Synertek

Systems Corporation

SYM-1 REFERENCE MANUAL

SYM-1 REFERENCE MANUAL

SSC Pub MAN-A-260006-D

Fifth Printing: January, 1981

Synertek Systems Corporation

P.O. BOX 552 SANTA CLARA, CALIFORNIA 95052 TEL. (408) 988-5689 TWX: 910-338-0135

SYM-1 REFERENCE MANUAL

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

* KIM is a product of MOS Technology, Inc.

CHAPTER 1

INTRODUCTION TO THE SYM COMPUTER

Whether you're a teacher or a student of computer science, a systems engineer or a hobbyist, you now own one of the most versatile and sophisticated single-board computers available today. The Synertek Systems SYM-1 is an ideal introduction to the expanding world of microprocessor technology as well as a powerful development tool for design of microcomputer-based systems. Fully assembled and thoroughly tested, the SYM-1 comes equipped with a 28-key dual-function keyboard for input and a 6-digit light emitting diode (LED) display for output. All that's needed to make your computer operational is a single 5-volt power supply.

Based on the popular and reliable 6502 Central Processing Unit (CPU), the SYM-1 is designed to permit flexible solutions to a wide range of application problems. A system monitor (SUPERMON) is stored in 4K bytes of Read Only Memory (ROM) furnished with the SYM-1 so you're free to concentrate on the application itself. But should you require customized system software, sockets are provided on the board for three additional ROM or Erasable PROM (EPROM) packages that can expand total ROM to 24K bytes. And by changing connections on the jumpers that have been designed for this purpose, the SYM-1 can be set up to respond to your own system software as soon as the power is turned on.

For working with data and programs, SYM-1 comes equipped with 1K of Random Access Memory (RAM), and sockets are available on the board for plug-in expansion up to 4K. Should additional memory be required for your application, an expansion port is provided which will allow additional ROM, PROM, RAM or I/O to be attached to the system up to the 65,536 maximum addressable limit for an 8-bit microprocessor.

While the keyboard and LED display included on the SYM-1 board will be sufficient for most users, other users may require the additional storage capability of audio cassette tape or the hard copy output of an RS-232 or a teletype terminal. Not only the serial interface, but also the hardware and software necessary for control of these devices is included on the SYM-1. Adding them to your system is simply a matter of properly wiring the appropriate connectors. Similarly, SYM-1 allows an oscilloscope to be added to the system to provide a unique 32-character display under software control.

And that's not all. A total of 51 active Input-Output (I/O) lines (expandable to 71 with the addition of a plug-in component) permit an almost endless variety of other peripheral devices to interface to the SYM-1, from floppy disk drives to full-ASCII keyboards and other computer systems.

Other key hardware and software features of SYM-1 include jumper-selectable and program-controlled write protection for selected areas of memory, four internal timers (expandable to six), four on-board buffers for direct control of high voltage or high current interfaces, and a debug facility that may be controlled either by a manual switch or by software. We could go on, but rather than merely list what the SYM-1 is capable of doing, let's move on to the rest of the manual and learn how to put it to work.

CHAPTER 2

HOW TO USE THE SYM REFERENCE MANUAL

This manual is designed both to help you get your SYM-1 running and to teach you to use it as fully as possible. Reading over the following chapter descriptions will give you an idea of how to proceed and where to look for help when you run into a problem. Although to get the most out of this manual you should read it thoroughly before attempting to operate your SYM-1, only Chapter 3 is essential before applying power and attempting simple operations.

You should read Chapter 3 before you even unpack your SYM-1. Following the handling instructions in that chapter will help insure that you do not inadvertently damage the microcomputer components. Chapter 3 also contains instructions for connecting the power supply, and a simple keyboard exercise to acquaint you with the SYM-1 and verify that the system is working properly. In addition, directions are provided for attaching an audio cassette recorder, teletype or any RS-232 compatible terminal to the system.

Chapter 4 provides you an overview of the hardware and software features of the SYM-1. The major Integrated Circuit (IC) devices are described, and the configuration of the various edge connectors is explained. Memory assignment is also discussed, as are the various hardware jumper options on SYM-1. A complete list of machine language and assembly language commands for the 6502 CPU is included in this chapter.

Chapter 5 provides complete operating instructions for the SYM-1. The color-coded keyboard layout is explained, the keys and their functions are defined, and you're shown how to form SYM monitor commands. Instructions for operating an audio cassette recorder, teletype terminal with paper tape unit, and RS-232 terminal are included with the appropriate monitor command descriptions. In addition, the features of the SYM-1 monitor are explained in detail.

Chapter 6 is where you'll learn to program the SYM-1 to handle your applications. We'll describe the program flow and assembly code for a small sample program and explain how to prepare it for entry to the SYM-1. Then we'll discuss how to execute it and how to find problems in it if it doesn't work the way you expected it to work. After you've completed this example program, you'll have a chance to try your hand at two more programs of increasing complexity.

Chapter 7 describes how to use an oscilloscope with your SYM-1 module to obtain a unique, 32-character display similar to that of a CRT. The hardware is present on your SYM-1 to allow this usage, and the software has been designed to allow you to write your own program to send characters to the oscilloscope. A sample program implementing this feature is discussed in the chapter.

Chapter 8 explains how to expand your SYM-1 system to include additional memory or peripheral devices. I/O techniques are also discussed, including how to configure an auxiliary expansion port.

Chapter 9 consists of a system flow chart and a discussion of advanced monitor and progamming techniques which will add flexibility and expandability to your SYM system. One of the unique things about the SYM-1 is its seemingly endless flexibility in software.

For example, you can create a sub-set of new monitor commands or an entirely new monitor by taking advantage of the way the system handles unrecognized commands. You can also make use of nearly all of the monitor as subroutines in your own programs, thus saving both programming time and memory space.

In addition to the chapters described above, several appendices located at the back of the manual include important service and other reference information. Appendix A explains what to do if your SYM-1 does not operate properly, becomes defective or requires service. Appendix B contains a complete parts list and a component layout diagram. Audio cassette tape formats are described in Appendix C, and the format for data stored on punched paper tape is outlined in Appendix D.

You will find that your SYM-1 will interface many devices designed to accompany the KIM computer. This compatability with KIM-related products is described in Appendix E. Appendix F explains how to create and use a sync tape for audio cassette operation. Appendices G and H contain Monitor Addenda and supplementary information relating to use of the SYM-1. Finally, Appendices I, J, K and L provide reference information on the SY6502, SY6522, SY6532 and SY2114 RAM IC devices.

The last item in the manual, which is not an appendix but an addendum, is a complete listing of the SYM-1 SUPERMON monitor program. Nothing is held back; you have the complete listing to allow you to use it any way you wish. Once you understand how the monitor works and the essentials of 6502 assembly language programming, this listing becomes an invaluable tool for implementing your own applications.

CHAPTER 3

PREPARING TO USE YOUR SYM COMPUTER

This chapter will take you, step-by-step, through the process of unpacking the SYM-1 and making it operational. After applying power and checking to see that the keyboard and display function properly, you will learn how to attach an audio cassette recorder, TTY, or CRT to the system.

3.1 PARTS CHECK

In addition to this manual, several other items are included with your microcomputer. Packed along with the SYM-1 microcomputer itself you should find a programming card containing a summary of 6502 instruction codes and SYM commands, a programming manual, a warranty card, which you should fill out and mail to Synertek Systems as soon as possible, and two edge connectors, one long and one short. Also included is a red plastic strip which serves as a faceplate over the lighted display. The terms of the warranty are explained on the warranty card. Also included with the computer is a packet of small rubber feet on which to mount your SYM-1 for table-top operation.

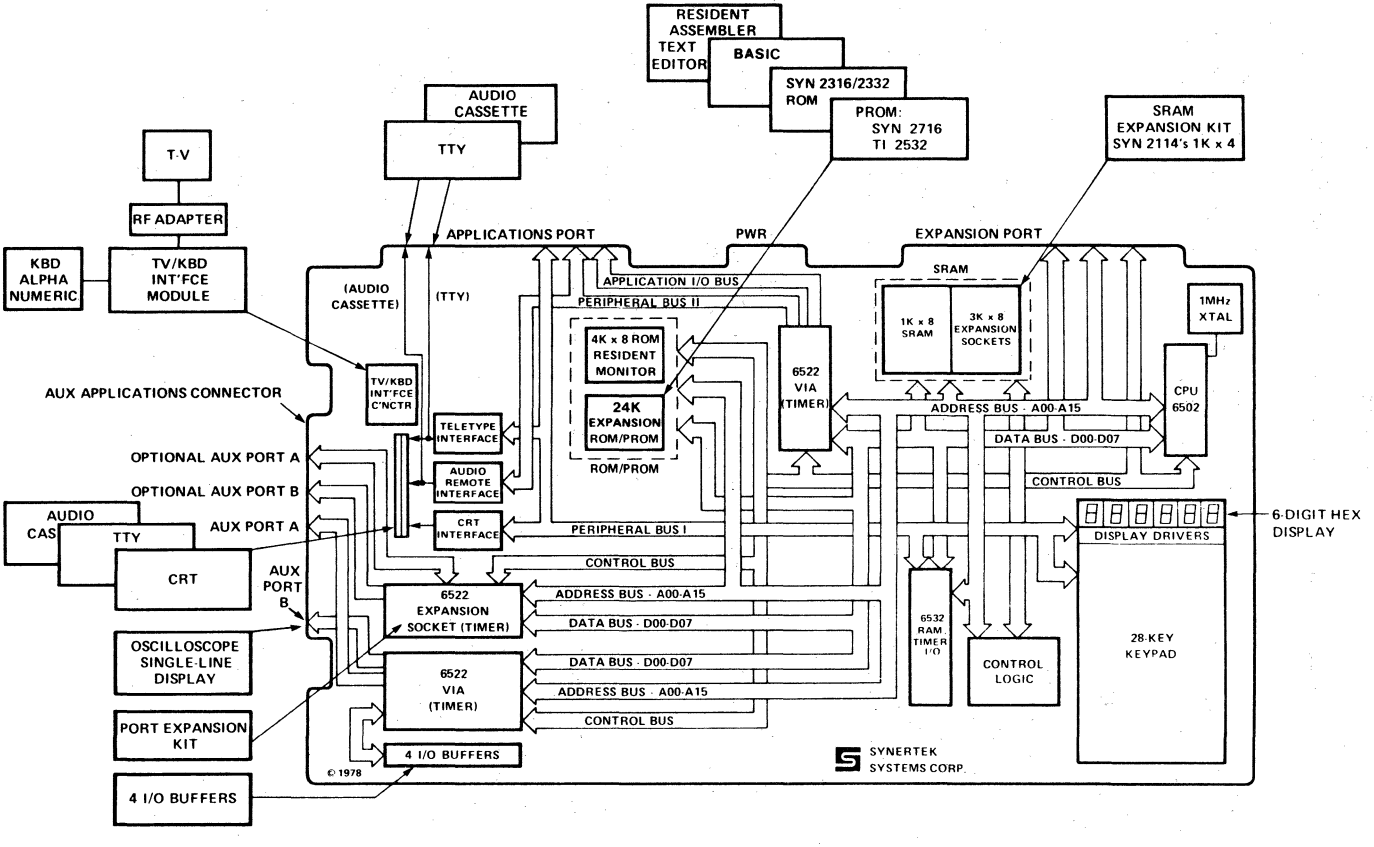
3.2 CAUTION ON MOS PARTS

The integrated circuits on your SYM-1 are implemented with Metal Oxide Silicon (MOS) technology and may be damaged or destroyed if accidentally exposed to high voltage levels. By observing a few simple precautions you can avoid a costly and disappointing mishap.

Static electricity is perhaps the least obvious, and thus most dangerous, source of voltage potential that can damage computer components. The SYM-1 is wrapped in special conductive material to protect it in shipping, and you should be careful to discharge any possible build-up of static electricity on your body before unpacking or handling the circuit board. Walking on a carpeted floor is especially liable to produce static electricity. Always touch a ground connection such as a metal window frame or an appliance with a three-pronged plug before handling your SYM-1, and avoid touching the pin connections on the back of the circuit board. Ungrounded or poorly grounded test equipment and soldering irons are other sources of potentially dangerous voltage levels. Make sure that all test equipment and soldering irons are properly grounded.

3.3 VISUAL CHECK

While observing the precautions described in section 3.2, take the SYM-1 from its box and remove the protective packing. Next, apply the small rubber mounting feet and place the SYM-1 on a flat surface with the keyboard facing you. Using Figure 3-1 you can identify the major system components and begin to familiarize yourself with the layout of the SYM-1 board. Chapter 4 describes the system in more detail, with appropriate schematics, but for now we're just concerned with powering-up and beginning operation.



3-1. FUNCTIONAL BLOCK DIAGRAM

3.4 RECOMMENDED POWER SUPPLIES

The SYM-1 microcomputer requires only the addition of a power supply to become fully operational. Any unit that supplies +5 Volts DC @ 1.5 amps and has adequate overload protection is acceptable. Synertek Systems does not recommend any particular make or model. Rather than buy an assembled power supply, you may want to build your own from one of the many kits available from hobby stores and mail order houses.

3.5 POWER SUPPLY CONNECTION

Now that you've obtained a 5-volt power supply, you're almost ready to power-up the SYM-1. Find the power supply edge connector (the smaller of the two edge connectors packed along with the microcomputer), and wire it as shown in Figure 3-2. Next, slide the connector onto the power connector pins located in the middle of the top edge of the board. Check to make sure that the wiring is correct and that the connector is properly oriented before attaching it to the board.

3.6 POWER-ON CHECK

Turn on the power supply. The red light to the left of the power connection should glow to indicate that power is reaching the board. The LED display above the keyboard should be completely blank, and a tone should be heard. Press the Carriage Return (CR) key. You should again hear the audible tone that is emitted when power is turned on or a key depression is sensed, and the display should show "SY1.1 . .". Carriage Return (CR) is the key that "logs you on" to the computer when first powering up or after pressing Reset (RST). If your computer isn't responding properly, turn off the power supply. Remove the power connector from the board and make sure that all wires are connected to the proper locations and are securely attached, then repeat the power-up procedure.

If after you recheck and repeat the power-up procedure, your SYM-1 does not respond as described above, refer to Appendix A for information on returning the unit for service.

3.7 KEYBOARD EXERCISE

Now that your SYM-1 is operational, let's try a small program to verify that the system is functioning properly. The program will add together two 8-bit binary numbers and store the result. As you enter the program, addresses and data will appear on the LED display as hexadecimal digits. Addresses are 16 bits long and thus will be represented by four hexadecimal digits, while data bytes are 8 bits long and will appear as two hex digits. Before entering the program, you may want to review the following listing of assembler code for the test program. The process of converting assembler code to machine language will be explained in Chapter 6.

```
                                MONITR  =  $8000
                                VALUE1  =  $0200
                                VALUE2  =  $0201
                                RESULT  =  $0202
                                   *    =  $0203
0203  18                        START      CLC
0204  D8                                   CLD
0205  AD 00 02                             LDA   VALUE1
0208  6D 01 02                             ADC   VALUE2
020B  8D 02 02                             STA   RESULT
020E  4C 00 80                             JMP   MONITR
                                           END
```

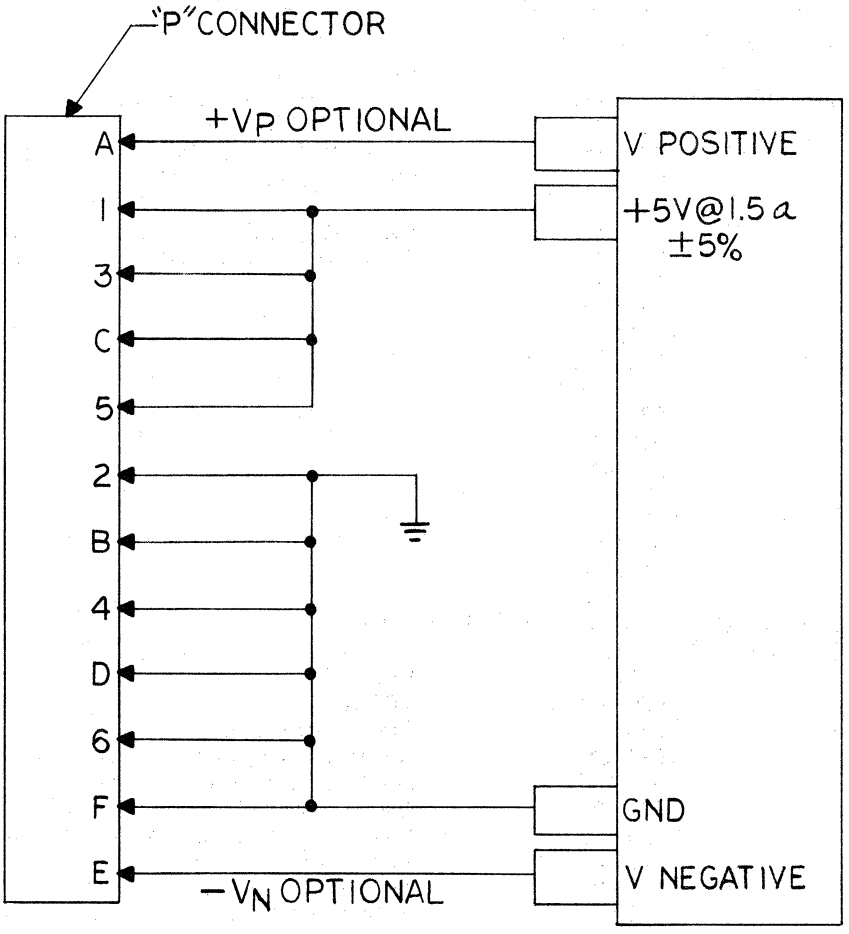


Figure 3-2. POWER SUPPLY CONNECTIONS

Now enter the program by following the steps listed below. Asterisks indicate the displayed data contained in the identified locations. Simulated key tops stand for function keys (e.g., (CR) for carriage return) The period displayed at the end of each entry sequence is SUPERMON's standard prompt character. As each data byte is entered, the address will automatically increment.

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(RESET)		
(CR)	SY1.1..	Keyboard log-on
(MEM) 200 (CR)	0200.**.	Display contents of location 0200.
C1	0201.**.	Store C1 (Hex) in 0200, display next location.
05	0202.**.	Store 05 (Hex) in 0201, display contents of 0202.
00	0203.**.	Store 00 (Hex) in 0202, display 0203
Enter Program:		
18	0204.**.	Store 18 (Hex) in 0203, display 0204
D8	0205.**.	Store D8 (Hex) in 0204, display 0205
AD	0206.**.	.
00	0207.**.	.
02	0208.**.	.
6D	0209.**.	.
01	020A.**.	
02	020B.**.	
8D	020C.**.	
02	020D.**.	
02	020E.**.	
4C	020F.**.	
00	0210.**.	
80	0211.**.	
(CR)	211.**..	
Check to see that program is entered correctly:		
(MEM) 200 (CR)	0200.C1.	VALUE1
	0201.05.	VALUE2
	0202.00.	RESULT
	0203.18.	Clear carry flag
	0204.D8.	Set status register for binary add
	0205.AD.	Load VALUE1 into accumulator
	0206.00.	Address of VALUE1, low order byte
	0207.02.	Address of VALUE1, high order byte
	0208.6D.	Add VALUE2 to accumulator
	0209.01.	Address of VALUE2, low order byte
	020A.02.	Address of VALUE2, high order byte
	020B.8D.	Store accumulator
	020C.02.	Address of RESULT, low order byte
	020D.02.	Address of RESULT, high order byte
	020E.4C.	JUMP to monitor
	020F.00.	Address of monitor, low order byte
	0210.80.	Address of monitor, high order byte
(CR)	210.80..	Exit from memory display and modify mode

Your program is now entered and ready to execute. The two numbers you will add together, C1 (Hex) and 05 (Hex), are stored in locations 0200 and 0201 respectively. The result will be stored in location 0202. The two digit hex codes you entered in

succeeding memory locations are the addresses, operands, and 6502 instruction codes necessary to add together two 8-bit binary numbers and return to the monitor program. To execute the program and display the result, perform the following steps:

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(GO) 203 (CR)	g 203 .	Execute program starting at location 0203
(MEM) 202 (CR)	0202.C6	Check result stored in location 0202
(CR)	202.C6. .	Exit from memory display and modify mode

Although this is a simple problem, it demonstrates the basic procedures for entering and executing a program on the SYM-1 as well as verifying that the system is operating properly.

3.8 ATTACHING AN AUDIO CASSETTE RECORDER

The program you entered in section 3.7 will remain stored in RAM memory only as long as the power remains on. As soon as the power is turned off, RAM data is lost, so to reuse the program you would have to enter it again from the keyboard. In order to provide you with a way to permanently store data and programs, SYM-1 is equipped with the hardware and software logic necessary to "talk to" an audio cassette recorder.

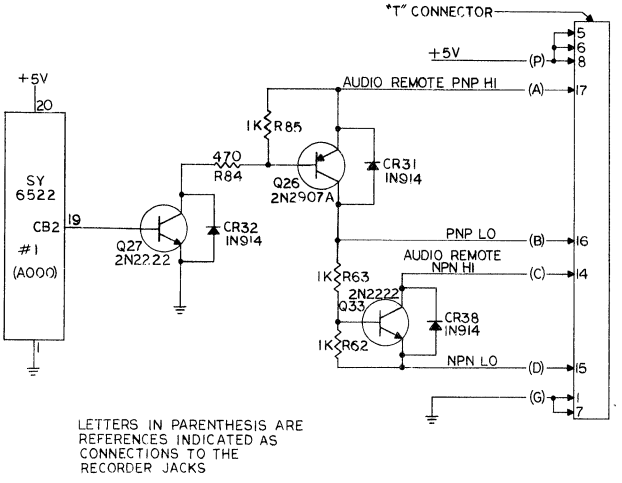
Since SYM-1 audio cassette operation involves high data transfer rates (185 bytes per second for HIGH-SPEED format), you should use a good quality recorder to ensure reliable performance. The unit should be equipped with an earphone jack for output, a microphone for input, a remote jack for remote control of the motor (optional), and standard controls for Play, Record, Rewind, and Stop. An additional feature that is useful but not essential is a tape counter. By keeping a record of counter values you can locate any program of data block manually without having to search the tape under program control at Play speed.

SYM-1 is designed to allow the cassette unit to be attached to either the Applications (A) or the Terminal (T) connector (requires a DB25 connector; see section 3.12). Refer to Figure 3-1 for the board location of these two connectors. Figure 4-3 shows how the Applications (A) edge connector should be wired for the cassette unit. The Terminal (T) connector should be wired as shown in Figure 4-3 if the unit is to be attached to the T connector. Keep the leads as short as possible and avoid running them near sources of electrical interference such as AC power cords. Always use the ground connection at the connector and do not ground directly to the power supply.

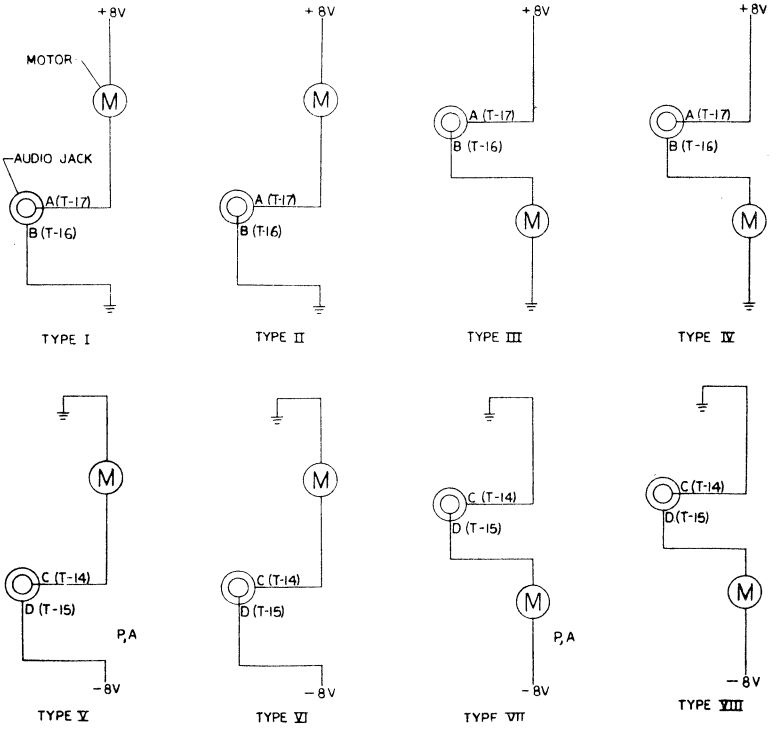
The remote control circuitry on the SYM-1 card allows a variety of cassette recorders to be used under software control. However, before you connect your remote control you must determine which type of connection is necessary for your particular recorder. Figure 3-3 illustrates the SYM-1 circuitry and eight different ways to hook it up. The following procedure can be used to determine which connection is necessary for your recorder:

1. Insert the remote control cable into your recorder. Install a tape in the unit.
2. Press play. The tape should not move. If it does, check the cable.
3. Measure the voltage at the center tip of the open end of the cable. (See Figure 3-4. Use ground reference from the EAR plug.) Record this as

AUDIO CASSETTE SYM REMOTE CONTROL CONNECTION



AUDIO CASSETTE RECORDER JACKS REMOTE CONTROL CONNECTIONS



TYPE V THRU VIII
REQUIRES P (T-8) TO BE
CONNECTED TO A (T-17)

Figure 3-3. REMOTE CONTROL TYPES AND CONNECTIONS

Table 3-1. AUDIO CASSETTE REMOTE CONTROL TYPE DETERMINATION

READING B (shield voltage)	READING A (center tip voltage)		
	-6v to -8v	GND	+6v to +8v
-6v to -8v		READING C GND Type VIII -8v Type V	
GND	READING C GND Type VII -8v Type VI		READING C GND Type I +8v Type IV
+6v to +8v		READING C GND Type II +8v Type III	

Reading C (shorted)

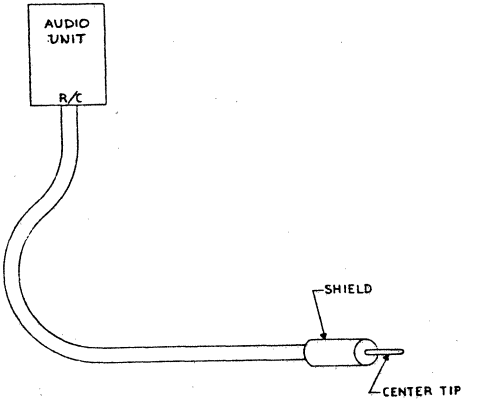


Figure 3-4. REMOTE CONTROL PLUG UNIT

Reading A. Typically this will be either +6 to +8 volts, -6 to -8 volts, or ground.

4. Measure voltage at the shield of the open end of the cable. Record this as Reading B. The same typical values stated in step 3 will apply. Readings A and B should not be the same.
5. Using a wire jumper, short the shield and center tip together. Your tape should now move. Measure the voltage at the center tip (do not remove the short). Record this as Reading C.
6. If your tape moves in step 2 or your tape does not move in step 5, check your cable for opens or shorts.
7. Use Table 3-1 to determine which type of connections to make for your recorder.
8. After you have found the proper category for your recorder, Figure 3-3 illustrates which connections to make.

3.9 SAVE AND LOAD EXERCISE

To check cassette unit operation, we'll "Save" on tape the program presented in Section 3.7, then load the program back into RAM. But before beginning tape operations, we must set the volume and tone controls on the recorder to the correct position. This is accomplished by creating and using a "sync" tape as described in Appendix F. Follow those procedures now, keeping in mind that we will save the program, and thus will also load it back into RAM, in HIGH-SPEED format.

After adjusting you recorder, enter the program from the keyboard as you did before. Insert a tape into the recorder. If your unit is equipped with remote control, place it in Record mode. Since the motor for the cassette is under software control, the tape will not advance. If your unit does not have remote control, do not place the unit in Record mode until just before pressing (CR) while entering the save command shown below including the carriage return, before placing the unit in Play Mode.

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(SAV 2) 3 (-) 200 (-) 210 (CR)	0-210.	Save locations 0200 to 0210 in a record with ID=03, in HIGH-SPEED format.

When recording starts the display will go blank. When recording is completed the display will re-light. All this should take approximately eight seconds. If your unit does not have remote control, stop the tape manually after the display re-lights.

Now rewind the tape to the starting point. If your unit has remote control, you will have to pull out the Remote jack from the recorder or keep your finger on the RST key.

To destroy the program stored in RAM, turn off system power, then turn it on again.

Log back onto the computer by pressing (CR), then place the cassette unit in Play mode if it is equipped with remote control. If you are operating the controls manually, you should first enter the load command shown below. Depress "PLAY" control on the cassette recorder before pressing (CR).

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(LD 2) 3 (CR)	..L3	Load HIGH-SPEED tape record with ID=03 into memory.

This command directs the SYM-1 to search for the tape record with ID=03. While the SYM-1 is searching, an "S." will be displayed. When reading begins, the AUDIO indicator LED will glow and the cassette ID will be displayed on the left digit (see Appendix M).

If you are operating the controls manually, turn the recorder OFF. Under remote control, the motor will stop automatically.

Now follow the instructions in Section 3.7 for executing the program. The result of the addition, C6 (Hex), should appear on the display. If the "S." did not disappear when reading in the program, or if the cassette otherwise did not respond as described above, check all wiring connections, verify the settings of the volume and tone controls and repeat the recording and playback procedures, making sure that each step is performed correctly. If after rechecking connections and repeating the procedure you are still unsuccessful, refer to Appendix A.

3.10 ATTACHING A TTY

To enable you to add a hard copy output device to your system, SYM-1 interfaces to a TTY terminal. Since the Teletype Model 33ASR is widely used and easily obtained, it will be used in the procedures and diagrams in this section. To interface other terminals, use the information given in this section as a general guide and consult the terminal instruction manual for different wiring and connection options.

Your TTY should be set for 20 mA current-loop operation. If it is not, follow the manufacturer's instructions for establishing this configuration. In addition, check to make sure that your TTY is set up to operate in full-duplex mode. You need not concern yourself with the TTY data transmission rate. SYM-1 assumes 110 bits-per-second (baud) for TTY terminals.

Just like an audio cassette recorder, a TTY may be attached to either the Applications (A) connector or using a DB25 (see section 3.12), to the Terminal (T) connector connection (See Figure 3-1). Figure 3-5A shows how the edge connector should be wired if the TTY will be attached to the "A" connector. Figure 3-5B shows the proper connections if it will be attached to the "T" connector. Wire the edge connector as appropriate for your application, then slide it into position. To "log on" to the terminal enter the following command at the on-board keyboard (not on the TTY keyboard).

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(RESET)		
(CR)	SY1.1 . .	Log-on to keyboard
(SHIFT) (JUMP) 1 (CR)	blank	Log-on to TTY

The TTY should respond with a carriage return and the TTY prompt character, a period. If it does not, turn off the power and re-check your connections, then power-up again.

3.11 TERMINAL EXERCISE

After the TTY prints the prompting character (".") as shown on the first line of the chart below, perform the rest of the steps listed to become acquainted with TTY operation. You will be entering a portion of the program presented in Section 3.7.

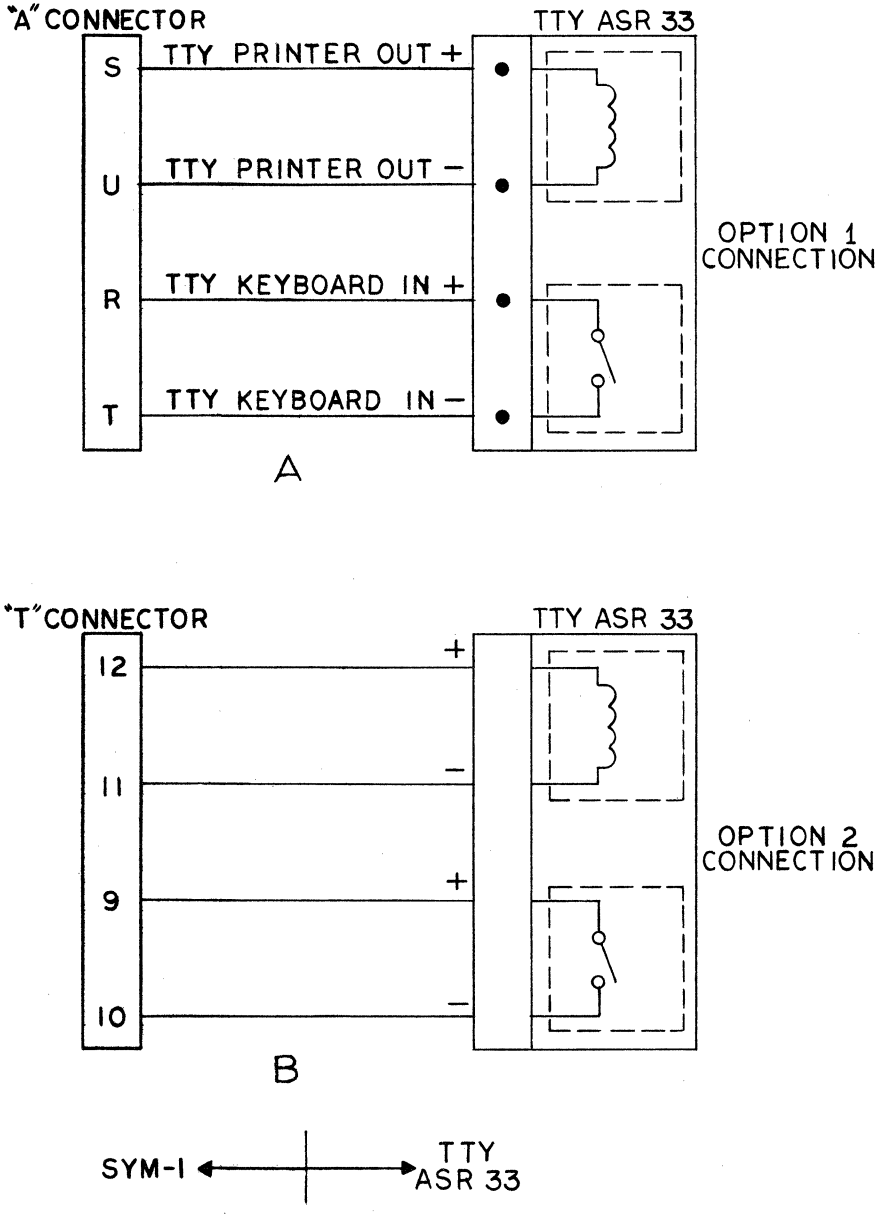


Figure 3-5. TTY I/O CONNECTIONS

YOU KEY IN	TTY PRINTS	EXPLANATION
	.	Prompt
M 200 (RETURN)	.M 200 0200,**,	Display contents of location 0200
C1	0200,**,C1 0201,**,	Store C1 (Hex) in 0200, display 0201
05	0201,**,05 0202,**,	Store 05 (Hex) in 0201, display 0202
(RETURN)	.	Return to monitor

3.12 ATTACHING A CRT

SYM-1 is equipped with an RS-232 interface to facilitate the use of such RS-232 devices as a full-ASCII keyboard and CRT display. Figure 3-6 shows how the proper DB25 connector, which may be easily obtained from an electronics supply house or computer hobby store, should be wired. The location of the interface on the SYM-1 board is shown in Figure 3-1. Some older units may need to be wired differently. Refer to the section on jumper options in Chapter 4.

3.13 CRT EXERCISE

Operating a CRT terminal, such as Synertek Systems' Keyboard Terminal Module, is very similar to operating a TTY. Names of keys and their functions may vary slightly depending on the device, so you should consult your CRT operating manual to find which keys correspond to the TTY keys used in the exercise in section 3.11. SYM-1 automatically adjusts to data transmission rates of 110, 300, 600, 1200, 2400, or 4800 baud for CRT operation. To set the baud rate, enter a "Q" on the CRT keyboard after powering-up (do not press any on-board keys). The CRT should respond with a ".", the terminal prompt character. Now repeat the exercise in Section 3.11 using the CRT keyboard.

In this chapter you have made your SYM-1 operational and learned how to attach several peripheral devices to the system. Let's move on to Chapter 4 and examine in detail the various features of SYM-1 hardware and software.

CRT I/O CONNECTIONS

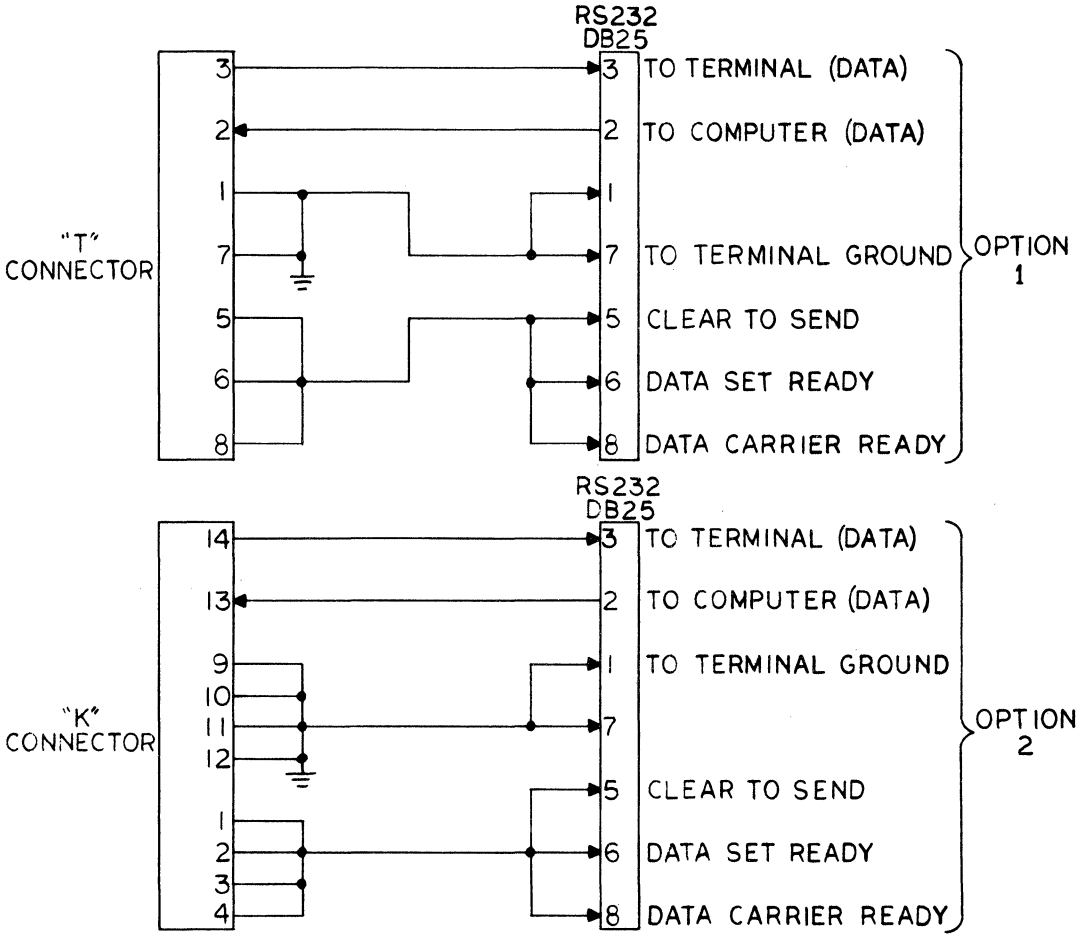


Figure 3-6. CRT I/O CONNECTIONS

CHAPTER 4

SYM-1 SYSTEM OVERVIEW

This chapter will describe your SYM-1 microcomputer system's hardware and software in sufficient detail to allow you to understand its theory of operation. Each integrated circuit (IC) component on the SYM-1 board is discussed and related to a functional block diagram. Each functional module is then discussed schematically and the I/O connectors are described. The system memory is then covered and the software is discussed briefly. Detailed data on the software itself is found in Chapter 5 of this manual.

4.1 HARDWARE DESCRIPTION

The SYM-1 microcomputer consists primarily of a 6502 CPU, one or more 6522 Versatile Interface Adapters (VIA), a 6532 Memory and I/O Controller and two types of memory involving any combination of several different components. Because of the flexibility of the memory structure, it is discussed in a separate section (4.2, below).

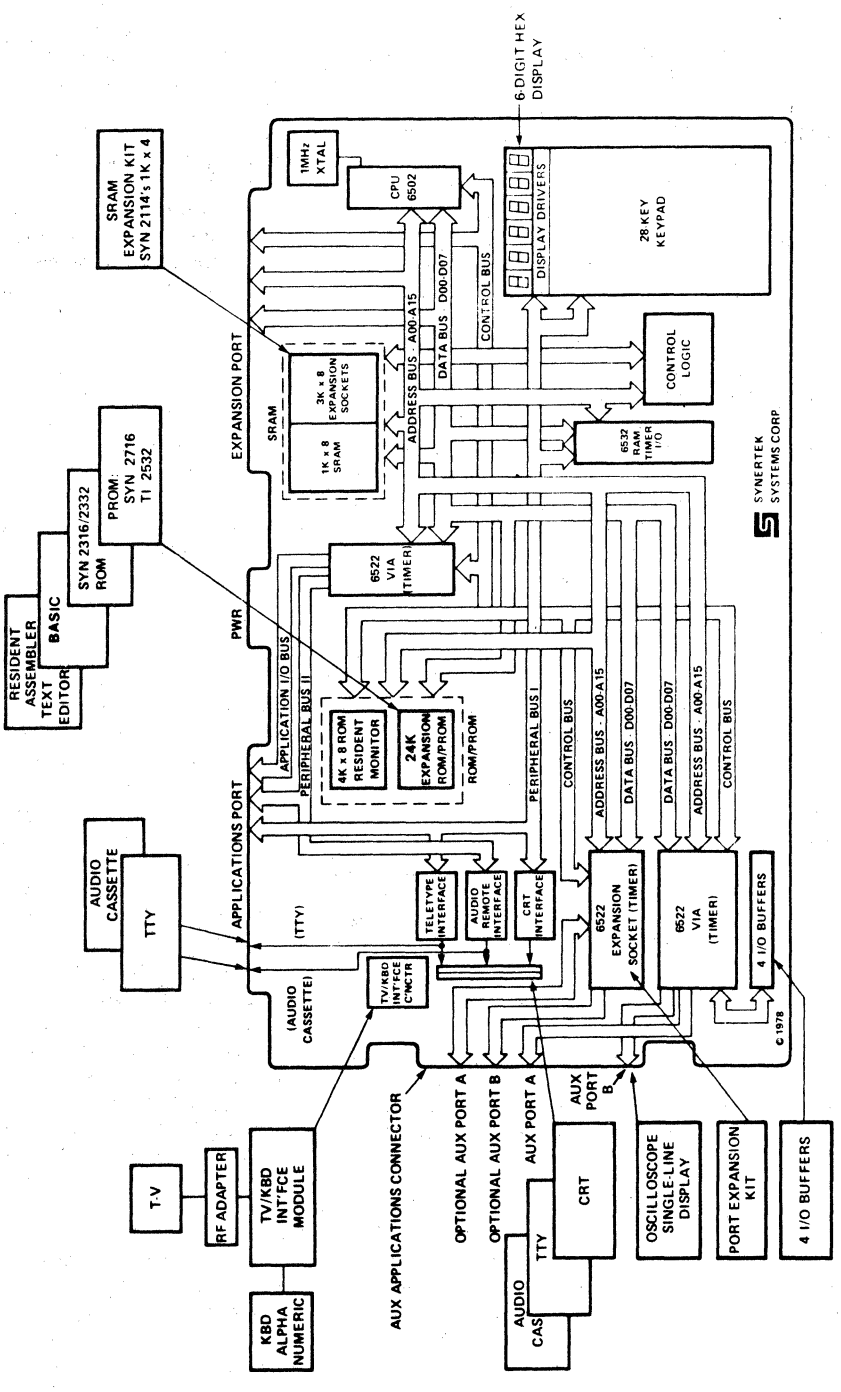
In any microcomputer system, all the components work together functionally as well as being physically interconnected. These connections are illustrated in Figure 4-1, a block diagram of the SYM-1 microcomputer system.

4.1.1 6502 CPU Description

The Central Processing Unit (CPU) of the SYM-1 microcomputer system is the 6502 microprocessor which is designed around a basic two-bus architecture--one full 16-bit address bus and an eight-bit data bus. Two types of interrupts are also available on the processor. Packaged in a 40-pin dual-in-line package, the 6502 offers a built-in oscillator and clock drivers. Additionally, the 6502 provides a synchronization signal which indicates when the processor is fetching an instruction (operation code) from program memory.

During the following discussion of the 6502, you should refer to the Data Sheets in this manual, which describe the pin connections for all three of the major types of devices present on the SYM-1 microprocessor system.

4.1.1.1 Bus Structure. The 6502 CPU is organized around two main busses, each of which consists of a separate set of parallel paths which can be used to transfer binary information between the components and devices in the SYM-1 system. The address bus transfers the address generated by the processor to the address inputs of the peripheral interface and memory devices (i.e., the 6522 and 6532 components). Note that in the Data Sheet for the 6502, the address lines originate at pins 9-20 and 22-25 of the 6502 CPU. These address lines go to pins 2-17 on the 6522 and/or to pins 2, 5-8, 10-15 and 34-40 on the 6532. Since the processor is almost always the only source of address generation in a system, an address bus is generally referred to as "unidirectional." That is the case with the SYM-1 microcomputer system. Since the address bus consists of 16 lines, the processor may read and write to a total of 65,536 bytes of storage (i.e., program memory words, RAM words, stack, I/O devices and other information), a condition which is normally referred to as a "64K memory capacity."



