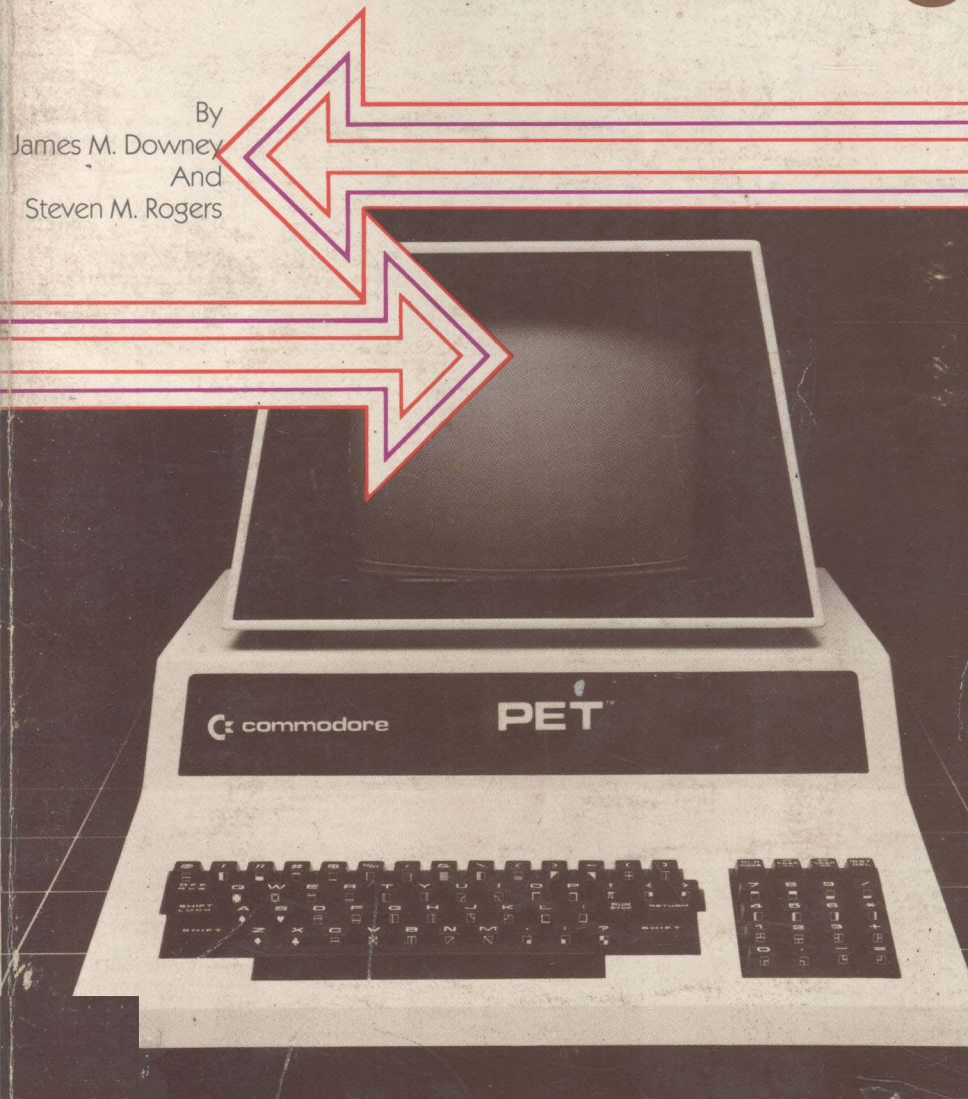
21795

PET® Interfacing

By
James M. Downey
And
Steven M. Rogers



BLACKSBURG CONTINUING EDUCATION SERIES™
edited by Titus, Titus & Larsen

PET® Interfacing

by

James M. Downey and Steven M. Rogers

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

PET is a registered trademark of Commodore Business Machines, Inc.

FIRST EDITION
FIRST PRINTING—1981

Inter
Libra

Cover

Printe

Preface

During the past few years, we have seen the small computer market grow remarkably as computers such as the TRS-80, APPLE, and PET became available. These small computers provided a great deal of computing power at very low cost, so they could be used in places where such computing power would have been prohibitively expensive just a few years before. While many users were able to use a computer by itself, many other users found it necessary to connect peripheral devices to their computer, using such devices as printers, analog converters, remote controllers, and others, for use in a wide variety of applications. Many people have found, however, that there are many devices that they would like to connect to their computer, but since standard interfaces do not exist, they have been faced with what seems to be a formidable task—designing and building an interface by themselves.

