TPM - II USER'S GUIDE

© 1983 Rising Star Industries

ClibPDF - www.fastio.com

TPM-II

USER'S GUIDE

&

Programmer's Reference Manual

Copyright (c) 1984
by Rising Star Industries, Inc.

ClibPDF - www.fastio.com

PART I

ClibPDF - www.fastio.com

PART II

ClibPDF - www.fastio.com

CHAPTER 10: TPM-II SYSTEM CALLS

ClibPDF - www.fastio.com

# # #

ClibPDF - www.fastio.com

CHAPTER I

INTRODUCTION

Welcome to the QX-10 and TPM-II. You're probably already familiar with Valdocs and the QX-10, and are aware of their power and flexibility. The following manual will now introduce you to the <u>TPM-II operating system</u>.

WHAT IS TPM-II?

TPM-II is an operating system, and in a moment we'll explain exactly what that means and why it's needed. For the present, let's simply state that TPM-II is the software "base" upon which Valdocs runs.

Your QX-10 runs <u>application programs</u>. Valdocs is an applications program, albeit a large one, that encompasses many different applications or jobs (e.g., editor, scheduler, graphics, electronic mail).

TPM-II is already contained on the Valdocs disk, and it works "invisibly" between Valdocs (the application) and the QX-10's hardware. With the exception of using the <E>xit to TPM-II option of the Valdocs MENU key, you never "see" TPM-II. It just sits there in the computer, quietly doing its job. So, why do you need this manual? Before we answer that question, let's examine the definition of an operating system a little more closely.

An operating system acts as a buffer of sorts between the computer and the application program (Valdocs or something else). TPM-II combines with the application program, and the two instruct the QX-10 how to perform the application at hand.
Perhaps an analogy is in order here.

It's common knowledge that the boys in Detroit stretch their design talents and budget by producing several car models which share the same chassis. The difference between a Camaro and a Firebird or a Mustang and a Capri is the sheet metal on the outside: beneath it all, they share a common chassis. TPM-II is the software "chassis" of the QX-10; the various application programs that make up Valdocs comprise the outside sheet metal that distinguishes one model from another. We'll discuss this distinction in greater detail later in the chapter, but for the moment let's divert our attention and see just how it is that you go about using TPM-II.

ClibPDF - www.fastio.com

HOW DO I USE TPM-II?

The first step in using TPM-II is to read the important parts of this manual, which will familiarize you with the operating features of TPM-II. The last section of this chapter outlines the manual contents. We have broken this guide into two parts: the first being for most users, the second containing technical material relevant to programmers. Don't worry that you may have a fortnight of reading ahead of you before you can use TPM-II--once you have read the first section of this manual you should be conversant with most of the necessary information.

The first step to becoming familiar with TPM-II is to become familiar with your application program. We'll assume that by now you're well acquainted with Valdocs. However, you might occasionally use an applications program other than Valdocs. On this score we can't give you much help, save this one piece of advice. Read your manual carefully. The time you invest in familiarizing yourself with the features and requirements of an application program will return ten-fold dividends.

One final point needs discussing here. The TPM-II operating system is very similar to, and completely compatible with the CP/M operating system. CP/M is the dominant operating system for 8-bit microcomputers. If you'll be using other applications programs in addition to Valdocs, your program's manual will undoubtedly mention that you will need a microcomputer running CP/M in order to use the program. TPM-II is not only compatible with CP/M, but it has many additional features. Therefore, be assured that you are not about to commit a grievous blunder when you attempt to run your non-Valdocs application program on your QX-10 with TPM-II.

Finally, a note to those readers who are "second generation" computer owners and have owned or used a microcomputer which ran under CP/M. Since the two operating systems are so similar, you will probably be able to start using TPM-II right away. We have included a TPM-II command summary in Appendix A, which should help you identify the slight differences in syntax and command names. However, we still strongly urge that you read this manual before you use your QX-10 too much. The enhancements which TPM-II provides over CP/M are significant, and you would be selling the system short if you didn't become familiar with them and use them.

WHAT'S AN OPERATING SYSTEM ANYWAY?

Let's dig a little deeper. An operating system is a specialized program that is an integral part of any computer, whether it is a personal computer or an IBM mainframe the size of a garage.

ClibPDF - www.fastio.com

This system provides the "housekeeping" functions necessary on any computer as well as providing the interface between an application program and the computer itself.

The phrase "application program" implies that the computer program which you paid so dearly for is going to assist you in some "application." But what is an application anyway? An application can best be described by the end result of the program. Accurately representing the financial condition of a business is the end result of account programs. Creating a printed document is the end result of the Valdocs editor. Providing factual answers to "what if" questions is the charge of a "spread-sheet" program. In short, an application is really the end result of any computer program.

In order to accomplish these applications, the program(s) must perform certain steps. For example, the accounting program must accept financial data, run certain checks on that information, and print the desired reports. Likewise, a word processing program (the Valdocs editor) must allow you to enter text, make corrections, and finally print the document. Each of these steps are called <u>tasks</u> in computerdom.