4-1. FUNCTIONAL BLOCK DIAGRAM

The other bus in the 6502 processor is called the data bus. It is an eight-bit bidirectional data path between the processor and the memory and interface devices. When data is moved from the processor to a memory location, the system performs a write; when the data is traveling from memory to the CPU, a read is being performed. Pins 26-33 on the 6502, 6522 and 6532 devices are all data lines connected to the data bus. The direction of the transfer of data between these pin connectors is determined by the output of the Read/Write (R/W, Pin 34) of the 6502. This line enables a write memory when it is "low" (when its voltage is below 0.4 VDC). Write is disabled and all data transfers will take place from memory to the CPU if the level is high (greater than 2.4 VDC).

One of the important aspects of the 6502 CPU is that it has two interrupt input lines available, Interrupt Request (labeled $\overline{IRQ}$ in the Data Sheet) and a Non-Maskable Interrupt (labelled $\overline{NMI}$).

Interrupt handling is one of the key aspects of microprocessor system design. Although the idea of interrupt handling is fairly simple, a complicating factor is the necessity for the processor to be able to handle multiple interrupts in order of priority (usually determined by the programmer) and not "losing track" of any of them in the process. These are concepts which you as a programmer-user of the SYM-1 will be concerned with only in advanced applications. The handling of user-generated interrupts is discussed elsewhere in this manual. If you do have occasion to alter pre-determined interrupt handling, it will be helpful for you to understand how the process works for the two types of interrupts in the 6502.

There are two main differences between the $\overline{IRQ}$ and $\overline{NMI}$ signals and their handling. First, $\overline{IRQ}$ will interrupt the CPU only if a specific flag--the Interrupt Disable Flag (I)--in the system's Processor Status Register is cleared, i.e., zero. If this flag is "set"--i.e., one--the $\overline{IRQ}$ is disabled until the flag is cleared. But an $\overline{NMI}$ request (as its name implies) always causes an interrupt, regardless of the status of the I-flag. The other main difference between the two types of interrupts is that the $\overline{IRQ}$ interrupt is "level sensitive." Any time the signal is less than 0.4 VDC and the Interrupt Disable flag is cleared, an interrupt will take place. In the case of $\overline{NMI}$, the interrupt is said to be "edge-sensitive" because it is dependent on a sequence of timing events. This interrupt will occur only if the signal goes "high" (i.e., exceeds 2.4 VDC) and then goes back to ground (less than 0.4 VDC). The interrupt occurs on the negative-going transition past 0.4 V.

The Data Sheet contains a summary of the 40 pins on the 6502 CPU and their function. Note that three of the pins--5, 35 and 36--are not connected on the 6502.

4.1.1.2 Summary. The 6502 CPU is a versatile processor. It was selected for your SYM-1 microprocessor system because of its overall functional characteristics, which facilitate its use in a wide variety of applications. Its role in the SYM-1 system will become clearer when we discuss programming and software in Section 4.3 and in Chapters 5 and 6.

4.1.2 6522 Description

The SY6522 Versatile Interface Adapter (VIA) is a highly flexible component used on the SYM-1 module to handle peripheral interfaces. Two of these devices are standard components on your SYM-1; a third may be added merely by plugging it into the socket (U28) provided. Control of the peripheral devices is handled primarily through the two eight-bit bi-directional ports. Each line of these ports can be programmed to act as

either an input or an output. Also, several of the peripheral I/O lines can be controlled directly from the two very powerful interval timers integrated into the chip. This results in the capability to 1) generate programmable frequencies, 2) count externally generated pulses, and 3) to time and monitor real time events.

A description of the pin designations on the SY6522 is contained in the Data Sheet enclosed with your SYM-1. It should be used in following the discussion of the operation of the component in the SYM-1 module which follows. The Memory Map of the SYM-1 module (Figure 4-10) will also be helpful during this discussion.

4.1.2.1 Processor Interface. Data transfers between the SY6522 and the CPU (6502) take place over the eight-bit data bus (DB0-DB7) only while the Phase Two Clock (Ø2) is high and the chip is selected (i.e., when CS1 is high and $\overline{CS2}$ is low). The direction of these data transfers is controlled by the Read/Write line (R/W). When this line is low, data will be transferred out of the processor into the selected 6522 register; when R/W is high and the chip is selected, data will be transferred out of the SY6522. The former operation is described as the write operation, the latter the read operation.

Four Register Select lines (RS0-RS3) are connected to the processor's address bus to allow the processor to select the internal SY6522 register which is to be accessed. There are 16 possible combinations of these four bits and each combination accesses a specific register. Because of the fact that the SY6522 is a programmable-addressable device, these RS line settings, in combination with the basic device address, form the specific register address shown in the 6522 Data Sheet.

Two other lines are used in the SY6522 interface to the 6502 processor. The Reset line ($\overline{RES}$) clears all internal registers to a logical zero state (except T1, T2 and SR), placing all peripheral lines in the input state. It also disables the timers, shift register and other on-chip functions and disables interrupting from the chip. The Interrupt Request line ($\overline{IRQ}$) generates a potential interrupt to the CPU when an internal interrupt flag is set and a corresponding interrupt enable bit is set to a logical "1." The resulting output signal is then "wire or'ed" with other similar signals in the system to determine when and whether to interrupt the processor.

4.1.2.2 Peripheral Interface. As we mentioned earlier, peripheral interface is handled largely over two eight-bit ports, with each of the 16 lines individually programmable to act as an input or output line. Port A consists of lines PA0-PA7 and Port B of lines PB0-PB7.

Three registers are used to access each of the eight-bit peripheral ports. Each port has a Data Direction Register (DDRA and DDRB), which is used in specifying whether the pins are to act as inputs or outputs. If a particular bit in the Data Direction Register is set to zero, the corresponding peripheral pin is acting as an input; if it is set to "1," the pin acts as an output point.

Each of the 16 peripheral pins is also controlled by a bit in the output register (ORA and ORB) and a similar bit in the Input Register (IRA and IRB). When the pin is programmed to act as an output, the voltage on the pin is controlled by the corresponding bit in the Output Register. A "1" in the appropriate Output Register causes the pin to go "high" (2.4 VDC or higher), and a zero causes it to go "low (0.4 VDC or lower).

Functionally, reading a peripheral port causes the contents of the appropriate Input Register to be transferred to the Data Bus.

The SY6522 has a number of sophisticated features which allow very positive control of data transfers between the processor and peripheral devices through the operation of "handshake" lines which involve the use of Peripheral Control Lines (CA1-CA2 and CB1-CB2). These operations are beyond the scope of this manual; if you are interested in further information, you should consult the data sheet enclosed.

4.1.3 6532 Description

Like the SY6522 described above, the SY6532 is used on the SYM-1 module to control peripheral interface. Only one SY6532 is furnished with your SYM-1 and no others are provided for.

From an operational standpoint, the SY6532 is quite similar to the SY6522. One key difference, particularly on your SYM-1 module, is the presence of a 128-byte x 8-bit RAM within the SY6532. This is the location referred to as "System RAM" in discussions of the software operation and in the Memory Map (Figure 4-10).

A description of the pin designations on the SY6532 is included in the enclosed Data Sheet. You will notice that, like the SY6522, the SY6532 contains 16 peripheral I/O pins divided into two eight-bit ports (lines PA0-PA7 and PB0-PB7). Each of these pins can be individually programmed to function in input or output mode. $\overline{IRQ}$ on the SYM-1 SY6532 is not connected.

The Address lines (A0-A6) are used with the RAM Select ($\overline{RS}$) line and the Chip Select lines (CS1 and $\overline{CS2}$) to address the SY6532. It is in this addressing that the SY6532 differs somewhat from the SY6522's on your SYM-1 module. To address the 128-byte RAM on the SY6532, CS1 must be high and $\overline{CS2}$ and $\overline{RS}$ must both be low. To address the I/O lines and the self-contained interval timer, CS1 and $\overline{RS}$ must be high and $\overline{CS2}$ must be low. In other words, CS1 is high and $\overline{CS2}$ is low to address the chip; $\overline{RS}$ is used to differentiate between addressing RAM and the I/O Interval Timer functions. Distinguishing between I/O lines and the Interval Timer is the function of Address Line 2 (A2), which is high to address the timer and low to address the I/O section. Again, the Memory Map in Figure 4-10 clarifies these operations since they are largely software-directed and address-dependent.

4.1.4 Functional Schematics

Understanding the electrical interfaces among the various components may be of some interest to you as you use and expand your SYM-1 microcomputer. The figures on the following pages include segmented schematics, where each figure provides an electronic overview of the interface between the CPU and its related component devices and peripherals.

Table 4-1 describes the contents of each figure in this group of schematic segments.

Table 4-1. INDEX OF SCHEMATIC SEGMENTS FIGURES 4-2 TO 4-9

Figure	Function/Segment Diagrammed
4-2	TTY and CRT Interface
4-3	Audio Cassette Interface
4-4	Audio Cassette Remote Control
4-5	I/O Buffer
4-6	Keyboard/Display
4-7	Control Section
4-8	Memory Section
4-9	Oscilloscope Output Driver

Table 4-2 provides, in summary form, a list of the connector points on the four SYM-1 connectors. This allows you to determine pin and connector configurations for various application options.

Table 4-2. CONNECTOR POINTS AND THEIR FUNCTIONS IN SYM-1

Key: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 Component Side
A B C D E F H J K L M N P R S T U V W X Y Z Solder Side

EXPANSION (E)

1	SYNC	A	AB0
2	RDY	B	AB1
3	Ø1	C	AB2
4	IRQ	D	AB3
5	RO	E	AB4
6	NMI	F	AB5
7	RES	H	AB6
8	DB7	J	AB7
9	DB6	K	AB8
10	DB5	L	AB9
11	DB4	M	AB10
12	DB3	N	AB11
13	DB2	P	AB12
14	DB1	R	AB13
15	DB0	S	AB14
16	18	T	AB15
17	DBOUT (1)	U	Ø2
18	POR	V	R/W
19	Unused	W	R/W
20	Unused	X	$\overline{\text{AUD TEST}}$
21	+5V	Y	Ø2
22	GND	Z	Ram-R/W

APPLICATION (A)

1	GND	A	+5V
2	APA3	B	$\overline{00}$
3	APA2	C	$\overline{04}$
4	APA1	D	$\overline{08}$
5	APA4	E	$\overline{0C}$
6	APA5	F	$\overline{10}$
7	APA6	H	$\overline{14}$
8	APA7	J	$\overline{1C}$
9	APB0	K	$\overline{18}$
10	APB1	L	Audio In
11	APB2	M	Audio Out (LO)
12	APB3	N	RCN-1 (1)
13	APB4	P	Audio Out (H1)
14	APA0	R	TTY KB RTN (+)
15	APB7	S	TTY PTR (+)
16	APB5	T	TTY KB RTN (-)
17	KB ROW O	U	TTY PTR (-)
18	KB COL F	V	KB ROW 3
19	KB COL B	W	KB COL G
20	KB COL E	X	KB ROW 2
21	KB COL A	Y	KB COL C
22	KB COL D	Z	KB ROW 1

AUXILIARY APPLICATION (AA)

1	GND	A	+5V
2	-VN	B	+VP
3	2 PA 1	C	2 PA 2
4	2 CA 2	D	2 PA 0
5	2 CB 2	E	2 CA 1
6	2 PB 7	F	2 CB 1
7	2 PB 5	H	2 PB 6
8	2 PB 3	J	2 PB 4
9	2 PB 1	K	2 PB 2
10	2 PA 7	L	2 PB 0
11	2 PA 5	M	2 PA 6
12	2 PA 3	N	2 PA 4
13	RES	P	3 CA 1
14	3 CB 1	R	SCOPE
15	3 PB 2	S	3 PB 3
16	3 PB 0	T	3 PB 1
17	3 PA 6	U	3 PA 7
18	3 PA 3	V	3 PA 0
19	3 PA 4	W	3 PA 1
20	3 PA 5	X	3 PA 2
21	3 PB 5 (B)	Y	3 PB 4 (B)
22	3 PB 7 (B)	Z	3 PB 6 (B)

(1) Jumper option
(B) Buffered

TABLE 4-2. CONNECTOR POINTS AND THEIR FUNCTIONS IN SYM-1 (Continued)

POWER (P)

1	+5V	A	+VP (optional)
2	GND	B	GND
3	+5V	C	+5V
4	GND	D	GND
5	+5V	E	-VN (optional)
6	GND	F	GND

TERMINAL (T)

1	GND
2	RS-232 IN
3	RS-232 OUT
4	N.C.
5	+5V
6	+5V
7	GND
8	+5V
9	TTY Keyboard IN +
10	TTY Keyboard IN -
11	TTY Printer OUT -
12	TTY Printer OUT +
13	N.C.
14	Audio Remote NPN HI
15	Audio Remote NPN LO
16	Audio Remote PNP LO
17	Audio Remote PNP HI
18	Audio IN
19	Audio GND
20	N.C.
21	Audio Out (HI)
22	N.C.
23	Audio Out (LO)
24	N.C.
25	Audio GND

KEYBOARD (K)

1	+5V
2	+5V
3	+5V
4	+5V
5	+VP
6	+VP
7	-VN
8	-VN
9	GND
10	GND
11	GND
12	GND
13	RS-232 IN
14	RS-232 OUT

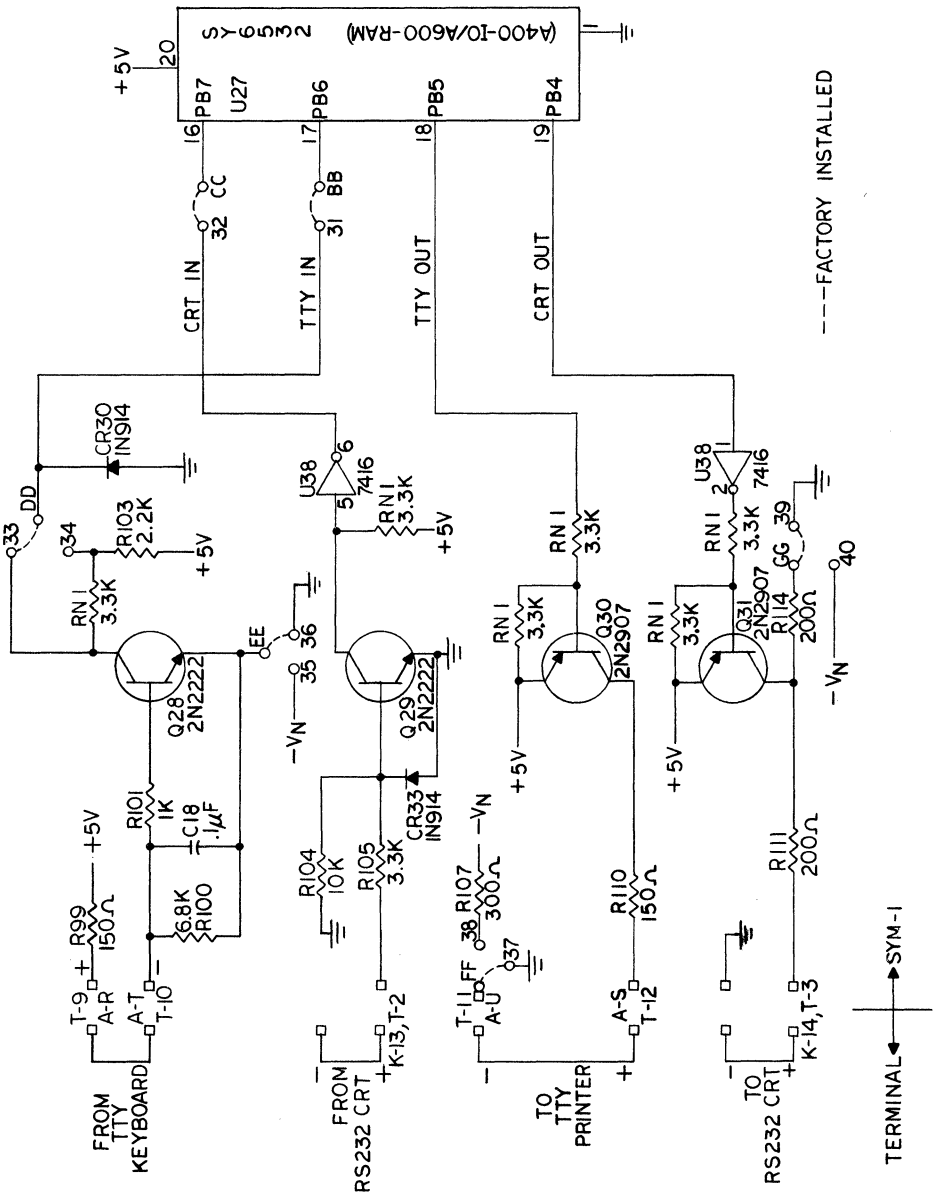


Figure 4-2. TTY/CRT INTERFACE SCHEMATIC

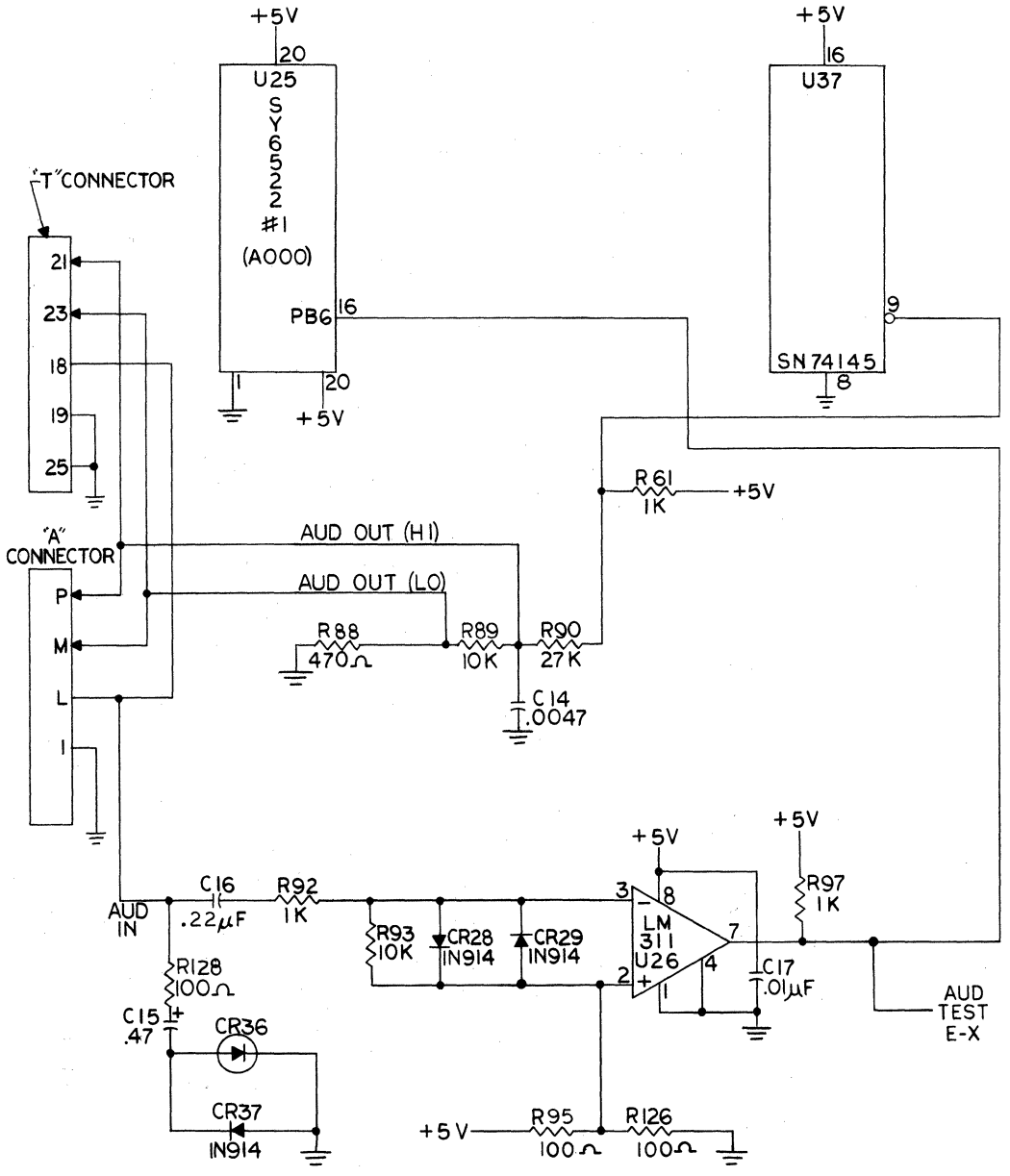


Figure 4-3. AUDIO CASSETTE INTERFACE SCHEMATIC

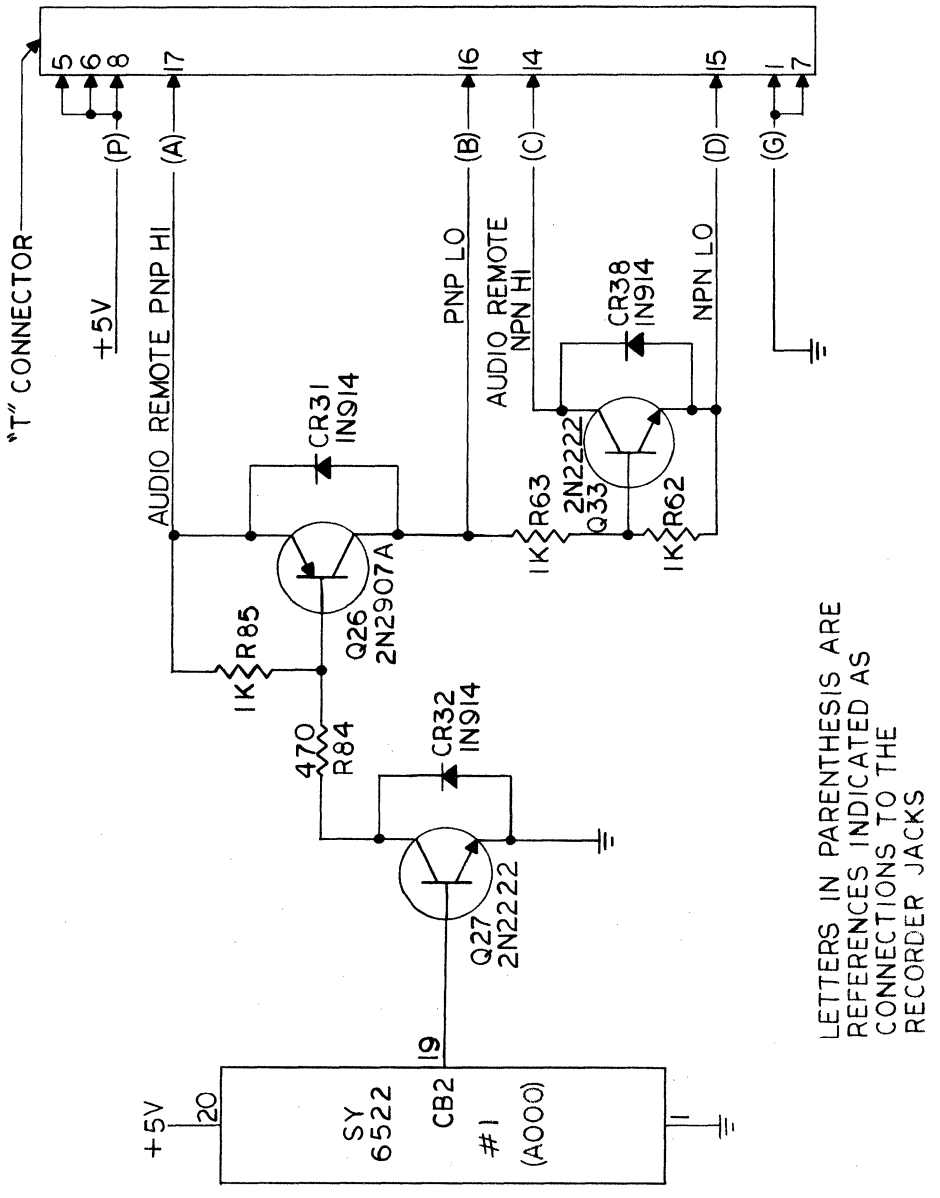
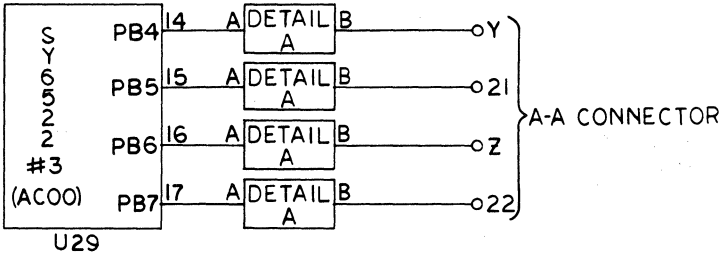


Figure 4-4. AUDIO CASSETTE REMOTE CONTROL

I/O BUFFERS



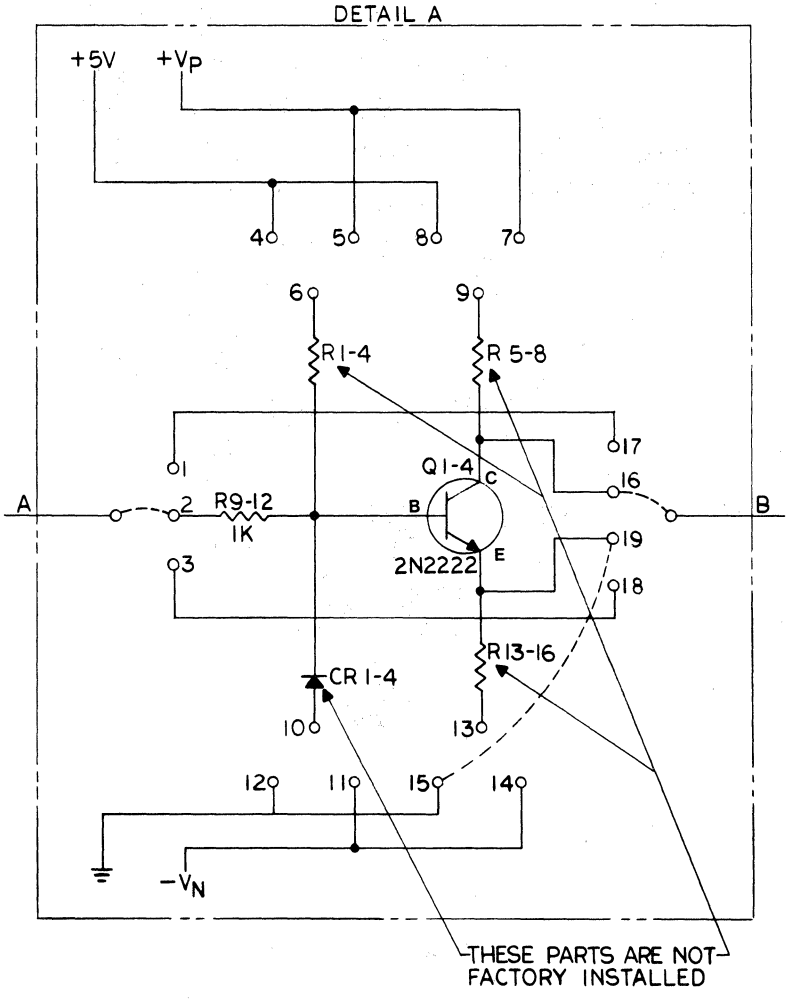


Figure 4-5. I/O BUFFERS SCHEMATIC

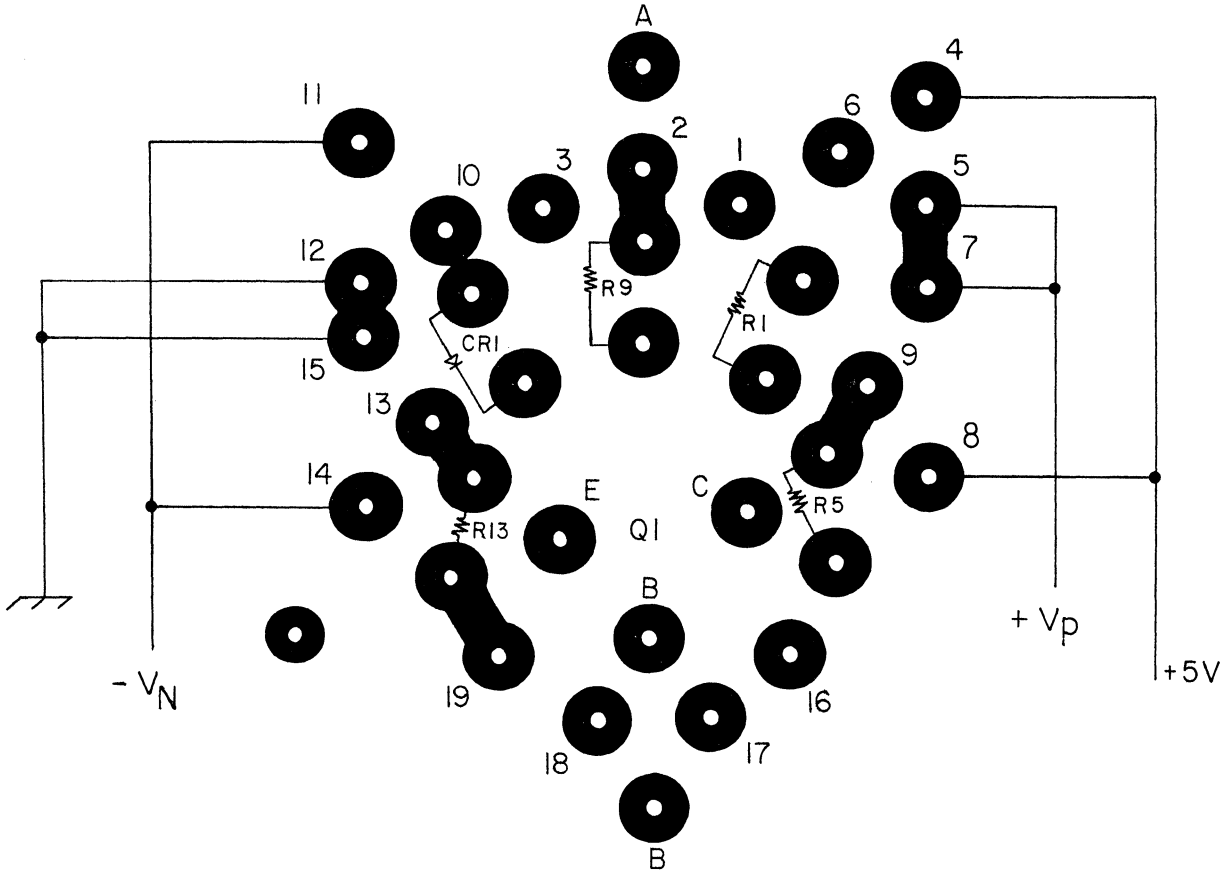


Figure 4-5a. I/O BUFFERS, PC LAYOUT BLOW-UP

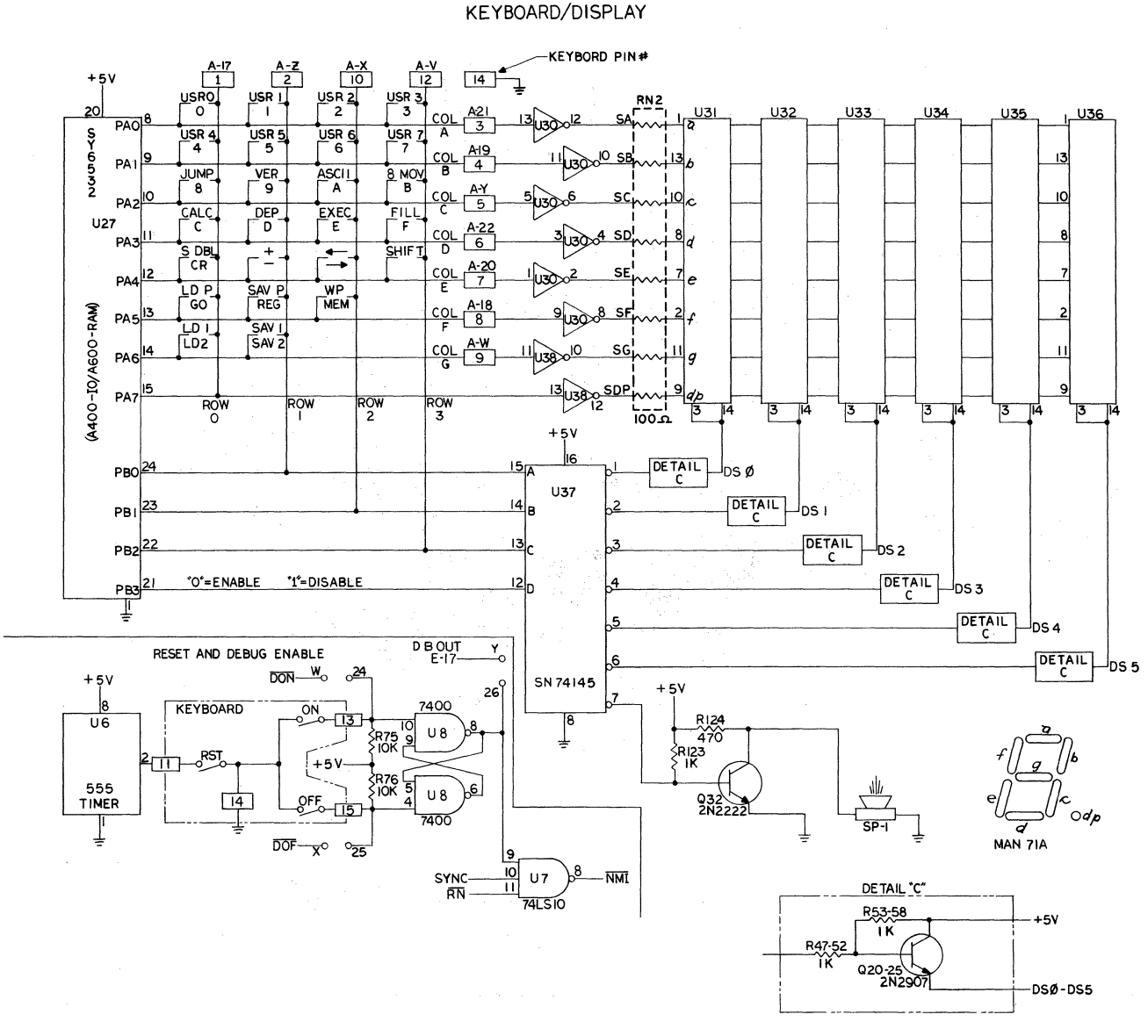


Figure 4-6. KEYBOARD/DISPLAY SCHEMATIC

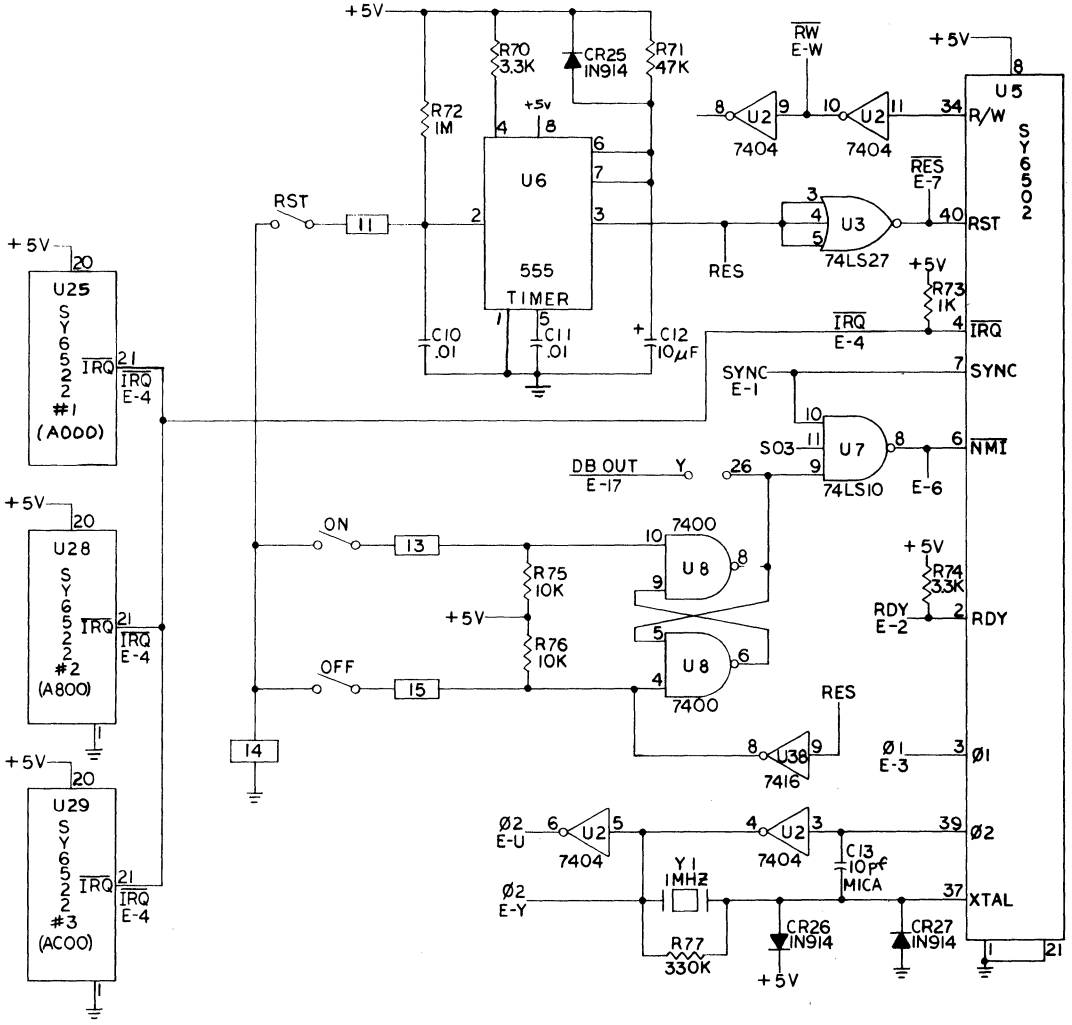


Figure 4-7. CONTROL SECTION SCHEMATIC

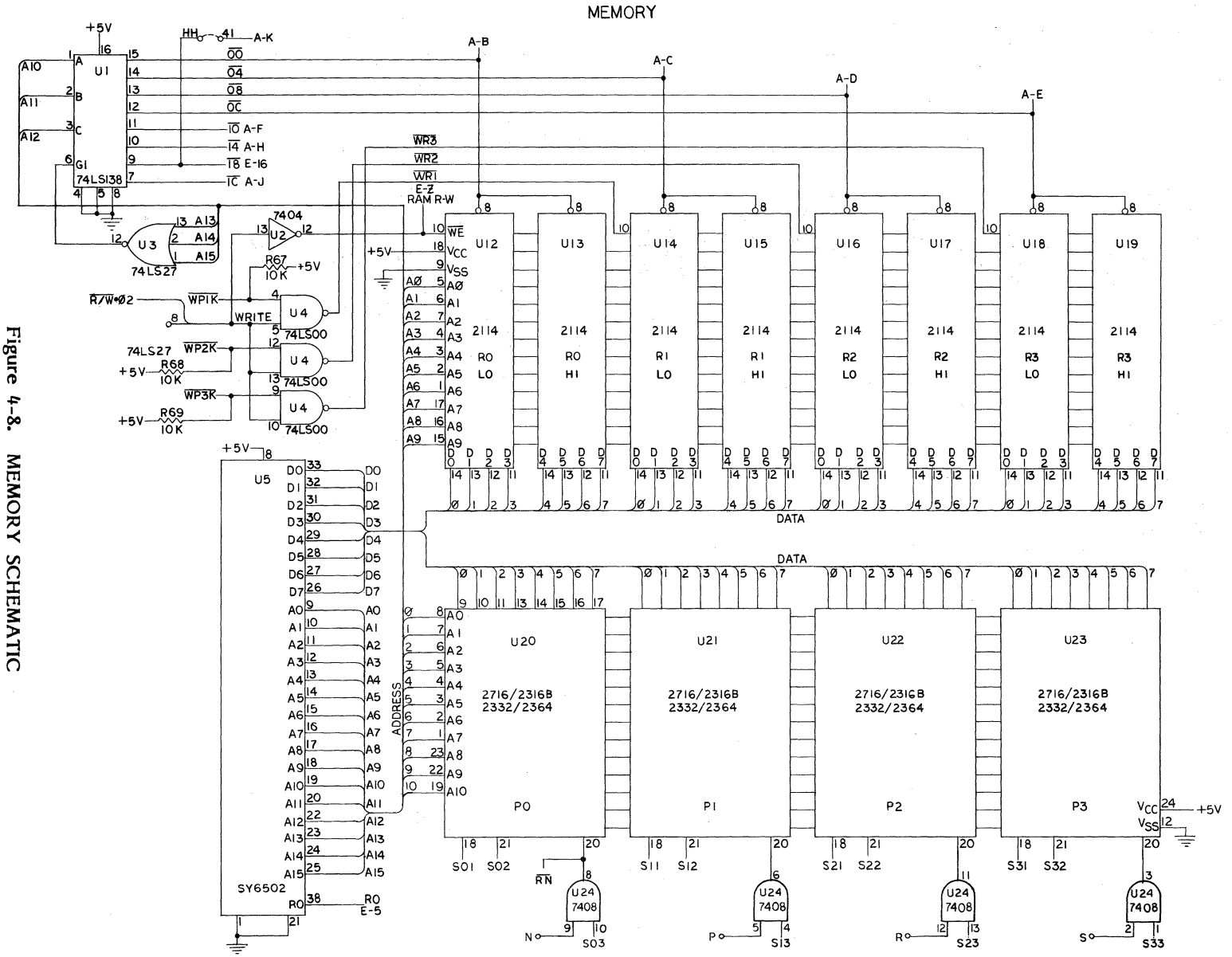


Figure 4-8. MEMORY SCHEMATIC

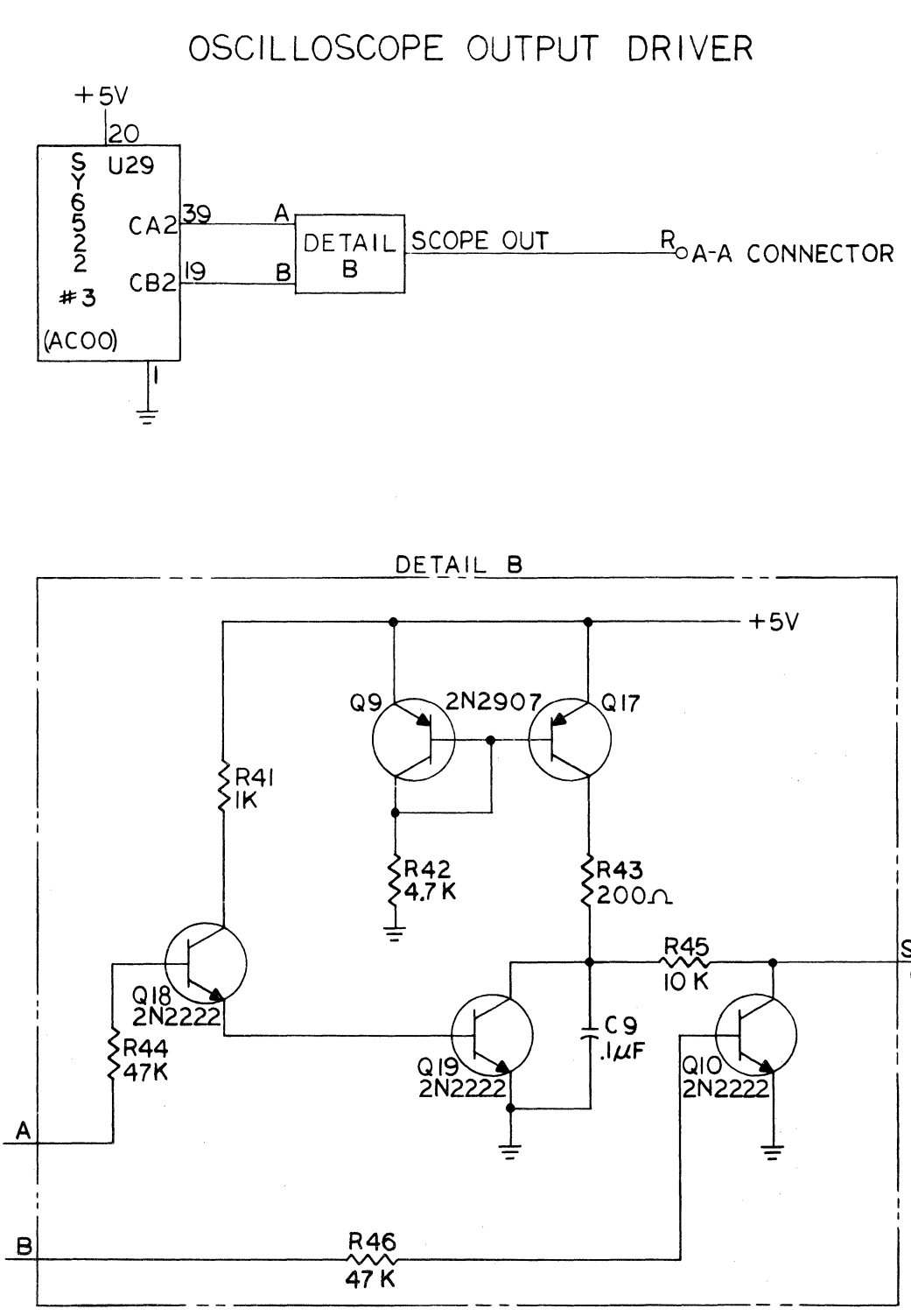


Figure 4-9. OSCILLOSCOPE OUTPUT DRIVER SCHEMATIC

4.2 MEMORY ALLOCATION

This section describes the standard memory allocation in your SYM-1 microcomputer system. It makes extensive use of the detailed Memory Map contained in Figure 4-10. Also described in this section is the technique by which ROM and RAM addressing and usage may be altered by using an array of on-board jumpers which allow you to modify and expand your SYM-1 memory. Expanding RAM memory using off-board components is taken up briefly in Section 4.2.3, although a detailed discussion of this is reserved for Chapter 8, "System Expansion".

