MICROCOMPUTERS

MICROCOMPUTERS

MICROCOMPUTERS

MICROCOMPUTERS

KIM-1 USER MANUAL

Publications Number 6500-15B

KIM-1

MICROCOMPUTER MODULE

USER MANUAL

AUGUST 1976

Second Edition

MOS TECHNOLOGY, INC.
950 Rittenhouse Road
Norristown, PA 19401

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

YOUR KIM-1 MICROCOMPUTER MODULE

Congratulations and welcome to the exciting new world of microcomputers! As the owner of a KIM-1 Microcomputer Module, you now have at your disposal a completely operational, fully tested, and very capable digital computer which incorporates the latest in microprocessor technology offered by MOS Technology, Inc. By selecting the KIM-1 module, you have eliminated all of the problems of constructing and debugging a microcomputer system. Your time is now available for learning the operation of the system and beginning immediately to apply it to your specific areas of interest. In fact, if you will follow a few simple procedures outlined in this manual, you should be able to achieve initial operation of your KIM-1 module within a few minutes after unpacking the shipping container.

Your KIM-1 module has been designed to provide you with a choice of operating features. You may choose to operate the system using only the keyboard and display included as part of the module. Next, you may add a low cost audio cassette tape recorder to allow storage and retrieval of your programs. Also, you may add a serial interfaced teleprinter to the system to provide keyboard commands, hard-copy printing, and paper tape read or punch capability.

At the heart of your KIM-1 system is an MCS 6502 Microprocessor Array operating in conjunction with two MCS 6530 arrays. Each MCS 6530 provides a total of 1024 bytes of Read-only Memory (ROM), 64 bytes of Random Access Memory (RAM), 15 Input/Output pins, and an Interval Timer. Stored permanently in the ROM's of the MCS 6530 arrays are the monitor and executive programs devised by MOS Technology, Inc. to control the various operating modes of the KIM-1 system.

The KIM-1 system is intended to provide you with a capable micro-computer for use in your "real-world" application. Accordingly, the system includes a full 1024 bytes of RAM to provide data and program storage for your application program. In addition, you are provided with 15 bidirectional input/output pins to allow interface control of your specific application. Finally, one of the interval timers included in the system is available for generation of time base signals required by your application.

Your KIM-1 system comes to you complete with all components mounted and tested as a system. You need not worry about timing signals (we've included a 1MHz crystal oscillator on the module), interface logic or levels between system components, or interface circuitry to peripheral devices. In fact, you need only apply the indicated power supply voltages to the designated pins to achieve full operation of your KIM-1 system.

We recommend that you read all of this manual before applying power to or attempting to operate your KIM-1 module. In the order presented, you will find:

- Chapter 2 - "hints and kinks" to help you achieve initial system operation
- Chapter 3 - a more detailed description of the KIM-1 system hardware and software
- Chapter 4 - operating procedures for all system modes
- Chapter 5 - an example of a typical application program using all of the features of the KIM-1 system.

At some future time, you may find it desirable to expand the KIM-1 system to incorporate more memory, different types of memory, or additional input/output capability. Again, we have tried to make system expansion as simple as possible with all required interface signals brought out to a special connector on the module. Watch for:

Chapter 6 - a guide to system expansion for increasing both memory and input/output capability

Despite our best efforts to provide you with a fully operable and reliable system, you might encounter some difficulties with your KIM-1 module. If so, refer to:

Chapter 7 - some guidance on warranty and service procedures for your KIM-1 module

Following the basic text of this manual, you will find a series of Appendices intended to provide you with detailed information on certain specialized subjects of interest to you in understanding the operation of the KIM-1 system.

Lastly, since this manual cannot presume to provide all of the technical information on the hardware or programming aspects of the MCS 6502 microprocessor array, we are including with your KIM-1 system two additional manuals for your reference. The Hardware Manual defines the various elements of the system, their electrical and interface characteristics, and the basic system architecture and timing. The Programming Manual provides the detailed information required to write effective programs using the MCS 6502 instruction program set.

So much for introductory comments! Now lets get started and see if we can get your KIM-1 Microcomputer Module doing some real work for you.

CHAPTER 2

GETTING STARTED

This chapter is intended to guide you through the first important steps in achieving initial operation of your KIM-1 Microcomputer Module. We will ask you to perform certain operations without explanation at this time as to why they are being done. In later sections of this manual, full explanations will be offered for every operating procedure.

2.1 PARTS COMPLEMENT

After unpacking the shipping container for your KIM-1, you should have located the following items:

3 Books - KIM-1 Users Manual
Hardware Manual
Programming Manual

1 Programming Card

1 System Schematic

1 KIM-1 Module

1 Connector (Already mounted on the Module)

1 Hardware Packet

1 Warranty Card

You may wish to save the shipping container and packing material should you need to return your KIM-1 module to us at some future date.

2.2 A FEW WORDS OF CAUTION

WARNING

Your KIM-1 module includes a number of MOS integrated circuits. All such circuits include protective devices to prevent damage resulting from inadvertant application of high voltage potentials to the pins of the device. However, normal precautions should be taken to prevent the application of high voltage static discharges to the pins of an MOS device.

Immediately before removal of the packing material from your KIM-1 module, you should develop the following precautionary habits:

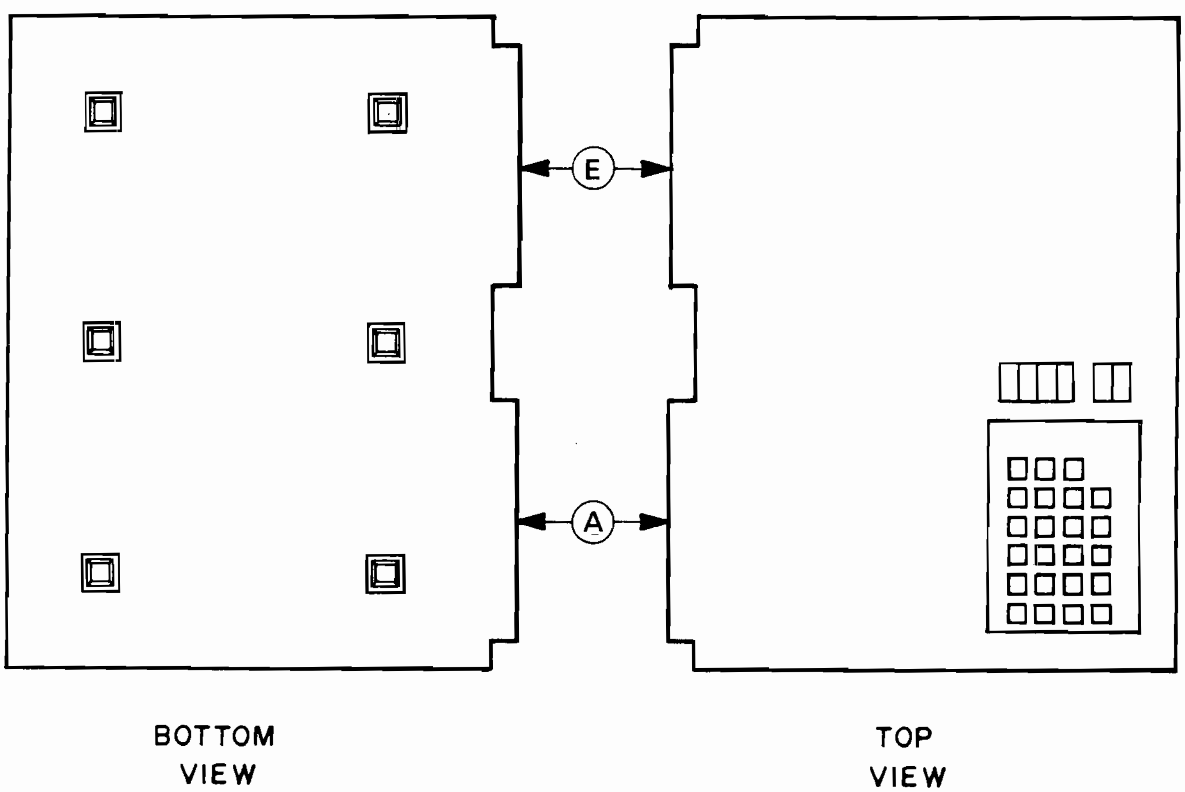
1. Discharge any static charge build up on your body by touching a ground connection before touching any part of your KIM-1 module. (This precaution is especially important if you are working in a carpeted area)
2. Be certain that soldering irons or test equipment used on the KIM-1 module are properly grounded and not the source of dangerously high voltage levels.

On a different subject, after unpacking your KIM-1 module, you will note the presence of a potentiometer. This adjustment has been set at the factory to insure correct operation of the audio cassette interface circuits. It should never be necessary for you to change the position of this potentiometer.

2.3 FIRST STEPS

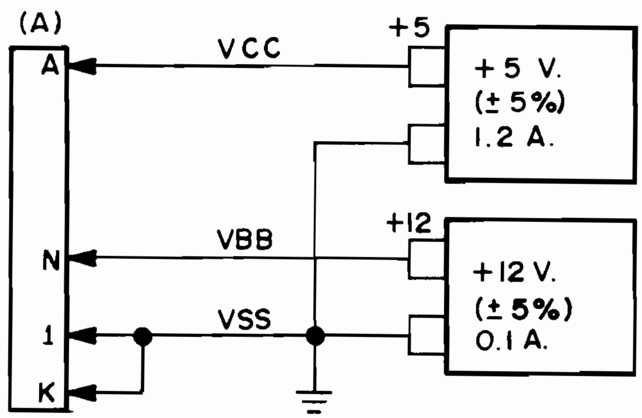
After unpacking the KIM-1 module, locate the small hardware packet and install the rubber pads provided. The rubber pads are located at the bottom of the module (see attached sketch) and act both to lift the card off your work surface and to provide mechanical support for the module while you depress keys.

Place the module such that the keyboard is to your lower right and observe that two connector locations extend from the module to your left. The connector area on the lower left is referred to as the Application connector (A). You will note that a 44 pin board edge connector is already installed at this location. The connector area to the upper left is for use by you for future system expansion and is referred to as the Expansion connector (E).



KIM-1 Module
FIGURE 2.1

Remove the (A) connector from the module and connect the pins as shown in the sketch.



Power Supply Connections
FIGURE 2.2

Reinstall the (A) connector making certain that the orientation is correct.

Note 1: The +12 volt power supply is required only if you will be using an audio cassette recorder in your system.

Note 2: The jumper from pin A-K to Vss (Pin A-1) is essential for system operation. If you expand your system later, this jumper will be removed and we'll tell you what to do to pin A-K.

Note 3: If you don't have the proper power supplies already available, you may wish to construct the low cost version shown with schematic and parts list in Appendix D. In any event, your power supply must be regulated to insure correct system operation and must be capable of supplying the required current levels indicated in the sketch.

Now, recheck your connections, turn on your power supplies, and depress [RS] (reset). You should see the LED display digits light as your first check that the system is operational. If not, recheck your hookup or refer to Appendix C (In Case of Trouble).

2.4 LETS TRY A SIMPLE PROGRAM

Assuming that you have completed successfully all of the steps thus far, a simple program now can be tried to demonstrate the operation of the system and increase your confidence that everything works properly. We'll be using only the keyboard and display on the module for this example. (In the next two sections we'll worry about the teleprinter and the audio cassette).

For our first example, we will add two 8 bit binary numbers together and display the result. We presume that you are familiar with the hexadecimal representation of numbers and the general rules for binary arithmetic.

First check and be sure that the slide switch in the upper right corner of the keyboard is pushed to the left (SST Mode is OFF). Now proceed with the following key sequence:

Press Keys	See On Display	Step #
[AD]	xxxx xx	1
[0] [0] [0] [2]	0002 xx	2
[DA]	0002 xx	3
[1] [8]	0002 18	4
[+] [A] [5]	0003 A5	5
[+] [0] [0]	0004 00	6
[+] [6] [5]	0005 65	7
[+] [0] [1]	0006 01	8
[+] [8] [5]	0007 85	9
[+] [F] [A]	0008 FA	10
[+] [A] [9]	0009 A9	11
[+] [0] [0]	000A 00	12
[+] [8] [5]	000B 85	13
[+] [F] [B]	000C FB	14
[+] [4] [C]	000D 4C	15
[+] [4] [F]	000E 4F	16
[+] [1] [C]	000F 1C	17

What you have just done is entered a program and stored it in the RAM at locations 0002 through 000F. You should have noticed the purpose of several special keys on your keyboard:

- [AD] - selects the address entry mode
- [DA] - selects the data entry mode
- [+] - increments the address without changing the entry mode
- [0] TO [F] - 16 entry keys defining the hex code for address or data entry

You've noticed as well that your display contains 6 digits. The four on the left are used to display the hex code for an address. The two on the right show the hex code for the data stored at the address shown. Therefore, when you pressed [AD] (step 1) and [0] [0] [0] [2] (step 2), you defined the address entry mode, selected the address 0002, and displayed the address 0002 in the four left-most display digits. Incidentally, when we show an "x" in the display chart, we mean that we don't know what will be displayed and we "don't care."

Next you pressed [DA] (step 3) followed by [1] [8] (step 4). Here, you have defined the data entry mode and entered the value 18 to be stored at your selected address 0002. Of course, the 18 then was displayed in the two right-most digits of your display.

You remained in the data entry mode but began to press [+] followed by a two digit number (steps 5 to 17). Note that each depression of the [+] key caused the address displayed to increase by one. The hex keys following the [+] key continued to enter the data field of the display. This procedure is merely a convenience when a number of successive address locations are to be filled.

If you made any mistakes in pressing the keys, you should have noticed that correcting an error is simply a matter of reentering the data until the correct numbers show on the display.

The program you have entered is a simple loop to add two 8 bit binary numbers together and present the result on the display. For a programmer, the listing of the program entered might appear as follows:

```
POINTL              = $FA
POINTH              = $FB
START               = $1C4F
0000                VAL1
0001                VAL2
0002    18          PROG     CLC
0003    A5 00                LDA VAL1
0005    65 01                ADC VAL2
0007    85 FA                STA POINTL
0009    A9 00                LDA #ØØ
000B    85 FB                STA POINTH
000D    4C 4F 1C             JMP START
```

Stated in simple terms, the program will clear the carry flag (CLC), load VAL1 into the accumulator (LDA VAL1), add with carry VAL2 to the accumulator (ADC VAL2), and store the result in a location POINTL (STA POINTL). A zero value is stored in a location POINTH (LDA #ØØ and STA POINTH) and the program jumps to a point labelled START (JMP START). This pre-stored program will cause the display to be activated and will cause the address field of your display to show the numbers stored in locations POINTH and POINTL. Note that the result of the addition has already been stored in location POINTL.

The hex codes appearing next to the address field of the listing are exactly the numbers you entered to store the program. We refer to these as machine language codes. For example, 4C is the hex code for the JMP instruction of the microprocessor. The next two bytes of the program define 1C4F (START) as the jump address.

As yet, you are not able to run the program because you have not yet entered the two variables (VAL1 and VAL2). Lets try an actual example:

Press Keys	See On Display	Step #
AD	000F 1C	17A
0 0 F 1	00F1 xx	17B
DA 0 0	00F1 00	18
AD	00F1 00	19
0 0 0 0	0000 xx	20
DA 0 2	0000 02	21
+ 0 3	0001 03	22
+ GO	0002 18	23

Steps 17A, 17B, and 18 insure that the binary arithmetic mode is selected.

Steps 19 to 21 store the hex value 02 in location 0000 (VAL1). Step 22 stores the hex value 03 in location 0001 (VAL2). Now we are ready to run the program. In step 23, the GO key causes the program to execute and the result, 05, appears in the right two digits of the address display. Although the problem appears trivial, it illustrates the basic principles of entering and executing any program as well as providing a fairly high assurance level that your KIM-1 module is operating properly.

You should try one more example using your stored program. Repeat steps 17A to 23 but substitute the value FF for VAL1 and VAL2 at locations 0000 and 0001. Now when you press the GO key, your display should read:

00FE xx

The answer is correct because:

	FF =	1111 1111
+	FF =	1111 1111
	FE	1111 1110

Try some more examples if you wish and then let's move on to the rest of the system.

2.5 ADDING A TAPE RECORDER

In the previous section, you entered and executed a program. If you turn off the power supplies to the system, your program is lost since the memory into which you stored your program is volatile. If you require the same program again, you would have to repower the system and reenter the program as in the previous example.

The KIM-1 system is designed to work with an audio cassette tape recorder/player to provide you with a medium for permanent storage of your programs or data. The cassette with recorded data may be reread by the system as often as you wish. In this section, you will connect the audio cassette unit to the system and verify its operation.

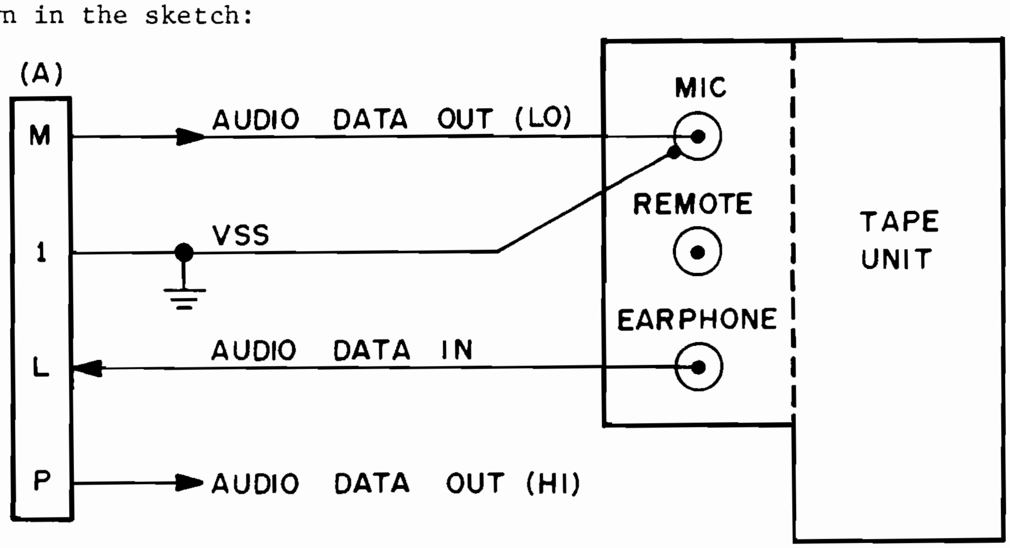
The recording technique used by the KIM-1 system and the interface circuits provided have been selected to insure trouble-free operation with virtually any type and any quality level audio cassette unit. (We have demonstrated correct operation with a tape unit purchased for less than $20.00 from a local discount outlet). In addition, tapes recorded on one unit may be played back to the system on a different unit if desired. We recommend, of course, that you make use of the best equipment and best quality tapes you have available.

In selecting a tape unit for use with your KIM-1 system, you should verify that it comes equipped with the following features:

1. An earphone jack to provide a source of recorded tape data to the KIM-1 system.
2. A microphone jack to allow recording of data from the KIM-1 system on the tape.
3. Standard controls for Play, Record, Rewind, and Stop.

Note: You should avoid certain miniaturized tape equipment intended for dictating applications where the microphone and speaker are enclosed within the unit and no connections are provided to external jacks. If such equipment is used, you will have to make internal modifications to reach the desired connection points.

To connect your tape unit to the KIM-1 module, turn off the power supplies and remove the connector (A) from the module. Add the wires shown in the sketch:



Audio Tape Unit Connections
FIGURE 2.3

Keep the leads as short as possible and avoid running the leads near sources of electrical interference. The connections shown are for typical, portable type units. The Audio Data Out (LO) signal has a level of approximately 15 mv. (peak) at pin M. Should you desire to use more expensive and elaborate audio tape equipment, you may prefer to connect the high level (1 volt peak) audio signal available at pin P to the "LINE" input of your equipment.

Return the connector (A) to its correct position on the KIM-1 module and turn on the power supplies. To verify the operation of your audio cassette equipment, try the following procedures:

1. Reenter the sample program following the procedures outlined in the previous section (2.4). Try the sample problem again to be sure the system is working correctly.
2. Install a cassette in your tape equipment and REWIND to the limit position.
3. Define the starting and ending address of the program to be stored and assign an identification number (ID) to the program.

Press Keys				See On Display	Step #
AD				xxxx xx	1
0	0	F	1	00F1 xx	2
DA		0	0	00F1 00	3
AD				00F1 00	4
1	7	F	5	17F5 xx	5
DA		0	0	17F5 00	6
+		0	0	17F6 00	7
+		1	0	17F7 10	8
+		0	0	17F8 00	9
+		0	1	17F9 01	10
AD				17F9 01	11
1	8	0	0	1800 xx	12

You will recall that the program we wish to store on tape was loaded into locations 0000 to 000F of the memory. Therefore, we define a starting address for recording as 0000 and store this in locations 17F5 and 17F6 (Steps 4 to 7). We define an ending address for recording as <u>one more than</u> the last step of our program and stored the value 0010 (= 000F + 1) in locations 17F7 and 17F8 (Steps 8,9). Finally we pick an arbitrary ID as 01 and store this value at location 17F9 (Step 10).

Note that before we use the audio cassette unit for recording or playing back, we <u>must</u> put 00 in location 00F1 (Steps 1,2 and 3).

The starting address of the tape recording program is 1800. In Steps 11 and 12 we set this address value into the system. If we were to press GO, the system would proceed to load data on to the magnetic tape. But first, we'd better start the tape!

4. Select the Record/Play mode of the tape recorder. Wait a few seconds for the tape to start moving and now:

   Press GO

5. The display will go dark for a short time and then will relight showing:

   0000 xx

6. As soon as the display relights, the recording is finished and you should STOP the tape recorder.

Now, you should verify that the recording has taken place correctly. This can be proven by reading the tape you have just recorded. Proceed as follows:

1. Rewind the tape cassette to its starting position.
2. Turn off the system power supplies and then later, turn them back on.

This has the effect of destroying your previously stored program which you already have recorded on tape.

3. Prepare the system for reading the tape as follows:

Press Keys	See On Display	Step #
RS		
AD	xxxx xx	1
0 0 F 1	00F1 xx	2
DA 0 0	00F1 00	3
AD	00F1 00	4
1 7 F 9	17F9 xx	5
DA	17F9 xx	6
0 1	17F9 01	7
AD	17F9 01	8
1 8 7 3	1873 xx	9
GO	(Dark)	10

The KIM-1 system is now looking for tape input data with the ID label 01. Recall that this is the same ID label we assigned when we recorded the program.

4. If your tape unit has a volume control, set the control at approximately the half way point.

5. If your tape unit has a tone control, set the control for maximum treble.

6. Now, turn on the tape using the PLAY mode. The tape will move forward and the system will accept the recorded data. As soon as the data record (ID=01) has been read, the display should relight showing:

   0000 xx

You may now stop the tape unit. If the display relights and shows;

FFFF xx

this means that the selected record has been located and read but that an error has occurred during the reading of the data. In this case, press the RS key and repeat the read tape procedures from the beginning. If the FFFF still shows on the display, repeat the entire recording and playback procedures checking each step carefully. If the problem persists, refer to Appendix C, (In Case of Trouble).