The scope of a task can be quite large or quite small. If you were a programmer, you could define a task as printing a report or printing a character. When we discuss operating systems, the scope of our definition of tasks tends to be narrow, on the order of "print a character."

Many applications have common tasks. One might even say that certain tasks are "universal" to applications programs. After all, what good would any program be if you couldn't enter data or commands from the keyboard and have data or prompts displayed on the screen? Or have the results of the computations printed on paper for posterity? Or store data for future use? Well, you get the picture - there are certain tasks, fairly narrow in scope, which are common to almost any application program.

These tasks are what an operating system is all about. The mechanics of inputting and outputting characters to the printer, screen, or keyboard fall under the supervision of TPM-II. Likewise, storing and retrieving data from files on the disks falls in its domain. These tasks are executed for all application programs. The unique tasks of each application, such as balancing ledgers, proportionally spacing characters, etc. form the substance of application programs.

TPM-II ANATOMY

In actuality, TPM-II is made up of two separate groups of programs: a <u>nucleus</u> and a set of programs called <u>utilities</u>. The

ClibPDF - www.fastio.com

nucleus performs the tasks outlined above -- controlling the input and output of characters as well as supervising the storage of information on the disks. This nucleus, though basically invisible during normal use, is present at all times -- providing the necessary interface between all programs and your QX-10.

In addition to the nucleus, there are the utilities -- programs that are similar to the application programs. Their purpose can best be described as the "care and feeding" of your QX-10, its disks, and the stored data.

Throughout the remainder of this manual we will focus on these utilities; they are the portion of TPM-II which you will use directly. Of course, the utilities use the nucleus just like your applications programs do, so technically you are using both. In actuality, however, you use the utilities, so for the balance we will have little else to say about the nucleus.

HOW TO USE THIS MANUAL

The manual is divided into two parts. The first section is a general introduction to TPM-II, intended for all users. We will cover all of the basics necessary to run an application program with TPM-II, using an approach that assumes no prior knowledge of operating systems, computer concepts, etc. We will guide you through the basics as quickly as possible so that you can get on with using your application programs on the QX-10. If you plan to use your QX-10 for running Valdocs and those application programs which you purchase, Part I is all you'll have to read.

Part II is intended primarily for programmers and others who need to "look under the hood" of TPM-II and see how it works. The material is fairly complex, and is written for individuals with considerable computer experience.

We will conclude this chapter with a sneak preview of the material that will be covered in the rest of Part I.

Chapter II contains an overview of several hardware and software concepts that should be understood in order for you to to get the most out of TPM-II.

Chapter III gets you started with TPM-II basics. We begin by examining the TPM-II file structure, including file naming, user numbers, and protection levels.

In Chapter IV we cover file-oriented commands and utilities, those parts of TPM-II which deal with individual files. We show you how to determine what files are on a disk, how to rename or erase a file, and the way in which you set or change various file attributes.

ClibPDF - www.fastio.com

Chapter V extends our field of vision, as we examine the utilities that operate on a disk, instead of just one file. We will show you how to initialize a disk, transfer files between disks, and determine how much free space remains on a disk.

Part I concludes with Chapter VI, in which we discuss TPM features other than those that deal with files or disks. We tell you how to use the QX-10 real-time clock, print a file, and how to take advantage of the TPM-II batch processing mode.

A WORD OF ENCOURAGEMENT

As you venture further into TPM-II, you will find that the material goes quite quickly. Time, in addition to money, must be spent on a personal computer, in order to get the most out of it. The time you spend here should be regarded as the second part of your investment in your QX-10. It will repay you many times over as you use your application programs to their fullest potential.

ClibPDF - www.fastio.com

CHAPTER II

THE QX-10: HARDWARE AND SOFTWARE CONCEPTS

At first glance, this chapter may seem to represent an unnecessary digression, so let's explain why it's been included. Valdocs is a completely integrated package -- you simply turn on the QX-10, insert a disk, and voila! Everything you need is right there at your fingertips.

Unfortunately, running application programs is not quite as simple. Any "outside" application programs you purchase won't perform all of the functions of Valdocs. And that's what TPM-II is all about. When you use Valdocs, you're like a theater-goer watching the production unfold on stage. Congratulations -- you've just been promoted to director. You must now familiarize yourself with each character in the script and cue them when they are to make their entrances. This chapter will introduce you to the players.

The documentation accompanying most application programs deals strictly with that program. We'll take a step back and provide an overall look at how the QX-10 operates. You should be able to relate this information to the discussions in the next four chapters.

QX-10 HARDWARE

The QX-10 can be broken down into three basic types of components: the Central Processing Unit (CPU), which is the "brains" of the system; the memory, which in the QX-10 stores programs and data for the CPU; and the Input/Output (I/O) devices, which are the CPU's contact with the "outside" world. Let's look at each of these individually and define their functions in greater detail.

The CPU

The CPU makes a computer compute. The QX-10's CPU is a Z80 microprocessor. This special integrated circuit executes 150 or so <u>instructions</u> or program steps. Using these instruction, a programmer can write a program to perform any number of application programs. The diversity and power of the microprocessor allows your QX-10 to perform word processing, accounting, forecasting, graphics, and a host of other applications.