4.2.1 Standard Memory Allocation

Figure 4-10 is a map of the standard memory allocation in your SYM-1 microcomputer. Provided with your system are 1K of on-board RAM, extending from location 0000 to 03FF in the Memory Map. Note that the top-most eight bytes (locations 00F8 to 00FF) in Page Zero of this 1K block are reserved for use by the system and should not be used by your programs. The remainder of Page Zero is largely similar to the rest of the RAM provided, but it also has some special significance for addressing which will become clearer in Section 4.3. Locations 0100-01FF in the 1K memory block furnished with your system are reserved for stack usage. Your programs may use this area, but you should use it for normal stack operations incidental to operating your programs. Locations 0200-03FF are general-use RAM for your program and data storage.

In addition to the 1K of on-board RAM furnished with your system, sockets are provided for 3K of plug-in RAM, allowing you to have 4K of on-board RAM memory. These sockets occupy memory locations 0400-0FFF.

The SUPERMON monitor resides in ROM at memory locations 8000-8FFF. (As you know, the SY6502 CPU addresses all memory and I/O identically, so that it is immaterial whether a specific address location is occupied by RAM, ROM or I/O devices.) The next 4K block, from 9000-9FFF, is reserved for future expasion of SUPERMON, although you may use those locations if you wish to do so, provided you remember that if you should obtain an expanded SUPERMON system in the future these addresses may be used.

Extending from A000-AFFF are the I/O devices on your SYM-1 module. As we have previously said, each port on the SY6522/SY6532 devices in SYM-1 is an addressable location. Sheets 2-6 of Figure 4-10 provide you with a detailed Memory Map breakdown of how these devices are addressed. Note that within the SY6532 is a 128 byte segment (locations A600-A7FF). This is the RAM which is resident on the SY6532 used by SYM-1 as System RAM. Sheet 4 of Figure 4-10 describes each memory location within System RAM in detail; you will need this data if you wish to make use of the capability of the system for modifications to SUPERMON. These modifications may include creating your own commands (as described in Chapter 5) which may be entered as if they were Monitor commands. Other such modifications making use of System RAM locations are described in Chapter 9 of this manual.

Memory locations B000-FF80 may be used by your programs, provided of course you have expanded memory to fill those address locations (see Chapter 8). Note, however, that if you plan to obtain the Synertek Systems 8K BASIC module at some later date, that module will occupy locations C000-DFFF. You should plan your applications programs accordingly. Locations FF80-FFFF are reserved for special use by the system, and should not be used in any of your applications code.

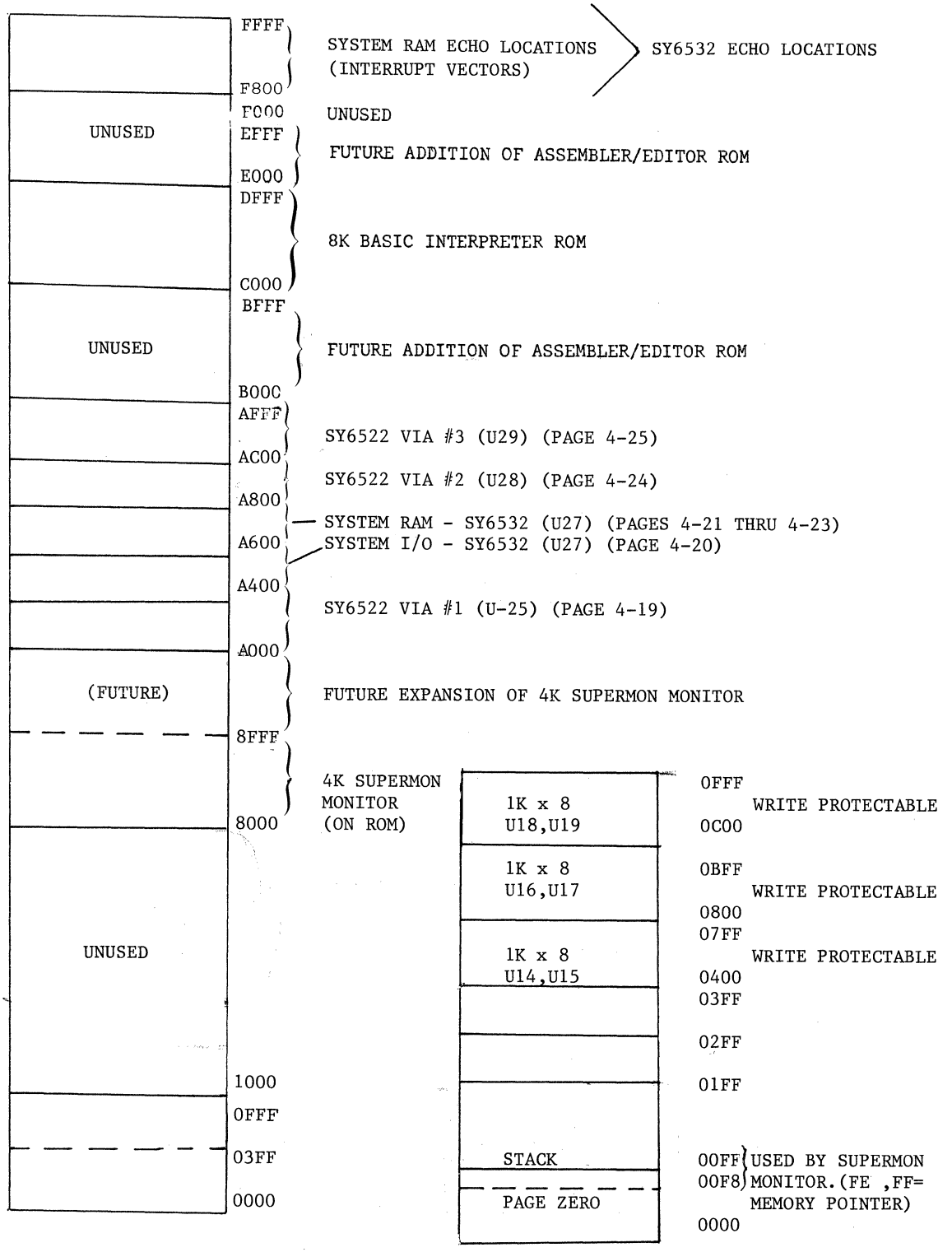


Figure 4-10. STANDARD MEMORY MAP, SYM-1

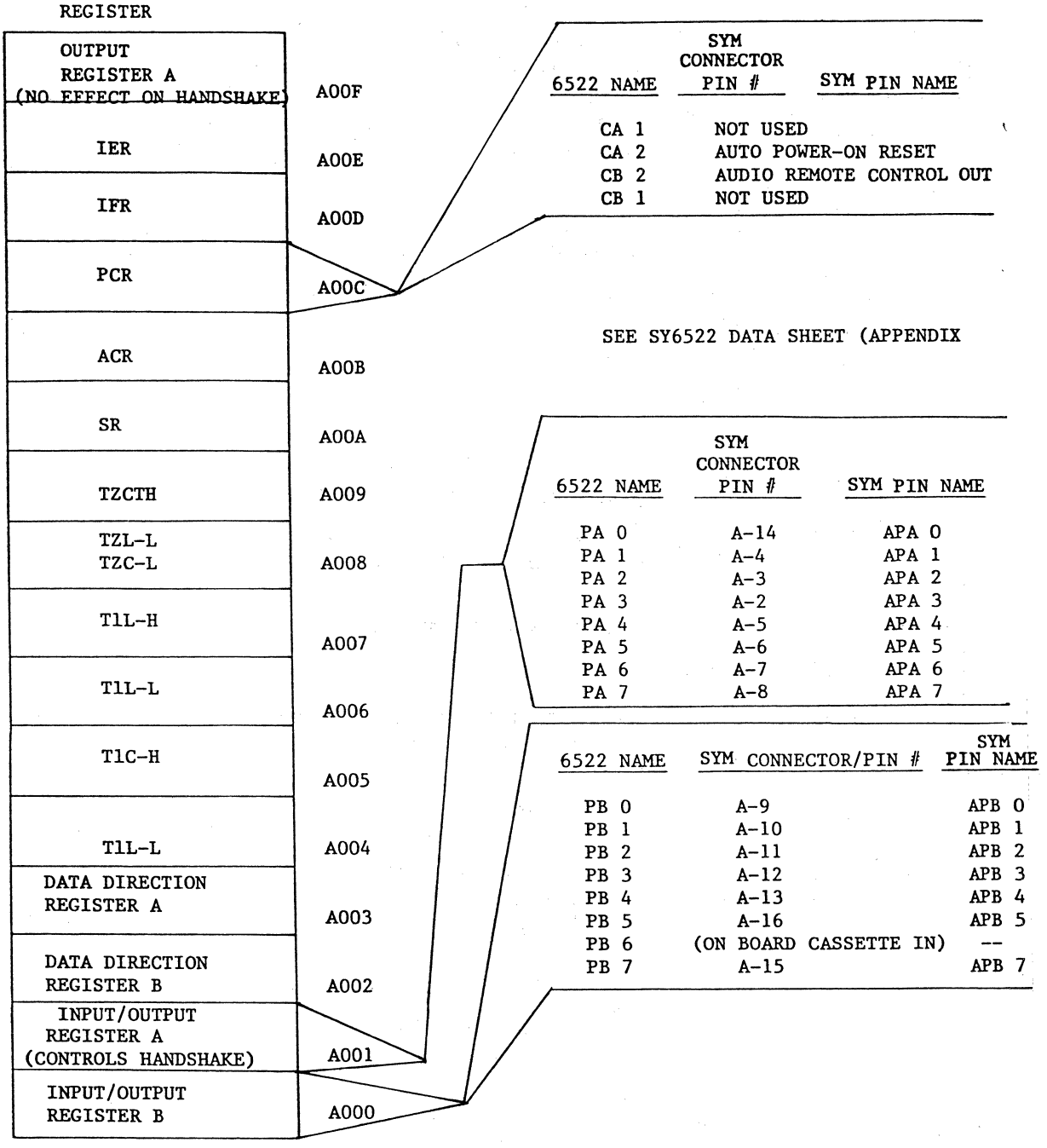


Figure 4-10 (Cont'd). MEMORY MAP FOR SY6522 VIA #1 (DEVICE U25)

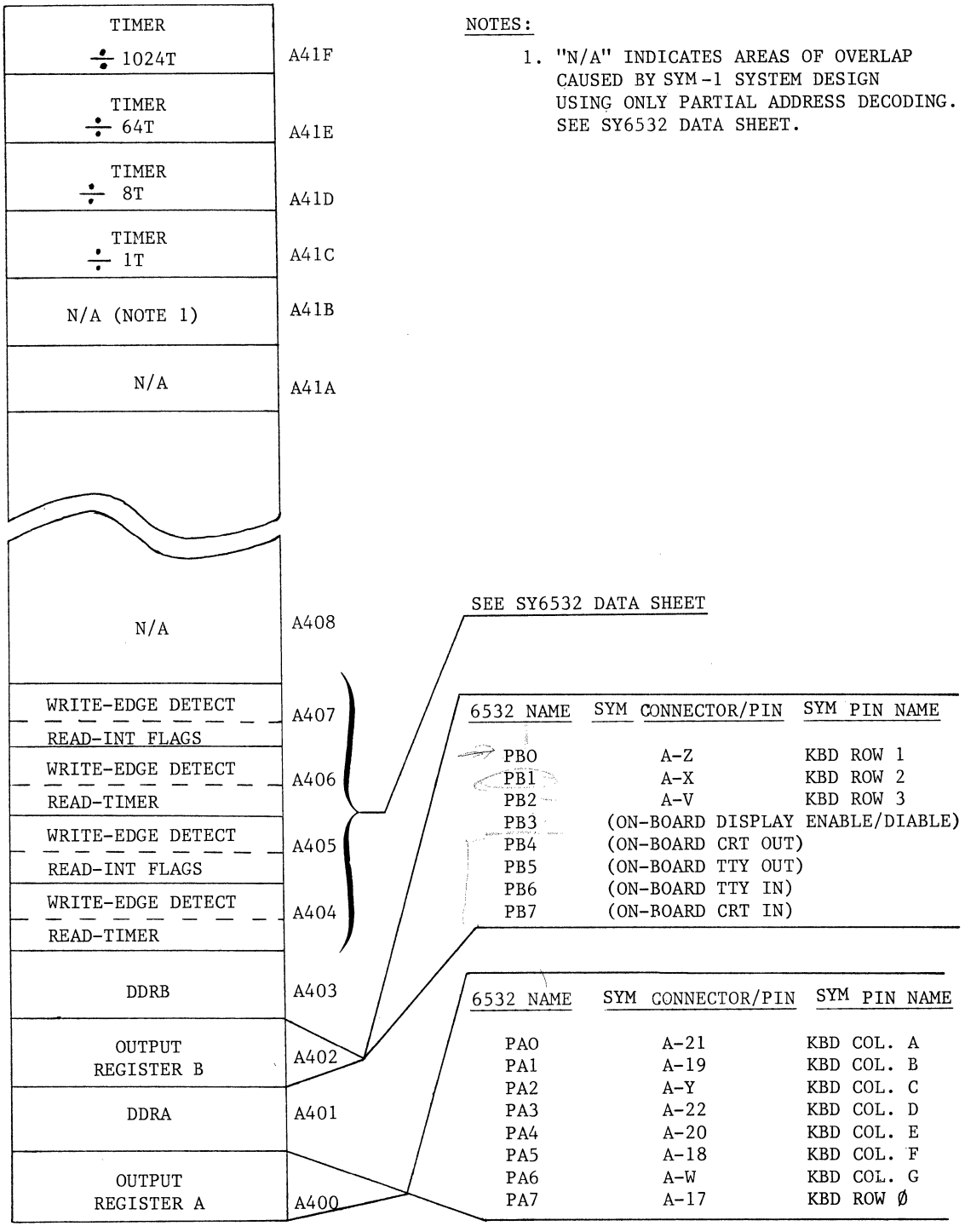


Figure 4-10 (Cont'd). MEMORY MAP FOR SY6532 (DEVICE U27)

SYMBOL	ADDRESS	DEFAULT VALUE	COMMENTS
IRQVEC	A67F	80	IRQ Vector
	A67E	0F	
RSTVEC	A67D	8B	RESET Vector
	A67C	4A	
NMIVEC	A67B	80	NMI Vector
	A67A	9B	
UIRQVC	A679	80	User IRQ Vector
	A678	29	
UBRKVC	A677	80	User Break Vector
	A676	4A	
TRCVEC	A675	80	Trace Vector
	A674	C0	
EXEVEC	A673	88	'Execute' Vector
	A672	7E	
SCNVEC	A671	89	
	A670	06	Display Scan Vector
	A66F	4C	
URCVEC	A66E	81	
	A66D	D1	Unrecognized Command Vector
	A66C	4C	
URSVEC	A66B	D1	
	A66A	81	Unrecognized Syntax Vector
	A669	4C	
INSVEC	A668	89	
	A667	6A	In Status Vector
	A666	4C	
OUTVEC	A665	89	
	A664	00	Output Vector
	A663	4C	
INVEC	A662	89	
	A661	BE	Input Vector
	A660	4C	
YR	A65F	00	
XR	A65E	00	
AR	A65D	00	
FR	A65C	00	User Registers
SR	A65B	FF	
PCHR	A65A	8B	
PCLR	A659	4A	
MAXRC	A658	10	Max. No. Bytes/Record, Paper Tape (Note 6)
LSTCOM	A657	00	Last Monitor Command
TV	A656	00	Trace Velocity (Note 5)
KSHFL	A655	00	Hex Keyboard Shift Flag
TOUTFL	A654	B0	In/Out Enable Flags (Note 4)
TECHO	A653	80	Terminal Echo (Note 3)
ERCNT	A652	00	Error Count (Note 2)
SDBYT	A651	4C	Baud Rate (Note 1)
PADBIT	A650	01	Number of Padbits on Carriage Return

Figure 4-10. SYSTEM RAM MEMORY MAP, SY6532

SYMBOL	ADDRESS	DEFAULT VALUE	COMMENTS
P1H	A64F	00	16-Bit Parameters
P1L	A64E	00	
P2H	A64D	00	
P2L	A64C	00	
P3H	A64B	00	
P3L	A64A	00	
PARNR	A649	00	No. of Parameters Entered
	A648	00	
	A647	00	Not Used
	A646	00	
RDIG	A645	06	Right-most Digit of Display Buffer
DISBUF	A644	86	
	A643	6E	
	A642	6D	Display Buffer
	A641	00	
	A640	00	
SCRF	A63F	00	
SCRE	A63E	00	Monitor Scratch Locations
SCRD	A63D	00	
TAPET2	A63C	5A	High Speed Tape Waveform
SCRB	A63B	00	Monitor Scratch Locations
SCR6	A636	00	
TAPET1	A635	33	High Speed Tape Waveform
SCR4	A634	00	Monitor Scratch Locations
SCR3	A633	00	
HSBDRY	A632	46	High Speed Tape Boundary
KMBDRY	A631	2C	KIM Tape Boundary
TAPDEL	A630	04	High Speed Tape Leader
JTABLE	A62F	D0	User Socket P3 (Jump Entry No. 7)
	A62E	00	
	A62D	C8	User Socket P2 (Jump Entry No. 6)
	A62C	00	
	A62B	03	0300 (Jump Entry 5)
	A62A	00	
	A629	02	0200 (Jump Entry 4)
	A628	00	
	A627	00	0000 (Jump Entry 3)
	A626	00	
	A625	8B	NEWDEV (Jump Entry 2) (Note 7)
	A624	64	
	A623	8B	TTY (Jump Entry 1)
	A622	A7	
	A621	C0	BASIC (Jump Entry 0)
	A620	00	
SCPBUF	A61F	--	Scope Buffer, No Defaults (32 locations)
	A600	--	

Figure 4-10. SYSTEM RAM MEMORY MAP, SY6532 (Continued)

NOTES - SYSTEM RAM

1. BAUD RATE -

BAUD	SDBYT
110	D5
300	4C
600	24
1200	10
2400	06
4800	01

2. ERCNT - Used by LD P, FILL, B MOV

   Count of bytes which failed to write correctly
   And invalid checksums up to $FF

3. TECHO - bit 7 - ECHO/NO ECHO

   bit 6 - OUTPUT/NO OUTPUT This bit is toggled everytime a control O (ASCII 0F) is encountered in the input stream.

4. TOUTFL - bit 7 = enable CRT IN
   bit 6 = enable TTY IN
   bit 5 = enable TTY OUT
   bit 4 = enable CRT OUT

5. TV - TRACE VELOCITY
   00 = SINGLE STEP

   non-zero - PRINT PROGRAM COUNTER AND ACCUMULATOR
   THEN PAUSE AND RESUME

   PAUSE DEPENDS ON TV
   (TRY TV = 09)

6. USER PC - DEFAULT = 8B4A = RESET

7. NEW DEV TO CHANGE BAUD RATE ON RS-232 INTERFACE.

Figure 4-10. SYSTEM RAM MEMORY MAP, SY6532 (Continued)

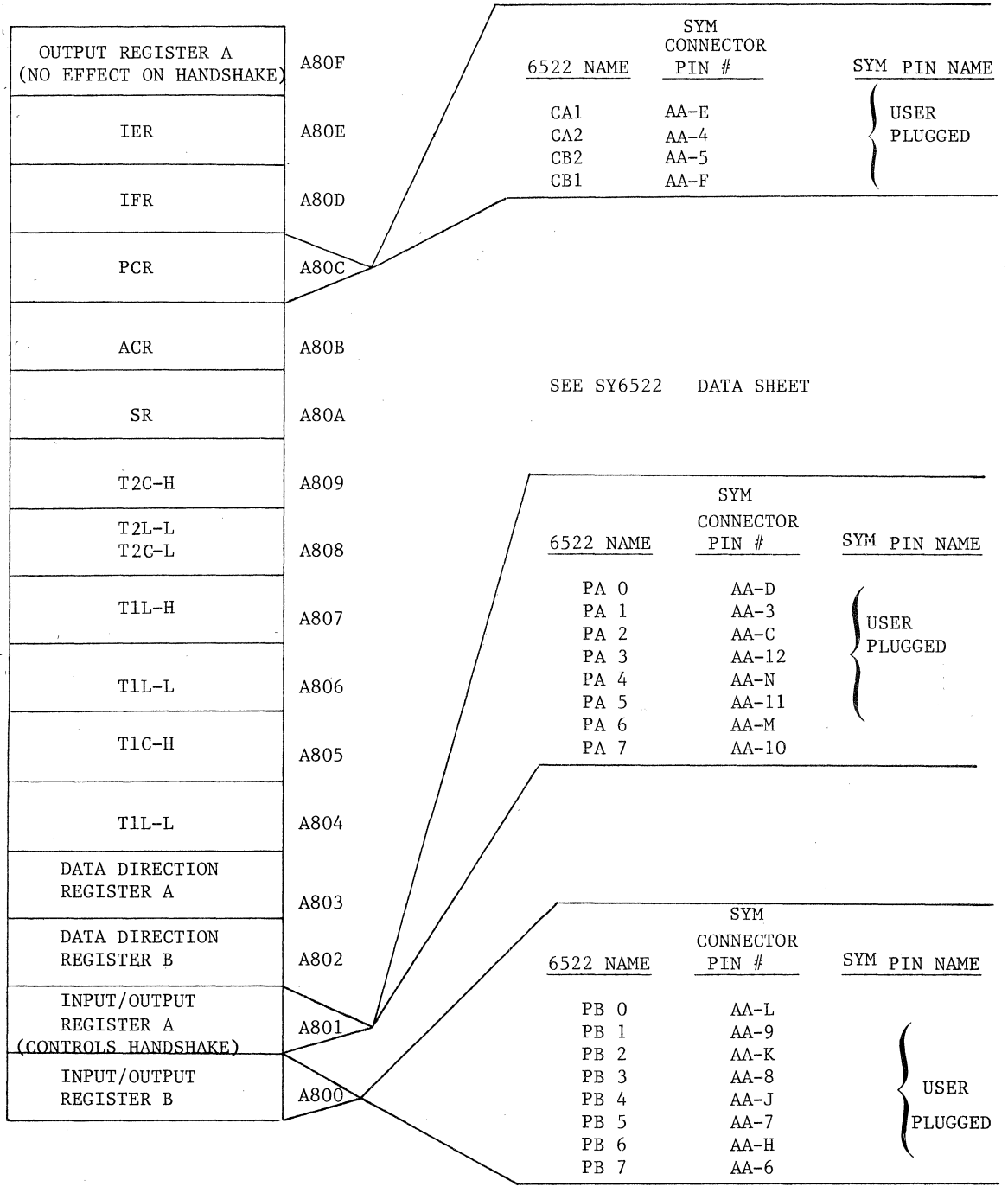


Figure 4-10. MEMORY MAP FOR SY6522 VIA #2 (DEVICE U28-USER SUPPLIED)
(Continued)

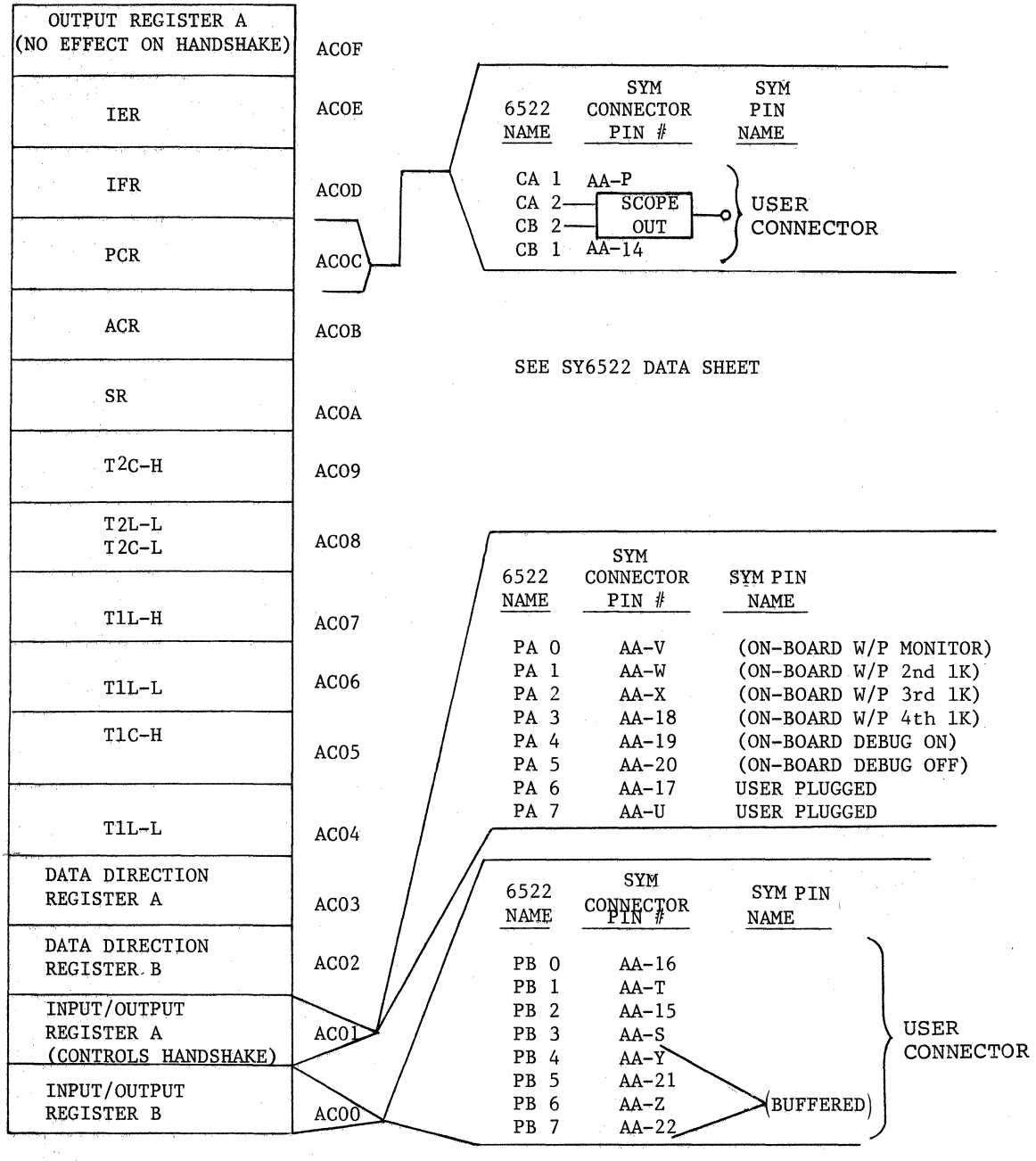


Figure 4-10. MEMORY MAP FOR SY6522 VIA #3 (DEVICE U29) (Continued)

4.2.2 Address Decoding Jumper Options

Four sockets (labeled P0-P3 on the board) for ROM, PROM or EPROM are provided with your SYM-1. Each socket may contain any of four different types of Read-Only Memory devices, up to a total of 28K. The four acceptable devices are the SY2716, the SY2316B, the SY2332 and the SY2364. Each device is slightly different, but they are all read-only memories. They may appear in any combination on a SYM-1 microcomputer system, provided their total capacity does not exceed 24K. But since the devices have different memory capacities, it is necessary to alter normal addressing to accomodate the specific devices selected.

To serve this purpose, we have provided a set of jumpers, located just to the left of the center of the board and directly under the two 74LS145's. The schematic in Figure 4-11 illustrates each useful jumper combination and Table 4-3 outlines them in greater detail. (Note that Table 4-3 contains other jumpers available on the SYM-1, not all of which pertain to memory use.) The broken lines in Figure 4-11 indicate the jumpers installed at the factory. Note, for example, that the first PROM socket, labeled P0 (device U20) is associated with the address group beginning with 8000. If it were necessary to change this configuration, you would remove the connection from Pin 1 of the lower address decoder U10 (74LS145) to jumper connection 7-J so that it becomes associated with a jumper combination which addresses the device you wish to address. Table 4-3a will assist you in configuring your selection of ROM correctly.

Near the bottom of the board below the speaker unit are four jumpers labeled JJ, KK, LL and MM. These enable Write Protection on the RAM in the four 1K blocks available on the board. Jumper 45-MM is factory-installed, enabling Write Protection on System RAM (the 128-byte block in the SY6532). As you add RAM later, or to Write Protect any of the on-board RAM aside from System RAM, you must connect the appropriate jumpers to enable the Write Protect function on the desired memory locations. RAM may be enabled for Write Protect in 1K blocks.

These jumpers offer you flexibility to adapt the SYM-1 board to your particular application. The jumpers will give you the ability to do the following:

- o Use 2K, 4K, or 8K byte ROM or PROM in each 24 pin socket.
- o Complete flexibility in selecting user PROM addressing.
- o Ability to auto power-on to any of the ROM/PROM sockets.
- o Write protect expansion RAM.

4.2.3 Off-Board Expandability

SYM-1 is expandable, on-board, up to 28K bytes of EPROM/ROM memory and 4K bytes of RAM, with 4K bytes of address space allocated to the on-board I/O devices. Further expansion of any combination of ROM, PROM, RAM or I/O can be implemented by using SYM's "E" (Expansion) connector to attach an auxiliary board containing the additional devices. Total expandability is limited only by the amount of addressing capability of the SY6502 CPU, i.e., 64K bytes.

Detailed instructions for implementing off-board expansion are contained in Chapter 8, "System Expansion."

4.2.4 I/O Buffers

Your SYM-1 board comes to you equipped with four specially configured I/O buffer circuits. (See Figure 4-5.) The circuit configuration and PC Board layout allow the user to configure these buffers in many ways.

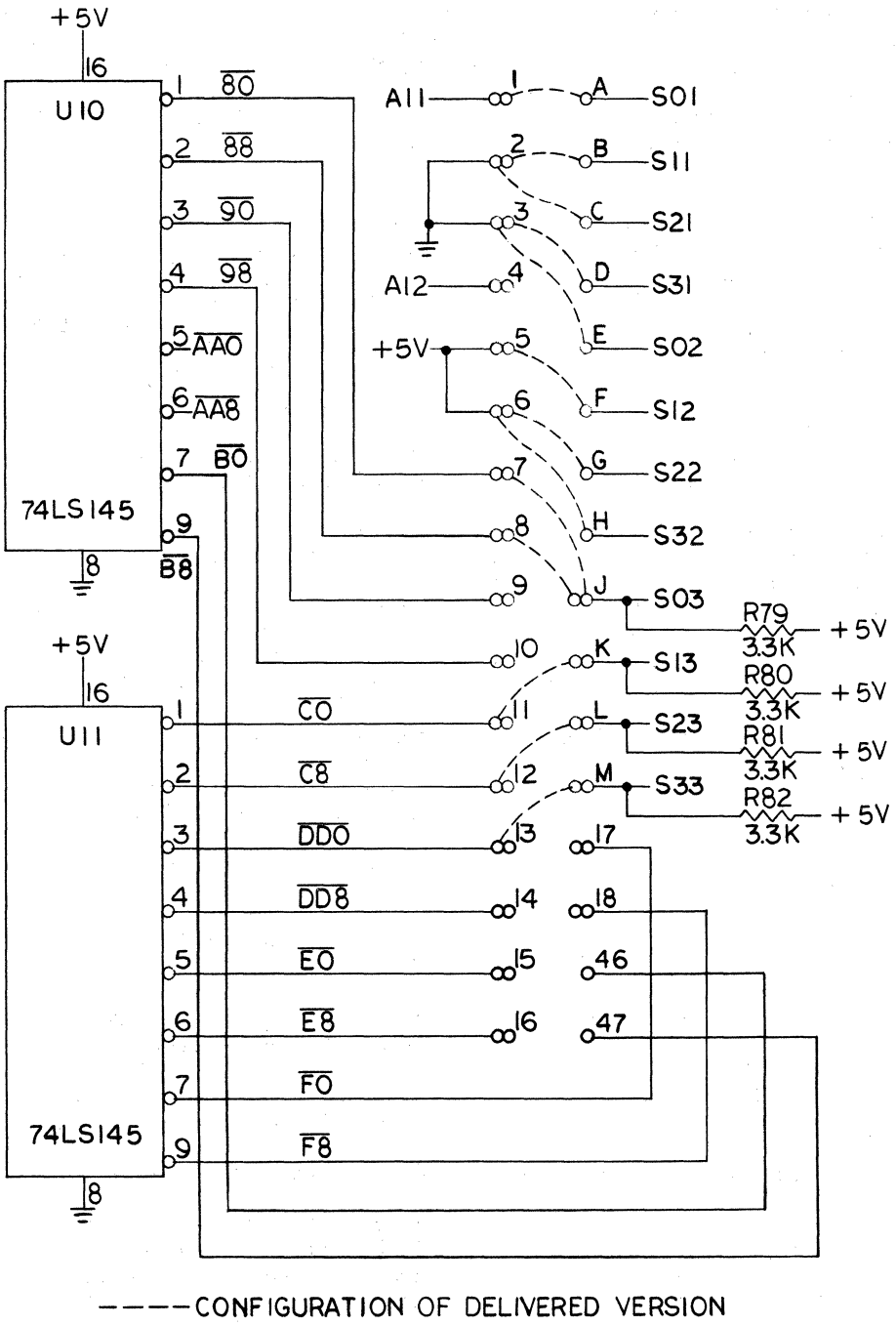


Figure 4-11. MEMORY ADDRESS DECODING JUMPER OPTIONS

Table 4-3. SYM-1 JUMPERS

JUMPER LETTER	POSITION NUMBER	DESCRIPTION
A,B,C,D E,F,G,H	1,2,3 4,5,6	PROM/ROM Device Select (See Table 4-3a)
J,K,L,M	7,8,9,10,11,12 13,14,15,16,17,18	ADDRESS SELECT (See Table 4-3b)
N	19 (1) 20	Auto Power-On to U20 (2) Disable Auto Power-On to U20
P	19 (1) 20	Auto Power-On to U21 (2) Disable Auto Power-On to U21
R	19 (1) 20	Auto Power-On to U22 (2) Disable Auto Power-On to U22
S	19 (1) 20	Auto Power-On to U23 (2) Disable Auto Power-On to U23
T U	21 22	Enables Monitor RAM at A0xx (3) Enables Monitor RAM at F8xx (3)
V	23	RCN-1 to connector A-N
W X	24 25	Enables Software Debug ON Enables Software Debug OFF
Y	26	DBOUT to connector E-17
BB CC	31 32	Connects TTY IN to PB6 @A402 Connects CRT IN to PB7 @A402
DD	33 34	To run TTY @ +5V and GND To run TTY @ +5V and -Vn (4)
EE	35 36	To run TTY @ +5V and -Vn (4) To run TTY @ +5V and GND
FF	37 38	To run TTY @ +5V and GND To run TTY @ +5V and -Vn (4)
GG	39 40	To run RS232 @+5V and GND To run RS232 @+5V and -Vn (5)
HH	41	Decode line $\overline{18}$ to connector A-K
JJ KK LL MM	42 43 44 45	Enable software write protect 3K block Enable software write protect 2K block Enable software write protect 1K block Enable software write protect monitor RAM

Table 4-3. SYM-1 JUMPERS (Continued)

NOTES

1 Only one socket (U20, U21, U22, U23) should be jumpered to position 19 at one time. The remaining three sockets should be jumpered to position 20.

2 See software consideration of auto power-on in Chapter 9.

3 One or both can be connected at the same time.

4 These positions require a recommended -9V to -15V supply applied to the power connector pin E. R107 should be adjusted (removed and replaced) for your proper current loop requirements.

Example: (for 60ma current loops and Vn = -10V)

a. Connect DD to 33
EE to 35
FF to 38

b. $R107 = \frac{Vn - 5V}{I} = \frac{(10 - 5)}{60\ ma} = 100$

R107 = 300Ω (as installed) for 20 ma current loop and Vn=-10V

5 For RS232 devices using other than LM1489 or equivalent input receivers (i.e., probably terminals older than ten years) then GG should be strapped to 40 and a -9V to -15V supply applied to the power connector pin E.

Table 4-3a. SYM-1 PROM/ROM DEVICE SELECT

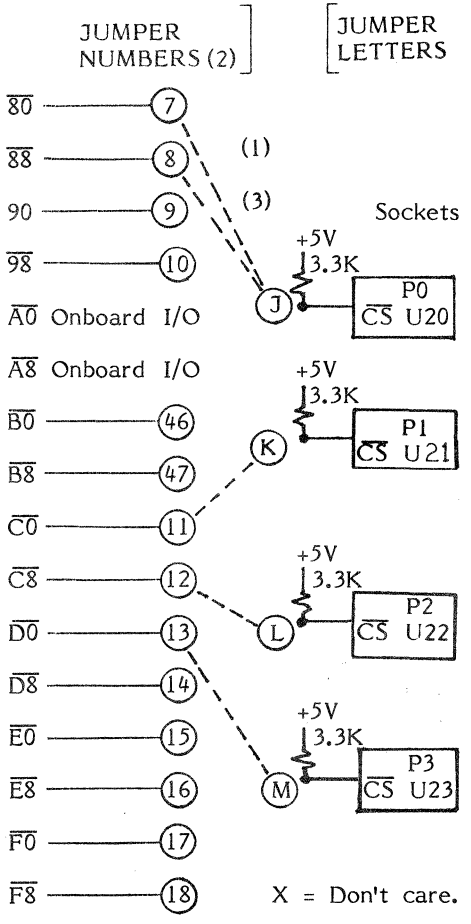
SOCKET LOCATION	SOCKET NAME	MEMORY DEVICE	JUMPER LETTER	POSITION NUMBER
U20	P0	2716	A E	2 or 3 5 or 6
U20	P0	2316	A E	2 or 3 2 or 3
U20	P0	2332	A E	1 2 or 3
U20	P0	2364	A E	1 4
U21	P1	2716	B F	2 or 3 5 or 6
U21	P1	2316	B F	2 or 3 2 or 3
U21	P1	2332	B F	1 2 or 3
U21	P1	2364	B F	1 4
U22	P2	2716	C G	2 or 3 5 or 6
U22	P2	2316	C G	2 or 3 5 or 6
U22	P2	2332	C G	1 2 or 3
U22	P2	2364	C G	1 4
U23	P3	2716	D H	2 or 3 5 or 6
U23	P3	2316	D H	2 or 3 2 or 3
U23	P3	2332	D H	1 2 or 3
U23	P3	2364	D H	1 4

NOTE: 2716 devices assumes Synertek, Intel or equivalent pin outs.

Table 4-3b. SYM-1 ADDRESS SELECT

High Order SYM-1 Address lines

2K Blocks	A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8
	1	0	0	0	0	X	X	X
	1	0	0	0	1	X	X	X
	1	0	0	1	0	X	X	X
	1	0	0	1	1	X	X	X
	1	0	1	0	0	X	X	X
	1	0	1	0	1	X	X	X
	1	0	1	1	0	X	X	X
	1	0	1	1	1	X	X	X
	1	1	0	0	0	X	X	X
	1	1	0	0	1	X	X	X
	1	1	0	1	0	X	X	X
	1	1	0	1	1	X	X	X
	1	1	1	0	0	X	X	X
	1	1	1	0	1	X	X	X
	1	1	1	1	0	X	X	X
	1	1	1	1	1	X	X	X



NOTES:

(1) Broken lines indicate delivered version of jumpers.
(2) Each jumper number represents a 2K address space decode.
(3) Jumper numbers can be wire or'ed to increase the address space of the $\overline{CS}$ on any socket (i.e., decoder is open collector.)

The single-stage circuit consists of a transistor and "circuit positions" for the user to add resistors, capacitors and dioders in any of many positions. This flexibility allows inverting and noninverting stages, input-resistive or capacitive coupling and much more. The user should refer to the schematic and P.C. layout in Figure 4-5a in order to completely understand this circuit.

4.3 SOFTWARE DESCRIPTION

Software on your SYM-1 microcomputer must be discussed from two perspectives. First, the SYM SUPERMON Monitor software which handles keyboard display, interrupts and other requirements for system operation must be understood. We will discuss this subject in succeeding sections. The second aspect of software is the microprocessor assembly language with which you will write your applications programs. A brief introduction to the 6502 instruction set is included later in this chapter.

In this chapter, we discuss the SYM-1 command language syntax only briefly; Chapter 5 contains a detailed discussion of each of the instructions in the set. Chapter 6 will help you through the process of using these and the 6502 language in applications programming by describing three selected sample programs.

4.3.1 Monitor Description - General

Figure 9-1 illustrates the general system flow of the SYM-1 SUPERMON Monitor software. As you can tell, the main program is simple and straightforward. Its purpose is to direct processing to the appropriate I/O or command routine, and for this reason it is thought of as a "driver"--it "drives" or directs the software.

The means by which the Monitor handles the direction of software flow is one of the unique features of the SYM-1 system and is worth a brief explanation at this point. We will discuss the subject in greater detail in Chapters 5 and 8.

When the SUPERMON Monitor receives a one- or two-character command from the on-board keyboard, TTY or CRT terminal, it then accepts 0-3 parameters associated with the command. The string of command and parameters (if any) is terminated by a carriage return. It is noteworthy that each instruction which may be entered by use of a single key on the on-board keyboard may also be entered with a similar command from a terminal.

Upon receiving a command and up to three parameters, SUPERMON checks to determine whether the command and its associated number of parameters is a defined combination. If so, the command is executed. Otherwise, an error message is printed or displayed showing the ASCII representation of the command which was not recognized.

For example, a "GO" with one parameter causes the program to pass control to the program stored at the memory location indicated by the parameter. Thus, a "GO" followed by "0200" instructs the system to begin executing the instructions stored starting at memory location 0200. A "GO" with no parameters (i.e., "GO" followed by a Carriage Return) will cause program execution to resume at the address stored in the "pseudo Program Counter" (memory locations A659 and A65A).

However, a "GO" command with two or three parameters is not a defined command in SUPERMON, and will result in a display or message of "Er 47". The "47" is the ASCII representation for a "G" and is designed to help you define the instruction or command which was not recognized.

The monitor is designed so that you can extend the range of defined command-parameter combinations by "intercepting" the error routine before it executes and designing your own series of pointers to memory locations to be associated with specific commands. Thus, you might wish to define a "GET" routine which could be entered at the keyboard with a "GO" and two parameters. You will learn how to do this in Chapter 9.

4.3.2 Software Interfacing

The SYM-1 Monitor is structured to be device-independent. Special requirements for device handling are "outside" the Monitor's central control routines, which isolate them from the Monitor's standard functions. Also, as we have indicated, SYM-1 commands may be entered from any device. It is not necessary to use the on-board keyboard to do so. This means you need not concern yourself with the details of I/O; they are handled internally.

4.3.3 6502 Microprocessor Assembly Language Syntax

The SY6502 microprocessor used on your SYM-1 is an eight-bit CPU, which means that eight bits of data are transferred or operated upon at a time. It has a usable set of 56 instructions used with 13 addressing modes. Instructions are divided into three groups.

Group One instructions, of which there are eight, are those which have the greatest addressing flexibility and are therefore the most general-purpose. These include Add With Carry (ADC), the logical AND (AND), Compare (CMP), the logical Exclusive OR (EOR), Load A (LDA), logical OR with Accumulator (ORA), Subtract With Carry (SBC) and Store Accumulator (STA).

Group Two instructions include those which are used to read and write data or to modify the contents of registers and memory locations.

The remaining 39 instructions in the SY6502 instruction set are Group Three instructions which operate with the X and Y registers and control branching within the program. You'll learn more about these instructions in the next section. More detailed information can be found in the Synertek Programming Manual for the SY6500 family.

An assembly language instruction consists of the following possible parts:

Label - Optional. Used to allow branching to the line containing the label and for certain addressing situations.

Mnemonic - Required. The mnemonic is a three-character abbreviation which represents the instruction to be carried out. Thus the mnemonic to store the contents of the accumulator in a specific memory location is "STA" (STore Accumulator).

Operand(s) - Some may be required, or none may be allowed. This depends entirely upon the instruction itself and may be determined from the later discussion.

Comment - Optional. Separated from last operand (or from the command mnemonic where no operand is used) by at least one blank. These words are ignored by the assembler program but are included only to allow the programmer and others to understand the program.