The purpose of this book on PET interfacing is to give you a better idea of how specialized interfaces can be built and used with this powerful computer system. BASIC language programs will be used throughout the book, so you should be familiar with this powerful programming language. Likewise, the designs use standard digital electronic components from the SN7400 (or equivalent) family, so you should at least have an understanding of what these devices do before you tackle the interface circuits that are provided for you.

Unlike many computers that simply provide generalized bus connectors, the Commodore PET computer has several special-purpose interface connectors that ease the job of interfacing the computer to "real world" devices.

These connectors provide access to the following input/output ports:

- The PET User Port.
- The Memory Expansion Port.
- The IEEE 488 Standard Interface Port.

Each of these ports has a specific use, and each is covered in detail in this book. Nowhere else will you find comparable information about interfacing the PET through each of these sets of signals.

Of particular interest is the IEEE-488 port. The signals provided for you at this port have been defined by the IEEE 488 standard, and many computer manufacturers and instrument makers have adopted this standard in their equipment. Thus, it is possible to easily interface these IEEE 488-compatible devices with a minimum amount of effort. In many cases, the interfacing simply involves the connection of standard cables between the various IEEE-488-compatible devices. Software is used to control the transfer of special information over the bus signal lines. Since more and more devices are being made available with IEEE 488-compatible I/O ports on them, we have paid special attention to the IEEE 488 bus. It is also possible to use this set of bus lines to interface devices to the PET, even if those devices do not use the standard bus conventions.

You should realize by now that this book is really a how-to-do-it book that covers many interfacing techniques for each of the three I/O ports. Each interfacing example is described in detail, with complete software and circuit details provided for you. There are many experiments that you can do to learn more about interfacing the PET computer, and to learn more about the control signals. The experiments in this book can be used with all versions of the PET computer that have a 25-by-40–character video display, including some of the older PET computers that have been upgraded with new read only memories (ROMs).

In the first two chapters, we will introduce you to the PET and guide you, step by step, through the construction of a simple breadboarding circuit that can be used to provide easy access to the signals that are available at the user port. This circuit is easy to put together using standard wire-wrap techniques, and you will find it to be very useful for further interfacing projects.

Chapter 3 covers many of the interfacing techniques, both elementary and advanced, that may be used to control the user port control signals. Included here are the instructions for using the BASIC-language commands that actually control the user port; PEEK, POKE, and WAIT. You will also learn the proper use of the AND, OR, and NOT operators to control I/O devices. When you have finished Chapter 3, you should have an excellent understanding of the use of the user port, and how the 6522 I/O chip is used in the PET. No assembly language programming is used in this chapter. How-

ever, assembly-language programmers should be able to quickly convert the example programs into the corresponding assembly-language program steps.

In Chapters 4, 5, and 6, we have taken a different approach to interfacing the PET, by using the memory expansion port. You will be able to use many of the interfacing techniques that were covered in Chapter 3. For example, you will learn about the various decoding techniques that are used to allow interfaces to select different I/O devices, and how the bidirectional data bus is used to transfer information to and from the computer and I/O devices. In Chapter 5, you will see how to construct a simple breadboard that can be used with the memory expansion port, and in Chapter 6, there are many experiments that you can do to reinforce your understanding of how the memory expansion port may be used to control practical interface circuits.

A summary in Chapter 7 provides some additional interfacing information, for example, how the PET may be interfaced to non-logic devices that may require either high currents, high voltages, or both. Information about analog converters is also provided.

Chapter 8 covers the use of the IEEE 488 bus on the PET. Detailed information describes the signals on the bus, and their uses. A special interface is also described; one that is used to control a printer. Since people will be interested in using the IEEE 488 bus for other applications, we have also provided a general-purpose interface circuit that will allow you to interface non-IEEE-488-compatible devices to the PET through the IEEE 488 bus. This is done by using another microprocessor chip to simulate the presence of a normal IEEE-488 I/O device on the bus. Complete details, including assembly-language listings and sources of parts, are provided for you. This general-purpose "listener/talker" can be used in many applications, since it is not specific to one device, or one type of interface.

We would like to thank the Blacksburg Group for their encouragement and assistance with this book.

