ;Save step command
LD A,2 ;Wait 2 ms to be sure
CALL MSDELY ; erase turned off
LD A,(IX+DKDATT) ;Get drive attributes
AND 3 ;Isolate step rate
OR C ;Combine with command
POP BC ;Restore BC
CALL FDCMD ;Issue step command
FDSTP1 CALL FDSEL ;Reselect the drive
IN A,(FDCCTL) ;Get the status
RRA ;Still busy?
JR C,FDSTP1 ;Loop if yes
LD A,(IX+DKDSTL) ;Settle time to A
JP MSDELY ;Delay & return

;
; Keep disk selected until not busy
; -----
F53C DD 7E 0A
F53F D3 F4
F541 C9

FDSEL LD A,(IX+DKDCSL) ;Select the drive
OUT (FDCSEL),A
RET

;
; Issue a command to the disk controller
; -----
F542 D3 F0
F544 3E 14
F546 3D
F547 20 FD

FDCMD OUT (FDCCTL),A ;Issue the command
LD A,20 ;Set delay counter
FDCMD1 DEC A ;Count down 16 usec
JR NZ,FDCMD1 ;Loop if not zero

```

```
; Set up for I/O to FDC
```

```

F54A E5 FDSET PUSH HL ;Save buffer address
F54B 3A 66 00 LD A,(0066H) ;Save NMI vector
F54E 32 58 F9 LD (NMITMP),A
F551 2A 67 00 LD HL,(0067H)
F554 22 59 F9 LD (NMITMP+1),HL
F557 3E C3 LD A,0C3H ;Set up new NMI vector
F559 32 66 00 LD (0066H),A
F55C 21 88 F5 LD HL,FDNMI
F55F 22 67 00 LD (0067H),HL
F562 E1 POP HL ;Get buffer address
F563 E3 EX (SP),HL ;Swap with return
F564 5E LD E,(HL) ;Get termination address
F565 23 INC HL
F566 56 LD D,(HL)
F567 23 INC HL
F568 EB EX DE,HL ;Termination to HL
F569 E3 EX (SP),HL ;Put on stack
F56A D5 PUSH DE ;Replace return address
F56B DD 56 0A LD D,(IX+DKDCSL) ;Set D=Select + bit 6
F56E CB F2 SET 6,D
F570 1E 02 LD E,2 ;Set E to DRQ mask
F572 0E F3 LD C,FDCDAT ;Set C to Data port
F574 CD 3C F5 CALL FDSEL ;Re-select the drive
F577 DD CB 04 76 BIT 6,(IX+DKDATT) ;Is this 8 inch drive?
F57B 28 02 JR Z,FDSET1 ;Go if not
F57D CB D8 SET 3,B ;Enable HLT delay
F57F 78 FDSET1 LD A,B ;Command to A
F580 CD 42 F5 CALL FDCMD ;Give it to the controller
F583 3E C0 LD A,0C0H ;Enable NMI from disk
F585 D3 E4 OUT (NMICTL),A
F587 C9 RET

```

```
; Non-maskable interrupt service routine
```

```

F588 E3 FDNMI EX (SP),HL ;Discard return, save HL
F589 AF XOR A ;Turn off NMI enable
F58A D3 E4 OUT (NMICTL),A
F58C 3A 58 F9 LD A,(NMITMP) ;Restore NMI vector
F58F 32 66 00 LD (0066H),A
F592 2A 59 F9 LD HL,(NMITMP+1)
F595 22 67 00 LD (0067H),HL
F598 DB F0 IN A,(FDCCTL) ;Read final status
F59A E1 POP HL ;Restore HL
F59B C9 RET

```

TRS-80 Model 4 BIOS Version 2.00+ Disk tables & parameters