The SY6502 allows 13 modes of addressing, which makes it one of the most flexible CPUs on the market. Table 4-4 describes these addressing modes briefly. Details may be found in the Synertek Programming Manual for the SY6500 family.

You will note that some of the addressing modes make use of Page Zero, a concept introduced briefly earlier in this chapter. Page Zero addressing modes are designed to reduce memory requirements and provide faster execution. When the SY6502 processor encounters an instruction using Page Zero addressing, it assumes the high-order byte of the address to be 00, which means you need not define that byte in your program. This technique is particularly useful in dealing with working registers and intermediate values. As the Memory Map (Figure 4-10, Sheet 1) shows, memory locations 0000-00FF make up Page Zero.

4.3.4 SY6502 Instruction Set

Table 4-5 provides you with a summary of the SY6502 instruction set. Each instruction is shown with its mnemonic, a brief description of the function(s) it carries out, and the corresponding "op code" for each of its valid addressing modes. The "op code" is the hexadecimal representation of the instruction and is what will appear when the instruction byte is displayed by SUPERMON.

When creating applications programs for your SYM-1, you will typically write them in the SY6502 assembly language mnemonic structure shown in Table 4-5, then perform a "hand assembly" to generate the "op codes" and operands. The process of hand assembling code is explained in greater detail in Section 6.2.2. You will be referring to this table--or to your SYM Reference Card--quite frequently during programming.

To understand some of the instructions, you should be aware of six "status register" flags which are set and reset by the results of program execution. Generally, these flags and their functions are:

N - Set to "1" by CPU when the result of the previous instruction is negative
Z - Set to "1" by CPU when the result of the previous instruction is zero
C - Set to "1" by CPU when the previous instruction results in an arithmetic "carry"
Set to "0" by CPU when the previous instruction results in "borrow" (subtract)
Also modified by shift, rotate and compare instructions.
I - When "1," IRQ to the CPU is held pending
D - When "1," CPU arithmetic is operates in decimal mode
V - Set to "1" by CPU when the result of the previous instruction causes an arithmetic overflow

The Synertek Programming Manual (MNA-2) discusses this subject in greater detail.

Table 4-4. SUMMARY OF SY6502 CPU ADDRESSING MODES

SY6502 INSTRUCTION SET SUMMARY

Addressing Modes

Mode	Description	# Bytes	Example		
IMPLIED	The operation performed is implied by the instruction.	1*	TAX	AA	Code for transfer A to X
ACCUMULATOR	The operation is performed upon the A register.	1	ROL A	2A	Code for rotate left A
IMMEDIATE	The data accessed is in the second byte of the instruction.	2	LDA #3	A9 03	Code for load A immediate Constant to use
ZERO PAGE	The address within page zero of the data accessed is in the second byte of the instruction.	2	LDA Z	A5 75	Code for load A zero page Low part of address on page zero
ZERO PAGE INDEXED BY X	The second byte of the instruction plus the contents of the X register (without carry) is the address on page zero of the data accessed.	2	LDA Z,X	B5 75	Code for zero page indexed by X Base address on page zero
ZERO PAGE INDEXED BY Y	The second byte of the instruction plus the contents of the Y register (without carry) is the address on page zero of the data accessed.	2	LDX Z,Y	B6 75	Code for zero page indexed by Y Base address on page zero
ABSOLUTE	The address of the data accessed is in the second and third bytes of the instruction.	3	LDA L	AD 47 02	Code for load A absolute Low part of address High part of address

*Except BRK which is two bytes when not using SUPERMON or when in DEBUG mode.

Table 4-4. SUMMARY OF SY6502 CPU ADDRESSING MODES (Continued)

Mode	Description	# Bytes	Example		
INDEXED BY X	The address in the second and third bytes of the instruction, plus the contents of the X register is the address of the data accessed.	3	LDA L,X	BD 47 02	Code for load A indexed by X Low part of base address High part of base address
INDEXED BY Y	The address in the second and third bytes of the instruction, plus the contents of the Y register is the address of the data accessed.	3	LDA L,Y	B9 47 02	Code for load A indexed by Y Low part of base address High part of base address
INDIRECT PRE-INDEXED BY X	The second byte of the instruction plus the contents of the X register (without carry) is the address on page zero of the two-byte address of the data accessed.	2	LDA (Z,X)	A1 75	Code for load A, indirect pre-indexed by X Base address on page zero
INDIRECT POST-INDEXED BY Y	The contents of the page zero two-byte address specified by the second byte in the instruction, plus the contents of the Y register is the address of the data accessed.	2	LDA (Z),Y	B1 75	Code for load A, indirect post-indexed by Y Base address of page zero
RELATIVE BRANCH	The second byte of the instruction contains the offset (in bytes) to branch address.	2	BEQ LOC	F0 07	Code for branch if equal Seven bytes ahead
INDIRECT JUMP	The address in the second and third bytes of the instruction is the address of the address to which the jump is made.	3	JMP (LOC)	6C 47 02	Code for jump indirect Low part of indirect address High part of indirect address

Table 4-5. SY6502 CPU Instruction Set Summary

6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
ADC	A + M + C → A, C Add memory to accumulator with carry			69	65	75		6D	7D	79	61	71			*	*	*	-	-	*	-
AND	A ∧ M → A "AND" memory with accumulator			29	25	35		2D	3D	39	21	31			*	*	-	-	-	-	-
ASL	[C] ← [7][6][5][4][3][2][1][0] ← 0 Shift left one bit (memory or accumulator		0A		06	16		0E	1E						*	*	*	-	-	-	-
BCC	Branch on C = 0 Branch on carry clear												90		-	-	-	-	-	-	-
BCS	Branch on C = 1 Branch on carry set												B0		-	-	-	-	-	-	-
BEQ	Branch on Z = 1 Branch on result zero												F0		-	-	-	-	-	-	-
BIT	A ∧ M, M_7 → N, M_6 → V Test bits in memory with accumulator				24			2C							M_7	*	-	-	-	M_6	-
BMI	Branch on N = 1 Branch on result minus												30		-	-	-	-	-	-	-
BNE	Branch on Z = 0 Branch on result not zero												D0		-	-	-	-	-	-	-
BPL	Branch on N = 0 Branch on result plus												10		-	-	-	-	-	-	-
BRK	Forced interrupt PC↓ P↓ Force break	00													-	-	-	-	-	-	1

Table 4-5. SY6502 CPU Instruction Set Summary (Continued)

6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
BVC	Branch on V = 0 Branch on overflow clear												50		-	-	-	-	-	-	-
BVS	Branch on V = 1 Branch on overflow set												70		-	-	-	-	-	-	-
CLC	0 → C Clear carry flag	18													-	-	0	-	-	-	-
CLD	0 → D Clear decimal mode flag	D8													-	-	-	-	0	-	-
CLI	0 → I Clear interrupt disable flag	58													-	-	-	0	-	-	-
CLV	0 → V Clear overflow flag	B8													-	-	-	-	-	0	-
CMP	A - M Compare memory and accumulator			C9	C5	D5		CD	DD	D9	C1	D1			*	*	*	-	-	-	-
CPX	X - M Compare memory and index X			E0	E4			EC							*	*	*	-	-	-	-
CPY	Y - M Compare memory and index Y			C0	C4			CC							*	*	*	-	-	-	-
DEC	M - 1 → M Decrement memory by one				C6	D6		CE	DE						*	*	-	-	-	-	-
DEX	X - 1 → X Decrement index X by one	CA													*	*	-	-	-	-	-

Table 4-5. SY6502 CPU Instruction Set Summary (Continued)

6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
DEY	Y - 1 → Y Decrement index Y by one	88													*	*	-	-	-	-	-
EOR	A ⊻ M → A "Exclusive-Or" memory with accumulator			49	45	55		4D	5D	59	41	51			*	*	-	-	-	-	-
INC	M + 1 → M Increment memory by one				E6	F6		EE	FE						*	*	-	-	-	-	-
INX	X + 1 → X Increment Index X by one	E8													*	*	-	-	-	-	-
INY	Y + 1 → Y Increment index Y by one	C8													*	*	-	-	-	-	-
JMP	(PC + 1) → PCL (PC + 2) → PCH Jump to new location							4C						6C	-	-	-	-	-	-	-
JSR	PC + 2 ↓, (PC + 1) → PCL (PC + 2) → PCH Jump to new location saving return address							20							-	-	-	-	-	-	-
LDA	M → A Load accumulator with memory			A9	A5	B5		AD	BD	B9	A1	B1			*	*	-	-	-	-	-
LDX	M → X Load index X with memory			A2	A6		B6	AE		BE					*	*	-	-	-	-	-
LDY	M → Y Load index Y with memory			A0	A4	B4		AC	BC						*	*	-	-	-	-	-

Table 4-5. SY6502 CPU Instruction Set Summary (Continued)

6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
LSR	0 → 7 6 5 4 3 2 1 0 → C Shift right one bit (memory or accumulator)		4A		46	56		4E	5E						0	*	*	-	-	-	-
NOP	No Operation	EA													-	-	-	-	-	-	-
ORA	A V M → A "OR" memory with accumulator			09	05	15		0D	1D	19	01	11			*	*	-	-	-	-	-
PHA	A ↓ Push accumulator on stack	48													-	-	-	-	-	-	-
PHP	P ↓ Push processor status on stack	08													-	-	-	-	-	-	1
PLA	A ↑ Pull accumulator from stack	68													*	*	-	-	-	-	-
PLP	P ↑ Pull processor status from stack	28													From Stack						
ROL	M or A 7 6 5 4 3 2 1 0 ← C Rotate one bit left (memory or accumulator)		2A		26	36		2E	3E						*	*	*	-	-	-	-
ROR	Rotate One Bit Right (Memory or Accumulator)		6A		66	76		6E	7E						*	*	*	-	-	-	-
RTI	P ↑ PC ↑ Return from interrupt	40													From Stack						
RTS	PC ↑, PC + 1 → PC Return from subroutine	60													-	-	-	-	-	-	-

Table 4-5. SY6502 CPU Instruction Set Summary (Continued)

6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
SBC	A - M - $\overline{C}$ → A Note: $\overline{C}$ = Borrow Subtract memory from accumulator with borrow			E9	E5	F5		ED	FD	F9	E1	F1			*	*	*	-	-	*	-
SEC	1 → C Set carry flag	38													-	-	1	-	-	-	-
SED	1 → D Set decimal mode flag	F8													-	-	-	-	1	-	-
SEI	1 → I Set interrupt disable flag	78													-	-	-	1	-	-	-
STA	A → M Store accumulator in memory				85	95		8D	9D	99	81	91			-	-	-	-	-	-	-
STX	X → M Store index X in memory				86		96	8E							-	-	-	-	-	-	-
STY	Y → M Store index Y in memory				84	94		8C							-	-	-	-	-	-	-
TAX	A → X Transfer accumulator to index X	AA													*	*	-	-	-	-	-
TAY	A → Y Transfer accumulator to index Y	A8													*	*	-	-	-	-	-
TSX	S → X Transfer stack pointer to index X	BA													*	*	-	-	-	-	-

Table 4-5. SY6502 CPU Instruction Set Summary (Continued)

6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
TXA	X → A Transfer index X to accumulator	8A													*	*	-	-	-	-	-
TXS	X → S Transfer index X to stack pointer	9A													-	-	-	-	-	-	-
TYA	Y → A Transfer index Y to accumulator	98													*	*	-	-	-	-	-

CHAPTER 5

OPERATING THE SYM

In this chapter you will learn how to operate your SYM-1. The keyboard functions are described, formation of monitor commands is discussed, and procedures for using an audio cassette, TTY or CRT are explained.

As you operate your SYM-1, you will be dealing with the system monitor, SUPERMON, which is a tool for entering, debugging and controlling your 6502 programs. The monitor also provides a wealth of software resources (notably subroutines and tables) which are available to your applications programs as they run on the SYM-1 system.

SUPERMON is a 4K-byte program which is stored on a single ROM chip located at addresses 8000-8FFF, as you learned in Chapter 4. It also uses locations 00F8-00FF for special purposes and special locations called "System RAM" located at addresses A600-A67F. These usages were outlined in detail in Chapter 4 and in the Memory Map.

Operationally, SUPERMON gets commands, parameters and data from its input channels (the HEX Keyboard, HKB; a teletype, TTY; a CRT terminal or RAM memory and others) and, based on this input, performs internal manipulations and various outputs (to the on-board LED display, TTY or CRT terminal screen or other peripheral devices).

5.1 KEYBOARD LAYOUT

The SYM-1 keyboard (see Figure 5-1) consists of 28 color-coded dual-function keys. The characters and functions on the lower half of the keys are entered by pressing the keys directly. To enter the functions shown in the upper halves of the keys, press SHIFT before you press the key you wish to enter. Remove your finger from SHIFT before pressing the second key. Very little pressure is necessary to actuate a key, and except for DEBUG, you will hear an audible tone when the computer senses that a key has been pressed. RST will cause a beep after a short delay.

The functions included on the SYM-1 provide you with a formidable array of programming tools. You can examine and modify the contents of memory locations and CPU registers, deposit binary or ASCII data in memory, move blocks of data from one area of memory to another, search memory for a specific byte, and fill selected memory locations with a specified data byte. You can also store a double byte of data with a single command, display the two's complement of a number, or compute an address displacement.

The RST, DEBUG ON and DEBUG OFF keys do not transmit any characters to the monitor, but perform the functions indicated by their names directly using hardware logic.

5.2 SYM COMMAND SYNTAX

As we have indicated earlier, each SYM-1 command entered from the on-board keyboard or other device may have from 0-3 parameters associated with it. Each command, with its string of parameters, is terminated by a **"CR"** (on the HKB) or a carriage return on a terminal device.

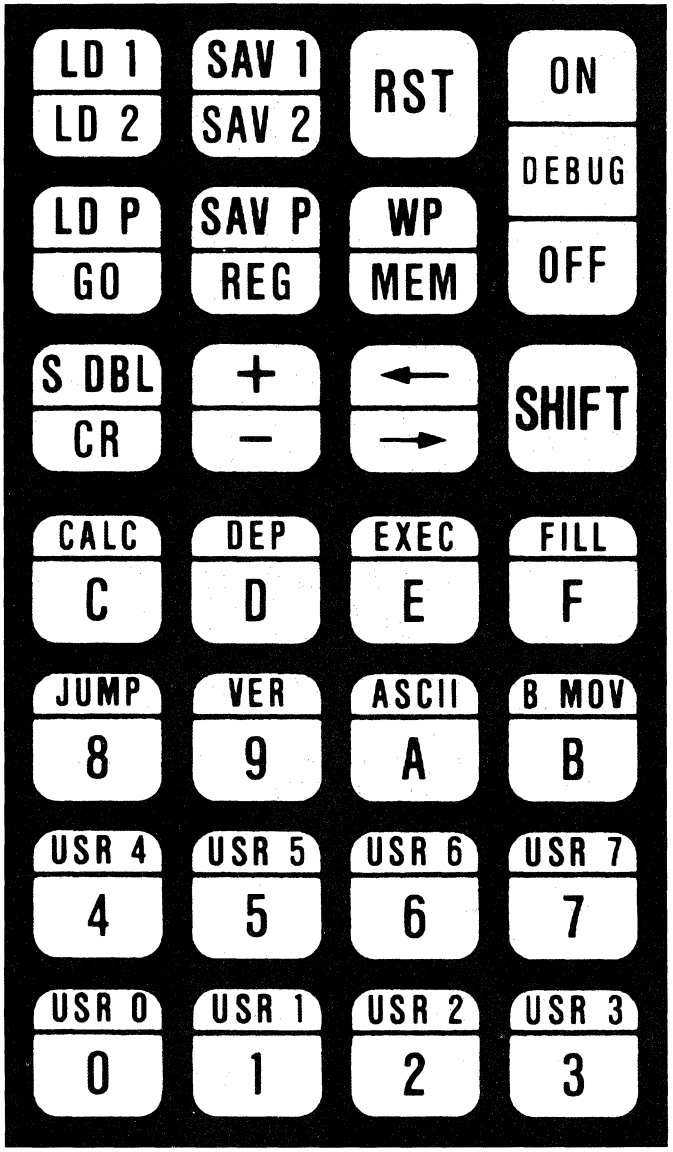


Figure 5-1 SYM-1 KEYBOARD

Table 5-1 summarizes the SYM-1 command set. The first column indicates the command, in both HKB and terminal format. The values (1), (2), and (3) refer to the values of the first, second, and third parameters entered. The term "old" is used to mean the memory location most recently referenced by any of the following commands: M, D, V, B, F, SD, S1, S2, SP, L1, L2, LP. All of these commands use locations 00FE and 00FF as an indirect pointer to memory; where a reference to "old" (or (OLD) in some cases) occurs, the former value remains in the memory pointer locations 00FE-00FF.

Note that in the second column of Table 5-1 we have provided you with the ASCII code for each instruction. Several of the commands do not have associated ASCII codes and use instead a computed "hash code." Hash codes are marked with an asterisk. You need not concern yourself with the means by which the hash code is determined, but you should note that SYM will display these values when the commands are entered with an incorrect syntax, i.e., if you make an error when entering these commands.

Table 5-2 provides you with a brief summary of the additional keys found on the on-board keyboard of the SYM-1. These are operational and special keys which do not generally have parameters associated with them, with the exception of the special user-function keys.

In the discussion of each monitor command which follows, the same basic format is followed. First, the appropriate segment of Table 5-1 is reproduced, for easy reference. Next, the command is described in some detail. Examples are used where they will make understanding the monitor command easier.

Because it is believed that most users of the SYM-1 will ultimately use a TTY to enter and obtain printouts of instruction strings, the remainder of Chapter 5 is designed to use the TTY keyboard function designations rather than those of the on-board keyboard. Remember, though, that both keyboards are functionally the same as far as SUPERMON is concerned. For this reason, we are also using a comma as a delimiter in the command string; the minus sign on the on-board keyboard (or, for that matter, on the TTY or CRT keyboard) may also be used for this purpose.

The examples provided were entered from a terminal device. When entering commands from the HKB, remember to use the (-) key instead of a comma to delimit parameters.

Table 5-1. SYM-1 COMMAND SUMMARY

Command	Code	Number of Associated Parameters			
HKB/TTY	ASCII	0	1	2	3
MEM M	4D	Memory Examine and modify, begin at (OLD)	Memory Examine and modify, begin at (1)	Memory Search for byte (1), in locations (OLD) - (2)	Memory Search for byte (1), in locations (2) - (3)
REG R	52	Examine and modify user registers PC, S,F,A,X,Y			
GO G	47	Restore all user registers and resume execution at PC	Restore user registers except PC = (1) S = FD, monitor return address is on stack		
VER V	56	Display 8 bytes with checksum beginning at (OLD)	Display 8 bytes with checksum beginning at (1)	Display (1)-(2), 8 bytes per line, with addresses and cumulative checksums	
DEP D	44	Deposit to memory, beginning at (OLD). CRLF/ address after 8 bytes, auto spacing	Deposit to memory, beginning at (1)		
CALC C	43		Calculate 0-(1) or two's complement of (1)	Calculate (1)-(2) or displacement	Calculate (1)+(2)-(3) or displacement with offset
BMOV B	42				Move all of (2) thru (3) to (1) thru (1)+(3)-(2)

* HASHED ASCII CODE

Table 5-1. SYM-1 COMMAND SUMMARY (Continued)

Command	Code	Number of Associated Parameters			
HKB/TTY	ASCII	0	1	2	3
JUMP J	4A		Restore user registers except PC= entry (1) of JUMP TABLE, S=FD, monitor return on stack		
SDBL SD	*10			Store high byte of (1) in (2) + 1 then lo byte of (1) in (2), good for changing vectors	
FILL F	46				Fill all of (2) - (3) with data byte (1)
WP W	57		Write protect user RAM according to lo 3 digits of (1)		
LD1 L1	*12	Load first KIM format record found into locations from which it was saved	Load KIM record with ID = (1) into locations from which it was saved	(1) must = FF load first KIM record found, but start at location (2)	
LD2 L2	*13	Load first hi speed record found into locations from which it was saved	load hi speed record with ID = (1)		(1) must = FF load first hi speed record found into (2) - (3)
LDP LP	*11	Load data in paper tape format. To signal end of file for tape without EOF record, type ;00 **CR** in on-line mode.			

* HASHED ASCII CODE

Table 5-1. SYM-1 Command Summary (Continued)

Command	Code	Number of Associated Parameters			
HKB/TTY	ASCII	0	1	2	3
SAVP SP	*1C			Save data from locations (1) - (2) in paper tape format. To create end of file record, unlock punch, switch to local mode, lock punch, type ;00 **CR**	
SAV1 S1	*1D				Save cassette tape locations (2) - (3) with ID = (1) KIM format
SAV2 S2	*1E				Save cassette tape locations (2) - (3) with ID = (1) hi speed format
EXEC E	45		Get monitor input from RAM, starting (1)	Get monitor input from RAM, starting (2) and store (1) for later use	Get monitor input from RAM, starting (3) and store (1) and (2) for later use.

* HASHED ASCII CODE

Table 5-2. OPERATIONAL AND SPECIAL KEY DEFINITION (ON-BOARD KEYBOARD ONLY)

Key	ASCII or *Hash Code	Description/Use
CR	OD	Carriage Return (terminates all command strings)
+	2B	Advance eight bytes
-	2D	Retreat eight bytes; also used to delimit parameters
→	3E	Advance one byte or register
←	3C	Retreat one byte
USRO	*14	All USR keys transmit the indicated Hash Code when entered as a command. The same hash codes can be sent from another terminal by entering UO (two characters, no spaces) through U7 as commands. These functions are not defined in SUPERMON and will cause the monitor to vector through the unrecognized command vector. See Chapter 9 for instructions on using this SUPERMON command feature to program your own special functions.
USR1	*15	
USR2	*16	
USR3	*17	
USR4	*18	
USR5	*19	
USR6	*1A	
USR7	*1B	
SHIFT	None	Next key entered is upper position of the selected key.
RST	None	System RESET. System RAM reinitialized to default values
DEBUG ON	None	Turn hardware Debug function "ON"
DEBUG OFF	None	Turn hardware Debug function "OFF"
ASCII	None	Next two keys entered (Hex) will be combined to form one ASCII character (e.g., SHIFT ASCII 4 D followed by a CR is the same as MEM followed by a carriage return).

* HASHED ASCII CODE

5.3 SYM-1 MONITOR COMMANDS

5.3.1 M (Display and/or Modify Memory)

Number of Associated Parameters			
0	1	2	3
Memory Examine and modify, begin at (OLD)	Memory Examine and modify, begin at (1)	Memory Search for for byte (1), in locations (OLD)-(2)	Memory Search for byte (1), in locations (2)-(3)

- The standard form for this command uses **one parameter** and is shown below.

M addr **CR**

SUPERMON will then display the address and the byte contained in the location "addr." The following options are then available:

1. Enter 2 Hex digits: bb is replaced and the next address and byte are displayed.

2. Enter colon (from terminal) and any character: bb is replaced with the ASCII code for the entered character.

3. Enter→or←(>or<from terminal): bb is left unchanged and addr+1 or addr-1, with its contents, is displayed.

4. Enter + or - : bb is left unchanged and addr+8 or addr-8 with its contents, is displayed.

5. Enter **CR** : Return to monitor command mode; bb unchanged.

- Another form of the display memory command uses **no parameter** as shown below:

M CR

This will cause SYM-1 to resume memory examine and modify at (OLD).

- The same memory (M) key may be used to search for a particular byte in memory, using **three parameters** in this form:

M bb,addr1,addr2 **CR**

This instructs the system to search for byte bb from addr1 to addr2. When an occurrence of bb is found, the location and contents are displayed, and all of the standard **M** options described above become available. In addition, a "G" entered following any halt will continue the search.

- Similarly, the **two parameter** sequence:

M bb,addr **CR**

will resume memory search for byte bb from (OLD) to addr.

The following examples demonstrate the various uses of memory display/modify commands. Characters entered by the user are underlined.

One Parameter

```
.M ↵                    Display memory location (OLD); return to Monitor
0215 BB ↵
.
```

```
.M A656↵                Display memory location A656
A656 00 0A              Put some data there; return to Monitor
A657 40 ↵
.
```

```
.M 200↵                 Display memory location 200
0200 10 :A              Replace data with ASCII code for A
0201 00 :B              Next location displayed; replace data with ASCII B
0202 00 :C              Next location displayed; replace data with ASCII C
0203 20 ↵               Return to Monitor
.
```

```
.M 0200↵                Display memory location 200
0200 41 >               Display next location; data unchanged
0201 42 >               Display next location; data unchanged
0202 43 _               Use space bar for same purpose as arrow
0203 20 _
0204 AF ↵               Return to Monitor
.
```

```
.M 0300 ↵               Display memory location 300
0300 B4 <               Display previous location; data unchanged
02FF BB <
02FE 44 <
02FD BB ↵               Return to Monitor
.
```

```
.M 0200 ↵               Display memory location 200
0200 41 +               Advance 8 bytes and display memory
0208 F0 _               Space used to advance one location; data unchanged
0209 06 _
020A 20 -               Reverse 8 bytes and display memory
0202 43 ↵               Return to Monitor
.
```

```
.M 0200↵                Display memory location 200
0200 41 ↵               Return to Monitor
.M↵                     Display (OLD) which is still 200
0200 41 ↵               Return to Monitor
.
```

```
.M 6C,8000,8400↓                    Search for 6C in range 8000-8400
801F,6C,-
8017,29,↓
.V↓
8017 29 10 F0 07 68 AA 68 28,D2
 02D2
.M 6C,8400↓                         Continue search
801F,6C,_
8020,F6,_
8021,FF,G                           Continue search
8026,6C,_
8027,F8,_
8028,FF,↓                           Halt search
.
```

5.3.2 R (Display and/or Modify User Registers)

Number of Associated Parameters			
0	1	2	3
Examine and modify user registers PC,S,F,A,X,Y			

- The only pre-defined form of this command is with **no parameters**, i.e.:

R CR

As soon as the command is entered, the contents of the PC are displayed as follows:

P 8B4A,

Using a forward arrow (→ or >), or a space, you may examine the next register. Registers are displayed in the order PC, S, F, A, X, Y, with wrap-around (i.e., PC is displayed after Y). Each register is named on the display or TTY printout; the letter X is displayed as ↳ .

To modify the displayed register, enter two or four digits (four only in the case of the PC). The register will be automatically modified and the next will be displayed. A **CR** will cause control to return to the monitor.

In the following example, we have modified the contents of the PC register to become 0200, and the A register to be set to 16. The other registers are not modified and at the conclusion of the complete register cycle and redisplay of PC, a **CR** is used to return to monitor command mode.

```
.R↓
P 8B4A, __                    Display registers
S   FF, __                    PC; space is used to advance
F   00, __
A   00, __
X   00, __
Y   00, __
P 8B4A, ↓                     PC redisplayed; return to Monitor
.

.R↓
P 8B4A,0200
S   FF,>
F   00,_
A   00,16                     Alter PC = 200, A = 16
X   00,↓
.
```

5.3.3 G (GO)

Number of Associated Parameters			
0	1	2	3
Restore all user registers and resume execution at PC	Restore user registers except PC = (1) S = FD; monitor return address is pushed onto stack		

- The GO command may be used with **no parameters** to restore all user registers and begin execution at PC:

G CR

- With **one parameter**, the command will restore user registers except that PC is set to addr, S is set to FD and SUPERMON's return address is pushed onto the stack. Thus, if a subroutine return is executed, it will result in a return to monitor command mode (with the user's stack not saved). Its format is as follows:

G addr **CR**

5.3.4 V (VERIFY)

Number of Associated Parameters			
0	1	2	3
Display 8 bytes with checksum beginning at (OLD)	Display 8 bytes with checksum beginning at (1)	Display (1)-(2), 8 bytes per line, with addresses and cumulative checksums	

- Pressing of the BREAK key on a CRT or any key on the HKP will stop printing without an error message.

- With **one parameter**, this command will result in the display of 8 bytes beginning at addr, with checksum. The format is as follows:

V addr CR

In this example, bytes stored in locations 200-207 are displayed, along with their checksum:

```
.V 200↓
0200 41 42 43 20 AF 88 C9 0D,F3
 02F3
.
```

Note that on the on-board display, only the two-byte checksum will be visible.

The checksum is a 16-bit arithmetic sum of all of the data bytes displayed. The low byte is displayed on the data line, and the full checksum on the next. The address is not included in the checksum.

- With **no parameters**, the command will display 8 bytes beginning at (OLD).

V CR

```
.V↓
0200 41 42 43 20 AF 88 C9 0D,F3
 02F3
.
```

- With **two parameters**, the "V" command will display memory from addr1 through addr2. Eight bytes per line are displayed, with cumulative checksums. A single byte checksum is included on each data line, and a final two-byte checksum is printed on a new line.

V addr1,addr2 CR

```
.V 8000,8015↓
8000 4C 7C 8B 20 FF 80 20 4A,5C
8008 81 20 71 81 4C 03 80 08,C6
8010 48 8A 48 BA BD 04,5B
 085B
.
```

5.3.5 D (Deposit)

Number of Associated Parameters			
0	1	2	3
Deposit to memory, beginning at (OLD), CRLF/address after 8 bytes, auto spacing	Deposit to memory, beginning at (1)		

- This command is used for entering data to memory from a terminal. With **one parameter**, this command instructs the system to output a **CR** and line feed and print addr. As each two-digit byte is entered, a space is output. If you enter a space (instead of a two-digit byte), you will cause two more spaces to be output, and that memory location will remain unchanged. ASCII data may be entered with the colon, as in the M command.

D addr CR

```
.D 200↓
0200 A9 3A 85 46 20 13 08 20
0208 EE 08 85 44 84 45 C6 46
0210 D0 F2 60 ↓
.
```

- As with other commands, the "D" with **no parameters** will deposit beginning at (OLD).

D CR

Notice that V and D line up, so that a line displayed with V may be altered with D, as shown below:

```
.V 200↓                                  Verify contents of 0200-0207
0200 A9 3A 85 46 20 13 08 20-09
 0209                                    Checksum
.D↓                                      Deposit memory from 0200; space to
0200 _ 0D _ 45 _ 80 03 ↓                 advance

.V 200↓                                  Re-verify contents of 0200-0207
0200 A9 0D 85 45 20 80 03 20-43
 0243                                    New checksum
.
.D 200↓                                  Deposit ASCII data at 200
0200 :A :B :C :D↓
.V 200↓
0200 41 42 43 44 73 61 6D 70-BB
 02BB
.
```

5.3.6 C (Calculate)

Number of Associated Parameters			
0	1	2	3
	Calculate 0-(1), the two's complement of (1)	Calculate (1)-(2) or displacement	Calculate (1)+(2)-(3) or displacement with offset

This command is used to do Hexadecimal arithmetic. It is very useful in programming to compute branch operands required for SY6502 instructions.

- With **one parameter**, it calculates 0 minus addr (i.e., the two's complement).

C addr **CR**

- With **two parameters**, the "C" command will calculate addr1 minus addr2 (i.e., displacement).

C addr1, addr2 **CR**

- With **three parameters**, the "C" command will calculate addr1 plus addr2 minus addr3 (i.e., displacement with offset).

C addr1,addr2,addr3 **CR**

5.3.7 B (Block Move in Memory)

Number of Associated Parameters			
0	1	2	3
			Move all of (2) thru (3) to (1) thru (1)+(3)-(2)

- This command is only defined for **three parameters** and is demonstrated by the following examples:

```
.B 200,300,320↓
.
```

Move 300 thru 320 to 200 thru 220.

```
.B 200,220,250↓
.
```

Move 220 thru 250 to 200 thru 230. No data is lost, even though the regions overlap.

```
.B 220,200,230↓
.
```

Move 230 thru 200 to 250 thru 220. (Note that this move occurs in the opposite direction. No data is lost.)

5.3.8 J (JUMP)

Number of Associated Parameters			
0	1	2	3
	Restore user registers except PC=entry (1) of JUMP TABLE, S=FD, monitor return pushed on stack		

- This command is only defined for **one parameter.**

J n CR

The parameter, n, must be in the range 0-7. All user registers are restored, except PC is taken from the JUMP TABLE in System RAM, and S=FD. The monitor return address is pushed onto the stack.

(Because the monitor return is on the stack, a JUMP to a subroutine is allowable.)

Note also that certain useful default addresses are inserted in the JUMP TABLE at Reset. (See Memory Map.)

5.3.9 SD (Store Double Byte)

Number of Associated Parameters			
0	1	2	3
		Store high byte of (1) in (2)+1 then low byte of (1) in (2). Good for changing vectors	

- This command is defined only for **two parameters** and is most useful for changing vectors.

SD addr1,addr2 **CR**

The example below was used to enter the address of the Hex keyboard input routine into INVEC, in correct order (low byte-high byte). Note that this vector could not have been altered with M, because after one byte had been altered, the vector would have pointed to an invalid address.

```
.SD 89BE.A661↓
.
```

5.3.10 F (Fill)

Number of Associated Parameters			
0	1	2	3
			Fill all of (2)-(3) with data byte (1)

- Defined only for **three parameters**, this command will fill the defined region of memory (addr1-addr2) with a specified byte (bb).

F bb,addr1,addr2 **CR**

For example:

```
.F EA,200,300↓
.
```

Fill the region 200 thru 300 with the byte EA, which is a NOP instruction.

5.3.11 W (Write Protect)

Number of Associated Parameters			
0	1	2	3
	Write protect user RAM according to 3 digits of (1)		

- This command is defined for only **one parameter.** To unprotect all of user RAM, the command is:

W 0 CR

Its general form is:

W $d_1d_2d_3$ CR

Where each of d_1, d_2, d_3 are the digits 0 (unprotect) or 1 (protect).

d_1	= 400-7FF	1K above first K of RAM
d_2	= 800-BFF	2K above first K of RAM
d_3	= C00-FFF	3K above first K of RAM

For example

```
.W 101↓
.
```

1 protect 400-7FF
0 unprotect 800-BFF
1 protect C00-FFF

Note that write protect applies to extended user RAM on-board, and also that it requires a jumper insertion (see Chapter 4).

5.3.12 E (Execute)

Number of Associated Parameters			
0	1	2	3
	Get monitor input from RAM, starting at (1)	Get monitor input from RAM, starting at (2) and store (1) for later use at A64C	Get monitor input from RAM, starting at (3) and store (1) and (2) for later user at A64E and A64C

- The standard form of the execute command uses **one parameter.**

E addr **CR**

SUPERMON adjusts its INPUT vectors to receive its input from RAM, beginning at addr. It is assumed that the user has entered a string of ASCII codes into RAM locations beginning at addr, terminated by a byte containing 00. When 00 is encountered, input vectors will be restored. The easiest way to enter these codes is to use the M command with the single-quote option (Section 5.3.1).

When E is used with **two** or **three parameters,** the additional parameters will be stored in system RAM at A64C and A64E. It is the user's responsibility to interpret them. (Note that the E command is vectored; see Chapter 9.)

```
.D 300
0300 :C :F :F :F :E :, 00 ↓
.
```

The sequence at 300 is part of a commonly used Calculate routine.

```
.E 300↓
.C FFFE,200,280↓
 FF7E
.
```

Notice that part of this C command came from RAM, and part was entered at the terminal.

5.4 CASSETTE AND PAPER TAPE COMMANDS

The SYM-1 handles cassette I/O in two formats, KIM-compatible format (8 bytes/sec), and SYM high-speed format (185 bytes/sec).

The S1 and L1 commands refer to KIM format, while the S2 and L2 commands refer to SYM high-speed format.

With each Save command you specify a two-digit ID, as well as starting and ending addresses. The ID, the addresses, and the contents of all memory locations from starting to ending address, inclusive, will be written to tape. Each Save command will create one **RECORD.**

You should be careful to assign unique ID's to different records on the same tape, and to label the tape with the ID's and addresses of all the records it contains.

While SYM is searching for a record or trying to synchronize to the tape, an "S" will be lit in the left-most digit of the display on the on-board keyboard. If the "S" does not turn off, SYM is unable to locate or to read the requested record.

5.4.1 S1, S2 (Save Cassette Tape)

Number of Associated Parameters			
0	1	2	3
(S1)			Save cassette tape, locations (2) - (3) with ID = (1) in KIM format
(S2)			Save cassette tape, locations (2) - (3) with ID = (1) in High Speed format

- These commands are discussed together, as their syntax is identical. Recall that S1 refers to KIM format while S2 refers to SYM high-speed format.

Both are defined only for **three parameters.**

S2 bb,addr$_s$,addr$_e$ **CR**

The first parameter is a 2-digit ID, which may be any value other than 00 or FF. It is followed by the starting address and the ending address. In the example below, all memory locations from 0200 thru 0280, inclusive are written to tape, and given the ID 05.

```
.S1 5,200,280
.
```

5.4.2 L2 (Load High-Speed Format Record)

Number of Associated Parameters			
0	1	2	3
Load first Hi Speed record found into locations from which it was saved	Load Hi Speed record with ID = (1)		(1) must = FF. Load first Hi Speed record found into (2) - (3)

- The standard form of this command uses **one parameter**, as follows:

L2 bb **CR**

The parameter bb is the ID of the record to be loaded. When found, the record ID will be displayed and the record will be loaded into memory, using the addresses saved in the record itself.

If the record bb is not the first high-speed record on the tape, the "S." light will go out as SYM reads through, but ignores, the preceding records. After each unselected record is read, the record ID will be displayed.

- With **no parameters** (or a single parameter of zero), the instruction will load the first high-speed format record found, without regard to its ID, using the addresses saved in the record itself.

L2 CR

or

L2 0 **CR**

- The L2 command exists in a third form, using **three parameters**, as follows:

L2 FF,addr1,addr2, **CR**

This usage will load a record into a **different** area of memory from where it was saved. The first parameter **must** be FF, followed by the requested starting and ending address. It is your responsibility to supply addr1 and addr2 such that their difference is the same as the difference of the addresses used to save the record.

5.4.3 L1 (Load KIM Format Record From Tape)

Number of Associated Parameters			
0	1	2	3
Load first KIM format record found into locations from which it was saved	Load Kim record with ID = (1) into locations from which it was saved	(1) must = FF. Load first KIM record found, but start at location (2)	

- The L1 command, used with **zero or** with **one parameter**, is identical in syntax to the L2 command (see Section 5.4.2, above).

- With **two parameters**, the L1 command is used to load into a **different region of the memory than that with which the record was saved.**

L1 FF,addr **CR**

The first parameter **must** be FF, followed by the requested starting address. No ending address is necessary, as the load operation will halt when the end of the record is found.

5.4.4 SP (Save Paper Tape)

Number of Associated Parameters			
0	1	2	3
		Save data from locations (1) - (2) in paper tape format. To create end of file record, unlock punch, switch to local mode, lock punch, type ;00 **CR**	

- Defined only for **two parameters**, this command will output data from RAM in paper tape format (see Appendix D).

SP addr1,addr2 **CR**

For example:

```
.SP 200.215↓
;10020034AB743B44BB44BB44BB44BB44BB44BB079A
;060210AC1BF49BD4BB03FD
.
```

5.4.5 LP (Load Paper Tape)

Number of Associated Parameters			
0	1	2	3
Load data in paper tape in format. To signal end of file for tape without EOF record, type ;00 **CR** in on-line mode			

- This command is defined for **no parameters** only. It will load memory with data in paper tape format (see Appendix D).

LP CR

5.5 USER-DEFINED FUNCTIONS

You may, as we have previously pointed out, write programs to be called from the on-board keyboard. You may do this by using any combination of command and number of parameters which is not already defined (e.g., B MOV with only two parameters) or by using any or all of the eight keys along the bottom two rows of the on-board keyboard (those labeled "USR 0" through "USR 7"). The exact means of implementing these special functions is discussed in detail in Chapter 9.

5.6 ERROR CODES

The SYM-1 microcomputer system handles error codes in an interactive way, with codes being designed to be determined by the context in which the error occurs. No table of error conditions and their meanings is therefore provided with this manual, since these are context dependent.

However, you should be aware of the general method by which errors are handled by your SYM-1 system.

When your SUPERMON encounters an error of some type, it displays a 2-digit representation of the byte which was being processed when the error was detected. For example, if you attempt to carry out a CALC command with no parameters (and you haven't defined such a routine yourself as explained in Chapter 9), the system will display a "43." which is the ASCII representation for the "C" which represents the CALC function.

Similarly, if you attempt to use an ID of 00 or FF with either SAV1 or SAV2, the system will display the ID used in error.

After the "er" message is printed, a new prompt (decimal point) is displayed, and SUPERMON waits for a new command. Note that you do not need to RESET when an error condition occurs, since that results in System RAM being cleared and necessitates a re-start of your routine. It is also worth noting that when you carry out an EXEC command at the on-board keyboard the system does not halt when an error occurs; rather, it continues in the same fashion as if new commands were coming directly from the keyboard. The error condition therefore flashes too rapidly on the LED display for you to see it. Command sequences to be executed by EXEC should be pretested prior to such use.

Some fixed error codes do exist in the monitor. Four such codes are used in audio cassette operations and are defined in Table 5-3. Additionally, if in carrying out LD P, FILL or B MOV commands you either attempt to store data in a non-existent or WRITE-protected memory location or if during execution of one of these commands a memory error occurs, the LED display will show the number of locations read incorrectly. This number will always stop at "FF" if it exceeds that number, so that the display will have some intelligible meaning.

Table 5-3. ERROR CODES IN AUDIO CASSETTE OPERATIONS

Code Displayed	Meaning
2F	Last-character error. The last character in a tape record should be a 2F. If that is not the case, the system displays the error code shown.
CC	Checksum error. Usually indicates data transfer problems. Re-position the tape and try again.
FF	In KIM-1 format loading, this error code means a non-Hexadecimal character has been encountered. This almost always means a synchronization error. Restart the procedure.
	In High-Speed format loading, a framing (i.e., synchronization) error is the cause. Restart the procedure.

The following examples provide some representative errors to enable you to become familiar with how they are reported on SYM-1 using a TTY or CRT.

```
.W 111↵
.F EA,300,400↵
ER 01
.
```

Memory location 400 write protected, therefore it could not be modified. One byte only in error.

```
.S2 200,280↵
ER 1E
.
```

S2 is not defined for two parameters. The hash code for S2 is 1E.

```
.C A,230,500,
ER 2C
.
```

Three parameters only permitted.

```
.C 200,2X
ER 58
```

X is not a valid Hex digit.

```
.S2 FF,200,280↵
ER FF
.
```

ID may not be FF or 00.

```
.L2 AA,200,280↵
ER AA
.
```

ID must be FF.

```
.M 6000↵
6000,60,F5?
6001,60,↵
.
```

No RAM at 6000, therefore it cannot be modified.

```
.D 8000↵
8000 AA? DD? ↵
.
```

ROM at 8000, therefore it cannot be modified

```
.D 200,280↵
ER 44
.
```

Deposit not defined for 2 parameters. D = ASCII 44.

```
.F EA,5000,6000↵
ER FF
.
```

No RAM at locations 5000-6000, therefore no modification was possible. The number of bytes which were not correctly changed is greater than or equal to 255 (decimal).

CHAPTER 6

PROGRAMMING THE SYM-1

Creating a program on the SYM-1 involves several steps. First, the input to the program and its desired output must be carefully defined. The flow of program logic is usually worked out graphically in the form of a flowchart. Next, the symbols on the flowchart are converted to assembly language instructions. These instructions are in turn translated into machine language, which is entered into memory and executed. If (as usual) the program does not run correctly the first time, you must debug it to uncover the errors in the program. This chapter illustrates the steps involved in creating a program to add two 16-bit binary numbers, and provides two other programming problems with suggested solutions. All three programs are designed to communicate basic programming principles and techniques and to demonstrate a programmer's approach to simple problems.

6.1 HARDWARE

All the sample programs listed here can be loaded and run on the basic SYM-1 with the minimum RAM. The only I/O devices required are the on-board keyboard and display.

If a printing or display terminal is available, by all means use it instead of the Hex keyboard provided. Both types are more comfortable for most users and allow much more data to be displayed at once.

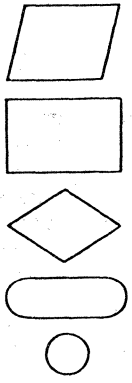
Connect the terminal cable to the appropriate connector on the left edge of the card as described in Chapter 3. Verify that the switches on the terminal are set for full-duplex operation and no parity. The duplexing mode switch will usually be labelled HALF/FULL or H/F; the parity switch will be labelled EVEN/ODD/NO. If your terminal has a CRT, wait for it to warm up. To log on to a terminal, enter a "Q" immediately after reset.

6.2 DOUBLE-PRECISION ADDITION

Since the eight bits of the accumulator can represent positive values only in the range 0-255 (00-FF Hex), 255 is the largest sum that can be obtained by simply loading one 8-bit number into the accumulator and adding another. But by utilizing the Carry Flag, which is set to "1" whenever the result of an addition exceeds 255, multiple-byte numbers may be added and the results stored in memory. A 16-bit sum can represent values greater than 65,000 (up to FFFF Hex). Adding 16-bit rather than 8-bit numbers is called "double-precision" addition, using 24-bit numbers yields triple precision, etc.

6.2.1 Defining Program Flow

Flowcharting is an orderly way of representing a procedure. Much easier to follow than a list of instructions, a flowchart facilitates debugging and also serves as a handy reference when using a program written weeks or months earlier. Some common flowcharting symbols are shown in Figure 6-1. below.



