8

AN MCS6502-BASED MICROCOMPUTER—THE KIM-1

INTRODUCTION

This chapter is written for persons who need to become acquainted with the KIM-1® microcomputer. This chapter uses the programmed instruction approach. Beginning with a discussion of the actual power supply connections, the chapter then proceeds to discuss the function of each key on the KIM-1 keyboard. The function of each key is highlighted by tables showing keys pressed and the corresponding display. After the keyboard entry of a simple problem is discussed, the use of the Teletype® model ASR33 as an input/output terminal is presented. The use of the Teletype is discussed in detail with reproductions of actual printed outputs and punched paper tapes being provided. Next the use of an audio cassette tape recorder is presented. Examples illustrate the detailed step by step procedure for recording onto magnetic tape from the microcomputer, and vice versa. Simple programming examples are used in all cases so that the reader can focus his attention on the operation of a microcomputer at machine or assembly language levels.

This chapter provides detailed information on the interaction between a user and a microcomputer when assembly level (or machine) language is used. After studying this chapter the reader will be in a position to decide whether this detailed level of interaction is acceptable,

® KIM-1 is a registered trademark of the MOS Technology Corporation or whether he should consider the use of a microcomputer equipped with a higher level language. A higher level language, such as BASIC, makes the computer much more transparent to the user. (The reader should be forewarned, however, that there are literally hundreds of different versions of BASIC, so purchase of a microcomputer with a resident BASIC does not solve all of the user's problems.)

8.1 WHAT IS A KIM–1?

A KIM-1 (Keyboard Input Monitor) is a microcomputer built around the 6502 microprocessor presently manufactured by MOS Technology and Rockwell. The KIM-1 presently (1977 - 1978) sells for a little over $200 and is a complete microcomputer which includes the following:

* The microprocessor
* 2048 ROM bytes
* 1024 + 128 RAM bytes
* 30 input/output pins
* 2 timers
* Interface for audio cassette
* Interface for a Teletype
* 23 key keyboard
* Display consisting of 6 seven bar characters

The KIM-1 is an example of the so-called single-board microcomputers, because, except for the power supplies, it is fabricated on a single printed circuit board measuring approximately 20 by 28 cm, (8½ by 11 inches). Figure 8.1-1 is a photograph of the KIM-1 and the single 5 volt power supply which is required when using only the keyboard, a Teletype, or other compatible terminal. An additional 12 volt power supply is required if an audio cassette tape recorder is added. The KIM-1 circuit board is shown in Figure 8.1-2.

The KIM-1 is typical of the many microprocessor based microcomputers which are presently being sold. It is the authors' opinion that this chapter will be of more use to the reader if one microcomputer is presented in depth as opposed to discussing several microcomputers in more general terms. Somewhat like learning to drive an automobile, or to ride a bicycle, the carryover from the use of one microcomputer to another is large so the adjustment period in learning to use a different microcomputer is surprisingly short.

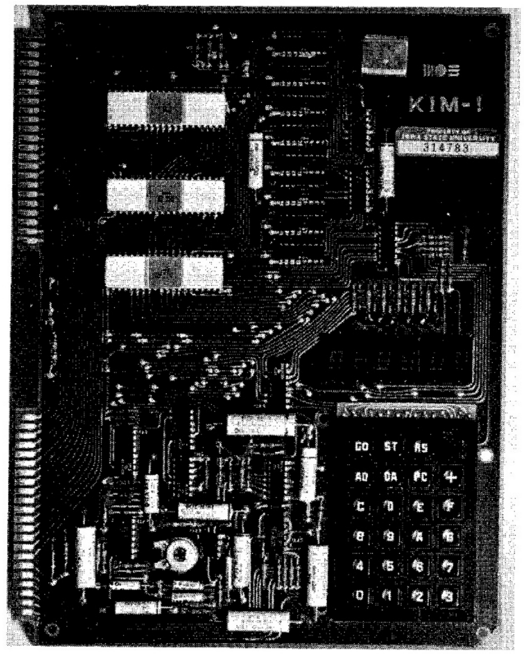
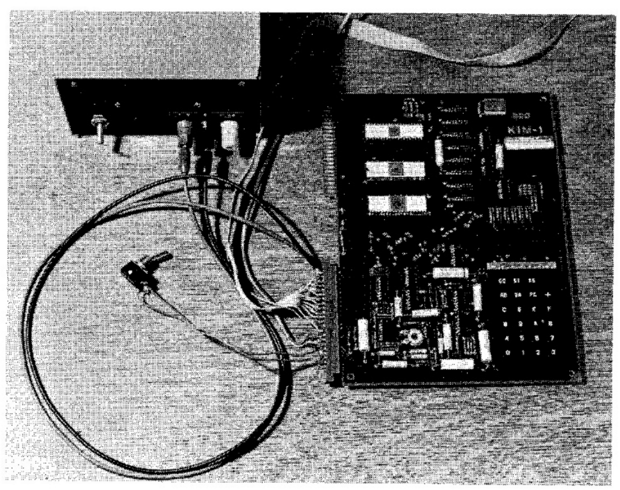


Figure 8.1-1(a) The KIM-1 microcomputer



(b) The KIM-1 microcomputer connected to a 5 volt power supply (Note Teletype mode switch.)

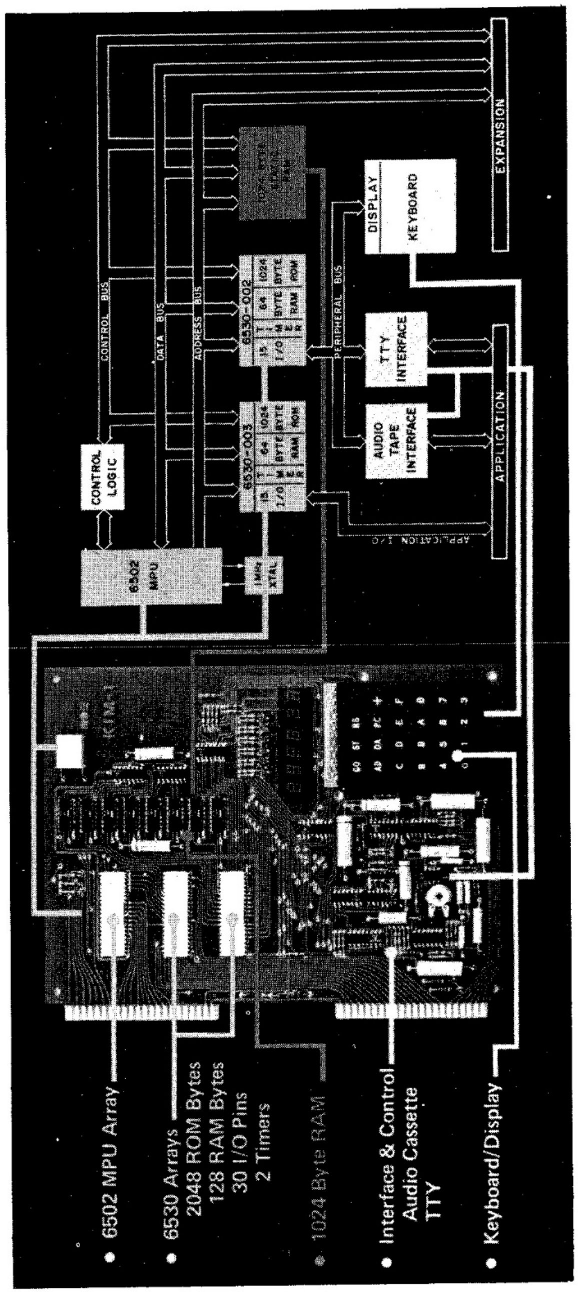


Figure 8.1-2 Identification of the major components on the KIM-1 microcomputer printed circuit board (Courtesy RCC News)

8.2 THE KIM–1 SYSTEM: (A Micro versus a Mini)

A microcomputer system, such as the KIM-1, differs from a minicomputer system in much more than just cost. With a minicomputer, such as the Digital Equipment Corporation PDP-11, the documentation provided by the manufacturer is extensive, diversified, and quite specialized. It is not unusual to find all of the following documents:

1. An Installer's Manual - provides unpacking information and power supply connections for the electrician.
2. An Operator's Manual - provides front panel information for the operator.
3. A Software Manual - provides information for the programmer.
4. A Hardware Manual - provides information about the theory of operation.
5. A Maintenance and Trouble Shooting Manual - provides the technician with schematic diagrams and periodic maintenance requirements, such as air filters which need to be changed or cleaned.

In the case of a microcomputer, the documentation is frequently very limited. (Fortunately, in the case of the KIM-1, rather extensive documentation is provided with three manuals; a Hardware Manual, a Software Manual, and a KIM-1 User's Manual.) Furthermore, in the case of the microcomputer, the installer, electrician, operator, programmer, and technician for maintenance and trouble shooting, are one and the same person.

8.3 AN EXAMPLE PROGRAM (EXAMPLE 8.1)

The purpose of this example program is to serve as a vehicle for illustrating the steps which are involved in going from the statement of a problem to a functioning computer program on the KIM-1 microcomputer. These steps will be the same, in principle, for any microprocessor; only the details will change from one particular microprocessor to another. These steps may be summarized as follows:

* statement of problem
* conversion to assembly language program
* conversion to machine language program
* entry of program into computer
* execution and debugging of program
* modification of program

The following additional steps are sometimes involved:

* entry and execution from a Teletype (includes use of paper tape)
* recording and playing back from an audio cassette tape recorder

Example 8.1. Add two positive numbers located intially in two memory locations and store the sum into a third memory location.

Comments. In order to keep this example as simple as possible:

1. We are considering only positive, one byte numbers.
2. We will not check for a carry out (overflow).

8.3.1 Example 8.1 – Assembly Language Program

The assembly language program for Example 8.1 is given in Table 8.3-1. The assembly language program is also frequently referred to as the "source code" and this designation is used in Table 8.3-1. The reader should note that the lefthand 4 columns are blank. At this time we could also insert the object code for all of the op codes, such as LDA, CLC, STA, etc. This has been done, and is shown in Table 8.3-2.

TABLE 8.3-1 Source Code for Example 8.1. Note that the object code columns are blank.

OBJECT CODE				*SOURCE CODE*			
MEMORY ADDRESS	*BYTE 1*	*BYTE 2*	*BYTE 3*	*LABEL*	*OP CODE*	*OPERAND*	*COMMENT*
				NUM1			Designate one of the two numbers to be added as NUM 1.
				NUM2			Designate the second of the two numbers to be added as NUM2.
				SUM			Designate the sum of the two numbers to be SUM.
						⋮	
					SED		Set decimal Mode.
					LDA	NUM1	Load the accumulator with the numerical value of NUM1.
					CLC		Clear Carry.
					ADC	NUM2	Add with carry NUM1 + NUM2. (There is no ADD instruction.)
					STA	SUM	Store the sum of two numbers into memory location SUM.
				WAIT	JMP	WAIT	The 6502 microprocessor has no halt or stop command. This is a jump to itself.

TABLE 8.3-2 Source Code but Only Partial Object Code for Example 8.1. The Memory Map is needed in order to complete this table.

OBJECT CODE				*SOURCE CODE*			
MEMORY ADDRESS	*BYTE 1*	*BYTE 2*	*BYTE 3*	*LABEL*	*OP CODE*	*OPERAND*	*COMMENT*
				NUM1			Designate one of the two numbers to be added as NUM1.
				NUM2			Designate the second of the two numbers to be added as NUM2.
				SUM			Designate the sum of the two numbers to be SUM.
						⋮	
	F8				SED		Set Decimal Mode.
	A5	ØØ			LDA	NUM1	Load the accumulator with the numerical value of NUM1.
	18				CLC		Clear carry.
	65	Ø1			ADC	NUM2	Add with carry NUM1 + NUM2. (There is no ADD instruction.)
	85	Ø2			STA	SUM	Store the sum of two numbers into memory location SUM.
	4C			WAIT	JMP	WAIT	The 6502 microprocessor has no halt or stop command. This is a jump to itself.

In order to complete Table 8.3-2, we need to decide where to store the program and where to store (or look for) data (the two numbers to be added and the sum). In other words, we need to know where RAM is located *and* if it is available to the user. Frequently, some bytes of RAM are reserved for use by the Monitor program in the microcomputer. As we shall see later, in the case of the KIM-1, 17 bytes (ØØEF to ØØFF in hex) are reserved for the KIM-1 Monitor program. (The user may still decide to use these reserved memory locations, but with the caution that the Monitor program will store its own data in these memory locations.

We are now ready to proceed to the next section in order to learn more about the KIM-1 hardware. In particular, we are anxious to study the KIM-1 Memory Map, which will tell us how much memory we have, what type it is, and where it is located.

8.4 KIM–1 MEMORY MAP AND TABLE

In order to be able to use any microcomputer (including the KIM-1) at the assembly language (or machine code) level, the user must know the answers to the following questions:

* How much memory is there?
* What type is it; (ROM, RAM, PROM, etc.)
* Where is it in memory? (i.e., What is its address?)

The answers to these questions are provided in the memory map and the memory table. The memory map for the KIM-1 is shown in Figure 8.4-1. For convenience of notation, memory is divided into pages. Each page is 256_{10} bytes long. For example, page Ø encompasses Ø through $FF while page 4 encompasses $4ØØ through $4FF. We will use decimal page numbers throughout this material. The reader will note that a user generated program may make use of the following areas of memory:

* all of page Ø except for ØØEF to ØØFF which are reserved for the KIM-1 Monitor program
* all of page 1 with the caution that the stack will use some of page 1
* all of page 2 and page 3
* in page 23 - all input/output locations from 1700 to 173F
  - all 64 bytes of RAM from 1780 to 17BF
  - an additional 44 bytes of RAM from 17C0 to 17EB

The memory table for the KIM-1 is given in Table 8.4-1. The reader will note that some of the information is the same as that provided in the memory map of Figure 8.4-1, but additional and more detailed address information is also provided. When using the KIM-1 the user will need to refer frequently to both of these memory information sources.

We will now use the information provided by the memory map and the memory table to complete the coding for Example 8.1 so that we may enter the program into the KIM-1 memory. This will be the subject for the next section.

8.5 MACHINE CODE FOR EXAMPLE 8.1

If we now refer back to Table 8.3-2, the reader will note that in order to complete that table, we need to decide where to store the program and where to store the data that will be used by the program in the RAM memory. It is usually a good programming technique to store the numerical data in page Ø because then we can use the so-called zero page instructions which require only two, rather than three, bytes

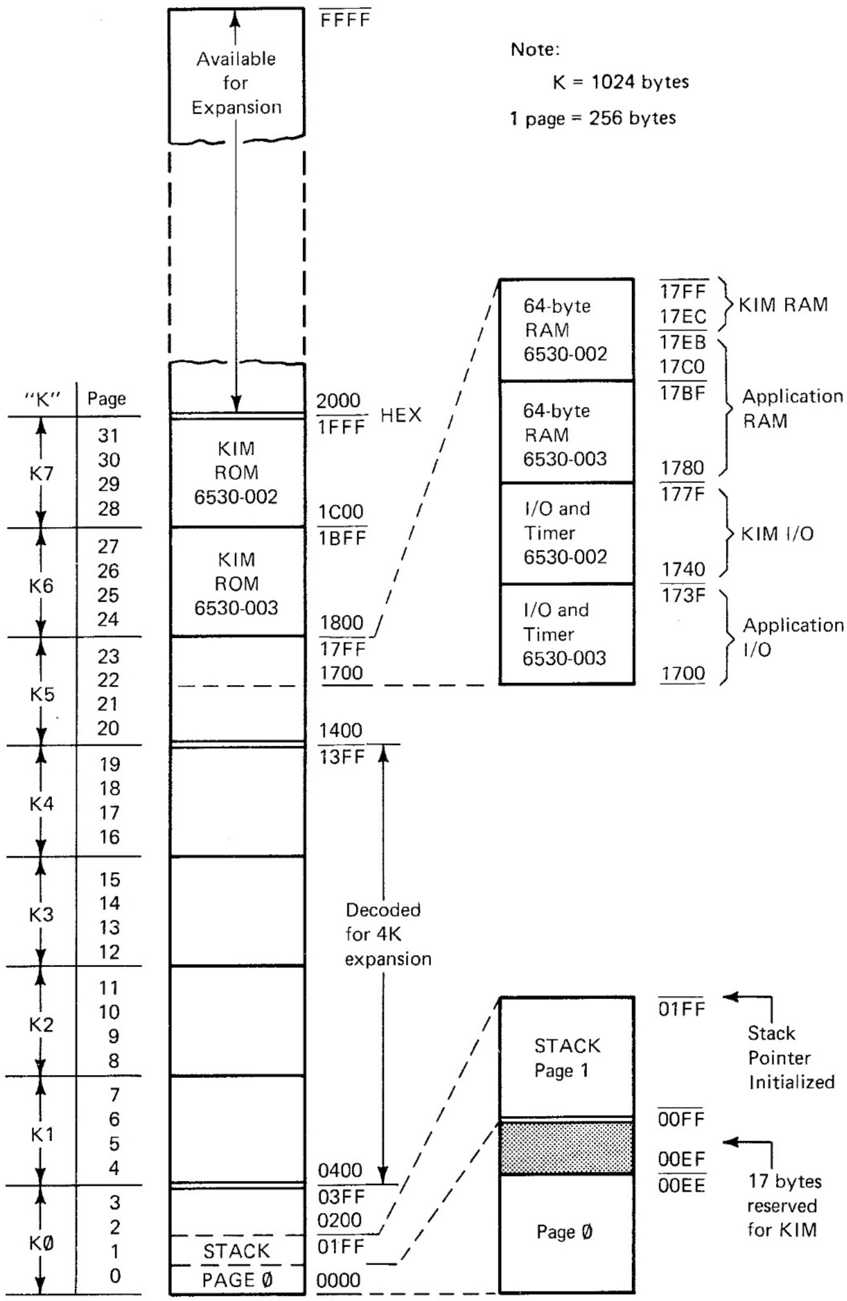


Figure 8.4-1 Memory map for the KIM-1 microcomputer (MOS Technology)

TABLE 8.4-1 Memory Table for the KIM-1 Microcomputer (Courtesy MOS Technology, Inc.)

ADDRESS	*AREA*	*LABEL*	*FUNCTION*
00EF	Machine Register Storage Buffer	PCL	Program Counter - Low Order Byte
00F0		PCH	Program Counter - High Order Byte
00F1		P	Status Register
00F2		SP	Stack Pointer
00F3		A	Accumulator
00F4		Y	Y-Index Register
00F5		X	X-Index Register
1700	Application I/O	PAD	6530-003 A Data Register
1701		PADD	6530-003 A Data Direction Register
1702		PBD	6530-003 B Data Register
1703		PBDD	6530-003 B Data Direction Register
1704 ↓ 170F	Interval Timer		6530-003 Interval Timer (see Table 3.7.1-1, page 144)
17F5	Audio Tape Load & Dump	SAL	Starting Address - Low Order Byte
17F6		SAH	Starting Address - High Order Byte
17F7		EAL	Ending Address - Low Order Byte
17F8		EAH	Ending Address - High Order Byte
17F9		ID	File Identification Number
17FA	Interrupt Vectors	NMIL	NMI Vector - Low Order Byte
17FB		NMIH	NMI Vector - High Order Byte
17FC		RSTL	RST Vector - Low Order Byte
17FD		RSTH	RST Vector - High Order Byte
17FE		IRQL	IRQ Vector - Low Order Byte
17FF		IRQH	IRQ Vector - High Order Byte
1800	Audio Tape	DUMPT	Start Address - Audio Tape Dump
1873		LOADT	Start Address - Audio Tape Load
1C00	STOP Key + SST		Start Address for NMI using KIM "Save Machine" Routine (Load in 17FA & 17FB)
17F7	Paper Tape Dump (Q)	EAL	Ending Address - Low Order Byte
17F8		EAH	Ending Address - High Order Byte

of code, and hence less memory. Also, these two byte, zero page, instructions require less time to execute. We will not use this programming technique here because we require only three bytes of memory for the numerical data and the program itself is so short.

Using the memory map data given in Figure 8.4-1, we decide to:

* store NUM1, NUM2, and the SUM into memory locations ØØØØ, ØØØ1, and ØØØ2 (base 16), respectively
* store the machine code into memory locations beginning with memory location ØØ1Ø (base 16)

As a result of making these memory allocation decisions, the machine code for Example 8.1 may now be completed and it is shown in Table 8.5-1.

We are now ready to load the machine code for Example 8.1 into the KIM-1 memory. This is the subject of the next section.

TABLE 8.5-1 Assembly and Machine Code for Example 8.1

OBJECT CODE				*SOURCE CODE*			
MEMORY ADDRESS	*BYTE 1*	*BYTE 2*	*BYTE 3*	*LABEL*	*OP CODE*	*OPERAND*	*COMMENT*
ØØØØ				NUM1			Designate one of the two numbers to be added as NUM1.
ØØØ1				NUM2			Designate the second of the two numbers to be added as NUM2.
ØØØ2				SUM			Designate the sum of the two numbers to be SUM.
					⋮		
ØØ1Ø	F8				SED		Set Decimal Mode.
ØØ11	A5	ØØ			LDA	NUM1	Load the accumulator with the numerical value of NUM1.
ØØ13	18				CLC		Clear Carry.
ØØ14	65	Ø1			ADC	NUM2	Add with carry NUM1+NUM2. (There is no ADD instruction.)
ØØ16	85	Ø2			STA	SUM	Store the sum of two numbers into memory location SUM.
ØØ18	4C	18	ØØ	WAIT	JMP	WAIT	The 6502 microprocessor has no halt or stop command. This is a jump to itself.

Comment 1: We have arbitrarily decided to store NUM1, NUM2, and SUM into memory locations ØØØØ, ØØØ1, and ØØØ2, respectively.

Comment 2: We have also arbitrarily decided to begin storing the object code into memory locations beginning with memory location ØØ1Ø.

8.6 ENTERING EXAMPLE 8.1 CODE INTO KIM–1

We are now ready to enter the machine code for Example 8.1 (as given in Table 8.5-1) into the KIM-1 memory locations we selected in Section 8.4.

The first step is to apply power to the KIM-1 microcomputer printed circuit board. In this case, we need 5 volts, ± 5%, with a current rating of at least 0.75 amperes. The circuit connections are shown in Figure 8.6-1. It is strongly recommended that these connections be checked very carefully each time the KIM-1 is used. Power supply polarity reversals can be catastrophic.

Table 8.6-1 presents a detailed key by key stroke entry of the machine code for Example 8.1 into the KIM-1 memory according to the decisions made and presented in Table 8.5-1. Table 8.6-1 also gives the information that the user will see on the display following the depression of the various keys. Comments are supplied, where necessary.

TABLE 8.6-1 Key by Keystroke Entry of Example 8.1 Code per TABLE 8.5-1 into KIM-1 Memory

Press Keys	*See on Display*	*Comments*
	blank	Apply 5 volts, ±5%, 0.75 amp. per Figure 8.6-1. There is no power ON-OFF switch on the KIM-1. The SST-ON switch is for selecting the single-step mode.
[RS]	xxxx xx	x implies an unpredictable display.
[AD]	no change	Puts KIM into address mode.
[0] [0] [1] [0]	0010 xx	Contents of memory location 0010 are displayed.
[DA]	no change	Puts KIM into DATA mode.
[F] [8]	0010 F8	Enters F8 code into memory location 0010.
[+]	0011 xx	Increments the memory address displayed by one. This is easier than pressing the four keys for location 0011, which would accomplish the same thing.
[A] [5]	0011 A5	Enters A5 code into memory location 0011.
[+] [0] [0]	0012 00	Enters 00 code into memory location 0012.
[+] [1] [8]	0013 18	Similarly, enters 18 code into memory location 0013, etc.
[+] [6] [5]	0014 65	
[+] [0] [1]	0015 01	
[+] [8] [5]	0016 85	
[+] [0] [2]	0017 02	
[+] [4] [C]	0018 4C	
[+] [1] [8]	0019 18	
[+] [0] [0]	001A 00	This completes the loading of the code for Example 8.1. It is suggested that the user go back to memory location 0010, and then, using the [+] key, check sequentially through memory location 001A for possible errors.

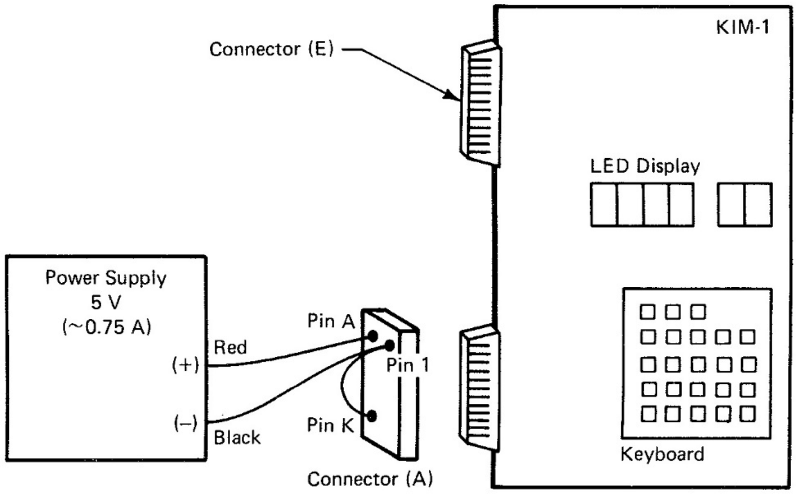


Figure 8.6-1 When used without Teletype, CRT, or audio tape recorder, the KIM-1 requires only one power supply connected as shown

8.7 EXECUTION OF EXAMPLE 8.1 FROM THE KIM–1 KEYBOARD

Table 8.7-1 presents a key by key stroke sequence for entering the numerical values of NUM1 and NUM2 and then executing the program using the [GO] key. We have arbitrarily selected NUM1 = 43 and NUM2 = 25. Again, we have presented the displays which the user will see after depressing the various keys and appended comments, where necessary. The resulting SUM is displayed in the last step of the table and is seen to be equal to 68, which is the correct answer in the decimal mode.

It is suggested that the user modify the program by changing the SED (set decimal mode) op code to CLD (clear decimal mode) and try different numerical values for NUM1 and NUM2 in order to gain a greater understanding of what the decimal and binary modes in the KIM-1 really mean. Problems are provided at the end of this chapter to help guide the user in this endeavor.

