CHAPTER VII

INTRODUCTION TO ADVANCED TPM-II FEATURES

The preceding chapters, in which we covered the basics of TPM-II, assumed no previous knowledge of operating system fundamentals, or of CP/M. Starting with this chapter, we're going to delve into the internals of TPM-II. Our assumption is that either you are interested in this material out of a misguided sense of curiosity or you wish to try your hand at some assembly language programming on the QX-10.

WHAT'S IN THIS SECTION

The first topic we'll discuss is the anatomy of the QX-10: its memory map, bank switching, and accessing the CMOS RAM and boot ROM. We'll tackle the other hardware components in the system: interfacing to the two keyboards, printer and modem ports, and video display. The layout of the diskette will be described in detail, including directory format and disk parameters. Although we will attempt to give sufficient information for programming, we will not get down to the schematic level; those so inclined are urged to consult the QX-10 Technical Manual.

Next, we review the tools provided for assembly language programming. TPM-II doesn't provide you with an assembler or linker, but in fact this isn't a real deficiency, since most programmers already have an assembler with which they are familiar and comfortable. The system does, however, have several debugging tools and aids.

The final chapters are devoted to detailing the TPM-II interface. Each of the available system calls are presented, along with general information regarding TPM-II/program interface.

WHAT ISN'T COVERED

The primary purpose of this half of the manual is to aid the experienced assembly language programmer in writing new programs, or converting existing programs for the QX-10. As such, it is not a tutorial or reference manual on assembly language programming. There are many good books available on the subject, and readers who are interested in expanding their horizons are urged to read one of these text before, or in conjunction with, the material presented here.

ClibPDF - www.fastio.com

Finally, we will not attempt to cover the internal makeup of the Valdocs system. Valdocs is comprised of a set of interactive and closely related modules. Many of its functions and features are a direct result of the flexibility of TPM-II. These features will described fully when we cover system calls. Other features are implemented in one of the several modules which comprise the Valdocs systems. The interaction of the Valdocs modules, as well as information on interfacing them to non-Valdocs programs is discussed in a separate manual.

END CHAPTER 7

ClibPDF - www.fastio.com

CHAPTER VIII

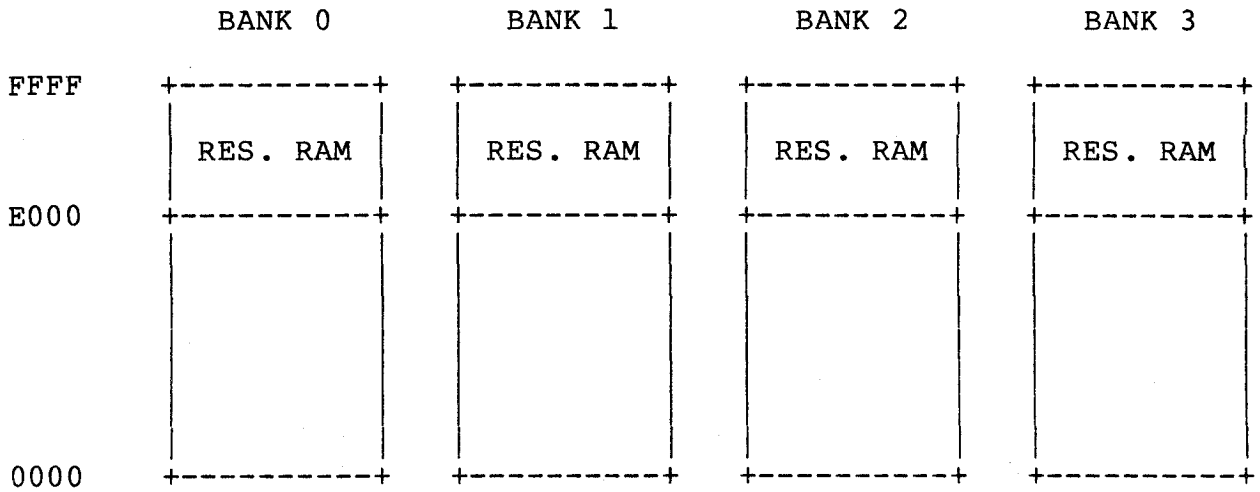
QX-10 ANATOMY

TPM-II is a flexible operating system that will run on any Z80-based microcomputer. The QX-10 version, however, has been modified extensively to take advantage of many unique hardware features. In this chapter we'll review the QX-10 hardware and introduce the component parts of TPM-II. Hopefully, by introducing them together we'll be able to convey the special interaction between TPM-II and the QX-10.

THE QX-10 MEMORY MAP

The QX-10 contains five major memory components. First, there are four banks of dynamic RAM, each bank containing 64K. The minimum RAM configuration is a single 64K bank, although most QX-10s (all Valdocs systems and the majority of non-Valdocs systems) are shipped with the four banks populated for a total dynamic RAM capacity of 256K.

In actual practice, not all of this 256K is available. The QX-10 has 8K of Resident RAM residing at the top of the 64K address space. As this 8K is common to all banks, each bank actually contains 56K of RAM. The memory map of the four banks and resident RAM is shown below:



The second memory component is the 2K boot ROM which holds the initial power-on initialization and boot code. The ROM overlays the dynamic RAM from memory locations 0000 - 07FF. At power on, the overlay is automatic to facilitate booting. The 2K ROM may be switched in and out of the memory map via an I/O port (we'll discuss this, along with controlling the bank switching

ClibPDF - www.fastio.com

shortly).