If the tape continues to run and the display does not relight, this means that the system has been unsuccessful in reading any data back from the tape. In this case, repeat the entire recording and playback procedures checking each step carefully. If the problem persists, refer to Appendix C, (In Case of Trouble).

7. Assuming that you have read the tape successfully, you now may verify that the program has been restored to memory by trying a sample problem. (02 + 03 = 05)

NOTE: The KIM-1 interface circuits for the audio tape system are designed so that you do not require special test equipment to set up correct operating levels. If you have followed the procedures indicated, the tape system should work without the need of any adjustments by you.

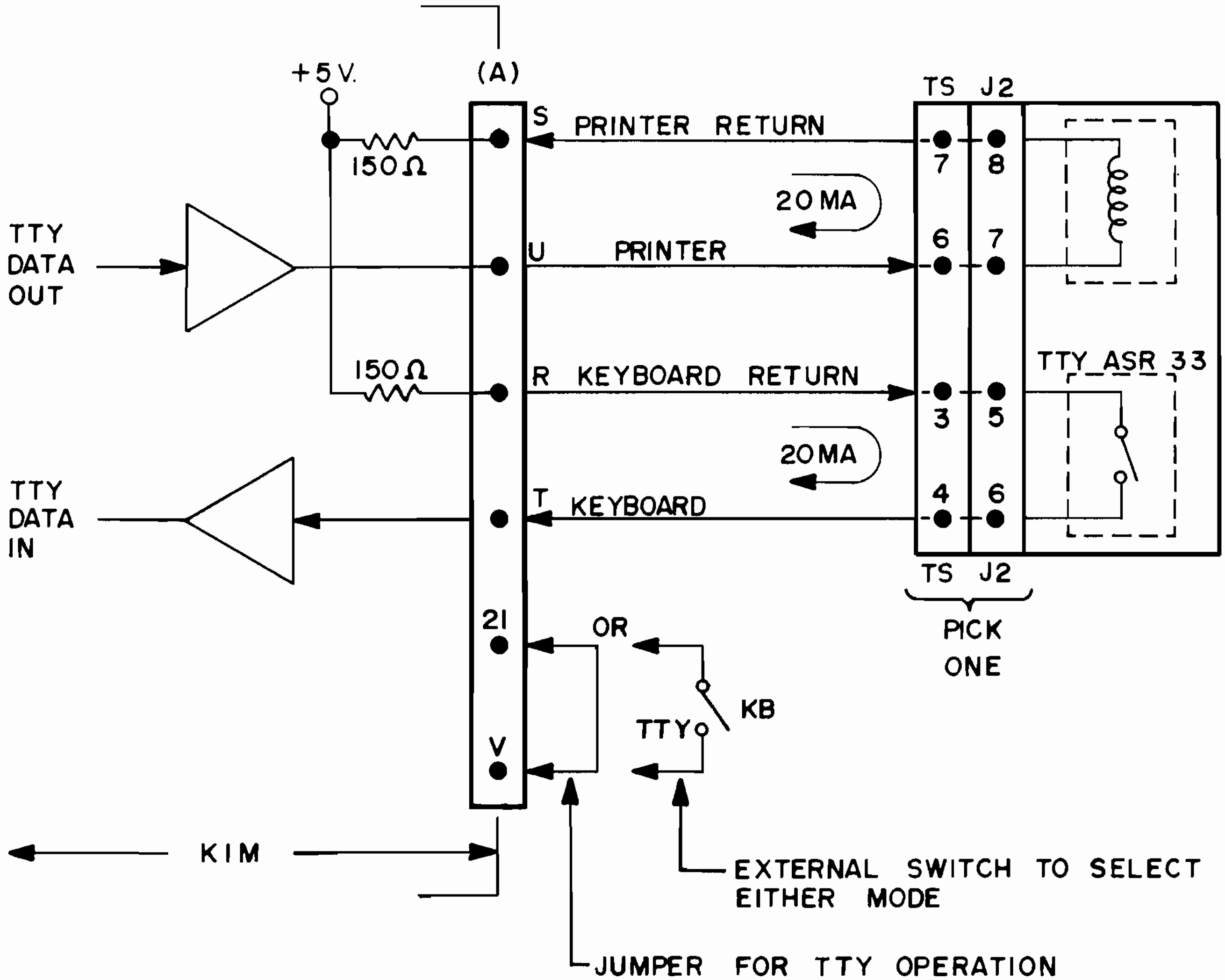
2.6 *ADDING A TELEPRINTER*

If you have access to a serial teleprinter, you may add such a unit to the KIM-1 system with very little effort. One of the more commonly available units of this type is the Teletype Model 33ASR which we will use for the purposes of illustration in this section. However, if you have available different equipment, you may use the information presented here as a guide to connecting your specific unit. In any case, we recommend you follow the directions offered by the equipment manufacturer in his instruction manual to effect the desired wiring and connection options.

The KIM-1 provides for a 4 wire interface to the TTY. Specifically, the "20 MA loop" configuration should be used and you should check that your TTY has been wired for this configuration. If not, you may easily change from "60 MA loop" to "20 MA loop" configurations following the manufacturers directions. The KIM-1 has been designed to work properly only with a teleprinter operating in full duplex mode. Check the literature supplied with your teleprinter if you are unsure if your unit is properly configured. You are not restricted to units with specific bit rates (10 CPS for TTY) since the KIM-1 system automatically adjusts for a wide variety of data rates (10CPS, 15CPS, 30CPS, etc.).

To connect the TTY to the system, proceed as follows:

1. Turn off system power and remove connector (A) from the module.
2. Add the wires shown in the sketch to connector (A) and to the appropriate connector on the TTY unit.



TTY Connections
FIGURE 2.4

3. The jumper wire from A-21 to A-V is used to define for the KIM-1 system that a teleprinter will be used as the only input/display device for the system. If you expect to use both TTY and the KIM-1 keyboard/display, you should install the switch shown instead of the jumper. Now, the switch, when open, will allow use of the keyboard and display on the KIM-1 module and, when closed, will select the teleprinter as the input/display device. (Of course, you may use a clip-lead instead of the switch if you desire).

4. Be sure pins A-21 and A-V are connected. Reinstall connector (A) and return power to the system. Turn-on the TTY.

5. Press the RS key on the KIM-1 module then press the RUB OUT key on the TTY. This step is most important since the KIM-1 system adjusts automatically to the bit rate of the serial teleprinter and requires this first key depression to establish this rate.

If everything is working properly you should immediately observe a message being typed as follows:

KIM

This is a prompting message telling you that the TTY is on-line and the KIM-1 system is ready to accept commands from the TTY keyboard.

Should the prompting message not be typed press the RS key on the KIM-1 keyboard and then the RUB OUT key on the TTY. If the "KIM" message still is not typed, recheck all connections and the TTY itself and try again. If the problem persists, refer to Appendix C, (In Case of Trouble).

6. Assuming that the TTY is operable, you may now try a simple group of operations to verify correct system operation:

Press Keys	See Printed	Step #
	KIM	
	xxxx xx	1
0 0 0 2	0002	2
SPACE	0002 xx	3
1 8 •	18.	4
	0003 xx	5
A 5 •	A5.	6
	0004 xx	7
LF	0003 A5	8
RUB OUT	KIM	
	xxxx xx	9

Step 1 shows the "KIM" prompting message. In Step 2, an address (0002) is selected followed by a space key in Step 3. The address cell 0002 together with the data stored at that location (xx) is printed. Step 4 shows the "modify cell" operation using the • key and the hex data keys preceding. Step 5 shows the incrementing to the next address cell (0003) after the • key. Note that the modification of cell 0002 also occurs. Steps 6 and 7 show the modification of data in cell 0003 and the incrementing to cell 0004. Step 8 shows the action of the LF key in backing up one cell to 0003 where we can see from the printout that the correct data (A5) has been stored at that location. Step 9 shows the reaction to the RUB OUT key in resetting the system and producing a new "KIM" prompting message. Note, by the way, that in this example you have repeated a portion of the program entry exactly as you did in Section 2.4 but this time using the TTY.

So much for now! If all of the operations have occurred properly, you may be certain that your TTY and KIM-1 module are working together correctly. We will describe in detail all of the other operations possible with the TTY in a later section of the manual.

If you have reached this point without problems, you now have completed all of the required system tests and may be confident that the KIM-1 module and your peripheral units are all working correctly. Our next task is to learn more about the KIM-1 system and its operating programs.

CHAPTER 3

THE KIM-1 SYSTEM

Up to this point you have been engaged in bringing up your KIM-1 system and verifying its correct operation. Now it's time to learn more about the various parts of the KIM-1, how the parts work together as a system, and how the operating programs control the various activities of the system. The diagrams included in this section together with your full sized system schematic will be helpful in understanding the elements of your KIM-1 module.

3.1 KIM-1 SYSTEM DESCRIPTION

Figure 3-1 shows a complete block diagram of the KIM-1 system. You should note first the presence of the MCS 6502 Microprocessor Array which acts as the central control element for the system. This unit is an 8 bit microprocessor which communicates with other system elements on three separate buses. First, a 16 bit address bus permits the 6502 to address directly up to 65,536 memory locations in the system. Next, an 8 bit, bidirectional data bus carries data from the 6502 array to any memory location or from any memory location back to the 6502 array. Lastly, a control bus carries various timing and control signals between the 6502 array and other system elements.

Associated with the 6502 array is a 1 MHz crystal which operates with an oscillator circuit contained on the 6502 array. This crystal controlled oscillator is the basic timing source from which all other system timing signals are derived. In particular, the $\emptyset_2$ signal generated by the 6502 array and used either alone, or gated with other control signals, is used as the system time base by all other system elements.

The 6502 microprocessor is structured to work in conjunction with various types of memory. In the KIM-1 system, all memory may be considered to be of the Read-only (ROM) or Read/Write (RAM) variety. The ROM portion of the memory provides permanent storage for the operating progams essential to the control of the KIM-1 system. You will note the inclusion of two devices, labelled 6530-002 and 6530-003. Each of these devices include a 1024 byte (8 bits per byte) ROM with different portions of the operating program stored permanently in each ROM.

RAM type memory is available at three locations in the system. Again, each of the 6530 arrays include 64 bytes of RAM primarily used for temporary data storage in support of the operating program. In addition, a separate 1024 byte RAM is included in the KIM-1 system and provides memory storage for user defined application programs and data.

Input/output controls for the system also are included within the 6530 arrays. Each 6530 array provides 15 I/O pins with the microprocessor and operating program defining whether each pin is an input pin or output pin, what data is to appear on the output pins, and reading the data appearing on input pins. The I/O pins provided on the 6530-002 are dedicated to interfacing with specific elements of the KIM-1 system including the keyboard, display, TTY interface circuit, and audio tape interface circuit. The 15 I/O pins on the 6530-003 are brought to a connector and are available for the user to control a specific application.

Finally, each 6530 array includes an interval timer capable of counting a specific number of system clocks to generate precise timing gates. The exact time interval is preset under program control. The interval timer on the 6530-003 array is available for a user defined application program and is not required by the operating programs.

Figure 3-1 shows a major block labelled Control Logic. Included under this category are an address decoder used for generation of chip select signals for the 6530 arrays and the static RAM. Also included is the logic required to debounce the keys for system reset (RS key) and program stop (ST key). Lastly, special logic is included to allow operation of the system in a "single instruction" mode to facilitate program debugging.

Figure 3-1 shows the keyboard/display logic interfacing with the I/O pins of the 6530-002. Also shown are the interface circuits for transmission of data to and reception of data from the TTY and audio tape units.

Figure 3-2 shows the detailed interconnections between the MCS 6502 and the two MCS 6530 arrays.

Figure 3-3 shows detailed logic and schematics for the control logic.

Figure 3-4 shows a detailed schematic of the static RAM.

Figure 3-5 and 3-6 show the detailed schematic of the keyboard and display logic and circuits.

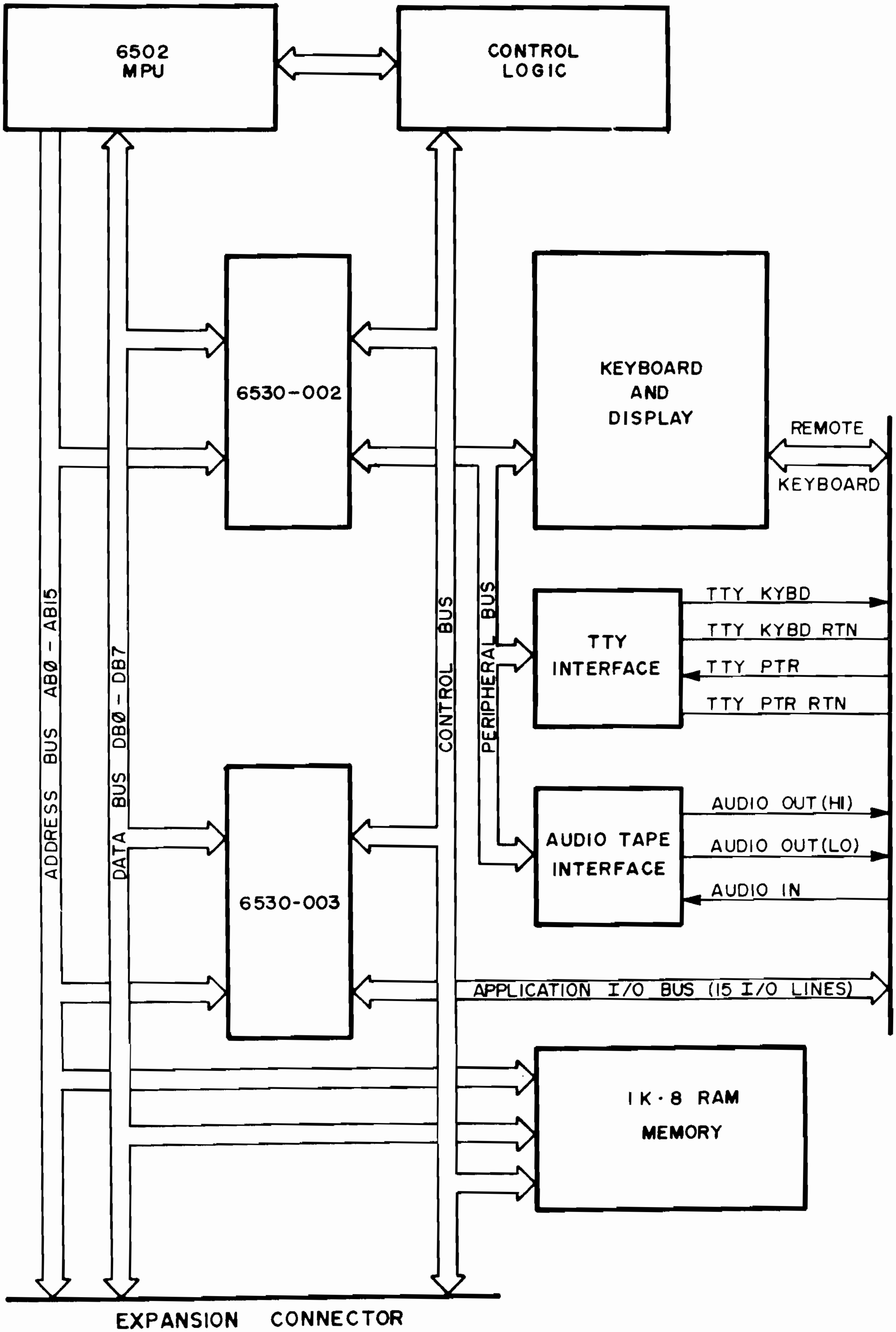
Figure 3-7 details the schematic of the TTY interface circuits.

Figure 3-8 details the schematic of the audio tape cassette interface circuits.

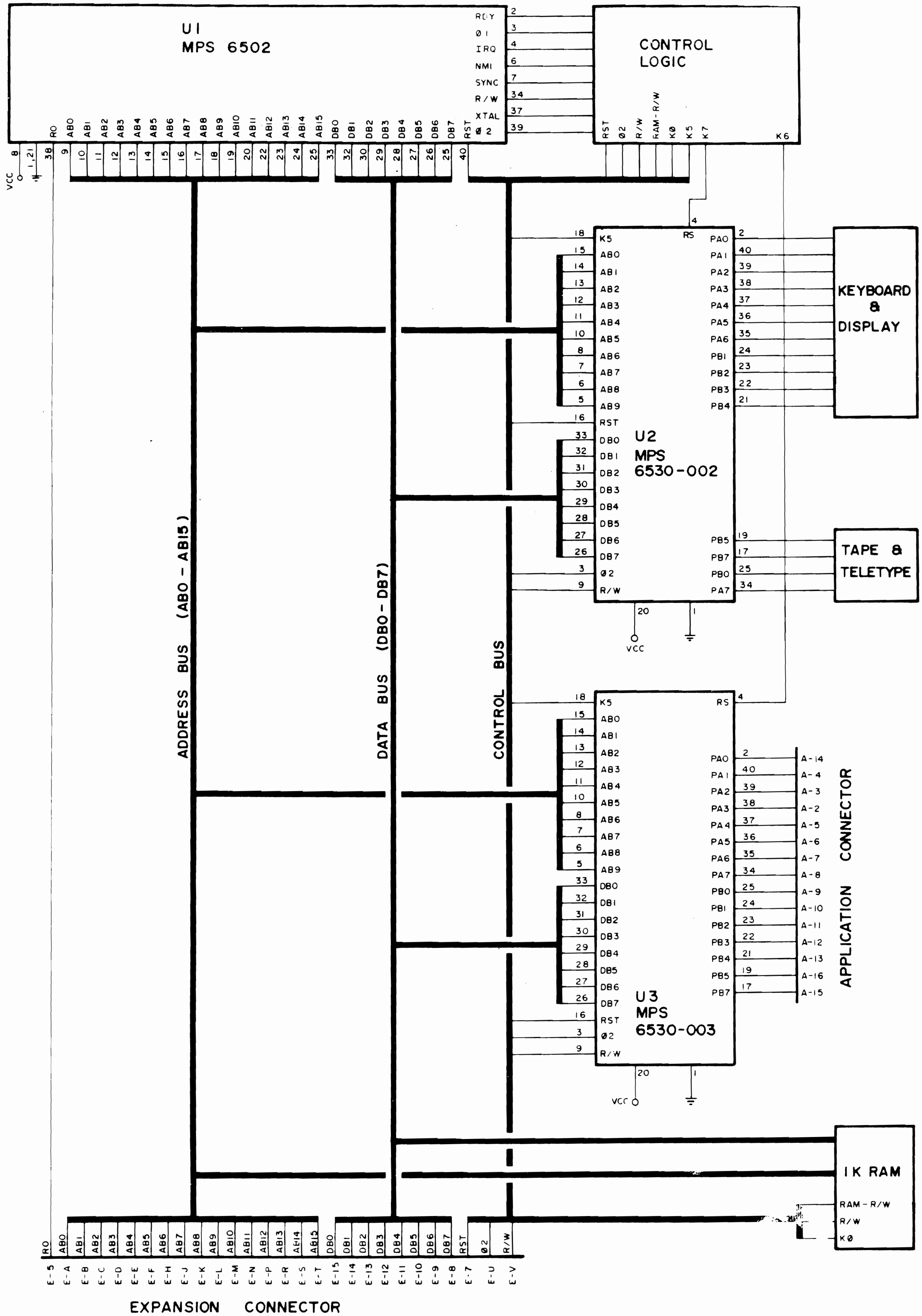
Figures 3-9 and 3-10 provide a summary of all signals available on either the Application connector or the Expansion Connector.

The fold-out system schematic shows all of the elements of the system connected together and all signals appearing on the module connectors.

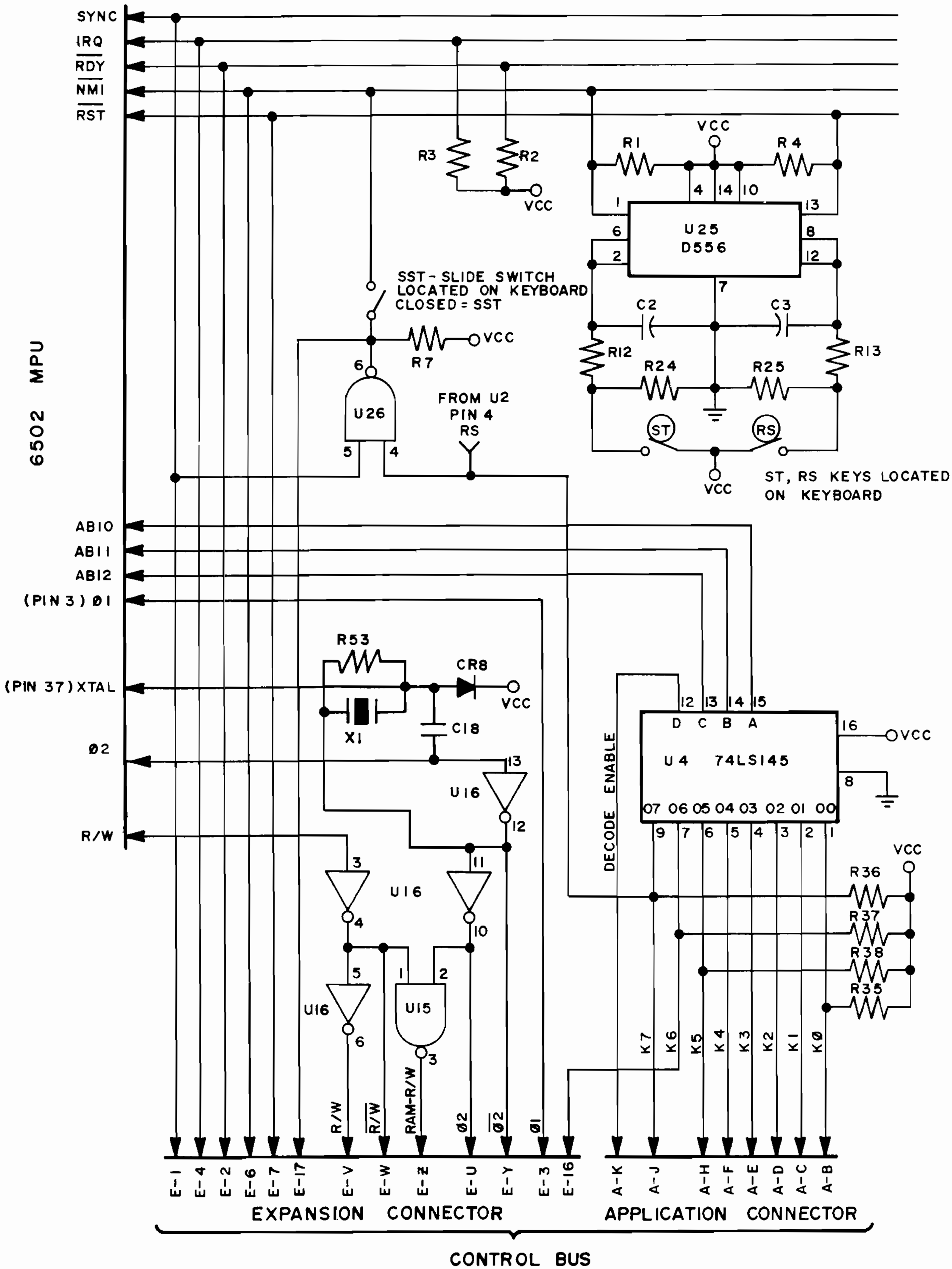
You may refer to the Hardware Manual included with your KIM-1 module for additional details on the operating characteristics of the 6502 and 6530 arrays as well as detailed information on system timing.