8.7.1 Single Step Execution of Example 8.1 Program

Table 8.7-2 presents a key by key stroke sequence for entering the numerical values for NUM1 and NUM2 and then executing the program

TABLE 8.7-1 Execution of the Example 8.1 Program from the KIM-1 Keyboard NUM1 + NUM2 = SUM where NUM1 = 43; NUM2 = 25

Press Keys	*See on Display*	*Comments*
[RS]	xxxx xx	Reset. x implies display is not predictable.
[AD]	no change	Puts KIM-1 into ADDRESS input mode.
[0] [0] [0] [0]	0000 xx	Present contents of memory location 0000 are displayed.
[DA]	no change	Puts KIM-1 into DATA input mode.
[4] [3]	0000 43	Enter 43 as the numerical value of NUM1 into location 0000.
[+] [2] [5]	0001 25	Enter 25 as the numerical value of NUM2 into location 0001.
[+] [0] [0]	0002 00	Clears memory location 0002 where the SUM will be placed.
[AD] [0] [0] [1] [0]	0010 F8	Prepares to run program by loading the beginning address (0010) of our program.
[GO]	blank	Program executes in literally a few microseconds.
[RS]	0010 F8	Reset KIM-1.
[AD] [0] [0] [0] [0]	0000 43	NUM1 is still equal to 43.
[+]	0001 25	NUM2 is still equal to 25.
[+]	0002 68	SUM is 68 and is the correct answer in the decimal mode. Recall that we had previously cleared this memory location.

TABLE 8.7-2 Single Step Execution of Example 8.1 Program from the KIM-1 Keyboard NUM1 + NUM2 = SUM where NUM1 = 43; NUM2 = 25

Press Keys	*See on Display*	*Comments*
[RS]	xxxx xx	Reset.
[AD] [1] [7] [F] [A]	17FA xx	This sequence of 3 lines must be performed in order to use the [ST] key and/or the single step mode. To be more specific: memory location 1C00 is stored into interrupt vector locations 17FA and 17FB.
[DA] [0] [0]	17FA 00	
[+] [1] [C]	17FB 1C	
[AD] [0] [0] [0] [0]	0000 xx	Present contents of memory location 0000 are displayed.
[DA] [4] [3]	0000 43	Enter 43 for NUM1 into location 0001.
[+] [2] [5]	0001 25	Enter 25 for NUM2 into location 0002.
[+] [0] [0]	0002 00	Clear location 0002 where SUM will be placed.
[AD] [0] [0] [1] [0]	0010 F8	Loads beginning address (0010) of our program.
[GO]	0011 A5	Each time the [GO] key is depressed, a single line of code is executed and the memory address of the next line of code is displayed. A "line of code" is defined in TABLE 8.5-1. Note: The same numbers are missing here as in column 1 of TABLE 8.5-1.
[GO]	0013 18	
[GO]	0014 65	
[GO]	0016 85	
[GO]	0018 4C	
[GO]	no change	
.	.	
.	.	
.	.	
[AD] [0] [0] [0] [0]	0000 43	NUM1 is still equal to 43.
[+]	0001 25	NUM2 is still equal to 25.
[+]	0002 68	SUM is 68 again. Recall that we had previously cleared this memory location to zero.
		Do *not* forget that the SST-ON switch is in the ON position.

for Example 8.1 in the *single step mode*. We have again selected NUM1 = 43 and NUM2 = 25. The usual display information and comments are given.

An important item for the reader to note in Table 8.7-2 is the need to properly initialize the interrupt vector locations 17FA and 17FB. This must always be done if the [ST] key and the single step switch mode are to function properly. For more information, please refer to the $\overline{\text{NMI}}$ section of the 6502 microprocessor discussion.

The KIM-1 user will find that the single step mode is used primarily during program debugging.

8.8 DECIMAL OR BINARY CODE

Any system which uses the 6502 microprocessor has the option of operating in either the decimal mode or the binary mode. In the case of the KIM-1, the designers choose not to include a CLD or SED command in the reset programs, so it is not possible to predict whether the KIM-1 will be in the decimal mode or in the binary mode when first energized. (These writers' experiences with about a dozen KIM-1s conform with this observation; about half come up on the binary mode and about half in the decimal mode; unfortunately, it is not always the same half.)

This uncertainty - decimal or binary mode - is responsible for most of the problems encountered by first-time KIM-1 users. These problems appear to the user to be one of the following two major types:

1. The numerical results to arithmetic operations do not appear to be correct.
2. Programs recorded on audio cassette tapes do not load.

Problems of the first type are solved if the user (programmer) develops the habit of preceding each arithmetic operation with a CLD or SED command. (The reader will recall that our solution to Example 8.1 in Table 8.2-1 followed this suggestion.)

Problems of the second type are solved if the user (operator) sets the KIM-1 into the binary or decimal mode immediately after powerup. This may be done by loading all 0s or all 1s into the memory location ØØF1. When we press the Go button, the contents of this location will be loaded into the P register. Actually only the decimal bit in the status register needs to be set to a 1 for the decimal mode and set to 0 for the binary mode, but it is much easier for the user to set all 1s or all 0s. (There are 64 different ways to set one bit to a 1 or to a 0 in a byte.)

8.9 KIM–1 KEYBOARD KEY FUNCTIONS

The KIM-1 keyboard key functions are summarized in Figure 8.9-1.

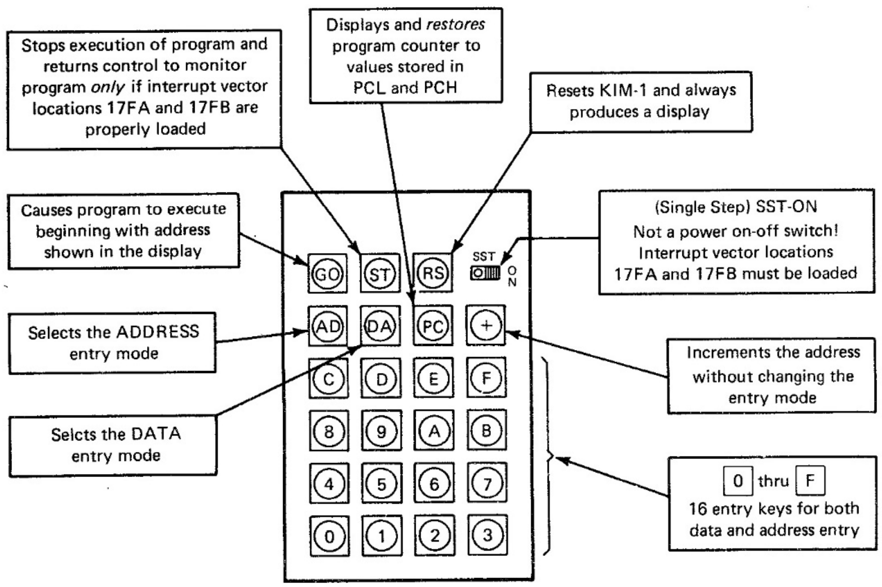


Figure 8.9-1 Summary of KIM-1 keyboard key functions and cautions

8.10 OPERATING THE KIM–1 VIA A TELETYPE

Before energizing a KIM-1 system that is using a serial teleprinter such as the TELETYPE® Model 33ASR (Automatic Send and Receive)* the following items should be verified.

1. The Model 33 is wired for full duplex 20 mA operation.
2. The 5 volt power supply is connected to the KIM-1 application connector as shown in Figure 8.6-1. Be sure a jumper is connected between pins 1 and K.

* Models 33KSR or RO can also be used.

3. Pins R, U, S, and T of the KIM application connector are connected to the Teletype as follows:

R	X-7	or	P2-8
U	X-6	or	P2-7
S	X-3	or	P2-5
T	X-4	or	P2-6

   X denotes Terminal Block X and P2 denotes Plug 2. (On the 33ASR Teletype terminal block X can be located by noting the ac line cord connected to X1 and X2. It is the second receptacle from the left on the top row as viewed from the rear of the machine.
4. The toggle switch TTY-KB (keyboard) is in the TTY (TELETYPE) position.

8.10.1 The KIM-1 Prompting Message

If you have checked all of the electrical connections specified in Sec. 8.1, you may now proceed to do the following:

1. Energize the 5 volt power supply.
2. Plug the TELETYPE into a 115 volts, 60 Hz. power outlet and turn the switch marked LINE-OFF-LOCAL to LINE. This switch is located at the right side of the small vertical front panel strip which bears the letters T E L E T Y P E as shown in Figure 8.10.1.

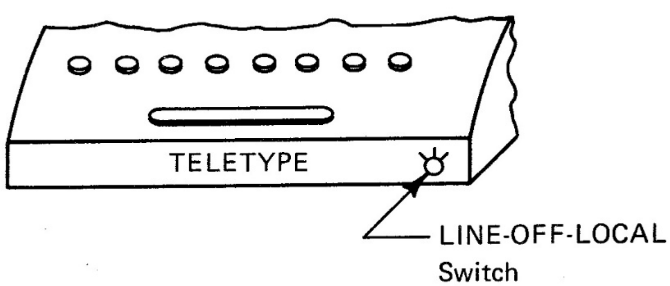


Figure 8.10-1 Closeup view of LINE-OFF-LOCAL switch on model 33ASR TELTEYPE

The following steps are most important because the KIM-1 system adjusts to the bit rate of the serial teleprinter and requires this sequence of key strokes to establish the proper bit rate:

Press KIM key	Press TTY key	See printed
RS	RUB OUT	KIM xxxx xx

where xxxx xx denotes an arbitrary, unpredictable set of six (6) printed characters. If everything is working properly, you should immediately observe the message KIM xxxx xx being typed in the two line format shown above. This is a prompting message telling you that the TTY is on-line and that the KIM-1 system is ready to accept commands from the TTY keyboard.

If the "KIM" prompting message is not typed, repeat the two key sequence given above; that is, press the RS key on the KIM-1 keyboard and the RUB OUT key on the TTY. If the "KIM" message is still not typed, recheck all connections and try again. If the problem still persists, obtain assistance.

8.10.2 Possible Input and Output Modes

Immediately after the "KIM" prompting message is typed on the TTY, the KIM-1 system is ready to accept commands from the TTY keyboard. As soon as you decide what type of input and/or output you desire, you may refer to Figure 8.10.2 and determine what sections describe the procedure you must follow.

8.10.3 Another Simple Example (Example 8.2)

The purpose of this simple example is to explain and to illustrate the different possible input and output modes using the TELETYPE Model 33ASR. It also illustrates another way of terminating a program. Rather than just jumping to itself, the program jumps into the Monitor Program and displays the numerical value of the SUM and its memory location.

Example 8.2. Add two positive integers located in two memory locations and store the sum into a third memory location. Output and display the sum and its memory location. It is not necessary to check for overflow. The solution to Example 8.2 is presented in Table 8.10-1.

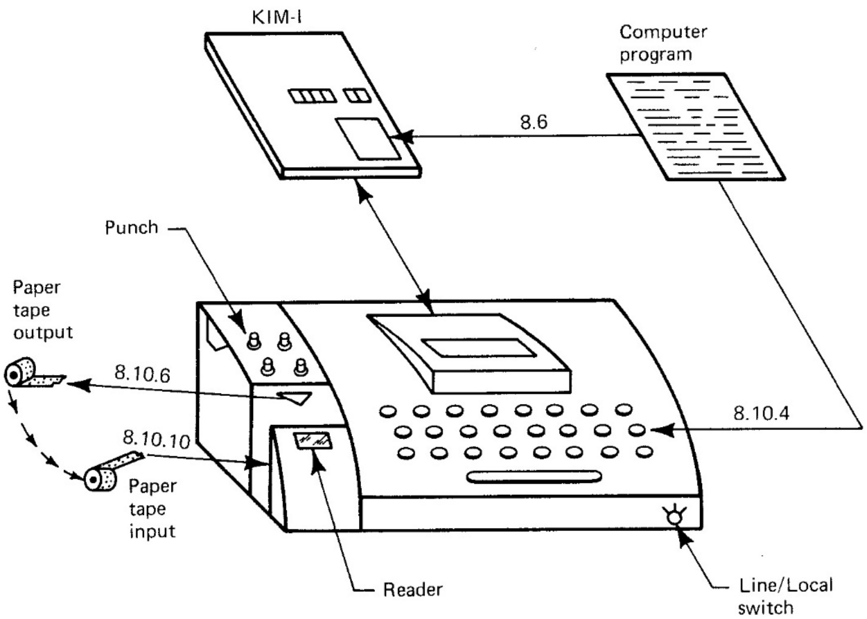


Figure 8.10-2 Pictorial view of possible input/ouput modes using a TELETYPE 33ASR with the KIM-1

TABLE 8.10-1 Source code and machine code for adding two numbers, storing, and displaying the sum. Refer to Example 8.2 for more details.

Memory Map	*Byte 1*	*Byte 2*	*Byte 3*	*SOURCE CODE LABEL*	*OP CODE*	*OPERAND*
ØØ				NUM1		
Ø1				NUM2		
Ø2				SUM		
Ø3	18			PROGRAM	CLC	
Ø4	F8				SED	
Ø5	A5	ØØ			LDA	NUM1
Ø7	65	Ø1			ADC	NUM2
Ø9	85	Ø2			STA	SUM
ØB	A9	Ø2			LDA	#Ø2
ØD	85	FA			STA	$FA
ØF	A9	ØØ			LDA	#$ØØ
11	85	FB			STA	$FB
13	4C	4F	1C		JMP	1C4F

8.10.4 Using the TELETYPE Keyboard to Enter a Program

While using the TELETYPE keyboard to enter a program, you may perform the following operations:

To Select an Address

1. Type four hex keys (0 to F) to define the address.
2. Press the [SPACE] bar.

The TTY will respond by typing the address code selected followed by a two hex digit code corresponding to the data stored at the selected address location.

Example:

Type:	1802 [SPACE]
TTY printer responds:	1802 8D

showing that data 8D is stored at memory location 1802, which is in ROM, so its contents cannot be changed.

Time saving hint: Leading zeros need not be entered.

Examples:

EF [SPACE] selects address 00EF
A [SPACE] selects address 000A
[SPACE] selects address 0000

To Modify an Address

1. Select an address as above. Don't forget to press the [SPACE] bar.
2. Type two hex characters to define the data to be stored at this address.
3. Press the TTY period key, hereafter denoted by ⊙.

Example:

Type:	∅234 [SPACE]
TTY printer responds:	∅234 xx
Type:	6D ⊙
TTY printer responds:	∅235 xx

Note: The selected address ∅234 has been incremented automatically by one to the next address ∅235 and the contents xx of memory location ∅235 have been displayed.

Time saving hint: Leading zeros need not be entered.

Examples:

A ⊙ enters data ∅A
⊙ enters data ∅∅

(This is a frequent source of errors for even experienced programmers who clear memory locations inadvertently by using this short-cut.)

To Step To Next Address (Without modifying the contents of the current address)

1. Press key (RETURN)

Example:

See printed:	1234 xx	
Type:		(RETURN)
Printer responds:	1235 xx	
Type:		(RETURN)
Printer responds:	1236 xx	
etc.		

To Step to Preceding Address (Without modifying the contents of the current address)

1. Press key (LF)

Example:

See printed:	1234 xx	
Type:		(LF)
Printer responds:	1233 xx	
Type:		(LF)
Printer responds:	1232 xx	
etc.		

To Abort Current Operation

1. Press key (RUB OUT)

Example:

See printed:	1264
Type:	(RUB OUT)
Printer responds:	KIM
	xxxx xx
Type:	1234 [SPACE]
Printer responds:	1234 xx

In this example, the (RUB OUT) key was used to correct an erroneous address selection.

Note: The (RUB OUT) key must be depressed after each depression of the KIM-1 [RST] key in order to allow the operating program to define the serial bit rate for the particular Teletype being used.

8.10.5 Using the TELETYPE Keyboard to Execute a Program

To Execute a Program (From the TTY keyboard)

1. Enter the starting address of the program followed by [SPACE].
2. Type (G) .[This is analogous to the [GO] key on the KIM-1 keyboard.]

Example 8.2 may be executed as follows:

Type:	ØØØ3 [SPACE]
See printed:	ØØØ3 18
Type:	(G) (on TELETYPE keyboard!)

Comments: In this example, program execution begins from location ØØØ3 and will continue until memory location ØØ15 is reached. In other cases, program execution continues until the [ST] or [RS] keys on the KIM-1 keyboard are depressed.

Actual TTY output for Sec. 8.10.3 Example 8.2 stored in the KIM-1 with 8 stored in memory location ØØØØ and 9 stored in memory location ØØØ1. As you may recall, the sum = 17 and is stored in memory location ØØØ2. The Teletype output is shown in Figure 8.10.3

			COMMENTS
		ØØØ3	Program begins in memory location ØØØ3.
ØØØ3	18	G	Note that G is typed on the same line as the previous response.
KIM			
ØØØ2	17		Sum = 17 and is stored in memory location ØØØ2.

Figure 8.10-3 Actual TELETYPE output for Example 8.2
NUM1 + NUM2 = SUM for NUM1 = 8 and NUM2 = 9
SUM = 17

To Execute a Program (Single step from TTY keyboard)

1. Enter the starting address of the program followed by [SPACE].
2. Slide switch on KIM-1 keyboard marked SST-ON to ON.
3. Type key Ⓖ once for each step of program you wish to have executed.

Example 8.2 may be executed from the TELETYPE keyboard in the *single-step* mode as follows:

Type:	ØØØ3 [SPACE]
See printed:	ØØØ3 18
Type:	Ⓖ and continue depressing the Ⓖ key until output does not change. See Figure 8.10.4 for the actual output as well as comments.

8.10.6 Using the TELETYPE Keyboard to Punch a Paper Tape

To Punch Paper Tape

1. Load and thread blank paper tape into the punch unit.
2. Decide on the starting address and ending address of the data block to be punched on the paper tape.

A paper tape for *Example 8.2* may be punched as follows:

			COMMENTS
		Ø3	*[SPACE] key does not print, of course!*
ØØØ3	18	G	*If you are having problems - - did you load the interrupt vector for proper [ST] key operation as described in Sec. 1.1.3?*
KIM			
ØØØ4	F8	G	
KIM			
ØØØ5	A5	G	*In the course of single stepping through a program, certain addresses will appear to be skipped. This is normal and to be expected, because all of the bytes involved in the execution of the single instructions are accessed and the program halts only on the first byte of each successive instruction.*
KIM			
ØØØ7	65	G	
KIM			
ØØØ9	85	G	
KIM			
ØØØB	A9	G	
KIM			
ØØØD	85	G	
KIM			
ØØØF	A9	G	
KIM			
ØØ11	85	G	
KIM			
ØØ13	4C	G	
KIM			
1C4F	2Ø	G	*From here on, this would repeat indefinitely.*
KIM			
1C4F	2Ø	G	
KIM			
1C4F	2Ø	G	
KIM			
1C4F	2Ø		
•			
•			
•			
		Ø2	
ØØØ2	17		*SUM = 17 and is stored in memory location ØØØ2.*

Figure 8.10-4 Actual TELETYPE output for the execution of Example 8.2 in the *single-step* mode
Numerical values: 8 + 9 = 17

Comments:

a) Load the program's ending address ØØ15 into KIM-1 memory locations 17F7 (EAL) and 17F8 (EAH) where

EAL = Ending Address - Low Order Byte
EAH = Ending Address - High Order Byte

b) A starting address = ØØØØ has been selected. We could have selected a starting address of ØØØ3, if we did not wish to include the numerical values of NUM1, NUM2, and SUM.

c) The key by key stroke sequence for Example 8.2 are:

Type:	17F7 [SPACE]	
See printed:	17F7 xx	
Type	15 ⊙	
See printed:	17F8 xx	
Type:	ØØ ⊙	
See printed:	17F9 xx	Comment: Ignore this memory location because our next step is to specify the starting memory location.
Type:	ØØØØ [SPACE]	
See printed:	ØØØØ xx	Activate punch by depressing button on punch marked ON.
Press key:	(Q)	Comment: Watch and listen to the tape being punched!

Please refer to Figure 8.10.5 for the actual printed output corresponding to Example 8.2. Figure 8.10.6 presents the actual punched paper tape for Example 8.2.

```
      Q
;18ØØØØØ8Ø91718F8A5ØØ65Ø185Ø2A9Ø285FAA9ØØ85FB4C4F1C22Ø4Ø812
;ØØØØØ1ØØØ1
```

Figure 8.10-5 Actual printed output for Example 8.2

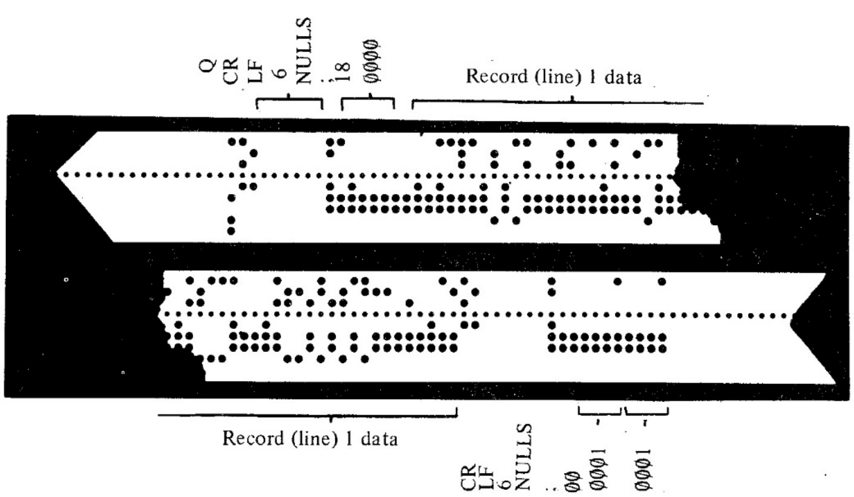
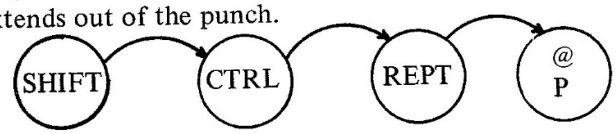


Figure 8.10-6 Actual punched paper tape for Example 8.2

A Handy Hint:

It is convenient to have a leader on each paper tape which has the minimum number of holes punched so that you can write your name, date, and other information. You can do this in three steps:

1. Press keys *in this order and hold down* until about 2 inches of tape extends out of the punch.



2. Tear off and discard this paper tape. The "V" cutting bar will form the tip of an arrow so that you will know in what direction the tape is to move. The sprocket holes, which are off-center, will prevent you from placing the tape in up-side-down.
3. Repeat Step 1 until you have the desired amount of leader at the front of your tape.

8.10.7 Using the TELETYPE Keyboard to List a Program

A printed record of the contents of the KIM-1 memory may be typed on the TELETYPE. The procedure is the same as for punching a paper tape *except* that the punch mechanism is not activated. This is assured if you press the OFF button located on top of the punch mechanism.

8.10.8 Using the TELETYPE Keyboard to List KIM-1 Memory

There is no difference between using the TELETYPE keyboard to list the contents of the KIM-1 memory and to list a program as described in Sec. 8.10.7.

8.10.9 KIM-1 Paper Tape Format

The paper tape DUMP and LOAD routines in the KIM-1 Monitor store and retrieve data in a format designed to reduce errors. Each byte of data is converted to two half bytes (nibbles). Each half byte (in hex) is translated to its ASCII equivalent and written out onto paper tape in ASCII form.

Each line of the output begins with a ";" character (ASCII 3B) to indicate the start of a valid record. The next byte (18 in hex) is the number of data bytes in that line. Then the starting address lower order byte (1 byte, 2 characters), starting address higher order byte (1 byte, 2 characters), and then then data (18_{16} bytes, 30_{16} characters) follow. Each record (line) is terminated by its checksum (2 bytes, 4 characters), a carriage return (ASCII ØD), line feed (ASCII ØA), and six "Null" characters (ASCII ØØ).

The last record (line) has zero bytes of data (indicated by ;ØØ). The starting address field is replaced by a four digit hex number representing the total number of data records (lines), followed by the checksum. A "XOFF" characters ends the transmission.

During a load, all incoming data is ignored until a ";" character is encountered. The receipt of nonASCII data or a mismatch between the calculated checksum and the checksum read from the tape will cause an error condition and the KIM-1 Monitor Program will cause the message '"KIM ERROR" to be printed. The checksum is calculated by adding all data in each record (line) except the ";" character.

Figure 8.10.7 illustrates and interprets how the above description of the paper tape format applies to the Example 8.2 TELETYPE printed output.

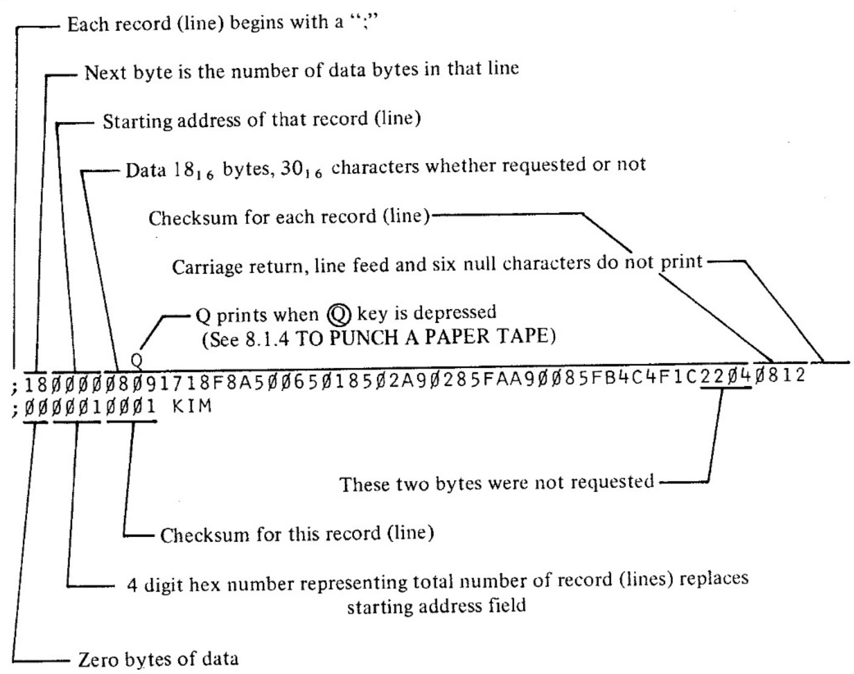


Figure 8.10-7 Illustration and interpretation of the KIM-1 paper tape format as it applies to Example 8.2 output

8.10.10 Reading a Paper Tape into the KIM-1

Paper tapes read into the KIM-1 system must have the format specified in Section 8.10.9. Paper tapes generated by the KIM-1 have, of course, this format. To read a paper tape with the proper format, proceed as follows:

1. Set the Paper Tape Reader switch to FREE.
2. Insert the tape into the tape reader mechanism being careful to align the sprocket holes. Move the tape slightly back and forth ~~to~~ make sure it is free.
3. Press [RS] on the KIM-1 keyboard and (RUB OUT) key on TELETYPE.
4. Type (L) on TELETYPE (mnemonic for Load).
5. Move the tape reader switch to START and remove your hand. Switch will automatically return to its center position, and the reader will begin to read the tape.

Caution: Keep fingers away from the paper tape reading mechanism!