ClibPDF - www.fastio.com

Memory

The QX-10 memory contains both program steps and data. In the process of executing a program, the CPU reads the program instructions in memory, and manipulates the data as instructed. Memory is measured in bytes. Each byte is sufficient to store a character; for example, the word "byte" would require 4 bytes of memory.

Since the average microcomputer has thousands of bytes of memory, a shorthand notation, kilobytes, is used. A kilobyte or Kbyte is 1,024 bytes. (1,024 is a "round number" in binary arithmetic just as 1,000 is a round number in decimal arithmetic.) Your QX-10 can contain anywhere from 64K to 256K of memory. When you use Valdocs all of the memory in your computer is used. However, most application programs only use 64K when run under TPM-II (this is not a limitation of TPM-II, but rather of the CP/M computers which these application programs were designed to run on). Therefore, we will discuss the available memory in your QX-10 as if there was only 64K.

Your QX-10 has three different types of memory, each with different characteristics and each serving a different purpose. The majority of memory is of a type called RAM, an acronym for random access memory. This is a fancy way of saying data and program steps can be stored and retrieved from this memory.

The second type of memory, ROM, can only be read from., which pretty much eliminates data storage. ROM is primarily used for storing program instructions. However, ROM does have one big advantage over RAM: the contents of RAM are erased when the power is turned off, while the contents of ROM are stored permanently. We will see the advantage of ROM in the next chapter when we look at how TPM-II gets "started" or booted.

The third type of memory, CMOS RAM, is a special low-power type of RAM drawing so little current that a battery can be used to power it when the main power to the rest of the QX-10 is turned off. This RAM is a cross between RAM and ROM since it can be used for data storage (the CPU can read and write to it), yet its contents aren't erased when the power is turned off.

From the user's point of view, the two QX-10 disk drives should be included in our discussion of memory. However, from the CPU's point of view, the drives are Input/Output devices, not memory devices, so we will save our discussion of them for the next section.

The next chapter covers memory in greater detail, including what you need to know about the physical aspects of memory (e.g., storage capacity, data loss during power downs), and memory

ClibPDF - www.fastio.com

usage. Since memory can be used for both storing data and programs, the CPU must divide up the available memory between each of these functions. In addition, both the TPM-II nucleus and your application program must share this memory at the same time.

Input/Output

Input/Output is perhaps the easiest component to understand, since it involves the user. I/O is the link between the CPU's execution of a program and you (the world outside the CPU and memory). I/O includes the screen, keyboard, printer, floppy disk drives, and any other peripherals you may add to the QX-10.

The QX-10 is flexible. It can be connected to many different types of peripherals that enable it to communicate with you or the outside world. Some of the peripherals are an integral part of TPM-II (such as the keyboard and screen). Others are not, and come with their own special application program. For example, a modem is a peripheral commonly added to QX-10s. It is not, however, a standard peripheral from TPM-II's point of view, so it does not interface with it. Instead, you must use a special communication application program to get the modem to function correctly with the QX-10 under TPM-II (of course you can run the modem without difficulty with Valdocs even if you use TPM-II for other applications).

The peripherals or I/O devices with which TPM-II interfaces are the keyboard, screen, printer, and the two floppy disk drives -- all part of the standard QX-10 system. An optional peripheral, which is supported by TPM-II, is a hard disk (often called a Winchester disk). Let's take a moment and see what each of these peripherals do.

Keyboard: The QX-10 is available with two different keyboards: HASCI and ASCII. The HASCI keyboard is designed especially for Valdocs. The ASCII, on the other hand, is designed for more general application programs. Either keyboard will work with TPM-II. In general, the keyboard is used for inputting all commands to TPM-II and most of the application programs you will be running. Remember that when you are using TPM-II the special Valdocs keys on the HASCI keyboard (the four groups of keys across the top of the keyboard) will not work with either TPM-II or your application program.

Screen: The screen will be used by both TPM-II and your application program to display prompts, data, and other information. The keyboard and the screen together are sometimes referred to as the console in various application programs, which simply refers to the I/O device(s) that the program can communicate with you. The QX-10 has some very powerful graphics capabilities that can be fully utilized under Valdocs, but you will find that most

ClibPDF - www.fastio.com

purchased application programs will not use them. Therefore, you should refrain from using the GRAPH SHIFT key to input these special characters unless you are certain that the application program you are running will take advantage of them.

Printer: The printer is used primarily for displaying the results of a program's computations. For example, an accounting program will print various reports once all of the data has been entered. A printer is primarily used with application programs. With the exception of providing a printed copy of what is displayed on the screen when necessary, the printer has little to do with TPM-II.

Floppy Disk Drives: In a sense, the floppy disk drives are the center of the TPM-II world. The various data and program files necessary for your application programs to run are stored here. As you will see in the following chapters, most of the features and commands of the TPM-II operating system have something to do with disk drives and files.