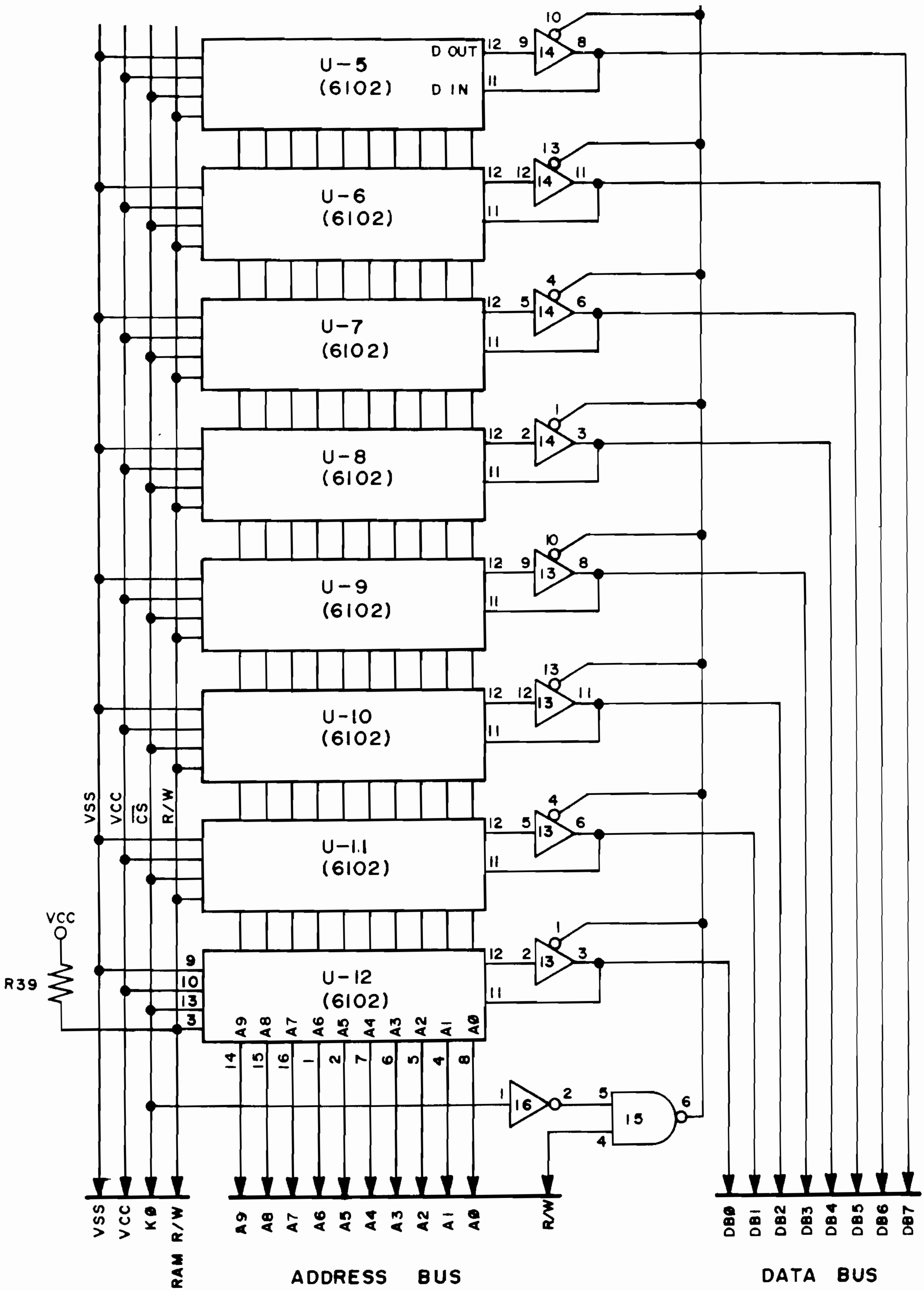
KIM-1 Block Diagram
FIGURE 3.1



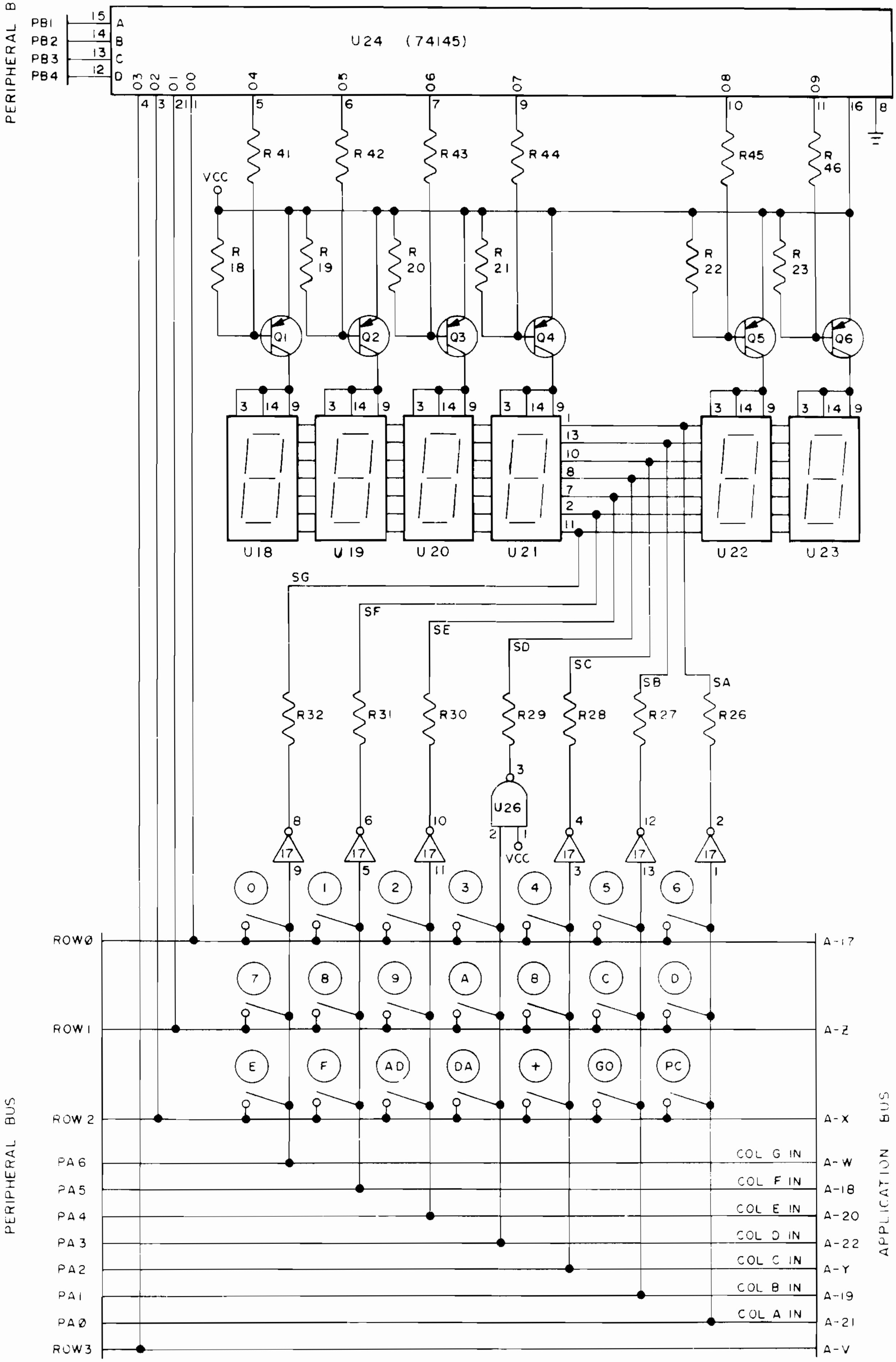
Detailed Block Diagram
FIGURE 3.2



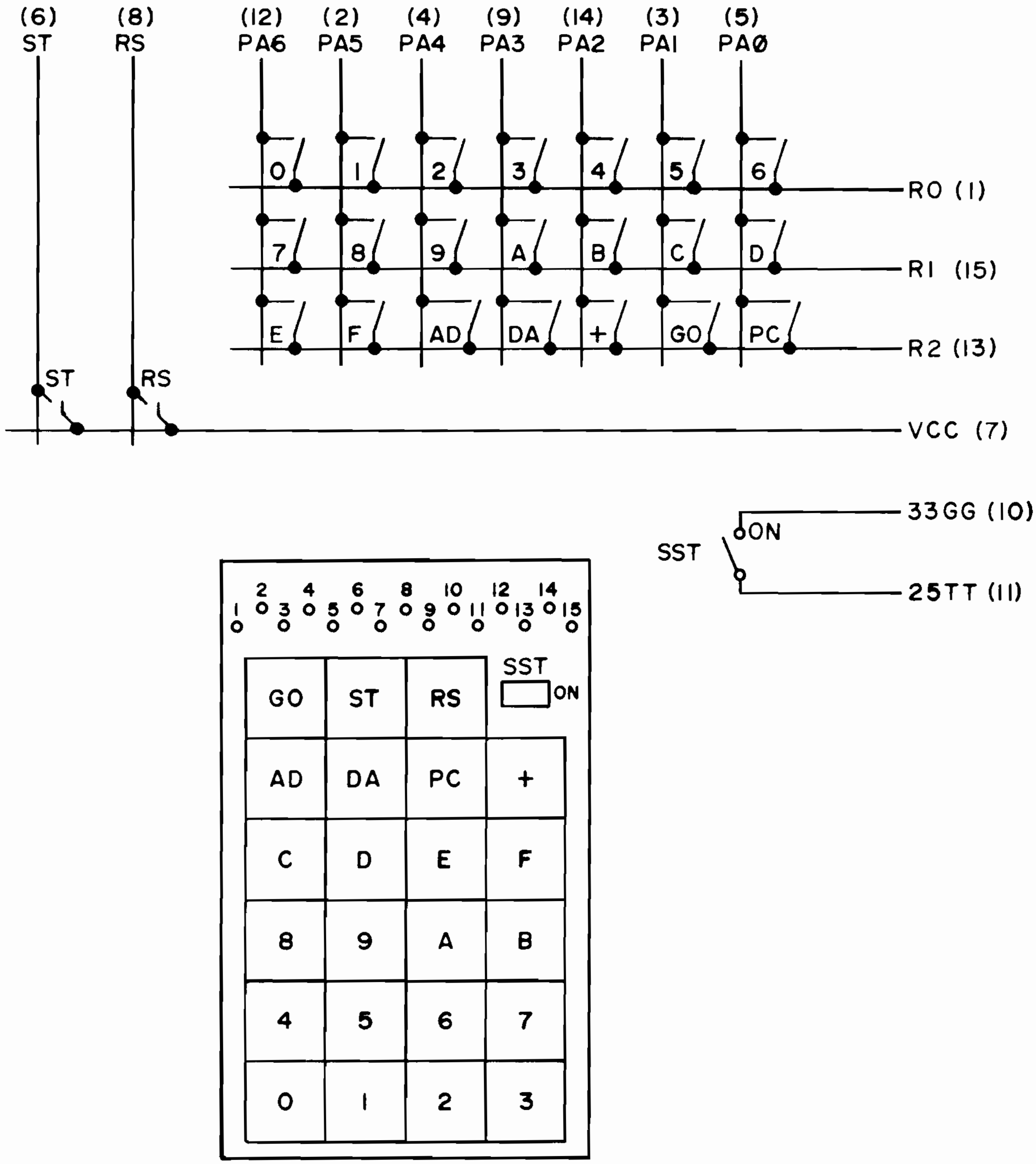
Control and Timing
FIGURE 3.3



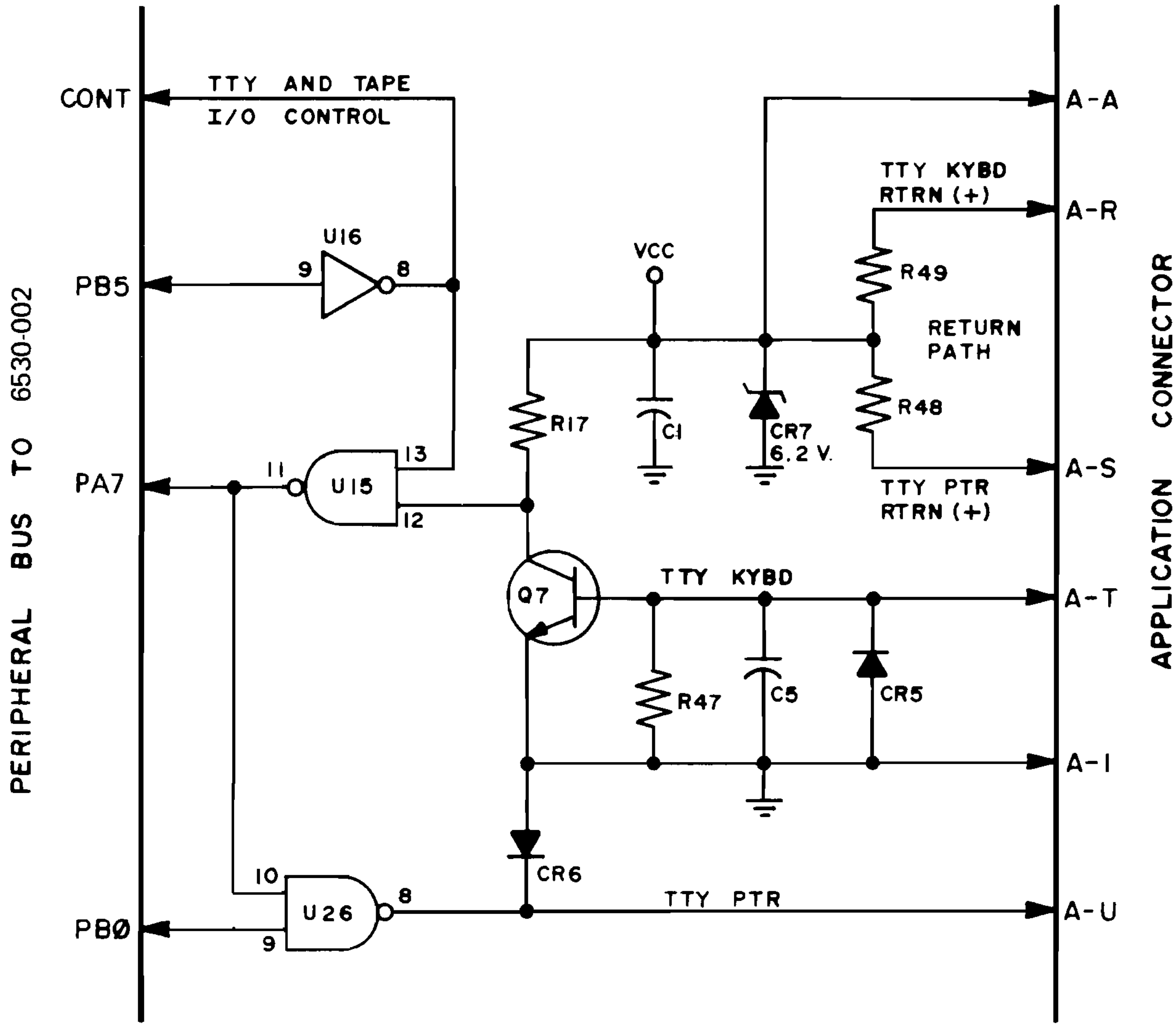
1Kx8 RAM Memory
FIGURE 3.4



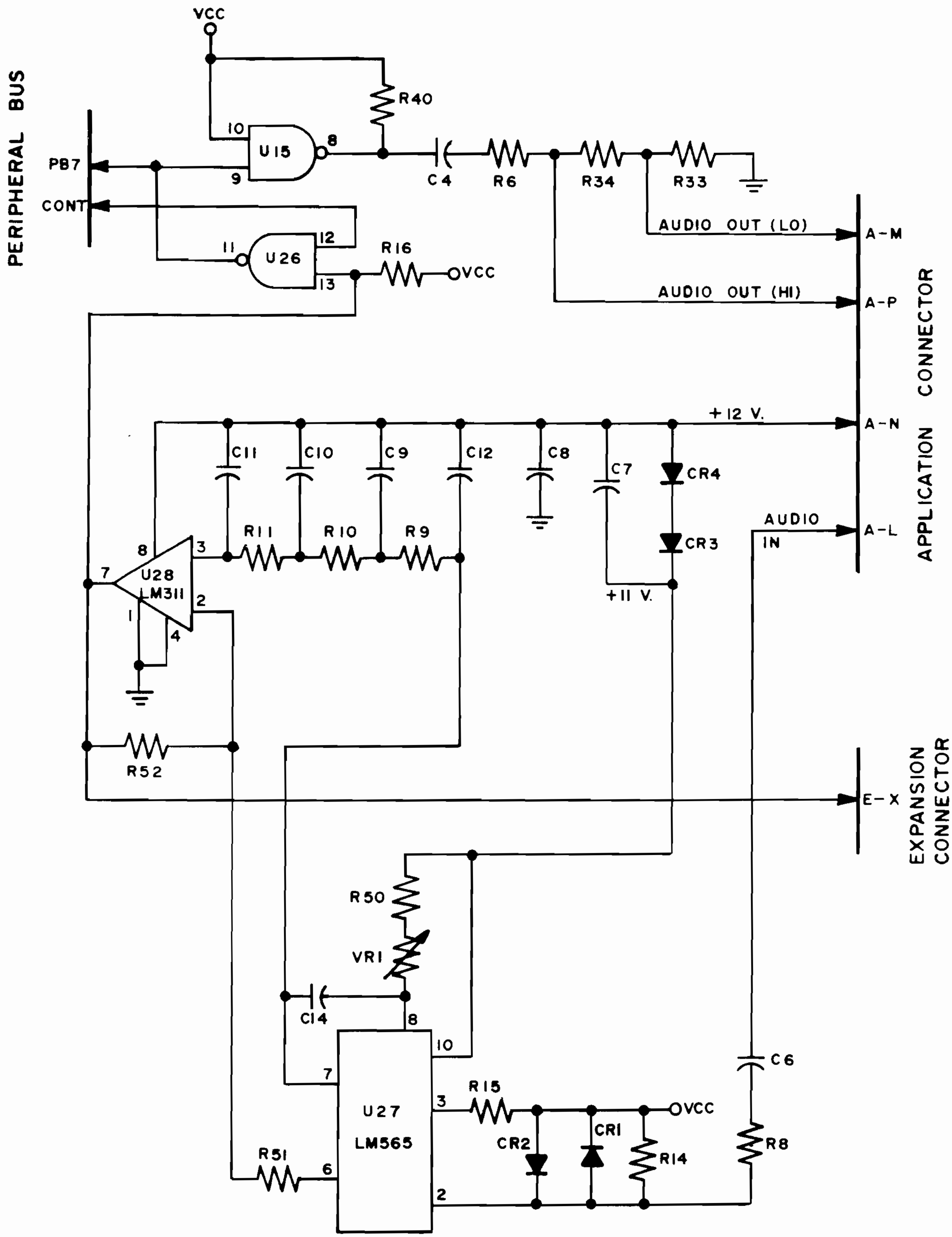
Keyboard and Display
FIGURE 3.5



Keyboard Detail
FIGURE 3.6



TTY Interface
FIGURE 3.7



Audio Tape Interface
FIGURE 3.8

22	KB Col D
21	KB Col A
20	KB Col E
19	KB Col B
18	KB Col F
17	KB Row Ø
16	PB5
15	PB7
14	PAØ
13	PB4
12	PB3
11	PB2
10	PB1
9	PBØ
8	PA7
7	PA6
6	PA5
5	PA4
4	PA1
3	PA2
2	PA3
1	VSS GND

Z	KB Row 1
Y	KB Col C
X	KB Row 2
W	KB Col G
V	KB Row 3
U	TTY PTR
T	TTY KYBD
S	TTY PTR RTRN(+)
R	TTY KYBD RTRN(+)
P	AUDIO OUT HI
N	+12v
M	AUDIO OUT LO
L	AUDIO IN
K	DECODE ENAB
J	K7
H	K5
F	K4
E	K3
D	K2
C	K1
B	KØ
A	VCC +5v

Application Connector
FIGURE 3.9

22	VSS GND
21	VCC +5
20	
19	
18	
17	SST OUT
16	K6
15	DBØ
14	DB1
13	DB2
12	DB3
11	DB4
10	DB5
9	DB6
8	DB7
7	RST
6	NMI
5	RO
4	IRQ
3	Ø1
2	RDY
1	SYNC

Z	RAM/R/W
Y	$\overline{Ø2}$
X	PLL TEST
W	$\overline{R/W}$
V	R/W
U	Ø2
T	AB15
S	AB14
R	AB13
P	AB12
N	AB11
M	AB10
L	AB9
K	AB8
J	AB7
H	AB6
F	AB5
E	AB4
D	AB3
C	AB2
B	AB1
A	ABØ

Expansion Connector
FIGURE 3.10

3.2 KIM-1 MEMORY ALLOCATION

It has been stated that the 6502 microprocessor array included in the KIM-1 system is capable of addressing any of 65,536 memory locations. Obviously, we have not included that much memory in your KIM-1 system and this section is intended to detail for you exactly what memory locations are included in the system and where they are located (their exact addresses).

Each byte of memory in the system is understood to include 8 bits. Also, you should note that any addressable location in the system may be performing any one of four functions:

1. A ROM byte - read-only memory in which we have stored the operating program.

2. A RAM byte - read/write memory for storage of variable data.

3. An I/O location - these locations include both direction registers which define the I/O pins to be either input pins or output pins, and the actual data buffer locations containing the data to be transmitted on output pins or the data read from input pins. Any I/O location may be viewed as a read/write memory location with a specific address.

4. An Interval Timer location - a series of addresses are reserved for each interval timer in the system. Again, you may write to the timer to define its counting period or read from the timer to determine its exact state.

Figure 3-11 shows a block diagram detailing all memory blocks in the KIM-1 system. Figure 3-12 provides a memory map showing all addressable locations included in the system and their relationship to each other. Note also the areas in the memory map indicated as available for expansion. (Section 6 of the manual provides more detail on the subject of memory expansion). Finally, Figure 3-13 provides a complete listing of all important memory locations and will be referenced frequently by you when writing your application programs.