Figure 8.10.8 illustrates the Teletype output corresponding to Example 8.2.

```
        LQ
;18ØØØØØ8Ø91718F8A5ØØ65Ø185Ø2A9Ø285FAA9ØØ85FB4C4F1C22Ø4Ø812
;ØØØØØ1ØØØ1 KIM

KIM
ØØØ1  Ø9
```

Figure 8.10-8 Actual TELETYPE printed output when paper tape for Example 8.2 is loaded into the KIM-1

8.11 ADDING AN AUDIO TAPE RECORDER TO THE KIM-1

8.11.1 Why Add an Audio Tape Recorder?

There are at least four reasons to add an audio tape recorder to the KIM-1 microcomputer:

1. Programs and data which the user has stored in the KIM-1 RAM may be saved before the power is turned off.
2. An audio tape recorder is much less expensive (as low as $20) compared to a paper tape punch and reader.

3. Computer programs and data which are stored on magnetic tape may be readily duplicated and exchanged with other KIM-1 users.
4. Magnetic audio tape, as compared to paper tape, permits a storage density, a smaller more convenient package to handle, and is reusable.

Although both reel-to-reel and cassette tape recorders may be used, the cassette type is used in almost all KIM-1 applications because of its small size, portability, and cost. The discussion that follows will apply to both reel-to-reel and cassette types. (Other types of tape recorders, such as miniaturized recorders without external jacks, are usually intended primarily for dictating applications and require internal modifications. Recorders requiring internal modifications will not be discussed here.)

8.11.2 The Audio Tape Recorder Connection

The audio tape recorder unit to be used should possess the following features:

1. A jack, usually labeled microphone or input, to permit recording the electrical signal provided by the KIM-1 circuit board onto the magnetic tape.
2. A jack, usually labeled earphone or external speaker, to supply the electrical signal of the appropriate level to the KIM-1 circuit board for loading into the microcomputer memory.
3. The standard control switches for PLAY, RECORD, REWIND, and STOP. Other features, such as FAST FORWARD or FAST REVERSE, are convenient for locating programs when more than one program is stored on a tape side.

With the power off to all of the components, check that all of the cables and jumpers are present as shown in Figure 8.11.1. All wires, except for the shielded wires, should be as short as possible and kept away from other wires which might introduce electrical noise.

The Audio Data Out (LO) at pin M has a level of approximately 15 mv peak. The Audio Data Out (HI) at pin P has a level of approximately 1 volt peak. Pin P is used with some of the more expensive tape recorders which have an input (usually labeled "LINE") that accepts higher voltage input signals. The lower level Audio Data Output at pin M is the one which is used with most small inexpensive audio cassette tape recorders.

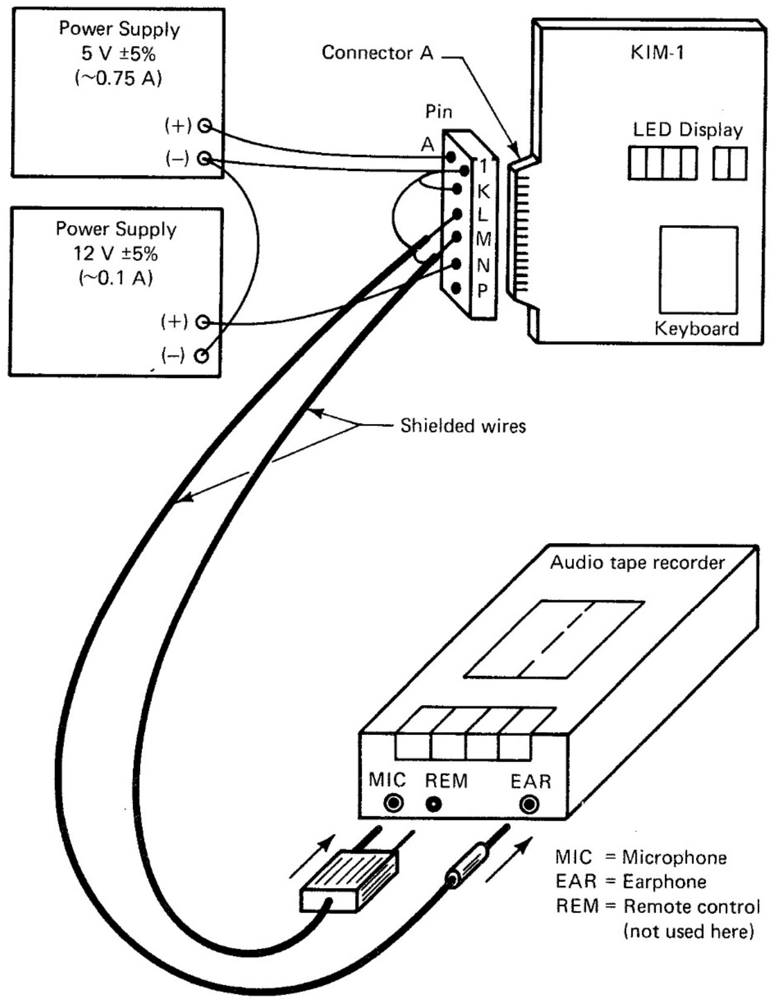


Figure 8.11-1 With an audio tape recorder, the KIM-1 system requires one 5 volt power supply, one 12 volt power supply, one jumper, and two shielded wires connected as shown above

8.11.3 How to Record On an Audio Tape

Before we list the steps to be followed for recording a program and/or data from memory on an audio magnetic tape, let us document the assumptions we are making:

a) There is a program and/or data stored in memory with a known starting and a known ending address.
b) The program and data have been thoroughly tested and the program produces known results given specific data. This prevents confusing program errors with recording/playback errors.
c) The special addresses for the Audio Tape Load and Audio Tape Dump subroutines are known or obtained from Figure 8.11.2.

The procedure for recording on an audio magnetic tape from the KIM-1 memory is:

(If not already on, turn on the 5 volt and the 12 volt power supplies and check power to the tape recorder, if it is not battery powered)**

Step 1.* Verify that the 6502 microprocessor is in the binary mode. This may be done by inspecting the contents of memory location $00F1 and recalling that the decimal mode bit (1 = True) is bit three (b3 of b0 - b7) of the processor status word (which is stored in memory location $00F1). Since bit 3 is zero for so many different hexadecimal numbers, it is frequently easier just to *clear* the decimal mode and place into the binary mode by performing the following steps:

Press Keys	*Display*	*Comments*
[AD] [0] [0] [F] [1]	00F1 xx	Selects the address mode and displays the contents of memory location $00F1.
[DA] [0] [0]	00F1 00	Stores 00 into memory location $00F1.

Step 2. Store the starting address in memory locations 17F5 and 17F6. Be sure the place the low order byte into 17F5 and the high order byte into 17F6.

Step 3. Store the ending address into memory locatins 17F7 and 17F8. Be sure to place the low order byte into 17F7 and the high order byte into 17F8.

The ending address is defined to be one greater than the last actual memory location of our program.

* If it were financially acceptable, we could add CLD instructions to the Audio Tape subroutines, manufacture new ROMs, and remove the need for Step 1.

** Some users have reported erratic operation when using the internal batteries in some of the tape recorders and they suggest using 115 volt 60 Hz adaptors instead.

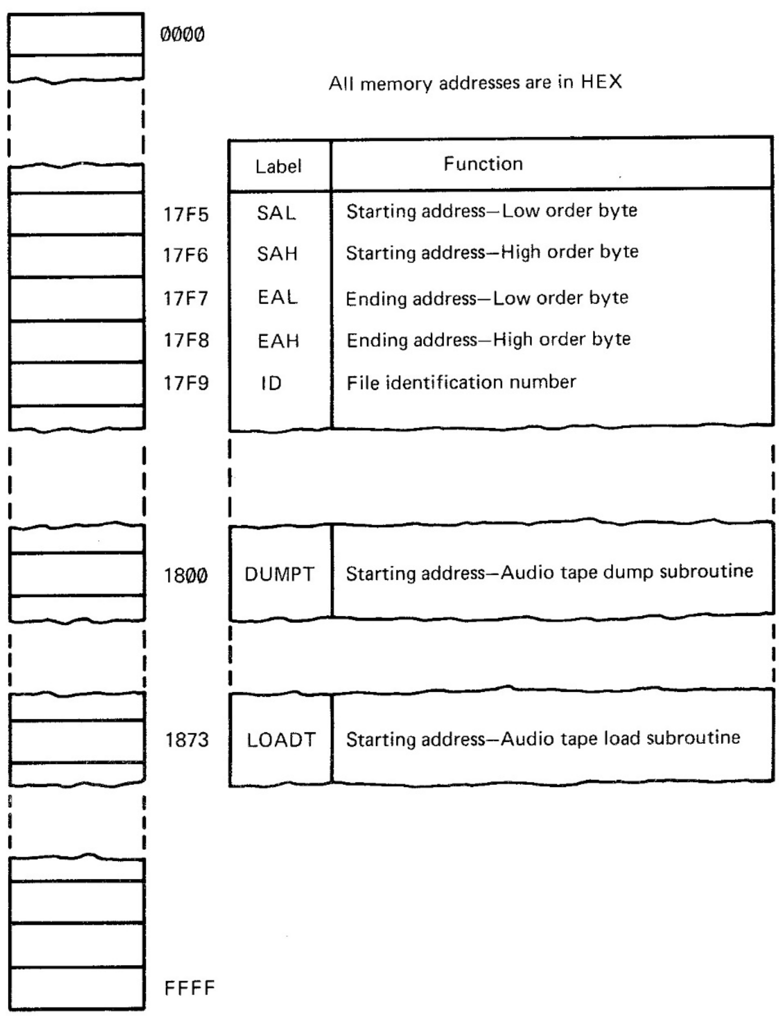


Figure 8.11-2 Special memory addresses for Audio Tape Load (1873) and Audio Tape Dump (18ØØ), starting address, ending address, and ID

Step 4. Pick a two digit hexadecimal number as the file identification number (ID) and store into memory location 17F9.

Do *not* choose ID = ØØ or ID = FF.

For an explanation, please refer to Section 8.11.6, Hint #4.

Step 5. Refer to Figure 8.11.2 and note that the starting address for the Audio Tape Dump subroutine is 18ØØ. Load 18ØØ into the field of the KIM-1. Do not press the [GO] key because the tape is not moving at this time.

Step 6. Select the RECORD mode of the tape recorder and wait a few seconds for the tape to start moving and to attain a constant speed.

Press [GO] key.

Comment: The display will become dark for a time and then the display will light and show

0000 xx where xx is arbitrary

The amount of time during which the display is dark varies with the length of the program. In our example, which involves only about a dozen memory locations, the display is dark for about 16 seconds.

The recording is complete as soon as the display relights.

Step 7. STOP the tape recorder.

Step 8. (optional) REWIND the tape cassette to its starting position.

Step 9. (optional) Listen to the tape. With experience, you will be able to identify the beginning 100 sync pulses, the program in the middle, and the terminating characters at the end of each record by their distinctive sounds.

8.11.4 How to Read an Audio Tape into the KIM-1

Before we list the steps to be followed for reading from an audio tape into the KIM-1 microcomputer, let us list the assumptions we are making:

a) The file identification number (ID) of the record we wish to transfer into the KIM-1 memory is known.

b) The starting address of the Audio Tape Load (LOADT) subroutine is known to be $1873.

The procedure for transferring information from an audio magnetic tape into the KIM-1 memory is:

(If not already on, turn on all power supplies.)

Step 1.* Verify that the 6502 microprocessor is in the binary mode and not in the decimal mode. Please refer to Section 8.11.3, Step 1, for details on how to do this.

Step 2. Place the cassette into the tape recorder and REWIND the tape, if necessary, so that the tape will start to move and will attain a constant speed about 3 seconds before reaching the spot where the record with the desired ID exists on the tape.

Step 3. Store the known file identification number (ID) into memory location $17F9.

Step 4. Load $1873 (the starting address of the Audio Tape Load subroutine) into the address field of the display.

Step 5. Press the GO key.

Comment: The display will become dark and the KIM-1 will search for a tape input with the specified ID number. The tape is not moving so the search is not successful.

Step 6. Set the volume control on the tape recorder to approximately its midpoint position.

Step 7. Place the tape recorder into the PLAY mode. The tape should begin to move forward.

Comment: As soon as the data record with the specified ID number is located and completely read in, the display will relight and show

0000 xx *where xx is arbitrary*

Step 8. STOP the tape recorder after the display relights.

In reading an audio tape into a KIM-1, three types of problems may be encountered.

Trouble #1. In step 7, the display relights and shows FFFF xx. This means that the record with the specified ID was located but the check sum test failed as a result of either a record or a playback error.

FIX: Repeat the entire recording and playback procedures checking each step carefully. In a short program, step through all of the

* If is were financially acceptable, we could add CLD instructions to the Audio Tape subroutines, manufacture new ROMs, and remove the need for Step 1.

memory locations involved and compare with your written program. If the problem still persists, refer to the KIM USERS MANUAL, Appendix C, "In Case of Trouble."

Trouble #2. In step 7, the tape continues to run to the end and the display remains dark. This usually means that the record with the specified ID number was not "found."

FIX: a) Verify that the record was actually on this cassette and on the correct side. (Cassettes have side A and side B recordings.) Repeat the entire recording and playback procedure checking each step carefully.

b) Actually listen to the recording through the speaker or an earphone. With some practice, you will be able to identify the beginning 100 sync pulses, the program in the middle, and the terminating characters at the end of each record by their distinctive sounds.

c) Verify again that the 6502 microprocessor is in the binary mode and not in the decimal mode.

Trouble #3. If you use the tape recorder to erase a tape, you may record a "noise" which seems to be related to a subharmonic of the clock frequency.

FIX: Disconnect the tape recorder from the KIM-1 board while using the tape recorder to erase the tape.

8.11.5 An Example of Record and Playback

The purpose of this example is to illustrate the procedure for recording programs and/or data on audio magnetic tape from the KIM-1 system, and vice versa.

Example 8.11.1. Record and play back into the KIM-1 memory a program which will add two decimal numbers located in two memory locations and will store the sum in a third memory location.

Solution: The source program code and machine code for this example are given in Table 8.11-1. Because the 6502 microprocessor has no stop, wait, or halt instructions, the last instruction in the program is an absolute jump to itself. Pressing the RESET key will cause the KIM-1 system to exit from this infinite loop. In subsequent examples, we will jump to a display subroutine.

TABLE 8.11-1 Source Program Code and Machine Code for Adding Two Numbers and Storing the Sum

Memory Map	*Byte 1*	*2*	*3*	*SOURCE CODE LABEL*	*OP CODE*	*OPERAND*	*Comments*
ØØ				NUM1			First number in location ØØ.
Ø1				NUM2			Second number in Ø1.
Ø2				SUM			Sum in location Ø2.
Ø3	F8			PROGRAM	SED		Set decimal mode.
Ø4	A5	ØØ			LDA	NUM1	Load NUM1 into accumulator.
Ø6	18				CLC		Clear carry.
Ø7	65	Ø1			ADC	NUM2	Add NUM2 to NUM1 with carry.
Ø9	85	Ø2			STA	SUM	Store SUM into location Ø2.
ØB	4C	ØB	ØØ	HALT	JMP	HALT	Jump to itself over and over
ØE							again.

Using the KIM-1 keyboard, load the program given in Table 8.11-1 into memory locations $Ø3 through $ØD. If you do not know how to do this, or have forgotten how, please refer to Section 8.6.

Now thoroughly test and verify that this program is correctly loaded and that it performs as expected. Be sure to load two numbers, run the program, and inspect the contents of memory location $Ø2 to see if the correct sum is present there.

To record a program from the KIM-1 on audio tape. Please refer to Section 8.11.3 and note that we have satisfied all of the assumptions stated there and we are ready to proceed with performing Steps 1 through 8. The results are summarized in Table 8.11-2.

To playback the audio tape into the KIM-1. The assumptions which we have made require us to know that:

a) The desired tape record ID is $11;

b) The starting address of the Audio Tape Load subroutine is $1873.

Recall that in this example, we have arbitrarily chosen $11 as the file ID. However, the starting address $1873 is fixed for the KIM-1 Monitor Program and will always have this value.

Please refer to Section 8.11.4 and note that we have satisfied all of the assumptions stated there and we are now ready to proceed with performing Steps 1 through 6. The results are given in Table 8.11.3.

TABLE 8.11-2 Solution to Example 8.1 for the case of recording on an audio tape from the KIM-1 Microcomputer. The step numbers corresond with those given in Section 8.11-3

Press Keys	*Display*	*Comments*
[RS]	xxxx xx	Step 1:
[AD] [0] [0] [F] [1]	00F1 xx	Places in binary mode by clearing
[DA] [0] [0]	00F1 00	decimal mode.
[AD] [1] [7] [F] [5]	17F5 xx	Step 2:
[DA] [0] [3]	17F5 03	Stores starting address ØØØ3 into
[+] [0] [0]	17F6 00	memory locations 17F5 (low byte) and 17F6 (high byte).
		Step 3:
[+] [0] [E]	17F7 0E	Stores ending address ØØØE into
[+] [0] [0]	17F8 00	memory locations 17F7 (low byte) and 17F8 (high byte).
[+] [1] [1]	17F9 11	Step 4: Loads ID = 11 into location 17F9.
		Step 5:
[AD] [1] [8] [0] [0]	1800 A9	Loads the starting address of the subroutine Audio Tape Dump.
On Tape Recorder [RECORD]		
wait about 3 seconds [GO]	dark (about 16 sec.)	Step 6: Program is being recorded.
	0000 xx	Recording is complete.
On Tape Recorder [STOP]		
	0000 xx	Step 7: Tape recorder is stopped.
	0000 xx	Step 8: (optional) Rewind tape.

TABLE 8.11-3 Solution to Example 8.11.1 for case of reading from an audio tape into the KIM-1 microcomputer memory. The step numbers correspond with those given in Section 8.11.4

Press Keys	*Display*	*Comments*
[RS]	xxxx xx	Step 1:
[AD] [0] [0] [F] [1]	00F1 xx	Places KIM-1 into binary mode
[DA] [0] [0]	00F1 00	by clearing the decimal mode.
		Step 2:
	00F1 00	Place cassette into tape recorder and rewind tape, if necessary.
[AD] [1] [7] [F] [9]	17F9 xx	Step 3:
[DA] [1] [1]	17F9 11	Load ID = 11 into memory location $17F9.
		Step 4:
[AD] [1] [8] [7] [3]	1873 A9	Loads the starting address of the subroutine Audio Tape Load.
		Step 5:
[GO]	dark	KIM-1 is searching for ID = 11.
		Step 6:
	dark	Set volume control to midpoint.
On tape recorder [PLAY]		
		Step 7:
	dark	Places recorder into PLAY mode and tape begins to move.
	(about 16 sec.) 0000 xx	Playback is complete.
[STOP]		
		Step 8:
		Stops tape recorder.

8.11.6 Some Helpful Tape Hints

The following hints are offered to help you make the most effective use of your cassette tapes without encountering some of the more commonly associated problems:

Hint #1. Voice messages may be added between the record data blocks on the tape. The KIM-1 system will ignore these audio messages when the tape is read back provided the voice messages occur only between record blocks. However, you will have to install an earphone or speaker in parallel with the KIM-1 audio tape data pin in order to hear the voice messages.

Hint #2. If more than one data block is to be recorded per cassette side, proceed as follows:

Case 1. Data blocks are to be recorded in sequence without rewinding between blocks.

You need only to specify the parameters of each new block (ID, SAL, SAH, EAH, EAL) and proceed with recording each block following the standard steps listed in Section 8.11.3. It is best to use unique ID numbers for each block.

Case 2. Tape has been rewound or the location of the end of the last data block has been lost.

You must know the ID number of the last recorded data block. Rewind the tape to well before the anticipated starting point and set up the parameters to read the last data block following the steps given in Section 8.11.3, Step 4.

If the data transfer is successful (as indicated by the display showing 0000 xx), you may proceed to load the next data block. Refer to Section 8.11.3.

Hint #3. Avoid placing a data block between two existing data blocks or in an area of the tape which has previously been used for recorded data. You may recall that data blocks may be of arbitrary length and hence variations in block length and tape speed may result in overlapping of recorded data blocks.

Hint #4. Recall that when *recording* on a magnetic tape, you cannot select ID = ØØ or ID = FF. However, when *reading* from a magnetic tape into the KIM-1 memory, we can use these ID numbers as follows:

Case 1. If we select ID = ØØ, the ID number recorded on the tape will be ignored and the system will read the first valid data block encountered on the tape. The data read from the tape will be loaded into memory addresses as specified on the tape.

Comment: This is useful if you forget your ID number or if the ID number was not provided to you. Note that you still need the original starting and ending addresses in order to locate and execute the program.

Case 2. If we select ID = FF, the ID number recorded on the tape will be ignored and the system will read the first valid data block encountered on the tape. In addition, the data block will be loaded into successive memory locations beginning at the address specified in locations 17F5 and 17F6 (starting address low, SAL, starting address high, SAH) instead of the locations specified on the tape.

Comment: This is useful if you forget, or do not know, the ID number and the starting address. Note that you still should know the maximum length of the program so that you don't "write over" and destroy other data stored in memory.

8.12.1 Manipulating the KIM-1 Display (Example 8.12.1)

The KIM-1 display is controlled by the 6530-002 I/O ports. A diagram of the KIM-1 display and 6530 RAM ROM I/O interface is shown in Figure 8.12.1. This diagram is simplified to the point where it does not show any of the drivers, inverters, or pin connections. The primary purpose of Figure 8.12.1 is to help to explain how the KIM-1 six character, 7 segment LED display may be used to display many of the alphabetical characters. In addition, it will be found that it is possible to blink and to shift the characters which are displayed.

The A-side data determines which of the seven segments are illuminated for the character selected by the B-side data. Segments A through G are selected by bits Ø through 6, respectively.

Example 8.12.1. Determine A-side and B-side data which will cause

E	r	r	o	r

to be displayed in the first four character positions.

Character	*A-side data*	*B-side data*
E	79	Ø8
r	5Ø	ØA
r	5Ø	ØC
o	5C	ØE
r	5Ø	1Ø

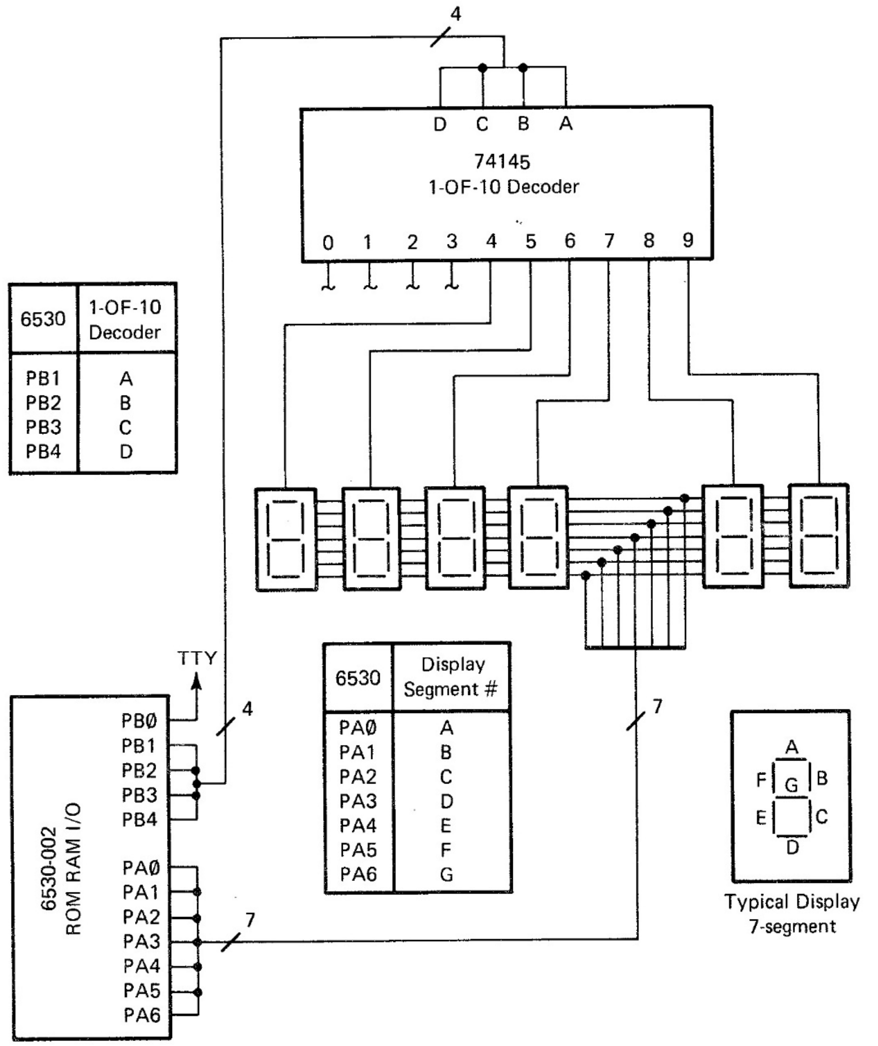


Figure 8.12-1 Simplified diagram of KIM-1 display and control by the 6530-002

Comment 1. A-side data are determined as follows:

	Segments On G F E D C B A	Hex Code
Character E	1 1 1 1 ∅ ∅ 1	79
Character r	1 ∅ 1 ∅ ∅ ∅ ∅	5∅
Character o	1 ∅ 1 1 1 ∅ ∅	5C

Comment 2. Computation of the B-side data may appear to be confusing at first. Because PB1, PB2, PB3, PB4 (instead of the more typical PB∅, PB1, PB2, PB3) are sent to the 1-0F-10 decoder, the decoder supplies an output which is only one-half of the actual input. In other words, we have to supply an input which is twice as large so that character positions ∅4, ∅5, ∅6, ∅7, ∅8, ∅9 actually require as B-side data inputs: ∅8, ∅A, ∅C, 1∅, and 12.

The specific memory addresses associated with the 6530-002 I/O ports are as follows:

174∅	SAD	A-side Data
1741	PADD	A-side Data Direction
1742	SBD	B-side Data
1743	PBDD	B-side Data Direction

A clearer perspective of the relative memory locations may be obtained by referring to Figure 8.12-2 which is Figure 8.4-1 with the above memory locations added. It may be seen that some pins on both data direction registers must be set for output. (Remember 1 = output and ∅ = input.) Just exactly which pins ports A and B are set for input and which pins are set for output is given in Figure 8.12-3.

A program which will load the A and B data direction registers with 7F and 1E respectively, is as follows:

Op Codes	*Source Codes*
A9 7F	LDA #7F
8D 41 17	STA PADD
A9 1E	LDA #1E
8D 43 17	STA PBDD

Example 8.12.2. Write a program which will display the letters "EE" in the two left-handmost characters of the KIM-1 display. The remaining four characters are to be dark.

The solution to Example 8.12.2 is shown in Table 8.12-1.