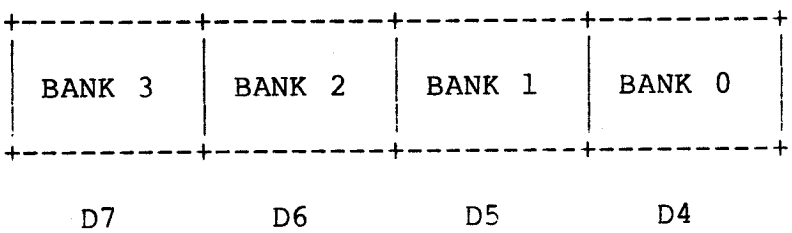
The third component is the CMOS RAM. The QX-10 contains 2K of battery-backed CMOS RAM for storing data across a power down. This RAM may be switched in or out of the memory map from locations 8000 - 87FF. When it's switched in, the CMOS RAM will overlay the dynamic RAM at those addresses.

The real-time clock chip contains 64 bytes of CMOS RAM, which is also battery-backed and maintains data across power downs. This RAM is used by the QX-10 and Valdocs to store system parameters. As such, your application programs may read the data, but should be careful when writing to it so that important system data is not overwritten. This RAM is I/O mapped, so it doesn't affect our memory map.

The final memory component is the video RAM. The QX-10 contains 128K of video memory for storing characters or pixels, depending on the display mode. The memory is not accessible by the Z80 CPU; instead, it is controlled by the 7220 graphics controller. The CPU and graphics controller communicate via I/O ports with DMA capability.

Controlling The Memory Components

The upper four bits of port 18H control the four RAM banks. Setting any of these bits to "1" will enable a bank as shown below. NOTE: DON'T SET MORE THAN ONE BIT AT A TIME TO "1"! You may disable the dynamic RAM altogether by setting the four upper bits of 18C to "0".



Remember that this only controls the 56K of RAM in each bank. The upper 8K of Resident RAM cannot be switched. (In reality, the resident RAM is the upper 8K of RAM bank 0. When the address decode logic detects an address of E000H - FFFFH, it overrides the bank select bits from port 18H and selects bank 0.)

In order to determine which bank is currently selected, the upper nibble of port 30H must be read. The lower nibble contains several status bits pertaining to the floppy disk drives and hence should be masked off. Bits D7 - D4 of 30H correspond to the same banks as their counterparts in 18H (shown above).

ClibPDF - www.fastio.com

The boot ROM is selected automatically at power on. It overrides the dynamic RAM in addresses 0000H - 07FFH. It is then deselected (and may subsequently be reselected) with bit 0 (D0) of I/O port 18H. Setting D0 to "1" disables the ROM, while making D0 a "0" reselects it.

The CMOS RAM is selected in a similar manner, except that I/O port 20H is used. Note that the CMOS RAM is initially deselected at power-on. To enable it to override the dynamic RAM from 8000H - 87FFH, set bit D0 of 20H to "1". To disable, turn D0 off.

The real-time clock RAM requires two I/O ports to access it. The address of the RAM (00H - 3FH) is output to port 3DH. The data for that address may then be read/written to port 3CH. Again, exercise caution when writing to this RAM as most of it is either currently used by the QX-10 or Valdocs, or is reserved for future use. There is, however, some useful information stored in it. The following table lists the current clock RAM usage:

CMOS CLOCK CHIP RAM LOCATIONS FOR THE EPSON QX-10

ADDRESS DEC	ADDRESS HEX	VALUE HEX	FUNCTION	DESCRIPTION
0	00	54	SECONDS	0-59 SEC.
1	01	00	SECONDS / ALARM	
2	02	13	MINUTES	0-59 MIN.
3	03	00	MINUTES / ALARM	
4	04	18	HOURS	0-23 HOUR
5	05	00	HOUR / ALARM	
6	06	03	DAYS OF WEEK	1-7 DAYS
7	07	06	DAYS OF MONTH	1-31 DAYS
8	08	09	MONTHS	1-12 MONTHS
9	09	83	YEAR	0-99 YEARS
10	0A	2A	REGISTER A	READ/WRITE REGISTER
11	0B	1A	REGISTER B	READ/WRITE REGISTER
12	0C	40	REGISTER C	READ ONLY REGISTER
13	0D	80	REGISTER D	READ ONLY REGISTER
14	0E	04	Bit 0 = 12/24 hr. Valdocs display	
15	0F	70	Video emulation mode byte	
16	10	00	System drive default (A)	
17	11	01	System User default (0)	
18	12	01	Data drive default (B)	
19	13	00	Data User default (0)	
20	14	00	System drive temp (always A)	
21	15	01	System User temp	

ClibPDF - www.fastio.com

22	16	03	Data drive temp
23	17	02	Data User temp
24	18	00	Print flags
25	19	01	Editor "Q" flags

26-30 (1A-1E) are RESERVED for future use.