Referring to Figure 3-12, note that the memory map shows a block of 8192 address locations all existing in the lowest address space within the possible 65,536 address locations. This address space is further divided into eight blocks of 1024 locations each. Each 1024 block is further divided into four pages of 256 locations each. The "K" reference defines a specific block of 1024 locations and refers to the "K" number of the address decoder included within the system control logic. The "page" reference defines a specific group of 256 addresses. A total of 32 pages (0 to 31) are included in the 8192 address locations. The hex codes for certain addresses are shown at strategic locations in the memory map.

Beginning from the highest address location of the 8192, note that the first 1024 block (K7) is assigned to the ROM of the 6530-002 and the second 1024 block (K6) is assigned to the ROM of the 6530-003. The entire operating program of the KIM-1 system is included in these two blocks.

Next in order, a portion of the K5 block is dedicated to the RAM, I/O, and Timer locations of the two 6530 arrays. An expanded view of this address space is shown in Figure 3-12. Note that the RAM addresses for the 6530-002 (Hex 17EC to 17FF) are reserved for use by the operating program and should not appear in a user generated application program. The same is true for the I/O and Timer locations of the 6530-002 which also are reserved for use by the operating programs.

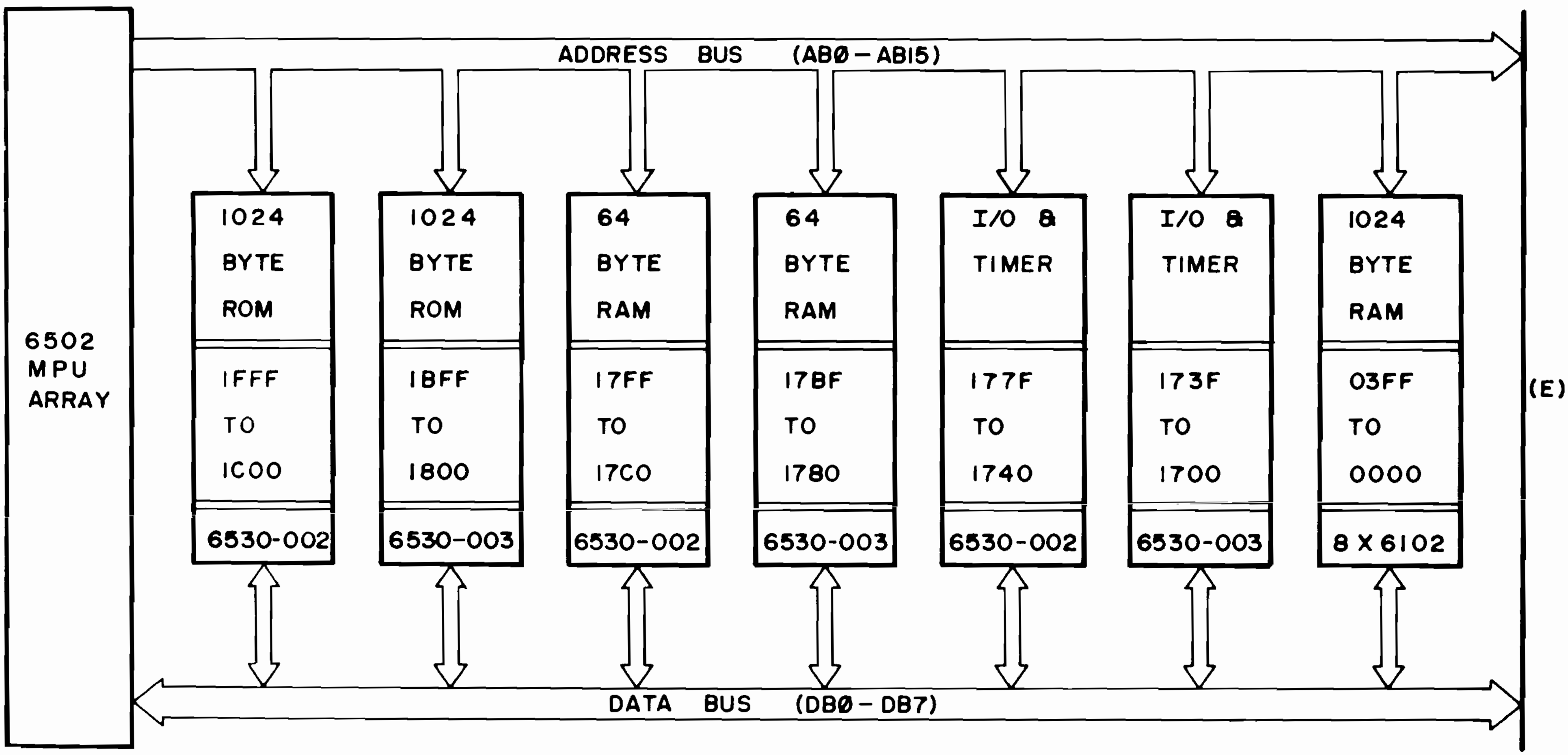
The next four blocks in order (K4, K3, K2, K1) are reserved for additional memory in an expanded system. In Section 6, the methods for adding memory will be discussed.

Finally, the lowest 1024 address locations (K0) are assigned to the static RAM included within the KIM-1 system. You should note that within this block, Page 0 and Page 1 have special significance. Page 1 is used as the system stack onto which return addresses and machine status words are pushed as the system responds to interrupts and subroutine commands. Page 0 has significance for certain of the special addressing modes available when programming for the 6502 microprocessor array.

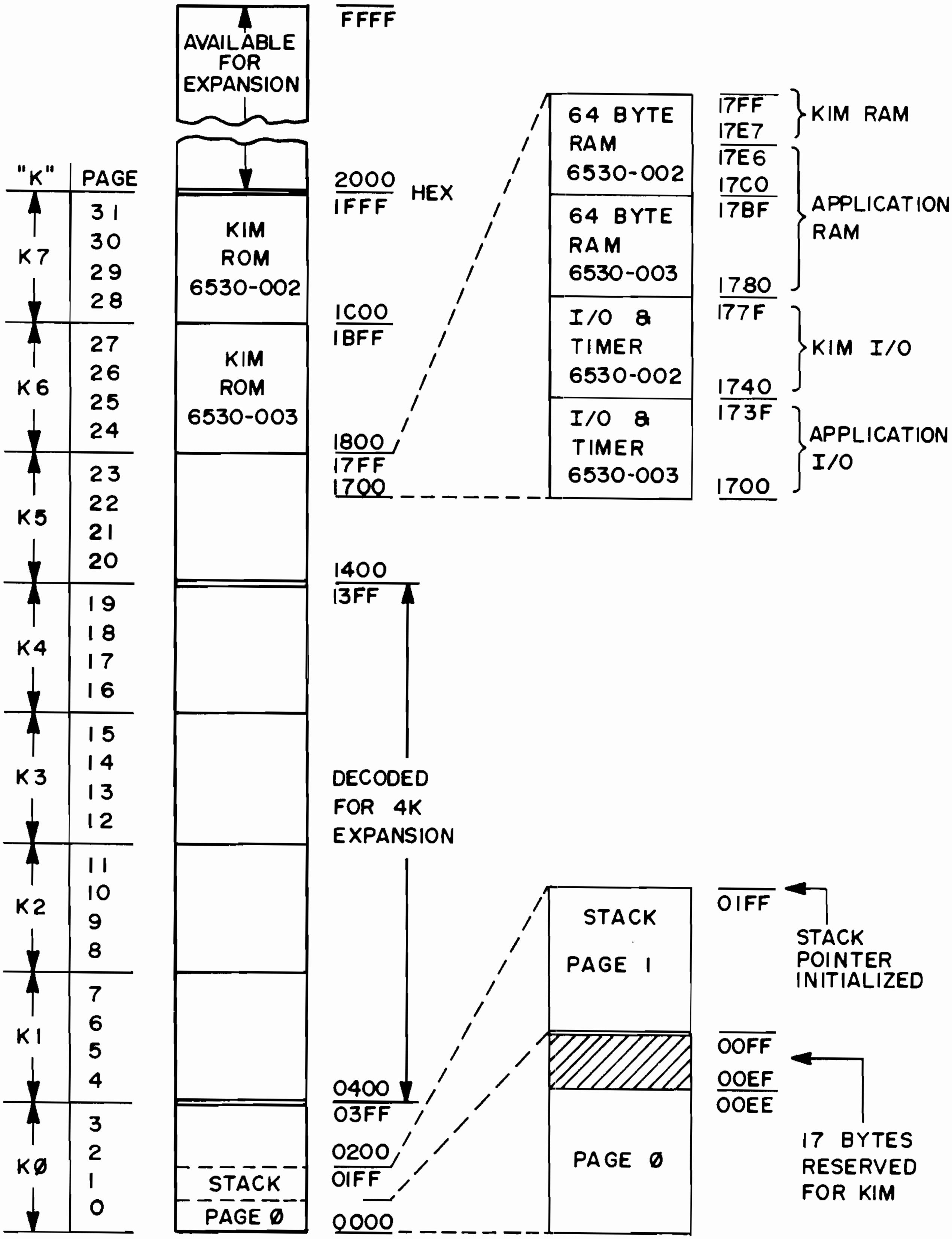
Figure 3-12 shows an expanded view of Page 0 and Page 1. Note that 17 addresses (00EF to 00FF) are reserved for use by the operating program and must never appear in the user generated application program. Also, note the comment that a maximum of eight locations may be required on the stack (Page 1) to service operating program interrupts.

In summary, the user generated application program may make use of the following areas of memory:

1. All of Page 0 except 00EF to 00FF
2. All of Page 1 (remember that the stack will extend an extra 8 bytes deep to accommodate the operating program).
3. All of Page 2 and Page 3.
4. In Page 23:
   - All I/O locations from 1700 to 173F
   - All 64 bytes of RAM from 1780 to 17BF
   - An additional 44 bytes of RAM from 17C0 to 17EB



Memory Block Diagram
FIGURE 3.11



Memory Map
FIGURE 3.12

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 Section 1.6 of Hardware Manual)
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

Special Memory Addresses
FIGURE 3.13

3.3 KIM-1 OPERATING PROGRAMS

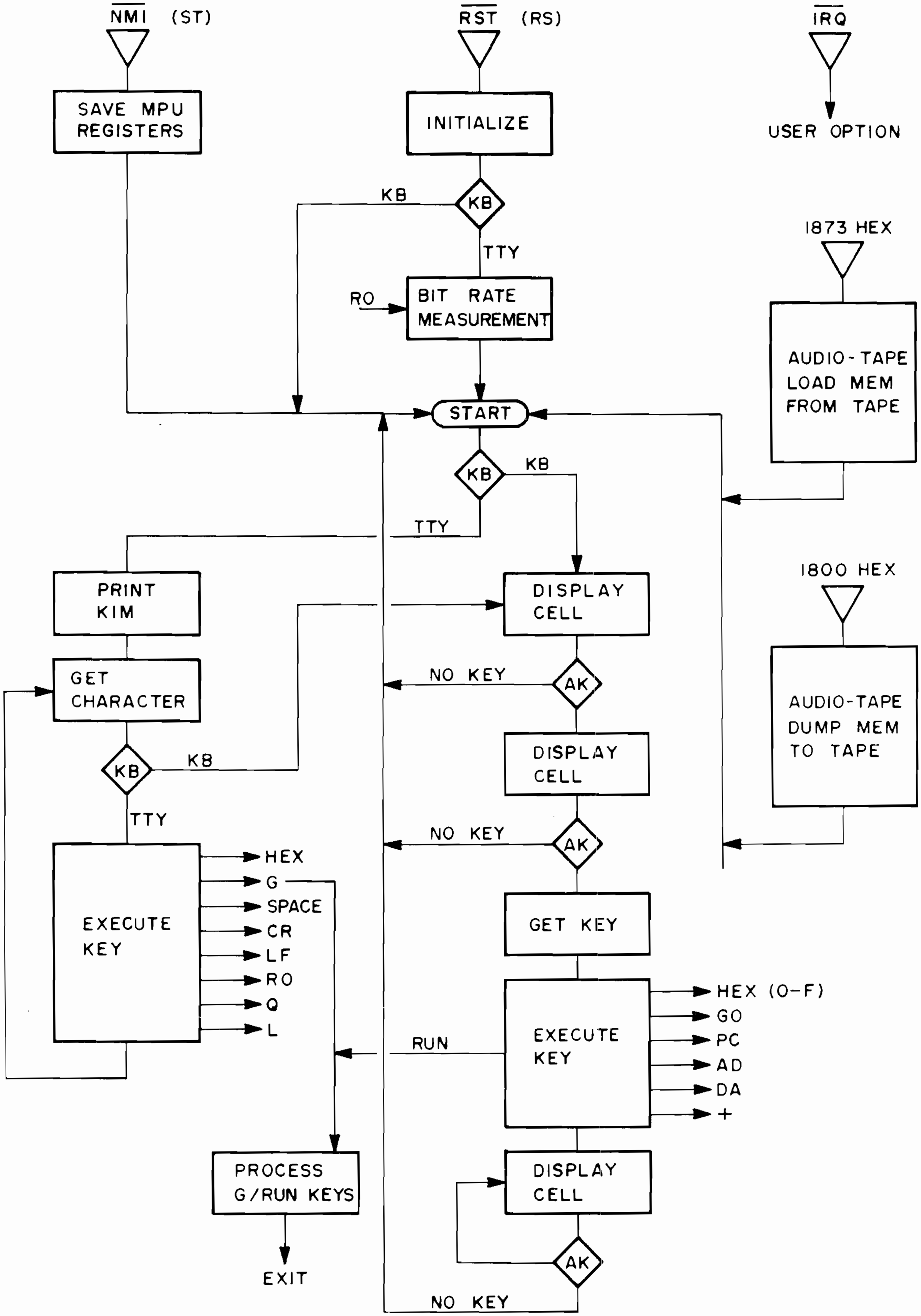
Figure 3-14 shows a simplified flow chart of the KIM-1 operating programs. This section provides a brief explanation of these programs to assist you in understanding the various operating modes of the system.

First, you should note that when power is first applied to your KIM-1 module and the RS (reset) key is depressed, control of the system automatically is assumed by the operating program. This is true, as well, for any succeeding depression of the reset key.

For each depression of the reset key, the system is initialized. At this time, stack pointer values are set, the I/O configuration is established, and essential status flags are conditioned. Next the program determines whether the system is to respond to TTY inputs or is to operate with the keyboard and display on the KIM-1 module.

If the TTY mode has been selected, the program halts and awaits a first key depression from the TTY (the RubOut Key). Upon receipt of this key depression, the program automatically performs a bit rate measurement and stores the correct value for use in receiving and decoding succeeding data transfers from the TTY. Note that this bit rate measurement is performed after each depression of the reset key.

The program will proceed immediately to a routine causing the prompting message ("KIM") to be typed on the TTY. Now, the program halts at the loop called "Get Character". As each key is depressed on the TTY, the coded data is accepted and analyzed in the routine called "Execute Key". The various keys depressed will cause the program to branch to the appropriate subroutines required to perform the desired operation. Upon completion of the individual key executions, the program returns to the "Get Key" loop and awaits the next key depression.



Flow Chart
FIGURE 3.14

Exit from the TTY processing loop will occur in response to:

1. A depression of the reset key,

2. A depression of the G key which initiates execution of the application program, or

3. A change in the mode from TTY to Keyboard/Display.

If, after system reset and initialization, the Keyboard/Display mode (KB) is determined to be in effect, the program will proceed directly to display, and keyboard scan routines. The program will cause the display scan to occur continuously ("Display Cell") until one of the keys on the keyboard is depressed (AK?). Key validation is performed during an additional scan cycle. If the key is truly depressed (not noise), the program proceeds to the routine called "Get Key" in which the exact key depressed is defined. Next, the program moves to the "Execute Key" routine where branches to appropriate execution routines will be performed. Finally, after key execution, the program returns to the "Display Cell" routine and waits for the key to be released. When no key is depressed, the program returns to the normal "Display Cell" routine and awaits the next key depression.

In either the TTY or KB modes, the audio tape load or dump routines may be executed using appropriate commands from the selected keyboards. In either case, completion of the tape load or dump routine allows the program to return to the "Start" position which will, as usual, activate the KIM-1 display or cause the "KIM" prompting message on the TTY.

You should note the use of the Stop key to activate the non-maskable interrupt input ($\overline{\text{NMI}}$) of the 6502 microprocessor array. Depression of this key causes an unconditional termination of program execution, a saving of machine status registers on the stack, and a return to the control of the operating program.

A second interrupt input is available and referred to as $\overline{\text{IRQ}}$. This interrupt may be defined by the user and will cause the program to jump to any location defined by the user in his program.

CHAPTER 4

OPERATING THE KIM-1 SYSTEM

Now that you have a better idea of what is included in your KIM-1 system and how it operates, its time to provide you with detailed procedures for all of the operations you can perform with the system. We will separate our operating procedures into three areas giving specific direction for the use of the KIM-1 keyboard and display, the audio tape recorder, and the serial teleprinter (TTY).

4.1 USING THE KIM-1 KEYBOARD AND DISPLAY

A brief study of your keyboard shows a total of 23 keys and one slide switch. First, let's list the purpose of each key:

[0] TO [F]	- Sixteen keys used to define the hex code of address or data
[AD]	- selects the address entry mode
[DA]	- selects the data entry mode
[+]	- increments the address by +1 but does not change the entry mode
[PC]	- recalls the address stored in the Program Counter locations (PCH, PCL) to the display
[RS]	- causes a total system reset and a return to the control of the operating program
[GO]	- causes program execution to begin starting at the address shown on the display
[ST]	- terminates the execution of a program and causes a return to the control of the operating program

You have seen in an earlier chapter that the six digit display includes a four digit display of an address (left four digits) and a two digit display of data (right two digits).

Using only the KIM-1 keyboard and display, you may perform any of the following operations:

1. Select an Address

   Press $\boxed{\text{AD}}$ followed by any four of the hex entry keys. The address selected will appear on the display. If an entry error is made, just continue to enter the correct hex keys until the desired address shows on the display. Regardless of what address is selected, the data field of the display will show the data stored at that address.

2. Modify Data

   After selecting the proper address, press $\boxed{\text{DA}}$ followed by two hex entry keys which correctly define the data to be stored at the selected address. The data entered will appear in the data field of the display to indicate that the desired code has already been entered.

   Note that it is possible for you to select an address of a ROM memory cell or even the address of a memory cell that does not exist in your system. In these cases, you will not be able to change the data display since it is clearly not possible for the system to write data to a ROM cell or a non-existent memory location.

3. Increment the Address

   By pressing the $\boxed{+}$ key the address displayed is automatically increased by +1. Of course, the data stored at the new address will appear on the display. This operation is useful when a number of successive address locations must be read or modified. Note that the use of the $\boxed{+}$ key will not change the entry mode. If you had previously pressed the $\boxed{\text{AD}}$ key, you remain in the address entry mode and a previous depression of the $\boxed{\text{DA}}$ means you remain in the data entry mode.

4. Recall Program Counter

Whenever the $\overline{NMI}$ interrupt pin of the 6502 microprocessor array is activated, the program execution in progress will halt and the internal registers of the 6502 are saved in special memory locations before the control of the system is returned to the operating program. In the KIM-1 system, the $\overline{NMI}$ interrupt may occur in response to a depression of the [ST] key (stop) or, when operating in the Single Step mode, after each program instruction is executed following the depression of the [GO] key.

The [PC] key allows you automatically to recall the value of the Program Counter at the time an interrupt occurred. You may have performed a variety of operations since the interrupt such as inspecting the contents of various machine registers stored at specific memory locations. However, when you press the [PC] key, the contents of the Program Counter at the time of the interrupt are recalled to the address field of the display. You now may continue program execution from that point by pressing the [GO] key.

5. Execute a Program

Select the starting address of the desired program. Now, press the [GO] key and program execution will commence starting with the address appearing on the display.

6. Terminate a Program

The [ST] key is provided to allow termination of program execution. As mentioned earlier, the [ST] key activates the $\overline{NMI}$ interrupt input of the 6502 microprocessor array.

Note: The [ST] key will operate correctly only if you store the correct interrupt vector at locations 17FA and 17FB. For most of your work with the KIM-1 system, you should store the address 1C00 in these locations as follows:

Now, when the $\overline{NMI}$ interrupt occurs, the program will return to location 1C00 and will proceed to save all machine registers before returning control to the operating program.

You should remember to define the $\overline{NMI}$ vector each time the power to the system has been interrupted. A failure of the system to react to the $\boxed{ST}$ key means you have forgotten to define the $\overline{NMI}$ vector.

7. Single Step Program Execution

   In the process of debugging a new program, you will find the single step execution mode helpful. To operate in this mode, move the SST slide switch to the ON position (to your right). Now, depress the $\boxed{GO}$ key for each desired execution of a program step. The display will show the address and data for the next instruction to be executed. Note that in the course of stepping through a program, certain addresses will appear to be skipped. A program instruction will occupy one, two, or three bytes of memory depending upon the type of instruction. In single instruction mode, all of the bytes involved in the execution of the instruction are accessed and the program will halt only on the first byte of each successive instruction.

   Note: SST mode also makes use of the $\overline{NMI}$ interrupt of the 6502 microprocessor array. Again, the NMI vector must be defined as described in (6) above if the SST mode is to work correctly.

This covers all of the standard operations you may perform from the KIM-1 keyboard. Using combinations of the operations described, you may wish to perform certain specialized tasks as follows:

1. Define the $\overline{IRQ}$ Vector

   You will recall that a separate interrupt input labelled IRQ is available as an input to the 6502 microprocessor array. If you wish to use this feature, you should enter the address to which the program will jump. The $\overline{IRQ}$ vector is stored in locations 17FE and 17FF.

2. Interrogate Machine Status

   We have mentioned that after an $\overline{NMI}$ interrupt in response to the $\boxed{ST}$ key or during the SST mode, the contents of various machine registers are stored in specific memory locations. If you wish to inspect these locations, their addresses are:

```
00EF = PCL
00F0 = PCH
00F1 = Status Register (P)
00F2 = Stack Pointer (SP)
00F3 = Accumulator (A)
00F4 = Y Index Register
00F5 = X Index Register
```

4.2 USING THE AUDIO TAPE RECORDER

There are two basic operations possible when working with your audio tape system. You may transfer data from the KIM-1 memory and record it on tape. Or, you may read back a previously recorded tape, transferring the data on tape into the KIM-1 memory.

<u>Recording on Audio Tape</u>

The procedure for recording on audio tape requires that you perform the following steps:

1. Clear decimal mode by entering 00 in location 00F1. Define an identification number (ID) for the data block you are about to record. This two digit number is loaded into address 17F9. Don't use ID = 00 or ID = FF.

2. Define the starting address of the data block to be transferred. This address is to be loaded into locations:

   17F5 = Starting Address Low (SAL)
   17F6 = Starting Address High (SAH)

3. Define the ending address as <u>one greater than</u> the last address in the data block to be recorded. The ending address is to be loaded into locations:

   17F7 = End Address Low (EAL)
   17F8 = End Address High (EAH)

As an example, assume you wish to record a data block from address 0200 up to and including address 03FF. (All of Pages 2 and 3). You wish to assign an ID number of 06 to this block. Using the KIM-1 keyboard, you should load the data shown into the addresses indicated so that:

```
00F1 = 00 (Clear Decimal Mode)
17F5 = 00 (SAL)
17F6 = 02 (SAH)
17F7 = 00 (EAL) }
17F8 = 04 (EAH) } = 03FF + 1
17F9 = 06 (ID)
```

Note that the ending address must be greater than the starting address for proper operation.