```

; *****
; * Disk Parameter Headers (DPH) for drives A-D & M *
; *****
F59C 85 F6 00 00 DPHA DEFW XLT0,0000H ;Drive A parameter header
F5A0 00 00 00 00 DEFW 0000H,0000H
F5A4 5B F9 EC F5 DEFW DBUF,DPB0
F5A8 D8 F8 40 F7 DEFW CHK0,ALL0
;
F5AC A3 F6 00 00 DPHB DEFW XLT1,0000H ;Drive B parameter header
F5B0 00 00 00 00 DEFW 0000H,0000H
F5B4 5B F9 01 F6 DEFW DBUF,DPB1
F5B8 F8 F8 A4 F7 DEFW CHK1,ALL1
;
F5BC C1 F6 00 00 DPHC DEFW XLT2,0000H ;Drive C parameter header
F5C0 00 00 00 00 DEFW 0000H,0000H

```

```

F5C4      5B F9 10 F0      DEFW      DBUF,DPB2
F5C8      18 F9 08 F8      DEFW      CHK2,ALL2

;
F5CC      DF F6 00 00      DPHD      DEFW      XLT3,0000H      ;Drive D parameter header
F5D0      00 00 00 00      DEFW      0000H,0000H
F5D4      5B F9 2B F6      DEFW      DBUF,DPB3
F5D8      38 F9 6C F8      DEFW      CHK3,ALL3

;
F5DC      00 00 00 00      DPHM      DEFW      0000H,0000H      ;Drive M parameter header
F5E0      00 00 00 00      DEFW      0000H,0000H
F5E4      5B F9 40 F6      DEFW      DBUF,DPBM
F5E8      00 00 D0 F8      DEFW      0000H,ALLM

;
; Offsets used to address Disk Parameter Header fields
; -----
00 00      DPHXLT      EQU      0      ;Skew translation table
08 00      DPHBUF      EQU      8      ;Directory buffer address
0A 00      DPHDPB      EQU      10     ;Disk Parameter Block
0C 00      DPHCSV      EQU      12     ;Check vector address
0E 00      DPHALV      EQU      14     ;Allocation vector address