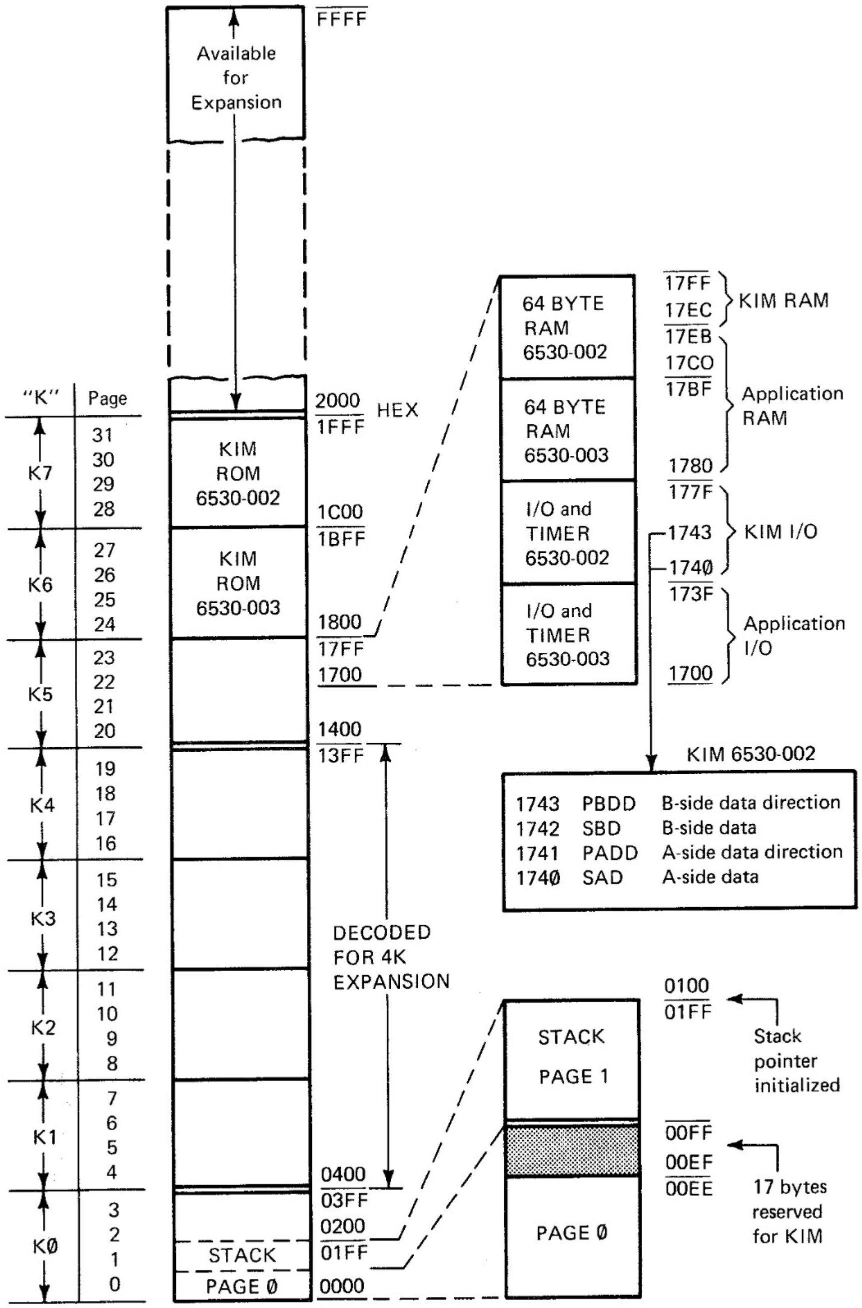


Figure 8.12-2 MEMORY MAP with 6530-002 ports A and B added (Courtesy MOS Technology)

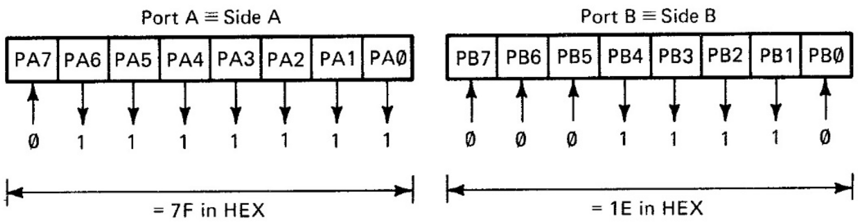


Figure 8.12-3 Input/Output pin configuration for ports A and B in Example 8.12.1
1 = output; 0 = input

TABLE 8.12-1 Source and Op Codes which display the letters "EE" in the KIM-1 display

Memory Map	*Byte 1*	*Byte 2*	*Byte 3*	*LABEL*	*SOURCE CODE OP CODE*	*OPERAND*	*Remarks*
ØØ30	A9	7F			LDA	#7F	Set output pins on
32	8D	41	17		STA	PADD	Port A.
35	A9	1E			LDA	#1E	Set output pins on
37	8D	43	17		STA	PBDD	Port B.
3A	A9	79			LDA	#79	Load the letter E.
3C	8D	4Ø	17		STA	SAD	
3F	A9	Ø8		AGAIN	LDA	#Ø8	Display the letter E
41	8D	42	17		STA	SBD	in the first position.
44	A9	ØA			LDA	#ØA	Display the letter E
46	8D	42	17		STA	SBD	in the second position.
49	4C	3F	ØØ		JMP	AGAIN	Jump back and display EE again.

8.12.2 Generating a Variable Frequency Square Wave (Example 8.12.3)

The purpose of this example is to illustrate one of the many different ways of defining the Input/Output interface ports of the KIM-1 microcomputer for both input and output. In order to be more specific and practical as well, we will use Example 8.12.3, which may be stated as follows:

Example 8.12.3*. Write a computer program for the KIM-1 which will generate a variable frequency square wave with amplitudes of 0 and +5 volts.

As shown in Figure 8.12-4, the eight pins of Port A are programmed such that one pin (PAØ) is the output and the other seven pins (PA1 to and including PA7) are connected to seven switches which serve as the input control to determine the frequency of the square wave.

It is further desired that the switch connected to PA1 is to be the least significant bit (LSB) and the switch connected to PA7 is to be the most significant bit (MSB). We would also like each switch to act as a 1 when closed, and as a 0 when open. If this is done, the status of the switches define binary numbers from zero (all switches open) to 127_{10} (with all switches closed).

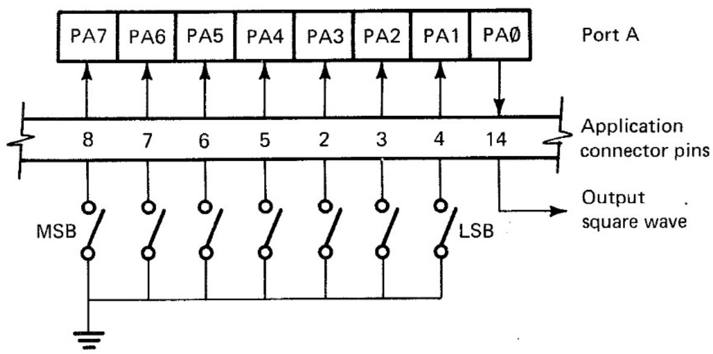


Figure 8.12-4 Input/Output connections for the variable frequency square wave generator (Example 8.12.3)

A Solution to Example 8.12.3. One of the many possible solutions to Example 8.12.3 is given in Figure 8.12-5, which presents in a modified form, both the assembly language program as well as the machine code. It is said to be "modified" form because of the additional column labeled MACHINE CYCLES. This column has been added because of the importance of determining exactly how much time (number of machine cycles) are required to execute each line of code.

* Example 8.12.3 is similar to the "Real Application" example given on page 55 of the KIM-1 USER MANUAL. (MOS Technology Inc.)

Memory Map	INSTRUCTION Byte 1	Byte 2	Byte 3	MACHINE CYCLES	LABEL	OP CODE	OPERAND	MACHINE CODE Remarks
Ø2ØØ	A9	Ø1		2	INIT	LDA	#$01	Define I/O. PAØ=Output=1. PA1-PA7=Input=0.
Ø2Ø2	8D	Ø1	17	4		STA	PADD	PADD = Port A Data Direction Register
Ø2Ø5	EE	ØØ	17	6	START	INC	PAD	Toggle PAØ pin only on Port A.
Ø2Ø8	AD	ØØ	17	4	READ	LDA	PAD	Read switch settings into accumulator.
Ø2ØB	49	FF		2		EOR	#$FF	Complement switch value because closed = 1.
Ø2ØD	4A			2		LSR	A	Shift accumulator one bit to right.
Ø2ØE	AA			2		TAX		Transfer COUNT into X-Index Register.
Ø2ØF	CA			2	DELAY	DEX		Loop to delay by amount specified by
Ø21Ø	1Ø	FD		3,2	BPL	DELAY		COUNT stored in the X-Index Register.
Ø212	3Ø	F1		3		BMI	START	Go back to START.
					PADD = $17Ø1			Defines absolute address of Port A Data Direction Register.
					PAD = $17ØØ			Defines absolute address of Port A Data Register as specified by the KIM Monitor.

Figure 8.12-5 Assembly language, Machine code, and MACHINE CYCLES for a variable frequency square wave generator (Example 8.12.3)

Execution of Example 8.12.3. Enter the program as usual, but before executing the program, load the NMI vector locations 17FA and 17FB with 00 and 1C respectively, so that the [ST] key will function properly. The program begins in memory location Ø2ØØ, so load this memory location, and press the [GO] key. The display will remain dark and the output square wave will be available at pin 14 of the Applications Connector. This output may now be connected to an audio amplifier to serve as a tone generator; or it may be applied to the horizontal input of an oscilloscope and used to calibrate the horizontal sweep of the oscilloscope.

The Output Square Wave. The square wave output is shown in Figure 8.12-6 along with equations for the frequency and period of the square wave as a function of the switch settings. When numerical values are substituted, the resulting frequencies and periods are:

Input Switch Settings Binary	Decimal	*Frequency (f = 1/T) Hz.*	*Period T = 1/f μsec.*
ØØØØØØ	0	21,740	46
111111	127_{10}	760	1362

Input Switch Settings		Frequency f = 1/T (Hz)	Period T = 1/f (μs)
Binary	Decimal		
ØØØØØØ	0	21,740	46
111111	127_{10}	760	1,362

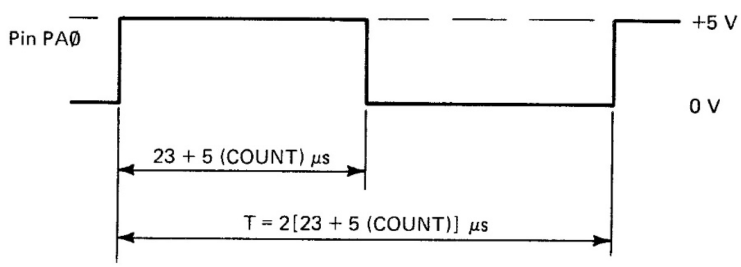


Figure 8.12-6 Output square wave for Example 8.12.3. Note that the period is T, the frequency is 1/T, and COUNT is determined by the input switch setting with $0 \leqslant \text{COUNT} \leqslant 127_{10}$

PROBLEMS

1. Use the KIM keyboard and display to inspect the contents of memory locations (in hex) ØØØØ, Ø4ØØ, 18ØØ, and FFFF. Record the contents as observed. Refer to Figure 8.4.1, the memory map for the KIM-1 microcomputer, and the 6530 Monitor listings to explain your observations.
2. List the contents of the following addresses: 1CØØ through 1CØA, 1746, ØØØØ through ØØ1Ø, and 178Ø through 179Ø.
3. Sketch and label all MCS6502 μP registers. Briefly describe their functions.
4. List the addresses where the following can be found in KIM: ROM, RWM, TIMERS, and I/O.
5. Use KIM keyboard and data keys to clear memory locations $ØØEF through $ØØFE. If this is not possible, explain.
6. Where are the internal registers (A, X, Y, S, P and Z) stored during a single-step operation?

7. Explain why it is necessary to load memory locations 17FA and 17FB before the KIM-1 microcomputer may be used in the single-step mode. Refer to 6530-002 listing between lines 600 and 620 and correlate with the use of the above two memory locations.

8. Refer to Table 8.7-2 which presents the single step execution of Example 8.1 and fill in the table below for each step.

PC	P	A	X	Y	S

9. Study the program below and understand how it works. (There is at least one error.) The intention of the program is to set the contents of locations $ØØØØ through $ØØFF to all ones ($FF).

```
ADDR    CODE                PROGRAM
Ø2ØØ    AE FF               LDX  #$FF
Ø2Ø2    18                  CLA
Ø2Ø3    95 ØØ       LOOP    STA  Ø, X
Ø2Ø5    CA                  DEX
Ø2Ø6    DØ FB               BNE  LOOP
Ø2Ø8    4C 4F  1C           JMP  $1C4F
```

10. If you use the KIM-1 microcomputer in the binary mode, what result would you expect if you added 34 and 27? What would you expect in the decimal mode? Use the solution to Example 8.1 and verify your answers using the KIM-1 microcomputer.

11. Modify the source code for Example 8.1 which is given in Table 8.3-1 so as to check for possible overflow. What would you suggest doing in the event of an overflow?

12. Modify the source code for Example 8.1 which is given in Table 8.3-1 so as to display the sum instead of "jumping to itself" at the end of the program. Display the sum in the four leftmost locations of the display. You will need to study the Monitor Listing for the 6530s in order to do this.

13. Modify the source code for Example 8.1 which is given in Table 8.3-1 for the case where the two positive numbers are each three bytes long. Your program should check for overflow and allow enough memory locations to store the sum. Use the decimal mode. Repeat for binary mode.

14. Write a program for execution on the KIM-1 microcomputer which will output all of the printable ASCII characters to a printer such as the Teletype Model ASR33. In the next line printed, move

all of the characters one character to the right and bring the character shifted off to the right back to the leftmost position. (In other words, circular shift right.) Repeat printing new lines until all characters have been printed in all possible positions. (This is a program which printer technicians find very useful when making printer adjustments.)

15. Punch a paper tape for Example 8.2 and compare to the punched paper tape shown in Figure 8.10-6. Explain any differences.
16. Write a program for execution on the KIM-1 which will display the characters as they are "read in" from a magnetic tape recorder. Display the character which has just been read in the leftmost position of the display and then shift the characters to the right to make room for each new character. (The characters will appear at a rate which may be too fast to permit reading every character reliably.)
17. Write a program for execution on the KIM-1 which will generate a very long string of sync characters, an ASCII SYN character ($16). Record approximately five minutes of only sync characters on a magnetic tape and then play them back using the program written in Problem 16. Vary the volume control and tone control (if your recorder has one) and determine the range for which synchronization is maintained. Determine the optimum settings for your tape recorder.
18. Modify the program given in Table 8.12-1 for Example 8.12.2 so that the two characters E E blink on and off at the rate of about once per second. Although the timers in the 6530s might be used, write a program which will accomplish the same thing. (Not all microcomputers have timers.)
19. Modify the program given in Table 8.12-1 for Example 8.12.2 so that six characters (all of which may be different) will be displayed on the KIM-1 display. (Caution: Be sure to turn off a character before proceeding to turn on the next character.) In order to demonstrate that your program works, display the six characters E, R, R, O, R, and a blank.
20. Modify the program given in Table 8.12-1 for Example 8.12.2 so that more than six characters may be displayed at the rate of six at a time and then shifted one character to the right and displayed again in an endless loop.

21. Add steps to your program for Problem 20 so that the display blinks on and off at the rate of about once per seond with on and off times being equal.
22. The solution to Example 8.12.3 in Figure 8.12-5 provides a frequency range from 760 to 21,740 Hz. What changes would be necessary to provide frequencies up to 50,000 Hz? What about for frequencies below 760 Hz?
23. Modify the program given in Figure 8.12-5 for Example 8.12.3 so that the postiive portion of the square wave represents only 10% of the period instead of 50%.

B

OPERATING PRINCIPLES OF THE KIM-1 MONITOR AND ON-BOARD I/O HARDWARE

B.1 INTRODUCTION

In Chapter 8, detailed methods for operating the KIM-1 system and associated peripheral units are described. This appendix will present some of the principles underlying these methods of operation. There are two principal reasons for such a presentation. First, the KIM-1 is an excellent example of a microprocessor-based system, and the underlying design principles present a useful case study for one who aspires to do such design himself. Second, the KIM-1 user will find an understanding of these principles very helpful in actual operation. For example, the user who wishes to use the on-board peripherals, such as the keyboard and the display for nonmonitor functions, will find an understanding of monitor subroutines an invaluable aid.

Figure 4.2-1 is a flow chart of the KIM-1 monitor program. It will serve as an important reference for most of the discussion in this Appendix.

Another important reference is the KIM-1 monitor listing itself, which is included herein as Section B.8.

B.2 SINGLE STEP OPERATION OF THE KIM MICROCOMPUTER

The normal mode of operation for any digital computer, including the KIM, is that in which instructions are executed sequentially at high speed. Automatic high-speed sequencing accounts for most of the unique capability of such systems. It is often desirable, however, to execute a sequence of instructions at manual speed, either to examine step-by-step program execution and thereby verify program integrity, or to diagnose reasons for program malfunction, i.e., to perform "debugging." The KIM system permits manual sequencing via the keyboard switch labeled SST (single step).

The 6502 CPU chip used in the KIM system has several built-in features which make single-step operation convenient to perform. One such feature is the control output designated as *SYNC*. The 6502 produces a pulse from this terminal during each machine cycle which, if completed, constitutes the fetch of an instruction op code from memory. Since such a cycle signals the end of an instruction execution, the ability to stop the computer when SYNC occurs would accomplish the single-step operation desired (assuming, of course, that the system can be restarted).

One other feature of the 6502 which could be used to accomplish single-step operation is the RDY (READY) control input. If, during the PHASE ONE clock of any machine cycle when a memory read is being performed, the RDY input is pulled low, the completion of the cycle will be inhibited at PHASE TWO clock time. (A memory write cycle will not be inhibited, but the subsequent read cycle will be.) The principal use of the RDY control is in managing transfers between the CPU and slow memory or I/O devices which may not respond to read requests with sufficient speed to keep up with the 6502 clock. Such devices can, by properly energizing the RDY input, force the CPU to wait until it is ready to deliver the requested data. A second use of this input is implemented by inverting and feeding back the SYNC pulse to the RDY input. This will cause the system to stop each time a SYNC pulse occurs. Therefore the system stops after the execution of each instruction. (Actually the SYNC pulse must be stored in a separate flip-flop. The flip-flop will inhibit system operation until it is cleared, at which time the system will fetch and execute the next instruction. Refer to Section 3.1, page 124, of the MOS HARDWARE MANUAL, reference 2, for a detailed description of one mechanization of this scheme.)

The KIM system does not use the RDY control to achieve single-step operation. Recall that one of the principal motivations for establishing single-step operation was to permit a leisurely examination of the result of executing each instruction. The KIM monitor affords a powerful mechanism for performing such an examination because the KIM system implements a procedure whereby, after each step of the program under study is executed, system control is passed back to this monitor. The monitor is then used to examine register contents, data values, etc. To understand the actual mechanization built into the KIM system, refer to Figure B.1. Gate U26, a two-input NAND gate, has as its inputs SYNC from the 6502 and a signal labeled K7. For the time being, assume that K7 is high or at the "1" level. If the single-step switch SST is closed, the occurrence of SYNC will pull the $\overline{\text{NMI}}$ input low, constituting a nonmaskable interrupt input to the 6502. Service of this interrupt is in the form of execution of the monitor routine which lights the display and scans the keyboard. This mode reflects the transfer of control to the monitor as described above. Pushing the GO key on the keyboard will cause other monitor routines (see the discussion of the KIM keyboard below) to be called. These routines will fetch the next user instruction, giving the desired result of stepping through one more step. At this point the process will be repeated.

The perceptive reader will have detected a logical flaw in the above operation. He will ask, "Since the execution of the monitor program is conceptually identical to the execution of the user program, why doesn't the system halt after the execution of each step in the monitor program?" This is the point where the role of the K7 signal must be considered. Examination of the logic diagram will confirm that the K7 signal is produced when an address in *page 7* (address range 1C∅∅ to 1CFF) is present on the KIM address bus. Since the 74145 decoder which generates K7 produces active-low outputs, when such an address is present, the K7 signal is at a low level. Thus the NAND gate driving the $\overline{\text{NMI}}$ input will not produce its active low output when a page 7 address is on the address bus, even though the SYNC signal denoting completion of instruction execution is generated. Since the keyboard/display monitor is located in page 7, it will be executed in uninterrupted fashion. This also has the effect of preventing one from stepping through any of the monitor routines located in page 7.

One final point about single-step operation in the KIM should be mentioned. As discussed more fully in Section B.3 on RESET and STOP operation in the KIM, proper execution of the STOP routine in response to an $\overline{\text{NMI}}$ request demands that the user, as a necessary step in system initialization, load the proper interrupt vectors into the system. Since single-step operation depends on this routine, such loading is necessary to properly enable this mode of operation. In plain English, this means that locations 17FA and 17FB must be loaded with address 1C∅∅, the beginning location of the STOP monitor routine.

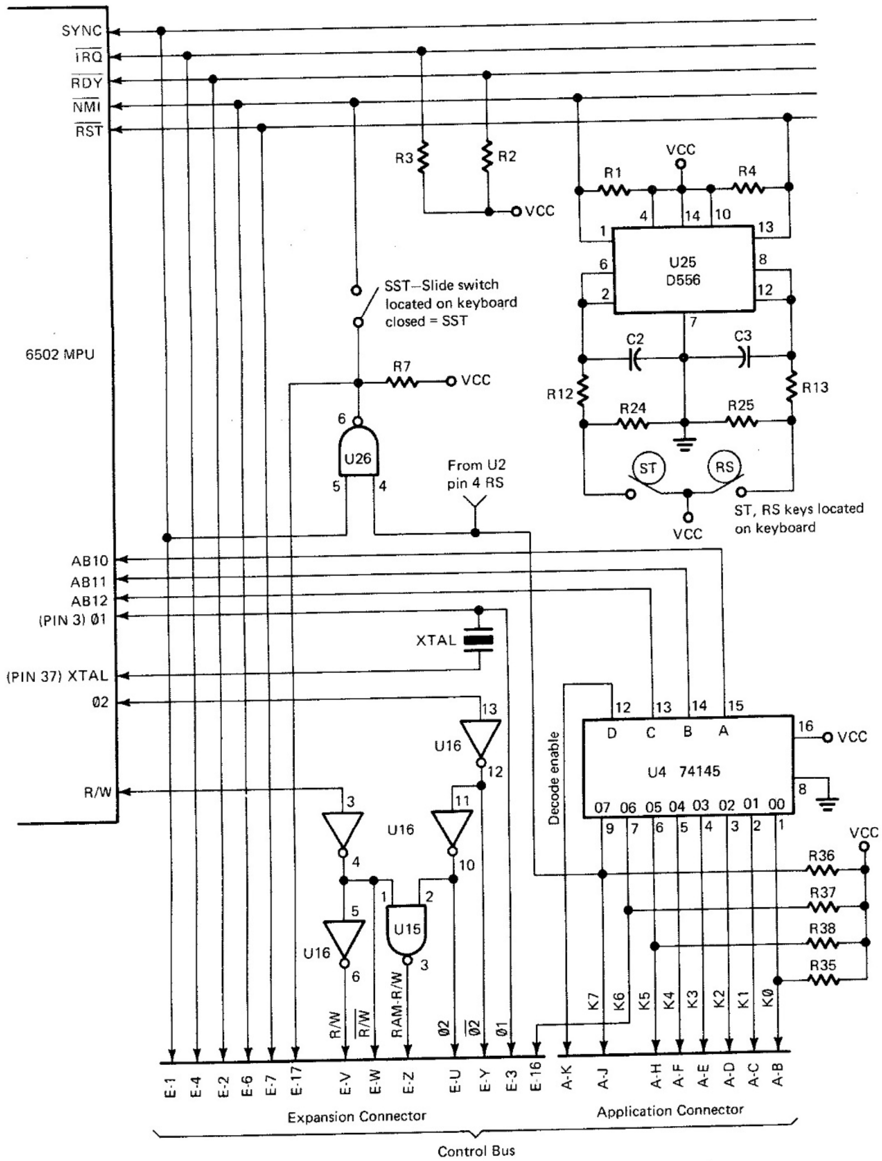


Figure B.1 KIM-1 control and timing circuitry

B.3 RESET AND STOP OPERATIONS

These two operations bear considerable similarity to one another and will therefore be discussed together. Each of them is initiated by depressing a keyboard key. Referring to the logic diagram of Figure B.1 it may be observed that pushing the ST (STOP) key introduces a positive-going voltage onto pins 2 and 6 of U25, a dual timer. This circuit's characteristics are such that this input will produce a negative-going transition at pin 1, which is delivered to the $\overline{\text{NMI}}$ input of the CPU. Before examining the effect of this input, let us digress and point out the principal function of the dual timer circuit. Its characteristics are such that, even if the ST keystroke manifests *switch bounce* (which is typically the case), the timer output will respond only to the *initial positive transition* at its inputs, effectively debouncing the switch.

Let us return now to an examination of the sequence of events which occur when the $\overline{\text{NMI}}$ input is energized. This constitutes a *nonmaskable interrupt* request and results in a somewhat feverish interval of activity by the CPU. The steps which constitute this activity are:

1. When the NMI request arrives, the current contents of the program counter and the status (P) register are pushed onto the system stack. This allows return to the interrupted routine when the interrupt has been serviced.
2. The contents of memory locations 1FFA and 1FFB are fetched from memory and placed in the program counter. This is an automatic step performed by the 6502 hardware in response to the NMI request. Locations 1FFA and 1FFB are ROM locations, which collectively contain the address 1C1C. (The actual addresses are FFFA and FFFB, but, in the KIM-1, the high-order three address bits are ignored in an unexpanded system.)
3. Normal processor activity resumes. This activity consists of placing the program counter contents on the address bus and performing a memory read. The contents of location 1C1C are thus fetched as an instruction op code.
4. Location 1C1C, also a ROM location, contains 6C, a *jump indirect* op code. The next two locations, 1C1D and 1C1E, are thus accessed by the CPU, producing address 17FA, also preprogrammed in ROM. The nature of the jump indirect instruction is such that this address (along with the next consecutive location 17FB) contains the actual address of the next instruction to be executed. Since 17FA and 17FB are RAM locations,

they must be previously loaded by the system user with the address of the first step in the NMI service routine he wishes to call. The STOP routine provided with the KIM monitor begins at location 1CØØ, and, if this is the desired routine, that value should have been loaded into 17FA and 17FB during system initialization.

5. The CPU begins execution of the STOP routine. Its details will not be discussed, but it generally consists of a continuous scan of the keyboard combined with a continuous display of the last memory location referenced by the user program along with the data stored at that location. (See the monitor listing of Section B.8 for details.)