INPUT/OUTPUT

PROCESSING

DECISION

TERMINATOR

ON PAGE CONNECTOR

Figure 6-1. COMMON FLOWCHARTING SYMBOLS

The object of our program is to add two 16-bit numbers, each stored in two bytes of RAM, and obtain a 16-bit result. The sequence of operations the processor must perform is shown in the flowchart in Figure 6-2.

To accomplish double-precision addition, first clear the Decimal Mode and the Carry Flags. (The addition is in binary, so the system must not be expecting decimal numbers. The Carry Flag is used in the program and must start at zero.) Load the low byte of the first 16-bit number into the accumulator and add the low order byte of the second number using an Add With Carry (ADC) command. The contents of the accumulator are the low order byte of the result. The Carry Flag is set if the low-byte sum was greater than FF (Hex).

You now store the accumulator contents in memory, load the high order byte of the first number into the accumulator, and add the high order byte of the second number. The ADC command automatically adds the carry bit if it is set. After the second addition, the contents of the accumulator are the high order byte of the result. The example below shows the addition of 384 and 128.

```
      0000  0001  1000  0000     384  (0180 Hex)
      0000  0000  1000  0000     128  (0080 Hex)

      Add low order bytes:  (clear carry)

            1000  0000
            1000  0000
Carry = 1   0000  0000

      Add high order bytes:  (carry = 1)

            0000  0001
            0000  0000
                     1  CARRY
Carry = 0   0000  0010

   Result = 0000 0010  0000  0000 = 512  (0200 Hex)
```

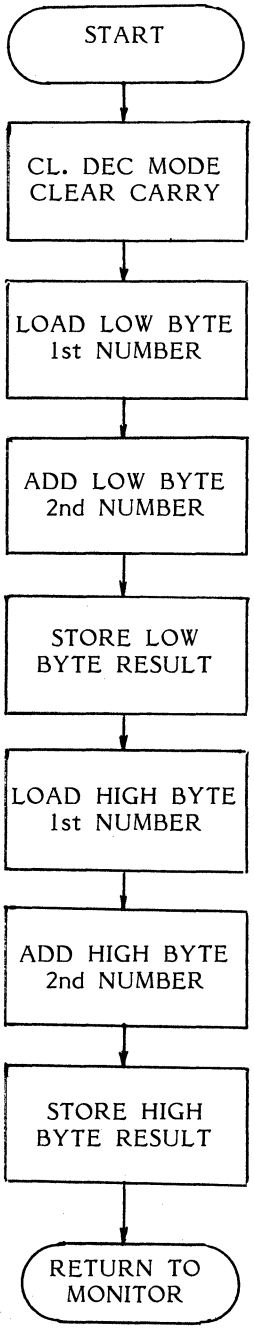


Figure 6-2. DOUBLE-PRECISION ADDITION FLOWCHART

6.2.2 Coding and "Hand Assembly"

Once you have flowcharted a program, you may "code" it onto a form like the one shown in Figure 6-3. SY6502 Microprocessor Assembly Language is described in Sections 4.3.3 and 4.3.4. Additional information is available in the Synertek "Programming Manual" (MNA-2) for the 6500 family. Figure 6-4 shows the coding for our example.

The first step involves finding the SY6502 commands that correspond to the operations specified in the flowchart. A summary of the commands and their mnemonic codes is given in Table 4-7. Arbitrary labels were assigned to represent the addresses of the monitor, the two addends and the sum and entered in the operand field. As written, the assembly language program does not specify where in memory the program and data will be stored.

To store and execute the program, you must "assemble" it by translating the mnemonics into hexadecimal command codes and assign the program to a set of addresses in user RAM. Performing this procedure with pencil and paper, rather than with a special assembler program, is "hand assembly".

The SUPERMON monitor begins at Hex location 8000, and the addends and the sum have been arbitrarily assigned to locations 0301 through 0306. You should note that the high and low order bytes of a 16-bit number need not be stored in contiguous locations, although they are in this example.

The program will be stored beginning in location 0200, another arbitrary choice. Data and programs may be stored anywhere in user RAM. Columns B1, B2, and B3 represent the three possible bytes in any 6502 instruction. B1 always contains the Hexadecimal operation code. B2 and B3 represent the operand(s). Looking at the coding form, you can see that the CLD and CLC instructions each occupy one byte and that the LDA instruction occupies three bytes. On your instruction set summary card, you'll see that the LDA mnemonic represents several different operation codes depending on the addressing mode chosen. AD indicates absolute addressing and specifies a three-byte command. When all the operation codes and operands have been translated into pairs of Hex digits, the program is ready to be entered into memory and executed.

6.2.3 Entering and Executing the Program

The procedure for entering the double precision addition program is shown below.

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(RST)		
(CR)	SY1.1..	
(MEM) 200 (CR)	0200.**.	Enter memory display and modify mode
D8	0201.**.	Store D8 in location 0200, advance to next location
18	0202.**.	Store 18 in location 0201, advance to next location
AD	0203.**.	.
02	0204.**.	.
03	0205.**.	.
6D	0206.**.	.
	.	
	.	
	.	
80	0217.**.	
(CR)	217.**..	Exit memory display and modify mode

SYNERTEK SYSTEMS CORPORATION

Page_____Of_____

PROGRAM_____

PROGRAMMER_____

DATE_____

ADDR	INSTRUCTIONS B1	B2	B3	LABEL	MNEMONIC	OPERAND	COMMENTS

Figure 6-3. SAMPLE CODING FORM

SYNERTEK SYSTEMS CORPORATION

Page_____ Of____

PROGRAM_____

PROGRAMMER_____

DATE_____

DOUBLE – PRECISION. ADD ROUTINE

ADDR	INSTRUCTIONS B1	B2	B3	LABEL	MNEMONIC	OPERAND	COMMENTS
200	D8				CLD		CLEAR DECIMAL MODE FLAG (MODE = 0)
201	18				CLC		CLEAR CARRY FLAG (CARRY=0)
202	AD	02	03		LDA	L1	LOAD LOW ORDER BYTE, FIRST NUMBER
205	6D	04	03		ADC	L2	ADD WITH CARRY, LOW ORDER BYTE, SECOND NUMBER
208	8D	06	03		STA	L3	STORE LOW ORDER BYTE, RESULT
20B	AD	01	03		LDA	H1	LOAD HIGH ORDER BYTE, FIRST NUMBER
20E	6D	03	03		ADC	H2	ADD WITH CARRY HIGH ORDER BYTE, SECOND NUMBER
211	8D	05	03		STA	H3	STORE HIGH ORDER BYTE, RESULT
214	4C	00	80		JMP	START	BRANCH TO MONITOR
301				H1	=	$301	HIGH ORDER BYTE OF FIRST NUMBER
302				L1	=	$302	LOW ORDER BYTE OF FIRST NUMBER
303				H2	=	$303	HIGH ORDER BYTE OF SECOND NUMBER
304				L2	=	$304	LOW ORDER BYTE OF SECOND NUMBER
305				H3	=	$305	HIGH ORDER BYTE OF RESULT
306				L3	=	$306	LOW ORDER BYTE OF RESULT
8000				START	=	$8000	MONITOR

Figure 6-4. DUAL-PRECISION ADD ROUTINE

The program is now entered. Examine each location to make sure that all values are correct. Then store the addend values in locations 0301-0304 as shown below. We'll use the numbers that were used in the example in Section 6.2.1, 0180 (Hex) and 0080 (Hex).

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(MEM) 301 (CR)	0301.**.	
01	0302.**.	Enter high order byte, first addend
80	0303.**.	Enter low order byte, first addend
00	0304.**.	Enter high order byte, second addend
80	0305.**.	Enter low order byte, second addend
(CR)	305.**..	

To execute the program, enter the command shown below.

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(GO) 200 (CR)	g 200.	Execute program starting at location 0200.

Now use MEM to examine locations 0305 and 0306. Verify that they are high and low order bytes of the result, 02 and 00. If you find other data at these locations, you will be pleased to know that the next section of this chapter tells you how to debug the program.

6.2.4 Debugging Methods

The first step in debugging is to make sure that the program and data have been entered correctly. Use the MEM command to examine the program starting address, and use the right-pointing arrow key to advance one location at a time and verify that the contents of each are correct. If you have a terminal, you can generate a listing by entering an SP command without turning on the tape punch or by using the VER command. Also examine the locations that contain the initial data.

If the program and data are correct, but the program still does not execute properly, you may want to use the SYM-1 DEBUG function. If DEBUG is ON when the execute (GO) command is entered, the program will execute the first instruction, then return control to the monitor. The address on the display will be the address of the first byte of the next instruction. If you again press GO (CR) to execute (do not specify an address this time), the computer will execute the next instruction, then halt as before. The program may be executed one step at a time in this manner.

By entering a non-zero Trace Velocity (at location A656), execution will automatically resume after a pause during which the Accumulator is displayed. Depress any key to halt automatic resumption.

After certain instructions, you will want to examine the contents of memory locations or registers. Use the MEM or REG commands, then resume operation by entering another GO command.

To examine the Carry Flag after the low order addition, for example, use the REG command as shown below.

YOU KEY IN	DISPLAY SHOWS	EXPLANATION
(ON)	unimportant	Turn DEBUG function ON
(GO) 200 (CR)	0201.2 .	Execute D8 instruction
(GO) (CR)	0202.2 .	Execute 18 instruction
(GO) (CR)	0205.2 .	Execute AD instruction
(GO) (CR)	0208.2 .	Execute 6D instruction, low order add with carry
(REG) (CR)	P 0208.	Program Counter
	r1 Fd.	Stack pointer
	r2 63.	Status register
(CR)	2 63.	End register examination
(GO) (CR)	020B.2 .	Execute 8D instruction
	.	
	.	

The Carry Flag is the lowest (rightmost) bit of the Status Register. To determine whether the flag was set, convert the Hex digits 63 to binary. The result of this conversion is 0110 0011, and since the low bit is "1", this confirms that the sum of the two low order bytes was greater than 255 (FF Hex).

You may turn the DEBUG switch OFF after any instruction. When you next press GO, the program will finish executing.

Since reading from or writing to any I/O port is the same as reading from or writing to a memory location, the DEBUG feature may also be used to debug I/O operations. When the port address is examined with a MEM command, the two Hex digits that represent data indicate the status of each line of the port. For example, if the value C2 is displayed, pin status is as follows:

PIN	7	6	5	4	3	2	1	0
STATUS	1	1	0	0	0	0	1	0

0 = Low
1 = High

For more advanced debugging techniques, including how to write and use your own trace routines, see Sections 9.5 and 9.6.

You now know how to code, enter, and debug programs on the SYM-1. Let's go look at two more examples that illustrate useful programming concepts.

6.3 CONDITIONAL TESTING

Most useful computer programs don't go in straight lines -- they don't simply execute a series of instructions in consecutive memory locations. They do perform different operations for different data by testing data words and jumping to different locations depending on the results of the test. Typical tests answer the following kinds of questions:

1. Is a selected bit of a specified data word a 1 or a 0?
2. Is a specified data word set to a selected ASCII character or numeric value?

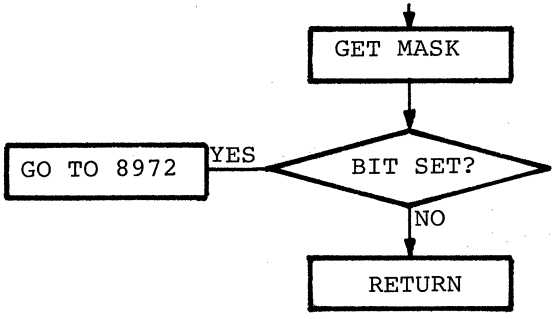
The sample program discussed below will answer question "1". It can be patched easily to answer question "2". You can use the principles you learn in the first two examples to make many more complicated tests.

Bit Testing

This sample program looks at the word in Hex location 31 and tests bit 3. If bit 3 is set to one, it jumps to location 8972; if bit 3 is zero, it returns to the executive. Location 8972 is a monitor subroutine that makes the SYM-1 go "beep".

The only problem involved is in isolating bit 3. The simplest way is to use a mask -- a word in memory with bit 3 set and none other. If we logically AND the mask with the sample word, the resultant value will be zero if bit 3 was zero and non-zero otherwise. The BIT test performs the AND and tests the value without altering the state of the accumulator.

Here is the flow chart. The code is in Figure 6-5. The mask (08 Hex) is in location 30, the test value in location 31.



Hint

If you wish to test bit 0 or bit 7 of a byte, you need not use a mask. Simply use a shift operation to place the selected bit in the CARRY status bit and use a BCC or BCS to test CARRY. This saves one or more program locations. Note that it alters the accumulator - you may have to shift it back for later processing.

Character, Value, or Magnitude Testing

To test whether a byte is exactly equal to an ASCII character or a value, use the Compare command or first set a mask location exactly equal to the character or value. Then use the EOR command to find the exclusive OR of the two values and test the result for zero. It will be zero if and only if the values were identical. Note that this destroys the test value -- keep another copy of it if you must use it again.

To test whether a byte is greater, equal to, or less than a given value, use the Compare command or set a mask to the test values and subtract it from the test value. The test value will be destroyed. Test the result to see whether it is positive, negative, or zero (this takes two sequential tests) and skip accordingly. Try writing a program that makes a series of magnitude tests to determine whether a given byte is an ASCII control character (0-1F Hex), punctuation mark, number, or letter. The values of the ASCII character set are listed on the summary instruction card.

SYNERTEK SYSTEMS CORPORATION
2589 Scott Boulevard
Santa Clara, California 95050
(408) 247-8940

Page_____ Of____

PROGRAM_____

PROGRAMMER_____

DATE_____

BIT 3 TEST ROUTINE

ADDR	INSTRUCTIONS B1	B2	B3	LABEL	MNEMONIC	OPERAND	COMMENTS
30	08			MASK			BIT 3 MASK
31	XX			TEST			(USE DIFFERENT VALUES)
200	A5	30			LDA	MASK	GET MASK
202	24	31			BIT	TEST	COMPARE AND SET ZERO BIT
204	FØ	03			BEQ	HOME	NO BIT, RETURN
206	4C	72	89		JMP	BEEP	BEEP ANNUNCIATOR
209	6Ø			HOME	RTS		RETURN

Figure 6-5. BIT 3 TEST ROUTINE

6.4 MULTIPLICATION

The sample program described here multiplies two one-byte unsigned integers and stores the results in two bytes. Note that in any base of two or more, the product of two numbers may be as long as the sum of the lengths of the numbers. In decimal, 99 X 99= 9801; in Hex FF X FF= FE01.

Since many programs will involve multiplication, it is not good practice to write a multiplication routine every time the need comes up. The sample is set up as a subroutine to allow it to be used by many programs. Serious programmers will usually wind up with libraries of subroutines specialized for their applications.

How to Multiply

Multiplication is normally introduced to students as a form of sequential addition. Humans can in fact multiply 22 (decimal) by 13 by performing an addition:

22 + 22 + 22 . . . . 22 = 286

This technique is of course foolish -- it involves a lot of work and a high probability of error. It would be easy to write a program that would multiply this way (try it) but it would be a terrible waste of time.

How then to multiply? We could use a table. Humans use memorized tables that work up to about 10 X 10:

7 X 8 = 56

Humans cannot, however, remember well enough to know that:

22 X 13 = 286

Computers, of course, can "remember" an arbitrarily large table. But the table for the problem at hand would have FFFF entries, which is far too many for practicality.

Humans solve the problem by breaking the multiplication down into smaller steps. We multiply one factor, one digit at a time, by each digit of the other factor in turn. Then we shift some of the partial products to the left and add:

```
    22
X   13
   ---
    66
   22
  ----
   286
```

We would multiply the binary equivalents of the numbers the same way:

```
   10110
    1101
   -----
   10110
      0
  10110
 10110
--------
100011110
```

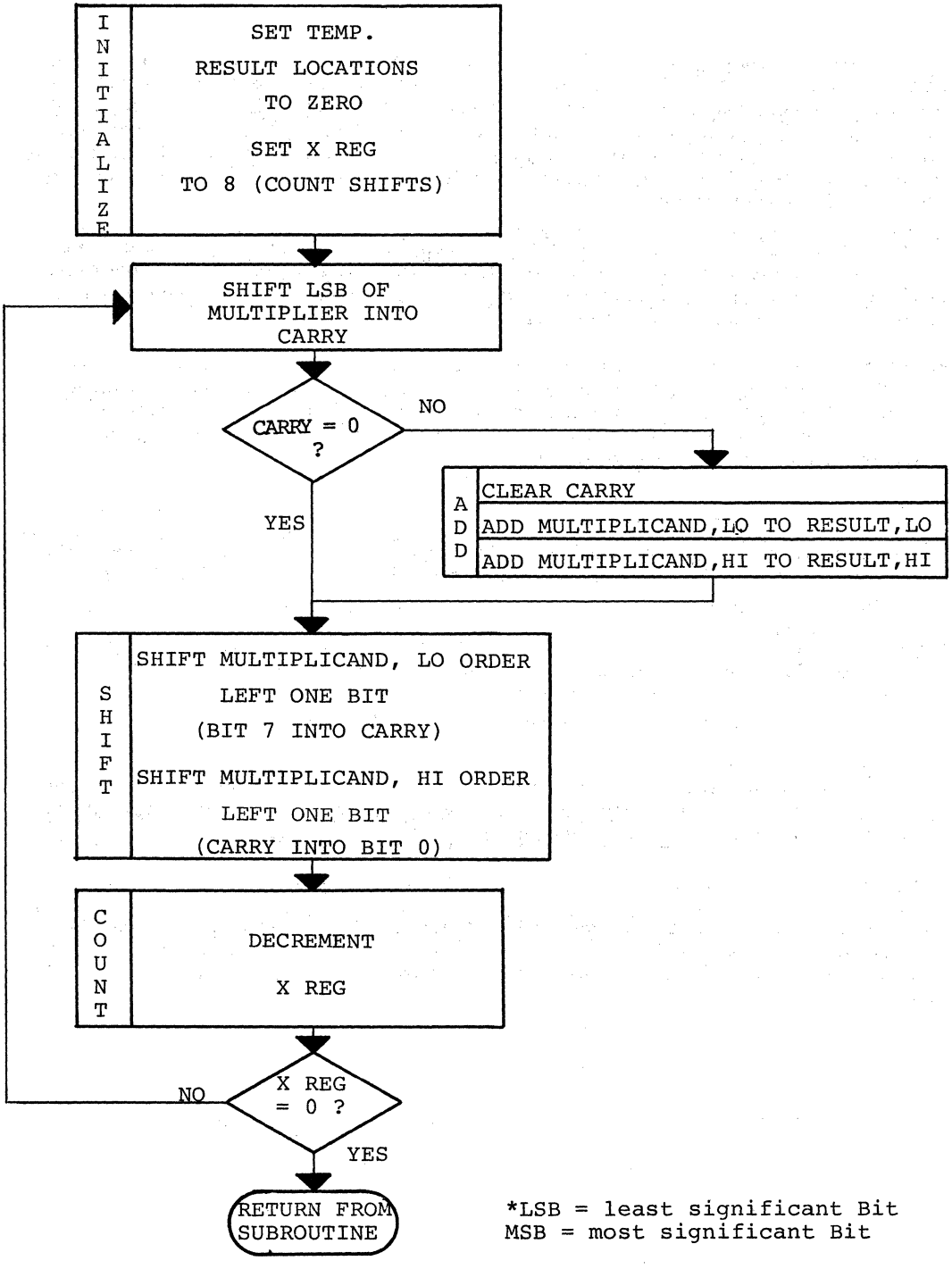


Figure 6-6. GENERAL MULTIPLICATION FLOWCHART

A little figuring will verify that the result is correct. Note that the "tables" for multiplying binary numbers by a single digit are very simple -- a number times one is itself; a number times zero is zero. We can multiply, then, by using a series of additions and shifts, as shown in the flow chart below. The first factor is eight bits long; the second is extended to two bytes (the high-order byte is zero), and the result goes into two bytes set initially to zero. The flowchart in Figure 6-6 is general and not suitable for direct coding.

This procedure could be coded quite easily. Each bit test on the first factor could be made with a different mask as shown in the previous example. Note, however, that the same basic set of instructions is repeated eight times, wasting memory space. A more efficient routine would loop over the same code eight times.

The more efficient routine could also use eight masks, but there's a simpler way. Simply shift the factor to the left once per addition. The bit to be tested will wind up in the CARRY indicator, and we can simply test that. Figure 6-7 is a more formal flowchart of the multiply routine as it is coded that it includes the coding details. The coding chart is shown in Figure 6-8.

Testing

The listing below shows one way to key in the program. The code occupies the RAM space from 200 to 222 Hex. The factor come from locations 21 and 22; the product goes to locations 23 and 24.

Note that the original factors are destroyed by the routine. If it is necessary to preserve them for other subroutines, simply copy them into unused memory locations and perform the multiplication on the copies.

Division

Try to write a parallel routine for performing integer division that divides a two-byte quotient and a two-byte remainder. You may wish to test the remainder and, if its MSB is one, round the result by incrementing the quotient.

Arithmetic

The examples given so far show some basic integer arithmetic techniques. They may be expanded easily for double-precision operation. (Multiply two bytes by two bytes for a four-bit product. Use double-precision addition and fifteen shifts instead of seven.)

MULTIPLIER = P

MULTIPLICAND = Q

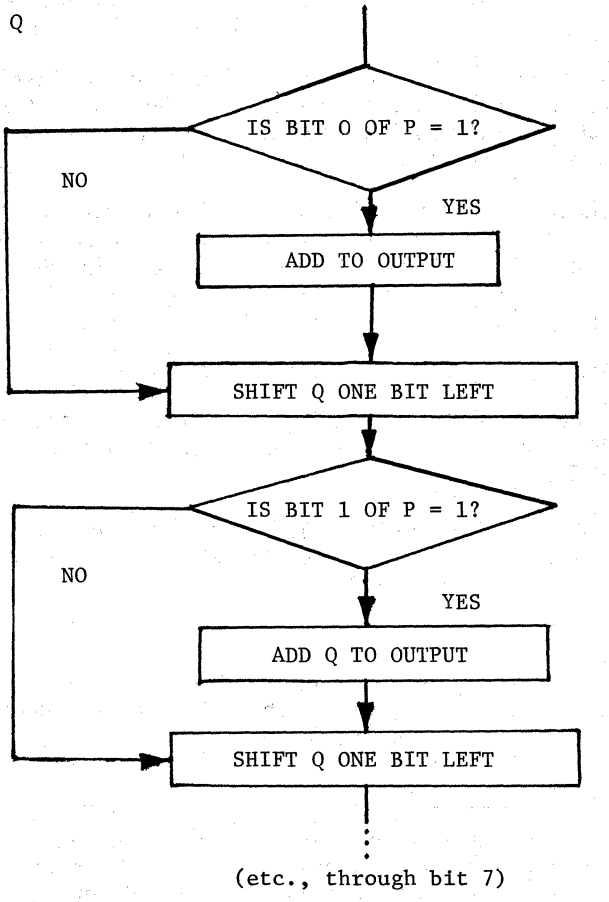


Figure 6-7. DETAILED MULTIPLICATION FLOWCHART

SYNERTEK SYSTEMS CORPORATION

Page_____ OF ____

PROGRAM _____

PROGRAMMER _____

DATE _____

SINGLE - PRECISION MULTIPLY ROUTINE

ADDR	INSTRUCTIONS B1	B2	B3	LABEL	MNEMONIC	OPERAND	COMMENTS
200	A9	00		MULT1	LDA	#0	ZERO ACCUMULATOR
202	85	20			STA	INHI (20)	SET TEMPORARY STORAGE LOCATION (20) TO ZERO
204	85	23			STA	OUTLO (23)	" LOW BYTE RESULT " (23) " "
206	85	24			STA	OUTHI (24)	" HIGH " " " (24) " "
208	A2	08			LDX	#8	SET X TO 8 TO COUNT SHIFTS
20A	46	22		MORE	LSR	IN2 (22)	SHIFT FACTOR RIGHT
20C	90	0D			BCC	ZERBIT (+D)	IF CARRY = 0 SKIP ADDITION, GO TO ZERBIT
20E	18				CLC		CLEAR CARRY
20F	A5	23			LDA	OUTLO	GET LOW BYTE ASSEMBLED SO FAR
211	65	21			ADC	IN1	ADD CURRENT TERM
213	85	23			STA	OUTLO	SAVE UPDATED LOW BYTE
215	A5	24			LDA	OUTHI	GET HIGH BYTE ASSEMBLED SO FAR
217	65	20			ADC	TEMPHI	ADD CURRENT TERM
219	85	24			STA	OUTHI	SAVE UPDATED HI BYTE
21B	06	21		ZERBIT	ASL	IN1	SHIFT LEFT FOR NEXT ADDITION
21D	26	20			ROL	INHI	SHIFT HIGH BYTE LEFT (ENTER CARRY)
21F	CA				DEX		DECREMENT INDEX REGISTER (COUNT ADDS)
220	D0	E8			BNE	MORE	IF X > 0, GO BACK AND DO NEXT ADD
222	60				RTS		DONE; GO BACK TO CALLING ROUTINE

Figure 6-8. SINGLE-PRECISION MULTIPLY ROUTINE

CHAPTER 7

OSCILLOSCOPE OUTPUT FEATURE

7.1 INTRODUCTION

Your SYM-1 module is hardware-equipped to allow you to use an ordinary oscilloscope as a display device. In this section, we will describe the hardware and connections between the system and the oscilloscope and also provide a listing of a software driver for this output. This listing is just one way of handling the oscilloscope output; you may wish to modify it or develop your own.

7.2 OPERATION OF OSCILLOSCOPE OUTPUT

The circuitry shown in the detail on the schematic (Figure 4-9) enables the SYM to output alphanumeric characters to an oscilloscope. The circuitry is adapted from a published schematic and was included on the SYM to help relieve the bottleneck found on most single-board computers, i.e., the 7 segment displays. Many things can be done with the scope-out circuit, like displaying alphanumeric characters, bar graphs, and game displays. The alphanumeric output is usually organized as 16 or 32 characters, each character being a 5-by-7 dot matrix. The characters could be English, Greek or Cuneiform, or could even be stick-men, cars, dog houses, or laser guns.

The "video" signal from the collector of Q10, is 3V peak-to-peak with a cycle time of about 50 ms (using the suggested software driver included in section 7.3). The sync pulse which begins the line should synchronize all triggered sweep scopes and most recurrent sweep scopes. In the driver which follows, sync could be brought out on a separate pin by replacing the code from SYNC to CHAR with a routine that would output a pulse on PB4 or some other output line.

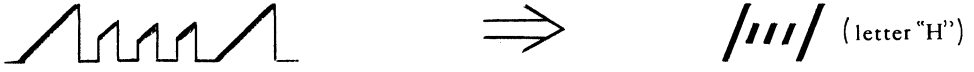
7.2.1 Connection Procedures

Connect the oscilloscope vertical input to pin R on connector AA ("scope out") and connect scope ground to pin 1 of connector AA (SYM ground). Start the software and adjust the scope for the stable 32-character display. If the sync pulse was output on PB4, connect the scope's trigger to pin 4 of connector AA.

7.2.2 Circuit Operation

The operation of the circuit is simple. Basically, the circuit is a sawtooth waveform generator whose output is sometimes the sawtooth and sometimes ground. The sawtooth is generated by the current source, Q9-Q17-R42-R43, charging C9. When C9 gets up to about 3V the discharge path, Q19-Q18-R41-R44, shorts it back to ground due to a pulse sent out by CA-2. The sawtooth waveform is shown below and forms the columns of the display.

By pulling the sawtooth to ground with Q10 any columns or portions thereof can be "removed" from the display. The result of this can be seen below:



The sawtooth is pulled to ground by bringing CB-2 high.

Because Q10 in the "ON" state will cause loading of C9 (thru R45) and C9 will charge a little more slowly, the time for a "dark" column should be slightly longer than for a "light" column.

If more than 8 vertical dots are desired, the charging rate of C9 must be slowed by lowering the charging current. R42 controls the charging current and can be increased up to about 10K before the loading effects of R45 get completely out of hand.

7.3 USING OUR SOFTWARE

The program listing in Table 7-1 is one way of handling oscilloscope output. After entering the program and character table and attaching an oscilloscope to the scope output, enter the following commands:

	Comments
.SD 500, A670**(CR)**	Change SCANVEC. (DISPLAY GOES BLANK)
.SD 58C, A664**(CR)**	Change OUTVEC.
.SD 560, A661**(CR)**	Change INVEC.

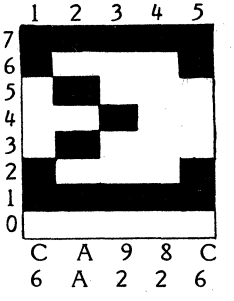
Now enter any stream of characters from the HKB to fill SCPBUF.

Put the scope input on AC couple and the trigger on DC couple. Adjust the time base, attenuation, and trigger until the display becomes readable. If your screen is very small, you may wish to change the number of characters per line by adjusting the value at location $0506.

Example: Creating translation table for scope driver.

Character: $\sum$ (Greek capital letter sigma)

Dot



Hex Equivalent

Each byte corresponds to a single column, with each bit corresponding to a single dot.

sigma = $C6, $AA, $92, $82, $C6

Bit Ø is always Ø to raise the character off of the Ground line.

Table 7-1. OSCILLOSCOPE OUTPUT DRIVER SOFTWARE LISTING

```
LINE # LOC      CODE        LINE

0002  0000                ; SCOPE LINE DRIVER 05/01/78
0003  0000                ; USES CHARACTER SET IN TABLE SYMBLS
0004  0000                ; 5 BYTES PER CHAR
0005  0000                ; ENTRY 'LINE' IS ANALOGOUS TO 'SCAND'
0006  0000                ; BELOW ROUTINES HKEY AND HDOUT INTERFACE TO HEX KB
0007  0000                ; CHAR SET PROVIDED IS FOR HEX KB
0008  0000                ; AND RELATED TO ASCII TABLE IN MONITOR ROM
0009  0000                ; THIS DRIVER CAN ACCESS A MAX OF 51 CHARS
0010  0000                TXTSHV =$8A06
0011  0000                SCNVEC =$A66F
0012  0000                GETKEY =$88AF
0013  0000                KEYQ   =$8923
0014  0000                SAVER  =$8188
0015  0000                BEEPP3 =$8975
0016  0000                ASCIM1 =$8BEE
0017  0000                RESALL =$81C4
0018  0000                PCR3   =$AC0C          ;CA2,CB2 = SCOPE
0019  0000                TXTCTR =$03FE
0020  0000                COLCTR =$03FF
0021  0000                TEXT   =$A600          ;SCOPE BUFFER IN SYS RAM
0022  0000                SYMBLS =$0400          ;CHARACTER TABLE
0023  0000                       *=$500
0024  0500  A9 EE          LINE   LDA #$EE        ;DISCHARGE CAP
0025  0502  8D 0C AC              STA PCR3
0026  0505  A9 21                 LDA #32+1      ;# CHARS PER LINE
0027  0507  8D FE 03              STA TXTCTR
0028  050A  A9 CC          SYNC   LDA #$CC        ;CHARGE CAP FOR SYNC
0029  050C  8D 0C AC              STA PCR3
0030  050F  A2 EA          LDLY   LDX #$EA        ;LONG DELAY !
0031  0511  CA                    DEX
0032  0512  D0 FC                 BNE LDLY+1
0033  0514  CE FE 03       CHAR   DEC TXTCTR      ;LOOP HERE FOR CHAR
0034  0517  AE FE 03              LDX TXTCTR
0035  051A  D0 03                 BNE POIMFG
0036  051C  4C 23 89       EXIT   JMP KEYQ        ;SCAN KB AND RETURN
0037  051F                ;
0038  051F  BD FF A5       POIMFG LDA TEXT-1,X    ;POINTER MANUFACTURER
0039  0522  0A                    ASL A          ;PTR X 4 + PTR
0040  0523  0A                    ASL A
0041  0524  18                    CLC
0042  0525  7D FF A5              ADC TEXT-1,X    ;MULT'PY BY 5
0043  0528  AA                    TAX
0044  0529  A9 06                 LDA #6
0045  052B  8D FF 03              STA COLCTR
0046  052E  A9 EE          COLUMN LDA #$EE        ;LOOP HERE FOR COL'S
0047  0530  8D 0C AC              STA PCR3       ;DISCHARGE CAP
0048  0533  CE FF 03              DEC COLCTR
0049  0536  30 DC                 BMI CHAR       ;BRANCH IF DONE W/6 COL'S
0050  0538  D0 02                 BNE COLUP
0051  053A  A2 00                 LDX #0         ;INTER CHAR SPACE
0052  053C  A9 EC          COLUP  LDA #$EC        ;START RAMP UP ...
0053  053E  8D 0C AC              STA PCR3       ;... BUT HOLD DOT DOWN
0054  0541  E8                    INX            ;NEXT COL
0055  0542  8A                    TXA            ;SAVE X
0056  0543  48                    PHA
```

Table 7-1. OSCILLOSCOPE OUTPUT DRIVER SOFTWARE LISTING (Continued)

```
LINE # LOC      CODE        LINE

0057  0544  BD FF 03            LDA SYMBLS-1,X  ;GET COL
0058  0547  A0 08               LDY #8          ;COUNT DOTS
0059  0549  88          DOT     DEY
0060  054A  30 0F               BMI CLEAN
0061  054C  4A                  LSR A           ;NEXT DOT IN CARRY
0062  054D  B0 04               BCS LIGHT       ;C SET = LIGHT, C CLEAR = DARK
0063  054F  A2 EC        DARK    LDX #$EC        ;PULL OUTPUT LOW
0064  0551  D0 02               BNE *+4
0065  0553  A2 CC        LIGHT   LDX #$CC        ;OUTPUT FOLLWS RAMP UP
0066  0555  8E 0C AC            STX PCR3
0067  0558  4C 49 05            JMP DOT
0068  055B  68          CLEAN   PLA             ;RESTORE X
0069  055C  AA                  TAX
0070  055D  4C 2E 05            JMP COLUMN
0071  0560               ;
0072  0560               ;
0073  0560  20 AF 88     HKEY    JSR GETKEY      ;GET KEY + ECHO TO SCOPE
0074  0563  20 88 81     SCPDSP  JSR SAVER       ;FILL SCPBUF FROM ASCII IN A
0075  0566  29 7F               AND #$7F
0076  0568  C9 07               CMP #$07        ;BELL?
0077  056A  D0 03               BNE NBELL
0078  056C  4C 75 89            JMP BEEPP3
0079  056F               ; SEARCH ASCII TABLE IN MONITOR ROM
0080  056F  A2 36        NBELL   LDX #$36
0081  0571  DD EE 8B     OUD2    CMP ASCIM1,X
0082  0574  F0 06               BEQ GOTX
0083  0576  CA                  DEX
0084  0577  D0 F8               BNE OUD2
0085  0579  4C C4 81            JMP RESALL      ;NOT IN TABLE
0086  057C  CA          GOTX    DEX
0087  057D  8A                  TXA
0088  057E  C9 0B               CMP #$0B        ;TABLE NOT CONTINUOUS
0089  0580  90 03               BCC GOOD
0090  0582  38                  SEC
0091  0583  E9 05               SBC #5          ;ADJUST DISCONTINUITY
0092  0585  CA          GOOD    DEX
0093  0586  20 06 8A            JSR TXTSHV      ;SHOVE SCPBUF DOWN
0094  0589  4C C4 81            JMP RESALL
0095  058C  20 63 05     HDOUT   JSR SCPDSP      ;CHAR TO SCPBUF AND SINGLE SCAN
0096  058F  4C 6F A6            JMP SCNVEC
0097  0592                      .END
```

Table 7-1. OSCILLOSCOPE OUTPUT DRIVER SOFTWARE LISTING (Continued)

```
;
; 8X5 MATRIX CHAR SET FOR SCOPE LINE DRIVER
; CONTAINS ALL HEX KB CHARS
; FIRST BYTE OF TABLE MUST BE 00
; EACH CHAR : FIRST BYTE = LEFTMOST COLUMN,
;             MSB = TOP DOT, LSB = 0, BIT 1 = BOTTOM DOT
;
*=$400 ;PAGE 4 ALLOCATED TO CHARACTER SET
.BYT $00,$7C,$92,$A2,$7C ;ZERO
.BYT $00,$42,$FE,$02,$00 ;ONE
.BYT $4E,$92,$92,$92,$62 ;TWO
.BYT $44,$82,$92,$92,$6C ;THREE
.BYT $18,$28,$48,$FE,$08 ;FOUR
.BYT $E4,$A2,$A2,$A2,$9C ;FIVE
.BYT $3C,$52,$92,$92,$0C ;SIX
.BYT $86,$88,$90,$A0,$C0 ;SEVEN
.BYT $6C,$92,$92,$92,$6C ;EIGHT
.BYT $60,$92,$92,$94,$78 ;NINE
.BYT $3E,$50,$90,$50,$3E ;A
.BYT $00,$1E,$86,$4A,$32 ;C/R
.BYT $10,$10,$10,$10,$10 ;DASH
.BYT $82,$44,$28,$10,$00 ;RIGHT ARROW
.BYT $FE,$FE,$FE,$FE,$FE ;SH
.BYT $7C,$82,$82,$8A,$4E ;G
.BYT $FE,$90,$98,$94,$62 ;R
.BYT $FE,$40,$30,$40,$FE ;M
.BYT $FE,$02,$02,$02,$02 ;L2
.BYT $44,$A2,$92,$8A,$44 ;S2
.BYT $80,$80,$80,$80,$80 ;U0
.BYT $02,$02,$02,$02,$02 ;U1
.BYT $82,$82,$82,$82,$82 ;U2
.BYT $FE,$00,$00,$00,$00 ;U3
.BYT $FE,$00,$00,$00,$FE ;U4
.BYT $1E,$12,$12,$12,$1E ;U5
.BYT $F0,$90,$90,$90,$F0 ;U6
.BYT $80,$80,$80,$80,$F0 ;U7
.BYT $04,$02,$02,$02,$FC ;J
.BYT $E0,$18,$06,$18,$E0 ;V
.BYT $FF,$FF,$FF,$FF,$FF ;ASCII
.BYT $FE,$92,$92,$92,$6C ;B
.BYT $7C,$82,$82,$82,$44 ;C
.BYT $FE,$82,$82,$82,$7C ;D
.BYT $FE,$92,$92,$82,$82 ;E
.BYT $FE,$90,$90,$80,$80 ;F
.BYT $44,$A2,$92,$8A,$44 ;SD
.BYT $10,$10,$7C,$10,$10 ;+
.BYT $00,$10,$28,$44,$82 ;<
.BYT $00,$00,$00,$00,$00 ;SH
.BYT $FE,$02,$02,$02,$02 ;LP
.BYT $44,$A2,$92,$8A,$44 ;SP
.BYT $FE,$04,$08,$04,$FE ;W
.BYT $FE,$02,$02,$02,$02 ;L1
.BYT $44,$A2,$92,$8A,$44 ;S1
.BYT $00,$06,$06,$00,$00 ;DECIMAL
.BYT $00,$00,$00,$00,$00 ;BLANK
.BYT $40,$80,$8A,$90,$60 ;QUESTION
.BYT $FE,$90,$90,$90,$60 ;P
.END
```

CHAPTER 8

SYSTEM EXPANSION

This chapter discusses the means by which you can expand your SYM-1 microcomputer system by adding memory and peripheral devices to its basic configuration. By now, you realize that data access, whether from RAM, PROM or ROM is a function of addressing interface devices (i.e., 6522's and 6532). Hardware has been built into your SYM-1 module to allow large-scale expansion of the system. A thorough understanding of the SYM-1 System Memory Map (Figure 4-10) will aid considerably in understanding how to expand your system.

8.1 MEMORY EXPANSION

Your SYM-1 module comes equipped with 1K of on-board RAM. It also contains all address decoding logic required to support an additional 3K on-board with no changes by you. In other words, to add 3K of on-board RAM, all you need to do is purchase additional SY2114 devices and plug them into the sockets provided on your board. Your PC board is marked for easy identification of 1K memory blocks. RO equals the lower 1K block and R3 equals the upper 1K block. LO means low order data lines (D0-D3) and HI means high order data lines (D4-D7).

You will recall that the lowest 8K memory locations are defined by an address decoder included on your SYM-1 module (a 74LS138). The eight outputs of this decoder ($\overline{00}$-$\overline{1C}$) each define a 1K block of addresses in the lowest 8K of the Memory Map. Four of the outputs ($\overline{00}$, $\overline{04}$, $\overline{08}$, $\overline{0C}$) are used to select the on-board static RAM. The remaining four outputs ($\overline{10}$, $\overline{14}$, $\overline{18}$, $\overline{1C}$) are used to interface to the Application Connector (Connector "A"), where you can use them to add another 4K of off-board memory. Again, no external decoding logic is required. By this simple means, you can convert your SYM-1 module into an 8K device quickly. Figure 8-1 shows you how to interface these decode lines at the connector for your SYM-1 system.

To go beyond this 8K size, conceivably up to the maximum 65K addressability limit of the SYM CPU, you could build or buy an additional memory board with on-board decoding logic. In this case, you will use the Expansion Connector (Connector "E") in a manner shown schematically in Figure 8-2. Note that the three high-order address bits (AB13-AB15) not used in the earlier expansion are brought to this connector as shown. These are then used with a decoder to create outputs $\overline{M0}$ through $\overline{M7}$, which in turn are used to select and de-select additional decoders (line receivers). You need add only as many decoders (one for each 8K block of memory) as you need for the expansion you require.

Incidentally, the line receivers shown in Figure 8-2 are provided for electrical reasons. There are loading limitations on the address bus lines of the 6502 CPU, which require the insertion of these receivers. (For your information, each 6502 address line is capable of driving one standard TTL load and 130pf of capacity.)

You should make a careful study of the loading limitations of the required SYM-1 lines before deciding on memory expansion size and devices. It is likely you will want to use additional buffer circuits to attain "cleaner" operation of your expanded memory in conjunction with your SYM-1 system.

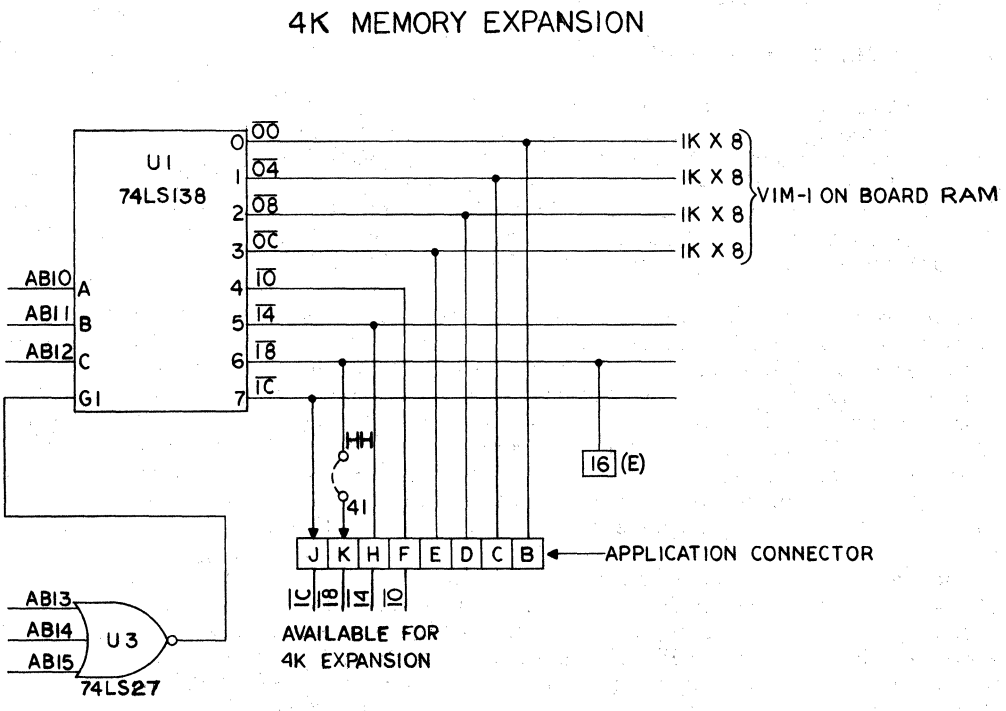


Figure 8-1. 4K MEMORY EXPANSION

MEMORY I/O EXPANSION TO 65K

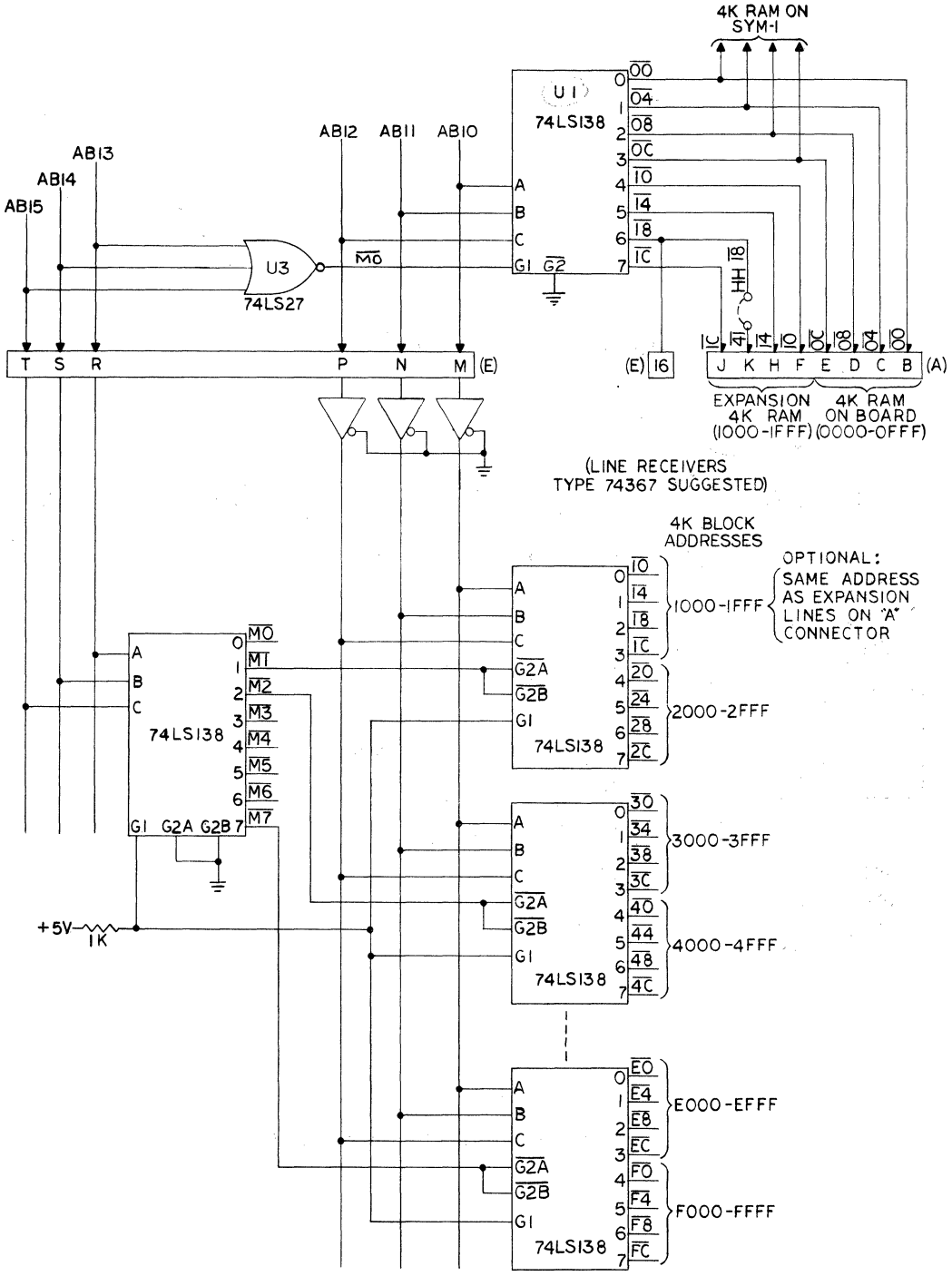


Figure 8-2. MEMORY - I/O EXPANSION TO 65K

8.2 PERIPHERAL EXPANSION

As you already know, the SYM-1 microcomputer system includes 51 I/O lines. This means, theoretically, that you could drive as many as 51 peripheral lines (plus 4 control lines) with your SYM-1.