4. Assuming that you are using a new cassette on which no data has been stored previously, insert the cassette in the unit and rewind the tape to its start position.

5. Select the starting address of the tape record program. This address is 1800.

6. Select the Play/Record mode of the audio unit and allow several seconds for the tape to begin to move.

7. Press the $\boxed{\text{GO}}$ key and the recording process will begin. The display will be blanked for a period and then will relight showing 0000 xx. This means that the data block selected has been recorded.

8. You may now stop the tape or allow some additional seconds of blank tape and then stop the unit.

Loading Data From Audio Tape

The procedure for loading data from an audio tape into the KIM-1 memory requires that you perform the following steps:

1. Clear decimal mode by entering 00 in location 00F1. Define the ID number of the data block to be loaded from tape. The ID number is loaded into address 17F9.

2. Select the starting address of the Tape Load program. This address is 1873_{HEX}.

3. Press the $\boxed{\text{GO}}$ key. The KIM-1 system is now waiting for the appearance of data from the tape unit.

4. Load the cassette and, presuming you do not know where on the tape the data block is recorded, rewind the tape to its starting position. Check the volume control setting.

5. Start the audio tape unit in its Play mode and observe that the tape begins to move.

6. Wait for the KIM-1 display to relight showing 0000 xx. This means the data block has been loaded successfully from the tape into the KIM-1 memory. If the display relights with FFFF xx, the correct data block has been found but there has been an error detected during the read operation. If the tape continues to run and the display never relights, the system has not been successful in finding the data block with the specific ID number you requested.

7. If in step (1), you had selected an ID = 00, 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 address as specified on the tape.

8. If, in step (1), you had selected an 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 (SAL, SAH) instead of the locations specified on the tape.

Special Operations with Audio Tape

The KIM-1 system causes data to be recorded on audio tape with a specific format as detailed in Appendix E. Each recorded data block is preceeded by a group of synchronizing characters together with an identification code to define the specific block. Data blocks may be of arbitrary length.

With a little care, there is no reason for you not to include a number of recorded data blocks on the same tape. If you are recording blocks in sequence and have not rewound the tape between blocks, you need only specify the parameters of each new block (ID, SAL, SAH, EAH, EAL) and proceed with recording the new block.

If the tape has been rewound, you will need to know the ID number of the last recorded data block. Rewind the tape to its starting point and set up the parameters required to read the last recorded data block. After reading this block, stop the tape and you may now proceed to add a new block or blocks to the tape.

If you wish, you may add voice messages between the recorded data blocks on the tape. The KIM-1 system will ignore these audio messages when the tape is read back. Of course, you will need to install an earphone or speaker in parallel with the KIM-1 audio tape data input pin in order to hear the voice messages.

We do not recommend that you attempt to record data blocks in areas of the tape which have been used previously for recorded data. Variations in tape speed and block lengths can result in overlapping of recorded data which may be read incorrectly by the KIM-1 system.

4.3 USING A SERIAL TELEPRINTER

The addition of a serial teleprinter (such as the Teletype Model 33ASR) to work with the KIM-1 system permits a variety of special operations to be performed. In all cases, you define desired operations by depressing the proper keys while simultaneously producing a hard-copy printed record of each operation. If your teleprinter is equipped with a paper tape reader/punch, you may generate or read paper tapes using the KIM-1 system. Using the serial teleprinter, you may perform the following operations:

Select an Address

Type four hex keys (0 to F) to define the desired address. Next, press the SPACE bar.

The printer will respond showing the address code selected followed by a two digit hex code for data stored at the selected address location:

Type:	1234	SPACE
Printer Responds:	1234	AF

showing that the data AF is stored at location 1234.

Modify Data

Select an address as in the previous section. Now type two hex characters to define the data to be stored at that address. Next type the ⊙ key to authorize the modification of data at the selected address:

Type:	1234	SPACE	
Printer Responds:	1234	AF	
Type:			6D ⊙
Printer Responds:	1235	B7	

Note that the selected address (1234) has been modified and the system increments automatically to the next address (1235).

Note: Leading zero's need not be entered for either address or data fields: For example:

EF	SPACE	selects address 00EF
E	SPACE	selects address 000E
A	⊙	enters data 0A
	⊙	enters data 00 (etc.)

Step to Next Address

Type CR to step to the next address without modifying the current address:

See Printed:	1234 AF	
Type:		CR
Printer Responds:	1235 B7	
Type:		CR
Printer Responds:	1236 C8	(etc.)

Step to Preceeding Address

Type LF to step back to the preceeding address:

See Printed:	1234 AF	
Type:		LF
Printer Responds:	1233 9D	
Type:		LF
Printer Responds:	1232 8E	(etc.)

Abort Current Operation

Type RUB OUT to terminate the current operation. The prompting message will be printed ("KIM") indicating that a new operation may proceed:

Type:	1264	RUB OUT
Printer Responds:	KIM xxxx xx	
Type:	1234	SPACE
Printer Responds:	1234 AF	

In the example, the RUB OUT key is used to correct an erroneous address selection.

Note: The RUB OUT key must be depressed after each depression of the KIM-1 reset key in order to allow the operating program to define the serial bit rate for the teleprinter.

Load Paper Tape

Paper Tapes suitable for use with the KIM-1 system are generated using the format shown in Appendix F. To read such a tape into the KIM-1 system, proceed as follows:

1. Load the punched paper tape on to the tape mechanism
2. Type (L)
3. Activate the paper tape reader

The paper tape will advance and data will be loaded into addresses as specified on the tape. A printed copy of the data read will be generated simultaneously with the reading of the paper tape.

Check-sums are generated during the reading of the paper tape and are compared to check-sums already contained on the tape. A check-sum error will cause an error message to appear in the printed copy.

Punch Paper Tape

The KIM-1 system can be used to punch paper tapes having the format described in Appendix F. The procedures for generating these tapes is as follows:

1. Define the starting address and ending address of the data block to be punched on the paper tape.

2. Load blank paper tape on the punch unit and activate the punch.

Type:		1 7 F 7 SPACE
See Printed:	17F7 xx	
Type:		F F •
See Printed:	17F8 xx	
Type:		0 3 •
See Printed:	17F9 xx	
Type:		2 0 0 SPACE
See Printed:	0200 xx	

You have now loaded the ending address (03FF) into address locations 17F7 (EAL) and 17F8 (EAH). The starting address (0200) is selected as shown.

3. Now type Q

   The paper tape will advance and punching of the data will proceed. Simultaneously, a printed record of the data will be typed.

List Program

A printed record of the contents of the KIM-1 memory may be typed. The procedure is the same as for punching paper tape except that the punch mechanism is not activated.

Execute Program

To initiate execution of a program using the TTY keyboard, the following procedures should be followed:

1. Enter the starting address of the program

2. Type G

   For example, to begin program execution from address location 0200:

Type:	2 0 0 SPACE
See Printed:	0200 xx
Type:	G

   Program execution begins from location 0200 and will continue until the ST or RS keys of the KIM-1 module are depressed. The single step feature may be employed while in the TTY mode.

CHAPTER 5

LET'S TRY A REAL APPLICATION

It is not practical in this manual to describe every possible application or programming technique. However, now that you have become familiar with the basic elements and operating procedures of the KIM-1 system, this section will show you how to apply what you have learned in a simple but realistic application example.

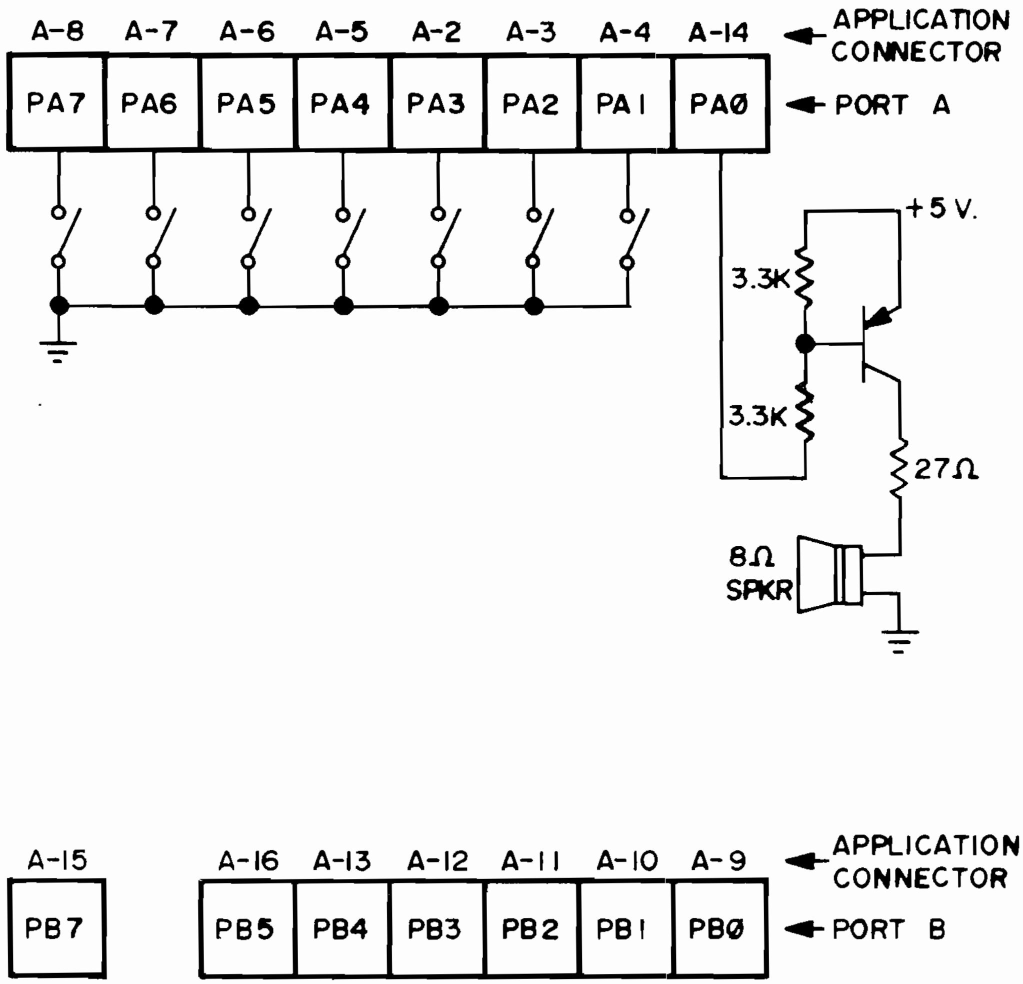
Our example will involve the generation of a variable frequency square wave which will be connected to a speaker to produce an audible tone. The frequency of the tone will be selected using a set of seven toggle switches. We will proceed through the example by defining the interface, writing and entering the program, and executing the program. Finally, we will study a series of program debugging techniques which will be useful to you for any new program you may write.

5.1 DEFINING THE INTERFACE

You will recall that a group of 15 I/O pins are brought to the Application connector from the 6530-003 array. The logic and circuit details concerning these I/O pins are described in Appendix H and in Section 1.6 of the Hardware Manual ("Peripheral Interface/Memory Device - - MCS 6530").

For our application example we will use eight of these I/O pins. One pin (PAØ) will be used as an output line to supply a square wave to a driver circuit and speaker. The other seven I/O pins (PA1 to PA7) are defined as input points with a SPST toggle switch connected to each. Figure 5-1 shows the circuit configuration for this example. Note that the remaining seven I/O pins (the PB port) are not used for this problem.

For the switches connected to the input pins, we would like the sense of the switch to be defined as a logic "0" when open and a logic "1" when closed. By connecting the switches to ground, we are producing exactly the opposite sense and must remember to complement the switch states with software when we write our program. Also, we must define now that the switch at PA1 is to be the LSB (least significant bit) and the switch at PA7 is to be the MSB (most significant bit) of the seven bit binary word formed by all seven switches. In this way, the state of the switches can define a binary number from zero (all switches open) to 127_{DEC} (all switches closed).



(THE B PORT IS NOT USED IN THIS EXAMPLE APPLICATION)

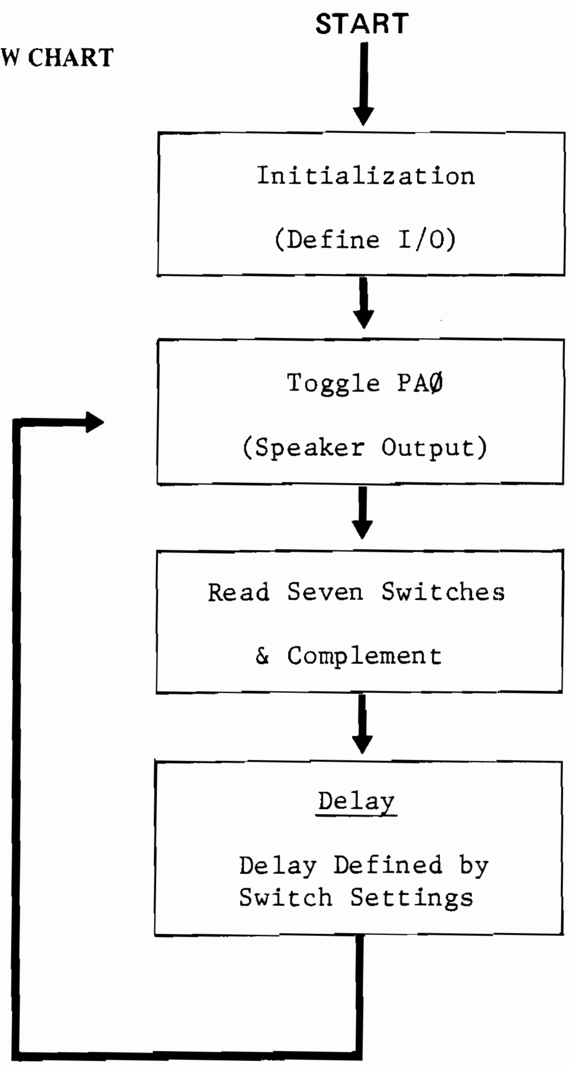
Speaker Application
FIGURE 5.1

5.2 WRITING THE PROGRAM

Having defined the interface for our application, we may proceed now to write our program. The effort proceeds in four stages:

1. Generate a flow chart
2. Generate assembly language code
3. Analyze the program
4. Generate machine language code

FLOW CHART



Briefly, our flow chart shows a first step of system initialization. During this step, we must define the I/O configuration of the system in that pin PAØ becomes the output to the speaker and that pins PA1 to PA7 become inputs from the seven switches.

After initialization, a loop is set up which begins by inverting the state of PAØ (Toggle PAØ). Next, the state of the switches is read and the data is complemented to produce the correct "sense" from the switches. The value so read is used to define a delay before returning to the start of the loop and again toggling the state of PAØ. A little thought will show that this loop will produce a square wave with a frequency determined by the setting of the seven switches.

Assembly Language Program

Our next task is to convert the simple flow chart into a program. The program is first written in "Assembly Language". You should refer to your Programming Manual to become familiar with all of the possible 6502 instructions (especially see Appendix B; Instruction Summary). Figure 5-2 shows the application example programmed in assembly language.

LABEL	OP CODE	OPERAND	MACHINE CYCLES	COMMENTS
INIT	LDA	#$01	2	Define I/O 0=Input 1=Output
	STA	PADD	4	PADD = PORT A DATA DIRECTION REG.
START	INC	PAD	6	Toggle PAØ, PA1-PA7 Inputs not affected
READ	LDA	PAD	4	READ switches into accumulator
	EOR	#$FF	2	Complement switch value
	LSR	A	2	Shift Accumulator 1 bit to right
	TAX		2	Transfer final count into X-Index
DELAY	DEX		2	Delay by an amount specified
	BPL	DELAY	3,2	By the count in the X-Index
	BMI	START	3	Go To START
PADD	=$1701			Define absolute address of Data Direction Reg. A
PAD	=$1700			Define absolute address of Data Reg. A

Assembly Language Listing
FIGURE 5.2

You will note that each line of the program is broken into several fields:

- A label field permitting you to assign a "name" to a specific location in the program.

- An Operation Code field (Op Code) in which the exact instruction to be executed is defined.

- An Operand Field where the exact data required by the instruction is defined together with certain symbols defining addressing modes or data formats. Symbols encountered generally in MOS Technology, Inc. manuals are:

#	Immediate Addressing
$	Hex Code
@	Octal Code
%	Binary Code
'	ASCII literal
=	Equates a label to a value

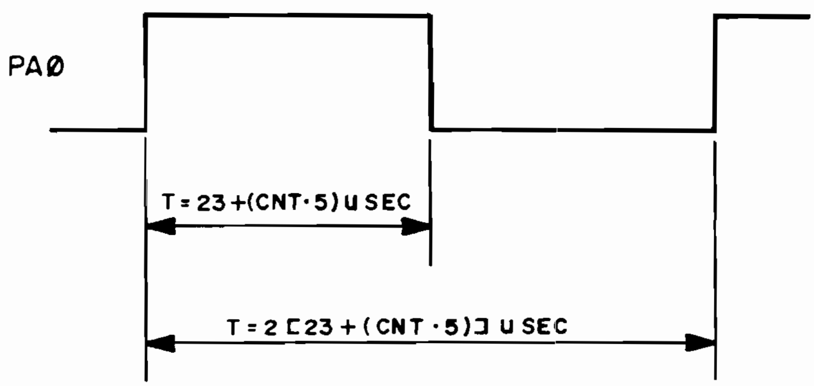
- A Machine Cycle field defining the total number of machine cycles required to execute an instruction. (This information is derived from Appendix B of the Programming Manual).

- A Comment Field where the programmer may define the intent of specific program steps.

Program Analysis

The inclusion of the "machine cycle" information of the program chart (Figure 5-2) allows us to analyze the exact timing relationships involved in our program example. Note that the KIM-1 system operates from a fixed frequency (1 MHz) oscillator with each machine cycle being 1μs. Therefore, an instruction like "INC PAD" which requires 6 machine cycles will be executed in a 6μs period.

By counting the total machine cycles occurring between each toggle of PAØ, an equation for the square wave frequency can be developed. The actual frequency is determined by the position of the seven switches, the number of machine cycles between each toggle of PAØ, and the basic clock rate (1 MHz) of the KIM-1 system. Figure 5-3 shows the waveform of the PAØ square wave and the derived equations for computing the exact frequency.



$$FREQ = \frac{1}{T} = \frac{10^6}{46 + 10 \cdot CNT} \text{ CPS}$$

NOTE: CNT EQUALS THE VALUE IN X-INDEX WHICH WAS CALCULATED FROM THE SEVEN SWITCHES $0 \leq CNT \leq 127$

Square Wave Output
FIGURE 5.3

Machine Language Coding

Our next problem is to convert our assembly language program into a program written in "machine language". The quickest and most foolproof method for accomplishing this conversion is by using the MOS Technology, Inc. Assembler (available for use on the time share services of United Computing Systems, Inc.). If you choose not to use this method, you will need to convert your source program to machine code using "paper-and-pencil" techniques.

You should proceed by constructing a table similar to that shown in Figure 5-4.

	INSTRUCTION			SOURCE CODE		
ADDRESS	**BYTE 1**	**BYTE 2**	**BYTE 3**	**LABEL**	**OP CODE**	**OPERAND**
Ø2ØØ	A9	Ø1		INIT	LDA	#$01
Ø2Ø2	8D	Ø1	17		STA	PADD
Ø2Ø5	EE	ØØ	17	START	INC	PAD
Ø2Ø8	AD	ØØ	17	READ	LDA	PAD
Ø2ØB	49	FF			EOR	#$FF
Ø2ØD	4A				LSR	A
Ø2ØE	AA				TAX	
Ø2ØF	CA			DELAY	DEX	
Ø21Ø	1Ø	FD			BPL	DELAY
Ø212	3Ø	F1			BMI	START
Ø214						

Machine Language Code Table
FIGURE 5.4

The source code contained in your assembly language program (Figure 5-2) is entered into the table first. A column is provided to allow you to define the specific address at which an instruction is located. The Instruction column provides space for defining one, two, or three byte instructions. (Please refer to Appendix B of the Programming Manual or to your Programming Card for specific Op Codes).

As an example, the first source instruction is LDA #$01 which, when translated, means load the accumulator with the byte stored in the next program location (hex 01). This is the "immediate" addressing mode defined by the "#" symbol. The Op Code for LDA# is A9. This value is entered in the first column under the heading, Instruction. The next column contains the hex 01 value defined by the source statement. The initial address for the program is inserted in the "Address" column as 0200 (an arbitrary selection). The total instruction LDA #$01 now occupies address locations 0200 and 0201.

The next available address is 0202 which is inserted in the "Address" column for the next source instruction. In this manner, you will proceed through all of the source statements decoding each and entering one, two, or three bytes of machine code as required in the "Instruction" column. The "Address" column will contain the address of the first byte of machine code (the Op Code) for each source statement.

In cases where the operand of the source statement is a symbol, the address to which the symbol has been equated should be filled in as the proper machine code. For example, the source statement "INC PAD" requires the incrementing of data stored at a location "PAD" defined in our assembly programs to have the address: PAD = 1700. Therefore, the address 1700 is entered as the second and third bytes of the source statement "INC PAD". (See Figure 5-4). Note also that when entering an address, such as 1700, the low order byte (00) is entered first and immediately after the Op Code and the high order byte (17) is entered next as the third byte of the instruction.

When dealing with branch instructions (BPL, BMI, etc.), you will need to calculate the exact value of the offset which may be either positive (branch forward) or negative (branch backward). You should refer to Section 4.1.1 of the Programming Manual to explore "Basic Concept of Relative Branching." As an example, the source statement "BMI START" (See Figures 5-2 and 5-4) requires a branch backward by (-15) locations to the address labelled "START" (from address 0213 backward to 0205 inclusive).

(The 2's complement of the -15 displacement is $F1_{HEX}$ which you should insert at location 0212). Had the branch been to a forward location the positive value of the offset would be inserted rather than the 2's complement value.