JAMES M. DOWNEY
STEVEN M. ROGERS

Contents

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

APPENDIX A

APPENDIX B

APPENDIX C

CHAPTER 1

Introduction to PET Microprocessor Hardware

The heart of the PET computer is a 6502 microprocessor. The 6502 is a single large scale integrated circuit that incorporates, in one 40-pin package, all of the necessary logic circuitry required by a digital computer. The arithmetic and logical functions which the PET performs are all done within this single integrated circuit. If you peer under the PET cover, however, you will notice that it contains not just one, but dozens of integrated circuits. Some of these integrated circuits are memory "chips" while the others are so-called "support chips." The latter serve to provide the necessary logic to interface the 6502 to the memory, the keyboard, the video display, and to the other input/output ports which are present in the PET. The 6502 processor in the PET is a vivid testimony of the giant stride in technology which the computer industry has experienced in recent years. Only a decade ago the equivalent computing power would have cost many thousands of dollars and would have required hundreds of integrated circuits. Today, the 6502 can be purchased for under $20.00. Before we proceed with the fundamentals of interfacing various devices to the PET, it will be useful to understand what the 6502 is, how it processes information, and how it is used in the PET.

BITS AND BYTES

It is convenient for a computer to use a binary or base-two arithmetic system. Only two digits are used in such a system, one

and zero. Since digital electronics also have only two defined states (a logic high, plus 5 volts for TTL logic, and a logic low, zero volts for TTL), it is possible to let one state represent a one and the other state represent a zero. Traditionally, the high state is the one and the low state is the zero. In a binary system, numbers have successive digits of ones and zeros with the rightmost digit representing the one's place, the next digit to the left, the two's place, the next the four's place, and so on. The powers of two and their decimal equivalents are:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

For example, the number 010011 represents one 1, one 2, no 4, no 8, one 16, and no 32. Its decimal equivalent is 19, which is simply the sum of 16+2+1. The largest binary number that you could represent with 6 digits is 111111 which has a decimal value of 63. To represent a larger number would require more digits. In the computer, a binary number is carried by a group of electrical circuits, each assigned to represent a specific digit in the number. The electrical state, high or low, determines if any digit is a one or a zero. By convention, each such binary digit is referred to as a *bit*. The entire group of bits which comprise a number is referred to as a word.

The 6502 uses an 8-bit word length (8 binary digits). This means that, for most operations, the computer operates on 8 bits at a time. The 8-bit word length is often referred to as a *byte*. The term byte was originally introduced by IBM. Their computers used a 16-bit, or larger, word length; but for certain operations, such as handling ASCII data types, it often was necessary to manipulate only 8 bits at a time. Specific instructions were incorporated into the computer's hardware to perform operations on these 8-bit subunits and the 8-bit subunit of a word was called a byte. Today most microprocessors use an 8-bit word length. So even though eight bits represent a full word in those microprocessors the term byte for their 8-bit word remains popular.

Eight bits limit the maximum value which can be expressed by a byte to 255. When numbers larger than 255 are to be handled they must be operated on as two or more bytes. In programming such multibyte operations care must be taken to keep track of which byte represents which portion of the number. It is important not to get them confused. Nevertheless, with careful programming, as has been done in PET's BASIC interpreter program, the 6502 can handle numbers much larger than 256 with ease. The calculations, however, are still performed within the 6502 using only 8 bits per step.

Most computer systems, including the PET, consist of 3 basic parts. The first is the central processor (in this case the 6502) which we

have already described. The central processor does all of the arithmetic and logical operations for the computer. The other two parts are the memory and the input/output (usually abbreviated I/O) devices.

MEMORY

Memory in a computer serves two major functions. First, it is the place where the computer program is stored. Second, it gives the computer the ability to temporarily save information for future use. Memory in the PET is organized into 8-bit bytes, in order to correspond to the 8-bit word length of the 6502. The 6502 has the ability to randomly access up to 65,536 separate bytes of memory. This is done by placing a binary code on 16 signal lines called the address