Using either Application Connector ("A" or "AA"), you can add most commercially available printers or other devices requiring parallel interfaces, although you will have to create your own software driver for the printer. Since the provision of that driver is, to some extent, dependent upon the printer you purchase, we do not attempt to discuss the implementation of the software in this manual.

You can expand your SYM-1 system's peripheral I/O capability easily and quickly merely by installing an additional SY6522 in the socket provided for that device. This will give you 16 additional on-board data lines with no requirement for additional work (beyond the software driver) on your part. To go beyond that level, you must use the Expansion Port (Connector "E") described earlier.

Again, we emphasize that the proper understanding and use of the Memory Map in Figure 4-10 will allow you to use your imagination in expanding the I/O capability of your SYM-1 system. Its flexibility is extremely broad and the fact that all I/O and memory are handled as an addressing function allows you expandability to the full capability of the 6502 CPU itself.

CHAPTER 9

ADVANCED MONITOR AND PROGRAMMING TECHNIQUES

This chapter contains information which you will find useful as you explore the more sophisticated capabilities of your versatile SYM-1 microcomputer system. As we have pointed out many times, the SYM-1 is the most flexible and expandable monitor of its kind. The SUPERMON monitor uses transfer vectors and other techniques to allow you to modify its operation, and these are provided in detail in this chapter. In addition, the extended use of debug and trace facilities, which are invaluable tools as your programming skill advances, are explained. The use of the Hex keyboard provided on your SYM-1 for configurations using a printer (or other serial device) without a keyboard is also described. And last, an example and discussion of extending SUPERMON's command repertoire.

9.1 MONITOR FLOW

SUPERMON is the 4K byte monitor program supplied with your SYM-1. It resides in locations 8000-8FFF on a single ROM chip. It shares the stack with user programs and uses locations 00F8-00FF in Page Zero. In addition, it uses locations A600-A67F (RAM on the 6532), which are referred to as 'System RAM'. Since these locations are dedicated to monitor functions SUPERMON write protects them before transferring control to user programs.

The flowcharts in Figures 9-1 through 9-5 will demonstrate the major structure of SUPERMON. You will notice that GETCOM (and its entry, PARM), DISPAT, and ERMSG are subroutines, and therefore available for your programs' use. Note that a JSR to ACCESS to remove write protection from System RAM is necessary before using most monitor routines. Also, notice that the unrecognized command flow (error) is vectored. Thus, you can extend the monitor with your own software.

9.2 MONITOR CALLS

SUPERMON contains many subroutines and entry points which you will want to use in order to save memory and code and avoid duplication of effort. Table 9-1 is a summary of calls and their addresses.

The three calls which you will most commonly use are:

```
JSR   ACCESS    (address 8B86) (must be called before using LED display)
JSR   INCHR     (address 8A1B)
JSR   OUTCHR    (address 8A47)
```

ACCESS is used to unwrite-protect system RAM. In performing the input/output, these routines save all registers and use INVEC and OUTVEC, so all you need be concerned with when using them are the ASCII characters passed as arguments in the accumulator.

9.3 MONITOR CALLS, ENTRIES AND TABLES

Table 9-1, which occupies the next several pages of this Chapter, provides you with a comprehensive list of important subroutine symbolic names, addresses, registers and functions of SUPERMON monitor calls, entry points and tables. With this data, you can more easily utilize SUPERMON to perform a wide variety of tasks. All (except those marked with an asterisk) are callable by JSR.

Table 9-1. MONITOR CALLS, ENTRIES AND TABLES

NAME	ADDRESS	REGISTERS ALTERED	FUNCTION (S)
*MONITR	8000		Cold entry to monitor. Stack, D flag initialized, System RAM unprotected.
*WARM	8003		Warm entry to monitor
USRENT	8035		User pseudo-interrupt entry - saves all registers when entered with JSR. Displays PC and code 3. Passes control to monitor.
SAVINT	8064	ALL	Saves registers when called after interrupt. Returns by RTS.
DBOFF	80D3	A,F	Simulates depressing debug off key.
DBON	80E4	A,F	Simulates depressing debug on key.
DBNEW	80F6	A,F	Release debug mode to key control.
GETCOM	80FF	A,F	Get command and 0-3 parameters. No error: A=0D (carriage return) Error: A contains erroneous entry.
DISPAT	814A	A,F	Dispatch to execute blocks. Dispatch to URCVEC if error. At return, if error: Carry set, A contains byte in error.
ERMSG	8171	F	If Carry set, print (CR)ER NN, where NN is contents of A.
SAVER	8188	None	Save all registers on stack. At return, stack looks like: F (See paragraph 9.9) A X Y
*RESXAF	81B8	restored	Jumped to after SAVER, restore registers from stack except A,F unchanged, perform RTS.
*RESXF	81BE	restored	Jumped to after SAVER, restore registers from stack except F unchanged, perform RTS.
*RESALL	81C4	restored	Jumped to after SAVER, restore all registers from stack, perform RTS.
INBYTE	81D9	A,F	Get 2 ASCII Hex digits from INCHR and pack to byte in A. If Carry set, V clear, first digit non-Hex. If Carry set, V set, second digit nonHex. N and Z reflect compare with carriage return if Carry set.

*Do not enter by JSR.

Table 9-1. MONITOR CALLS, ENTRIES AND TABLES (Continued)

NAME	ADDRESS	REGISTERS	FUNCTION (S)
PSHOVE	8208	X,F	Shove Parms down 16 bits; Move: P2 to P1 P3 to P2 zeros to P3
PARM	8220	A,F	Get 0 to 3 parameters. Return on (CR) or error. A contains last character entered. Flags reflect compare with (CR).
ASCN1B	8275	A,F	Convert ASCII character in A to 4 bits in LO nibble of A. Carry set if non-Hex.
OUTPC	82EE	A,X,F	Print user PC. At return, A=PCL, X=PCH.
OUTXAH	82F4	F	Print X,A (4 Hex digits)
OUTBYT	82FA	F	Print A (2 Hex digits)
NIBASC	8309	A,F	Convert LO nibble of A to ASCII Hex in A.
COMMA	833A	F	Print comma.
CRLF	834D	F	Print **(CR) (LF).**
DELAY	835A	F,X	Delay according to TV. (Relation is approximately logarithmic, base=2). Result of INSTAT returned in Carry.
INSTAT	8386	F	If key down, wait for release. Carry set if key down. (Vectored thru INSVEC)
GETKEY	88AF	A,F	Get key from Hex keyboard (more than one if SHIFT or ASCII key used) return with ASCII or HASH code in A. Scans display while waiting (vectored through SCNVEC).
HDOUT	8900	A,X,Y,F	ASCII character from A to Hex display, scan display once, return with Z=1 if key down.
SCAND	8906	A,X,Y,F	Scan the LED display once from the data in DISBUF. Return Z set if a key on hex keyboard is down.
KEYQ	8923	A,F	Determine if key down on Hex keyboard. If down, then Z=1.
KYSTAT	896A	A,F	Determine if key down. If down, then Carry set.
BEEP	8972	None	BEEP on-board beeper.
HKEY	89BE	A,F	Get key from Hex keyboard and echo in DISBUF. ASCII returned in A. Scans display while waiting (vectored thru SCNVEC)

*Do not enter by JSR

Table 9-1. MONITOR CALLS, ENTRIES AND TABLES (Continued)

NAME	ADDRESS	REGISTERS	FUNCTION (S)
OUTDSP	89C1	None	Convert ASCII in A to segment code, put in DISBUF.
TEXT	8A06	F	Shove scope buffer down, push A onto SCPBUF.
INCHR	8A1B	A,F	Get character (vectored thru INVEC). Drop parity, convert to upper case. If character CTL O (0F), toggle Bit 6 of TECHO and get another.
NBASOC	8A44	A,F	Convert low nibble of A to ASCII, output (vectored thru OUTVEC).
OUTCHR	8A47	None	Output ASCII from A (vectored thru OUTVEC). Output inhibited by Bit 6 of TECHO.
INTCHR	8A58	A,F	Get character from serial ports. Echo inhibited by Bit 7 of TECHO. Baud rate determined by SDBYT. Input, echo masked with TOUTFL.
TSTAT	8B3C	A,F	See if break key down on terminal. If down, then Carry set.
*RESET	8B4A	All	Initialize all registers, disable POR, stop tape, initialize system RAM to default values, determine input on keyboard or terminal, determine baud rate, cold monitor entry.
*NEWDEV	8B64		Determine baud rate, cold monitor entry.
ACCESS	8B86	None	Un-write protect System RAM.
NACCESS	8B9C	None	Write protect System RAM.
*TTY	8BA7	A,X,F	Set vectors, TOUTFL, and SDBYT for TTY.
*DFTBLK	8FA0	Table	Default block - entirely copied into System RAM (A620 - A67F) at reset.
*ASCII	8BEF	Table	Table of ASCII codes and HASH codes.
*SEGS	8C29	Table	Table of segment codes corresponding to ASCII codes (above).

*Do not enter by JSR

MAIN MONITOR FLOW

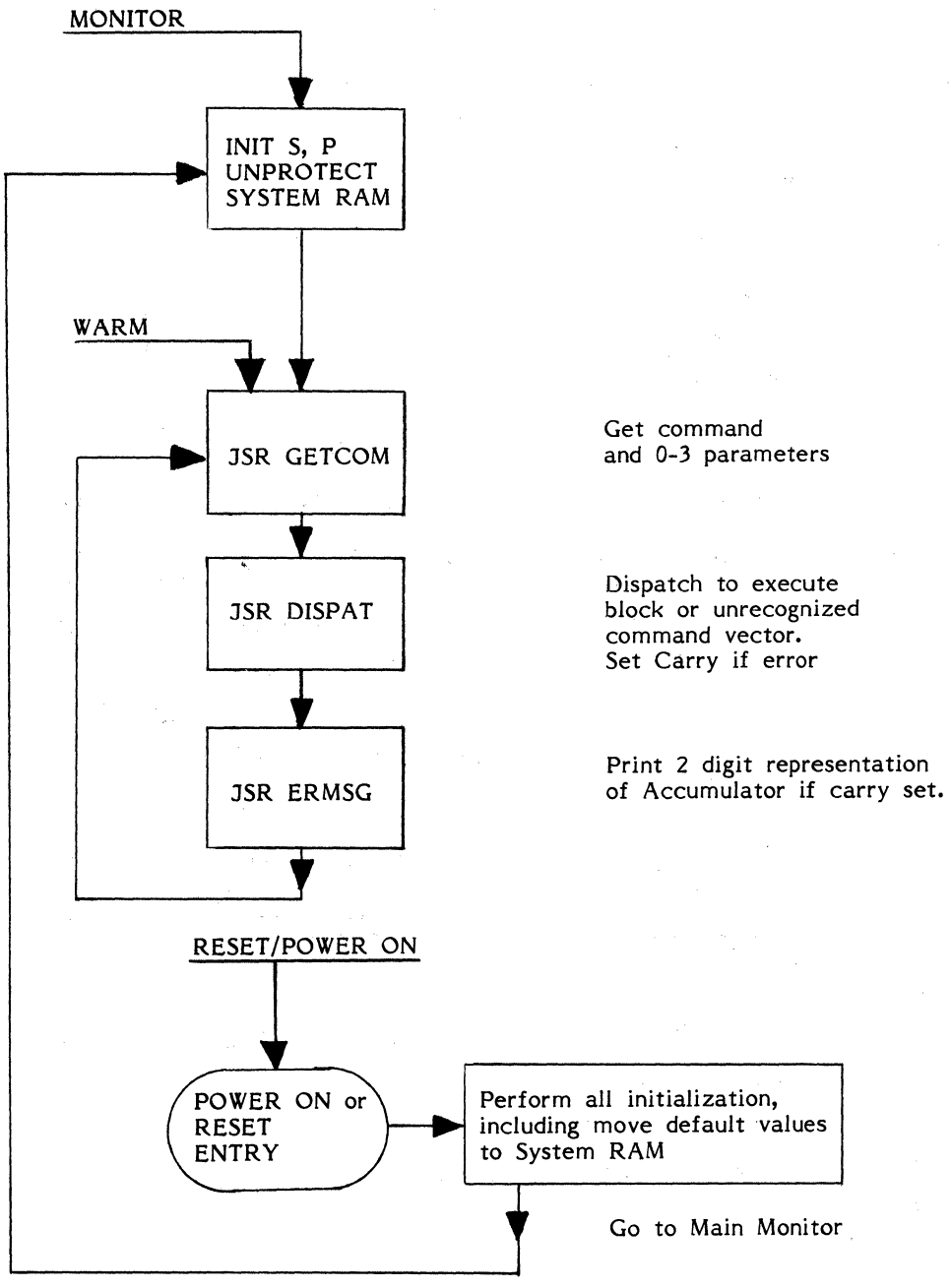


Figure 9-1. MAIN MONITOR FLOW

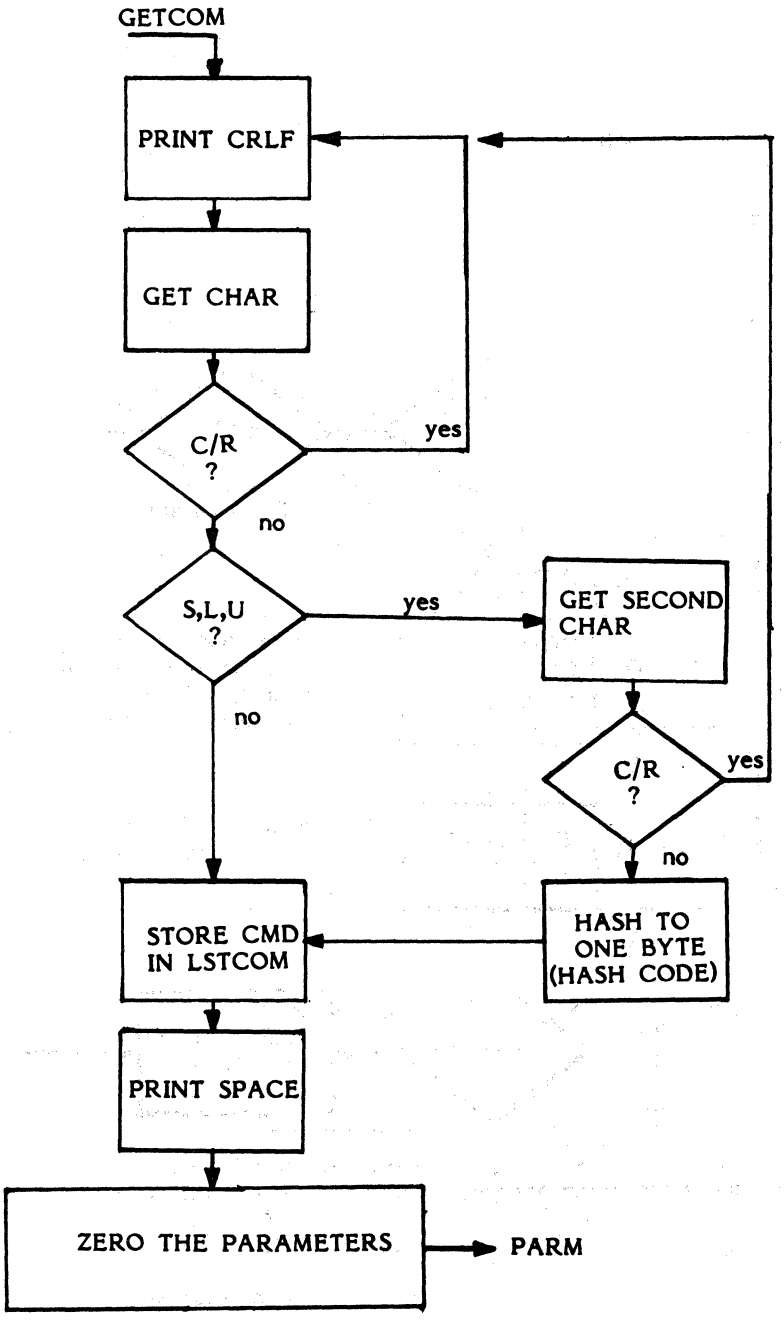


Figure 9-2. GETCOM FLOWCHART

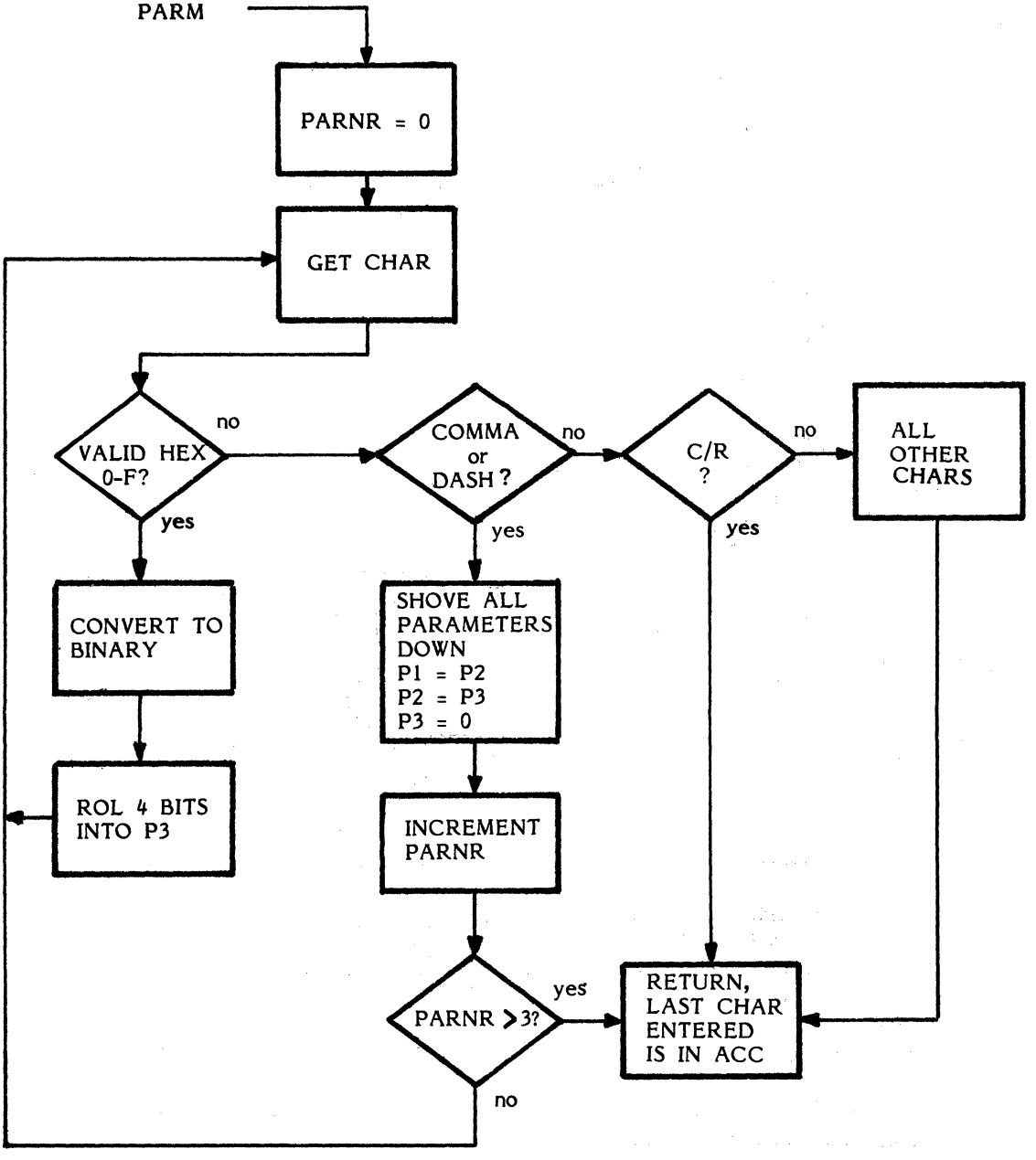


Figure 9-3. PARM FLOWCHART

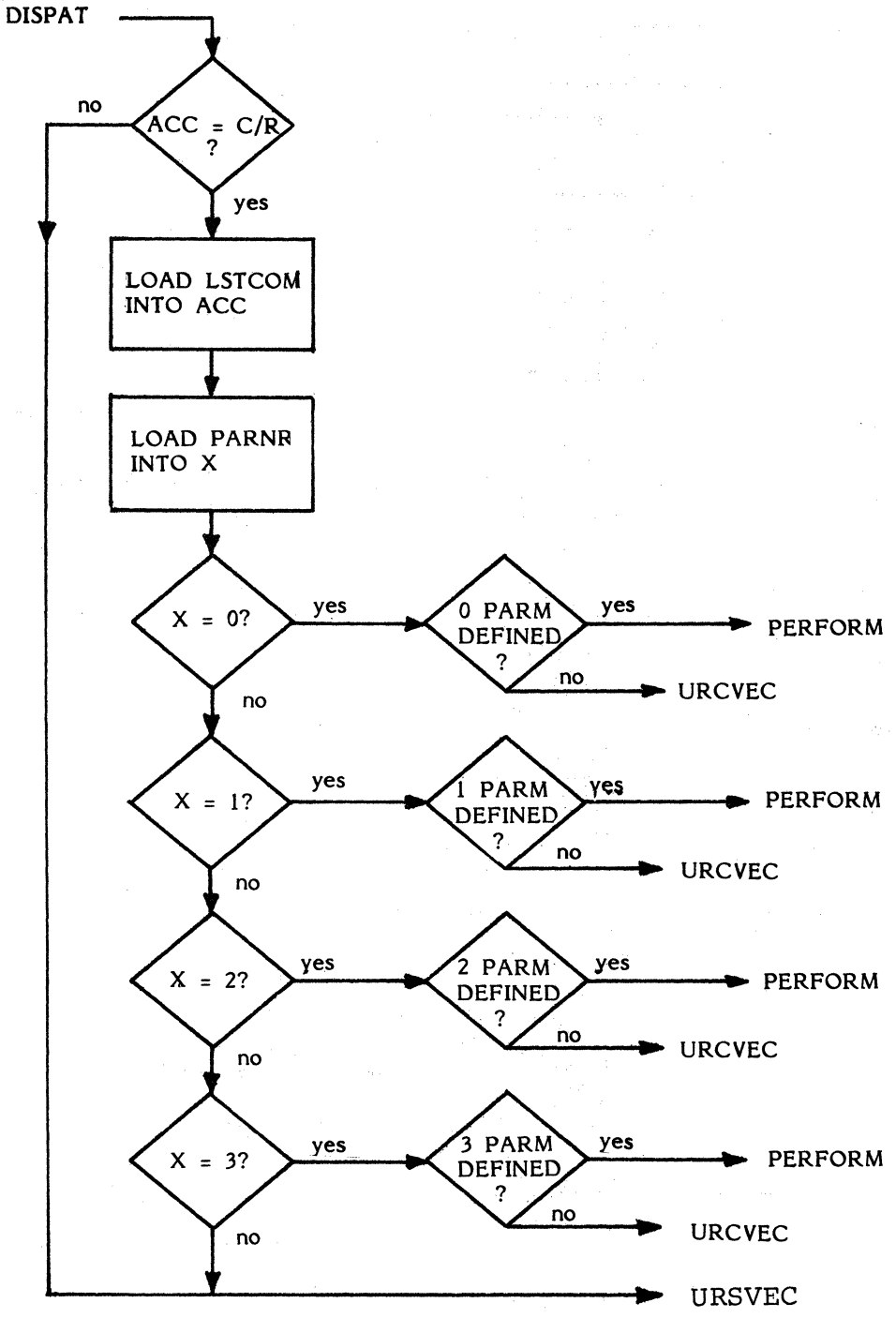


Figure 9-4. DISPAT FLOWCHART

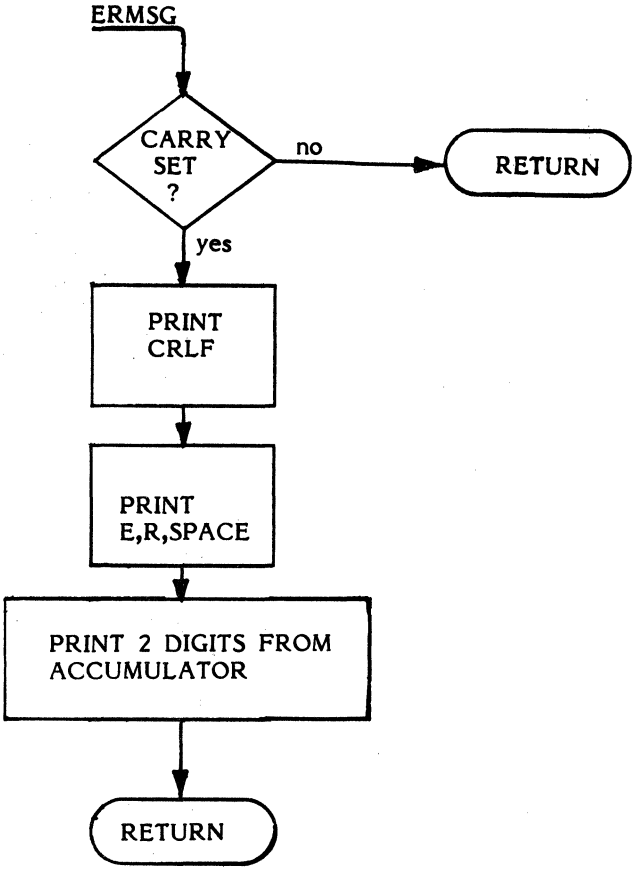


Figure 9-5. ERMSG FLOWCHART

9.4 VECTORS AND INTERRUPTS

A concept which is very important in understanding the SY6502 and SUPERMON is that of a transfer vector. A transfer vector consists of two or three locations at a fixed address in memory. These locations contain an address, or a Hex 4C **(JMP)** and an address. The address is in low-order, high-order byte order.

As an example, consider the function of outputting a character. In some cases, the character is to go to the display, in others to a terminal device. The action required in each case is radically different. It would be inefficient, in code and in time, to make the decision before outputting each character. The solution is a transfer vector. Whenever SUPERMON must output a character, it performs a JSR to OUTCHR. OUTCHR saves all registers, then performs a JSR to OUTVEC (at A663, in System RAM). If you are working at the Hex keyboard OUTVEC will contain a JMP HDOUT. HDOUT is the subroutine which will enter a character, in segment code, into the display buffer. If you are using a TTY or CRT, OUTVEC will contain a JMP TOUT. TOUT is the subroutine which sends a character, one bit at a time, to the serial I/O ports. When HDOUT or TOUT performs an RTS, control passes back to OUTCHR. OUTCHR restores the registers and performs an RTS, returning control to the caller.

Notice that the calling routine need not worry where the output is going. It is all taken care of by OUTCHR and OUTVEC.

When a vector is to be referenced by a JMP Indirect, only two bytes are required. Two-byte vectors are normally used only for interrupts.

An **INTERRUPT** is a method of transferring program control, or interrupting, the processor during execution. There are three interrupts defined on the SY6502:

NMI -- non-maskable interrupt
RST -- reset/power-on
IRQ -- interrupt request

When one of these interrupts occurs, the processor pushes the PC register and the Flags register onto the stack, and gets a new PC from the **INTERRUPT VECTOR.** The interrupt vectors are located at the following addresses:

FFFA,FFFB -- NMI
FFFC,FFFD -- RESET
FFFE,FFFF -- IRQ

These locations must contain the addresses of programs which will determine the cause of the interrupt, and respond appropriately.

In the SYM-1, System RAM (A600-A67F) is duplicated at FF80-FFFF (it is "echoed" there). On Reset, SUPERMON points these vectors to its own interrupt-handling routines. When an interrupt occurs, SUPERMON displays the address where the interrupt occurred with one of the following codes indicating the cause of the interrupt:

0 * = BRK instruction
1 = IRQ
2 = NMI
3 = USER ENTRY (caused by JSR to USRENT at 8035)

Because all registers are saved, a **(G) (CR)** will cause execution to resume at the point of interruption. The user can intercept interrupt handling by inserting pointers to user interrupt routines in TRCVEC, UBRKVC, NMIVEC, or IRQVEC. See Section 9.8.2 for a discussion of the User Entry pseudo-interrupt. Table 9-2 describes all vectors used by the Monitor.

Table 9-2. SUPERMON VECTORS

NAME	LOCATION	FUNCTION
INVEC	A660-A662	Points to input driver.
OUTVEC	A663-A665	Points to output driver.
INSVEC	A666-A668	Points to routine which determines whether or not a key is down.
URCVEC	A66C-A66E	Unrecognized command. All unrecognized commands and parameter entry errors vectored here. Points to a sequence of: SEC - Set Carry RTS - Return
SCNVEC	A66F-A671	Points to routine which performs one scan of display from DISBUF.
EXEVEC	A672-A673	Points to RIN - get ASCII from RAM subroutine. The Execute (E) command temporarily replaces INVEC with EXEVEC, saving INVEC in SCRA, SCRB. The Hi byte of EXEVEC must be different from the Hi byte of INVEC.
TRCVEC	A674-A675	May be used to point to user trace routine after TRCOFF (See Section 9.6).
UBRKVC	A676-A677	May be used to point to user BRK routine after IRQVEC.
UIRQVC	A678-A679	May be used to point to user NON-BRK IRQ routine after IRQVEC.
NMIVEC	A67A-A67B	Points to routine which saves registers, determines whether or not to trace, based on TV.
IRQVEC	A67E-A67F	Points to routine which saves registers, determines whether or not BRK has occurred, and continues thru UBRKVC or UIRQVC.

9.5 DEBUG ON and TRACE

When the DEBUG ON key on your SYM-1 is depressed, DEBUG mode is established. In DEBUG mode, an NMI interrupt occurs every time an instruction is fetched from an address that is not within the monitor. SUPERMON's response is to save the registers and display the PC, with code 2 (for NMI). With each **(G) (CR)**, one instruction of the user program will be executed. This is called Single-Stepping.

In order to TRACE, alter the Trace Velocity (TV, at A656) to a non-zero value. (09 is a good value.) If you now enter **(G) (CR)**, SUPERMON will display the PC and the contents of the accumulator, pause, and resume execution. Addresses and accumulator contents will flash by one at a time. To stop the flow, depress any key (Hex keyboard) or the BREAK key (terminal). Execution will halt. A **(G) (CR)** will resume execution. The length of the delay is related to TV (not linearly; try different values) and, of course, the baud rate, if you are working from a terminal.

9.6 USER TRACE ROUTINES

As the complexity of your programs increases, you may wish to implement other types of trace routines. To demonstrate how this is done, an example of a user trace routine is provided in Figure 9-6. It prints the op code of the instruction about to be executed, instead of the accumulator contents.

But first of all, we don't want to be interrupted during trace mode by responding to an interrupt (a problem called recursion). SUPERMON will handle this by turning DEBUG OFF, then back ON. However, to implement this program control of DEBUG, you must add jumpers W24 and X25 to your SYM-1 board (see Chapter 4).

Now that you have added the jumpers, we are ready to enter the program UTRC and change vectors.

First, enter the program UTRC as given in Figure 9-6. Then change NMIVEC to point to TRCOFF, which will save registers, turn DEBUG OFF, and vector thru TRCVEC:

SD 80C0,A67A **(CR)**

Now, point TRCVEC to UTRC.

SD 0380,A674 **(CR)**

Enter a non-zero value in TV, depress DEBUG ON, and you're ready to trace.

NOTE: BRK instructions with DEBUG ON will operate as two-byte instructions and should be programmed as 00,EA (BRK,NOP).

Also, the first instruction after leaving SUPERMON will not be traced.

```
LINE #  LOC     CODE        LINE

0002  0000              ; UTRC - USER TRACE ROUTINE -
0003  0000              ; PRINT NEXT OP CODE INSTEAD OF ACCUMULATOR
0004  0000              ;
0005  0000              OPPCOM =$8337          ;PRINT PC, PRINT COMMA
0006  0000              PCLR   =$A659
0007  0000              PCHR   =$A65A
0008  0000              OBCRLF =$834A          ;PRINT BYTE FROM ACC, PRINT CRLF
0009  0000              DELAY  =$835A          ;DELAY BASED ON TV
0010  0000              WARM   =$8003          ;WARM MONITOR ENTRY
0011  0000              TRACON =$80CD          ;TURN TRACE ON, RESUME EXECUTION
0012  0000              TV     =$A656          ;TRACE VELOCITY
0013  0000              ;
0014  0000                     *=$380          ;PUT IN HI RAM (ENTIRELY RELOCATE/
0015  0380  20 37 83    UTRC   JSR OPPCOM      ;PRINT PC, COMMA
0016  0383  AD 59 A6           LDA PCLR        ;USE PC AS PTR TO OP CODE
0017  0386  85 F0              STA $F0
0018  0388  AD 5A A6           LDA PCHR
0019  038B  85 F1              STA $F1
0020  038D  A0 00              LDY #0
0021  038F  B1 F0              LDA ($F0),Y     ;PICK UP OP CODE
0022  0391  20 4A 83           JSR OBCRLF      ;OUTPUT OP CODE, CRLF
0023  0394  AE 56 A6           LDX TV          ;GET TRACE VELOCITY
0024  0397  F0 05              BEQ NOGO        ;NOGO IF ZERO
0025  0399  20 5A 83           JSR DELAY       ;DELAY ACCORDING TO TV
0026  039C  90 03              BCC GO          ;CARRY SET IF KEY DOWN
0027  039E  4C 03 80  NOGO     JMP WARM        ;HALT
0028  03A1  4C CD 80  GO       JMP TRACON      ;CONTINUE
0029  03A4                     .END
```

Figure 9-6. LISTING OF SAMPLE USER TRACE ROUTINE

USER TRACE EXAMPLE

```
.V 200,20A (CR)
  0200 A9 00 A9 11 A9 22 A9 33,0A
  0208  4C 00 02,58
0358
.SD 80C0,A67A (CR)          Vector modification
.SD 380,A674 (CR)           Vector modification
.G 200 (CR)                 Single-Step (Remember
0202,A9                     to set DEBUG ON before
                            each (G) (CR)
 G (CR)
0204,A9

.M A656(CR)
A656,00,09(CR)              Trace Velocity = 9
A657,4D (CR)
.G 200 (CR)
0202,A9
0204,A9
0206,A9
0208,4C                     Continuous trace of op codes
0200,A9
0202,A9
0204,A9
0206,A9
0208,4C
0200,A9
0202,A9
```

9.7 MIXED I/O CONFIGURATIONS

The Reset routine that is activated when power is turned on or RST is pressed establishes the terminal I/O configuration by loading a specified value into a location in System RAM, TOUTFL (A654). The high-order four bits of TOUTFL define which terminal devices may be used for input and output. A "1" signifies that a device is enabled, a "0" that it is disabled. The meaning of each bit and the values assigned at system reset are shown below. The routine referenced by entry (1) in the JUMP table will enable the TTY for input. For other configurations, load the appropriate value into TOUTFL.

TOUTFL bit:	7	6	5	4
default value:	1	0	1	1
meaning:	CRT INPUT	TTY INPUT	TTY OUTPUT	CRT OUTPUT

Bits 6 and 7 of another location in System RAM, TECHO (A653), are used t serial output (bit 6) and to control echo to a terminal (bit 7). Bit 6 may be by entering "(CONTROL) O" (0F Hex) on the terminal keyboard or in software. possible values for TECHO are shown below.

TECHO	80	echo output	(default value)
	C0	echo no output	
	40	no echo no output	
	00	no echo output	

With this information, you can alter the SUPERMON standard I/O configurations to suit your special needs. A common use would be routing your output to a terminal while using the Hex keyboard as an input device. Two possible ways of doing this will be discussed.

First, by merely altering SDBYT and OUTVEC, your input and echo will use the on-board keyboard and display, while Monitor and program output will go to the serial device. Choose the proper baud rate value for your device from the following table and put it in SDBYT (at A651) with the "M" command. Then enter the address of TOUT into OUTVEC from the hex keyboard as follows:

.SD 8AA0,A664 **(CR)**

Terminal Baud Rate	Value Placed in SDBYT
110	D5
300	4C
600	24
1200	10
2400	06
4800	01

Second, if you wish your input to be echoed on the terminal device, a small program must be entered. First, complete the sequence discussed above. Then, enter the following program:

```
UIN   JSR  GETKEY    20   AF   88
      BIT  TECHO     2C   53   A6
      BPL  UOUT      10   03
      JMP  OUTCHR    4C   47   8A
UOUT  RTS            60
```

Enter the program called "UIN" above at any user RAM location. Then use the "SD" command to put the address of UIN into INVEC (at A661) as follows:

.SD (UIN),A661 **(CR)** **(ENTER AT HKB)**

where (UIN) is the address of the program UIN.

9.8 USER MONITOR EXTENSIONS

Having read the section on Monitor flow, you will have noticed that unrecognized commands and parameter entries are vectored through URCVEC (A66C-A66E), which normally points to a SEC, RTS sequence at 81D1. By pointing URCVEC to a user-supplied routine in RAM or PROM, SUPERMON can easily be extended. The following example will illustrate the basic principle; many more sophisticated extensions are left to your imagination.

9.8.1 Monitor Extension Example

This example will define U0 with two parameters as a logical AND. The parameters and the result are in Hexadecimal.

```
LOGAND   CMP  #$14        ;USR0
         BNE  NEXT
         CPX  #2          ;two parms
         BNE  NEXT
DOAND    LDA  P2H
         AND  P3H         ;here's the 'and' hi
         TAX
         LDA  P2L
         AND  P3L         ;'and' lo
         JSR  CRLF        ;get new line
         JSR  SPACE
         JMP  OUTXAH      ;PRINT X and A
NEXT     SEC              ;last
         RTS
        .END
```

To attach LOGAND to the monitor, it must be assembled (probably by hand), entered into memory, and URCVEC altered to contain a JMP to LOGAND. Notice that more than one command could have been added, by pointing NEXT to the next possible command, instead of a RTS.

9.8.2 SUPERMON As Extension to User Routines

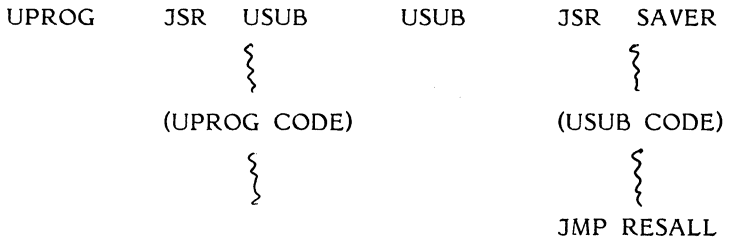
Because SUPERMON contains a user entry, it can easily be appended to your software. An example of the utility of this feature is a user trace routine, which could have an 'M' command, which would direct it to make SUPERMON available to the user. Here's what the code would look like.

```
UTRACE
. . .                                   Trace code

          JSR INCHR
          CMP #'M
          BNE ELSE
          JSR USRENT
          JMP UTRACE
          . . .

ELSE                                    Code executed if character
                                        input is not 'M'.
```

In this example, the user will type an 'M' to get into monitor, and a **(G) (CR)** to return to the calling portion of UTRACE. Note that the user PC and S registers should not be modified while in monitor if a return to UTRACE is intended.

9.9 USE OF SAVER AND RES ROUTINES

SAVER and the RES routines are designed to be used with subroutines. Their usage is as follows:



In this example, UPROG calls USUB. USUB calls SAVER, performs its function, and then jumps to RESALL. RESALL restores all registers and returns to UPROG. If RESXF or RESXAF were used instead of RESALL, UPROG would receive the F, or F and A registers as left by USUB.

APPENDIX A

IMMEDIATE ACTION

Your SYM-1 microcomputer has been thoroughly tested at the factory and carefully packed to prevent damage in shipping. It should provide you with years of trouble-free operation. If your unit does not respond properly when you attempt to apply power, enter commands from the keyboard, or attach peripheral devices to the system, do not immediately assume that it is defective. Re-read the appropriate sections of this manual and verify that all connections have been properly wired and all procedures properly executed.

If you finally conclude that your SYM-1 is defective, you should return it for repair to an authorized service representative. Specific instructions for obtaining a service authorization number and shipping your unit are contained with warranty information on the card entitled "LIMITED WARRANTY AND SERVICE PLAN" that is included with system reference material.