5.3 ENTERING THE PROGRAM

With the program now reduced to machine language code, you may enter the program address and data codes listed in Figure 5-4 following the procedures detailed in Section 2.4. The procedure for entering the program is as follows:

Press Keys	See On Display
AD 0 2 0 0	0200 xx
DA A 9	0200 A9
+ 0 1	0201 01
+ 8 D	0202 8D
+ 0 1	0203 01
+ 1 7	0204 17
+ E E	0205 EE
+ 0 0	0206 ØØ
+ 1 7	0207 17
+ A D	0208 AD
+ 0 0	0209 ØØ
+ 1 7	020A 17
+ 4 9	020B 49
+ F F	020C FF
+ 4 A	020D 4A
+ A A	020E AA
+ C A	020F CA
+ 1 0	0210 10
+ F D	0211 FD
+ 3 0	0212 30
+ F 1	0213 F1

Key Sequences: Enter Program
FIGURE 5.5

5.4 EXECUTING THE PROGRAM

With the program entered, you may proceed to program execution. First, if the $\overline{NMI}$ vector has not been defined previously, enter the vector as follows:

Press Keys	See Displayed
AD 1 7 F A	17FA xx
DA 0 0	17FA 00
+ 1 C	17FB 1C

This procedure insures that the ST key will be effective in terminating the program. Now, select the starting address of your program (0200) and begin execution as follows:

Press Keys	See Displayed
AD 0 2 0 0	0200 A9
GO	(Dark)

The program will now execute. If your seven selector switches all are open, you will probably hear no sound from the speaker because the square wave frequency is too high. If all selector switches are closed, you will hear in the speaker the lowest frequency that can be generated with the program as currently written. You may experiment with other combinations of switch settings to hear a variety of tones from the speaker.

Depression of the ST key will cause the program execution to stop (the tone will terminate) and the KIM-1 display will relight. The display will show the address and data for the next instruction to be executed (probably 020F or 0210 since this is the delay loop where the program spends most of its running time).

5.5 PROGRAM DEBUGGING AND MODIFICATION

If your program did not execute correctly, you would follow a debugging procedure involving the following steps:

Step 1: List the Program

First make sure you have entered the program steps correctly. Select the starting address (AD 0 2 0 0) and observe that the correct data (A9) is displayed. Now, using the + key, step through the remaining program locations checking for the correct data stored in each location.

Step 2: Single Step the Program

Follow the procedures listed in Section 5-4 for program execution but before depressing the GO key, place the SST slide switch in the ON position. Now, press the GO key and the first instruction will be executed. The display will relight indicating that the operating program is again in control of the system. The address displayed will be the address of the first byte of the next instruction to be executed. You may press the GO key again to execute the next instruction or you may choose to investigate changes in the contents of machine registers stored in selected memory locations (See Figure 3-13). The procedure detailed in Figure 5-6 gives a good indication of the various operations you may wish to perform in the SST mode.

Step 3: Check the I/O Operations

If program entry has been verified and program execution in the SST mode appears to be normal, you may wish to verify the correct operation of your specific I/O configuration.

You should recall that writing to or reading from any I/O port is the same as reading from or writing to any other memory location in the system. Therefore, if you select the address of an I/O port, the KIM-1 display will show you the hex code for the data being read from that address and thus, directly indicate the state of each I/O pin in the port. For example, the address of the I/O port used for your sample program is 1700. Press AD 1 7 0 0 and the display will show the hex code corresponding to the settings of your selector switches. If you change the positions of your selector switches, you will see the hex code change in the data field of the display.

Now, leave the same address (1700) selected and press the DA key. If you press any of the hex keys 0 to F, you will write the data to the I/O port (1700). Since seven of the pins of this I/O port are defined as inputs, only one (PAØ) will act as an output and will respond to the data entered by you from the keyboard. Try alternating rapidly between the 0 and 1 keys and you should hear clicking in the speaker indicating that you are successfully toggling the PAØ pin.

This concept of using the KIM-1 keyboard and display to exercise and verify the operation of I/O ports is a generally useful technique for debugging the hardware portions of most specific applications.

Press Keys	See Displayed	Comments
AD 0 2 0 0	0200 A9	Select first instruction address
SST ON	0200 A9	Set SST to ON; All selector switches open
GO	0202 8D	Accumulator now loaded with $01
GO	0205 EE	PADD now loaded
GO	0208 AD	PAØ now toggled
GO	020B 49	Switch values (PA1-PA7) now loaded
GO	020D 4A	Accumulator now complemented
GO	020E AA	Accumulator now right shifted 1 Bit
AD 0 0 F 3	00F3 xx	Display Accumulator
+	00F4 xx	Display Y - INDEX
+	00F5 00	Display X - INDEX
PC	020E AA	Restore PC (TAX will execute next)
GO	020F CA	Accumulator now loaded in X-INDEX
AD 0 0 F 3	00F3 00	Display Accumulator
+	00F4 xx	Display Y-INDEX
+	00F5 00	Display X-INDEX (A=0→X)
PC	020F CA	Restore PC
GO	0210 10	DEX now completed
AD 0 0 F 5	00F5 FF	Display X-INDEX (X<0)
PC	0210 10	Restore PC
GO	0212 30	No branch (Result of DEX <u>not</u> positive)
GO	0205 EE	Branch (Result of DEX <u>is</u> negative).

SST Mode: Sample Operation
FIGURE 5.6

CHAPTER 6

EXPANDING YOUR SYSTEM

In earlier sections you have learned that the MCS 6502 Microprocessor Array is capable of directly addressing up to 65,536 locations (bytes) of memory. (Usually abbreviated to 65K where "K" for the remainder of this section is to mean 1024 memory locations). In this section, we will discuss first the techniques for adding memory or I/O locations to the system and next, the proper handling of interrupt vectors in an expanded system.

6.1 MEMORY AND I/O EXPANSION

In the KIM-1 system, the management of input/output data is handled exactly the same as transfers to or from any other memory location in the system. There are no instructions dealing specifically with input/output transfers. Instead, transfer of data is accomplished by reading from or writing to registers connected to the data bus and to I/O pins in specific I/O interface devices (such as the 6530 array). These registers have a specific address in the system just as does any other memory location. Therefore, when we speak of expanding the memory of the KIM-1 system, we are defining the methods for expanding both the real memory (RAM, ROM, PROM, etc.) as well as the I/O ports since they are both treated exactly alike as far as address assignments are concerned.

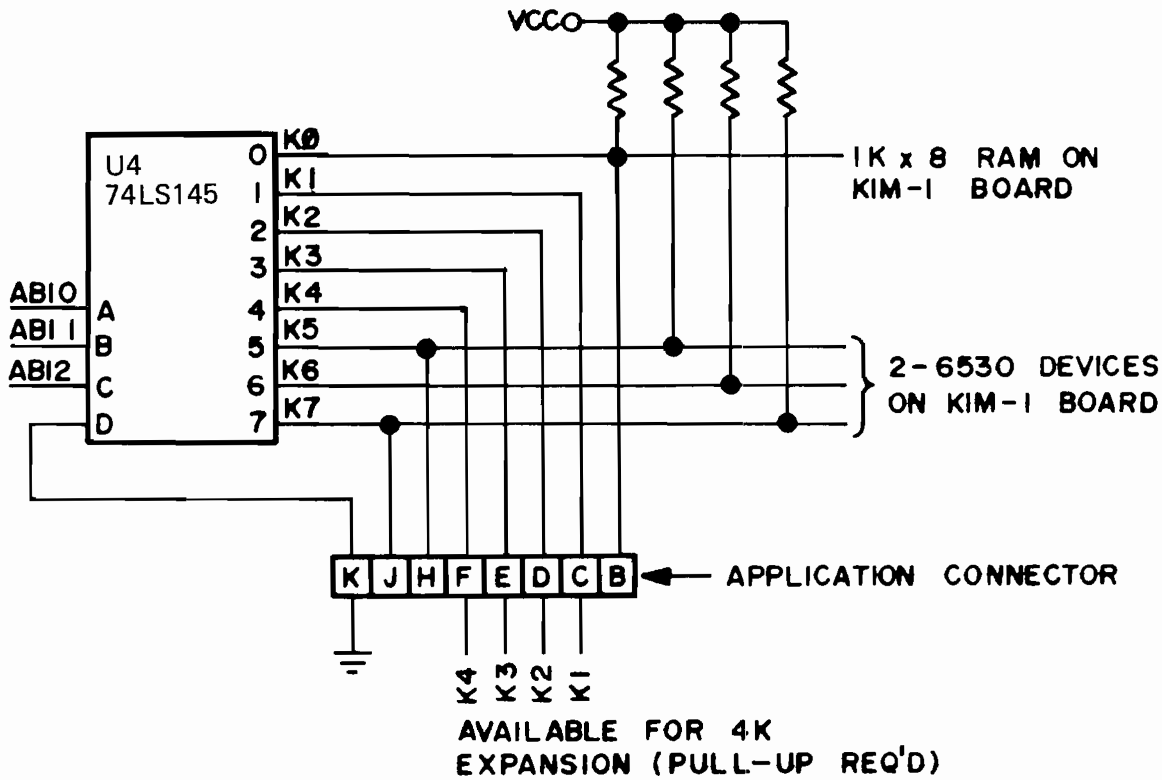
The first and most easilly implemented memory expansion is the addition of up to 4K of memory space. You will recall that the lowest 8K memory locations are defined by an address decoder included on the KIM-1 module, (Device U4 on the schematic). The eight outputs of this decoder (KØ to K7) each define a 1K block of addresses in the lowest 8K of the memory map. Three of the outputs (K5, K6, K7) are used to select ROM, RAM, I/O and Timer locations on the two 6530 arrays while a fourth (KØ) is used to select the 1024 locations of the static RAM memory. The remaining four outputs (K1, K2, K3, K4) are not used on the KIM-1 module but instead, are brought out to the Expansion connector for use as chip selects for memory or I/O additions.

Figure 6-1 shows the proper method for deriving the four chip select signals for the additional 4K of memory. Note that one of input pins of the decoder (D) was brought out to the Application Connector. It was this pin which we asked you to connect to ground in Chapter 2 of this manual. As long as this point remains connected to ground, the decoder will always select the lowest 8K addresses of the memory field regardless of the state of AB13, AB14, and AB15.

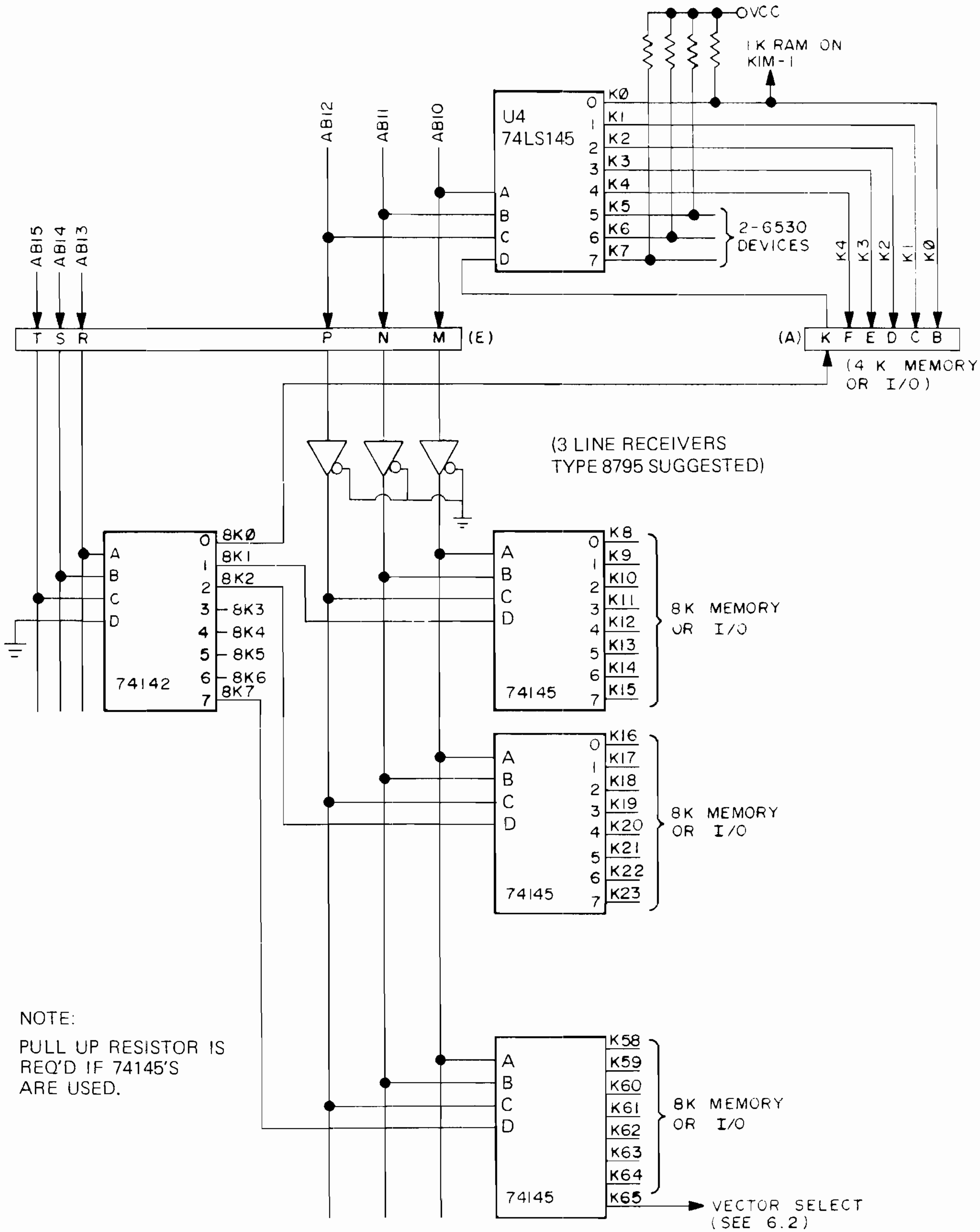
If you wish to expand the memory and I/O address space beyond the lower 8K addresses, you must arrange to de-select the lower 8K memory block while selecting some other 8K block. One suggested method for expanding beyond the lower 8K space is shown in Figure 6-2.

Note that the three high order address bits (AB13, AB14, AB15) are connected to a decoder. The eight outputs of the decoder act to divide the total 65K memory space into eight blocks of 8K each (8KØ, 8K1, etc.). Now, the 8KØ output may be returned as the fourth input (D) to the decoder (U4) on the KIM-1 module causing the proper selection and de-selection of this block within the total address space. The remaining seven outputs (8K1 to 8K7) may be used to select and de-select the additional decoders shown in Figure 6-2. You need add only as many decoders (one for each 8K block of memory) as you need for your desired memory expansion.

A word of caution is in order when you decide to add memory to your system. You have noticed the inclusion of the line receivers for the AB10, AB11, and AB12 signals, (See Figure 6-2). These devices are included because of loading limitations placed on the address bus lines of the 6502 array (Each such line is capable of driving one standard TTL load and 130pf of capacity. See Appendix G).



4K Expansion
FIGURE 6.1



65K Expansion
FIGURE 6.2

Before deciding how to expand your system, we recommend a careful study of all of the loading limitations of the KIM-1 signals since almost certainly you will require additional buffering circuits if correct operation is to be achieved.

6.2 INTERRUPT VECTOR MANAGEMENT

We have referred several times in earlier sections to the interrupt features of the 6502 Microprocessor Array. We suggest now a careful reading of Section 9 of the Programming Manual for the subject "Reset and Interrupt Considerations".

In summary, there are three possible types of interrupt: Reset, NMI, and IRQ. Each will occur in response to an activation of one of the three pins of the 6502 array ($\overline{\text{RST}}$, $\overline{\text{NMI}}$, $\overline{\text{IRQ}}$). In response to these inputs, the 6502 array will fetch the data stored at a specific pair of addresses and load the data fetched into the program counter. The addresses are <u>hardware</u> determined and not under the control of the programmer. The specific addresses for each type of interrupt are:

FFFA, FFFB - $\overline{\text{NMI}}$ Vector
FFFC, FFFD - $\overline{\text{RST}}$ Vector
FFFE, FFFF - $\overline{\text{IRQ}}$ Vector

You will note that these addresses define the highest six locations in the 65K memory map.

In the KIM-1 system, three address bits (AB13, AB14, AB15) are not decoded at all. Therefore, when the 6502 array generates a fetch from FFFC and FFFD in response to a $\overline{\text{RST}}$ input, these addresses will be read as 1FFC and 1FFD and the reset vector will be fetched from these locations. You now see that all interrupt vectors will be fetched from the top 6 locations of the lowest 8K block of memory which is the only memory block decoded for the unexpanded KIM-1 system.

It is typical in any system to store the interrupt vectors in ROM so that they are immediately available after power-on. However, it is desirable that for the $\overline{\text{NMI}}$ and $\overline{\text{IRQ}}$ interrupts, the programmer be allowed to define as a variable the exact vector to which these interrupts will direct the system. Accordingly, the $\overline{\text{NMI}}$ and $\overline{\text{IRQ}}$ vector locations contain an indirect jump instruction referencing a RAM location into which the programmer will store the specific vector for the two types of interrupt. In the KIM-1 system, locations 17FA and 17FB contain the actual $\overline{\text{NMI}}$ vector and 17FE with 17FF contain the actual $\overline{\text{IRQ}}$ vector. The $\overline{\text{RST}}$ vector is not handled in this manner and always directs the system to the first step of the power-on initialization routine.

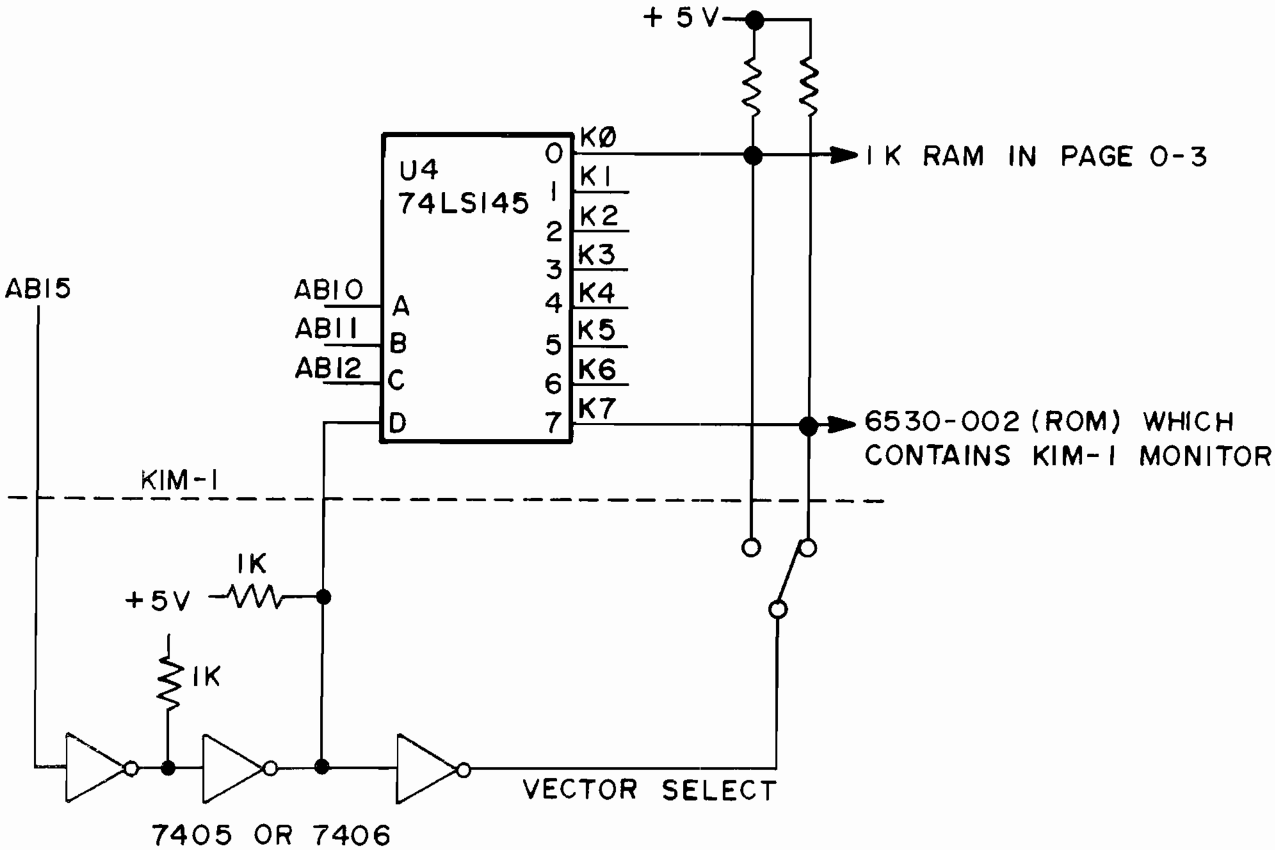
But what happens if we expand our memory above the lowest 8K block included in the KIM-1 system? Recall that we now must use AB13, AB14, and AB15 to decode the additional address locations of the memory. By so doing, the interrupt vector locations are no longer located in the K7 memory block since the decoder (U4) is de-selected in response to the addresses generated by the 6502 array in fetching the interrupt vectors (FFFA for example). We would have the same problem even in an unexpanded system if we wished to use a $\overline{\text{RST}}$ vector and initialization routine different than what the KIM-1 system provides and if the $\overline{\text{RST}}$ vector was to be located in a 1K block lower than K7 (KØ for instance).

The solution to this dilemma is to generate logically a special signal for interrupt select. Referring to Figure 6-2, a special signal called "Vector Select" is created to define the highest 1K memory block (K65). The fetch of any interrupt vector will cause this signal to go low "Select". Assuming that the K65 state is not used to select RAM, this signal may be "wire-or'd" with any one of the other "K" signals (KØ to K64) to define exactly which 1K block is to contain the interrupt vectors.

As an example, assume that you have connected the K65 "Vector Select" line to the KØ line. When a $\overline{\text{RST}}$ occurs, the 6502 array generates a fetch from locations FFFC and FFFD. These addresses cause K65 to be selected which, in turn, accesses the KØ field of the memory and causes the actual fetch of the $\overline{\text{RST}}$ vector from locations 03FC and 03FD. (Had you chosen to connect K65 to K7, the fetch of the reset vectors would occur from locations 1FFC and 1FFD).

In this way, the highest six addresses of any 1K block of memory may be used to supply the interrupt vectors for the system. If desired, a switch could be installed to allow you to select different areas of memory as the source locations for the interrupt vectors. (By the way, we selected the 75145 type decoders in Figure 6-2 specifically to allow the "wire-or" of K65 with any other K. This is possible because the 75145 decoder is provided with open-collector outputs which allows "wire-or" of several states using an external load resistor.)

An even simpler arrangement using the "Vector Select" approach is shown in Figure 6-3. Here, the KIM-1 system is assumed to have only the lower 8K of memory in place. The address decoder (U4) is de-selected using the AB15 signal which becomes "true" whenever an interrupt vector fetch is initiated by the system. The same signal (AB15) is inverted and "wire-or'd" through a switch to the KØ or the K7 chip select lines. Now, depending upon the position of the switch, interrupt vectors will be fetched from the top 6 addresses of either block KØ or K7. KØ in the KIM-1 system is the RAM and K7 is the ROM in the 6530-002 array (the operating program). In this way, you may have two different sets of interrupt vectors in your system and may select which set is to be used with a simple switch.



Vector Selection
FIGURE 6.3

CHAPTER 7

WARRANTY AND SERVICE

Should you experience difficulty with your KIM-1 module and be unable to diagnose or correct the problem, you may return the unit to MOS Technology, Inc. for repair.