26	1A	00
27	1B	00
28	1C	00
29	1D	00
30	1E	00

31	1F	C4	Mail protocol options byte
32	20	83	IOBYTE
33	21	26	KEYBOARD REPEAT RATE
34	22	00	KEYBOARD STATUS
35	23	05	BAUD RATE FOR EXTERNAL SERIAL PORT
36	24	01	VIDEO STATUS
37	25	02	NULLS ON TTY DEVICE OUTPUT
38	26	03	RESERVED FOR SCHEDULER
39	27	0F	Debug & expert status flags
40	28	07	MODEM FILE COUNT KEEPER (BYE)
41	29	00	RINGS BEFORE AUTO ANSWER - 1(MAIL&BYE)
42	2A	00	Used by SPOOLER
43	2B	00	Reserved for EDITor
44	2C	05	MODEM-- TEMP. BAUD RATE
45	2D	02	Printer TYPE Byte
46	2E	00	RESERVED FOR MAIL
47	2F	61	RESERVED FOR MAIL

48-61 (30-3D) are currently undefined.

48	30	00
49	31	00
50	32	00
51	33	00
52	34	00
53	35	00
54	36	00
55	37	00
56	38	00
57	39	00
58	3A	00
59	3B	00
60	3C	00

ClibPDF - www.fastio.com

61	3D	00	
62	3E	AA	Indicates valid setup done
63	3F	A2	SEE IF INITIALIZED

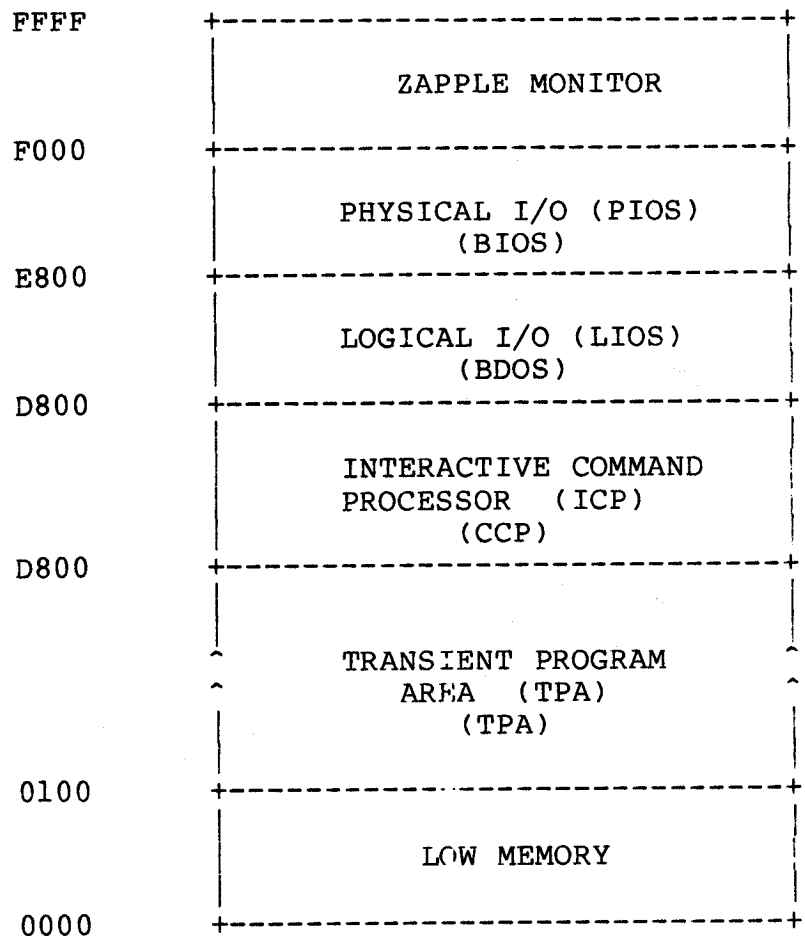
A few caveats regarding memory usage are in order at this point. First, remember that application programs written with CP/M in mind don't even know that banks 1 - 3 exist. If you write a program to run under TPM-II on the QX-10, and want to take advantage of the bank switch feature, use only banks 2 and 3. Bank 1 is reserved for the use of TPM-II, which performs some bank switching of its own in order to maximize both the number of features it supports and the size of the Transient Program Area.

TPM-II MEMORY MAP

As we just mentioned, most programs running under TPM-II know nothing of bank switching. Therefore, the effective memory map is simply a contiguous 64K of dynamic RAM. This memory can be divided into six functional areas. We've listed them from the top of memory down, describing their function and interaction. TPM-II is compatible with CP/M, so it stands to reason that certain portions of the TPM-II memory map will bear a resemblance to that of CP/M. In order to facilitate your understanding, we have included the CP/M equivalent where applicable.

(See next page)

ClibPDF - www.fastio.com



ZAPPLE MONITOR: The ZAPPLE monitor portion of TPM-II serves a dual purpose. First, it contains the low-level physical drivers for the QX-10 hardware. It operates in conjunction with PIOS (see below) to provide all of the hardware-dependent code necessary for the operation of TPM-II. In addition, it contains a Monitor for debugging and/or patching code.