This rather lengthy list of activities can be confusing and some care is necessary to sort out some of the terminology encountered. The various elements of 6502 literature refer to the contents of addresses FFFA and FFFB (which translate into 1FFA and 1FFB in limited address space of the unexpanded KIM) collectively as an *interrupt vector*, since they point into memory toward the requested service routine. In the particular case of the KIM, however, the desire to permit the system user to choose not only the monitor STOP routine but any other routine of his choosing as a response to an NMI request results in several of the extra steps outlined above. Since the 17FA, 17FB locations contain the actual address of the first location of the desired service routine, the contents of these locations are referred to in the KIM literature as the NMI interrupt vectors.

The RST (RESET) key on the KIM results in a sequence of activity which closely resembles that discussed above. The other half of the U25 dual timer debounces the switch, applying a request input to the $\overline{\text{RST}}$ input of the 6502. Once again, a sequence of activity ensues, involving fetching interrupt vectors from locations 1FFC and 1FFD. These locations contain the address 1C22, which, when sent to memory as a request for op code, yields not a jump indirect op code, but the actual first instruction in the RESET routine, which is a system initialization routine.

It is interesting to reflect on the fundamental difference between the NMI and RST responses to service requests. The most obvious difference is that the user has some flexibility in the use of the NMI routine, since by loading interrupt vectors of his choice into the appropriate RAM locations, he can control the response to such requests. It may seem that this flexibility is missing relative to the RST routine because the KIM designers saw no need for substitution of a user-defined response in place of the one provided by the system monitor. The actual reason for the difference is, however, more fundamental. Note that the resources of the monitor program must be used to load the appropriate interrupt vectors into the NMI locations. In order to call upon these resources, the sytem must enter a state in which a scan of the keyboard is performed. Once this state is entered, the keys can be used to call monitor routines to perform desired activities. Placing the system into this state requires the use of the system initialization routine. If this routine also depended upon the preloading of interrupt vectors, note that, somewhat embarrassingly, there would be no way to perform this preloading. The fixed interrupt vectors for RESET are therefore necessary in order to initiate proper system activity. This general procedure is one of several types most often described as *bootstrap procedures*.

B.4 OPERATION OF THE KIM KEYBOARD AND DISPLAY

Chapter 8 describes in some detail the operational manner in which the user can interact with the KIM system via the keyboard and display units. Beyond considerations of simply being able to operate the KIM system, however, it is of considerable educational value to examine in some detail the methods by which these system elements are interfaced to the KIM system, because they constitute excellent examples of peripheral units and their interface to a microprocessor system.

As the management of the display and keyboard interfaces is discussed, reference will be made to the routines which comprise the KIM operating system. The reader will find it useful to refer to the listing of these programs (included in Section B.8) as the discussion proceeds, not only to understand the actual KIM software mechanization, but also to learn by example how such software is written. The KIM designers achieved considerable efficiency by sharing one I/O port for the dual functions of display output and keyboard input and by sharing keyboard scan routines between manual keyboard and Teletype interfaces. This achievement of efficiency is, unfortunately, at the expense of simplicity of understanding in some cases, particularly to one who is examining such methods for the first time. An attempt will be made in this discussion to avoid some of the detail of the implementation which tends to obscure a basic understanding of the system. For example, in the discussion, the display drive mechanization will be separated from the keyboard scan mechanization.

B.4.1 Display Drive in the KIM System

Figure B.2 shows the principal elements in the display drive system. The diagram is simplified by omitting the electrical interface elements. The logical information flow is correct. The peripheral interface adaptor labeled 6530-002 is the key I/O element in this system. The display itself consists of six seven-segment display elements and, as configured, may conveniently be thought of as a two-dimensional array. Control over which *character* is displayed is provided by the A port of the 6530. Lines PAØ through PA6 are routed (through display drivers not shown in the diagram) to the seven LED segments (A through G, as shown in the figure) of each display unit. The other display selection dimension is activated by a 74145 BCD decoder as shown. The active output of this decoder (active-low) is a function of the four-bit code at the ABCD inputs. Any one of 10 binary codes (regular BCD decoding) produces an output from one of the decoder terminals. Thus, to drive a particular display element, the CPU must place the appropriate four-bit code in the SBD (Side B Data Register) positions 1 through 4. If these 4 bit positions are programmed as outputs (by appropriate loading of the Port B Data Register, PBDD), then this code is present on the PB1 through PB4 lines, and will appropriately drive the display array.

Output positions Ø through 3 of the decoder of Figure 4.3 are not used in the display drive. They function as part of the keyboard scan system and as part of the Teletype/keyboard selection control.

Table B.1 indicates the data which must be loaded into the B port Data Register (SBD) to appropriately select any one of the display elements.

TABLE B.1 KIM Display Character Position Selection Codes

To Select Character Position (Left-to-Right)	*Active Decoder Output Must Be*	*Decoder Inputs Must Be D C B A*	*Port B Data Register Contents Must Be* (In Hex)*
1	4	Ø 1 Ø Ø	Ø8
2	5	Ø 1 Ø 1	ØA
3	6	Ø 1 1 Ø	ØC
4	7	Ø 1 1 1	ØE
5	8	1 Ø Ø Ø	1Ø
6	9	1 Ø Ø 1	12

*Since only bits 1 through 4 of the Port B data register affect the decoder, there are a number of other hex values which could be used in each of these postions. The only requirement is that bits 1 through 4 must be identical to the pattern for the values shown.

Since only output positions 1 through 4 affect the display, there are a number of redundant combinations which may be used. The hex values shown in the table are those which assume that Øs are to be used in output positions Ø, 5, 6, and 7. (Actually, the 6530-002 has no PB6 output.) In the KIM, these other outputs are used for other interfacing functions and it may be that in some cases some value other than Ø might be desired.

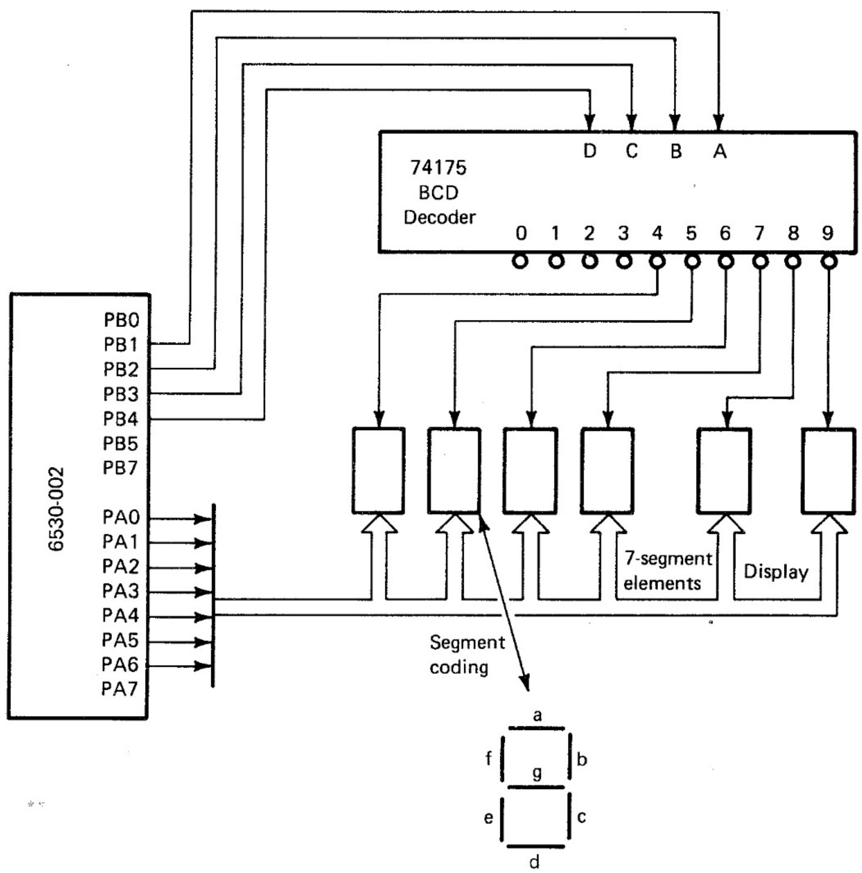


Figure B.2 Simplified block diagram of KIM-1 display drive interface

Table B.2 is a listing of some of the possible 128 characters which can be produced by a seven-segment display. In each case, the ABCDEFG element combination which produces the character is expressed in terms of the hex value which, when produced by the PAØ-6 output bus, will light the appropriate segments. Once again, since the PA7 output does not participate in the display drive, it is assigned the value Ø in the table. Since this terminal is used by the KIM system as a serial Teletype input port, its actual output assignment is immaterial.

In summary, to display a desired character in one of the display positions, the following steps must be performed:

1. Program the peripheral Data Direction Registers PADD and PBDD to configure PAØ-6 and PB1-4 as outputs.
2. Load the B side Data Register (SBD) with a value appropriate for the selection of the desired display element, as given in Table B.1.
3. Load the A side Data Register (SAD) with a value appropriate to the desired character, as given by Table B.2.

A program for performing the above steps is quite simple to write, but is very limited in application, since only one display position is affected by it. A more complex program is used by the KIM monitor, and is designated as SCAND (for Scan Display).

B.4.2 Scand, The KIM Display Drive Program

This program is written as a subroutine and begins on line 1057, page 25, of the KIM monitor listing (Section B.8). The first three steps of the program fetch a byte (two characters) to be displayed and stored in INH, the location reserved for the high byte of the input buffer. The next two steps program the A side of the 6530 port as an output. Next index register X is loaded with Ø9 to serve as a pointer to the rightmost digit position of the display. Index register Y is loaded with Ø3 preparatory to counting the number of characters to be displayed (6 characters, three bytes). Then routine SCAND1 is entered.

SCAND1 starts by fetching the first byte to be displayed, shifts it right four positions to isolate a single character, and then jumps to subroutine CONVD (Convert Display) which starts at line 1085. CONVD searches a conversion table (line 1202) similar in function to Table B.2. It places the appropriate data value found in the table on the PAØ-6 outputs, moves the preset value of index register X (set to Ø9 previously) to SBD, selecting the rightmost digit position of the display, and exe-

TABLE B.2 KIM Display Character Codes

Character	*Symbol Displayed*	*Segments Lit*	*Hex Code*
1		BC	Ø6
2		ABDEG	5B
3		ABCDG	4F
4		BCFG	66
5		ACDGF	6D
6		ACDEFG	7D
7		ABC	Ø7
8		ABCDEFG	7F
9		ABCDFG	6F
Ø		ABCDEF	3F
A		ABCEFG	77
B		CDEFG	7C
C		ADEF	39
D		BCDEG	5E
E		ADEFG	79
F		AEFG	71
G		?	?
H		BCEFG	76
I		BC	Ø6
J		BCDE	1E
K		?	?
L		DEF	38
M		?	?
N		?	?
O		CDEG	5C
P		ABEFG	73
Q		?	?
R		EG	5Ø
S		ACDFG	6D
T		EFG	7Ø
U		CDE	1C
V		?	?
W		?	?
X		?	?
Y		BEFG	72
Z		?	?

cutes a delay routine which lights the display for 500 instruction cycles. CONVD finally increments index X by 2 (to prepare a pointer to the next leftmost digit position the next time around the loop), restores the value in index register Y, and returns to the calling routine SCAND1. This routine refetches the same byte, masks out the most significant 4 bits to isolate the *other* character in the byte and calls CONVD again to display the second character. Upon return, index Y is decremented, and, if it is not zero, (zero signifies that all characters have been displayed) routine SCAND1 is reentered to display the next two characters.

When all six characters have been displayed, signified by Y having been decremented to Ø, the display is turned off by forcing SBD bits 1 through 4 to Ø, so that no display element is selected. Peripheral port A is next reconfigured as an input and a jump to AK (get any key) is performed, initiating a scan of the keyboard.

B.4.3 Operation of the KIM Keyboard

The keyboard mounted on the KIM circuit board, like the display, is in a very real sense nothing more than a peripheral unit which interacts with the CPU via the 6530-002 peripheral interface unit. The function of the various keys in providing system control will not be discussed here because Chapter 8 covers this in great detail. However, a somewhat detailed examination of the control of this keyboard will be made, to promote understanding of the more general problem of peripheral control, in this case, the control of an input device.

Figure B.3 is a simplified logic diagram which illustrates the principal components which are involved in KIM keyboard interaction. The 74145 decoder is the same one previously discussed relative to display management. In that case the active decoder outputs were those labeled 4 through 9, but, as mentioned there, outputs Ø through 3 are the important ones in connection with keyboard interaction. Actually, output 3 is reserved for use as an indicator that the Teletype (rather than the KIM keyboard/display) is the selected controlling device, see Section B.6. Outputs Ø through 2 are connected to rows Ø through 2 of the KIM keyboard, as shown in Figure B.3. Basic keyboard operation consists of sequentially selecting rows Ø, 1, and 2 of the keyboard via an operating program. Any depressed key in a selected row will couple this active-low selection signal (through the vertical keyboard bus line connected to that key) back into one of the port A inputs (assuming that port A is programmed to serve as an input). After the sequential scan of the three rows is complete, the information gathered from the port by the CPU

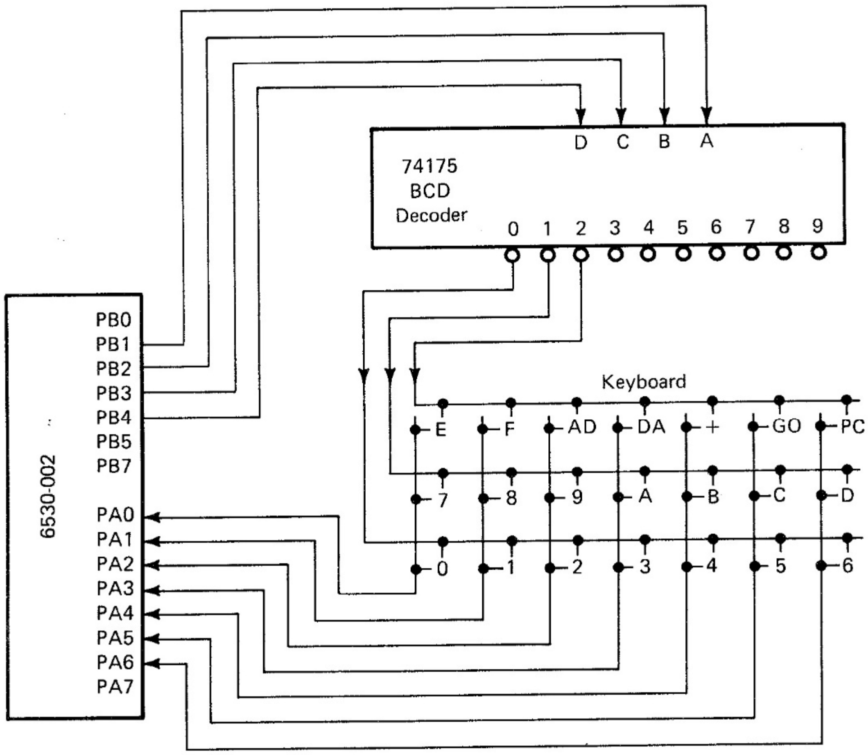


Figure B.3 Simplified block diagram of KIM-1 keyboard scan system

during the scan can be interpreted and the depressed key identified. The CPU can then branch to a routine designed to service that key. Note that the programmer has complete liberty in assigning each key its role in the system, since the software he provides to service each keystroke is whatever he chooses it to be.

B.4.4 AK (Get Any Key) Keyboard Service Routine

To examine one implementation of a keyboard scan routine, we will examine the principal steps found in the KIM keyboard service routine. Recall that, upon completion of the display management routine (SCANDS) the KIM monitor software, after reconfiguring the port A I/O interface as an input, performs a subroutine jump to a routine called AK (any key), which may be found in the KIM monitor listing beginning on line 1037.

The first pair of steps in AK set index register Y and index register X values to Ø3 and Ø1, respectively. The Y value is used to count through the 3 rows of the display and the X value is used, via the BCD decoder and port B output, to sequentially select the three rows. The accumulator is loaded with all 1s (FF hex), and row Ø selected by storing the X value in SBD. The X value is then incremented by 2 to prepare for selection of row 2 during the next pass. Any key depressed on row Ø will send a low-level signal (a "0") back to the corresponding port A input line. Thus, when the next monitor instruction forms the logical AND of SAD contents with the accumulator, the corresponding accumulator bit will become Ø. After this operation, Y is decremented and tested for zero, with a branch back to the scan routine if Y is not zero. Thus three such scans, one for each row will be performed. Any key depression will change the original FF value in the accumulator to some other value.

The next three program steps are included to set up the Teletype selection for potential future use and to take care of the possiblity that the Teletype, even though not being used, may have transferred data to the KIM.

A final EOR (exclusive-or) instruction complements the entire accumulator. Thus, if no keys were depressed during the scan, an all-zero accumulator state will be reached. If one or more keys were depressed, the accumulator will contain a nonzero value.

After the AK routine is finished, a RTS (Return from Subroutine) instruction is executed. This transfers control back to the SCAND routine. Reexamination of this routine, however, discloses that the JSR (Jump to Subroutine) to AK was its last step. However, SCAND, as it turns out, is reached via JSR instructions itself, so the actual return will be to the routine which called SCAND in the first place. There are several such calls in the monitor, all on page 17 of the listing. In particular, note the pair of calls on lines 662 and 664. A return from SCAND the first time with a nonzero accumulator value will result in a second call of SCAND, again followed by a test for zero accumulator. Once again the problem of unreliable logical outputs from mechanical switch contacts is being addressed. The KIM CPU refuses to believe the answer given by just one pass of SCAND, so it calls it one more time just to make sure the key is still depressed during the second scan. If it is, the routine designated GETK (line 667) is entered. This routine has as its function the determination of what key was pressed, information AK does not yield.

B.4.5 GETK and GETKEY Monitor Routines

GETK begins with a jump to subroutine GETKEY, beginning on line 1114 of the monitor listing. GETKEY is a moderately involved routine and its details will not be expounded here. Figure B.4 is a flow chart of GETKEY. Suffice it to say that when it is entered, it scans the KIM keyboard in a manner similar to that done during the AK routine, except that much more processing of the data is performed. The result of this processing is that, when a return from GETKEY occurs, a code is left in the accumulator which uniquely specifies the key that was depressed. Table B.3 shows the association between the keyboard keys and the codes produced by GETKEY. Note that a code with a hex value greater than 15 represents a so-called "illegal" code, and is an indicator that the key was not steadily depressed during the execution of the scan, or that two keys or more were depressed. Such an illegality is rare, because it is extremely difficult to create a keystroke so short as to not appear rather slow to the KIM monitor program. Also, two keys would have to be depressed with near-perfect simultaneity to be detected as an illegal activity by the GETKEY routine.

Returning to the flow of events engendered by the original keystroke, return from GETKEY is back to GETK. The key code having been placed in A, the next few steps of GETK are simply comparisons with code values corresponding to NO KEY, PC, ADDRESS MODE, DATA MODE, STEP (+), and GO key depressions. In each of these cases a jump to a subroutine for handling that particular activity is performed, after which a return to normal keyboard scan will be made. If none of the above codes are found, the code must be a data value held in the lower 4 bit positions of A. In this case, two principal activities must be performed by routine DATA. First, a determination as to whether the keystroke was intended to affect Data or Address portions of the display is made. The routine accomplishes this by examining the MODE value (held at the reserved address ØØFF). If ADDRESS mode is selected, the routine loads the display buffer area with the proper characters, shifts the characters around in such a way as to reflect a right-to-left insertion of new characters to the display, and fetches the data at the resulting memory address. This data is loaded into the display buffer area so that when a return is made to the SCAND routine, the display will indicate the new address and the data stored there. If DATA mode is selected, the address presently held in the display buffer will be used to read memory, shift the low-order data

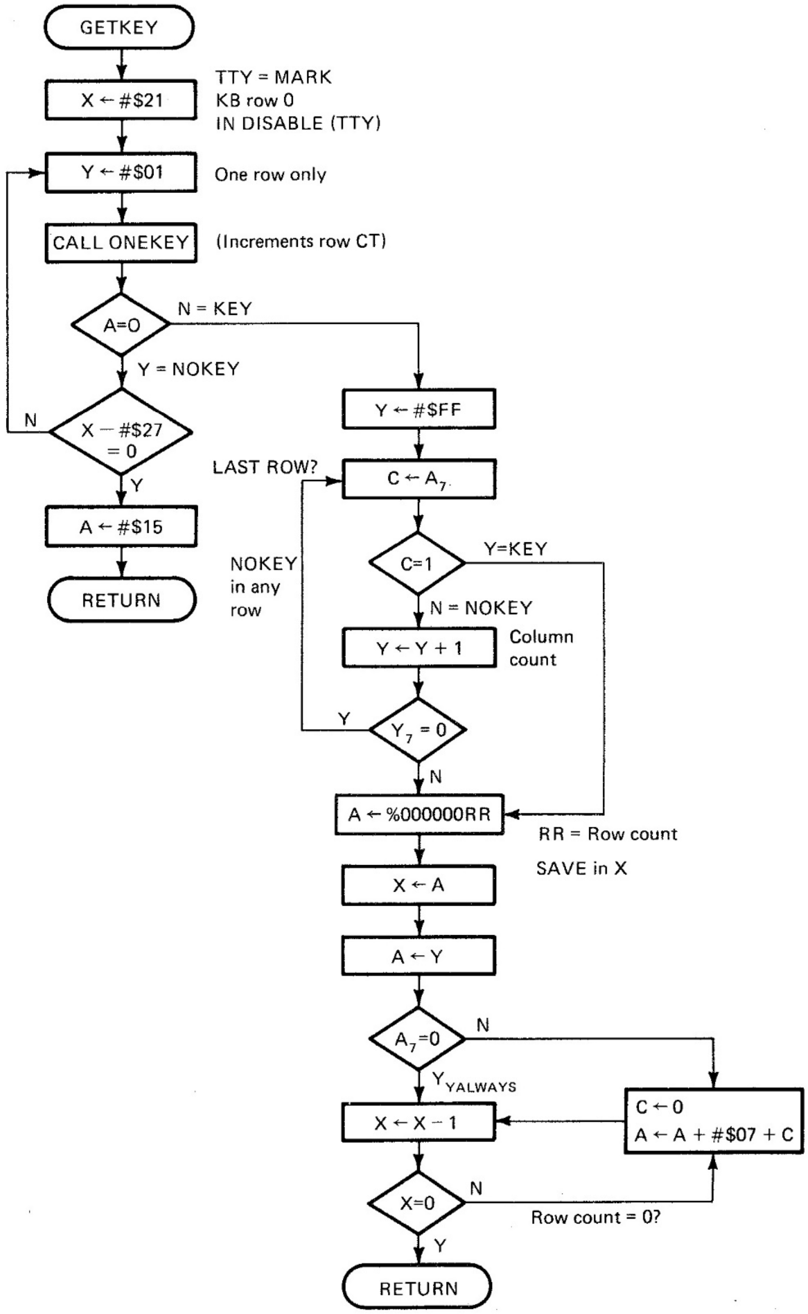


Figure B.4 Flow chart - GETKEY monitor routine

TABLE B.3 KIM Monitor Keystroke Codes

Key Depressed	*Code Placed in A by GETKEY (hex)*
NONE	15
PC	14
AD	10
DA	11
+	12
GO	13
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
0	00
A	A0
B	0B
C	0C
D	0D
E	0E
F	0F
Illegal	> 15

character left one hex position and insert the new value (obtained by GETKEY) in the vacated position. The modified value is then rewritten into memory and a return to SCAND (via START, line 636) is performed.

In reading the KIM monitor listing, it is easy to become confused by some program activities which appear to be meaningless or at least obscure. This is a common reaction when trying to obtain an understanding of another programmer's thought processes via a listing of his program. Novice programmers should remember this when writing their own programs, and should try to follow the following two rules:

1. Be very liberal in your use of comments in your programs.

2. Try to avoid being too clever. It may cost a little extra code to to things in a straightforward manner as opposed to developing short but intricate routines. Be forewarned that such "cleverness" will not only be confusing to another worker trying to understand your program listing, but to yourself when you are trying to use or debug the routine a couple of days after having written it.

B.5 USE OF KIM SUBROUTINES

The discussions of the previous sections involved examinations of some of the activity managed by the KIM monitor. In looking over this operation, you may have noticed that much use is made in the KIM monitor of subroutines performing desired operations. This rather fortunate implementation, one which is a good model for programmers in general, makes available to the KIM user a variety of routines for use outside the normal monitor usage. Table B.4 is a listing of those subroutines included in the portion of the KIM monitor provided for management of display, keyboard, and Teletype. The user can call these routines by simply using the JSR instruction and providing the starting location of the desired routine. They may be thought of as a rather limited "library" available to the system user. The only difference between these subroutines and those you might write for yourself lies in their behavior during the single-step operation of the KIM. Since they all lie in page 7 of memory, you cannot step through them. This does *not* mean that you cannot use them, however, in programs you might like to step through. The only effect will be that in stepping through a JSR instruction which calls one of these routines, you will find that the entire subroutine will be executed before the KIM stops and waits for the next instruction in sequence, the one following the JSR. Note that, in such a case, you are using the subroutine for your own purposes, and the KIM monitor is using the same subroutine, without any conflict between the two uses.

This concept of use of a subroutine library is a very important one. You will find in your future use of microcomputers that the establishment of such a library for use by yourself or others will greatly enhance your ability to use your system without the necessity to develop all software "from the ground up" each time a new and different problem or application is to be examined. You will also find the use of subroutines developed by other users to be an important aid.

TABLE B.4 KIM Monitor Subroutines

Subroutine Name	*Starting Address*	*Description*	*Comments*
INITS	1E88	Initializes system, performing the following steps: 1. Set keyboard to address mode. 2. Program 6530 A port terminals as inputs. 3. Program the lower 6 6530 B port terminals as outputs, and the upper bit as an input (for TTY mode). 4. Loads Ø3 into B port data register (setting up test for TTY mode). 5. Clear decimal mode of KIM arithmetic unit.	
INIT1	1E8C	Same as INITS, except does not affect keyboard mode.	
SCAND	1F19	Drives KIM display, displaying 4-character address and data stored in memory at that address. Branches to AK routine at completion to test for occurrence of KIM keyboard depression. Returns to calling program after AK.	Address to be displayed stored at POINTL, POINTH.
SCANDS	1F1F	Identical to SCAND, except simply displays data at POINTL, POINTH, INH.	6 hex digits to be displayed must be stored at POINTL, POINTH, AND INH.
AK	1EFE	Scans KIM keyboard to see if any key is depressed. Loads A with Ø if no key, with some nonzero number if key.	PA port must be programmed as input.
GETKEY	1F6A	Scans KIM keyboard, producing code (in A) identifying key depressed (see Table B.4).	PA port must be programmed as input.
INCPT	1F63	Increments the 16-bit value in POINTL, POINTH.	