7.1 IN-WARRANTY SERVICE

All KIM series Microcomputer Modules are warranted by MOS Technology, Inc. against defects in workmanship and materials for a period of ninety (90) days from date of delivery. During the warranty period, MOS Technology, Inc. will repair or, at its option, replace at no charge components that prove to be defective provided that the module is returned, shipping prepaid, to:

KIM Customer Service Department
MOS Technology, Inc.
950 Rittenhouse Road
Norristown, Pennsylvania 19401

This warranty does not apply if the module has been damaged by accident or misuse, or as a result of repairs or modifications made by other than authorized personnel at the above captioned service facility.

No other warranty is expressed or implied. MOS Technology, Inc. is not liable for consequential damages.

7.2 OUT-OF-WARRANTY SERVICE

Beyond the ninety (90) day warranty period, KIM modules will be repaired for a reasonable service fee. All service work performed by MOS Technology, Inc. beyond the warranty period is warranted for an additional ninety (90) day period after shipment of the repaired module.

7.3 POLICY ON CHANGES

All KIM series modules are sold on the basis of descriptive specifications in effect at the time of sale. MOS Technology, Inc. shall have no obligation to modify or update products once sold. MOS Technology, Inc. reserves the right to make periodic changes or improvements to any KIM series module.

7.4 SHIPPING INSTRUCTIONS

It is the customer's responsibility to return the KIM series module with shipping charges prepaid to the above captioned service facility.

For in-warranty service, the KIM module will be returned to the customer, shipping prepaid, by the fastest economical carrier.

For out-of-warranty service, the customer will pay for shipping charges both ways. The repaired KIM module will be returned to the customer C.O.D. unless the repairs and shipping charges are prepaid by the customer.

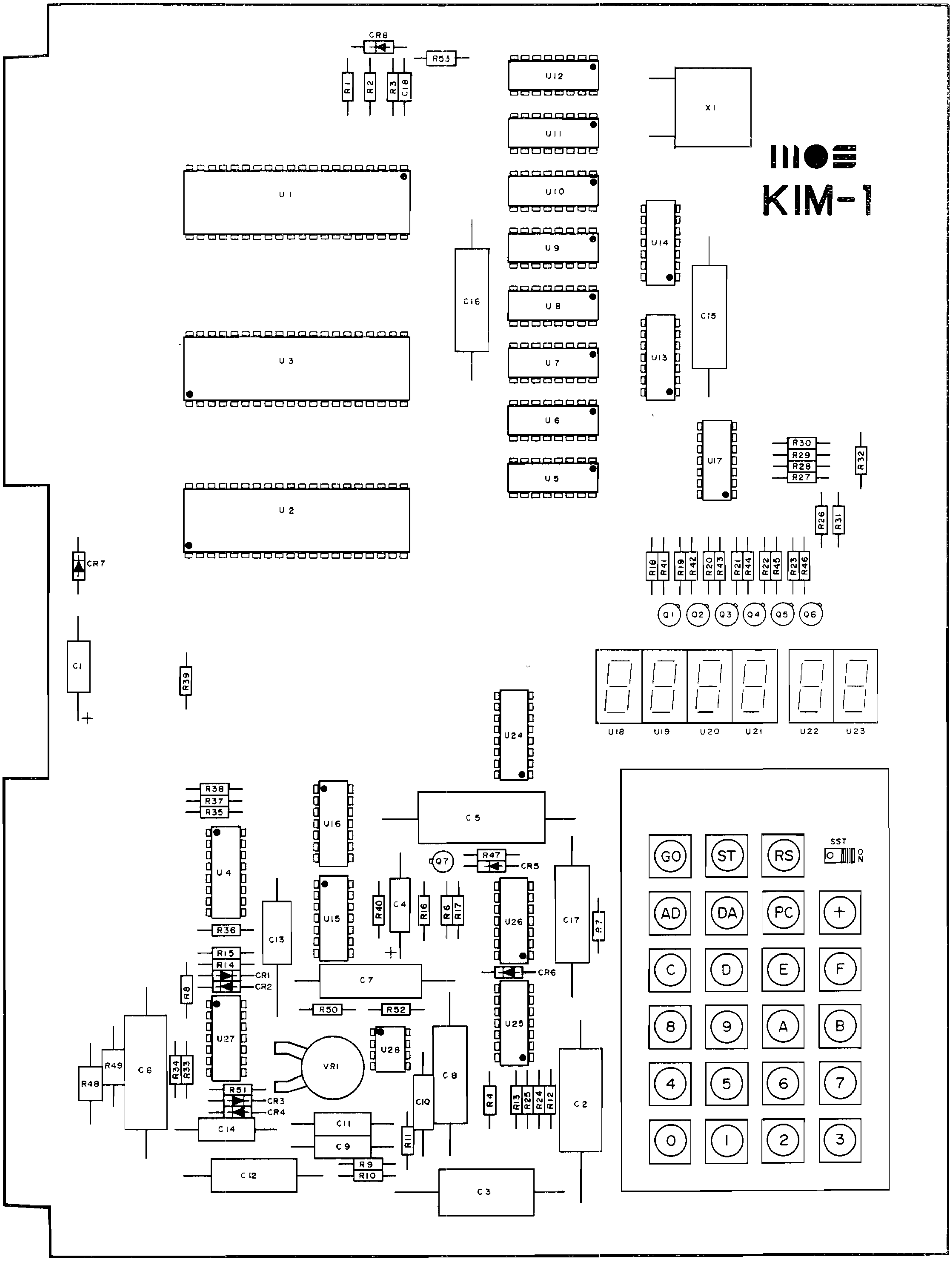
Please be certain that your KIM module is safely packaged when returning it to the above captioned service facility.

APPENDIX A

ITEM	PART	QTY.	DESCRIPTION
1.	U1	1	6502 Microprocessor
2.	U2	1	6530 ROM RAM I/O Chip-02
3.	U3	1	6530 ROM RAM I/O Chip-03
4.	U5 through U12	8	6102 RAM 500ns Acc,Øns
5.	U18 through U23	6	7 SEG .3" Red Display
6.	U25	1	556 Timer IC
7.	U27	1	565 Phase Lock Loop
8.	U28	1	311 Comparator
9.	U24	1	74145 BCD Decoder IC
10.	U13 & U14	2	74125 TRI STATE Buffer
11.	U15	1	7400 Quad Nand IC
12.	U16	1	7404 Hex Inverter IC
13.	U17	1	7406 Hex Inv. O/C IC
14.	U26	1	7438 Quad Nand O/C IC
15.	CR1,2,3,4,&8	5	20 MA. 50v Diode - IN914
16.	CR5, CR6	2	1A 50v Diode - IN4001
17.	CR7	1	6.2v ½w Z. Diode - 1N4735
18.	Q7	1	NPN Transistor B>20, VCE>12 - 2N5371
19.	Q1 through Q6	6	PNP Transistor B>20, VCE>6 - 2N5375
20.	R24 & R25	2	47KΩ ±10% ¼w Resistor
21.	R1,2,3,4, & 6	5	3.3KΩ ±10% ¼w Resistor
22.	R34 & R50	2	2.2KΩ ±10% ¼w Resistor
23.	R12-R17, R41-R46	12	1.0KΩ ±10% ¼w Resistor
24.	R35 through R40	6	560Ω ±10% ¼w Resistor
25.	R18-R23, R47	7	220Ω ±10% ¼w Resistor
26.	R33	1	47Ω ±10% ¼w Resistor
27.	R52	1	5 Meg. ±10% ¼w Resistor
28.	R51	1	30KΩ ±5% ¼w Resistor
29.	R7,R8,R9,R10&R11	5	10KΩ ±5% ¼w Resistor
30.	R48, R49	2	150Ω ±5% ½w
31.	R26 through R32	7	82Ω ±5% ¼w
32.	VR1	1	5KΩ Potentiometer
33.	C2, C3, C6	3	.22±10% uf.>12 wv. cap
34.	C1, C4	2	1uf+80-10%>12WV Cap
35.	C5	1	.33 uf±10%>12WV Cap
36.	C7,C8,C15,C16,C17	5	.1uf+80-10%>12WV Cap
37.	C9, C10, C11	3	.0068uf±10%>12WV
38.	C12	1	.047uf±10%>12WV
39.	C13	1	.022uf±10%>12WV
40.	C14	1	.001uf±10%>12WV
41.		1	44 Pin Edge Conn. (Vector #R644)
42.	X1	1	1 MHz XTAL
43.		1	PCB.
44.		1	24 Key KBD
45.		6	Rubber Pads
46.		1	Shipping Bag (Static Free)
47.		1	Shipping Box
48.		1	Hardware Manual
49.		1	Software Manual
50.		1	KIM Manual
51.		1	Warranty Card
52.		1	Wall Chart
53.		2	#2 x ¼ SS Screws (Keyboard)
54.		1	Program Card
55.	C18	1	10pf CAP
56.	R53	1	330K ¼w Resistor
57.	U4	1	74LS145 BCD Decoder 1C

APPENDIX B

KIM-1 PARTS LAYOUT



APPENDIX C

IN CASE OF TROUBLE

SYMPTOM: Display Not Lit

1. Test +5 volt power supply. Using a VOM check for +5 volts between Pin E-21 and E-22. Also check for +5 volts between Pin A-A and Pin A-1. KIM-1 power supply should be set at +5v ± 5%.

2. Test KB/TTY option wiring (Figure 2-4). Pin A-21 should not be connected to Pin A-V.

3. Make sure decoder is enabled. See Figure 2-2 and insure that Pin A-K is connected to ground.

4. Depress the reset key and check all other keys to insure that no key is stuck.

5. Place a VOM between Pin E-21 (+5v) and Pin E-7 (Reset). Alternately depress and release the reset key checking to see if the voltage swings from (>4v) to (<1v).

6. Test Pin E-V ($\emptyset_2$) with an oscilloscope and insure 1 MHz operation.

SYMPTOM: Cannot Dump to Audio Tape
Cannot Load From Audio Tape

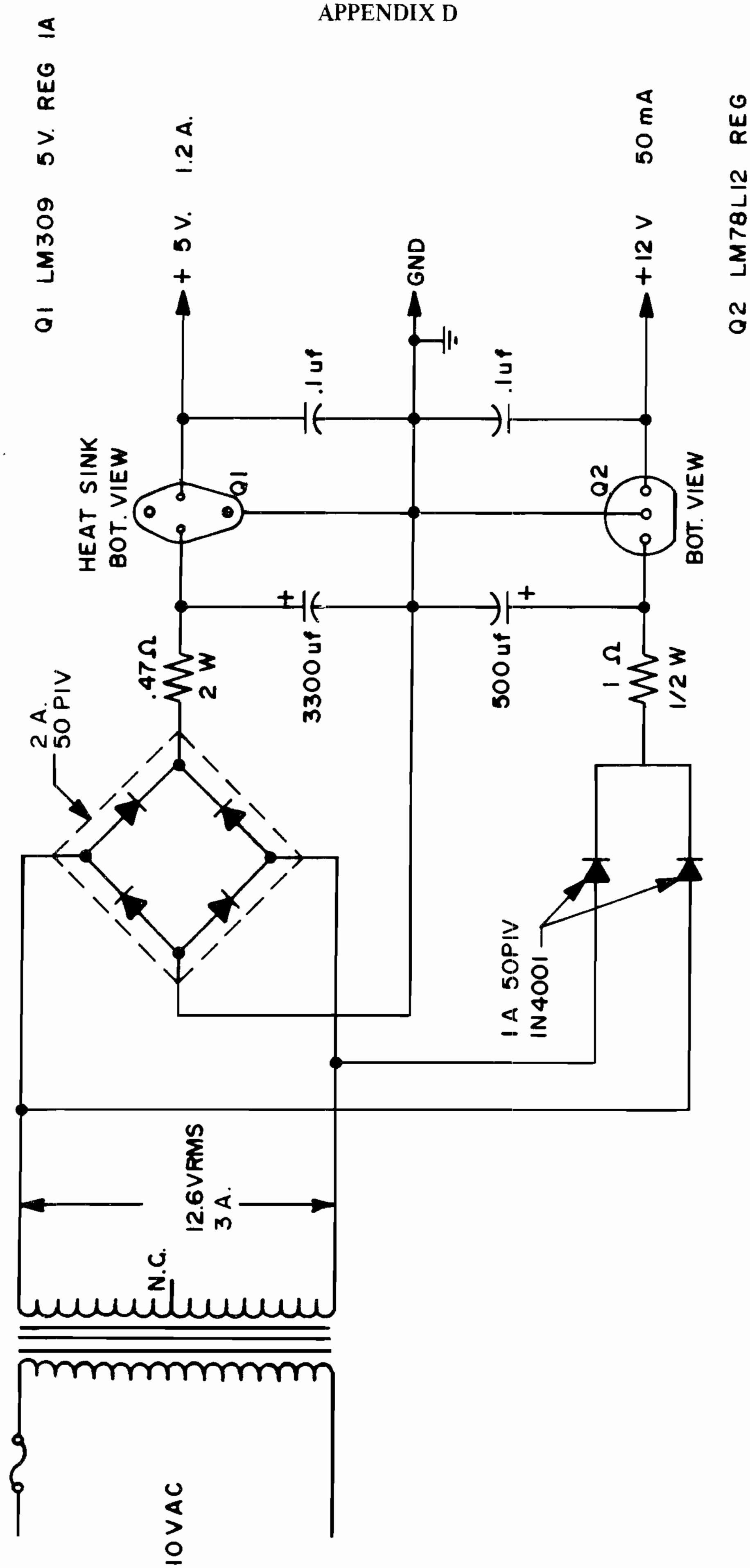
1. Test +12 volt power supply. Using a VOM check for +12 volts between Pin A-N (+12v) and Pin A-1 (GND). Set power supply to +12v ± 5%. (See Figure 2-2).

2. Check volume control on the tape recorder (Set at half way point).

3. Make sure that you are using the proper tape output pin. See Figure 2-3.
4. Check the tape interface circuit by disconnecting the tape recorder and shorting Pin A-P (Audio Out High) to Pin A-L (Audio In). Set up KIM-1 monitor to dump a section of memory. Using an oscilloscope observe data at Pin E-X (PLL TEST). See Appendix E for correct data format and calibration procedure.
5. Record voice on a section of tape and play it back to insure that the tape recorder is working. Connect another tape recorder to the system or try another cassette.
6. Make sure Status Register (Location 00F1) has been loaded with data value "00".
7. Make sure Tone Control is set to High.

SYMPTOM: <u>TTY Interface Problems</u>

1. Make sure that Pin A-21 is connected to Pin A-V (Figure 2-4) to allow TTY operation.
2. Compare the connections on Figure 2-4 with interface schematics in your TTY manual (or any other serial teleprinter).
3. Depress the reset key on the KIM-1 keyboard followed by a rub out character from the TTY.



Suggested Power Supply

APPENDIX E

AUDIO TAPE FORMAT

Data is stored out onto your audio cassette recorder in a specific format designed to insure an error free recovery. In the unlikely event that a playback error does occur, several "ERROR DETECTION" methods are incorporated to warn you of this condition.

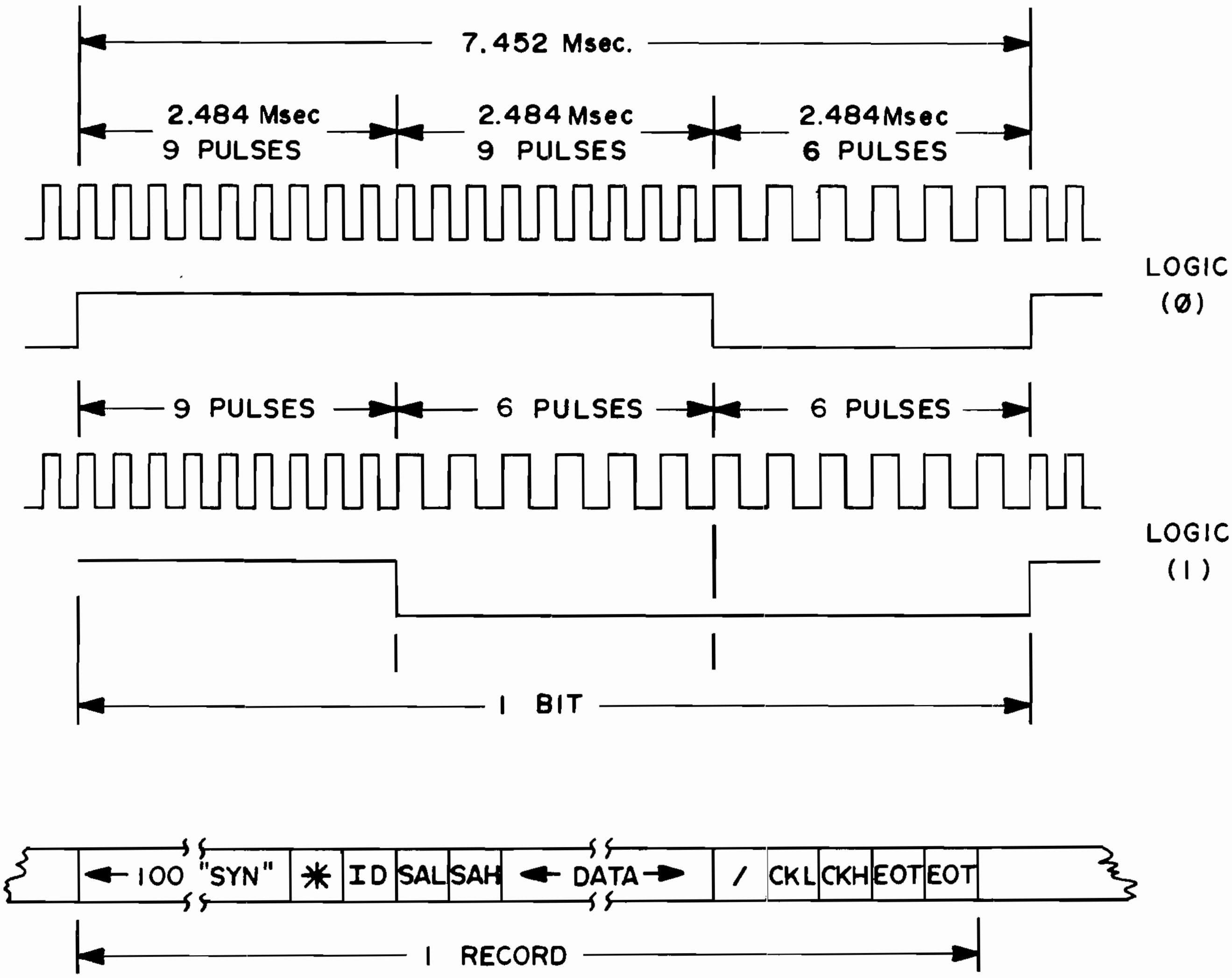
Data is transmitted to the tape recorder in the form of serial "ASCII" encoded characters (seven data bits plus Parity bit). Data retrieved from the memory is converted into this form by separating each byte into two half bytes. The half bytes are then converted into their ASCII equivalents.

Each record transmitted begins with a leader of one hundred "SYN" characters (ASCII 16) followed by a * character (ASCII 2A). During playback, this pattern allows your micro-computer to detect the start of a valid data record and synchronize to the serial data stream. Following the *, the record identification number (ID), and starting address low (SAL) and the starting address high (SAH) are transmitted. The data specified by the starting (SAL, SAH) and ending limits (EAL, EAH) is transmitted next followed by a "/" character (ASCII 2F) to indicate the end of the data portion of the record. Following the "/" two "CHECK-SUM" bytes are transmitted for comparison with a calculated check-sum number during playback to further insure that a proper data retrieval has taken place. Two "EOT" characters (ASCII Ø4) mark the end of record transmission.

Each transmitted bit begins with a 3700 hertz tone and ends with a 2400 hertz tone. "Ones" have the high to low frequency transition at one-third of the bit period. "Zeros" have the transition at two-thirds of the period. During playback the 565 phase locked loop locks to, and tracks these two frequencies producing (through the 311 comparator) a logic "1" pulse of one-third the bit period for a "One". A pulse two thirds the bit period is likewise produced for a "Zero". Your microcomputer uses a software controlled algorithm for converting this signal into eight bit data words.

The frequency shift keyed phase lock loop method of data recovery is relatively insensitive to amplitude and phase variations. The "FREE RUNNING" frequency of the phase lock loop has been adjusted at the factory to a frequency half way between the two data frequencies (called the Center Frequency). This adjustment is accomplished by strapping Pin A-P (Audio Out High) to Pin A-L (Audio In). A program starting at address $1A6B_{HEX}$ provides the center frequency reference that allows the loop to be adjusted by potentiometer VR1. Pin E-X (PLL TEST) is monitored with a voltmeter while the pot is rotated until the voltmeter reading is at the transition point between a logical "1" (+5v) and "0" (GND).

THIS ADJUSTMENT HAS BEEN FACTORY PRESET AND SHOULD ONLY REQUIRE ADJUSTMENT DUE TO COMPONENT REPLACEMENT!



Audio Tape Format
FIGURE E-1

APPENDIX F

PAPER TAPE FORMAT

The paper tape LOAD and DUMP routines store and retrieve data in a specific format designed to insure error free recovery. Each byte of data to be stored is converted to two half bytes. The half bytes (whose possible values are Ø to F_{HEX}) are translated into their ASCII equivalents and written out onto paper tape in this form.

Each record outputted begins with a ";" character (ASCII 3B) to mark the start of a valid record. The next byte transmitted (18_{HEX}) or (24_{10}) is the number of data bytes contained in the record. The record's starting address High (1 byte, 2 characters), starting address Lo (1 byte, 2 characters), and data (24 bytes, 48 characters) follow. Each record is terminated by the record's check-sum (2 bytes, 4 characters), a carriage return (ASCII 0D), line feed (ASCII ØA), and six "NULL" characters (ASCII ØØ).

The last record transmitted has zero data bytes (indicated by ;ØØ). The starting address field is replaced by a four digit Hex number representing the total number of data records contained in the transmission, followed by the records usual check-sum digits. A "XOFF" character ends the transmission.

```
;180000FFEEDDCCBBAA0099887766554433221122334455667788990AFC
;0000010001
```

During a "LOAD" all incoming data is ignored until a ";" character is received. The receipt of non ASCII data or a mismatch between a records calculated check-sum and the check-sum read from tape will cause an error condition to be recognized by KIM. The check-sum is calculated by adding all data in the record except the ";" character.

The paper tape format described is compatible with all other MOS Technology, Inc. software support programs.

APPENDIX G

6502 CHARACTERISTICS

Clocks ($\emptyset_1$, $\emptyset_2$)

The MCS 6502 is supplied with an internal clock generator. The frequency of this clock is crystal controlled.

Address Bus (A_0-A_{15})

These outputs are TTL compatible, capable of driving one standard TTL load and 130pf.

Data Bus (D_0-D_7)

Eight pins are used for the data bus. This is a bi-directional bus, transferring data to and from the device and peripherals. The outputs are tri-state buffers capable of driving one standard TTL load and 130pf.