PHYSICAL I/O SYSTEM (PIOS): PIOS contains all of the code necessary to complete the hardware interface with the QX-10. Working in conjunction with ZAPPLE, PIOS provides the high-level physical drivers. This function is equivalent to that of BIOS in CP/M. The PIOS is directly equivalent to BIOS -- it contains all of the standard jump vectors, its base address is stored in memory locations 0001 and 0002 (warm boot jump), and it resides above the logical I/O section.

LOGICAL I/O SYSTEM (LIOS): LIOS performs all file handling and logical I/O tasks for TPM-II. Programs communicate with LIOS via system calls providing a consistent interface to the QX-10 hardware. LIOS serves the same function in TPM-II as BDOS does in CP/M.

ClibPDF - www.fastio.com

INTERACTIVE COMMAND PROCESSOR (ICP): The ICP provides the interface between the user and the nucleus. The code for all resident commands is contained here. The ICP is equivalent to the CCP under CP/M.

TRANSIENT PROGRAM AREA (TPA): The TPA isn't a section of TPM-II in the same sense as PIOS or LIOS. The TPA contains no code; rather, it is the space allocated for application programs to run in. The TPA extends from 0100H to the bottom of the LIOS module. The program that runs in the TPA may overwrite the ICP if the program is large enough. The ICP is reloaded after each warm boot, in the event that the exit program overwrote all or a portion of it. The TPA serves the same function as CP/M's TPA.

LOW MEMORY: The bottom 256 bytes of bank 0 memory is reserved for TPM-II operation. This memory block is referred to as Low Memory, and it contains several important TPM-II mechanisms, such as the IOBYTE, warm boot and system call jump vectors, default disk I/O buffer, and default FCB. As with other components of TPM-II, Low Memory is functionally compatible with its CP/M counterpart.

LOW MEMORY DETAIL

The lowest 256 bytes of bank 0 are generally reserved for TPM-II operation. The following table lists the memory blocks and functions contained within this area. These locations are the same as the first page of a CP/M system.

ADDRESS	SIZE	FUNCTION
0000H	3	Warm Boot vector. Locations 0001H & 0002H may be used to determine the first location of the PIOS jump table by subtracting 0003H from this address.
0003H	1	IOBYTE image. The value of the I/O byte can't be set by writing a new value to this location. The Set IOBYTE function call must be used (see Chapter 10).
0004H	1	Current Disk Drive. A=00H, B=01H, etc.
0005H	3	TPM-II entry point. Locations 0006H and 0007H contain the address of the beginning of LIOS. Subtracting 0001H from it will yield the last address in the TPA.
0008H	40	Z-80 Restart Vectors 1 - 5 (RST1 - RST5). Block of memory is currently unused by TPM-II. It

ClibPDF - www.fastio.com

		can be used for vectors to support the Z80 RST1-RST5 instructions.
0030H	8	Z-80 Restart 6 Vector (RST6). Reserved for use by ZDDT in the QX-10.
0038H	8	Z-80 Restart 7 Vector (RST7). Reserved for use by ZAPPLE in the QX-10.
0040	28	TPM-II reserved memory. Contains track, sector, DMA address, current date and time, and other system data.
005CH	36	Default File Control Block (FCB). The FCB is described fully in Chapter 10.
0080H	128	Default command line and disk DMA buffer.

We will temporarily suspend our discussion of the TPM-II memory map, as it will be explored in greater depth in Chapter 9 when we discuss the TPM-II system calls. These system calls constitute the primary interface between application programs and TPM-II (and hence the QX-10).

QX-10 HARDWARE

TPM-II recognizes four logical I/O devices:

* CONSOLE Device
* READER Device/Serial Input
* PUNCH Device/Serial Output
* LIST or PRINTER Device

These I/O devices are supported via system calls. Of the four, the Console and List devices are the most important. The Reader and Punch devices, as such, are primarily of interest to a computer paleontologist. Since one seldom encounters a paper tape punch and reader, they have been re-assigned to the serial port on the QX-10.

Each logical I/O device has its own particular characteristics. The Console device is a bidirectional I/O device, i.e., it can both output and input characters. The List and Punch devices are output-only, and the Reader is input-only. Beyond this, the Console and List devices have certain system characteristics not possessed by the Punch and Reader. For example, all input and output to the Console may be echoed to the List device by the operator typing a CTRL-P. These features are discussed in Chap-

ClibPDF - www.fastio.com

ter 3 under the heading "TPM-II Control Characters."

The standard release version of TPM-II supports three physical I/O devices:

* Keyboard/Screen

* Serial Port

* Parallel Printer Port

These physical I/O devices are linked to one or more of the logical I/O devices. The standard assignments are:

Console = Keyboard/Screen

Reader = Serial Port Input

Punch = Serial Port Output

List = Parallel Printer Port

These assignments may be changed, or the current assignment displayed using the IOMOD utility. To display the current assignments, invoke IOMOD with no command line. The default I/O assignments are used in the following example:

```
A>IOMOD
C=C R=T P=T L=L

A>
```

The nomenclature used by IOMOD can be a little confusing: "C" is used for both the logical Console device and the physical Keyboard/Screen (the "C" abbreviates to CRT). Remember that they are different. Also, the Serial I/O port is referred to as "T" (for Teletype), and the Parallel Printer port as "L" (for Lineprinter).