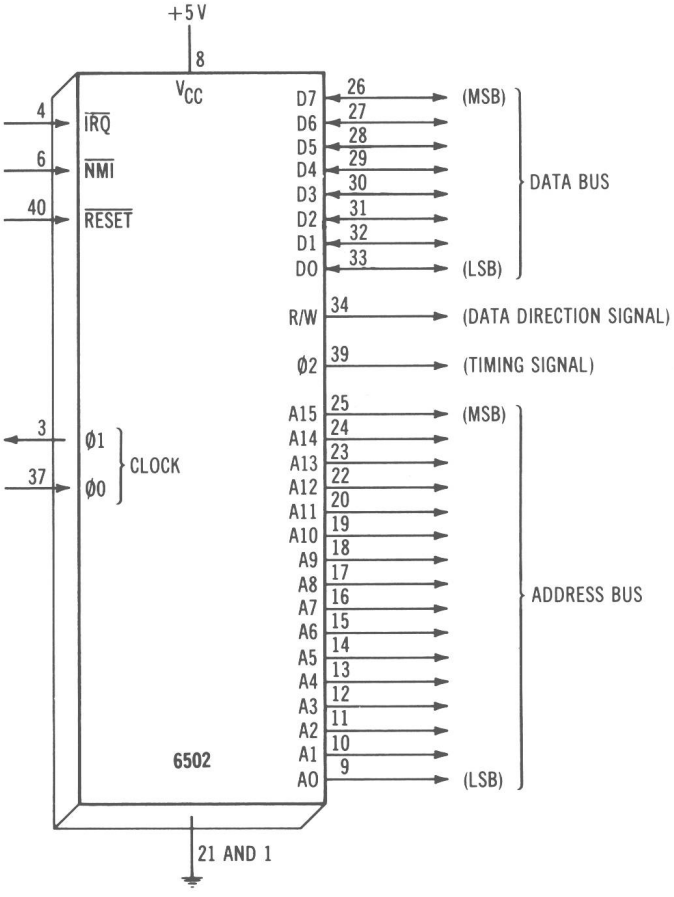


Fig. 1-1. Major signals on the 6502 microprocessor.

bus. Each code selects a unique word of memory. Fig. 1-1 shows a diagram of the pin configuration for the 6502. The 16 address lines can be seen at the right side of the figure. Of course, 65,536 is 2^{16} and represents all possible binary combinations of the 16 address lines. Though the 6502 can access up to 65,536 words of memory, it is not necessary to completely fill all of this space. The PET computers come in different sizes—4K, 8K, etc. The K refers to the number 1024 which is a fundamental unit of memory. Thus, a 4K PET actually has only 4096 bytes of its memory space filled with memory for program storage. Memory in the PET is divided into two types: read/write memory and read only memory.

Read/Write Memory

As its name suggests you can both read from as well as write to read/write (R/W) memory. Such memory is often referred to as random access memory (RAM). This is, of course, a misnomer, since all memory types can be randomly accessed. Nevertheless, the term RAM enjoys wide usage in this context and will probably continue to do so. The way the 6502 communicates with memory is quite simple. Fig. 1-1 shows that there are 8 data lines on the 6502. The data lines are the route by which all data enters and exits the 6502 (arrows indicate direction of information flow). Since the data flows in both directions on these lines, we refer to them as a bidirectional data bus. Because data can be either entering or exiting the 6502 on the same set of wires, this could obviously lead to confusion. To avoid such confusion, the computer signals memory that data is either to enter or exit the 6502 by the state of the read/write (R/W) line. If the R/W line is in a logical high state (+5 volts), then data will be read into the 6502 via the data bus. If the R/W line is at a logic low (0 volts), then data will be sent from the 6502 on the data bus. Since the 6502 operates in the range of a million operations each second, timing of these data transfers is very critical. Precise timing is accomplished by the use of a *strobe* pulse from the phase-two clock which tells the memory that data should be transferred at that exact moment. For example, a typical read cycle from memory would proceed as follows: First, the location within memory would be selected by the 6502 placing the proper binary code on the 16 address lines. At the same time the 6502 causes the R/W line to go high, designating a read cycle. The memory chip responds by selecting the proper word of memory, determined by the code on the address lines, and by getting that 8-bit word ready to send to the 6502 over the data lines. The actual data transfer is accomplished, however, only when the phase-two clock line goes from its high to its low state. Timing diagrams of both memory-read and memory-write cycles appear in Fig. 1-2. The direction of information flow