;
; *****
; * Disk Parameter Blocks (DPB) for drives A-D & M *
; *****

F5EC      EC F5      DPB0      EQU      $      ;Drive 0 parameter block
F5EE      24 00      DEFW      36      ; Records per track
F5EF      04        DEFB      4        ; Block shift count
F5F0      0F        DEFB      15       ; Block mask
F5F1      01        DEFB      1        ; Extent mask count
F5F3      54 00      DEFW      84      ; Highest allocation block
F5F5      7F 00      DEFW      127     ; Highest directory #
F5F7      C0        DEFB      192     ; Initial allocation 0
F5F9      00        DEFB      0        ; Initial allocation 1
F5FB      20 00      DEFW      32      ; Directory check size
F5FD      02 00      DEFW      2        ; Reserved track count
F5FE      12        DEFB      18       ; Sectors per track
F5FF      01        DEFB      1        ; Sector size code
F600      55 F6      DEFW      D0DCB    ; Drive DCB Address
F601      80        DEFB      80H      ; Drive option bits
F602      01        DEFB      1        ; Drive format ID code
F603      01 F6      DPB1      EQU      $      ;Drive 1 parameter block
F604      24 00      DEFW      36      ; Records per track
F605      04        DEFB      4        ; Block shift count
F606      0F        DEFB      15       ; Block mask
F607      01        DEFB      1        ; Extent mask count
F608      54 00      DEFW      84      ; Highest allocation block
F609      7F 00      DEFW      127     ; Highest directory #
F60A      C0        DEFB      192     ; Initial allocation 0
F60B      00        DEFB      0        ; Initial allocation 1
F60C      20 00      DEFW      32      ; Directory check size
F60D      02 00      DEFW      2        ; Reserved track count
F60E      12        DEFB      18       ; Sectors per track
F60F      01        DEFB      1        ; Sector size code
F610      61 F6      DEFW      D1DCB    ; Drive DCB Address
F611      80        DEFB      80H      ; Drive option bits
F612      01        DEFB      1        ; Drive format ID code
F613      16 F6      DPB2      EQU      $      ;Drive 2 parameter block
F614      24 00      DEFW      36      ; Records per track
F615      04        DEFB      4        ; Block shift count
F616      0F        DEFB      15       ; Block mask
F617      01        DEFB      1        ; Extent mask count
F618      54 00      DEFW      84      ; Highest allocation block
F619      7F 00      DEFW      127     ; Highest directory #
F61A      C0        DEFB      192     ; Initial allocation 0
F61B      00        DEFB      0        ; Initial allocation 1
F61C      20 00      DEFW      32      ; Directory check size
F61D      02 00      DEFW      2        ; Reserved track count
F61E      12        DEFB      18       ; Sectors per track
F61F      01        DEFB      1        ; Sector size code
F620      61 F6      DEFW      D1DCB    ; Drive DCB Address
F621      80        DEFB      80H      ; Drive option bits
F622      01        DEFB      1        ; Drive format ID code

```

F621	20 00	DEFW	2	; Directory check size
F623	02 00	DEFW	2	; Reserved track count
F625	12	DEFB	18	; Sectors per track
F626	01	DEFB	1	; Sector size code
F627	6D F6	DEFW	D2DCB	; Drive DCB Address
F629	80	DEFB	80H	; Drive option bits
F62A	01	DEFB	1	; Drive format ID code
;				
	2B F6	DPB3	EQU	\$; Drive 3 parameter block
F62B	24 00	DEFW	36	; Records per track
F62D	04	DEFB	4	; Block shift count
F62E	0F	DEFB	15	; Block mask
F62F	01	DEFB	1	; Extent mask count
F630	54 00	DEFW	84	; Highest allocation block
F632	7F 00	DEFW	127	; Highest directory #
F634	C0	DEFB	192	; Initial allocation 0
F635	00	DEFB	0	; Initial allocation 1
F636	20 00	DEFW	32	; Directory check size
F638	02 00	DEFW	2	; Reserved track count
F63A	12	DEFB	18	; Sectors per track
F63B	01	DEFB	1	; Sector size code
F63C	79 F6	DEFW	D3DCB	; Drive DCB Address
F63E	80	DEFB	80H	; Drive option bits
F63F	01	DEFB	1	; Drive format ID code
;				
	40 F6	DPBM	EQU	\$; Drive M parameter block
F640	00 02	DEFW	512	; Records per track
F642	03	DEFB	3	; Block shift count
F643	07	DEFB	7	; Block mask
F644	00	DEFB	0	; Extent mask count
F645	3F 00	DEFW	63	; Highest allocation block
F647	1F 00	DEFW	31	; Highest directory #
F649	80	DEFB	128	; Initial allocation 0
F64A	00	DEFB	0	; Initial allocation 1
F64B	00 00	DEFW	0	; Directory check size
F64D	00 00	DEFW	0	; Reserved track count
F64F	00	DEFB	0	; Sectors per track
F650	00	DEFB	0	; Sector size code
F651	00 00	DEFW	0000H	; Drive DCB Address
F653	00	DEFB	0	; Drive option bits
F654	00	DEFB	0	; Drive format ID code
;				
; Offsets used to address Disk Parameter Block (DPB) fields				
; -----				
00 00	DPBRPT	EQU	0	;Records Per Track
02 00	DPBBSH	EQU	2	;Block Shift factor
03 00	DPBBLM	EQU	3	;Block Mask
04 00	DPBEXM	EQU	4	;Extent Mask
05 00	DPBDSM	EQU	5	;Drive capacity
07 00	DPBDRM	EQU	7	;Directory Maximum
09 00	DPBAL0	EQU	9	;Initial Allocation 0
0A 00	DPBAL1	EQU	10	;Initial Allocation 1
0B 00	DPBCKS	EQU	11	;Check area size
0D 00	DPBOFF	EQU	13	;Reserved track count
0F 00	DPBSPT	EQU	15	;Sectors Per Track
10 00	DPBSSZ	EQU	16	;Sector Size code
11 00	DPBDCB	EQU	17	;Drive DCB address
13 00	DPBOPT	EQU	19	;Drive option bits
:	:	:	:	; 7=Density (0=S, 1=D)
:	:	:	:	; 6=Sides (0=S, 1=D)
:	:	:	:	; 5=Step (0=Norm, 1=2 x)
:	:	:	:	; 4=Data (0=Norm, 1=Inv)
:	:	:	:	; 3=Side 1 (0=Norm, 1=Bias)
:	:	:	:	; 2=Track # (0=Norm, 1=Bias)
:	:	:	:	; 1-0=Reserved
14 00	DPBDID	EQU	20	;Disk format ID #