APPENDIX B

PARTS LIST

MATERIALS AND ACCESSORIES

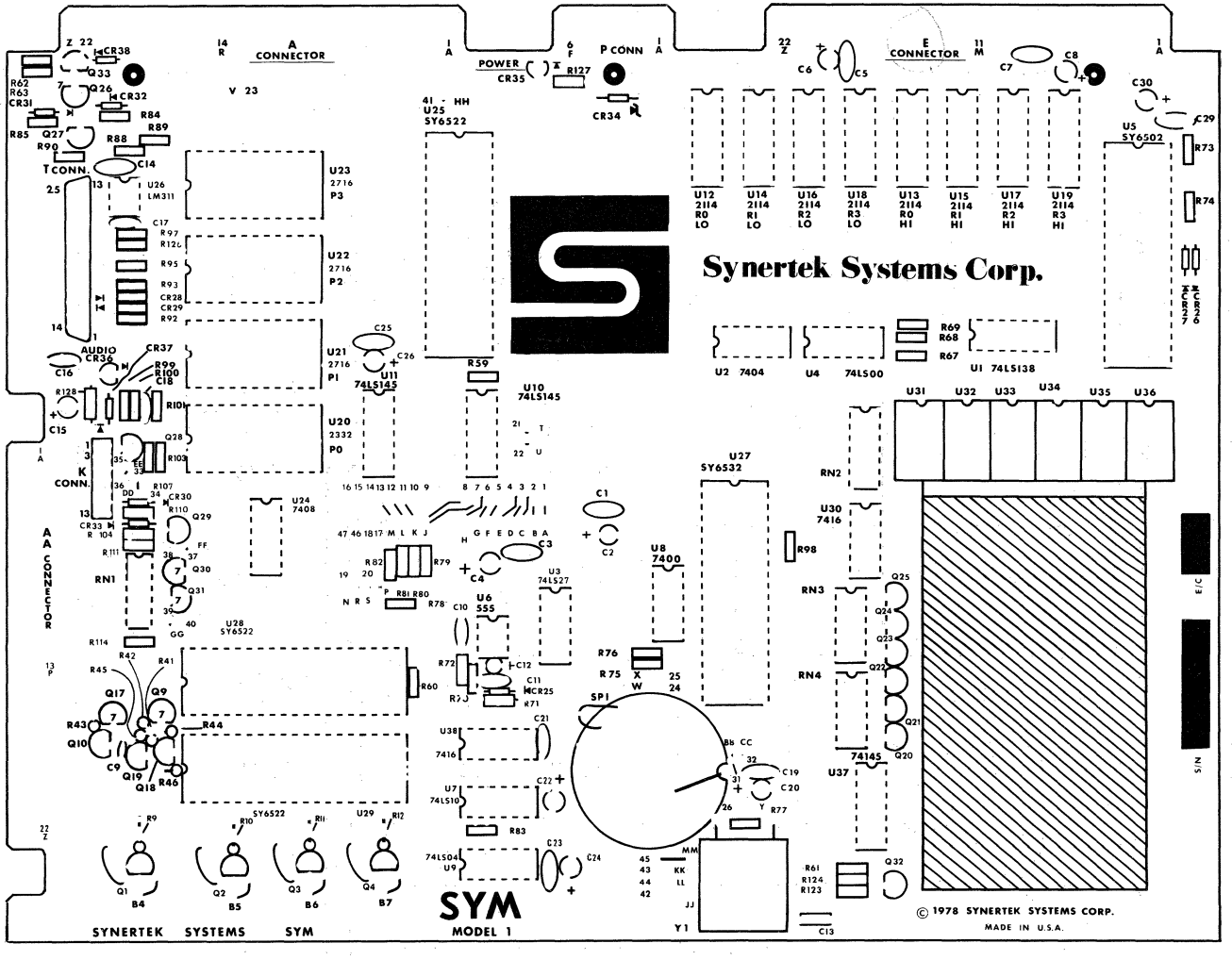
QTY.	DESCRIPTION	MANUFACTURER/PART NUMBER
1	CONNECTOR, DUAL 22/44	Microplastic 15622DPIS
1	CONNECTOR, DUAL 6/12	Teka TP3-061-E04
6	RUBBER FEET	3M SJ5018
1	SYNERTEK SOFTWARE MANUAL	MNA-2
1	SYM-1 REFERENCE CARD	SRC-1
1	SYM REFERENCE MANUAL	MNA-1
1	SYM-1 PC BOARD ASSEMBLY	
1	RED FILTER	

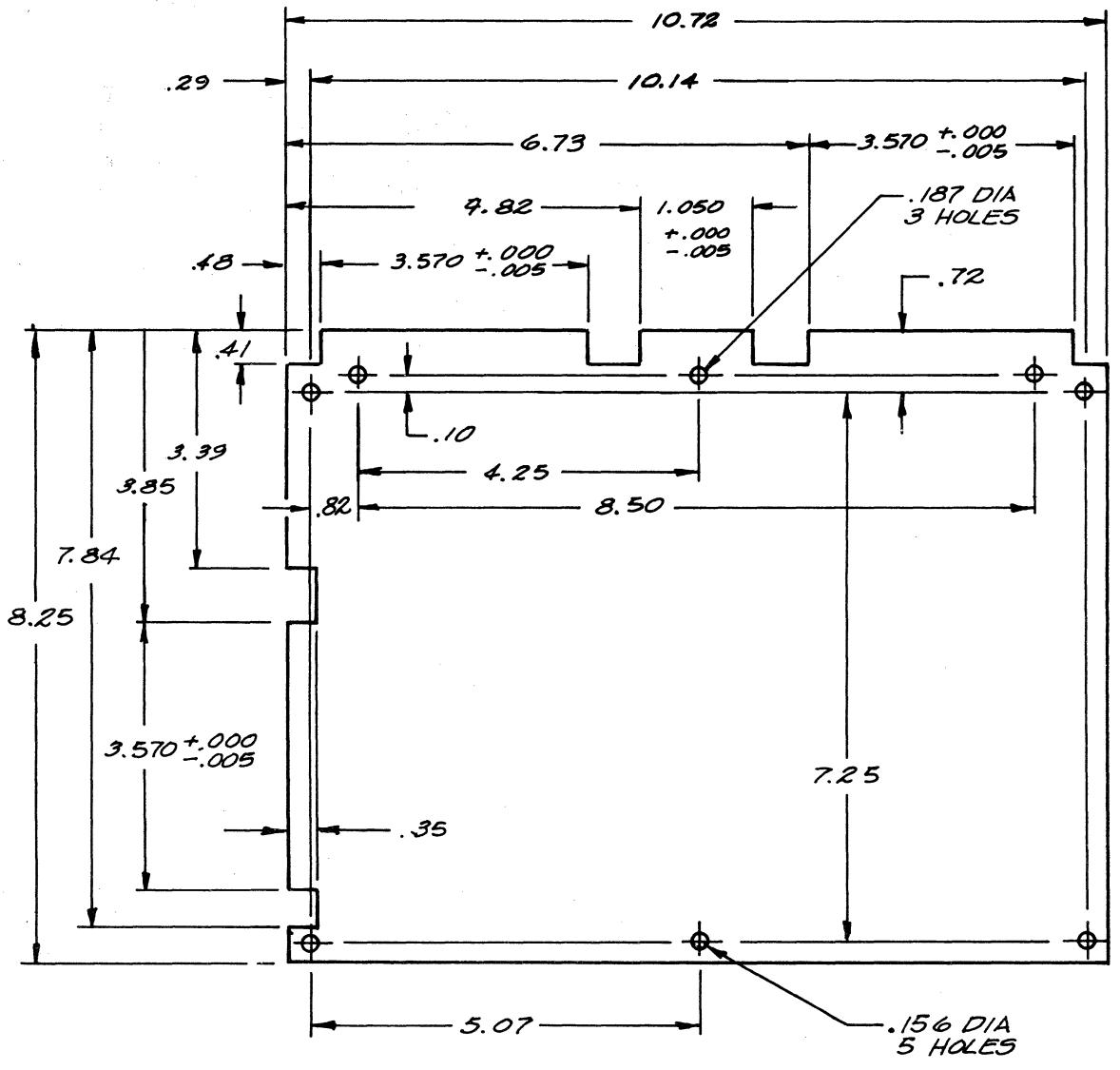
SYM-1 PC BOARD COMPONENTS

QTY.	DESCRIPTION	MFR. NO.	REFERENCE DESIGNATION
1	CPU	SYP6502	U5
2	VIA	SYP6522	U25,U29
1	RAM-I/O	SYP6532	U27
2	4K BIT RAM	SYP2114	U12, U13
1	32K BIT ROM	SYP2332	U20
1	NAND GATE	7400	U8
1	HEX INVERTER	7404	U2
1	AND GATE	7408	U24
2	HEX INVERTER-O.C.	7416	U30, U38
1	NAND GATE	74LS00	U4
1	HEX INVERTER	74LS04	U9
1	TRIPLE NOR GATE	74LS27	U3
1	TIMER	555	U6

QTY.	DESCRIPTION	MFR. NO.	REFERENCE DESIGNATION
1	DECODER	74LS138	U1
1	TRIPLE 3 INPUT NAND	74LS10	U7
1	DECODER	74145	U37
2	DECODER	74LS145	U10, U11
1	COMPARATOR	311	U26
1	RES-100 ohm, ¼W, 5%	RF14J100B	R128
3	RES-200 ohm, ¼W, 5%	RF14J200B	R43, 111, 114
1	RES-300 ohm, ¼W, 5%	RF14J300B	R107
4	RES-470 ohm, ¼W, 5%	RF14J470B	R84, 88, 124, 127
14	RES-1K, ¼W, 5%	RF14J1KB	R9-12, 41, 61-63, 73, 78, 85, 92, 97, 101, 113, 123
1	RES-1M, ¼W, 5%	RF14J1MB	R72
1	RES-2.2K, ¼W, 5%	RJ14J2.2KB	R103
14	RES-3.3K, ¼W, 5%	RF14J3.3KB	R42, 59, 60, 70, 74, 79-82, 87, 94, 95, 98, 126
10	RES-10K, ¼W, 5%	RF14J10KB	R45, 67-69, 75, 76, 83, 89, 93, 104
3	RES-47K, ¼W, 5%	RF14J47KB	R44, 46,71
1	RES-330K, ¼W, 5%	RF14J330KB	R77
2	RES-27K, ¼W, 5%	RF14J27KC	R90, 96
2	RES-150 ohm, ¼W, 5%	RF14J150B	R99, 110
1	RES-6.8K, ¼W, 5%	RF14J6.8KB	R100
1	CAP-10pf	DM15100J	C13
13	CAP - .01 mfd, 100V	DB203YZ1032	C1, 3, 5, 7, 10, 11, 17, 19, 21, 23, 25, 29
10	CAP - 10 mfd, 25V	T368B106K025PS	C2, 4, 6, 8, 12, 20, 22, 24, 26, 30

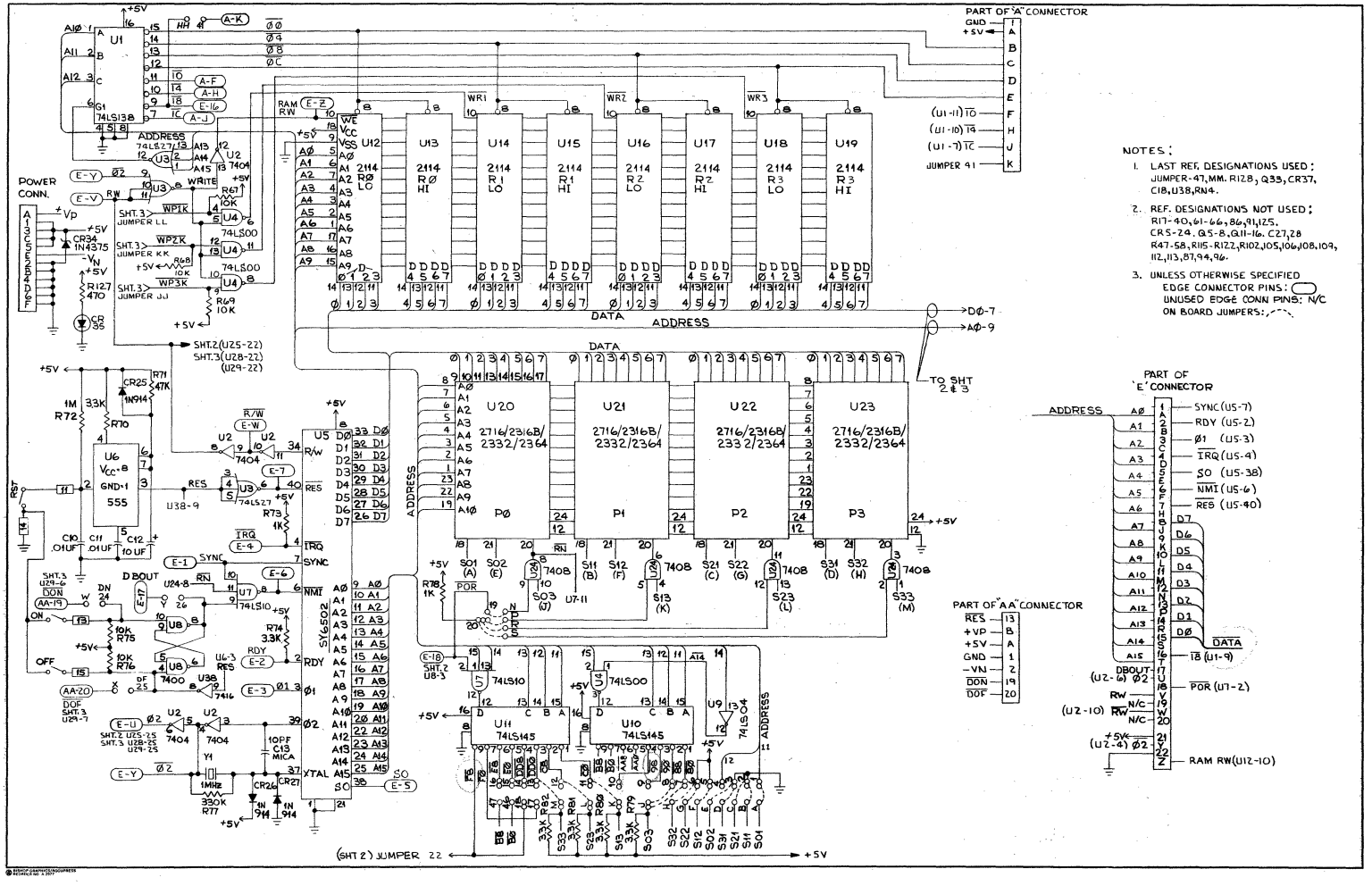
QTY.	DESCRIPTION	MFR. NO.	REFERENCE DESIGNATION
3	CAP - .1 mfd, 50V	3429-050E-104M	C9, 18
1	CAP - .22mfd,		C16
2	CAP - .47 mfd	C330C474M5V5EA	C15
1	CAP - .0047 mfd	UR2025100X7R472K	C14
12	NPN TRANSISTOR	2N2222A	Q1-4, 10, 18, 19, 27-29, 32, 33
11	PNP TRANSISTOR	2N2907A	Q9, 17, 20-26, 30, 31
11	DIODE, G.P.	1N914	CR25-33, 37, 38
1	DIODE, ZENER	1N4735	CR34
4	SOCKET - 24-PIN DIP	TIC8424-02	SK20-23
5	SOCKET - 40-PIN DIP	TIC8440-02	SK5, 25, 27-29
8	SOCKET - 18-PIN DIP	TIC8418-02	SK12-19
1	KEYBOARD		KB1
1	PC BOARD		PC1
6	7-SEGMENT DISPLAY, 0.3"	MAN 71A	U31-36
2	LED	RL4850	CR35,36
1	SPEAKER	70057	SP1
1	CRYSTAL	CY1A	Y1
	TAPE - 1½" x 2" STRIP		
1	RES. PACK - 150 ohm	898-3-R150	RN2
1	RES. PACK - 3.3K ohm	899-3-R3.3K	RN1
2	RES. PACK - 1K ohm	899-3-R1K	RN3, RN4


A CONNECTOR
POWER
P CONN
E CONNECTOR
Synertek Systems Corp.
SYM
MODEL 1
SYNERTEK SYSTEMS SYM
© 1978 SYNERTEK SYSTEMS CORP.
MADE IN U.S.A.

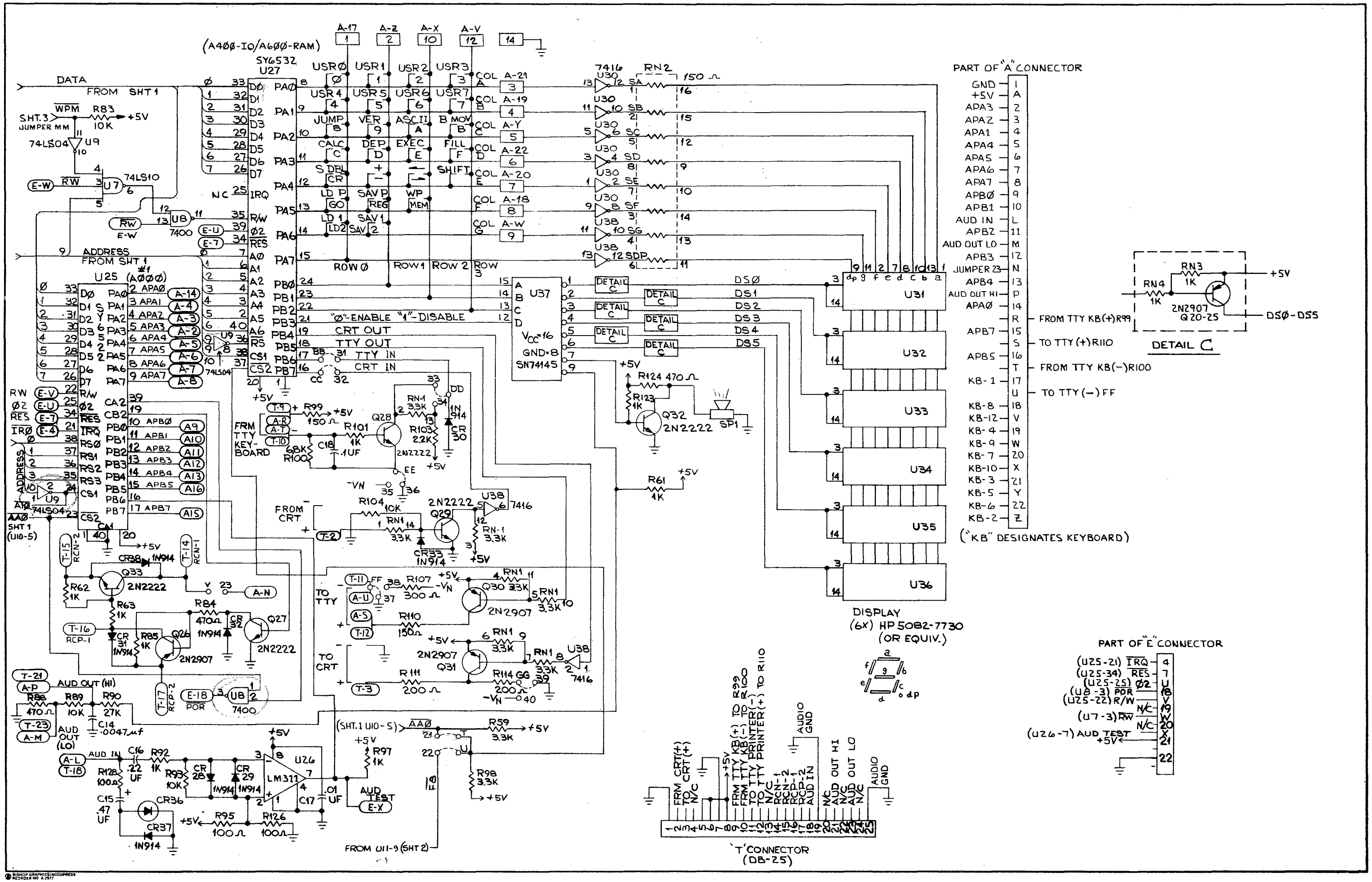


OUTLINE DRAWING

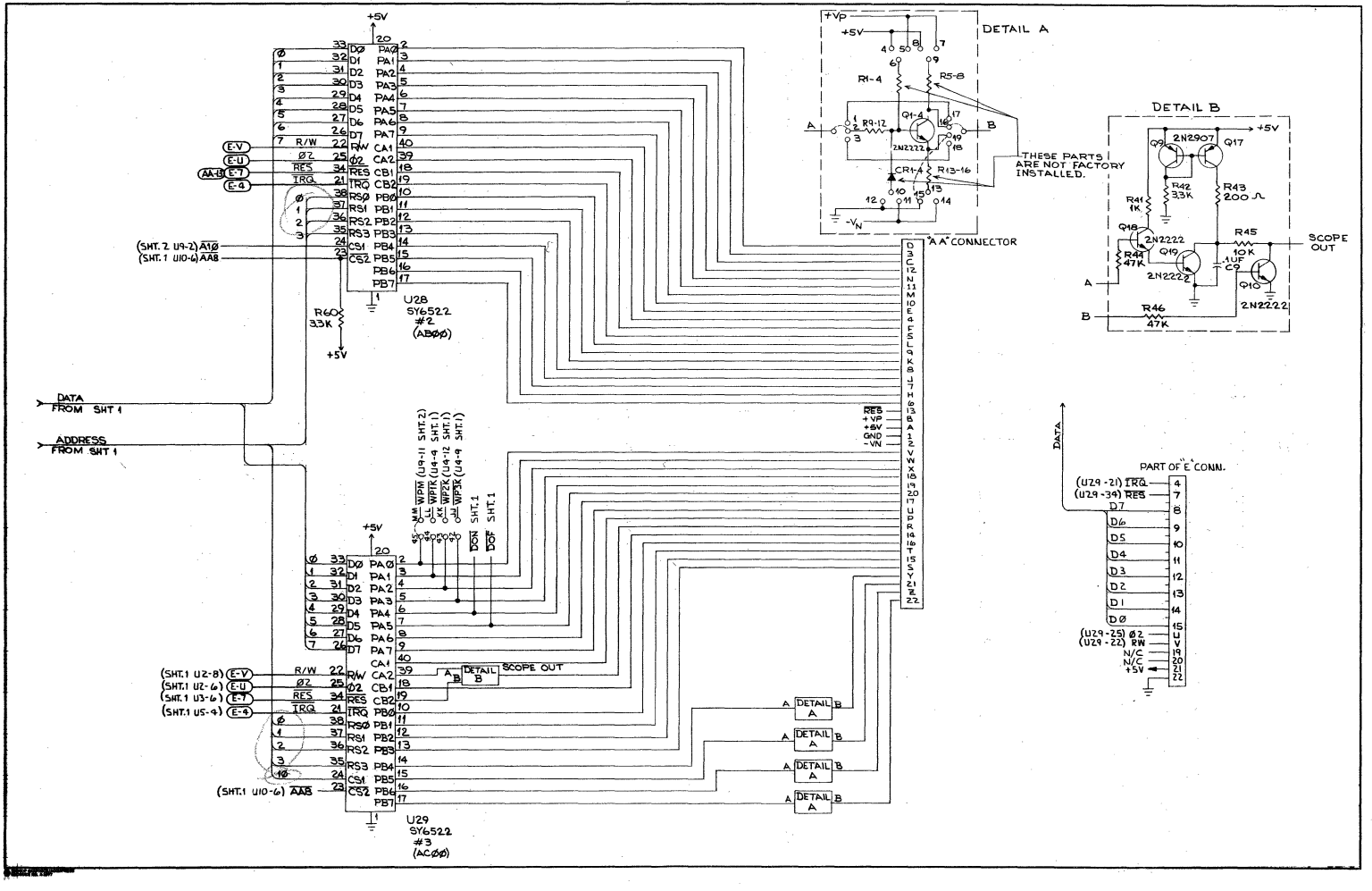
B-5


PART OF 'A' CONNECTOR
GND
+5V
JUMPER 41
NOTES:
1. LAST REF. DESIGNATIONS USED: JUMPER-47, MM. R128, Q33, CR37, C18, U38, RN4.
2. REF. DESIGNATIONS NOT USED: R17-40, 61-66, 86, 91, 125. CR5-29, 35-8, G11-16, C27, 28 R47-58, R115-R122, R102, 105, 106, 108, 109, 112, 113, 87, 94, 96.
3. UNLESS OTHERWISE SPECIFIED EDGE CONNECTOR PINS: UNUSED EDGE CONN PINS: N/C ON BOARD JUMPERS:
POWER CONN.
ADDRESS
DATA
TO SHT 2 & 3
PART OF 'E' CONNECTOR
PART OF 'AA' CONNECTOR
SHT.2 (U25-22)
SHT.3 (U28-22)
(U29-22)
(SHT 2) JUMPER 22
RAM RW(U12-10)

B-6



(A400-IO/A600-RAM)
SY6532
U27
DATA
FROM SHT 1
ADDRESS
FROM SHT 1
U25 (A000)
SHT.3 WPM
JUMPER MM
R83 10K
74LS04 U9
74LS10 U7
7400 U8
USR0 USR1 USR2 USR3
USR4 USR5 USR6 USR7
JUMP VER ASCII B MOV
CALC DEP EXEC FILL
S DBL SHIFT
LD P SAV P WP
GO REG MEM
LD 1 SAV 1
ROW 0 ROW 1 ROW 2 ROW 3
"0"-ENABLE "1"-DISABLE
CRT OUT
TTY OUT
TTY IN
CRT IN
U37
SN74145
7416
RN2 150 Ω
DETAIL C
DS0 DS1 DS2 DS3 DS4 DS5
U31 U32 U33 U34 U35 U36
DISPLAY
(6X) HP 5082-7730
(OR EQUIV.)
PART OF "A" CONNECTOR
GND +5V APA3 APA2 APA1 APA4 APA5 APA6 APA7 APB0 APB1 AUD IN APB2 AUD OUT LO APB3 JUMPER 23 APB4 AUD OUT HI APA0 APB7 APB5 KB-1 KB-8 KB-12 KB-4 KB-9 KB-7 KB-10 KB-3 KB-5 KB-6 KB-2
FROM TTY KB(+)R99
TO TTY (+)R110
FROM TTY KB(-)R100
TO TTY (-) FF
("KB" DESIGNATES KEYBOARD)
RN3 1K
RN4 1K
2N2907
Q20-25
+5V
DS0-DS5
DETAIL C
FRM TTY KEY-BOARD
FROM CRT
TO TTY
TO CRT
Q28 2N2222
Q29 2N2222
Q30 2N2907
Q31 2N2907
Q32 2N2222
Q33 2N2222
Q26 2N2907
Q27 2N2222
SP1
R124 470 Ω
R61 1K
AUD OUT (HI)
AUD OUT (LO)
AUD IN
LM311
U26
AUD TEST
(SHT.1 U10-5) AA0
FROM U11-9 (SHT 2)
POR
"T" CONNECTOR
(DB-25)
FRM CRT(+)
TO CRT(+)
N/C
+5V
FRM TTY KB(+) TO R99
FRM TTY KB(-) TO R100
TO TTY PRINTER(-)
TO TTY PRINTER(+) TO R110
AUDIO GND
RCN-1
RCP-1
RCN-2
RCP-2
AUD IN
AUD OUT HI
AUD OUT LO
PART OF "E" CONNECTOR
(U25-21) IRQ
(U25-34) RES
(U25-25) Ø2
(U8-3) POR
(U25-22) R/W
(U7-3) RW
(U26-7) AUD TEST


+5V
DETAIL A
+VP
THESE PARTS ARE NOT FACTORY INSTALLED.
"A" CONNECTOR
DETAIL B
SCOPE OUT
U28 SY6522 #2 (AB00)
U29 SY6522 #3 (AC00)
R60 33K
DATA FROM SHT 1
ADDRESS FROM SHT 1
(SHT. 2 U9-2) A10
(SHT. 1 U10-6) AA8
(SHT.1 U2-8) E-V
(SHT.1 U2-6) E-U
(SHT. 1 U3-6) E-7
(SHT.1 U5-6) E-4
R/W
IRQ
RES
WPW (U9-11 SHT.2)
WPTK (U4-9 SHT. 1)
WP2K (U4-12 SHT. 1)
WP3K (U4-9 SHT.1)
DON SHT.1
DOF SHT.1
PART OF "E" CONN.
(U29-21) IRQ
(U29-34) RES
(U29-25) Ø2
(U29-22) RW
+5V
GND
-VN

APPENDIX C

AUDIO TAPE FORMATS

HIGH-SPEED FORMAT -- High speed data transfer takes place at 185 bytes per second. Every byte consists of a start bit (0), followed by eight data bits. The least significant bit is transmitted first. A "1" bit is represented by 1 cycle of 1400 Hz, while a "0" bit is represented by ½ cycles of 700 Hz. Physical record format is shown below.

6 sec. of SYN chars.	*	ID	SAL	SAH	EAL +1	EAH +1	DATA	/	CKL	CKH	EOT	EOT

6 sec. of SYN (16 Hex) - Allows the tape to advance beyond the leader and creates an inter-record gap. (Controlled by TAPDEL $A630.) Filled with ASCII synch characters that allow the SYM-1 to synchronize with the data stream.

* (2A Hex) - ASCII character that indicates the start of a valid record.

ID - Single byte that uniquely identifies the record.

SAL - Low order byte of the Starting Address from which data was taken from memory.

SAH - High order byte of the Starting Address from which data was taken from memory.

EAL +1 - Low order byte of the address following the Ending Address from which data was taken from memory.

EAH +1 - High order byte of the address following the Ending Address from which data was taken from memory.

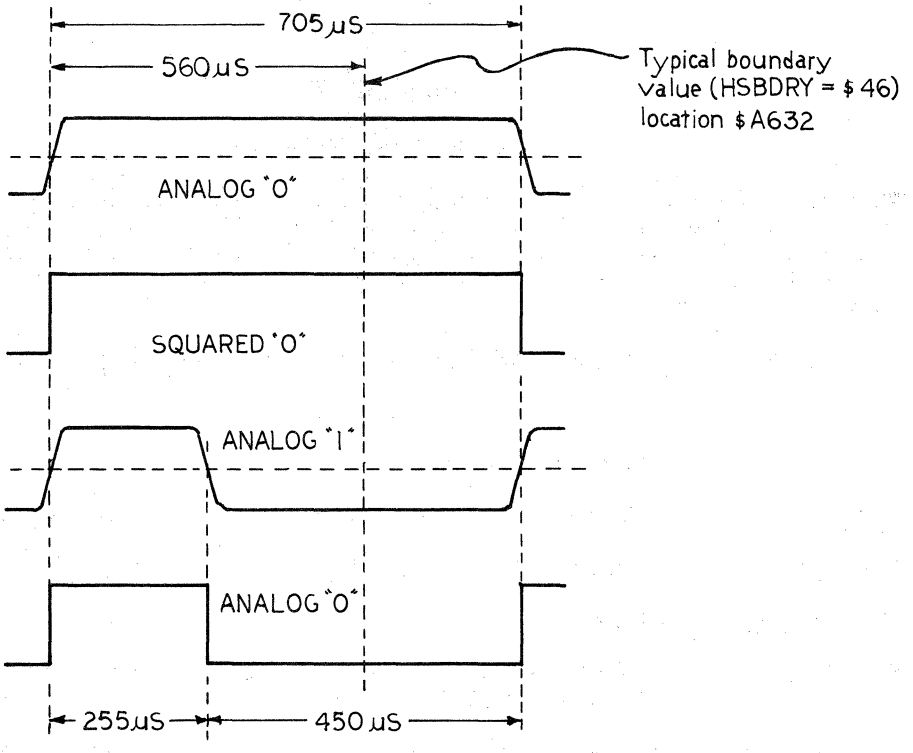
DATA - Data bytes.

/ (2F Hex) - ASCII character that indicates the end of the data position of a record.

CKL - Low order byte of a computed checksum.

CKH - High order byte of a computed checksum.

EOT (04 Hex) - ASCII characters that indicate the end of the tape record.



NAME	LOCATION	DEFAULT VALUE	DESCRIPTION
HSBDRY	A632	$ 46 (560µS)	HS Boundary
TAPET1	A635	$ 33 (255µS)	HS First Half "1" Bit
TAPET2	A63C	$ 5A (450µS)	HS Second Half "1" Bit

HIGH SPEED AUDIO FORMAT BIT WAVEFORMS

KIM FORMAT -- Data transfer in KIM format takes place at approximately 8 bytes per second. A "1" bit is represented by 9 cycles of 3600 Hz followed by 18 cycles of 2400 Hz, while a "0" bit is represented by 18 cycles of 3600 Hz followed by 6 cycles of 2400 Hz. Each 8-bit byte from memory is represented by two ASCII characters. The byte is separated into two half-bytes, then each half-byte is converted into an ASCII character that represents a Hex digit. The least significant bit is transmitted first. The KIM physical record format is shown below.

128 SYN chars.	*	ID	SAL	SAH	DATA	/	CKL	CKH	EOT	EOT

The sync characters, the ASCII characters "*" (2A Hex) and "/" (2F Hex) as well as ID, SAL, SAH, CKL, CKH and EOT serve the same functions as in HIGH-SPEED format. Sync characters, *, / and EOT are represented by single ASCII characters, while the remaining record items require two ASCII characters. Note that EAL and EAH are not used in the KIM format.

APPENDIX D

PAPER TAPE FORMAT

When data from memory is stored on paper tape, each 8-bit byte is separated into two half-bytes, then each half-byte is converted into an ASCII character that represents a Hex digit (O-F). Consequently, two ASCII characters are used to represent one byte of data. In the paper tape record format shown below, each N, A, D, and X represents one ASCII character.

; N_1N_0 $A_3A_2A_1A_0$ $(D_1D_0)_1$ $(D_1D_0)_2$ . . . $(D_1D_0)_n$ $X_3X_2X_1X_0$

;	- Start of record mark
N_1N_0	- Number of data bytes in (Hex) contained in the record
$A_3A_2A_1A_0$	- Starting address from which data was taken
(D_1D_0)-$(D_1D_0)_n$	- Data
$X_3X_2X_1X_0$	- 16-bit checksum of all preceding bytes in the record including N_1N_0 and $A_3A_2A_1A_0$, but excluding the start of record mark.

A single record will normally contain a maximum of 16 (10 Hex) data bytes. This is the system default value that is stored in system RAM at power-up or reset in location MAXRC (A658). You can substitute your own value by storing different number in MAXRC. To place an end of file after the last data record saved, place the TTY in local mode punch on, and enter ;00 followed by **(CR).**

APPENDIX E

SYM COMPATABILITY WITH KIM PRODUCTS

If you are a SYM-1 user who has peripheral devices which you have previously used with the KIM system or software which has been run on a KIM module, you'll find SYM to be generally upward compatible with your hardware and software. The following two sections describe the levels of compatability between the two systems to allow you to undertake any necessary modifications.

E.1 HARDWARE COMPATABILITY

Table E-1 describes the upward compatability between SYM and KIM at the Expansion (E) connector, while Table E-2 describes the compatability on the Applications (A) connector.

I/O port addresses differ between the two systems; you should consult the Memory Map in Figure 4-10 for details.

Power Supply inputs are provided on a separate connector with SYM-1, which means that if you have been using your power supply with a KIM device it will be necessary to rewire its connections to use the special connector on the SYM-1 board.

E.2 SOFTWARE COMPATABILITY

Table E-3 lists important user-available addresses and routines in the KIM-1 monitor program and their counterparts in SYM-1's SUPERMON. Most of the routines do not perform identically in the two systems. Before using them, check their operation in Table 9-1.

Table E-1. EXPANSION CONNECTOR (E) COMPATABILITY

SYM DESCRIPTION	SYM NAME	PIN #	KIM NAME	KIM DESCRIPTION
Jumper (Y,26) Selectable: OFF - Open Pin ON - Debug On/Off Output (U8-8)	DBOUT	17	SSTOUT	From (SYNC o NOT MONITOR) U26-6
Power On Reset Signal Output: "0" After power on "1" When reset by software	$\overline{POR}$	18		No equivalent

Table E-2. APPLICATION CONNECTOR (A) COMPATABILITY

SYM DESCRIPTION	SYM NAME	PIN #	KIM NAME	KIM DESCRIPTION
Jumper (V,23) Selectable: OFF - Open Pin ON - Remote Audio Control Out	AUD.RC	N	+12V	+12V Not required on SYM
Jumper (HH,41) Selectable: OFF Open Pin ON $\overline{\text{ICXX}}$ Decode Out		K	DECODE Enable	Enable 8K Decoder

Table E-3. SYM-KIM SOFTWARE COMPATABILITY

SYM		KIM		FUNCTION
Label	Address(es)	Label	Address(es)	
PCLR	A659	PCL	00EF	Program Counter - low
PCHR	A65A	PCH	00F0	Program Counter - high
FR	A65C	PREG	00F1	Status Register
SR	A65B	SPUSER	00F2	Stack Pointer
AR	A65D	ACC	00F3	Accumulator
YR	A65F	YREG	00F4	Y - Register
XR	A65E	XREG	00F5	X - Register
SCR6	A636	CHKHI	00F6	Checksum - low
SCR7	A637	CHKSUM	00F7	Checksum - high
P2L	A64C	SAL	17F5	Start Addr Low - audio/paper tape
P2H	A64D	SAH	17F6	Start Addr High - audio/paper tape
P3L	A64A	EAL	17F7	End Addr+1 Low - audio/paper tape
P3H	A64B	EAH	17F8	End Addr+1 High - audio/paper tape
P1L	A64E	ID	17F9	ID Byte audio Tape
NMIVEC	A67A-B FFFA-B	NMIV	17FA-B FFFA-B	NMI Vector
RSTVEC	FFFC-D	RSTV	17FC-D FFFC-D	Reset Vector
IRQVEC	A67E-F FFFE-F	IRQV	17FE-F FFFE-F	IRQ Vector
DUMPT	8E87	DUMPT	1800	Dump memory to audio tape
LOADT	8C78	LOADT	1873	Load memory from audio tape
CHKT	8E78	CHKT	194C	Compute checksum for audio tape
OUTBTC	8F4A	OUTBTC	195E	Output one KIM byte
HEXOUT	8F52	HEXOUT	196F	Convert LSD of A to ASCII AND write to audio tape

Table E-3. SYM-KIM SOFTWARE COMPATABILITY (Continued)

SYM		KIM		FUNCTION
Label	Address(es)	Label	Address(es)	
OUTCHT	8F5D	OUTCHT	197A	Write one ASCII character to audio tape
RDBYT	8E2C	RDBYT	19F3	Read one byte from audio tape
PACKT	8E3E	PACKT	1A00	Pack ASCII to nibble
RDCHT	8E61	RDCHT	1A24	Read one character from audio tape
RDBITK	8E0F	RDBIT	1A41	Read one bit from tape
SVNMI	809B	SAVE	1C00	Monitor NMI entry
RESET	8B4A	RST	1C22	Monitor RESET entry
OUTPC	82EE	PCCMD	1CDC	Display PC
INCHR	8A1B	READ	1C6A	Get character
LP2B+7	841E	LOAD	1CE7	Load paper tape
SP2B+4	869C	DUMP	1D42	Save paper tape
OUTS2	8319	PRTPNT	1E1E	Print pointer
OUTBYT	82FA	PRTBYT	1E3B	Print 1 byte as 2 ASCII character
INCHR	8A1B	GETCH	1E5A	Get character
DLYF	8AE6	DELAY	1ED4	Delay 1 bit time
DLYH	8AE9	DEHALF	1EEB	Delay ½ bit time
INSTAT	8386	AK	1EFE	Determine if key is down
OUTDSP	89C1	SCAND	1F19	Output to LED display
SCAND	8906	SCANDS	1F1F	Scan LED display
INCCMP	82B2	INCPT	1F63	Increment pointer
GETKEY	88AF	GETKEY	1F6A	Get key
CHKSAD	82DD	CHK	1F91	Compute checksum
INBYTE	81D9	GETBYT	1F9D	Get 2 Hex characters and pack

APPENDIX F

CREATING AND USING A SYNC TAPE

To read serial data from tape, the SYM-1 makes use of a stream of SYNC characters which form the leader of every tape record. For a complete description of audio tape record formats, refer to Appendix C.

The audio signal appears on the T and A connectors in two forms: Audio Out (HI) and Audio Out (LO). The only difference between these signals is their magnitude. It is usually best to connect Audio Out (LO) to the MIC input of your recorder.

When the SYM-1 searches for a record, an 'S' and a decimal point are displayed until the SYNC characters are recognized. However, if the volume and tone controls on the recorder are not set correctly, the SYNC characters will not be recognized, the 'S' on the display will not go out, and no data will be loaded into memory.

Before attempting to save and load data for the first time, it will be helpful to generate a SYNC tape to use for adjusting the controls.

To generate a SYNC tape, modify memory location TAPDEL, A630 to FF:

```
.M A630 )
A630, 04, FF
A631, 2C, )
```

Place a blank tape in the recorder, depress RECORD and PLAY on the recorder, and enter a SAVE command:

```
.S2 1-200-201 )
```

The length of the tape leader, determined by TAPDEL will be over 6 minutes.

When the recording is finished (the display re-lights), rewind the tape. Enter the load command:

```
.L2 )
```

The 'S' and decimal point should light. Start the tape with the PLAY button on the recorder and adjust the volume and tone controls on the recorder until the 'S' goes out and stays out.

You can now remove the SYNC tape and proceed to save and load actual programs and data.

APPENDIX G

MONITOR ADDENDA

1. The DBOFF routine does not debounce the DEBUG-ON switch; therefore, user programs should not be interrupted by depressing DEBUG-ON while using a user trace routine or while OUTVEC points to a user routine. (This will cause recursive interrupts.)

2. The audio cassette software will not read or write location $FFFF. Use $A67F ($A600 thru $A67F is echoed at $FF80 thru $FFFF).

3. The DEBUG-ON switch bounces, therefore it should not be used to interrupt user programs while using a user trace routine or while OUTVEC points to a user routine. (This will cause recursive interrupts.)

APPENDIX H

SUPPLEMENTARY INFORMATION

Changing Automatic Log-On

After power is applied to the SYM, SUPERMON waits for the keyboard or the device connected to PB7 on the 6532 (normally the RS232 device) to become active. PB6 (the current loop device) is ignored because a disconnected current loop always looks active.

If you expect always to log-on a current-loop device, the following jumper change will eliminate the necessity of entering (SHIFT) (JUMP) (1):

Change CC-32 and BB-31
to CC-31 and BB-32

Now the log-on for your current loop device is simply a "Q", entered at the device. (Note that you cannot now log-on automatically to the keyboard unless the current loop device is connected, and powered-up.)

Using On-Board LED Display

Because of the extensive use of transfer vectors in SUPERMON, the same monitor calls can be used to activate the LED display as for terminal devices. The major difference is that you must call ACCESS (address 8B86) before outputting the first character in order to remove write-protection from the display buffer (DISBUF, address A640 thru A645).

If the SYM-1 was logged-on to from the HKB, each call to OUTCHR (address 8A47) will examine the ASCII character in the Accumulator, look up its segment code, shift everything in the display buffer of segment codes left one digit, place the new code in the rightmost digit, and scan the display once.

If the SYM-1 was logged-on to the HKB, each call to INCHR (address 8A1B) will scan the display from the codes in DISBUF continuously until a key is depressed (2 keys in the case of SHIFT keys, 4 in the case of SHIFT ASCII keys). The key will be fully debounced, the beeper beeped, the ASCII or HASHED ASCII code taken from a table, and passed back to the caller in the Accumulator. The Flags will reflect a compare with carriage-return.

Other useful routines are:

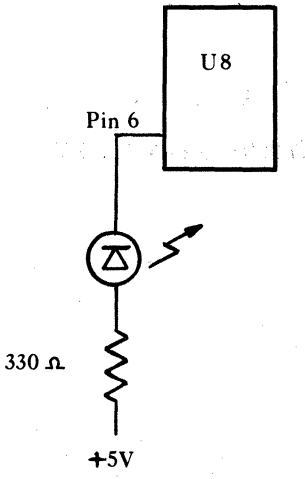
GETKEY (88AF)	Same as description of INCHR above, but disregard log-on and no compare performed.
OUTDSP (89C1)	Same as description of OUTCHR above, but disregard log-on.
KEYQ (8923)	Test for key depressed on HKB. On return, Z Flag = 1 if key down.
SCAND (8906)	Scan display once from segment codes in DISBUF. On return, Flags reflect call to KEYQ.

INSTAT (8386) — If logged-on to HKB, check for key down (else check for BREAK key). On return, carry set if key down (or BREAK key). Leading edge of key debounced.

See also chapter 9 for discussion of monitor calls.

Adding DEBUG Indicator

While using trace routines which turn DEBUG on and off, it is often desirable to have an external indication of the DEBUG state. The addition of an LED and a resistor as follows will achieve this.



U8 is a 14 pin package located above the beeper.

The LED will remain on while DEBUG is on.

APPENDIX I

SY6502 DATA SHEET

8-Bit Microprocessor Family

SY6500

MICROPROCESSOR PRODUCTS

APRIL 1979

- Single 5 V ±5% power supply
- N channel, silicon gate, depletion load technology
- Eight bit parallel processing
- 56 Instructions
- Decimal and binary arithmetic
- Thirteen addressing modes
- True indexing capability
- Programmable stack pointer
- Variable length stack
- Interrupt capability
- Non-maskable interrupt
- Use with any type or speed memory
- Bi-directional Data Bus
- Instruction decoding and control
- Addressable memory range of up to 65 K bytes
- "Ready" input
- Direct memory access capability
- Bus compatible with MC6800
- Choice of external or on-board clocks
- 1 MHz, 2 MHz, and 3 MHz operation
- On-chip clock options
  * External single clock input
  * Crystal time base input
- 40 and 28 pin package versions
- Pipeline architecture

The SY6500 Series Microprocessors represent the first totally software compatible microprocessor family. This family of products includes a range of software compatible microprocessors which provide a selection of addressable memory range, interrupt input options and on-chip clock oscillators and drivers. All of the microprocessors in the SY6500 family are software compatible within the group and are bus compatible with the MC6800 product offering.

The family includes six microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single phase inputs or crystals provide the time base. The external clock versions are geared for the multi-processor system applications where maximum timing control is mandatory. All versions of the microprocessors are available in 1 MHz, 2 MHz, and 3 MHz maximum operating frequencies.