Hard Disk Drive: A hard disk or Winchester disk drive is a first cousin to floppy disk drives. It performs the same functions, but has two characteristics that make it more desirable than a floppy disk: first, it stores considerably more information; second, it is considerably faster in storing and retrieving data. (A floppy disk stores up to 376K of user data, while a hard disk can store 5 Mbytes --five megabytes or 5 million -- to 20 Mbytes.)

The final I/O device pertaining to the TPM-II operating system is the Clock-Calendar. The QX-10 has a special integrated circuit that keeps track of the correct date and time. The clock-calendar is connected to the same battery as the CMOS RAM so that it continues to run, even when the power is turned off.

During the course of this manual we will refer to the components we have just examined. Hopefully, this brief introduction will provide sufficient background.

SOFTWARE

As difficult as hardware concepts and terms can be to wrestle into a state of comprehension, they often seem simple compared to the nefarious "software" side of your QX-10. At least hardware presents you with a physical manifestation. Software, on the other hand, is either "the stuff on a disk" or, even worse, something which floats around out there in the Ether. It's easy to see why your printer won't work properly when there's no paper in it. But when software fails to work as we expect it to, we find ourselves almost helpless to make it behave.... Unless we understand the component parts of software!

ClibPDF - www.fastio.com

We do not mean to imply that software is a generically defective product. Rather, a lack of understanding of software often causes us to make unreasonable demands upon it (it's like asking our printer to make perfect copies of a letter without paper in it).

Software is not usually thought of as containing separate components. But just as your QX-10 has a separate keyboard, screen, and printer, so your software has discernible parts. During the remainder of this chapter we will attempt to introduce you to this "software hierarchy."

We will discuss four levels of software, each having certain characteristics that separate it from the others. These groups and their characteristic are outlined below.

Machine Code

Machine code is software at its most basic level, the 1s and 0s that the CPU reads. Machine code, or Object code as it is sometimes called, is the actual stream of bytes the QX-10 reads off the disk and that the CPU executes. All programs must eventually be reduced to this level in order to run.

Machine code is readable by the CPU, which is great from the CPU's point of view, but it would make life difficult if you were a programmer and had to write programs that looked like

```
10110101  10011010  10110011  10100000  10110001  01000001
```

You get the picture. This is perfectly readable to your QX-10, but even an experienced programmer would have a difficult time interpreting it.

Computer languages provide the solution to this dilemma. A computer language is like a Berlitz guide for humans travelling inside a computer. It converts commands and instructions (such as PRINT "PRESS RETURN WHEN READY"), which are discernible to humans, into the 1s and 0s that keep your QX-10 humming merrily along.

There are three levels of translation, which, added to the machine code, give us our four-tiered software hierarchy. First, there is assembly language, then high-level languages, and finally, application programs.

Assembly Language

If you've ever read an introduction to a book that was translated from a foreign language, you know that the translation is almost always mentioned. In some cases, the translation may be

ClibPDF - www.fastio.com

criticized as being too literal, with the result that the cadence of the original language comes out in stumbling sentences, and idioms sound like they came in from left field. The other extreme is the translator who takes such liberties that he or she is, in essence, a plagiarist.

Assembly language is a literal translation. A program written in assembly consists of instructions which have a one-to-one correspondence with machine code. The instructions in assembly language are called <u>mnemonics</u> (a mnemonic is a device designed to aid the memory). The mnemonics for assembly language are obtained by freeze-drying a phrase which describes a particular CPU instruction. For example, one of the instructions which the Z80 microprocessor in your QX-10 understands is "JUMP TO MEMORY LOCATION X." The mnemonic for this instruction is

```
JMP X
```

What the assembly language does is translate this instruction or mnemonic into machine code, in this case:

```
11000011 10000001 00100011
|      | |               |
-------- -----------------
    |           |
"Jump to" "memory location 33,305"
```

You needn't worry too much about assembly language. Unless you are a professional programmer, you probably won't do any programming in assembly language. We will therefore, leave this topic to the Pros from Dover and proceed to high-level languages.

High-Level Languages

High-level languages are the next step up. They cater to the human side of a programmer rather than the CPU. The instructions are understandable and the languages tend to include instructions which perform larger tasks than would an assembly language mnemonic. For example, the instruction

```
PRINT "PRESS RETURN WHEN READY"
```

is a valid instruction in the BASIC language and would print the prompt enclosed in quotes on the screen.

The actual execution of this single instruction would take hundreds of CPU instruction. Therefore, it would take many assembly language instructions to execute this same task if you weren't programming in a high-level language. Which brings us to the advantage of high-level languages: each instruction accomplishes far more than an assembly language instruction. Since

ClibPDF - www.fastio.com

the tasks each high-level language instruction executes are more extensive, a programmer may program an application with fewer instructions, or instructions that are too broad in scope for effective assembly language programming.

Many high-level languages are available for your QX-10. BASIC is the one most commonly used, but FORTRAN, PASCAL, COBOL, and others are among your choices. Of course programming in a high-level language on your QX-10 is entirely optional, but many individuals other than professional programmers are likely to engage in it. A number of good books on programming are available in most computer stores and we recommend you read up on it if you are at all interested.