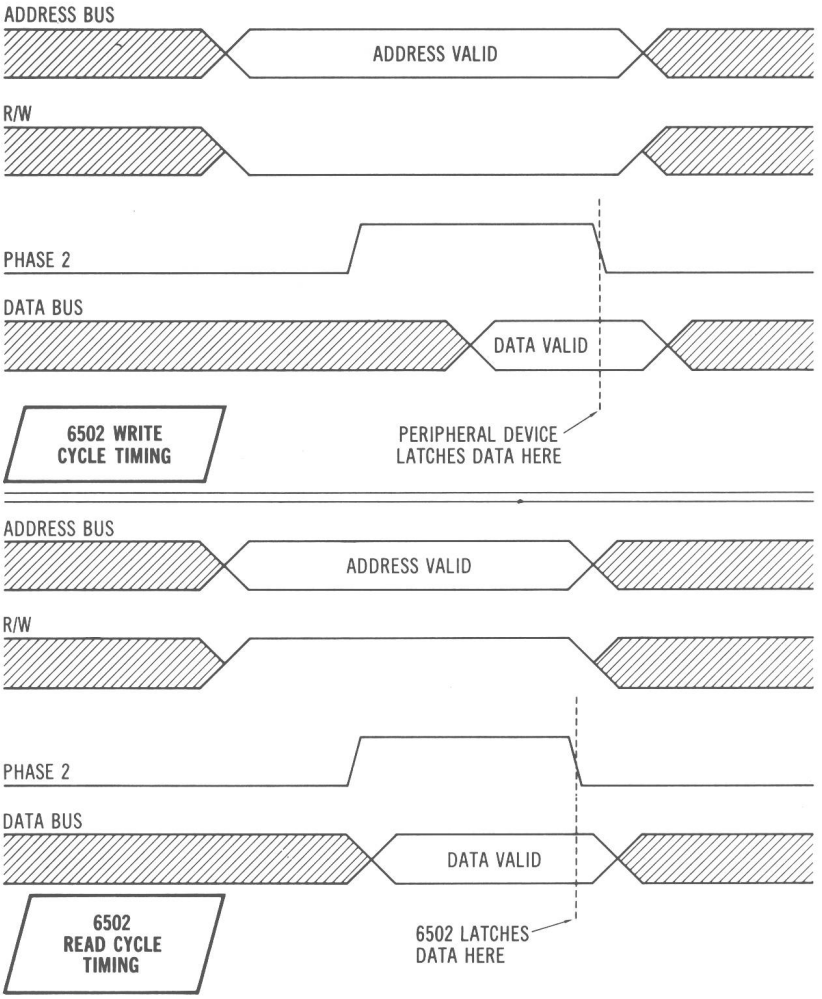


Fig. 1-2. Timing for data transfer in the 6502 microprocessor.

on the data lines is indicated by the status of the R/W line and the precise timing for data transfer is controlled by the trailing edge of the phase-two clock. The shaded areas indicate that the signals are undefined at that time.

Read Only Memory

The second type of memory found in the PET is read only memory (ROM). As its name implies, the computer can read the contents of ROM but cannot store any new data in it. The advantage of a

ROM is that the program it contains is in essence permanently stored in the computer so that it is always available. ROMs have played an important role in the development of microcomputers. Two major advances, which have lowered computer costs are microprocessor chips such as the 6502 and semiconductor R/W memory. Until recently, computers used magnetic core type memory, which was very expensive. Semiconductor memory, as used in the PET, is much less expensive; however, it has one shortcoming. Magnetic core memory retains its data, even when the power is removed, but semiconductor R/W memory does not. We refer to this type of memory as being *volatile,* since the memory contents seem to "evaporate" each time the computer is turned off. This volatile memory created a serious problem for computers using semiconductor memory. Computers using magnetic core memory traditionally had elaborate front panels, through which an initialization program could be tediously entered by hand. Since magnetic core memory is nonvolatile, the initialization program would stay in memory once it was entered. Thus, that job should only have to be done once. In a computer with volatile memory, however, one was required to reenter the initialization program by hand each time the computer was turned on before it could operate. To overcome this inconvenience, read only memories were incorporated in these machines to hold the initialization programs. In fact, ROMs worked so well that the expensive front panel was no longer required, and most of today's microcomputers have eliminated the facility for directly loading memory from switches on the front panel.

Not only does the PET have an initialization program in ROM, it also has the entire BASIC interpreter incorporated in ROM so that it is ready with a high-level language as soon as it is turned on. Read only memories come in many varieties. Some, like those in the PET, must be programmed at the factory, while others can be field programmed and are called programmable read only memory (PROM). One of the most popular types of ROM is the EPROM which can be erased with ultraviolet light for reprogramming.