Ready (RDY)

This input signal allows the user to single cycle the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one ($\emptyset_1$) will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two ($\emptyset_2$) in which the Ready signal is high. This feature allows microprocessor interfacing with low speed PROMS as well as fast (max. 2 cycle) Direct Memory Access (DMA). If Ready is low during a write cycle, it is ignored until the following read operation.

Interrupt Request ($\overline{IRQ}$)

This TTL level input requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, therefore transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state(for control to the memory vector) located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3KΩ external register should be used for proper wire-OR operation.

Non-Maskable Interrupt ($\overline{NMI}$)

A negative going edge on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{IRQ}$ will be performed, regardless of the state of the interrupt mask flag. The vector address loaded into the program counter, low and high, are locations FFFA and FFFB respectively. The instructions loaded at these locations causes the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ also requires an external 3KΩ resistor to Vcc for proper wire-OR operations.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupts lines that are sampled during $\emptyset_2$ (phase 2) and will begin the appropriate interrupt routine on the $\emptyset_1$ (phase 1) following the completion of the current instruction.

Set Overflow Flag (S.O.)

This TTL level input signal allows external control of the overflow bit in the Status Code Register.

SYNC

This output line is provided to identify those cycles in which the microprocessor is doing an Op Code fetch. The SYNC line goes high during $\emptyset_1$ of an Op Code fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the $\emptyset_1$ clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

RESET

This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory vector locations FFFC and FFFD. This is the start location for program control.

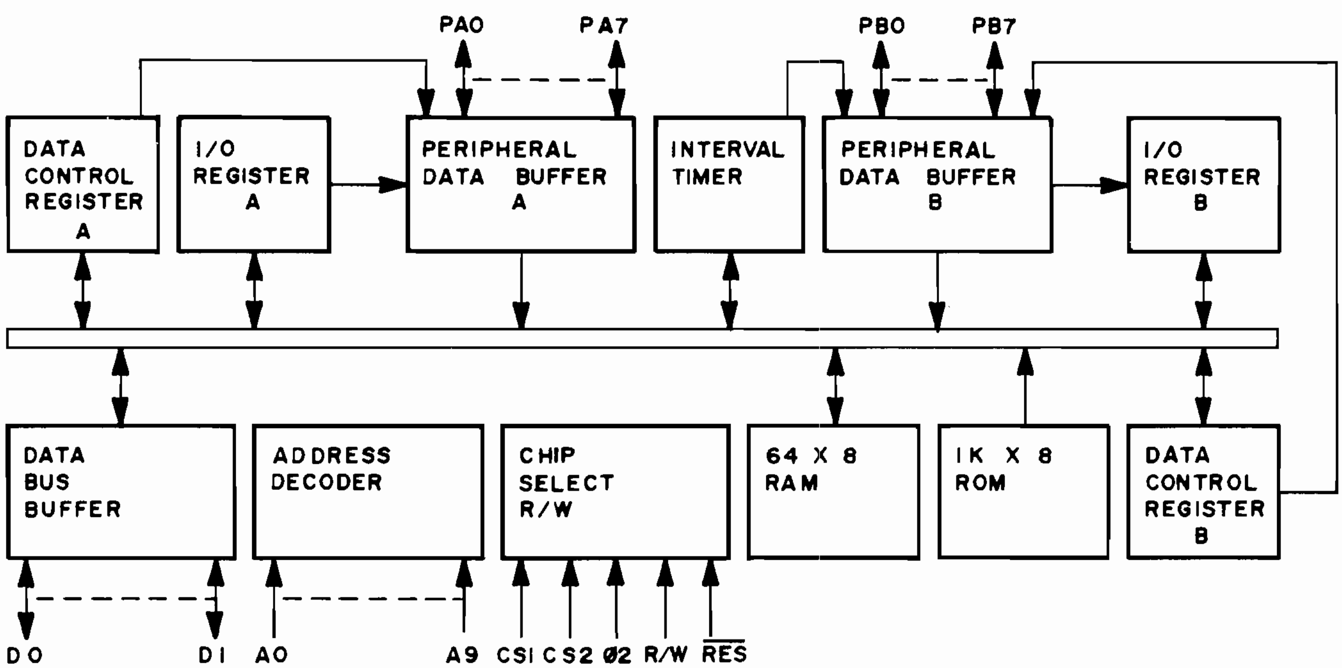
After Vcc reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

APPENDIX H

6530 CHARACTERISTICS

The MCS 6530 is designed to operate in conjunction with the MCS 650X Microprocessor Family. It is comprised of a mask programmable 1024 x 8 ROM, a 64 x 8 static RAM, two software controlled 8 bit bi-directional data ports allowing direct interfacing between the microprocessor unit and peripheral devices, and a software programmable interval timer with interrupt, capable of timing in various intervals from 1 to 262,144 clock periods.



MCS 6530 Block Diagram
FIGURE H.1

Reset ($\overline{RES}$)

During system initialization a Logic "0" on the $\overline{RES}$ input will cause a zeroing of all four I/O registers. This in turn will cause all I/O buses to act as inputs thus protecting external components from possible damage and erroneous data while the system is being configured under software control. The Data Bus Buffers are put into an OFF-STATE during Reset. Interrupt capability is disabled with the $\overline{RES}$ signal. The $\overline{RES}$ signal must be held low for at least one clock period when reset is required.

Input Clock

The input clock is a system Phase Two clock which can be either a low level clock ($V_{IL} < 0.4$, $V_{IH} > 2.4$) or high level clock ($V_{IL} < 0.2$, $V_{IH} = Vcc \, {}^{+.3}_{-.2}$).

Read/Write (R/W)

The R/W signal is supplied by the microprocessor array and is used to control the transfer of data to and from the microprocessor array and the MCS 6530. A high on the R/W pin allows the processor to read (with proper addressing) the data supplied by the MCS 6530. A low on the R/W pin allows a write (with proper addressing) to the MCS 6530.

Interrupt Request ($\overline{IRQ}$)

The $\overline{IRQ}$ pin is an interrupt pin from the interval timer. This same pin, if not used as an interrupt, can be used as a peripheral I/O pin (PB7). When used as an interrupt, the pin should be set up as an input by the data direction register. The pin will be normally high with a low indicating an interrupt from the MCS 6530.

Data Bus (DØ-D7)

The MCS 6530 has eight bi-directional data pins (DØ-D7). These pins connect to the system's data lines to allow transfer of data to and from the microprocessor array. The output buffers remain in the off state except when a Read operation occurs.

Peripheral Data Ports

The MCS 6530-002, MCS 6530-003 both have 15 pins available for peripheral I/O operations. Each pin is individually software programmable to act as either an input or an output. The 15 pins are divided into 2 8-bit ports, PAØ-PA7 and PBØ-PB7. PB6 was used as a chip select and is not available to the user. The pins are set up as an input by writing a "0" into the corresponding bit of the data direction register. A "1" into the data direction register will cause its corresponding bit to be an output. When in the input mode, the peripheral output buffers are in the "1" state and a pull-up device acts as less than one TTL load to the peripheral data lines. On a Read operation, the microprocessor unit reads the peripheral pin. When the peripheral device gets information from the MCS 6530 it receives data stored in the data register. The microprocessor will read correct information if the peripheral lines are greater than 2.0 volts for a "1" and less than 0.8 volts for a "0" as the peripheral pins are all TTL compatible. Pins PAØ and PBØ are also capable of sourcing 3 ma at 1.5v, thus making them capable of Darlington drive. Pin PB7 has no internal pull-up (to allow collector-oring with other devices).

Address Lines (AØ-A9)

There are 10 address pins. In addition to these 10, there is the ROM SELECT pin. The above pins, AØ-A9 and ROM SELECT, are always used as addressing pins. There are 2 additional pins which are mask programmable and can be used either individually or together as CHIP SELECTS. They are pins PB5 and PB6. When used as peripheral data pins they cannot be used as chip selects. PB5 was used as a data pin while PB6 was used as a chip select and is not available to the user.

A block diagram of the internal architecture is shown in Figure H-1. The MCS 6530 is divided into four basic sections, RAM, ROM, I/O and TIMER. The RAM and ROM interface directly with the microprocessor through the system data bus and address lines. The I/O section consists of 2 8-bit halves. Each half contains a Data Direction Register (DDR) and an I/O Register.

ROM 1K Byte (8K Bits)

The 8K ROM is in a 1024 x 8 configuration. Address lines AØ-A9, as well as RS0 are needed to address the entire ROM. With the addition of CS1 and CS2, seven MCS 6530's may be addressed, giving 7168 x 8 bits of contiguous ROM.

RAM 64 Bytes (512 Bits)

A 64 x 8 static RAM is contained on the MCS 6530. It is addressed by AØ-A5 (Byte Select), RSØ, A6, A7, A8, A9 and CS1.

Internal Peripheral Registers

There are four internal registers, two data direction registers and two peripheral I/O data registers. The two data direction registers (A side and B side) control the direction of the data into and out of the peripheral pins. A "1" written into the Data Direction Register sets up the corresponding peripheral buffer pin as an output. Therefore, anything then written into the I/O Register will appear on that corresponding peripheral pin. A "0" written into the DDR inhibits the output buffer from transmitting data to or from the I/O Register. For example, a "1" loaded into data direction register A, position 3, sets up peripheral pin PA3 as an output. If a "0" had been loaded, PA3 would be configured as an input and remain in the high state. The two data I/O registers are used to latch data from the Data Bus during a Write operation until the peripheral device can read the data supplied by the microprocessor array.

During a read operation the microprocessor is not reading the I/O Registers but in fact is reading the peripheral data pins. For the peripheral data pins which are programmed as outputs the microprocessor will read the corresponding data bits of the I/O Register. The only way the I/O Register data can be changed is by a microprocessor Write operation. The I/O Register is not affected by a Read of the data on the peripheral pins.

Interval Timer

1. Capabilities

The KIM-1 Interval Timer allows the user to specify a preset count of up to 256_{10} and a clock divide rate of 1, 8, 64 or 1024 by writing to a memory location. As soon as the write occurs, counting at the specified rate begins. The timer counts down at the clock frequency divided by the divide rate. The current timer count may be read at any time. At the user's option, the timer may be programmed to generate an interrupt when the counter counts down past zero. When a count of zero is passed, the divide rate is automatically set to 1 and the counter continues to count down at the clock rate starting at a count of FF (-1 in two's complement arithmetic). This allows the user to determine how many clock cycles have passed since the timer reached a count of zero. Since the counter never stops, continued counting down will reach 00 again, then FF, and the count will continue.

2. Operation

a. Loading the timer

The divide rate and interrupt option enable/disable are programmed by decoding the least significant address bits. The starting count for the timer is determined by the value written to that address.

Writing to Address	Sets Divide Ratio To	Interrupt Capability Is
1704	1	Disabled
1705	8	Disabled
1706	64	Disabled
1707	1024	Disabled
170C	1	Enabled
170D	8	Enabled
170E	64	Enabled
170F	1024	Enabled

b. Determining the timer status

After timing has begun, reading address location 1707 will provide the timer status. If the counter has passed the count of zero, bit 7 will be set to 1, otherwise, bit 7 (and all other bits in location 1707) will be zero. This allows a program to "watch" location 1707 and determine when the timer has timed out.

c. Reading the count in the timer

If the timer has not counted past zero, reading location 1706 will provide the current timer count and disable the interrupt option; reading location 170E will provide the current timer count and enable the interrupt option. Thus the interrupt option can be changed while the timer is counting down.

If the timer has counted past zero, reading either memory location 1706 or 170E will restore the divide ratio to its previously programmed value, disable the interrupt option and leave the timer with its current count (not the count originally written to the timer). Because the timer never stops counting, the timer will continue to decrement, pass zero, set the divide rate to 1, and continue to count down at the clock frequency, unless new information is written to the timer.

d. Using the interrupt option

In order to use the interrupt option described above, line PB7 (application connector, pin 15) should be connected to either the $\overline{\text{IRQ}}$ (Expansion Connector, pin 4) or $\overline{\text{NMI}}$ (Expansion Connector, pin 6) pin depending on the desired interrupt function. PB7 should be programmed as in input line (it's normal state after a RESET).

> NOTE: If the programmer desires to use PB7 as a normal I/O line, the programmer is responsible for disabling the timer interrupt option (by writing or reading address 1706) so that it does not interfere with normal operation of PB7. Also, PB7 was designed to be wire-ORed with other possible interrupt sources; if this is not desired, a 5.1K resistor should be used as a pull-up from PB7 to +5v. (The pull-up should NOT be used if PB7 is connected to $\overline{\text{NMI}}$ or $\overline{\text{IRQ}}$.)

APPENDIX I

KIM-1 PROGRAM LISTINGS

```
CARD # LOC      CODE         CARD
                          ;              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   333333
                          ;                       0    0   0    0        3
                          ;              ------   0    0   0    0        3
                          ;              ------   0    0   0    0   333333
                          ;              ------   0    0   0    0        3
                          ;                       0    0   0    0        3
                          ;                       000000   000000   333333
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;      COPYRIGHT
                          ;      MOS TECHNOLOGY, INC
                          ;      DATE OCT 18 1975 REV D
                          ;
                          ;
                          ;
                          ;      6530-003 IS AN AUDIO CASSETT TAPE
                          ;      RECORDER ENTENSION OF THE BASIC
                          ;      KIM MONITOR
                          ;
                          ;      IT FEATURES TWO BASIC ROUTINES
                          ;      LOADT-LOAD MEM FROM AUDIO TAPE
                          ;      DUMPT-STOR MEM ONTO AUDIO TAPE
                          ;
                          ;      LOADT
                          ;        ID=00     IGNORE ID
                          ;        ID=FF     IGN. ID USE SA FOR START ADDR
                          ;        ID=01-FE  IGN.ID USE ADDR ON TAPE
                          ;
                          ;      DUMPT
                          ;        ID=00     SHOULD NOT BE USED
                          ;        ID=FF     SHOULD NOT BE USED
                          ;        ID=01-FE  NORMAL ID RANGE
                          ;        SAL       LSB STARTING ADDRESS
                          ;        SAH       MSB
                          ;        EAL       LSB ENDING ADDRESS
                          ;        EAH       MSB
                          ;
```

```
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              YREG     *=*+1 Y INDEX
       00F5              XREG     *=*+1 X 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
```

```
CARD # LOC      CODE        CARD
  106  17F8             EAH     *=*+1               HI ENDING ADDRESS
  107  17F9             ID      *=*+1
  108                   ;
  109                   ;       INTERRUPT VECTORS
  110                   ;
  111  17FA             NMIV    *=*+2               STOP VECTOR (STOP=1C00)
  112  17FC             RSTV    *=*+2               RST VECTOR
  113  17FE             IRQV    *=*+2               IRQ VECTOR (BRK= 1C00)
  114                   ;
```

```
LOC      CODE        CARD
1800                        *=$1800
                     ;
                     ;      INIT VOLATILE EXECUTION BLOCK
                     ;      DUMP MEM TO TAPE
                     ;
1800  A9 AD          DUMPT  LDA   #$AD        LOAD ABSOLUTE INST
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
                     ;
```

```
LOC    CODE        CARD
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
                   ;
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
```

```
CARD # LOC    CODE        CARD
       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
       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
```



```
CARD # LOC    CODE        CARD
       192D   85 FB              STA    POINTH
       192F   4C 4F 1C           JMP    START
                          ;
```

```
CARD #  LOC    CODE       CARD
                          ;
                          ;       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
        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
```

```
CARD # LOC     CODE        CARD
       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
```

```
  LOC     CODE          CARD
  19CC    A9 C3                LDA    #195
  19CE    8D 44 17             STA    CLK1T
  19D1    A9 A7                LDA    #$A7
  19D3    8D 42 17             STA    SBD          SET PB7=1
  19D6    2C 47 17    ZRO2     BIT    CLKRDI
  19D9    10 FB                BPL    ZRO2
  19DB    A9 C3                LDA    #195
  19DD    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
```

```
CARD # LOC    CODE       CARD
       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    #$08        READ 8 BITS
       1A29   20 41 1A   RDCHT1 JSR    RDBIT       GET NEXT DATA BIT
       1A2C   4E EA 17          LSR    SAVX+1      RIGHT SHIFT CHAR
       1A2F   0D EA 17          ORA    SAVX+1      OR IN SIGN BIT
       1A32   8D EA 17          STA    SAVX+1      REPLACE CHAR
       1A35   CA                DEX
       1A36   D0 F1             BNE    RDCHT1
                         ;
       1A38   AD EA 17          LDA    SAVX+1      MOVE CHAR INTO A
       1A3B   2A                ROL    A           SHIFT OFF PARITY
       1A3C   4A                LSR    A
       1A3D   AE EB 17          LDX    SAVX+2
       1A40   60                RTS
                         ;
                         ;      THIS SUB GETS ONE BIT FROM
                         ;      TAPE AND RETURNS IT IN SIGN OF A
                         ;
       1A41   2C 42 17   RDBIT  BIT    SBD         WAIT FOR END OF START BIT
       1A44   10 FB             BPL    RDBIT
       1A46   AD 46 17          LDA    CLKRDT      GET START BIT TIME
       1A49   A0 FF             LDY    #$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
```

```
LOC    CODE        CARD
                   ;
                   ;       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
                   ;
                   ;
                   ;       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
                   ;
```

```
CARD # LOC     CODE       CARD
  522                  ;
  523                  ;
  524                  ;
  525                  ;
  526                  ;              666666   555555   333333   000000
  527                  ;              6        5             3   0    0
  528                  ;              6        5             3   0    0
  529                  ;              666666   555555   333333   0    0
  530                  ;              6    6        5        3   0    0
  531                  ;              6    6        5        3   0    0
  532                  ;              666666   555555   333333   000000
  533                  ;
  534                  ;
  535                  ;
  536                  ;                       000000   000000   222222
  537                  ;                       0    0   0    0        2
  538                  ;              ------   0    0   0    0        2
  539                  ;              ------   0    0   0    0   222222
  540                  ;              ------   0    0   0    0   2
  541                  ;                       0    0   0    0   2
  542                  ;                       000000   000000   222222
  543                  ;
```

```
CARD # LOC      CODE        CARD
                      ;
                      ;
                      ;
                      ;               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
                      ;
                      ;
                      ;
                      ;
```

```
CARD # LOC     CODE       CARD
       1000                         *=$1000
                          ;
                          ;
       1000    85 F3      SAVE     STA    ACC        KIM ENTRY VIA STOP (NMI)
       1002    68                  PLA               OR BRK (IRQ)
       1003    85 F1               STA    PREG
       1005    68         SAVE1    PLA               KIM ENTRY VIA JSR  (A LOST)
       1006    85 EF               STA    PCL
       1008    85 FA               STA    POINTL
       100A    68                  PLA
       100B    85 F0               STA    PCH
       100D    85 FB               STA    POINTH
       100F    84 F4      SAVE2    STY    YREG
       1011    86 F5               STX    XREG
       1013    BA                  TSX
       1014    86 F2               STX    SPUSER
       1016    20 88 1E            JSR    INITS
       1019    4C 4F 10            JMP    START
                          ;
       101C    6C FA 17   NMIT     JMP    (NMIV)     NON-MASKABLE INTERRUPT TRAP
       101F    6C FE 17   IRQT     JMP    (IRQV)     INTERRUPT TRAP
                          ;
       1022    A2 FF      RST      LDX    #$FF       KIM ENTRY VIA RST
       1024    9A                  TXS
       1025    86 F2               STX    SPUSER
       1027    20 88 1E            JSR    INITS
                          ;
                          ;
       102A    A9 FF      DETCPS   LDA    #$FF       COUNT START BIT
       102C    8D F3 17            STA    CNTH30     ZERO CNTH30
       102F    A9 01               LDA    #$01       MASK HI ORDER BITS
       1031    2C 40 17   DET1     BIT    SAD        TEST
       1034    D0 19               BNE    START      KEYBD SSW TEST
       1036    30 F9               BMI    DET1       START BIT TEST
       1038    A9 FC               LDA    #$FC
       103A    18         DET3     CLC               THIS LOOP COUNTS
       103B    69 01               ADC    #$01       THE START BIT TIME
       103D    90 03               BCC    DET2
       103F    EE F3 17            INC    CNTH30
       1042    AC 40 17   DET2     LDY    SAD        CHECK FOR END OF START BIT
       1045    10 F3               BPL    DET3
       1047    8D F2 17            STA    CNTL30
       104A    A2 08               LDX    #$08
       104C    20 6A 1E            JSR    GET5       GET REST OF THE CHAR
                          ;                          TEST CHAR HERE
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;        MAKE TTY/KB SELECTION
                          ;
```

```
CARD # LOC                CARD
   1C4F            START  JSR   INIT1
   1C5                    LDA   #$01
                          BIT   SAD
                          BNE   TTYKB
   1C     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
   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
```

```
CARD #  LOC    CODE       CARD
        1F60   A4 FC             LDY    TEMP        RESTORE Y
        1F62   60                RTS
                          ;
                          ;             SUB TO INCREMENT POINT
                          ;
        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
        1F86   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
```

```
LOC      CODE        CARD
                     ;       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
1FB4    C9 40      HEXNUM CMP   #$40        CONVERT TO HEX
1FB6    30 03             BMI   UPDATE
1FB8    18         HEXALP CLC
1FB9    69 09             ADC   #$09
1FBB    2A         UPDATE ROL   A
1FBC    2A                ROL   A
1FBD    2A                ROL   A
1FBE    2A                ROL   A
1FBF    A0 04             LDY   #$04        SHIFT INTO I/O BUFFER
1FC1    2A         UPDAT1 ROL   A
1FC2    26 F8             ROL   INL
1FC4    26 F9             ROL   INH
1FC6    88                DEY
1FC7    D0 F8             BNE   UPDAT1
1FC9    A9 00             LDA   #$00        A=0 IF HEX NUM
1FCB    60         UPDAT2 RTS
                     ;
1FCC    A5 F8      OPEN   LDA   INL         MOVE I/O BUFFER TO POINT
1FCE    85 FA             STA   POINTL
1FD0    A5 F9             LDA   INH         TRANSFER INH- POINTH
1FD2    85 FB             STA   POINTH
1FD4    60                RTS
                     ;
                     ;
                     ;       END OF SUBROUTINES
```

```
CARD # LOC      CODE        CARD
 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
```



```
CARD # LOC      CODE        CARD
 1206                     ;
 1207                     ;
 1208                     ;
 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	748	808	814	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	498	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	1085	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							
DUMPT4	1865	173	152							
DUMPV	1E01	873	867							
DUMP0	1D48	781	826							
DUMP1	1D4E	785	◆◆◆◆							

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES								
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	648	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	1168	****								
HEXOUT	196F	323	314	316							
HEXTA	1E4C	938	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						

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
LOADV	1E04	874	869
LOAD10	192B	271	268
LOAD11	18C2	223	218
LOAD12	190F	258	182
LOAD13	18FA	248	254
LOAD2	1D0E	746	751
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

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
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
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	1DA9	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	18AB	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
```

MOS TECHNOLOGY, INC.
VALLEY FORGE CORPORATE CENTER
950 RITTENHOUSE ROAD, NORRISTOWN, PA. 19401
TEL: (215) 666-7950 TWX:510/660/4033