Application Programs

If you read politely through the last few sections, all the while thinking "this is well and good, but I don't want to program!", then the following is for you. Applications programs are the last stop. They take the steps necessary to perform the things you want to do with your QX-10 (such as spreadsheets or accounting), and convert these tasks into high-level or assembly language instructions so that the CPU can understand what it's supposed to do.

Instead of telling the computer what it is supposed to do (programming) you are simply entering data and informing the computer of tasks you would like performed. Ninety to one-hundred percent of the time spent in front of your QX-10 will involve using application programs of one kind or another (Valdocs is an application program).

In the next four chapters we will introduce you to the TPM-II commands and file structure. The information we have just presented should make this journey a little easier.

END CHAPTER 2

ClibPDF - www.fastio.com

CHAPTER III

TPM-II BASICS

The introductions are out of the way and it's time to roll up our sleeves and start using TPM-II. We will begin by describing how to get it started, files and filenames, and the overall way in which TPM-II works.

TPM-II COMPONENTS

Let's take a moment to review the two components of the TPM-II operating system: the nucleus and the utilities. The nucleus is the basic part of TPM-II which remains in memory at all times. It executes common I/O and file management tasks for application programs, as well as several rudimentary system management or "housekeeping" tasks.

The second portion of TPM-II are the various utilities which provide the advanced system management tasks. These utilities allow you to make new disks, back-up copies of your files, print files on the printer, etc. They are, in fact, nothing more than specialized programs designed to help you get the maximum use out of your QX-10.

Due to the extent of the functions these utilities provide, they require more instructions than can be stored in the memory of your QX-10 at one time. Therefore, they are stored on a disk, just like your other application programs. In this respect, these utilities differ from the rudimentary commands contained in the nucleus of TPM-II. These commands are simple enough that they can all be stored in memory at the same time, and still leave sufficient room for your programs to run in.

The importance of all of this is that the nucleus commands are available to you at all times, whereas the utilities are available only if they are on one of the disk currently in your QX-10. We will make the distinction between the commands executed by the nucleus of TPM-II (which we will henceforth call resident commands) and utilities as we discuss each one.

STARTING TPM-II

In order to run an application program under TPM-II, you must start TPM-II up, or boot it. The process is essentially the same as starting Valdocs: reading the program into the internal memory of your QX-10 from a disk.

If you are using Valdocs, TPM-II is already in memory. TPM-II performs the I/O and file management tasks for Valdocs, as it does for other application programs. In other words, in order to run a application program, the program is simply substituted in memory for Valdocs. In order to exit Valdocs, press the MENU key which is the left-most of the four keys labeled "APPLICATIONS." Valdocs will display a menu of options, and you should select "EXIT VALDOCS TO TPM-II." (Note: You must be in the "Intermediate" or higher level in Valdocs is order to exit to TPM-II. In the "Beginner" level, this option will not be displayed on the menu. In order to change levels, you must run the SETUP program (see pages 6 through 8 of your Valdocs Users Guide).

There is a limitations to entering TPM-II in this manner; it has to do with the distinction between resident commands and utilities, which we mentioned earlier. The Valdocs System Disk, which you insert in the left-hand drive, has just enough room for the Valdocs program files. At best, you will be able to squeeze one or two utilities on the disk. Almost any application program you wish to run will not fit in the remaining space left on the Valdocs System Disk. The net effect is that you end up sacrificing a disk drive and get nothing in return.

The preferred method is to have a different disk for your application program. This disk will contain both TPM-II and the application program(s) itself. With all but the largest application programs, you should have plenty of room left over for the TPM-II utilities. Starting TPM-II is exactly the same as starting Valdocs. First, turn on your QX-10, or press the RESET button (located below the right-hand disk drive). In a few seconds the familiar message

```
INSERT DISK
```

should appear on the screen. Insert a disk with TPM-II in drive A (the left-hand drive, just as with Valdocs). If this the first time you are using TPM-II you won't have a disk with TPM-II <u>and</u> an application program on it. We will show you how to make new work disks in Chapter 5. For now, let's use a <u>copy</u> of the TPM-II master disk which you received in the back of this manual.

****

USE A COPY, NOT THE ACTUAL MASTER DISK!! Use the Valdocs COPY DISK feature to make a copy of the master disk.

****

Once you have inserted the disk in drive A and a few seconds have elapsed, TPM-II will print the sign-on message:

ClibPDF - www.fastio.com

```
                    **** BEGIN CRT ****

             TPM-II for QX-10 256K V2.XXF 03/24/83
             Serial Number:000001

             A>

                     **** END CRT ****
```

This message may vary slightly with subsequent versions, but in general it will contain the version number, the date the version was released (not the current date), and your serial number. Immediately below that, you will see an innocuous little thing called the system prompt. The system prompt simply means that TPM-II has been loaded into memory and is waiting for your command.