IOMOD can be used to make new assignments. The following table lists all of the legal logical/physical device combinations. The general rule, however, is that any physical device can be assigned to a logical device as long as they have the same characteristics. For example, the logical Console must be capable of both input and output operations. Therefore, an output-only physical device, such as the parallel printer, can't be assigned to the Console.

ClibPDF - www.fastio.com

LOGICAL DEVICE:	CONSOLE	READER	PUNCH	LIST
PHYSICAL DEVICES:	CRT	Console	Console	Console
	TTY	Terminal	Terminal	Terminal
	User	User 1	User 1	List
		User 2	User 2	User

One final note regarding I/O assignments: for those of you who are used to CP/M, TPM-II stores the IOBYTE in memory location 0003H. In order to modify it, system call 8 (Set IOBYTE) must be used. Writing a new IOBYTE assignment to 0003H will change the assignment under CP/M, but not under TPM-II.

Disk Drives

The QX-10 is supplied with two double-density, double-sided floppy diskette drives. Data is stored in 512-byte sectors, with 10 sectors on each track and 40 tracks on each side. This provides a total capacity of 400K per diskette. However, not all of this space is available for program and data storage.

Both sides of the first two tracks are reserved for TPM-II. In addition, 4K is taken up with the diskette directory. TPM-II allocates enough room to store 128 directory entries on each diskette. A directory entry consists of a 32-byte block containing the name, size, sector allocations, and other pertinent information about the files. Directory entries will be discussed in greater detail in Chapter 9, under File Control Blocks. For now, you may regard the directory area as simply a second preallocated disk area.

The total amount of space available for data and program storage is 376K on each diskette. This capacity remains the same, regardless of whether or not the system tracks (0 and 1) contain TPM-II or how many directory entries are used. The two tracks and eight sectors are permanently allocated to these functions, and hence are unavailable for other storage.

END CHAPTER 8

ClibPDF - www.fastio.com

CHAPTER IX

ASSEMBLY LANGUAGE PROGRAMMING TOOLS

The following chapter examines the TPM-II tools that will assist you in writing and debugging assembly language programs. The basic TPM-II utilities, when added to your editor and assembler, will provide you with an extremely flexible and capable environment for program development.

ASSEMBLERS, LINKERS, AND EDITORS

TPM-II doesn't provide an assembler (and linker in the case of a relocatable assembler) or an editor for program generation. As we mentioned earlier, most programmers already have a Z80 assembler with which they are familiar and comfortable.

If you don't have an assembler, we recommend the Computer Design Labs' Z80 Assembler, available from:

CDL
342 Columbus Ave.
Trenton, NJ

This assembler produces relocatable Z80 code (CDL provides a Z80 linker and library manager to accompany it). Several other good relocatable assemblers are available, notably Digital Research's RMAC and Microsoft's MACRO-80. These assemblers produce relocatable object files (.REL files) in a format different from that of the CDL assembler, and one that is incompatible with the ZDDT debugger. Of course, ZDDT will work with the linked absolute object code files produced by their respective linkers, but if you intend to do an extensive amount of programming, you may wish to consider the CDL assembler, or one which produces .REL files in a compatible format.

In addition to an assembler, you will need to secure an editor with which to write the code for your programs. The Valdocs editor is not compatible with assemblers; characters in the file won't be stored in proper format as it is not possible to produce a file with a non-indexed file name.

ClibPDF - www.fastio.com

THE TPM-II DEBUGGERS

TPM-II provides two assembly language debugging tools. The first, the ZAPPLE monitor, is completely contained within TPM-II. ZAPPLE was discussed in the previous chapter when we examined the physical drivers for the QX-10's various I/O devices. In addition to these low-level drivers, ZAPPLE contains the code to support 17 different commands which can be used for debugging programs.

In addition to ZAPPLE, TPM-II provides a second debugger, ZDDT, which is actually an extension of ZAPPLE. If you review the command repertoire of both, you will notice a significant amount of overlap. Certain command are unique to each program, however. ZDDT has the ability to load a file into memory for debugging or modification. ZAPPLE contains several commands dealing with QX-10 hardware, which aren't supported in ZDDT. Taken together, these two programs provide an extensive set of commands that will make your debugging tasks much easier.

Since the programs are, for the most part, one and the same, we will document them as such. Each command will be covered in detail, and we will discuss not only its function and syntax, but also whether its supported by ZDDT only, ZAPPLE only, or both. In actual use, you will be able to switch back and forth between the two. Each has a different prompt character, so it's easy to tell which one you're in. The prompt for ZDDT is "-", while ZAPPLE's prompt is ">".

Invoking ZDDT And ZAPPLE

ZDDT is invoked the same as any other TPM-II utility. You may start ZDDT with or without a file name in the command line. If you enter a file name, ZDDT will be loaded into memory, and the file specified will be loaded next, starting a memory location 0100H. As ZDDT is loaded, it relocates itself to area in memory usually occupied by the ICP. ZDDT is the same size as the ICP, so it extends right up to the base of LIOS, which leaves room for the program (or file) to be read in its normal location in memory (starting at the base of the TPA).

ZDDT will sign on with the following message:

```
ZDDT V2.43 [TPM] -07/27/80
#XXXXXX
```


ClibPDF - www.fastio.com

•

If you don't specify a file name in the command line. The "XXXXXX" in the second line is the serial number. If you enter a file name in the command line, the message

NEXT=nnnn

will be displayed immediately below the serial number. This represents the memory location (rounded up to the nearest 128 bytes) of the first byte of available memory above the file which has been loaded in. Following the sign-on message, the ZDDT prompt is displayed ("-"), and you're ready to go.