TABLE B.4 (Continued)

Subroutine Name	*Starting Address*	*Description*	*Comments*
OPEN	1FCC	Moves data from INL to POINTL and from INH to POINTH.	
CONVD	1F48	1. Converts hex digit in A to code suitable for driving KIM display element. 2. Selects display element with contents of X (see section on driving KIM display). 3. Drive display element for about 0.5 milliseconds. 4. Increments X by 2, points to next rightmost display element.	Hex digit to be displayed in lower half of A, with upper half cleared. X contains code selecting display element. X will be modified (incremented by 2) by the routine.
GETCH	1E5A	Get one character from Teletype, placing it in A.	X unchanged, Y=FF on return.
GETBYT	1F9D	Get two hex characters from Teletype (as one byte) and pack into INH and INL (See PACK).	X unchanged, Y=Ø on return.
PACK	1FAC	Shift character in A into the INH, INL string. Loses the high-order character of INH. If A does not contain a hex digit, return with no execution.	Clears A on return (if transfer successful).
CHK	1F91	Computes check sum (used with paper tape routine).	
PRTST	1FD5	Prints (via Teletype) a string of ASCII characters from TOP + X to TOP (TOP is a table prestored in ROM).	X must contain pointer into table.
CRLF	1E2F	Prints (via Teletype) a carriage return and line feed.	
OUTCH	1EAØ	Prints (via Teletype) one character (in A).	A contains character to be printed, X unchanged, Y=FF on return.

OUTSP	1E9E	Prints (via Teletype) a space.	X unchanged, Y=FF on return.
PRTPNT	1E1E	Prints (via Teletype) the contents of POINTL and POINTH as 4 hex characters.	
PRTBYT	1E3B	Prints (via Teletype) a byte (in A) as two hex characters.	A contains byte at start and at return.
HEXTA	1E4C	Extracts lower 4 bits of A, prints (via Teletype) as hex character.	
DELAY	1ED4	Delay one Teletype bit time.	Delay depends on baud rate constant (CNTL30, CNTLH30) stored during initialization.
DEHALF	1EEB	Like DELAY, except delays one-half bit time.	

B.6 KIM KEYBOARD/TELETYPE SELECTION CONTROL

The KIM system permits the user to select as his primary I/O device either the integral keyboard/display or a console device such as an ASR33 Teletype. It is instructive to examine the means by which the KIM system goes about implementing the choice of option for I/O.

If one reads only the bare operating procedures for the KIM, he finds that the steps involved in setting up the TTY option include wiring of a connector jumper or a switch which is used to select the option. This may suggest that some wiring of hardware is being accomplished, but in fact the selection is performed almost entirely in software. Refer to Figure B.5 which shows a logic drawing of the 6530-002 peripheral interface unit and the switch wired in to permit implementation of the TTY option. Note that output number Ø3 is tied through this switch to PAØ of the 6530. PAØ should be configured as an input. During the initialization program executed during a RESET operation (a small portion of which is shown in both listing and flow chart form in Figure 4.7), the Port B data register (SBD) is loaded with Ø7, producing an active-low output at the Ø3 terminal of the BCD decoder. If the TTY mode switch is closed, this signal will be coupled into the PAØ input. When the program shown executes a BIT (bit test) instruction, (comparing the contents of PA to Ø1) and then follows this with a BNE (branch on nonzero), the branch will not be taken, since the zero

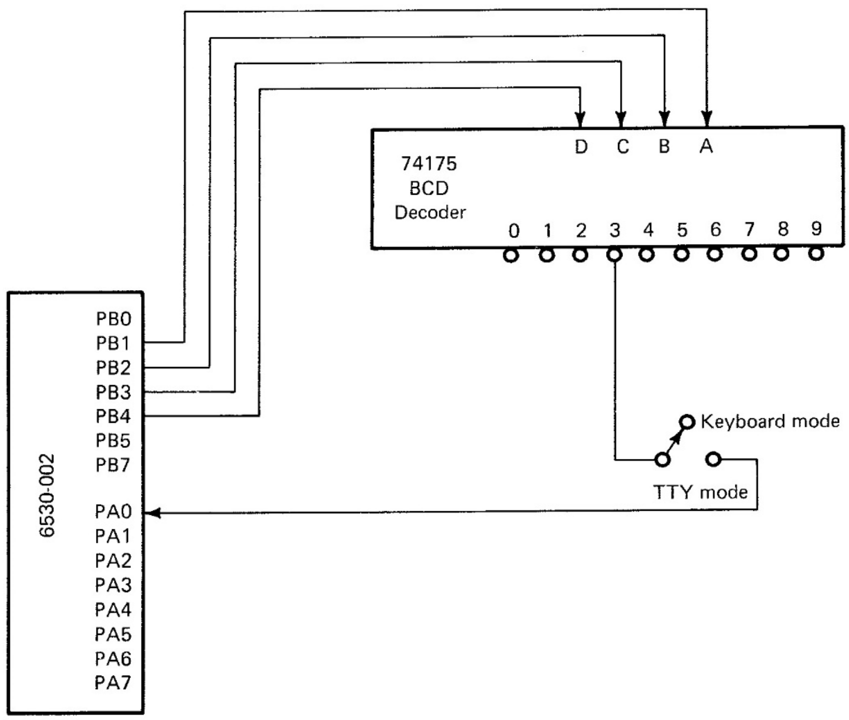


Figure B.5 KIM-1 keyboard/Teletype mode selection circuitry

flag will have been cleared by the bit test operation. Thus the program sequence will not be broken, and the monitor will enter a Teletype handling software mode. If, on the other hand, the BIT, BNE sequence resulted in a zero flag set, indicating that PA∅) was 1, a branch to monitor routine START would result, corresponding to keyboard/display selection. (Note that this means that an unconnected input to PA∅ is interpreted as a high level or a "1").

There are several branch points in the monitor program which require testing for TTY mode. In each case, a procedure similar to the one described is executed by the software. (See, for example, lines 614, 637, and 659 of the KIM monitor listing.)

B.7 TELETYPE I/O INITIALIZATION

Figure B.6 is a diagram which shows the interface circuitry required to properly connect the 6530-002 to a 20-milliampere current-loop

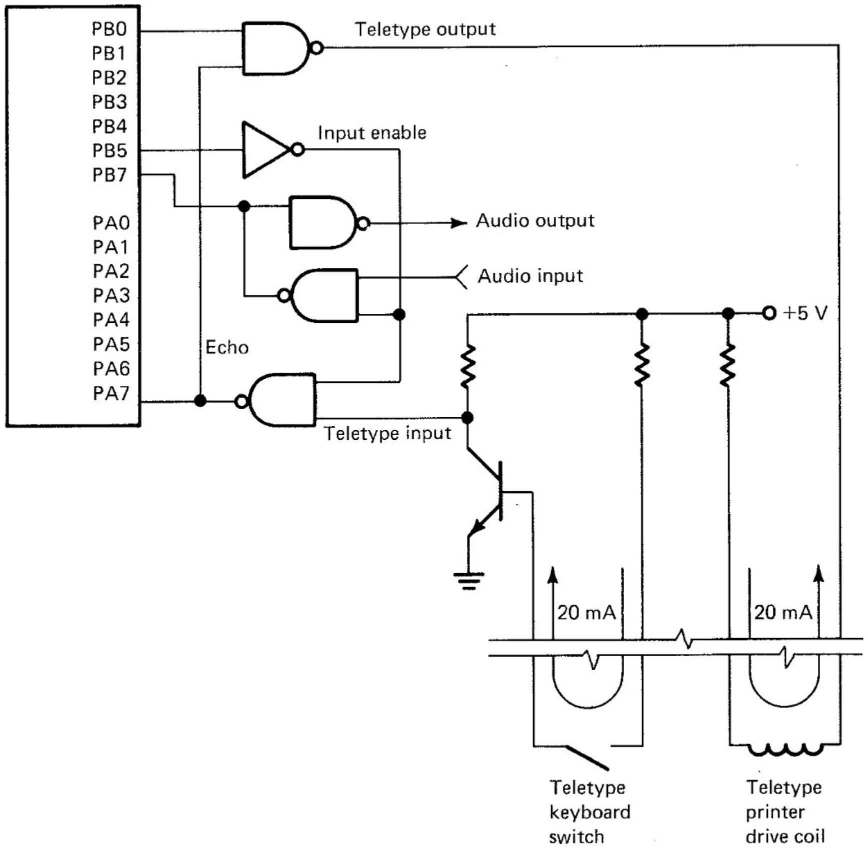


Figure B.6 Teletype current-loop interface circuitry

interface such as is found in a standard ASR33 Teletype. (Also shown is a portion of the audio cassette interface, which will not be discussed here.)

Chapter 5 discusses asynchronous communication principles in some detail, so it is assumed that basic Teletype operation is understood. From the diagram it can be seen that 6530 initialization to support this interface requires that:

1. PBØ be programmed as an output.
2. A high or "1" value be sent to PBØ.
3. PA7 be programmed as an input.
4. PB5 be programmed as an output.

5. PB5 be programmed as an output.
6. A low or "0" value be sent to PB5.

The KIM-1 monitor established the 6530-002 interface during an initialization sequence called INITS (lines 966-977). The reader should verify that the steps listed here are performed during this routine.