DISK DRIVES

The QX-10 has two disk drives, labeled A and B. That may not sound like the most profound statement you've ever heard, but it's important. You must use a drive label in naming files, as well as many of the commands and utilities we will be discussing shortly. The drive label is followed by a colon, ":" if it attached to a filename or part of a command. For example, let's say you have a file named MYFILE (we will discuss file naming conventions in detail next, so hang on for a moment), and let's also assume you have a copy of it on both drives. The file on drive A would be notated A:MYFILE, and the one on drive B, B:MYFILE.

In addition, there is the current drive to consider. The current drive is the drive label TPM-II will "fill in" for you if you leave out the drive label. The current drive is indicated in the system prompt. The prompt A> means that drive A is the current drive, while a B> would indicate drive B. Each time you boot or start TPM-II, drive A is automatically selected as the current drive. Thus, if you boot the system and simply enter MYFILE, TPM-II will interpret this as A:MYFILE.

In order to change from one drive to another, simply enter the drive label (don't forget the colon) of the new current drive after the system prompt. When the new system prompt is displayed, the drive label of the new drive will be displayed:

```
A>B:
B>
```

FILES AND FILENAMES

All information that is stored on a disk is stored in files. This applies not only to data, such as characters in a letter, numbers in a balance sheet, names in a database, but also to program instructions. We make the distinction based on whether the information stored is of value to us (data) or to the QX-10 (programs). They are all the same to TPM-II.

Naming Files

Each file must have a name. Files are given names when first created, either automatically by a program, or by you. The name of each file consists of two parts, a filename and a filetype. A filename can be up to eight characters in length, and can contain any valid character except = . ; : ? * [ ] _. The filetype has the same character limitations, and can be up to three characters in length. The following are example of acceptable file names:

```
CHPT1.TXT          LETTER.TXT
PAYROLL.DAT        FORECAST.'83
MYFILE             LIST.COM
```

Any combination of a valid filename and filetype can be used to name your file. However, there are some "conventions" of which you should be aware so that you don't inadvertently get into trouble. The filename is usually selected to identify the contents of the file, while the filetype indicates the type of file.

Let's start with the type of file. Files can contain different types of information: some contain programs, data files with accounting or other financial data, text files containing letters, etc. Each type should be designated by a different filetype. Once you start using an application program, there'll be several types of files that will be used over and over. Make up a set of filetypes you will use on all files of a single type, so that you have some consistency throughout your files. It doesn't really matter what these filetypes are (within limits), provided they mean something to you and they're applied consistently.

There's one small exception there. Certain filetypes are commonly used by certain application programs. These are the conventions we alluded to earlier. Even if you aren't currently using an application program which uses a certain "reserved" filetype, it is best not to use a filetype in an unconventional manner. You never know what application programs you might be using in the future. The commonly encountered filetypes are listed below:

ClibPDF - www.fastio.com

Filetype	Common Usage
.ASC	ASCII source file
.ASM	Assembler source file
.BAK	Back-up file created by a text editor
.BAS	BASIC source file
.COM	Command file (program) for programs compatible with CP/M 2.2
.CTL	Control file for batch processing
.DOC	Document file
.FOR	FORTRAN source file
.HEX	Intel hexadecimal format object file
.IMG	Image of the operating system
.INT	BASIC internal format file
.LIB	Library file
.LNK	Linker command file
.MAC	Macro assembler source file
.PLT	Pilot program
.PRN	Print file
.REL	Relocatable object file
.SUB	SUBMIT program command file
.SYS	TPM-II System file (program). Also for programs compatible with CP/M 1.4.
.TXT	Text file
.$$$	Temporary file

Don't worry if some of these descriptions don't make sense to you at this point. Many of these filetypes are used only when programming, hence the user of application programs will seldom encounter them. However, several are used commonly by the TPM-II utilities and your application programs (i.e., .BAK, .$$$, .SYS). These should be memorized and used in context.

Filenames, on the other hand, are completely up to you. Suffice it to say that some consistent and rational scheme must be devised when choosing filenames. The true test of a filename is not whether or not you can deduce the contents from the filename the next day; pick your filenames so that you will be able to tell what's in a file six months from now.

One more word of advice. Even though TPM-II allows you to name a file with a blank or null filetype (i.e., BADFILE), it is <u>not</u> recommended. You will find that providing every file with a filetype is well worth the effort.

Wildcards

A rational naming scheme for files will benefit you in more ways than simple memory retention. Files that have been properly named will lend themselves to the use of <u>wildcards</u>. Before you

ClibPDF - www.fastio.com

conjure up visions of smoke-filled rooms, bourbon, and poker, let us assure you that wildcards have everything to do with files and nothing with cards. TPM-II has two wildcards: the "?" and "*" characters. In general, wildcards act much like wildcards in poker (hence their name) in that they may assume the value of any character.