```

; *****
; * Disk Device Control Blocks (DCB) for drives 0-3 & M *
; *****
F655 55 F6      D0DCB EQU $      ;Drive 0 DCB
      C3 FB F3   JP      FDD      ; Driver vector
F658 01         DEFB 01H         ; Drive select bits
F659 00         DEFB 00H         ; Drive attribute bits
F65A 02         DEFB 2          ; Start up delay in 1/4 sec
F65B 0F         DEFB 15         ; Settle time in Ms
F65C 28         DEFB 40         ; Number of tracks
F65D 16         DEFB 22         ; Precomp turn-on track
F65E FF         DEFB 255        ; Current track
F65F 00         DEFB 0          ; Current select bits
F660 00         DEFB 0          ; Logical track #

;
;D1DCB EQU $      ;Drive 1 DCB
      JP      FDD      ; Driver vector
F661 61 F6      DEFB 02H         ; Drive select bits
      C3 FB F3   DEFB 00H         ; Drive attribute bits
F664 02         DEFB 2          ; Start up delay in 1/4 sec
F665 00         DEFB 15         ; Settle time in Ms
F666 02         DEFB 40         ; Number of tracks
F667 0F         DEFB 22         ; Precomp turn-on track
F668 28         DEFB 255        ; Current track
F669 16         DEFB 0          ; Current select bits
F66A FF         DEFB 0          ; Logical track #
F66B 00
F66C 00

;
;D2DCB EQU $      ;Drive 2 DCB
      JP      FDD      ; Driver vector
F66D 6D F6      DEFB 04H         ; Drive select bits
      C3 FB F3   DEFB 00H         ; Drive attribute bits
F670 04         DEFB 2          ; Start up delay in 1/4 sec
F671 00         DEFB 15         ; Settle time in Ms
F672 02         DEFB 40         ; Number of tracks
F673 0F         DEFB 22         ; Precomp turn-on track
F674 28         DEFB 255        ; Current track
F675 16         DEFB 0          ; Current select bits
F676 FF         DEFB 0          ; Logical track #
F677 00
F678 00

;
;D3DCB EQU $      ;Drive 3 DCB
      JP      FDD      ; Driver vector
F679 79 F6      DEFB 08H         ; Drive select bits
      C3 FB F3   DEFB 00H         ; Drive attribute bits
F67C 08         DEFB 2          ; Start up delay in 1/4 sec
F67D 00         DEFB 15         ; Settle time in Ms
F67E 02         DEFB 40         ; Number of tracks
F67F 0F         DEFB 22         ; Precomp turn-on track
F680 28         DEFB 255        ; Current track
F681 16         DEFB 0          ; Current select bits
F682 FF         DEFB 0          ; Logical track #
F683 00
F684 00

;
; Offsets used to access Disk DCB fields
; -----
00 00      DKDDVR EQU 0      ;Driver address
03 00      DKDSEL EQU 3      ;Drive select bits
04 00      DKDATT EQU 4      ;Drive attribute bits
;
;      ; 7=Sides (0=S, 1=D)
;      ; 6=Type (0=5, 1=8)
;      ; 5-2=Reserved
;      ; 1-0=Step rate (0-3)
05 00      DKDSTD EQU 5      ;Drive start-up delay in Ms
06 00      DKDSTL EQU 6      ;Drive settle time in Ms
07 00      DKDNTK EQU 7      ;Number of tracks
08 00      DKDPTO EQU 8      ;Precomp turn-on track
09 00      DKDCTK EQU 9      ;Current track
0A 00      DKDCSL EQU 10     ;Current select bits

```