B.8 KIM-1 MONITOR PROGRAM LISTING

This section consists of a reproduction of the KIM-1 Monitor listing, included with permission of MOS Technology, Inc.

```
CARD # LOC     CODE       CARD
    3                 ;             666666   555555   333333   000000
    4                 ;             6        5             3   0    0
    5                 ;             6        5             3   0    0
    6                 ;             666666   555555   333333   0    0
    7                 ;             6    6        5        3   0    0
    8                 ;             6    6        5        3   0    0
    9                 ;             666666   555555   333333   000000
   10                 ;
   11                 ;
   12                 ;
   13                 ;                      000000   000000   333333
   14                 ;                      0    0   0    0        3
   15                 ;             ------   0    0   0    0        3
   16                 ;             ------   0    0   0    0   333333
   17                 ;             ------   0    0   0    0        3
   18                 ;                      0    0   0    0        3
   19                 ;                      000000   000000   333333
   20                 ;
   21                 ;
   22                 ;
   23                 ;
   24                 ;
   25                 ;       COPYRIGHT
   26                 ;       MOS TECHNOLOGY, INC
   27                 ;       DATE OCT 18 1975 REV D
   28                 ;
   29                 ;
   30                 ;
   31                 ;       6530-003 IS AN AUDIO CASSETT TAPE
   32                 ;       RECORDER ENTENSION OF THE BASIC
   33                 ;       KIM MONITOR
   34                 ;
   35                 ;       IT FEATURES TWO BASIC ROUTINES
   36                 ;       LOADT-LOAD MEM FROM AUDIO TAPE
   37                 ;       DUMPT-STOR MEM ONTO AUDIO TAPE
   38                 ;
   39                 ;       LOADT
   40                 ;         ID=00      IGNORE ID
   41                 ;         ID=FF      IGN. ID USE SA FOR START ADDR
   42                 ;         ID=01-FE   IGN.ID USE ADDR ON TAPE
   43                 ;
   44                 ;       DUMPT
   45                 ;         ID=00      SHOULD NOT BE USED
   46                 ;         ID=FF      SHOULD NOT BE USED
   47                 ;         ID=01-FE   NORMAL ID RANGE
   48                 ;         SAL        LSB STARTING ADDRESS
   49                 ;         SAH        MSB
   50                 ;         EAL        LSB ENDING ADDRESS
   51                 ;         EAH        MSB
   52                 ;
```

```
CARD # LOC      CODE       CARD
                           ;
                           ;       EQUATES
                           ;       SET UP FOR 6530-002 I/O
                           ;
                           SAD     =$1740              6530 A DATA
                           PADD    =$1741              6530 A DATA DIRECTION
                           SBD     =$1742              6530 B DATA
                           PBDD    =$1743              6530 B DATA DIRECTION
                           CLK1T   =$1744              DIV BY 1 TIME
                           CLK8T   =$1745              DIV BY 8 TIME
                           CLK64T  =$1746              DIV BY 64 TIME
                           CLKKT   =$1747              DIV BY 1024 TIME
                           CLKRDI  =$1747              READ TIME OUT BIT
                           CLKRDT  =$1746              READ TIME
                           ;
       0000                        *=$00EF
                           ;       MPU REG. SAVX AREA IN PAGE 0
                           ;
       00EF                PCL     *=*+1 PROGRAM CNT LOW
       00F0                PCH     *=*+1 PROGRAM CNT HI
       00F1                PREG    *=*+1 CURRENT STATUS REG.
       00F2                SPUSER  *=*+1 CURRENT STACK POINT
       00F3                ACC     *=*+1 ACCUMULATOR
       00F4                XREG    *=*+1 X INDEX
       00F5                YREG    *=*+1 Y INDEX
                           ;
                           ;       KIM FIXED AREA IN PAGE 0
                           ;
       00F6                CHKHI   *=*+1
       00F7                CHKSUM  *=*+1
       00F8                INL     *=*+1 INPUT BUFFER
       00F9                INH     *=*+1 INPUT BUFFER
       00FA                POINTL  *=*+1 LSB OF OPEN CELL
       00FB                POINTH  *=*+1 MSB OF OPEN CELL
       00FC                TEMP    *=*+1
       00FD                TMPX    *=*+1
       00FE                CHAR    *=*+1
       00FF                MODE    *=*+1
                           ;
                           ;       KIM FIXED AREA IN PAGE 23
                           ;
       0100                        *=$17E7
       17E7                CHKL    *=*+1
       17E8                CHKH    *=*+1               CHKSUM
       17E9                SAVX    *=*+3
       17EC                VEB     *=*+6               VOLATILE EXECUTION BLOCK
       17F2                CNTL30  *=*+1               TTY DELAY
       17F3                CNTH30  *=*+1               TTY DELAY
       17F4                TIMH    *=*+1
       17F5                SAL     *=*+1               LOW STARTING ADDRESS
       17F6                SAH     *=*+1               HI STARTING ADDRESS
       17F7                EAL     *=*+1               LOW ENDING ADDRESS
       17F8                EAH     *=*+1               HI ENDING ADDRESS
       17F9                ID      *=*+1
                           ;
                           ;       INTERRUPT VECTORS
                           ;
       17FA                NMIV    *=*+2               STOP VECTOR (STOP=1C00)
       17FC                RSTV    *=*+2               RST VECTOR
       17FE                IRQV    *=*+2               IRQ VECTOR (BRK= 1C00)
                           ;
       1800                        *=$1800
                           ;
                           ;       INIT VOLATILE EXECUTION BLOCK
                           ;       DUMP MEM TO TAPE
                           ;
       1800     A9 AD      DUMPT   LDA     #$AD        LOAD ABSOLUTE INST
```

```
CARD #  LOC     CODE         CARD
        1802   8D EC 17             STA   VEB
        1805   20 32 19             JSR INTVEB
                            ;
        1808   A9 27                LDA   #$27        TURN OFF DATAIN PB5
        180A   8D 42 17             STA   SBD
        180D   A9 BF                LDA   #$BF        CONVERT PB7 TO OUTPUT
        180F   8D 43 17             STA   PBDD
                            ;
        1812   A2 64                LDX   #$64        100 CHARS
        1814   A9 16        DUMPT1  LDA   #$16        SYN CHAR'S
        1816   20 7A 19             JSR   OUTCHT
        1819   CA                   DEX
        181A   D0 F8                BNE   DUMPT1
                            ;
                            ;
        181C   A9 2A                LDA   #'*         START CHAR
        181E   20 7A 19             JSR   OUTCHT
                            ;
        1821   AD F9 17             LDA   ID          OUTPUT ID
        1824   20 61 19             JSR   OUTBT
                            ;
        1827   AD F5 17             LDA   SAL         OUTPUT STARTING
        182A   20 5E 19             JSR   OUTBTC      ADDRESS
        182D   AD F6 17             LDA   SAH
        1830   20 5E 19             JSR   OUTBTC
                            ;
        1833   AD ED 17     DUMPT2  LDA   VEB+1       CHECK FOR LAST
        1836   CD F7 17             CMP   EAL         DATA BYTE
        1839   AD EE 17             LDA   VEB+2
        183C   ED F8 17             SBC   EAH
        183F   90 24                BCC   DUMPT4
                            ;
        1841   A9 2F                LDA   #'/         OUTPUT END OF DATA CHR
        1843   20 7A 19             JSR   OUTCHT
        1846   AD E7 17             LDA   CHKL        LAST BYTE HAS BEEN
        1849   20 61 19             JSR   OUTBT       OUT PUT   NOW OUTPUT
        184C   AD E8 17             LDA   CHKH        CHKSUM
        184F   20 61 19             JSR   OUTBT
                            ;
                            ;
        1852   A2 02                LDX   #$02        2 CHAR'S
        1854   A9 04        DUMPT3  LDA   #$04        EOT CHAR
        1856   20 7A 19             JSR   OUTCHT
        1859   CA                   DEX
        185A   D0 F8                BNE   DUMPT3
                            ;
        185C   A9 00                LDA   #$00        DISPLAY 0000
        185E   85 FA                STA   POINTL      FOR NORMAL EXIT
        1860   85 FB                STA   POINTH
        1862   4C 4F 1C             JMP   START
                            ;
        1865   20 EC 17     DUMPT4  JSR   VEB         DATA BYTE OUTPUT
        1868   20 5E 19             JSR   OUTBTC
                            ;
        186B   20 EA 19             JSR   INCVEB
        186E   4C 33 18             JMP   DUMPT2
                            ;
                            ;       LOAD MEMORY FROM TAPE
                            ;
                            ;
        1871   0F 19        TAB     .WORD LOAD12
        1873   A9 8D        LOADT   LDA   #$8D        INIT VOLATILE EXECUTION
        1875   8D EC 17             STA   VEB         BLOCK WITH STA ABS.
        1878   20 32 19             JSR   INTVEB
                            ;
```

```
  LOC    CODE          CARD
  187B   A9 4C                LDA   #$4C       JUMP TYPE RTRN
  187D   8D EF 17             STA   VEB+3
  1880   AD 71 18             LDA   TAB
  1883   8D F0 17             STA   VEB+4
  1886   AD 72 18             LDA   TAB+1
  1889   8D F1 17             STA   VEB+5
                       ;
  188C   A9 07                LDA   #$07       RESET PB5=0 (DATA IN)
  188E   8D 42 17             STA   SBD
                       ;
  1891   A9 FF         SYNC   LDA   #$FF       CLEAR SAVX FOR SYNC AREA
  1893   8D E9 17             STA   SAVX
                       ;
  1896   20 41 1A      SYNC1  JSR   RDBIT      GET A BIT
  1899   4E E9 17             LSR   SAVX       SHIFT BIT INTO CHAR
  189C   0D E9 17             ORA   SAVX
  189F   8D E9 17             STA   SAVX
  18A2   AD E9 17             LDA   SAVX       GET NEW CHAR
  18A5   C9 16                CMP   #$16       SYN CHAR
  18A7   D0 ED                BNE   SYNC1
                       ;
  18A9   A2 0A                LDX   #$0A       TEST FOR 10 SYN CHARS
  18AB   20 24 1A      SYNC2  JSR   RDCHT
  18AE   C9 16                CMP   #$16
  18B0   D0 DF                BNE   SYNC       IF NOT 10 CHAR  RE-SYNC
  18B2   CA                   DEX
  18B3   D0 F6                BNE   SYNC2
                       ;
                       ;
  18B5   20 24 1A      LOADT4 JSR   RDCHT      LOOK FOR START OF
  18B8   C9 2A                CMP   #'*        DATA CHAR
  18BA   F0 06                BEQ   LOAD11
  18BC   C9 16                CMP   #$16       IF NOT *  SHOULD BE SYN
  18BE   D0 D1                BNE   SYNC
  18C0   F0 F3                BEQ   LOADT4
                       ;
  18C2   20 F3 19      LOAD11 JSR   RDBYT      READ ID FROM TAPE
  18C5   CD F9 17             CMP   ID         COMPARE WITH REQUESTED ID
  18C8   F0 0D                BEQ   LOADT5
  18CA   AD F9 17             LDA   ID         DEFAULT 00 READ RECORD
  18CD   C9 00                CMP   #$00       ANYWAY
  18CF   F0 06                BEQ   LOADT5
  18D1   C9 FF                CMP   #$FF       DEFAULT FF IGNOR SA ON
  18D3   F0 17                BEQ   LOADT6     TAPE
  18D5   D0 9C                BNE   LOADT
                       ;
  18D7   20 F3 19      LOADT5 JSR   RDBYT      GET SA FROM TAPE
  18DA   20 4C 19             JSR   CHKT
  18DD   8D ED 17             STA   VEB+1      SAVX IN VEB+1,2
  18E0   20 F3 19             JSR   RDBYT
  18E3   20 4C 19             JSR   CHKT
  18E6   8D EE 17             STA   VEB+2
  18E9   4C F8 18             JMP   LOADT7
                       ;
  18EC   20 F3 19      LOADT6 JSR   RDBYT      GET SA BUT IGNORE
  18EF   20 4C 19             JSR   CHKT
  18F2   20 F3 19             JSR   RDBYT
  18F5   20 4C 19             JSR   CHKT
                       ;
                       ;
  18F8   A2 02         LOADT7 LDX   #$02       GET 2 CHARS
  18FA   20 24 1A      LOAD13 JSR   RDCHT      GET CHAR(X)
  18FD   C9 2F                CMP   #'/        LOOK FOR LAST CHAR
  18FF   F0 14                BEQ   LOADT8
  1901   20 00 1A             JSR   PACKT      CONVERT TO HEX
```

```
LOC   CODE        CARD
1904  D0 23              BNE    LOADT9      Y=1 NON-HEX CHAR
1906  CA                 DEX
1907  D0 F1              BNE    LOAD13
                  ;
1909  20 4C 19           JSR    CHKT        COMPUTE CHECKSUM
190C  4C EC 17           JMP    VEB         SAVX DATA IN MEMORY
190F  20 EA 19    LOAD12 JSR    INCVEB      INCREMENT DATA POINTER
1912  4C F8 18           JMP    LOADT7
                  ;
1915  20 F3 19    LOADT8 JSR    RDBYT       END OF DATA COMPARE CHKSUM
1918  CD E7 17           CMP    CHKL
191B  D0 0C              BNE    LOADT9
191D  20 F3 19           JSR    RDBYT
1920  CD E8 17           CMP    CHKH
1923  D0 04              BNE    LOADT9
1925  A9 00              LDA    #$00        NORMAL EXIT
1927  F0 02              BEQ    LOAD10
                  ;
1929  A9 FF       LOADT9 LDA    #$FF        ERROR EXIT
192B  85 FA       LOAD10 STA    POINTL
192D  85 FB              STA    POINTH
192F  4C 4F 1C           JMP    START
                  ;
                  ;
                  ;      SUBROUTINES FOLLOW
                  ;
                  ;      SUB TO MOVE SA TO VEB+1,2
                  ;
1932  AD F5 17    INTVEB LDA    SAL
1935  8D ED 17           STA    VEB+1
1938  AD F6 17           LDA    SAH
193B  8D EE 17           STA    VEB+2
193E  A9 60              LDA    #$60        RTS INST
1940  8D EF 17           STA    VEB+3
1943  A9 00              LDA    #$00        CLEAR CHKSUM AREA
1945  8D E7 17           STA    CHKL
1948  8D E8 17           STA    CHKH
194B  60                 RTS
                  ;
                  ;      COMPUTE CHKSUM FOR TAPE LOAD
                  ;      RTN USES Y TO SAVX A
                  ;
194C  A8          CHKT   TAY
194D  18                 CLC
194E  6D E7 17           ADC    CHKL
1951  8D E7 17           STA    CHKL
1954  AD E8 17           LDA    CHKH
1957  69 00              ADC    #$00
1959  8D E8 17           STA    CHKH
195C  98                 TYA
195D  60                 RTS
                  ;
                  ;      OUTPUT ONE BYTE USE Y
                  ;      TO SAVX BYTE
                  ;
195E  20 4C 19    OUTBTC JSR    CHKT        COMP CHKSUM
1961  A8          OUTBT  TAY                SAVX DATA BYTE
1962  4A                 LSR    A           SHIFT OFF LSD
1963  4A                 LSR    A
1964  4A                 LSR    A
1965  4A                 LSR    A
1966  20 6F 19           JSR    HEXOUT      OUT PUT MSD
1969  98                 TYA
196A  20 6F 19           JSR    HEXOUT      OUT PUT LSD.
196D  98                 TYA
```

```
LOC     CODE        CARD
196E    60                  RTS
                    ;
                    ;       CONVERT LSD OF A TO ASCII
                    ;       AND OUTPUT TO TAPE
                    ;
196F    29 0F       HEXOUT  AND     #$0F
1971    C9 0A               CMP     #$0A
1973    18                  CLC
1974    30 02               BMI     HEX1
1976    69 07               ADC     #$07
1978    69 30       HEX1    ADC     #$30
                    ;
                    ;       OUTPUT TO TAPE ONE ASCII
                    ;       CHAR  USE SUB'S ONE + ZRO
                    ;
197A    8E E9 17    OUTCHT  STX     SAVX
197D    8C EA 17            STY     SAVX+1
1980    A0 08               LDY     #$08        START BIT
1982    20 9E 19    CHT1    JSR     ONE
1985    4A                  LSR     A           GET DATA BIT
1986    B0 06               BCS     CHT2
1988    20 9E 19            JSR     ONE         DATA BIT=1
198B    4C 91 19            JMP     CHT3
198E    20 C4 19    CHT2    JSR     ZRO         DATA BIT=0
1991    20 C4 19    CHT3    JSR     ZRO
1994    88                  DEY
1995    D0 EB               BNE     CHT1
1997    AE E9 17            LDX     SAVX
199A    AC EA 17            LDY     SAVX+1
199D    60                  RTS
                    ;
                    ;
                    ;       OUTPUT 1 TO TAPE
                    ;       9 PULSES 138 MICROSEC EACH
                    ;
199E    A2 09       ONE     LDX     #$09
19A0    48                  PHA                 SAVX A
19A1    2C 47 17    ONE1    BIT     CLKRDI      WAIT FOR TIME OUT
19A4    10 FB               BPL     ONE1
19A6    A9 7E               LDA     #126
19A8    8D 44 17            STA     CLK1T
19AB    A9 A7               LDA     #$A7
19AD    8D 42 17            STA     SBD         SET  PB7=1
19B0    2C 47 17    ONE2    BIT     CLKRDI
19B3    10 FB               BPL     ONE2
19B5    A9 7E               LDA     #126
19B7    8D 44 17            STA     CLK1T
19BA    A9 27               LDA     #$27
19BC    8D 42 17            STA     SBD         RESET PB7=0
19BF    CA                  DEX
19C0    D0 DF               BNE     ONE1
19C2    68                  PLA
19C3    60                  RTS
                    ;
                    ;
                    ;       OUTPUT 0 TO TAPE
                    ;       6 PULSES 207 MICROSEC EACH
                    ;
19C4    A2 06       ZRO     LDX     #$06
19C6    48                  PHA                 SAVX A
19C7    2C 47 17    ZRO1    BIT     CLKRDI
19CA    10 FB               BPL     ZRO1
19CC    A9 C3               LDA     #195
19CE    8D 44 17            STA     CLK1T
19D1    A9 A7               LDA     #$A7
```

```
LOC     CODE        CARD
1903    8D 42 17            STA     SBD         SET PB7=1
1906    2C 47 17    ZRO2    BIT     CLKRDI
1909    10 FB               BPL     ZRO2
190B    A9 C3               LDA     #195
190D    8D 44 17            STA     CLK1T
19E0    A9 27               LDA     #$27
19E2    8D 42 17            STA     SBD         RESET PB7=0
19E5    CA                  DEX
19E6    D0 DF               BNE     ZRO1
19E8    68                  PLA                 RESTORE A
19E9    60                  RTS
                    ;
                    ;       SUB TO INC VEB+1,2
                    ;
19EA    EE ED 17    INCVEB  INC     VEB+1
19ED    D0 03               BNE     INCVE1
19EF    EE EE 17            INC     VEB+2
19F2    60          INCVE1  RTS
                    ;
                    ;       SUB TO READ BYTE FROM TAPE
                    ;
19F3    20 24 1A    RDBYT   JSR     RDCHT
19F6    20 00 1A            JSR     PACKT
19F9    20 24 1A    RDBYT2  JSR     RDCHT
19FC    20 00 1A            JSR     PACKT
19FF    60                  RTS
                    ;
                    ;       PACK A=ASCII INTO SAVX
                    ;       AS HEX DATA
                    ;
1A00    C9 30       PACKT   CMP     #$30
1A02    30 1E               BMI     PACKT3
1A04    C9 47               CMP     #$47
1A06    10 1A               BPL     PACKT3
1A08    C9 40               CMP     #$40
1A0A    30 03               BMI     PACKT1
1A0C    18                  CLC
1A0D    69 09               ADC     #$09
1A0F    2A          PACKT1  ROL     A
1A10    2A                  ROL     A
1A11    2A                  ROL     A
1A12    2A                  ROL     A
1A13    A0 04               LDY     #$04
1A15    2A          PACKT2  ROL     A
1A16    2E E9 17            ROL     SAVX
1A19    88                  DEY
1A1A    D0 F9               BNE     PACKT2
1A1C    AD E9 17            LDA     SAVX
1A1F    A0 00               LDY     #$00        Y=0 VALID HEX CHAR
1A21    60                  RTS                 Y=0 VALID HEX
1A22    C8          PACKT3  INY                 Y=1 NOT HEX
1A23    60                  RTS
                    ;
                    ;       GET 1 CHAR FROM TAPE AND RETURN
                    ;       WITH CHAR IN A  USE SAVX+1 TO ASM CHAR
                    ;
1A24    8E EB 17    RDCHT   STX     SAVX+2
1A27    A2 08               LDX     #$08        READ 8 BITS
1A29    20 41 1A    RDCHT1  JSR     RDBIT       GET NEXT DATA BIT
1A2C    4E EA 17            LSR     SAVX+1      RIGHT SHIFT CHAR
1A2F    0D EA 17            ORA     SAVX+1      OR IN SIGN BIT
1A32    8D EA 17            STA     SAVX+1      REPLACE CHAR
1A35    CA                  DEX
1A36    D0 F1               BNE     RDCHT1
                    ;
```

```
LOC   CODE      CARD
1A38  AD EA 17          LDA   SAVX+1       MOVE CHAR INTO A
1A3B  2A                ROL   A            SHIFT OFF PARITY
1A3C  4A                LSR   A
1A3D  AE EB 17          LDX   SAVX+2
1A40  60                RTS
                  ;
                  ;       THIS SUB GETS ONE BIT FROM
                  ;       TAPE AND RETURNS IT IN SIGN OF A
                  ;
1A41  2C 42 17    RDBIT BIT   SBD          WAIT FOR END OF START BIT
1A44  10 FB             BPL   RDBIT
1A46  AD 46 17          LDA   CLKRDT       GET START BIT TIME
1A49  A0 FF             LDY   #$FF         A=256-T1
1A4B  8C 46 17          STY   CLK64T       SET UP TIMER
                  ;
1A4E  A0 14             LDY   #$14
1A50  88          RDBIT3 DEY               DELAY 100 MICROSEC
1A51  D0 FD             BNE   RDBIT3
                  ;
1A53  2C 42 17    RDBIT2 BIT  SBD
1A56  30 FB             BMI   RDBIT2       WAIT FOR NEXT START BIT
                  ;
1A58  38                SEC
1A59  ED 46 17          SBC   CLKRDT       (256-T1)-(256-T2)=T2-T1
1A5C  A0 FF             LDY   #$FF
1A5E  8C 46 17          STY   CLK64T       SET UP TIMER FOR NEXT BIT
                  ;
1A61  A0 07             LDY   #$07
1A63  88          RDBIT4 DEY               DELAY 50 MICROSEC
1A64  D0 FD             BNE   RDBIT4
                  ;
1A66  49 FF             EOR   #$FF         COMPLEMENT SIGN OF A
1A68  29 80             AND   #$80         MASK ALL EXCEPT SIGN
1A6A  60                RTS
                  ;
                  ;       DIAGNOSTICS
                  ;          MEMORY
                  ;          PLLCAL
                  ;
                  ;
                  ;
                  ;       PLLCAL OUTPUT 166 MICROSEC
                  ;       PULSE STRING
                  ;
1A6B  A9 27       PLLCAL LDA  #$27
1A6D  8D 42 17          STA   SBD          TURN OFF DATIN PB5=1
1A70  A9 BF             LDA   #$BF         CONVERT PB7 TO OUTPUT
1A72  8D 43 17          STA   PBDD
                  ;
1A75  2C 47 17    PLL1  BIT   CLKRDI
1A78  10 FB             BPL   PLL1
1A7A  A9 9A             LDA   #154         WAIT 166 MICRO SEC
1A7C  8D 44 17          STA   CLK1T
1A7F  A9 A7             LDA   #$A7         OUTPUT PB7=1
1A81  8D 42 17          STA   SBD
                  ;
1A84  2C 47 17    PLL2  BIT   CLKRDI
1A87  10 FB             BPL   PLL2
1A89  A9 9A             LDA   #154
1A8B  8D 44 17          STA   CLK1T
1A8E  A9 27             LDA   #$27         PB7=0
1A90  8D 42 17          STA   SBD
1A93  4C 75 1A          JMP   PLL1
                  ;
                  ;
```

```
CARD # LOC    CODE        CARD
                    ;      INTERRUPTS PAGE 27
                    ;
       1A96                *=*+$0164    RESERVED FOR TEST
       1BFA   6B 1A  NMIP27 .WORD PLLCAL
       1BFC   6B 1A  RSTP27 .WORD PLLCAL
       1BFE   6B 1A  IRQP27 .WORD PLLCAL
                    ;
                    ;
                    ;
                    ;
                    ;
                    ;            666666   555555   333333   000000
                    ;            6        5             3   0    0
                    ;            6        5             3   0    0
                    ;            666666   555555   333333   0    0
                    ;            6    6        5        3   0    0
                    ;            6    6        5        3   0    0
                    ;            666666   555555   333333   000000
                    ;
                    ;
                    ;
                    ;                     000000   000000   222222
                    ;                     0    0   0    0        2
                    ;            ------   0    0   0    0        2
                    ;            ------   0    0   0    0   222222
                    ;            ------   0    0   0    0   2
                    ;                     0    0   0    0   2
                    ;                     000000   000000   222222
                    ;
                    ;
                    ;
                    ;
                    ;          COPYRIGHT
                    ;          MOS TECHNOLOGY INC.
                    ;          DATE  OCT 13 1975    REV E
                    ;
                    ;    KIM   :TTY INTERFACE
                    ;          :KEYBOARD INTERFACE
                    ;          :7 SEG 6 DIGIT DISPLAY
                    ;
                    ;
                    ;    TTY CMDS:
                    ;          G    GOEXEC
                    ;          CR   OPEN NEXT CELL
                    ;          LF   OPEN PREV. CELL
                    ;          .    MODIFY OPEN CELL
                    ;          SP   OPEN NEW CELL
                    ;          L    LOAD (OBJECT FORMAT)
                    ;          Q    DUMP  FROM OPEN CELL ADDR TO HI LIMIT
                    ;          RO   RUB OUT  - RETURN TO START (KIM)
                    ;               ((ALL ILLEGAL CHAR ARE IGNORED))
                    ;
                    ;    KEYBOARD CMDS:
                    ;        ADDR  SETS MODE TO MODIFY CELL ADDRESS
                    ;        DATA  SETS MODE TO MODIFY DATA IN OPEN CELL
                    ;        STEP  INCREMENTS TO NEXT CELL
                    ;        RST   SYSTEM RESET
                    ;        RUN   GOEXEC
                    ;        STOP  $1C00 CAN BE LOADED INTO NMIV TO
                    ;              USE STOP FEATURE
                    ;        PC    DISPLAY PC
                    ;
                    ;        CLOCK IS NOT DISABLED IN SIGMA 1
                    ;
                    ;
```

```
LOC      CODE        CARD
                     ;
                     ;
1C00                         *=$1C00
                     ;
                     ;
1C00   85 F3         SAVE    STA   ACC         KIM ENTRY VIA STOP (NMI)
1C02   68                    PLA               OR BRK (IRQ)
1C03   85 F1                 STA   PREG
1C05   68            SAVE1   PLA               KIM ENTRY VIA JSR  (A LOST)
1C06   85 EF                 STA   PCL
1C08   85 FA                 STA   POINTL
1C0A   68                    PLA
1C0B   85 F0                 STA   PCH
1C0D   85 FB                 STA   POINTH
1C0F   84 F5         SAVE2   STY   YREG
1C11   86 F4                 STX   XREG
1C13   BA                    TSX
1C14   86 F2                 STX   SPUSER
1C16   20 88 1E              JSR   INITS
1C19   4C 4F 1C              JMP   START
                     ;
1C1C   6C FA 17      NMIT    JMP   (NMIV)      NON-MASKABLE INTERRUPT TRAP
1C1F   6C FE 17      IRQT    JMP   (IRQV)      INTERRUPT TRAP
                     ;
1C22   A2 FF         RST     LDX   #$FF        KIM ENTRY VIA RST
1C24   9A                    TXS
1C25   86 F2                 STX   SPUSER
1C27   20 88 1E              JSR   INITS
                     ;
                     ;
1C2A   A9 FF         DETCPS  LDA   #$FF        COUNT START BIT
1C2C   8D F3 17              STA   CNTH30      ZERO CNTH30
1C2F   A9 01                 LDA   #$01        MASK HI ORDER BITS
1C31   2C 40 17      DET1    BIT   SAD         TEST
1C34   D0 19                 BNE   START       KEYBD SSW TEST
1C36   30 F9                 BMI   DET1        START BIT TEST
1C38   A9 FC                 LDA   #$FC
1C3A   18            DET3    CLC               THIS LOOP COUNTS
1C3B   69 01                 ADC   #$01        THE START BIT TIME
1C3D   90 03                 BCC   DET2
1C3F   EE F3 17              INC   CNTH30
1C42   AC 40 17      DET2    LDY   SAD         CHECK FOR END OF START BIT
1C45   10 F3                 BPL   DET3
1C47   8D F2 17              STA   CNTL30
1C4A   A2 08                 LDX   #$08
1C4C   20 6A 1E              JSR   GET5        GET REST OF THE CHAR
                     ;                         TEST CHAR HERE
                     ;
                     ;
                     ;
                     ;
                     ;
                     ;       MAKE TTY/KB SELECTION
                     ;
1C4F   20 8C 1E      START   JSR   INIT1
1C52   A9 01                 LDA   #$01
1C54   2C 40 17              BIT   SAD
1C57   D0 1E                 BNE   TTYKB
1C59   20 2F 1E              JSR   CRLF        PRT CR LF
1C5C   A2 0A                 LDX   #$0A        TYPE OUT KIM
1C5E   20 31 1E              JSR   PRTST
1C61   4C AF 1D              JMP   SHOW1
                     ;
1C64   A9 00         CLEAR   LDA   #$00
1C66   85 F8                 STA   INL         CLEAR INPUT BUFFER
```

```
CARD # LOC   CODE          CARD
       1C68  85 F9                STA  INH
       1C6A  20 5A 1E    READ     JSR  GETCH        GET CHAR
       1C6D  C9 01                CMP  #$01
       1C6F  F0 06                BEQ  TTYKB
       1C71  20 AC 1F             JSR  PACK
       1C74  4C DB 1D             JMP  SCAN
                         ;
                         ;        MAIN ROTINE FOR KEY BOARD
                         ;        AND DISPLAY
                         ;
       1C77  20 19 1F    TTYKB    JSR  SCAND        IF A=0 NO KEY
       1C7A  D0 D3                BNE  START
       1C7C  A9 01       TTYKB1   LDA  #$01
       1C7E  2C 40 17             BIT  SAD
       1C81  F0 CC                BEQ  START
       1C83  20 19 1F             JSR  SCAND
       1C86  F0 F4                BEQ  TTYKB1
       1C88  20 19 1F             JSR  SCAND
       1C8B  F0 EF                BEQ  TTYKB1
                         ;
       1C8D  20 6A 1F    GETK     JSR  GETKEY
       1C90  C9 15                CMP  #$15
       1C92  10 BB                BPL  START
       1C94  C9 14                CMP  #$14
       1C96  F0 44                BEQ  PCCMD        DISPLAY PC
       1C98  C9 10                CMP  #$10         ADDR MODE=1
       1C9A  F0 2C                BEQ  ADDRM
       1C9C  C9 11                CMP  #$11         DATA MODE=1
       1C9E  F0 2C                BEQ  DATAM
       1CA0  C9 12                CMP  #$12         STEP
       1CA2  F0 2F                BEQ  STEP
       1CA4  C9 13                CMP  #$13         RUN
       1CA6  F0 31                BEQ  GOV
       1CA8  0A          DATA     ASL  A            SHIFT CHAR INTO HIGH
       1CA9  0A                   ASL  A            ORDER NIBBLE
       1CAA  0A                   ASL  A
       1CAB  0A                   ASL  A
       1CAC  85 FC                STA  TEMP         STORE IN TEMP
       1CAE  A2 04                LDX  #$04
       1CB0  A4 FF       DATA1    LDY  MODE         TEST MODE 1=ADDR
       1CB2  D0 0A                BNE  ADDR         MODE=0 DATA
       1CB4  B1 FA                LDA  (POINTL),Y GET DATA
       1CB6  06 FC                ASL  TEMP         SHIFT CHAR
       1CB8  2A                   ROL  A            SHIFT DATA
       1CB9  91 FA                STA  (POINTL),Y STORE OUT DATA
       1CBB  4C C3 1C             JMP  DATA2
                         ;
       1CBE  0A          ADDR     ASL  A            SHIFT CHAR
       1CBF  26 FA                ROL  POINTL       SHIFT ADDR
       1CC1  26 FB                ROL  POINTH       SHIFT ADDR HI
       1CC3  CA          DATA2    DEX
       1CC4  D0 EA                BNE  DATA1        DO 4 TIMES
       1CC6  F0 08                BEQ  DATAM2       EXIT HERE
                         ;
       1CC8  A9 01       ADDRM    LDA  #$01
       1CCA  D0 02                BNE  DATAM1
                         ;
       1CCC  A9 00       DATAM    LDA  #$00
       1CCE  85 FF       DATAM1   STA  MODE
       1CD0  4C 4F 1C    DATAM2   JMP  START
                         ;
       1CD3  20 63 1F    STEP     JSR  INCPT
       1CD6  4C 4F 1C             JMP  START
                         ;
       1CD9  4C C8 1D    GOV      JMP  GOEXEC
```

```
CARD # LOC      CODE        CARD
                            ;
                            ;
                            ;       DISPLAY PC BY MOVING
                            ;       PC TO POINT
                            ;
       1CDC   A5 EF        PCCMD   LDA    PCL
       1CDE   85 FA                STA    POINTL
       1CE0   A5 F0                LDA    PCH
       1CE2   85 FB                STA    POINTH
       1CE4   4C 4F 1C             JMP    START
                            ;
                            ;       LOAD PAPER TAPE FROM TTY
                            ;
       1CE7   20 5A 1E     LOAD    JSR    GETCH       LOOK FOR FIRST CHAR
       1CEA   C9 3B                CMP    #$3B        SMICOLON
       1CEC   D0 F9                BNE    LOAD
       1CEE   A9 00        LOADS   LDA    #$00
       1CF0   85 F7                STA    CHKSUM
       1CF2   85 F6                STA    CHKHI
                            ;
       1CF4   20 9D 1F             JSR    GETBYT      GET BYTE CNT
       1CF7   AA                   TAX                SAVE IN X INDEX
       1CF8   20 91 1F             JSR    CHK         COMPUTE CHKSUM
                            ;
       1CFB   20 9D 1F             JSR    GETBYT      GET ADDRESS HI
       1CFE   85 FB                STA    POINTH
       1D00   20 91 1F             JSR    CHK
       1D03   20 9D 1F             JSR    GETBYT      GET ADDRESS LO
       1D06   85 FA                STA    POINTL
       1D08   20 91 1F             JSR    CHK
                            ;
       1D0B   8A                   TXA                IF CNT=0 DONT
       1D0C   F0 0F                BEQ    LOAD3       GET ANY DATA
                            ;
       1D0E   20 9D 1F     LOAD2   JSR    GETBYT      GET DATA
       1D11   91 FA                STA    (POINTL),Y  STORE DATA
       1D13   20 91 1F             JSR    CHK
       1D16   20 63 1F             JSR    INCPT       NEXT ADDRESS
       1D19   CA                   DEX
       1D1A   D0 F2                BNE    LOAD2
       1D1C   E8                   INX                X=1  DATA RECORD
                            ;                         X=0  LAST RECORD
       1D1D   20 9D 1F     LOAD3   JSR    GETBYT      COMPARE CHKSUM
       1D20   C5 F6                CMP    CHKHI
       1D22   D0 17                BNE    LOADE1
       1D24   20 9D 1F             JSR    GETBYT
       1D27   C5 F7                CMP    CHKSUM
       1D29   D0 13                BNE    LOADER
                            ;
       1D2B   8A                   TXA                X=0  LAST RECORD
       1D2C   D0 B9                BNE    LOAD
                            ;
       1D2E   A2 0C        LOAD7   LDX    #$0C        X-OFF KIM
       1D30   A9 27        LOAD8   LDA    #$27
       1D32   8D 42 17             STA    SBD         DISABLE DATA IN
       1D35   20 31 1E             JSR    PRTST
       1D38   4C 4F 1C             JMP    START
                            ;
       1D3B   20 9D 1F     LOADE1  JSR    GETBYT      DUMMY
       1D3E   A2 11        LOADER  LDX    #$11        X-OFF ERR KIM
       1D40   D0 EE                BNE    LOAD8
                            ;
                            ;       DUMP TO TTY
                            ;       FROM OPEN CELL ADDRESS
                            ;       TO LIMHL,LIMHH
```

```
CARD #  LOC     CODE        CARD
                            ;
        1D42    A9 00       DUMP    LDA    #$00
        1D44    85 F8               STA    INL
        1D46    85 F9               STA    INH         CLEAR RECORD COUNT
        1D48    A9 00       DUMP0   LDA    #$00
        1D4A    85 F6               STA    CHKHI       CLEAR CHKSUM
        1D4C    85 F7               STA    CHKSUM
                            ;
        1D4E    20 2F 1E    DUMP1   JSR    CRLF        PRINT CR LF
        1D51    A9 3B               LDA    #$3B         PRINT SMICOLON
        1D53    20 A0 1E            JSR    OUTCH
        1D56    A5 FA               LDA    POINTL      TEST POINT GT OR ET
        1D58    CD F7 17            CMP    EAL          HI LIMIT GO TO EXIT
        1D5B    A5 FB               LDA    POINTH
        1D5D    ED F8 17            SBC    EAH
        1D60    90 18               BCC    DUMP4
                            ;
        1D62    A9 00               LDA    #$00         PRINT LAST RECORD
        1D64    20 3B 1E            JSR    PRTBYT      0 BYTES
        1D67    20 CC 1F            JSR    OPEN
        1D6A    20 1E 1E            JSR    PRTPNT
                            ;
        1D6D    A5 F6               LDA    CHKHI       PRINT CHKSUM
        1D6F    20 3B 1E            JSR    PRTBYT      FOR LAST RECORD
        1D72    A5 F7               LDA    CHKSUM
        1D74    20 3B 1E            JSR    PRTBYT
        1D77    4C 64 1C            JMP    CLEAR
                            ;
        1D7A    A9 18       DUMP4   LDA    #$18        PRINT 24 BYTE CNT
        1D7C    AA                  TAX                SAVE AS INDEX
        1D7D    20 3B 1E            JSR    PRTBYT
        1D80    20 91 1F            JSR    CHK
        1D83    20 1E 1E            JSR    PRTPNT
                            ;
        1D86    A0 00       DUMP2   LDY    #$00        PRINT 24 BYTES
        1D88    B1 FA               LDA    (POINTL),Y GET DATA
        1D8A    20 3B 1E            JSR    PRTBYT      PRINT DATA
        1D8D    20 91 1F            JSR    CHK         COMP CHKSUM
        1D90    20 63 1F            JSR    INCPT       INCREMENT POINT
        1D93    CA                  DEX
        1D94    D0 F0               BNE    DUMP2
                            ;
        1D96    A5 F6               LDA    CHKHI       PRINT CHKSUM
        1D98    20 3B 1E            JSR    PRTBYT
        1D9B    A5 F7               LDA    CHKSUM
        1D9D    20 3B 1E            JSR    PRTBYT
        1DA0    E6 F8               INC    INL         INCREMENT RECORD CNT
        1DA2    D0 02               BNE    DUMP3
        1DA4    E6 F9               INC    INH
        1DA6    4C 48 1D    DUMP3   JMP    DUMP0
                            ;
        1DA9    20 CC 1F    SPACE   JSR    OPEN          OPEN NEW CELL
        1DAC    20 2F 1E    SHOW    JSR    CRLF          PRINT CR LF
        1DAF    20 1E 1E    SHOW1   JSR    PRTPNT
        1DB2    20 9E 1E            JSR    OUTSP         PRT SPACE
        1DB5    A0 00               LDY    #$00          PRINT DATA SPECIFIED
        1DB7    B1 FA               LDA    (POINTL),Y BY POINT AD = LDA EXT
        1DB9    20 3B 1E            JSR    PRTBYT
        1DBC    20 9E 1E            JSR    OUTSP         PRT SPACE
        1DBF    4C 64 1C            JMP    CLEAR
                            ;
        1DC2    20 63 1F    RTRN    JSR    INCPT         OPEN NEXT CELL
        1DC5    4C AC 1D            JMP    SHOW
                            ;
        1DC8    A6 F2       GOEXEC  LDX    SPUSER
```

```
LOC      CODE          CARD
1DCA     9A                    TXS
1DCB     A5 FB                 LDA   POINTH       PROGRAM RUNS FROM
1DCD     48                    PHA                OPEN CELL ADDRESS
1DCE     A5 FA                 LDA   POINTL
1DD0     48                    PHA
1DD1     A5 F1                 LDA   PREG
1DD3     48                    PHA
1DD4     A6 F4                 LDX   XREG         RESTORE REGS
1DD6     A4 F5                 LDY   YREG
1DD8     A5 F3                 LDA   ACC
1DDA     40                    RTI
                        ;
1DDB     C9 20          SCAN   CMP   #$20         OPEN CELL
1DDD     F0 CA                 BEQ   SPACE
1DDF     C9 7F                 CMP   #$7F         RUB OUT (KIM)
1DE1     F0 1B                 BEQ   STV
1DE3     C9 0D                 CMP   #$0D         NEXT CELL
1DE5     F0 DB                 BEQ   RTRN
1DE7     C9 0A                 CMP   #$0A         PREV CELL
1DE9     F0 1C                 BEQ   FEED
1DEB     C9 2E                 CMP   #'.          MODIFY CELL
1DED     F0 26                 BEQ   MODIFY
1DEF     C9 47                 CMP   #'G          GO EXEC
1DF1     F0 D5                 BEQ   GOEXEC
1DF3     C9 51                 CMP   #'Q          DUMP FROM OPEN CELL TO HI LIMIT
1DF5     F0 0A                 BEQ   DUMPV
1DF7     C9 4C                 CMP   #'L          LOAD TAPE
1DF9     F0 09                 BEQ   LOADV
1DFB     4C 6A 1C              JMP   READ         IGNORE ILLEGAL CHAR
                        ;
1DFE     4C 4F 1C       STV    JMP   START
1E01     4C 42 1D       DUMPV  JMP   DUMP
1E04     4C E7 1C       LOADV  JMP   LOAD
                        ;
1E07     38             FEED   SEC
1E08     A5 FA                 LDA   POINTL       DEC DOUBLE BYTE
1E0A     E9 01                 SBC   #$01         AT POINTL AND POINTH
1E0C     85 FA                 STA   POINTL
1E0E     B0 02                 BCS   FEED1
1E10     C6 FB                 DEC   POINTH
1E12     4C AC 1D       FEED1  JMP   SHOW
                        ;
1E15     A0 00          MODIFY LDY   #$00         GET CONTENTS OF INPUT BUFF
1E17     A5 F8                 LDA   INL          INL AND STOR IN LOC
1E19     91 FA                 STA   (POINTL),Y SPECIFIED BY POINT
1E1B     4C C2 1D              JMP   RTRN
                        ;
                        ;      END OF MAIN LINE
                        ;      SUBROUTINES FOLLOW
                        ;
                        ;
                        ;
                        ;      SUB TO PRINT POINTL,POINTH
                        ;
1E1E     A5 FB          PRTPNT LDA   POINTH
1E20     20 3B 1E              JSR   PRTBYT
1E23     20 91 1F              JSR   CHK
1E26     A5 FA                 LDA   POINTL
1E28     20 3B 1E              JSR   PRTBYT
1E2B     20 91 1F              JSR   CHK
1E2E     60                    RTS
                        ;
                        ;      PRINT STRING OF ASCII CHAR FROM
                        ;      TOP+X TO TOP
                        ;
```

```
CARD # LOC      CODE        CARD
       1E2F   A2 07      CRLF   LDX   #$07
       1E31   BD D5 1F   PRTST  LDA   TOP,X
       1E34   20 A0 1E          JSR   OUTCH
       1E37   CA                DEX
       1E38   10 F7             BPL   PRTST      STOP ON INDEX ZERO
       1E3A   60         PRT1   RTS
                         ;
                         ;      PRINT 1 HEX BYTE AS TWO ASCII CHAR'S
                         ;
       1E3B   85 FC      PRTBYT STA   TEMP
       1E3D   4A                LSR   A          SHIFT CHAR RIGHT 4 BITS
       1E3E   4A                LSR   A
       1E3F   4A                LSR   A
       1E40   4A                LSR   A
       1E41   20 4C 1E          JSR   HEXTA      CONVERT TO HEX AND PRINT
       1E44   A5 FC             LDA   TEMP       GET OTHER HALF
       1E46   20 4C 1E          JSR   HEXTA      CONVERT TO HEX AND PRINT
       1E49   A5 FC             LDA   TEMP       RESTORE BYTE IN A AND RETURN
       1E4B   60                RTS
                         ;
       1E4C   29 0F      HEXTA  AND   #$0F       MASK HI 4 BITS
       1E4E   C9 0A             CMP   #$0A
       1E50   18                CLC
       1E51   30 02             BMI   HEXTA1
       1E53   69 07             ADC   #$07       ALPHA HEX
       1E55   69 30      HEXTA1 ADC   #$30       DEC HEX
       1E57   4C A0 1E          JMP   OUTCH      PRINT CHAR
                         ;
                         ;      GET 1 CHAR FROM TTY
                         ;      RETURN FROM SUB WITH CHAR IN A
                         ;      X IS PRESERVED AND Y RETURNED = FF
                         ;
       1E59   86 FD      GETCH  STX   TMPX       SAVE X REG
       1E5B   A2 08             LDX   #$08       SET UP 8 BIT CNT
       1E5D   A9 01             LDA   #$01
       1E60   2C 40 17   GET1   BIT   SAD
       1E63   D0 22             BNE   GET6
       1E65   30 F9             BMI   GET1       WAIT FOR START BIT
       1E67   20 D4 1E          JSR   DELAY      DELAY 1 BIT
       1E6A   20 EB 1E   GET5   JSR   DEHALF     DELAY 1/2 BIT TIME
       1E6D   AD 40 17   GET2   LDA   SAD        GET 8 BITS
       1E70   29 80             AND   #$80       MASK OFF LOW ORDER BITS
       1E72   46 FE             LSR   CHAR       SHIFT RIGHT CHARACTER
       1E74   05 FE             ORA   CHAR
       1E76   85 FE             STA   CHAR
       1E78   20 D4 1E          JSR   DELAY      DELAY 1   BIT TIME
       1E7B   CA                DEX
       1E7C   D0 EF             BNE   GET2       GET NEXT CHAR
       1E7E   20 EB 1E          JSR   DEHALF     EXIT THIS RTN
                         ;
       1E81   A6 FD             LDX   TMPX
       1E83   A5 FE             LDA   CHAR
       1E85   2A                ROL   A          SHIFT OFF PARITY
       1E86   4A                LSR   A
       1E87   60         GET6   RTS
                         ;
                         ;      INITIALIZATION FOR SIGMA
                         ;
       1E88   A2 01      INITS  LDX   #$01       SET KB MODE TO ADDR
       1E8A   86 FF             STX   MODE
                         ;
       1E8C   A2 00      INIT1  LDX   #$00
       1E8E   8E 41 17          STX   PADD       FOR SIGMA USE SADD
       1E91   A2 3F             LDX   #$3F
       1E93   8E 43 17          STX   PBDD       FOR SIGMA USE SBDD
```

```
LOC    CODE        CARD
1E96   A2 07                LDX   #$07      ENABLE DATA IN
1E98   8E 42 17             STX   SBD       OUTPUT
1E9B   D8                   CLD
1E9C   78                   SEI
1E9D   60                   RTS
                    ;
                    ;       PRINT 1 CHAR  CHAR=A
                    ;       X IS PRESERVED  Y RETURNED = FF
                    ;       OUTSP   PRINTS 1 SPACE
                    ;
1E9E   A9 20        OUTSP   LDA   #$20
1EA0   85 FE        OUTCH   STA   CHAR
1EA2   86 FD                STX   TMPX
1EA4   20 D4 1E             JSR   DELAY     10/11 BIT CODE SYNC
1EA7   AD 42 17             LDA   SBD       START BIT
1EAA   29 FE                AND   #$FE
1EAC   8D 42 17             STA   SBD
1EAF   20 D4 1E             JSR   DELAY
1EB2   A2 08                LDX   #$08
1EB4   AD 42 17     OUT1    LDA   SBD       DATA BIT
1EB7   29 FE                AND   #$FE
1EB9   46 FE                LSR   CHAR
1EBB   69 00                ADC   #$00
1EBD   8D 42 17             STA   SBD
1EC0   20 D4 1E             JSR   DELAY
1EC3   CA                   DEX
1EC4   D0 EE                BNE   OUT1
1EC6   AD 42 17             LDA   SBD       STOP BIT
1EC9   09 01                ORA   #$01
1ECB   8D 42 17             STA   SBD
1ECE   20 D4 1E             JSR   DELAY     STOP BIT
1ED1   A6 FD                LDX   TMPX      RESTORE INDEX
1ED3   60                   RTS
                    ;
                    ;       DELAY 1 BIT TIME
                    ;       AS DETERMEND BY DETCPS
                    ;
1ED4   AD F3 17     DELAY   LDA   CNTH30    THIS LOOP SIMULATES THE
1ED7   8D F4 17             STA   TIMH      DETCPS SECTION AND WILL DELAY
1EDA   AD F2 17             LDA   CNTL30    1 BIT TIME
1EDD   38           DE2     SEC
1EDE   E9 01        DE4     SBC   #$01
1EE0   B0 03                BCS   DE3
1EE2   CE F4 17             DEC   TIMH
1EE5   AC F4 17     DE3     LDY   TIMH
1EE8   10 F3                BPL   DE2
1EEA   60                   RTS
                    ;
                    ;                   DELAY HALF BIT TIME
1EEB   AD F3 17     DEHALF  LDA   CNTH30    DOUBLE RIGHT SHIFT OF DELAY
1EEE   8D F4 17             STA   TIMH      CONSTANT FOR A DIV BY 2
1EF1   AD F2 17             LDA   CNTL30
1EF4   4A                   LSR   A
1EF5   4E F4 17             LSR   TIMH
1EF8   90 E3                BCC   DE2
1EFA   09 80                ORA   #$80
1EFC   B0 E0                BCS   DE4
                    ;
                    ;       SUB TO DETERMINE IF KEY IS
                    ;       DEPRESSED OR COMDITION OF SSW
                    ;             KEY NOT DEP OR TTY MODE     A = 0
                    ;             KEY DEP OR KB MODE          A NOT ZERO
                    ;
                    ;
1EFE   A0 03        AK      LDY   #$03       3 ROWS
```

```
LOC    CODE          CARD
1F00   A2 01                 LDX    #$01         DIGIT 0
                     ;
1F02   A9 FF         ONEKEY LDA     #$FF
1F04   8E 42 17      AK1     STX    SBD           OUTPUT DIGIT
1F07   E8                    INX                  GET NXT DIGIT
1F08   E8                    INX
1F09   2D 40 17              AND    SAD          INPUT SEGMENTS
1F0C   88                    DEY
1F0D   D0 F5                 BNE    AK1
                     ;
1F0F   A0 07                 LDY    #$07
1F11   8C 42 17              STY    SBD
                     ;
1F14   09 80                 ORA    #$80
1F16   49 FF                 EOR    #$FF
1F18   60                    RTS
                     ;
                     ;       SUB  OUTPUT TO 7 SEGMENT DISPLAY
                     ;
1F19   A0 00         SCAND   LDY    #$00         GET DATA SPECIFIED
1F1B   B1 FA                 LDA    (POINTL),Y   BY POINT
1F1D   85 F9                 STA    INH          SET UP DISPLAY BUFFER
1F1F   A9 7F         SCANDS  LDA    #$7F         CHANGE SEG
1F21   8D 41 17              STA    PADD         TO OUTPUT
                     ;
1F24   A2 09                 LDX    #$09         INIT DIGIT NUMBER
1F26   A0 03                 LDY    #$03         OUTPUT 3 BYTES
                     ;
1F28   B9 F8 00      SCAND1  LDA    INL,Y        GET BYTE
1F2B   4A                    LSR    A            GET MSD
1F2C   4A                    LSR    A
1F2D   4A                    LSR    A
1F2E   4A                    LSR    A
1F2F   20 48 1F              JSR    CONVD        OUTPUT CHAR
1F32   B9 F8 00              LDA    INL,Y        GET BYTE AGAIN
1F35   29 0F                 AND    #$0F         GET LSD
1F37   20 48 1F              JSR    CONVD        OUTPUT CHAR
1F3A   88                    DEY                 SET UP FOR NXT BYTE
1F3B   D0 EB                 BNE    SCAND1
1F3D   8E 42 17              STX    SBD          ALL DIGITS OFF
1F40   A9 00                 LDA    #$00         CHANGE SEG
1F42   8D 41 17              STA    PADD         TO INPUTS
1F45   4C FE 1E              JMP    AK           GET ANY KEY
                     ;
                     ;       CONVERT AND DISPLAY HEX
                     ;       USED BY SCAND ONLY
                     ;
1F48   84 FC         CONVD   STY    TEMP         SAVE Y
1F4A   A8                    TAY                 USE CHAR AS INDEX
1F4B   B9 E7 1F              LDA    TABLE,Y      LOOK UP CONVERSION
1F4E   A0 00                 LDY    #$00         TURN OFF SEGMENTS
1F50   8C 40 17              STY    SAD
1F53   8E 42 17              STX    SBD          OUTPUT DIGIT ENABLE
1F56   8D 40 17              STA    SAD          OUT PUT SEGMENTS
                     ;
1F59   A0 7F                 LDY    #$7F         DELAY 500 CYCLES APPROX.
1F5B   88            CONVD1  DEY
1F5C   D0 FD                 BNE    CONVD1
                     ;
1F5E   E8                    INX                 GET NEXT DIGIT NUM
1F5F   E8                    INX                 ADD 2
1F60   A4 FC                 LDY    TEMP         RESTORE Y
1F62   60                    RTS
                     ;
                     ;              SUB TO INCREMENT POINT.
```

```
LOC     CODE        CARD
                    ;
1F63    E6 FA       INCPT   INC     POINTL
1F65    D0 02               BNE     INCPT2
1F67    E6 FB               INC     POINTH
1F69    60          INCPT2  RTS
                    ;
                    ;       GET KEY FROM KEY BOARD
                    ;       RETURN WITH A=KEY VALUE
                    ;       A GT. 15 THEN ILLEGAL OR NO KEY
                    ;
                    ;
1F6A    A2 21       GETKEY  LDX     #$21        START AT DIGIT 0
1F6C    A0 01       GETKE5  LDY     #$01        GET 1 ROW
1F6E    20 02 1F            JSR     ONEKEY
1F71    D0 07               BNE     KEYIN       A=0 NO KEY
1F73    E0 27               CPX     #$27        TEST FOR DIGT 2
1F75    D0 F5               BNE     GETKE5
1F77    A9 15               LDA     #$15        15=NO KEY
1F79    60                  RTS
1F7A    A0 FF       KEYIN   LDY     #$FF
1F7C    0A          KEYIN1  ASL     A           SHIFT LEFT
1F7D    B0 03               BCS     KEYIN2      UNTIL Y=KEY NUM
1F7F    C8                  INY
1F80    10 FA               BPL     KEYIN1
1F82    8A          KEYIN2  TXA
1F83    29 0F               AND     #$0F        MASK MSD
1F85    4A                  LSR     A           DIV BY 2
1F96    AA                  TAX
1F87    98                  TYA
1F88    10 03               BPL     KEYIN4
1F8A    18          KEYIN3  CLC
1F8B    69 07               ADC     #$07        MULT (X-1) TIMES A
1F8D    CA          KEYIN4  DEX
1F8E    D0 FA               BNE     KEYIN3
1F90    60                  RTS
                    ;
                    ;       SUB TO COMPUTE CHECK SUM
                    ;
1F91    18          CHK     CLC
1F92    65 F7               ADC     CHKSUM
1F94    85 F7               STA     CHKSUM
1F96    A5 F6               LDA     CHKHI
1F98    69 00               ADC     #$00
1F9A    85 F6               STA     CHKHI
1F9C    60                  RTS
                    ;
                    ;       GET 2 HEX CHAR'S AND PACK
                    ;       INTO INL AND INH
                    ;       X PRESERVED  Y RETURNED = 0
                    ;       NON HEX CHAR WILL BE LOADED AS NEAREST HEX EQU
                    ;
1F9D    20 5A 1E    GETBYT  JSR     GETCH
1FA0    20 AC 1F            JSR     PACK
1FA3    20 5A 1E            JSR     GETCH
1FA6    20 AC 1F            JSR     PACK
1FA9    A5 F8               LDA     INL
1FAB    60                  RTS
                    ;
                    ;       SHIFT CHAR IN A INTO
                    ;       INL AND INH
                    ;
1FAC    C9 30       PACK    CMP     #$30        CHECK FOR HEX
1FAE    30 1B               BMI     UPDAT2
1FB0    C9 47               CMP     #$47        NOT HEX EXIT
1FB2    10 17               BPL     UPDAT2
```

```
CARD # LOC    CODE       CARD
 1168  1FB4  C9 40      HEXNUM CMP    #$40       CONVERT TO HEX
 1169  1FB6  30 03             BMI    UPDATE
 1170  1FB8  18         HEXALP CLC
 1171  1FB9  69 09             ADC    #$09
 1172  1FBB  2A         UPDATE ROL    A
 1173  1FBC  2A                ROL    A
 1174  1FBD  2A                ROL    A
 1175  1FBE  2A                ROL    A
 1176  1FBF  A0 04             LDY    #$04       SHIFT INTO I/O BUFFER
 1177  1FC1  2A         UPDAT1 ROL    A
 1178  1FC2  26 F8             ROL    INL
 1179  1FC4  26 F9             ROL    INH
 1180  1FC6  88                DEY
 1181  1FC7  D0 F8             BNE    UPDAT1
 1182  1FC9  A9 00             LDA    #$00       A=0 IF HEX NUM
 1183  1FCB  60         UPDAT2 RTS
 1184                   ;
 1185  1FCC  A5 F8      OPEN   LDA    INL        MOVE I/O BUFFER TO POINT
 1186  1FCE  85 FA             STA    POINTL
 1187  1FD0  A5 F9             LDA    INH        TRANSFER INH- POINTH
 1188  1FD2  85 FB             STA    POINTH
 1189  1FD4  60                RTS
 1190                   ;
 1191                   ;
 1192                   ;      END OF SUBROUTINES
 1194                   ;
 1195                   ;      TABLES
 1196                   ;
 1197  1FD5  00         TOP    .BYTE $00,$00,$00,$00,$00,$00,$0A,$0D,'MIK'
 1197  1FD6  00
 1197  1FD7  00
 1197  1FD8  00
 1197  1FD9  00
 1197  1FDA  00
 1197  1FDB  0A
 1197  1FDC  0D
 1197  1FDD  4D 49 4B
 1198  1FE0  20              .BYTE ' ',$13,'RRE',' ',$13
 1198  1FE1  13
 1198  1FE2  52 52 45
 1198  1FE5  20
 1198  1FE6  13
 1199                   ;
 1200                   ;             TABLE HEX TO 7 SEGMENT
 1201                   ;             0   1   2   3   4   5   6   7
 1202  1FE7  BF         TABLE  .BYTE $BF,$86,$DB,$CF,$E6,$ED,$FD,$87
 1202  1FE8  86
 1202  1FE9  DB
 1202  1FEA  CF
 1202  1FEB  E6
 1202  1FEC  ED
 1202  1FED  FD
 1202  1FEE  87
 1203                   ;             8   9   A   B   C   D   E   F
 1204  1FEF  FF                .BYTE $FF,$EF,$F7,$FC,$B9,$DE,$F9,$F1
 1204  1FF0  EF
 1204  1FF1  F7
 1204  1FF2  FC
 1204  1FF3  B9
 1204  1FF4  DE
 1204  1FF5  F9
 1204  1FF6  F1
 1206                   ;
 1207                   ;
 1208                   ;
```

```
CARD # LOC      CODE        CARD
 1209                       ;
 1210                       ;       INTERRUPT VECTORS
 1211                       ;
 1212  1FF7                         *=$1FFA
 1213  1FFA   1C 1C         NMIENT .WORD NMIT
 1214  1FFC   22 1C         RSTENT .WORD RST
 1215  1FFE   1F 1C         IRQENT .WORD IRQT
 1216                              .END

END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 4
NUMBER OF ERRORS =    0,  NUMBER OF WARNINGS =    0
```

```
        SYMBOL TABLE

SYMBOL    VALUE  LINE DEFINED        CROSS-REFERENCES

ACC       00F3         76    587  851
ADDR      1CBE        694    687
ADDRM     1CC8        701    673
AK        1EFE       1037   1080
AK1       1F04       1041   1046
CHAR      00FE         90    950  951  952  959  984  994
CHK       1F91       1141    734  738  741  749  808  914  899  902
CHKH      17E8         97    158  265  289  299  301
CHKHI     00F6         82    730  755  782  799  819 1144 1146
CHKL      17E7         96    156  262  288  297  298
CHKSUM    00F7         83    729  758  783  801  821 1142 1143
CHKT      194C        295    234  237  242  244  256  308
CHT1      1982        336    344
CHT2      198E        341    338
CHT3      1991        342    340
CLEAR     1C64        645    803  836
CLKKT     1747         65   ****
CLKRDI    1747         66    355  361  378  384  499  505
CLKRDT    1746         67    459  471
CLK1T     1744         62    358  364  381  387  501  508
CLK64T    1746         64    461  473
CLK8T     1745         63   ****
CNTH30    17F3        101    613  622 1010 1022
CNTL30    17F2        100    625 1012 1024
CONVD     1F48       1065   1071 1074
CONVD1    1F5B       1094   1095
CRLF      1E2F        908    640  785  829
DATA      1CA8        680   ****
DATAM     1CCC        704    675
DATAM1    1CCE        705    702
DATAM2    1CD0        706    699
DATA1     1CB0        686    698
DATA2     1CC3        697    692
DEHALF    1EEB       1022    947  956
DELAY     1ED4       1010    946  953  986  990  997 1003
DETCPS    1C2A        612   ****
DET1      1C31        615    617
DET2      1C42        623    621
DET3      1C3A        619    624
DE2       1EDD       1013   1018 1027
DE3       1EE5       1017   1015
DE4       1EDE       1014   1029
DUMP      1D42        778    873
DUMPT     1800        121   ****
DUMPT1    1814        131    134
DUMPT2    1833        148    177
DUMPT3    1854        163    166
```

```
SYMBOL   VALUE  LINE DEFINED      CROSS-REFERENCES

DUMPT4   1865      173   152
DUMPV    1E01      873   867
DUMP0    1D48      781   826
DUMP1    1D4E      785   ****
DUMP2    1D86      811   817
DUMP3    1D96      826   824
DUMP4    1D7A      805   792
EAH      17F8      106   151   791
EAL      17F7      105   149   789
FEED     1E07      876   861
FEED1    1E12      882   880
GETBYT   1F9D     1154   732   736   739   746   754   757   770
GETCH    1E5A      940   643   725  1154  1156
GETK     1C8D      667   ****
GETKEY   1F6A     1114   667
GETKE5   1F6C     1115  1119
GET1     1E60      943   945
GET2     1E6D      948   955
GET5     1E6A      947   627
GET6     1E87      962   944
GOEXEC   1DC8      841   711   865
GOV      1CD9      711   679
HEXALP   1FB8     1170   ****
HEXNUM   1FB4     1163   ****
HEXOUT   196F      323   314   316
HEXTA    1E4C      928   922   924
HEXTA1   1E55      933   931
HEX1     1978      328   326
ID       17F9      107   140   224   226
INCPT    1F63     1104   708   749   815   938
INCPT2   1F69     1107  1105
INCVEB   19EA      397   176   258
INCVE1   19F2      400   398
INH      00F9       85   647   760   825  1059  1179  1187
INITS    1E88      966   600   609
INIT1    1E8C      969   636
INL      00F8       84   646   779   823   885  1066  1072  1158  1178  1185
INTVEB   1932      281   123   185
IRQENT   1FFE     1215   ****
IRQP27   1BFE      519   ****
IRQT     1C1F      604  1215
IRQV     17FE      113   604
KEYIN    1F7A     1122  1117
KEYIN1   1F7C     1123  1126
KEYIN2   1F82     1127  1124
KEYIN3   1F8A     1133  1136
KEYIN4   1F8D     1135  1132
LOAD     1CE7      725   727   762   874
LOADER   1D3E      771   759
LOADE1   1D3B      770   756
LOADS    1CEE      728   ****
LOADT    1873      183   231
LOADT4   18B5      216   221
LOADT5   18D7      233   225   228
LOADT6   18EC      241   230
LOADT7   18F8      247   239   259
LOADT8   1915      261   250
LOADT9   1929      270   252   263   266
LOADV    1E04      874   869
LOAD10   192B      271   268
LOAD11   18C2      223   218
LOAD12   190F      258   182
LOAD13   18FA      248   254
LOAD2    1D0E      746   751
```