A two-step process is required to invoke the ZAPPLE monitor. First, the monitor must be enabled. In order to enable ZAPPLE, you must get into Valdocs. Using the MENU key, get into the SETUP program, and select EXPERIENCE LEVEL. Set the level to "experienced", then exit to the Editor. Now, enter the SETUP program again. This time, you'll see an option entitled

<C>ntrl-\

Move the cursor over the "C" and press the RETURN key. ZAPPLE is now enabled. Remember, you must first change the Experience level to "expert", and then re-enter the menu to enable ZAPPLE. The reason for this "double access" is to prevent the inexperienced user from accidentally enabling ZAPPLE.

If you're wondering about the nomenclature, ZAPPLE is envoked by typing CONTROL-\. Unlike ZDDT, ZAPPLE can be invoked at any time, even in the middle of executing another program. The ZAPPLE prompt is ">". When ZAPPLE is first invoked, it displays this prompt, followed by the address it will return to in order to continue execution of the interrupted program.

ClibPDF - www.fastio.com

ZAPPLE And ZDDT Commands

The various ZDDT and ZAPPLE commands are listed below in alphabetical order. The commands consist of a single letter (either upper- or lowercase), followed by certain required and optional parameters. Required parameters are enclosed in square brackets "[ ]", while the optional ones use angled brackets "< >".

ASSIGN I/O DEVICES

Command: A
ZDDT/ZAPPLE: ZAPPLE
Syntax: A[l]=[p]

Sets the TPM-II I/O Byte (which determines the logical to physical assignments of the QX-10's input/output devices). TPM-II supports four logical devices, the Console, Punch, Reader, and List devices. There are three possible physical I/O devices. The A command uses the first letters of each device to make the assignments:

Logical Devices	Physical Devices
C - Console	T - TTY (RS232 port)
R - Reader	C - CRT/Keyboard
P - Punch	L - Lineprinter (parallel port)
L - List	

BOOT TPM-II

Command: B
ZDDT/ZAPPLE: ZAPPLE
Syntax: B

Only valid if the TPM-II operating system has not been booted up! Usually, TPM-II will be booted when the system is reset and a system diskette is in drive A:. However, if the option switch on the rear panel is set so that switch #2 is down, ZAPPLE will be read in from the diskette and sign-on, but the rest of TPM-II will NOT be booted. At this point, the "B" command may be issued to boot the system. This command is primarily used for system debugging. Once TPM-II has been booted, the command is de-activated.

ClibPDF - www.fastio.com

CMOS CLOCK & RAM I/O

Command: C
ZDDT/ZAPPLE: ZAPPLE
Syntax: CO[address],[value]
CI[address]

The hardware of the QX-10 contains a battery operated CMOS clock chip containing 50 bytes of battery back-up RAM in addition to the clock. The RAM is read and written via two I/O ports. To read one of the addresses, use the "CI" (CMOS input) command, and to write a new value, "CO" (CMOS output). For example:

```
>CO32,55
```

writes the value 55H to memory location 32H. Data that has been read is displayed in binary format. If you issue an input command to that same port, the value 55H will be displayed as follows:

```
>CI32 01010101
```

RESTART ZDDT

Command: C or K
ZDDT/ZAPPLE: ZDDT
Syntax: C or K

Both the "C" and "K" commands restart ZDDT. The sign-on message will be re-displayed, but all memory locations will remain undisturbed (ZDDT is restarted, but it is not re-loaded from the disk).

DISPLAY MEMORY

Command: D
ZDDT/ZAPPLE: Both
Syntax: D[start address],[end address],<bytes per line>

Allows you to view the contents of memory in hexadecimal form. The start and end address must be specified. The ZDDT and ZAPPLE versions have slightly different display formats. ZDDT displays only the hexadecimal value of each byte, while ZAPPLE also displays each byte's ASCII equivalent (provided it is in the legal ASCII range of 20H to 7FH. Other values are simply displayed as a ".").

The number of bytes per line can be modified with both versions. The default is 16 bytes per line, but any value up to 255

ClibPDF - www.fastio.com

can be selected. However, it is recommended that you choose an "even" hexidecimal value, such as 32, 64, 8, so that reading the display doesn't become an impossible task.

FILL MEMORY

Command: F
ZDDT/ZAPPLE: Both
Syntax: F[start address],[end address],[byte]

Writes the specified byte into each memory location from the starting address, up to and including the end address.

GOTO MEMORY WITH OPTIONAL BREAKPOINTS

Command: G
ZDDT/ZAPPLE: Both
Syntax:
G[address],<breakpoint address 1>,<breakpoint address 2>

Starts execution of a program in memory. The program counter of the Z80 is set to the address you enter, and the program begins normal execution at that point.

Optional breakpoint addresses may be entered. These breakpoints are implemented by replacing the existing instructions with RST instructions at these addresses. ZDDT uses RST 6, while ZAPPLE uses RST 7.