The "?" will match any single character. For example, let's say you have the payroll data for a company stored on a disk. There are twelve files, PAY-01.DAT, PAY-02.DAT, etc., through PAY-12.DAT. If you wanted to copy these files from one disk to another, you could list each one by name, but this would require that you enter each of the twelve names. Instead, you could simply enter PAY-??.DAT. PAY-??.DAT would reference the following files:

```
PAY-01.DAT          PAY-02.DAT          PAY-03.DAT
DAY-04.DAT          PAY-05.DAT          PAY-06.DAT
PAY-07.DAT          PAY-08.DAT          PAY-09.DAT
PAY-10.DAT          PAY-11.DAT          PAY-12.DAT
```

Whereas the "?" will match any single character, the "*" will match any number of characters. In essence, the "*" is the same as saying "fill with '?'s." For example, *.TXT is equivalent to ????????.TXT. In practice, the "*" wildcard is used to reference all files with a common filename or common filetype. If you wish to reference all text files on a disk (filetype .TXT) then you can enter *.TXT. Likewise, if you want a set of files with the same filename, say MYFILE, simply type MYFILE.*. Another useful combination, *.*, references all files on a disk.

In the next chapter we will present the commands and utilities that deal with files. Many examples of the wildcard's use and utility occur.

User Groups

The files on a disk are grouped into <u>user groups</u>, a simple and effective way to keep all the files for different applications or projects separate. TPM-II allows up to 256 different user groups to be assigned. Each time you boot TPM-II you will automatically be placed in User Group 0.

Changing from one User Group to another is similar to changing the current disk drive. Simply enter the new User Group number, followed by a colon. For example to change from User Group 0 to 5, you would enter:

```
A>5:
```

The current User Group isn't readily apparent from the system

ClibPDF - www.fastio.com

prompt, so if you forget which one you're in, type:

```
                A>USER
```

TPM-II will respond with the current User Group number.

Different User Groups can be active on different drives at the same time. For example, drive A may be in User Group 0 while B is in Group 3. Each can be set independently of the other. You can also change User Group numbers at the same time that you switch the current drive. Let's assume you have just booted TPM-II so both drives A and B are in User Group 0. If you want to switch to drive B, User Group 7, simply enter:

```
                A>B7:
                B>
```

Protection Levels

The final file attribute is Protection Level. TPM-II supports seven different protection levels. In essence, the various protection levels limit how much a file can be "modified" or changed. For example, you can protect a file from being erased from the disk, changed by adding new data, or having data deleted from it.

Table 3-1 summarizes the characteristics of each level, along with providing a description of their common usage. For example, a program file would never be modified, so it would be placed in a higher protection level than a data file that is constantly updated.

(See next page for the table)

ClibPDF - www.fastio.com

**** TABLE 3 - 1 ****

Level	Protection	Common Uses
0	No protection	Data files which undergo constant updating.
1	File marker; no protec-tion	Same as Protection Level 0.
2	File can't be replaced	Data files which you want to update but don't want to be copied over
3	File can be written to	Data files which you want to update but don't want to be "copied over".
4	Data can be added to but not deleted from	Data files which you want to extend but don't want data deleted or "edited".
5	File can be read by a program and can't be erased	Certain application program files.
6	File can only be executed and can't be erased	Application program files of type .COM or .SYS. File is invisible.
7	File can't be executed	File is invisible and not accessible or read. Not commonly used.

******

ClibPDF - www.fastio.com

When a file is first created, it has no protection, or a Protection Level of 0. In order to give a file a higher Protection Level, you'll use the PROT command. Type PROT, followed by the name of the file, and the new Protection Level in brackets. In order to change the file MYFILE to Protection Level 5, enter:

```
A>PROT MYFILE <5>
```

The current Protection Level of each file is displayed when you use either the DIR or LIST commands. DIR and LIST allow you to see what files are on a particular disk. We'll go into these commands in depth in the next chapter, but for now let's just look at a simple example of using the DIR command. If you enter

```
A>DIR
```

a list of all files on the current disk drive (in this case drive A:) will be displayed on the screen. You'll notice the numbers in brackets which follow each file; they're the current Protection Level of each file.

**** BEGIN CRT ****

```
A>DIR

SYSGEN   SYS <5> | FORMAT   SYS <5> | LIST     SYS <5> | FILES   SYS <5>
ZPIP     SYS <5> | ZDDT     SYS <5> | SUBMIT   SYS <5> | COMPARE SYS <5>
COPYDISK SYS <5> | SET-TIME SYS <5>
```

**** END CRT ****

The screen above is a directory of the files of the TPM-II master disk which you received with this manual. If you want to try it out, use the copy of this disk which you made earlier. Since these files are all programs, they have been set to Protection Level 5.

Summary Of File Attributes

Let's take a moment to summarize the TPM-II file attributes before we continue:

* Each file must have a name, consisting of a filename containing up to eight characters and a filetype of up to three characters.

* The filename can be anything you choose, but to avoid future problems you should only use "conventional" filetypes.

* Each file name needs a drive reference. If you leave it off, TPM-II will "fill in" the current drive as the drive reference.

ClibPDF - www.fastio.com

For example, if the current drive is B: the file name MYFILE is equivalent to B:MYFILE.

* A file is placed into a User Group. TPM-II allows up to 256 different user groups. Each disk drive may have a different User Group selected.

* A drive label and User Group may be combined together. B7: will change to drive B:, User Group 7.

* Each file has a Protection Level associated with it. There are seven levels, each with its own characteristics.