```

0B 00      DKDLTK EQU 11 ;Logical track #
;
; *****
; * Disk sector translation tables *
; * Space reserved for 30 sectors per track maximum *
; *****
F685 01 03 05 07 XLT0 DEFB 1,3,5,7,9,11,13,15,17,2
F689 09 0B 0D 0F
F68D 11 02
F68F 04 06 08 0A DEFB 4,6,8,10,12,14,16,18,0,0
F693 0C 0E 10 12
F697 00 00
F699 00 00 00 00 DEFB 0,0,0,0,0,0,0,0,0,0
F69D 00 00 00 00
F6A1 00 00
;
F6A3 01 03 05 07 XLT1 DEFB 1,3,5,7,9,11,13,15,17,2
F6A7 09 0B 0D 0F
F6AB 11 02
F6AD 04 06 08 0A DEFB 4,6,8,10,12,14,16,18,0,0
F6B1 0C 0E 10 12
F6B5 00 00
F6B7 00 00 00 00 DEFB 0,0,0,0,0,0,0,0,0,0
F6BB 00 00 00 00
F6BF 00 00
;
F6C1 01 03 05 07 XLT2 DEFB 1,3,5,7,9,11,13,15,17,2
F6C5 09 0B 0D 0F
F6C9 11 02
F6CB 04 06 08 0A DEFB 4,6,8,10,12,14,16,18,0,0
F6CF 0C 0E 10 12
F6D3 00 00
F6D5 00 00 00 00 DEFB 0,0,0,0,0,0,0,0,0,0
F6D9 00 00 00 00
F6DD 00 00
;
F6DF 01 03 05 07 XLT3 DEFB 1,3,5,7,9,11,13,15,17,2
F6E3 09 0B 0D 0F
F6E7 11 02
F6E9 04 06 08 0A DEFB 4,6,8,10,12,14,16,18,0,0
F6ED 0C 0E 10 12
F6F1 00 00
F6F3 00 00 00 00 DEFB 0,0,0,0,0,0,0,0,0,0
F6F7 00 00 00 00
F6FB 00 00
;
; >>---> End of disk resident portion of BIOS <---<<
;
TRS-80 Model 4 BIOS Version 2.00+ BIOS extension for CP/M 2.2 version 2.2x
;
; *****
; * Patch code loaded by EXBIOS *
; *****
FE80 ORG CCP+2A80H ;Patch area
;
FE80 FD CB 13 76 BIOSEX BIT 6,(IY+DPBOPT) ;Double-sided disk?
FE84 C8 RET Z ;Exit if not
FE85 D5 PUSH DE ;Save DE
FE86 57 LD D,A ;Save drive select
FE87 FD CB 13 4E BIT 1,(IY+DPBOPT) ;Alternate sides?
FE8B 28 24 JR Z,BIOSX4 ;Return to BIOS if not
FE8D DD 5E 07 LD E,(IX+DKDNTK) ;Track count to E
FE90 7B LD A,E ;Check track size
FE91 FE 28 CP 40 ;Is it other than 40?
FE93 20 08 JR NZ,BIOSX1 ;Go if yes

```


FE95	FD CB 13 66		BIT	4,(IY+DPBOPT)	;Is this 35T SuperBrain?
FE99	28 02		JR	Z,BIOSX1	;Go if not
FE9B	1E 23		LD	E,35	;Set track count to 35
FE9D	79	BIOSX1	LD	A,C	;Get track #
FE9E	93		SUB	E	;At end of side 0?
FE9F	30 03		JR	NC,BIOSX2	;Go if yes
FEA1	AF		XOR	A	;Set Z flag
FEA2	18 0F		JR	BIOSX5	;Return to BIOS
FEA4	FD CB 13 46	BIOSX2	BIT	0,(IY+DPBOPT)	;Going inside out?
FEA8	28 04		JR	Z,BIOSX3	;Go if not
FEAA	ED 44		NEG		;Compute correct track #
FEAC	83		ADD	A,E	
FEAD	3D		DEC	A	
FEAE	87	BIOSX3	ADD	A,A	;Double track # &
FEAF	3C		INC	A	; force side 1
FEB0	4F		LD	C,A	;Save new track #
FEB1	F6 FF	BIOSX4	OR	0FFH	;Reset Z flag for return
FEB3	7A	BIOSX5	LD	A,D	;Restore drive select
FEB4	D1		POP	DE	;Restore registers
FEB5	C9		RET		;Back to BIOS
			:		
			:	>>--->	End of BIOS patch