Once a breakpoint has been reached, you can execute any valid command. In order to continue program execution, you must issue another "G" command, with or without a new starting address. If you enter a starting address, execution will begin at this new address. If not, it will be the address in the "P" register. In most cases, this will be the address after the breakpoint.

The "G" command can be used to exit from either ZAPPLE or ZDDT. If you haven't traced any program execution, you can use the "G" command without an address. ZAPPLE will return you to the address you were at when you typed CONTROL-\ to invoke ZAPPLE, and ZDDT will warm boot TPM-II. Otherwise, ZAPPLE will return you to the instruction after the last breakpoint. Entering "G0" with ZDDT will execute a warm boot.

ClibPDF - www.fastio.com

HEXADECIMAL ARITHMETIC

Command: H
ZDDT/ZAPPLE: Both
Syntax: H[value 1],[value 2]

Handy for those who are not lightning quick with their hexadecimal math (and who is?). It displays the sum and difference of the two values. All arithmetic is done as 16-bit quantities, but if a carry bit is generated, it is not displayed. The sum appears on the right-hand side, and the difference on the left, as shown in the following example:

```
>H2000,4F23
6F23 D0DD
```

MEMORY TEST

Command: J
ZDDT/ZAPPLE: ZDDT
Syntax: J[start address],[end address]

Provides a non-destructive test of the memory block specified by the start and end addresses. If the memory is O.K., the ZDDT prompt ("-") will simply be redisplayed. If an error is encountered, the address of the defective byte will be printed along with a byte printed in binary notation to indicate the defective bit(s). The defective bits will be marked with a "1", whereas while the good bits will be marked with "0". The following example shows that bit 3 of address 405F is defective:

```
-J100,5000
405F  00001000
```

MOVE MEMORY BLOCK

Command: M
ZDDT/ZAPPLE: Both
Syntax: M[start address],[end address],[destination address]

Moves a block of memory from one location to another. The block is defined by its starting and ending addresses. Contents of this block are moved to the block starting with the destination address. You must take care that the two blocks don't overlap, or you may wind up with some not-so-funny results!

ClibPDF - www.fastio.com

PUT ASCII FROM KEYBOARD INTO MEMORY

Command: P
ZDDT/ZAPPLE: Both
Syntax: P[address]

Turns your QX-10 into a dictating machine. All characters entered at the keyboard are placed in memory starting at the specified address. Typing a CONTROL-D terminates the command. As "P" exits, prints the address of the next byte that would have been stored, to facilitate storing the characters with the TPM-II SAVE command.

QUERY I/O PORTS

Command: Q
ZDDT/ZAPPLE: ZAPPLE
Syntax: QI[port]
QO[port],[byte]

Used to directly read or write one of the QX-10 I/O ports. The input command (QI) displays the value in binary form. Any of the Z80's 256 ports can be accessed with this command. The two examples below show an input and an output operation:

```
QI18  00101000
QO71,7
```

READ A FILE

Command: R
ZDDT/ZAPPLE: ZDDT
Syntax: R<bias>

Allows you to read a file into memory from the disk. If no bias or offset is specified, the file is loaded into memory starting at location 100H. If a bias is specified, it is added to 100H to determine the starting memory address. For example, if a bias of 200H is entered, the file will start at memory location 300H.

Once the "R" command has been entered, with or without the bias, ZDDT will ask you for the name of the file you want read into memory. The following example will read the file LIST.SYS into memory with a 20FFH bias (starting at location 21FFH). The "R" command will display the next available memory location (rounded up to the nearest 128) once the file has been loaded into memory.

ClibPDF - www.fastio.com

```
-R20FF
NAME>LIST.SYS

NEXT=27FF
```

SUBSTITUTE MEMORY

Command: S
ZDDT/ZAPPLE: Both
Syntax: S[address]

Used to alter a byte in main memory. The current value of the selected address is displayed; you then have the option of changing it, or skipping to the next value by hitting the space-bar. To terminate the command, enter a return instead of pressing the space-bar.

The current address is printed on every 8-byte boundary. A backspace will cause a re-examination of the previous byte.

TYPE OUT MEMORY

Command: T
ZDDT/ZAPPLE: Both
Syntax: T[starting address],[end address],<bytes per line>

Operates in a manner very similar to the "D" command. The contents of memory from the start to end addresses are displayed on the screen. Using the "T" command, the values are displayed in ASCII characters rather than hexadecimal digits. This command is most useful for examining files consisting of ASCII characters. Like the "D" command, you can optionally change the number of bytes displayed on each line. The default is 64 bytes per line.