PERIPHERAL DEVICES

The third fundamental part of any computer is the group of devices that serve as inputs and outputs. These are often referred to as I/O devices. Obviously, no computer would be of much value unless it had some means of receiving data from the outside world, and, likewise, a means for outputting its results. The PET, for example, has several means of receiving input data, such as the keyboard, the cassette drive, and the IEEE 488 bus. Since the purpose of this book is to show the reader how to interface his own peripheral

devices to the PET, we are very interested in this aspect of the computer.

The 6502 handles peripheral devices a little differently than most of the other microprocessors in that it simply treats them as memory. An input device, such as the keyboard, becomes a type of read only memory. Similarly, an output device, such as a printer, becomes a type of write only memory. Just like any other memory, peripheral devices must be assigned a unique 16-bit address code, which can be recognized when that code is present on the address bus. Logic in the interface must recognize when the appropriate address code is present and alert the device to get ready for an exchange of information over the data lines that will occur when the strobe pulse arrives.

ORGANIZATION OF THE PET

With the above background in mind, we can now describe how the 6502 is specifically implemented in the PET. Although the 6502 allows for 64K of memory or peripheral devices, the PET only fills a fraction of this memory space. Fig. 1-3 shows a map of the PET memory. The "top" of the memory space is filled with ROMs and I/O devices. The bottom region of memory is filled with read/write memory. The space in the middle is open and can be filled with either R/W memory to increase the program storage space for BASIC, ROMs to perform special functions, or additional I/O devices.

When the power is turned on, the 6502 is automatically reset. On reset, the 6502 begins executing the program stored in ROM at the top of the memory. This program causes the PET to first see how much R/W memory it has. Starting with location 1024, it sequentially stores a code at each word of memory, and then reads it back again. If the memory is good it will read back the same number it stored. It continues this until it reaches a memory location which did not successfully store the number. This will either be a defective memory location or, hopefully, the top of the R/W memory. The initialization program then tells the BASIC interpreter how much R/W memory is present so that it does not try to store or execute a BASIC program which requires more memory than is available. After the memory size has been determined, the 6502 begins executing the program stored in the BASIC ROMs. From that point on, the system is under control of the BASIC interpreter program.

BASIC is a high-level language that does not normally make provisions for manipulating individual memory locations or I/O devices other than the keyboard and the screen. To facilitate the use of non-standard I/O devices, several extensions to the language have been added to PET BASIC. The first of these is the PEEK statement. The PEEK statement allows the user to see what value is actually

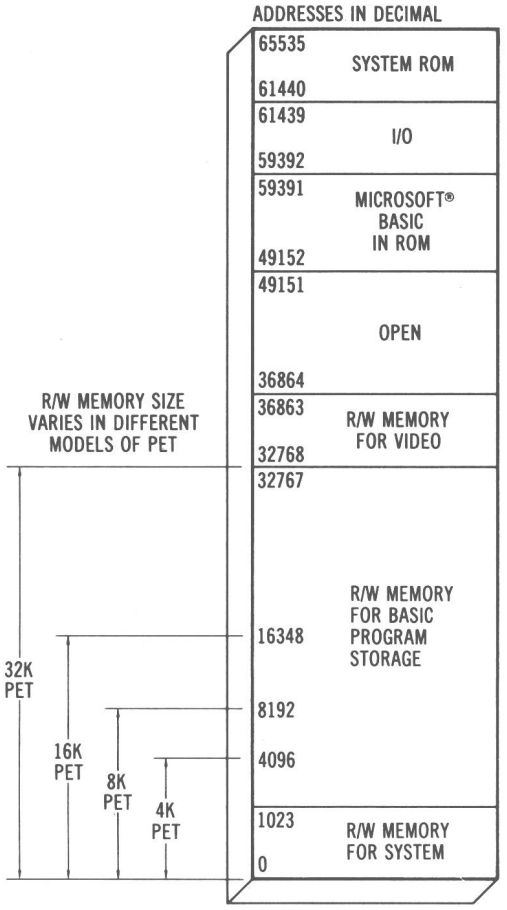


Fig. 1-3. The PET memory map.

stored in any memory location. The statement PRINT PEEK (20000) will cause the decimal value of the contents of memory location 20,000 to be printed on the screen. If an input device were interfaced to the PET at decimal address 20,000, then every time the above statement was executed, the number input to the PET by that device would appear on the screen.

A companion command to PEEK is POKE. POKE stores a number into memory instead of reading from memory. Thus, POKE 20000, 200 causes the binary equivalent of 200 to be stored at memory location 20,000. If an output device "resided" at that address then that command would result in the binary equivalent of 200 being outputted to that device.