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
LOAD3	1D1D	754	744
LOAD7	1D2E	764	◆◆◆◆
LOAD8	1D30	765	772
MODE	00FF	91	686 705 967
MODIFY	1E15	884	863
NMIENT	1FFA	1213	◆◆◆◆
NMIP27	1BFA	517	◆◆◆◆
NMIT	1C1C	603	1213
NMIV	17FA	111	603
ONE	199E	353	336 339
ONEKEY	1F02	1040	1116
ONE1	19A1	355	356 368
ONE2	19B0	361	362
OPEN	1FCC	1185	796 828
OUTBT	1961	309	141 157 159
OUTBTC	195E	308	144 146 174
OUTCH	1EA0	984	787 910 934
OUTCHT	197A	333	132 138 155 164
OUTSP	1E9E	983	831 835
OUT1	1EB4	992	999
PACK	1FAC	1164	651 1155 1157
PACKT	1A00	413	251 405 407
PACKT1	1A0F	421	418
PACKT2	1A15	426	429
PACKT3	1A22	433	414 416
PADD	1741	59	970 1061 1079
PBDD	1743	61	128 496 972
PCCMD	1CDC	717	671
PCH	00F0	73	594 719
PCL	00EF	72	591 717
PLLCAL	1A6B	493	517 518 519
PLL1	1A75	498	499 511
PLL2	1A84	505	506
POINTH	00FB	87	170 272 595 696 720 737 790 843 881 897 1106 1188
POINTL	00FA	86	169 271 592 688 691 695 718 740 747 788 812 833 845 877 879 886 900 1058 1104 1186
PREG	00F1	74	589 847
PRTBYT	1E3B	917	795 800 802 807 813 820 822 834 898 901
PRTPNT	1E1E	897	797 809 830
PRTST	1E31	909	642 767 912
PRT1	1E3A	913	◆◆◆◆
RDBIT	1A41	457	200 441 458
RDBIT2	1A53	467	468
RDBIT3	1A50	464	465
RDBIT4	1A63	476	477
RDBYT	19F3	404	223 233 236 241 243 261 264
RDBYT2	19F9	406	◆◆◆◆
RDCHT	1A24	439	209 216 248 404 406
RDCHT1	1A29	441	446
READ	1C6A	648	870
RST	1C22	606	1214
RSTENT	1FFC	1214	◆◆◆◆
RSTP27	1BFC	518	◆◆◆◆
RSTV	17FC	112	◆◆◆◆
RTRN	1DC2	838	859 887
SAD	1740	58	615 623 638 660 943 948 1044 1089 1091
SAH	17F6	104	145 283
SAL	17F5	103	143 281
SAVE	1C00	587	◆◆◆◆
SAVE1	1C05	590	◆◆◆◆
SAVE2	1C0F	596	◆◆◆◆
SAVX	17E9	98	198 201 202 203 204 333 334 345 346 427 430 439 442 443 444 448 451

```
SYMBOL  VALUE  LINE DEFINED        CROSS-REFERENCES

SBD     1742          60   126  195  360  366  383  389  457  467  494  503
                           510  766  974  987  989  992  996 1000 1002 1041
                          1049 1077 1090
SCAN    1DDB         854   652
SCAND   1F19        1057   657  662  664
SCANDS  1F1F        1060  ****
SCAND1  1F28        1066  1076
SHOW    1DAC         829   839  882
SHOW1   1DAF         830   643
SPACE   1D79         828   855
SPUSER  00F2          75   599  608  841
START   1C4F         636   171  273  601  616  658  661  669  706  709  721
                           768  872
STEP    1CD3         708   677
STV     1DFE         872   857
SYNC    1891         197   211  220
SYNC1   1896         200   206
SYNC2   19AB         209   213
TAB     1871         182   189  191
TABLE   1FE7        1202  1087
TEMP    00FC          88   684  689  917  923  925 1085 1099
TIMH    17F4         102  1011 1016 1017 1023 1026
TMPX    00FD          89   940  958  985 1004
TOP     1FD5        1197   909
TTYKB   1C77         657   639  650
TTYKB1  1C7C         659   663  665
UPDATE  1FBB        1172  1169
UPDAT1  1FC1        1177  1181
UPDAT2  1FCB        1183  1165 1167
VEB     17EC          99   122  148  150  173  184  188  190  192  235  238
                           257  282  284  286  397  399
XREG    00F4          77   597  849
YREG    00F5          78   596  850
ZRO     19C4         376   341  342
ZRO1    19C7         378   379  391
ZRO2    19D6         384   385
```

```
INSTRUCTION  COUNT

     ADC         13
     AND          9
     ASL          7
     BCC          4
     BCS          5
     BEQ         26
     BIT         12
     BMI          9
     BNE         44
     BPL         15
     BRK          0
     BVC          0
     BVS          0
     CLC          8
     CLD          1
     CLI          0
     CLV          0
     CMP         38
     CPX          1
     CPY          0
     DEC          2
     DEX         14
     DEY          8
     EOR          2
     INC          7
     INX          5
     INY          2
     JMP         31
     JSR        115
     LDA        108
     LDX         29
     LDY         25
     LSR         22
     NOP          0
     ORA          6
     PHA          5
     PHP          0
     PLA          5
     PLP          0
     ROL         18
     RTI          1
     RTS         28
     SBC          5
     SEC          3
     SED          0
     SEI          1
     STA         81
     STX         14
     STY          7
     TAX          3
     TAY          3
     TSX          1
     TXA          3
     TXS          2
     TYA          4

# SYMBOLS =  204 (LIMIT =  400)      # BYTES = 1690 (LIMIT = 4096)
# LINES = 1242 (LIMIT = 1500)      # XREFS =  646 (LIMIT =  900)
STOP         0
```