VERIFY MEMORY BLOCKS

Command: V
ZDDT/ZAPPLE: Both
Syntax: V[start address],[end address],[2nd start address]

Compares the block of memory specified by the start and end addresses, with another block of memory of the same length, beginning at the second start address. A byte-by-byte comparison will be made. If any differences are found, the address of the first block is printed, followed by the byte found in that address, and then the corresponding byte in the second block. In

ClibPDF - www.fastio.com

the example below, the fourth byte of the two blocks didn't match:

```
>V100,200,8000

0103 FF 7F
```

EXAMINE PROCESSOR REGISTERS

Command: X
ZDDT/ZAPPLE: Both
Syntax: X<'><r>

Used to display and alter the contents of the Z80's internal registers. The ZDDT version will display only the "main" registers, while ZAPPLE can display both the "main" and "prime" register sets. If "X" only (or "X'" in ZAPPLE to indicate the "prime" register set) is entered, the content of the entire register set is displayed, in the following format:

```
-X
A=18 B=AA C=06 D=00 E=04 F=44 H=04 L=2A M=3E P=0219 S=AC96 I=00
>X'
A=01 B=00 C=06 D=00 E=04 F=94 H=04 L=2A M=3E X=0219 Y=01FB R=45
```

Individual registers can be displayed and altered by entering the letter assigned to that register after the "X" or "X'". The current contents of the register will be displayed, and you can then modify them in a manner similar to the "S" command. If you enter a new value, the contents will be changed to that. Pressing the space bar will display the contents of the next register (the register label won't be displayed, but they go in order, from left to right, as they're displayed). To terminate the command press RETURN.

SEARCH FOR BYTE STRING

Command: Y
ZDDT/ZAPPLE: Both
Syntax: Y[byte],<byte>,<byte>,.....,<byte>

Searches memory for all occurrences of the byte string entered. The string may be from 1 to 255 characters in length. The address of the first byte of each matching string is printed.

ClibPDF - www.fastio.com

LAST MEMORY SPACE

Command: Z
ZDDT/ZAPPLE: Both
Syntax: Z

Displays the last available memory location before the beginning of either ZDDT or ZAPPLE, depending on which debugger you're in when the command is issued.

OTHER ASSEMBLY LANGUAGE TOOLS

In addition to the two TPM-II debugging commands that were just covered, there are several other programs that will help immensely in your programming. These utilities and commands are covered in the remainder of the chapter.

The SAVE Command

Once you've debugged and/or modified a program using ZDDT or ZAPPLE, it must be saved permanently on a disk, via the SAVE command. SAVE is part of the ICP, so it's always available, and it isn't read into the bottom of the TPA (and right over the program you want to save!) when it's invoked.

SAVE writes the contents of RAM, starting at memory location 100H into a file with the name you specify in the command line. You must also enter the amount of memory that is to be saved. This quantity is expressed in pages (256-byte blocks), in either decimal or hexidecimal notation. If the length field in the command line ends with the letter "H", it is assumed to be a hexadecimal number. If an "H" is not encountered, SAVE assumes that the field contains the number of pages to be saved in decimal notation.

The syntax of SAVE is:

```
A>SAVE n filename.typ
```

where n is the length field, and filename.typ is any legal TPM-II file name. If the file already exists, SAVE will write over it without asking, so make sure that your patches are solid before you replace the old version of a program with the newly patched one.

ClibPDF - www.fastio.com

```
A>SAVE 7 LIST.SYS
```

will save memory locations 100H - 7FFH as the file LIST.SYS, and

```
A>SAVE 0AH PAYROLL.DAT
```

will save the first 10 pages (100H - AFFH) as PAYROLL.DAT.

The hexidecimal notation is particularly helpful when you are using ZDDT. ZDDT will display the next memory location after the program (NEXT=nnnn). If you take the first two digits, you will have the number of pages you need to save, in hexidecimal. For example, if ZDDT shows the next memory location to be

```
NEXT=74A3
```

saving 74H pages with the following command:

```
SAVE 74H PROGRAM.COM
```

will save the program and relieve you of the task of converting 74H into its decimal equivalent.

The FILES Utility

The FILES utility allows you to look at the directory entry of every file on a disk. An explanation of the directory will be reserved for the next chapter where we'll discuss File Control Blocks, directory entries, and numerous other topics relating to the TPM-II file mechanism.

The name of each file is printed, followed by a hexadecimal display of the directory contents for that file, printed in eight groups of four hexadecimal digits that represent the entire 32-byte contents of the directory entry. The directory entries for multi-extent files are not always grouped together. The following example shows the files PIP.SYS and LIST.SYS on the current disk.:

```
A>FILES
PIP     SYS
 A0504950  20202020 20535953 0000001F 02030000 00000000  00000000
00000000
LIST    SYS
 A04C4953  54202020 20535953 0000000C 06000000 00000000  00000000
00000000

A>
```


ClibPDF - www.fastio.com

The DUMPER Utility

The DUMPER utility displays the contents of a file in both hexadecimal and ASCII notation. The format of the display is very similar to the ZAPPLE "D" command. DUMPER directs its output to the List device instead of the Console.

The contents of the file are displayed in rows, sixteen bytes at a time. Both hexadecimal and ASCII values are displayed. The address at the beginning of each line is the offset from the beginning of the file. DUMPER prints the name of the file being displayed as well as the current date and time at the top of every page, to make it easier to identify the listing at a later date.

The following example shows the first few lines from the output of DUMPER displaying the file ZDDT.SYS:

```
A>DUMPER ZDDT.SYS

                ZDDT                08/02/83  20:02

  0:   31  80  01  2A  06  00 7C D6 08 32 3A  01  67  2E  00  E5
1..*..|..2:.g...
 10:   EB  21  00  02  E5  21 00 03 06 08 E3  4E  23  E3  79  1F
 20:   4F  3E  00  D2  29  01 3A 3A 01 86 12  13  23  05  C2  1E
O>..).::....#...

END CHAPTER 9
```


ClibPDF - www.fastio.com