RUNNING A PROGRAM WITH TPM-II

Running your application program is quite simple. Each program is stored in a file. Program files must have a filetype of either .COM or .SYS. This tells TPM-II that the contents of the file are program instructions that the CPU can execute directly (machine code). You'll notice that the utilities on the TPM-II master disk all are either .COM or .SYS files. In other words, there is no difference between running an application program which you purchased and any one of the TPM-II utilities. We'll show you some examples of utilities, but remember that the same information applies to your application programs.

We'll use the LIST utility in the examples appearing throughout the rest of this section. The LIST utility is similar in function to the DIR command used earlier; LIST determines what files are on a particular disk. It displays files by filetype, i.e., all files with the same filetype are grouped together on the display. In addition, both the Protection Level and User Group number of the file are shown.

In order to start a program, simply enter the <u>filename</u>. The filetype and the period which separate the two are not entered! The LIST utility is stored in the file LIST.SYS. To run this program enter

```
        A>LIST
```

Remember the difference between a resident command and a utility. A resident command, such as DIR is always available. In order to run LIST, the file LIST.SYS must be on drive A:. If you don't have a disk with LIST.SYS on it in drive A: (the copy of the TPM-II master disk will have this program on it), insert it now. If you use the copy of the TPM-II master, the screen should resemble the one below when you run LIST:

ClibPDF - www.fastio.com

**** BEGIN CRT ****

```
A>LIST

   DRIVE A:        06/17/83       15:14:40

COM=> WIDTH      0:<5>  4K
SYS=> SYSGEN     0:<5> 18K  |  FORMAT     0:<5>  4K | LIST      0:<5>  2K
      ZPIP       0:<5>  8K  |  ZDDT       0:<5> 24K | SUBMIT    0:<5>  8K
      COMPARE    0:<5>  4K  |  COPYDISK   0:<5>  2K | SET-TIME  0:<5>  2K

Total Directory Entries = 10 [  11]
```

**** END CRT ****

Sometimes you'll want to be able to run a program from a disk that is not the current one. To do this, place the drive label of the disk the program resides on in front of the filename. To change the above example so that LIST.SYS will be read off drive B:, you would enter

```
A>B:LIST
```

If you don't remember which drive a program is currently on, TPM-II will come to your aid. If it can't find the program you requested on the current drive (or the drive you specify with the drive label), it will search all of the drives in your system to determine whether or not the file exists on another drive. When it finds the file, TPM-II will read it off the drive and execute it.

Communicating With Your Program

Certain application programs require a small amount of "meaningful dialogue" with their operators before they can properly execute their intended task. TPM-II allows you to enter a command line when you start a program. Instead of pressing the RETURN key after you have typed the program's filename, enter a space and then type your command line. A command line can be up to 127 characters in length, and can contain any character.

Let's use the LIST utility as an example. You can ask LIST to display data on certain files only, instead of the whole disk. If you only want to display the files containing text (filetype "*.TXT". This will instruct LIST to ignore all files that don't have the specified filetype.

```
A>LIST *.TXT
```

Different programs use this TPM-II feature for different reasons. Consult the documentation that came with your program to see if it uses the command line, and in what manner.

ClibPDF - www.fastio.com

TPM-II Control Characters

When you enter a command to TPM-II, the characters you type go into a buffer, a sort of holding tank. TPM-II doesn't actually interpret your command until you press the RETURN key. Several commands, consisting of control characters, allow you to edit or correct this buffer. A control character is entered by holding down a CTRL key (located on either side of the space bar), then typing any alpha character. Control characters are notated "CTRL-" followed by the alpha character. For example, control character S is notated "CTRL-S". CTRL-H is the backspace command. It will delete the last character you entered. CTRL-U will erase the entire line if you wish to start over.

In addition to these two editing commands, several other control characters are important. CTRL-C will reboot TPM-II. If you wish to exit any TPM-II utility, simply enter a CTRL-C at any input. The program will be terminated and the system prompt will reappear. The CTRL-B command is used to alert TPM-II when you change disks.

YOU MUST ENTER EITHER A CTRL-B OR CTRL-C EACH TIME YOU CHANGE DISKS!.

The CTRL-P command is used to turn the printer echo on or off. Typing CTRL-P the first time will tell TPM-II to copy everything that is printed on the screen from that point on to the printer. This is useful for obtaining permanent copies of directory listings. Typing CTRL-P a second time will turn the echo off.

One final note on special keys. Technically this key isn't a control character -- rather, it is a HASCI keyboard key having a different meaning outside of Valdocs. The MAR REL (margin release) key in the upper left-hand corner of the keyboard produces the ESCAPE code under TPM-II. You may very well encounter a reference to the ESCAPE key in your application program documentation; the ESCAPE key is almost universally standard on CRT terminals. Pressing the MAR REL key produce the proper ASCII ESCAPE code.

These are the basics of TPM-II. You should now be familiar with the basic operation of TPM-II, the file naming scheme used by TPM-II, and a couple of the basic TPM-II commands. In the next two chapters we'll cover all of the available commands and utilities.

END CHAPTER 3

ClibPDF - www.fastio.com