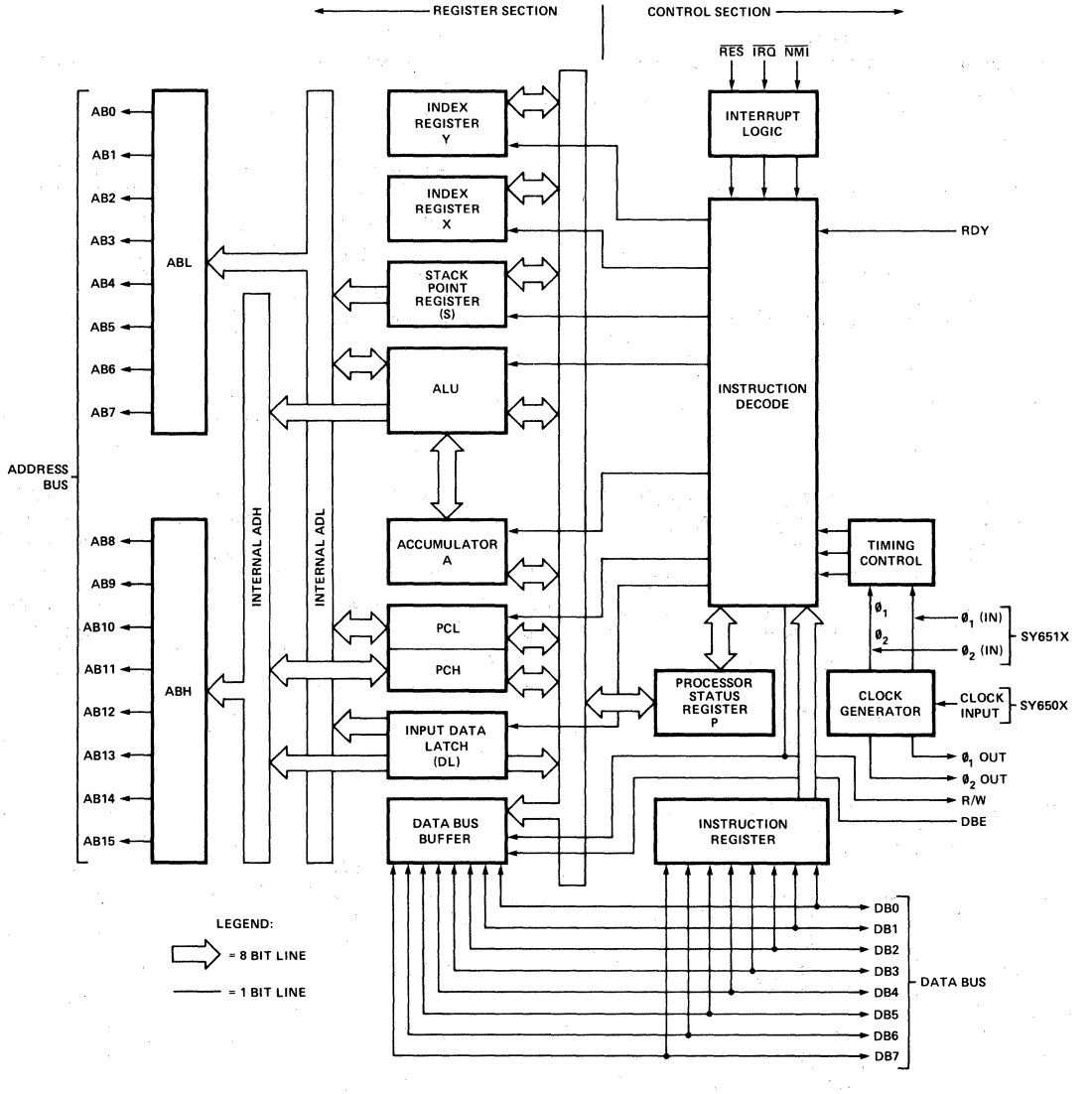
MEMBERS OF THE FAMILY

PART NUMBERS		CLOCKS	PINS	$\overline{IRQ}$	$\overline{NMI}$	RDY	ADDRESSING
Plastic	Ceramic						
SYP6502	SYC6502	On-Chip	40	√	√	√	16 (64 K)
SYP6503	SYC6503	"	28	√	√		12 (4 K)
SYP6504	SYC6504	"	28	√			13 (8 K)
SYP6505	SYC6505	"	28	√		√	12 (4 K)
SYP6506	SYC6506	"	28	√			12 (4 K)
SYP6507	SYC6507	"	28			√	13 (8 K)
SYP6512	SYC6512	External	40	√	√	√	16 (64 K)
SYP6513	SYC6513	"	28	√	√		12 (4 K)
SYP6514	SYC6514	"	28	√			13 (8 K)
SYP6515	SYC6515	"	28	√		√	12 (4 K)

COMMENTS ON THE DATA SHEET

The data sheet is constructed to review first the basic "Common Characteristics" — those features which are common to the general family of microprocessors. Subsequent to a review of the family characteristics will be sections devoted to each member of the group with specific features of each.

SY6500 INTERNAL ARCHITECTURE



NOTE:
1. CLOCK GENERATOR IS NOT INCLUDED ON SY651X.
2. ADDRESSING CAPABILITY AND CONTROL OPTIONS VARY WITH EACH OF THE SY6500 PRODUCTS.

D.C. CHARACTERISTICS

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature	T_A	0 to +70	°C
Storage Temperature	T_{STG}	−55 to +150	°C

COMMENT

This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0V ± 5%, T_A = 0–70°C)

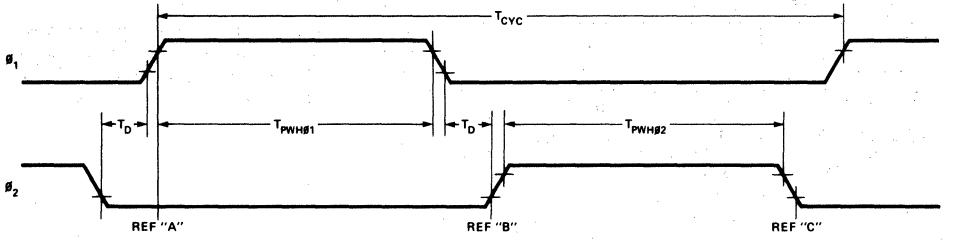
($\emptyset_1$, $\emptyset_2$ applies to SY651X, $\emptyset_{o\,(in)}$ applies to SY650X)

Symbol	Characteristic	Min.	Max.	Unit
V_{IH}	Input High Voltage			
	Logic, $\emptyset_{o\,(in)}$ (650X)	+2.4	V_{CC}	
	$\emptyset_1$, $\emptyset_2$ (651X)	V_{CC} − 0.5	V_{CC} + 0.25	V
V_{IL}	Input Low Voltage			
	Logic, $\emptyset_{o\,(in)}$ (650X)	−0.3	+0.4	
	$\emptyset_1$, $\emptyset_2$ (651X)	−0.3	+0.2	V
I_{IL}	Input Loading (V_{in} = 0 V, V_{CC} = 5.25 V) RDY, S.O.	−10	−300	μA
I_{in}	Input Leakage Current (V_{in} = 0 to 5.25 V, V_{CC} = 0)			
	Logic (Excl. RDY, S.O.)	–	2.5	μA
	$\emptyset_1$, $\emptyset_2$ (651X)	–	100	μA
	$\emptyset_{o\,(in)}$ (650X)	–	10.0	μA
I_{TSI}	Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V, V_{CC} = 5.25 V) DB0-DB7	–	10	μA
V_{OH}	Output High Voltage (I_{LOAD} = −100μAdc, V_{CC} = 4.75 V) SYNC, DB0-DB7, A0-A15, R/W	2.4	–	V
V_{OL}	Output Low Voltage (I_{LOAD} = 1.6mAdc, V_{CC} = 4.75 V) SYNC, DB0-DB7, A0-A15, R/W	–	0.4	V
P_D	Power Dissipation			
	1 MHz and 2 MHz	–	700	mW
	3 MHz	–	800	mW
C	Capacitance (V_{in} = 0, T_A = 25°C, f = 1 MHz)			
C_{in}	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, S.O., DBE	–	10	
	DB0-DB7	–	15	
C_{out}	A0-A15, R/W, SYNC	–	12	pF
$C_{\emptyset o(in)}$	$\emptyset_{o\,(in)}$ (650X)	–	15	
$C_{\emptyset 1}$	$\emptyset_1$ (651X)	–	50	
$C_{\emptyset 2}$	$\emptyset_2$ (651X)	–	80	

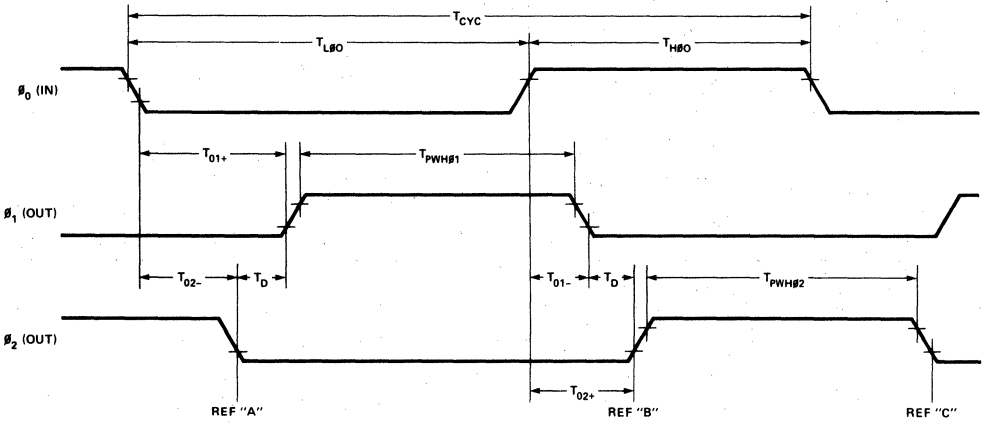
Note: $\overline{IRQ}$ and $\overline{NMI}$ require 3 K pull-up resistors.

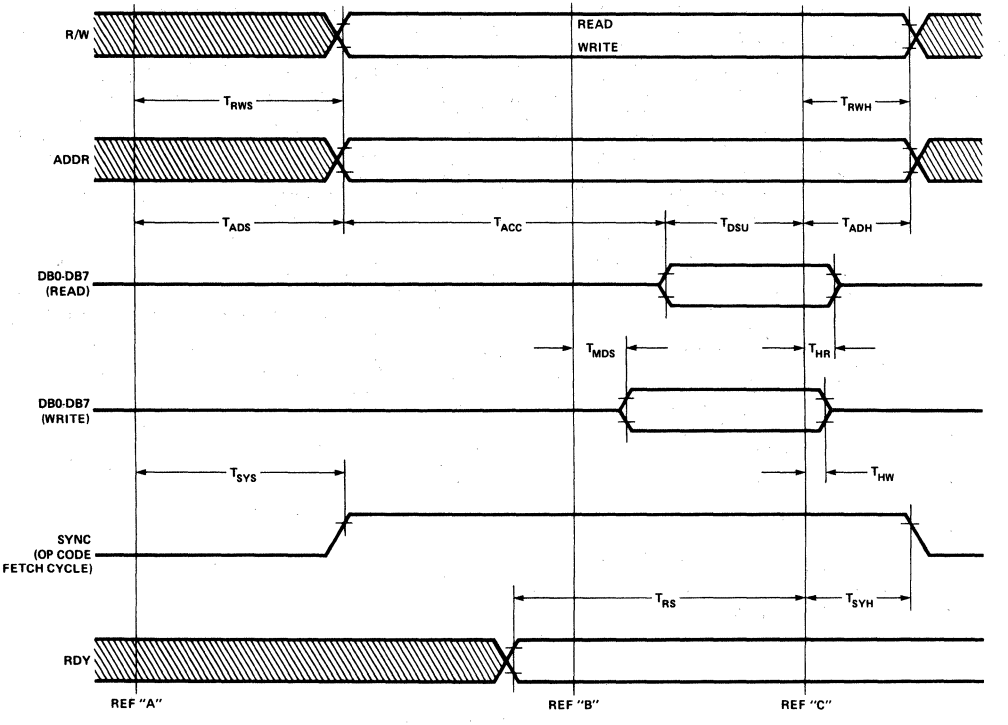
TIMING DEFINITIONS

SY651X INPUT CLOCK TIMING



SY650X INPUT CLOCK TIMING





DYNAMIC OPERATING CHARACTERISTICS

(V_{CC} = 5.0 ± 5%, T_A = 0° to 70°C)

Device Type	Parameter	Note	Symbol	1 MHz		2 MHz ⑥		3 MHz ⑦		Unit
				Min.	Max.	Min.	Max.	Min.	Max.	
651X	Cycle Time		T_{CYC}	1.00	40	0.50	40	0.33	40	μs
	$\emptyset_1$ Pulse Width		$T_{PWH\emptyset_1}$	430	–	215	–	150	–	ns
	$\emptyset_2$ Pulse Width		$T_{PWH\emptyset_2}$	470	–	235	–	160	–	ns
	Delay Between $\emptyset_1$ and $\emptyset_2$		T_D	0	–	0	–	0	–	ns
	$\emptyset_1$ and $\emptyset_2$ Rise and Fall Times	①	T_R, T_F	0	25	0	20	0	15	ns
650X	Cycle Time		T_{CYC}	1.00	40	0.50	40	0.33	40	μs
	$\emptyset_{o(IN)}$ Low Time	②	$T_{L\emptyset_o}$	480	–	240	–	160	–	ns
	$\emptyset_{o(IN)}$ High Time	②	$T_{H\emptyset_o}$	460	–	240	–	160	–	ns
	$\emptyset_o$ Neg to $\emptyset_1$ Pos Delay	⑤	T_{01+}	10	70	10	70	10	70	ns
	$\emptyset_o$ Neg to $\emptyset_2$ Neg Delay	⑤	T_{02-}	5	65	5	65	5	65	ns
	$\emptyset_o$ Pos to $\emptyset_1$ Neg Delay	⑤	T_{01-}	5	65	5	65	5	65	ns
	$\emptyset_o$ Pos to $\emptyset_2$ Pos Delay	⑤	T_{02+}	15	75	15	75	15	75	ns
	$\emptyset_{o(IN)}$ Rise and Fall Time	①	T_{RO}, T_{FO}	0	10	0	10	0	10	ns
	$\emptyset_{1(OUT)}$ Pulse Width		$T_{PWH\emptyset_1}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	ns
	$\emptyset_{2(OUT)}$ Pulse Width		$T_{PWH\emptyset_2}$	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	ns
	Delay Between $\emptyset_1$ and $\emptyset_2$		T_D	5	–	5	–	5	–	ns
	$\emptyset_1$ and $\emptyset_2$ Rise and Fall Times	①③	T_R, T_F	–	25	–	25	–	15	ns
650X 651X	R/W Setup Time		T_{RWS}	–	225	–	140	–	110	ns
	R/W Hold Time		T_{RWH}	30	–	30	–	15	–	ns
	Address Setup Time		T_{ADS}	–	225	–	140	–	110	ns
	Address Hold Time		T_{ADH}	30	–	30	–	15	–	ns
	Read Access Time		T_{ACC}	–	650	–	310	–	170	ns
	Read Data Setup Time		T_{DSU}	100	–	50	–	50	–	ns
	Read Data Hold Time		T_{HR}	10	–	10	–	10	–	ns
	Write Data Setup Time		T_{MDS}	–	175	–	100	–	75	ns
	Write Data Hold Time		T_{HW}	60	–	60	–	30	–	ns
	Sync Setup Time		T_{SYS}	–	350	–	175	–	100	ns
	Sync Hold Time		T_{SYH}	30	–	30	–	15	–	ns
	RDY Setup Time	④	T_{RS}	200	–	200	–	150	–	ns

NOTES:

① Measured between 10% and 90% points on waveform.

② Measured at 50% points.

③ Load = 1 TTL load +30 pF.

④ RDY must never switch states within T_{RS} to end of $\emptyset_2$.

⑤ Load = 100 pF.

⑥ The 2 MHz devices are identified by an "A" suffix.

⑦ The 3 MHz devices are identified by a "B" suffix.

PIN FUNCTIONS

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

The SY651X requires a two phase non-overlapping clock that runs at the V_{CC} voltage level.

The SY650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled. Clock generator circuits are shown elsewhere in this data sheet.

Address Bus (A_0-A_{15}) (See sections on each micro for respective address lines on those devices.)

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

Data Bus (DB_0-DB_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 three-state buffers, capable of driving one standard TTL load and 130 pF.

Data Bus Enable (DBE)

This TTL compatible input allows external control of the three-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal operation DBE would be driven by the phase two ($\emptyset_2$) clock, thus allowing data output from microprocessor only during $\emptyset_2$. During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low. This signal is available on the SY6512, only.

Ready (RDY)

This input signal allows the user to halt 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 low. 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. Ready transitions must not be permitted during $\emptyset_2$ time.

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 any interrupt to be recognized. A 3KΩ external resistor should be used for proper wire-OR operation.

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

A negative going transition 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 interrupt mask flag. The vestor address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, thereby transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ also requires an external 3KΩ resistor to V_{CC} 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.)

A NEGATIVE going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of $\emptyset_1$.

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 ($\overline{RES}$)

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 V_{CC} reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this time the R/W and SYNC signal will become valid.

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

Read/Write (R/W)

This output signal is used to control the direction of data transfers between the processor and other circuits on the data bus. A high level on R/W signifies data into the processor; a low is for data transfer out of the processor.

PROGRAMMING CHARACTERISTICS

INSTRUCTION SET – ALPHABETIC SEQUENCE

ADC	Add Memory to Accumulator with Carry	DEC	Decrement Memory by One	PHA	Push Accumulator on Stack
AND	"AND" Memory with Accumulator	DEX	Decrement Index X by One	PHP	Push Processor Status on Stack
ASL	Shift left One Bit (Memory or Accumulator)	DEY	Decrement Index Y by One	PLA	Pull Accumulator from Stack
				PLP	Pull Processor Status from Stack
BCC	Branch on Carry Clear	EOR	"Exclusive-or" Memory with Accumulator		
BCS	Branch on Carry Set			ROL	Rotate One Bit Left (Memory or Accumulator)
BEQ	Branch on Result Zero	INC	Increment Memory by One	ROR	Rotate One Bit Right (Memory or Accumulator)
BIT	Test Bits in Memory with Accumulator	INX	Increment Index X by One	RTI	Return from Interrupt
BMI	Branch on Result Minus	INY	Increment Index Y by One	RTS	Return from Subroutine
BNE	Branch on Result not Zero				
BPL	Branch on Result Plus	JMP	Jump to New Location	SBC	Subtract Memory from Accumulator with Borrow
BRK	Force Break	JSR	Jump to New Location Saving Return Address	SEC	Set Carry Flag
BVC	Branch on Overflow Clear			SED	Set Decimal Mode
BVS	Branch on Overflow Set	LDA	Load Accumulator with Memory	SEI	Set Interupt Disable Status
		LDX	Load Index X with Memory	STA	Store Accumulator in Memory
CLC	Clear Carry Flag	LDY	Load Index Y with Memory	STX	Store Index X in Memory
CLD	Clear Decimal Mode	LSR	Shift One Bit Right (Memory or Accumulator)	STY	Store Index Y in Memory
CLI	Clear Interrupt Disable Bit				
CLV	Clear Overflow Flag	NOP	No Operation	TAX	Transfer Accumulator to Index X
CMP	Compare Memory and Accumulator			TAY	Transfer Accumulator to Index Y
CPX	Compare Memory and Index X	ORA	"OR" Memory with Accumulator	TSX	Transfer Stack Pointer to Index X
CPY	Compare Memory and Index Y			TXA	Transfer Index X to Accumulator
				TXS	Transfer Index X to Stack Pointer
				TYA	Transfer Index Y to Accumulator

ADDRESSING MODES

Accumulator Addressing

This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

Immediate Addressing

In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing

In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

Zero Page Addressing

The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Zero Page Addressing – (X, Y indexing)

This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y." The effective address is calcuated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing – (X, Y indexing)

This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X," and "Absolute, Y." The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing

In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing

Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is −128 to +127 bytes from the next instruction.

Indexed Indirect Addressing

In indexed indirect addressing (referred to as (Indirect,X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

Indirect Indexed Addressing

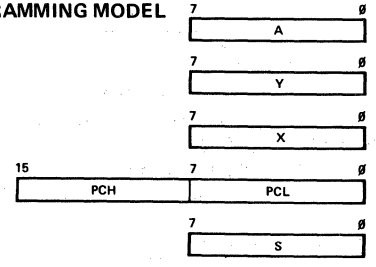
In indirect indexed addressing (referred to as (Indirect),Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

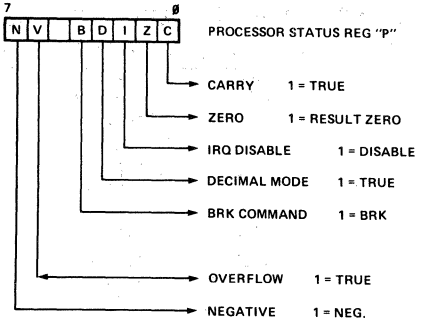
Absolute Indirect

The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter.

PROGRAMMING CHARACTERISTICS

PROGRAMMING MODEL





INSTRUCTION SET – OP CODES, EXECUTION TIME, MEMORY REQUIREMENTS

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			(IND, X)			(IND), Y			Z PAGE, X			ABS, X			ABS, Y			RELATIVE			INDIRECT			Z PAGE, Y			CONDITION CODES					
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
ADC	A + M + C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										✓	✓	✓	–	–	✓
AND	A ∧ M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										✓	✓	–	–	–	–
ASL	C ← 7 0 ← Ø				ØE	6	3	Ø6	5	2	ØA	2	1										16	6	2	1E	7	3												✓	✓	✓	–	–	–
BCC	BRANCH ON C=Ø (2)																															90	2	2							–	–	–	–	–	–
BCS	BRANCH ON C=1 (2)																															B0	2	2							–	–	–	–	–	–
BEQ	BRANCH ON Z=1 (2)																															F0	2	2							–	–	–	–	–	–
BIT	A ∧ M				2C	4	3	24	3	2																															M_7	✓	–	–	–	M_6
BMI	BRANCH ON N=1 (2)																															30	2	2							–	–	–	–	–	–
BNE	BRANCH ON Z=Ø (2)																															D0	2	2							–	–	–	–	–	–
BPL	BRANCH ON N=Ø (2)																															10	2	2							–	–	–	–	–	–
BRK	(See Fig. 1)													00	7	1																									–	–	–	–	–	–
BVC	BRANCH ON V=Ø (2)																															50	2	2							–	–	–	–	–	–
BVS	BRANCH ON V=1 (2)																															70	2	2							–	–	–	–	–	–
CLC	Ø → C													18	2	1																									–	–	Ø	–	–	–
CLD	Ø → D													D8	2	1																									–	–	–	–	Ø	–
CLI	Ø → I													58	2	1																									–	–	–	Ø	–	–
CLV	Ø → V													B8	2	1																									–	–	–	–	–	Ø
CMP	A-M (1)	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										✓	✓	✓	–	–	–
CPX	X-M	E0	2	2	EC	4	3	E4	3	2																															✓	✓	✓	–	–	–
CPY	Y-M	C0	2	2	CC	4	3	C4	3	2																															✓	✓	✓	–	–	–
DEC	M-1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													✓	✓	–	–	–	–
DEX	X-1 → X													CA	2	1																									✓	✓	–	–	–	–
DEY	Y-1 → Y													88	2	1																									✓	✓	–	–	–	–
EOR	A ⊻ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										✓	✓	–	–	–	–
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													✓	✓	–	–	–	–
INX	X + 1 → X													E8	2	1																									✓	✓	–	–	–	–
INY	Y + 1 → Y													C8	2	1																									✓	✓	–	–	–	–
JMP	JUMP TO NEW LOC				4C	3	3																													6C	5	3				–	–	–	–	–	–
JSR	(See Fig. 2) JUMP SUB				20	6	3																																		–	–	–	–	–	–
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										✓	✓	–	–	–	–

MNEMONIC	OPERATION	IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			(IND, X)			(IND), Y			Z PAGE, X			ABS, X			ABS, Y			RELATIVE			INDIRECT			Z PAGE, Y			CONDITION CODES					
		OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							86	4	2	✓	✓	–	–	–	–
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													✓	✓	–	–	–	–
LSR	Ø → 7 0 → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													Ø	✓	✓	–	–	–
NOP	NO OPERATION													EA	2	1																									–	–	–	–	–	–
ORA	A ∨ M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										✓	✓	–	–	–	–
PHA	A → Ms S-1 → S													48	3	1																									–	–	–	–	–	–
PHP	P → Ms S-1 → S													08	3	1																									–	–	–	–	–	–
PLA	S + 1 → S Ms → A													68	4	1																									✓	✓	–	–	–	–
PLP	S + 1 → S Ms → P													28	4	1																									(RESTORED)					
ROL	7 0 C				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													✓	✓	✓	–	–	–
ROR	C 7 0				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													✓	✓	✓	–	–	–
RTI	(See Fig. 1) RTRN INT.													40	6	1																									RESTORED)					
RTS	(See Fig. 2) RTRN SUB													60	6	1																									–	–	–	–	–	–
SBC	A-M-$\bar{C}$ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										✓	✓	(3)	–	–	✓
SEC	1 → C													38	2	1																									–	–	1	–	–	–
SED	1 → D													F8	2	1																									–	–	–	–	1	–
SEI	1 → I													78	2	1																									–	–	–	1	–	–
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										–	–	–	–	–	–
STX	X → M				8E	4	3	86	3	2																															96	4	2	–	–	–	–	–	–
STY	Y → M				8C	4	3	84	3	2													94	4	2																	–	–	–	–	–	–
TAX	A → X													AA	2	1																									✓	✓	–	–	–	–
TAY	A → Y													A8	2	1																									✓	✓	–	–	–	–
TSX	S → X													BA	2	1																									✓	✓	–	–	–	–
TXA	X → A													8A	2	1																									✓	✓	–	–	–	–
TXS	X → S													9A	2	1																									–	–	–	–	–	–
TYA	Y → A													98	2	1																									✓	✓	–	–	–	–

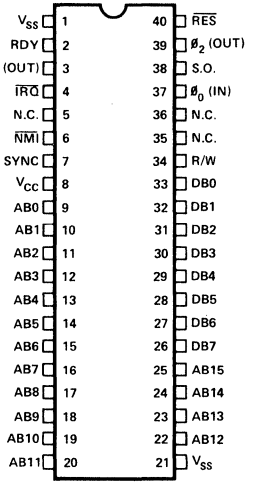
(1) ADD 1 TO "N" IF PAGE BOUNDARY IS CROSSED
(2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT = BELOW
(4) IF IN DECIMAL MODE Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
Ms MEMORY PER STACK POINTER

+ ADD
– SUBTRACT
∧ AND
∨ OR
⊻ EXCLUSIVE OR
✓ MODIFIED

– NOT MODIFIED
M_7 MEMORY BIT 7
M_6 MEMORY BIT 6
N NO CYCLES
NO BYTES

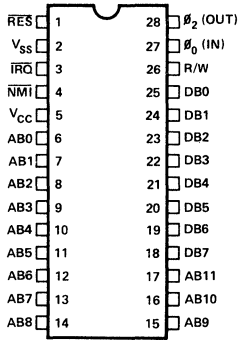
SY6502 – 40 Pin Package



Features

- 65K Addressable Bytes of Memory
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- On-the-chip Clock
  - √ TTL Level Single Phase Input
  - √ Crystal Time Base Input
- SYNC Signal
  (can be used for single instruction execution)
- RDY Signal
  (can be used for single cycle execution)
- Two Phase Output Clock for Timing of Support Chips

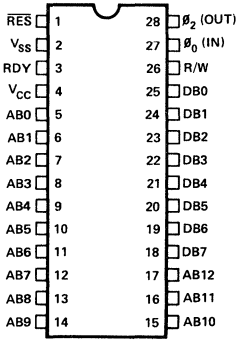
SY6503 – 28 Pin Package



Features

- 4K Addressable Bytes of Memory (AB00-AB11)
- On-the-chip Clock
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- 8 Bit Bi-Directional Data Bus

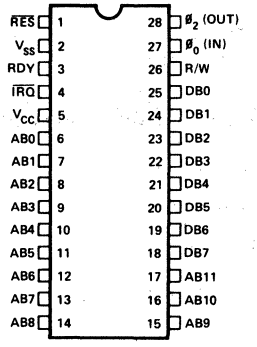
SY6504 & SY6507 – 28 Pin Package



Features

- $\overline{IRQ}$ Interrupt (6504 only)
- RDY Signal (6507 only)
- 8K Addressable Bytes of Memory (AB00-AB12)
- On-the-chip Clock
- 8 Bit Bi-Directional Data Bus

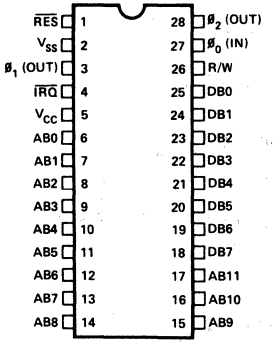
SY6505 — 28 Pin Package



Features

- 4K Addressable Bytes of Memory (AB00-AB11)
- On-the-chip Clock
- $\overline{IRQ}$ Interrupt
- RDY Signal
- 8 Bit Bi-Directional Data Bus

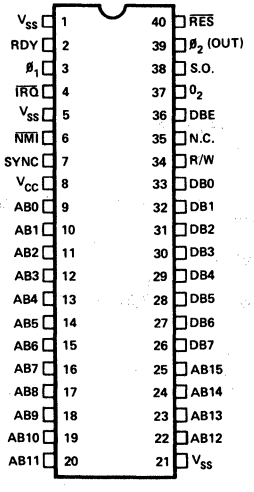
SY6506 — 28 Pin Package



Features

- 4K Addressable Bytes of Memory (AB00-AB11)
- On-the-chip Clock
- $\overline{IRQ}$ Interrupt
- Two phases off
- 8 Bit Bi-Directional Data Bus

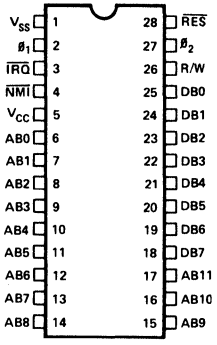
SY6512 — 40 Pin Package



Features

- 65K Addressable Bytes of Memory
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- RDY Signal
- 8 Bit Bi-Directional Data Bus
- SYNC Signal
- Two phase input
- Data Bus Enable

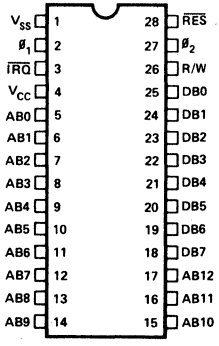
SY6513 – 28 Pin Package



Features

- 4K Addressable Bytes of Memory (AB00-AB11)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- 8 Bit Bi-Directional Data Bus

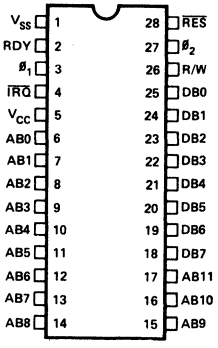
SY6514 – 28 Pin Package



Features

- 8K Addressable Bytes of Memory (AB00-AB12)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- 8 Bit Bi-Directional Data Bus

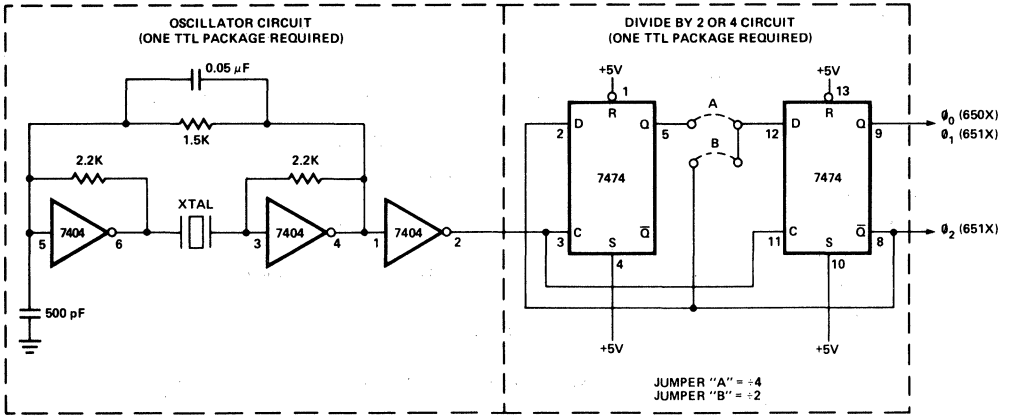
SY6515 – 28 Pin Package



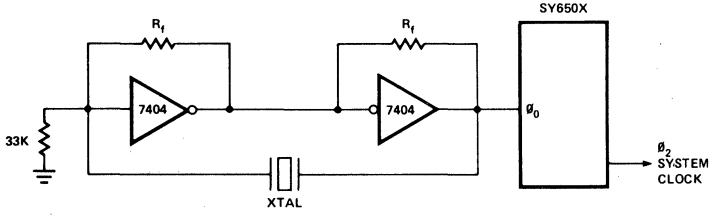
Features

- 4K Addressable Bytes of Memory (AB00-AB11)
- Two phase clock input
- IRQ Interrupt
- 8 Bit Bi-Directional Data Bus

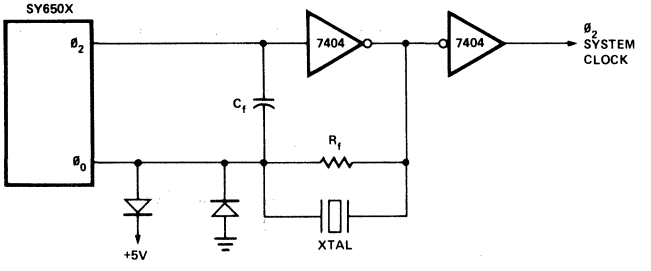
CLOCK GENERATION CIRCUITS



CRYSTAL FREQUENCY	OUTPUT FREQUENCY	
	÷2	÷4
3.579545 MHz	1.7897 MHz	0.894886 MHz
4.194304 MHz	2.097152 MHz	1.048576 MHz



R_f = 1.8K ohms
XTAL = 1 MHz – 5 MHz
(CTS KNIGHT MP SERIES OR EQUIVALENT)



R_f = 330K ohms
C_f = 10 pF
XTAL – CTS KNIGHT MP SERIES OR EQUIVALENT

APPENDIX J

SY6522 DATA SHEET

Versatile Interface Adapter (VIA)

Synertek INCORPORATED

SY6522 SY6522A

MICROPROCESSOR PRODUCTS

Preliminary

APRIL 1979

- Two 8-Bit Bidirectional I/O Ports
- Two 16-Bit Programmable Timer/Counters
- Serial Data Port
- Single +5V Power Supply
- TTL Compatible
- CMOS Compatible Peripheral Control Lines
- Expanded "Handshake" Capability Allows Positive Control of Data Transfers Between Processor and Peripheral Devices
- Latched Output and Input Registers
- 1 MHz and 2 MHz Operation

The SY6522 Versatile Interface Adapter (VIA) is a very flexible I/O control device. In addition, this device contains a pair of very powerful 16-bit interval timers, a serial-to-parallel/parallel-to-serial shift register and input data latching on the peripheral ports. Expanded handshaking capability allows control of bi-directional data transfers between VIA's in multiple processor systems.

Control of peripheral devices is handled primarily through two 8-bit bi-directional ports. Each line can be programmed as either an input or an output. Several peripheral I/O lines can be controlled directly from the interval timers for generating programmable frequency square waves or for counting externally generated pulses. To facilitate control of the many powerful features of this chip, an interrupt flag register, an interrupt enable register and a pair of function control registers are provided.

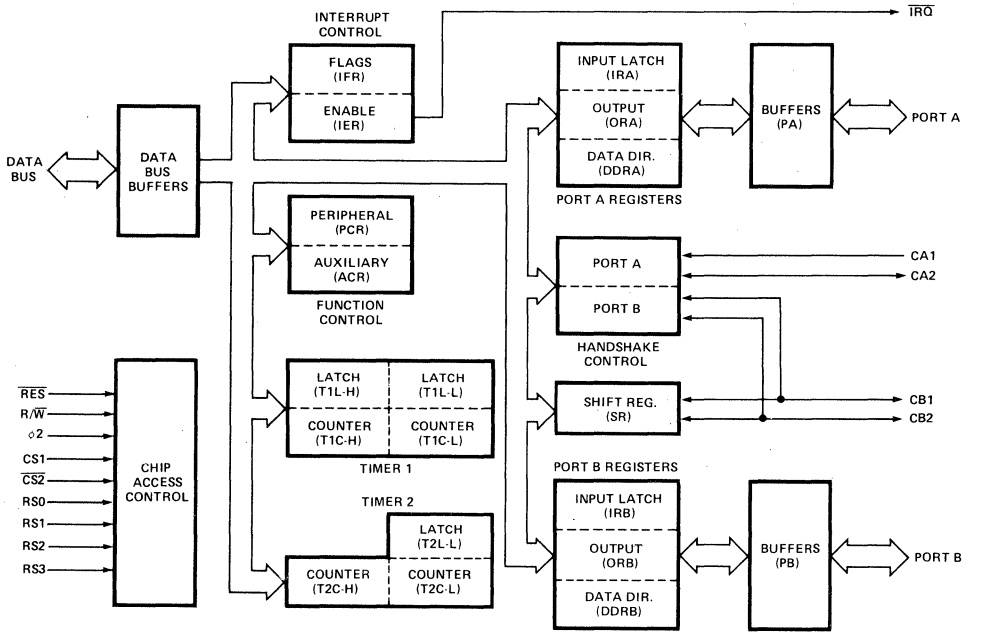


Figure 1. SY6522 Block Diagram

ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	V
Input Voltage	V_{IN}	-0.3 to +7.0	V
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	-55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages. However, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0V ± 5%, T_A = 0-70°C unless otherwise noted)

Symbol	Characteristic	Min.	Max.	Unit
V_{IH}	Input High Voltage (all except $\phi 2$)	2.4	V_{CC}	V
V_{CH}	Clock High Voltage	2.4	V_{CC}	V
V_{IL}	Input Low Voltage	-0.3	0.4	V
I_{IN}	Input Leakage Current — V_{IN} = 0 to 5 Vdc R/$\overline{W}$, $\overline{RES}$, RS0, RS1, RS2, RS3, CS1, $\overline{CS2}$, CA1, $\Phi 2$	—	±2.5	μA
I_{TSI}	Off-state Input Current — V_{IN} = .4 to 2.4V V_{CC} = Max, D0 to D7	—	±10	μA
I_{IH}	Input High Current — V_{IH} = 2.4V PA0-PA7, CA2, PB0-PB7, CB1, CB2	-100	—	μA
I_{IL}	Input Low Current — V_{IL} = 0.4 Vdc PA0-PA7, CA2, PB0-PB7, CB1, CB2	—	-1.6	mA
V_{OH}	Output High Voltage V_{CC} = min, I_{load} = -100 μAdc PA0-PA7, CA2, PB0-PB7, CB1, CB2	2.4	—	V
V_{OL}	Output Low Voltage V_{CC} = min, I_{load} = 1.6 mAdc	—	0.4	V
I_{OH}	Output High Current (Sourcing) V_{OH} = 2.4V V_{OH} = 1.5V (PB0-PB7)	 -100 -1.0	 — —	 μA mA
I_{OL}	Output Low Current (Sinking) V_{OL} = 0.4 Vdc	1.6	—	mA
I_{OFF}	Output Leakage Current (Off state) $\overline{IRQ}$	—	10	μA
C_{IN}	Input Capacitance — T_A = 25°C, f = 1 MHz (R/$\overline{W}$, $\overline{RES}$, RS0, RS1, RS2, RS3, CS1, $\overline{CS2}$, D0-D7, PA0-PA7, CA1, CA2, PB0-PB7) (CB1, CB2) ($\Phi 2$ Input)	 — — —	 7.0 10 20	 pF pF pF
C_{OUT}	Output Capacitance — T_A = 25°C, f = 1 MHz	—	10	pF
P_D	Power Dissipation	—	700	mW

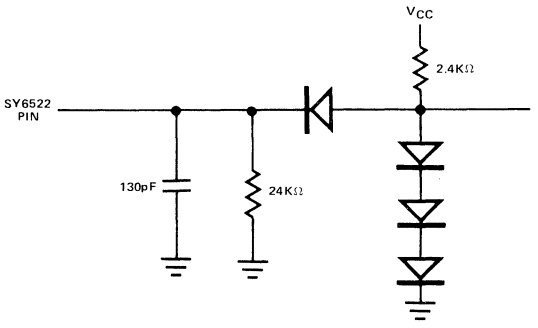


Figure 2. Test Load (for all Dynamic Parameters)

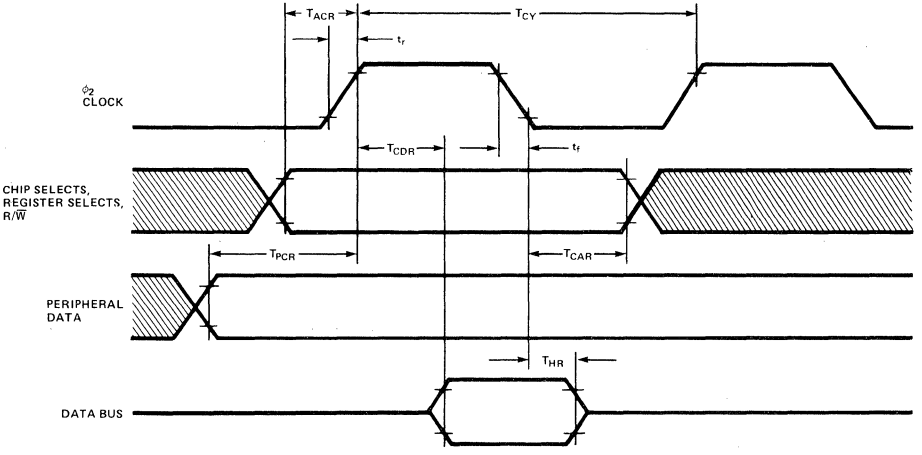


Figure 3. Read Timing Characteristics

READ TIMING CHARACTERISTICS (FIGURE 3)

Symbol	Parameter	SY6522		SY6522A		Unit
		Min.	Max.	Min.	Max.	
T_{CY}	Cycle Time	1	50	0.5	50	μs
T_{ACR}	Address Set-Up Time	180	–	90	–	ns
T_{CAR}	Address Hold Time	0	–	0	–	ns
T_{PCR}	Peripheral Data Set-Up Time	300	–	300	–	ns
T_{CDR}	Data Bus Delay Time	–	395	–	200	ns
T_{HR}	Data Bus Hold Time	10	–	10	–	ns

NOTE: tr, tf = 10 to 30ns.

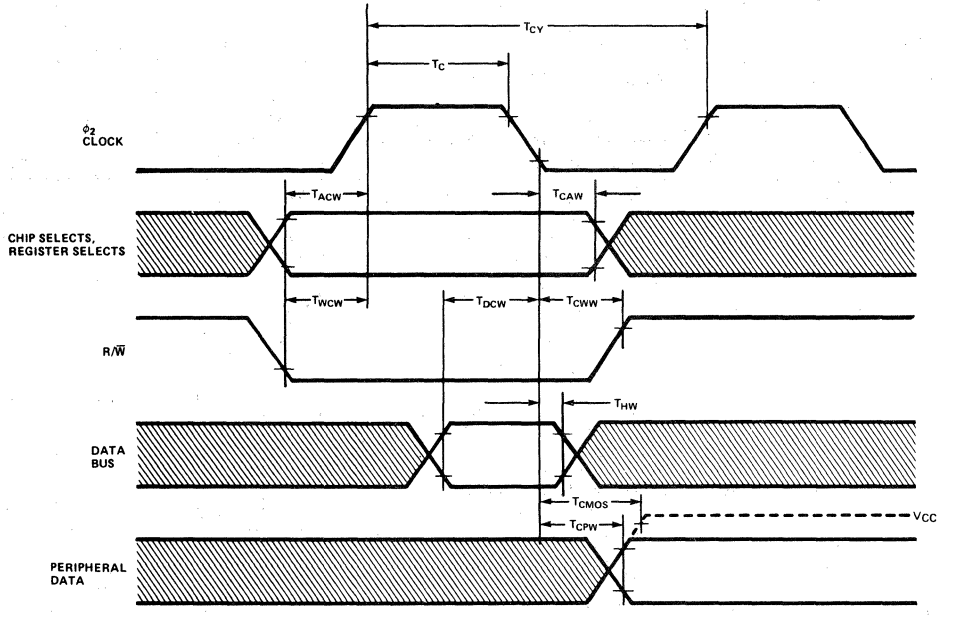


Figure 4. Write Timing Characteristics

WRITE TIMING CHARACTERISTICS (FIGURE 4)

Symbol	Parameter	SY6522		SY6522A		Unit
		Min.	Max.	Min.	Max.	
T_{CY}	Cycle Time	1	50	0.50	50	μs
T_C	$\phi 2$ Pulse Width	0.47	25	0.25	25	μs
T_{ACW}	Address Set-Up Time	180	–	90	–	ns
T_{CAW}	Address Hold Time	0	–	0	–	ns
T_{WCW}	R/$\overline{W}$ Set-Up Time	180	–	90	–	ns
T_{CWW}	R/$\overline{W}$ Hold Time	0	–	0	–	ns
T_{DCW}	Data Bus Set-Up Time	300	–	150	–	ns
T_{HW}	Data Bus Hold Time	10	–	10	–	ns
T_{CPW}	Peripheral Data Delay Time	–	1.0	–	1.0	μs
T_{CMOS}	Peripheral Data Delay Time to CMOS Levels	–	2.0	–	2.0	μs

NOTE: tr, tf = 10 to 30ns.

PERIPHERAL INTERFACE CHARACTERISTICS

Symbol	Characteristic	Min.	Max.	Unit	Figure
t_r, t_f	Rise and Fall Time for CA1, CB1, CA2, and CB2 Input Signals	–	1.0	μs	–
T_{CA2}	Delay Time, Clock Negative Transition to CA2 Negative Transition (read handshake or pulse mode)	–	1.0	μs	5a, 5b
T_{RS1}	Delay Time, Clock Negative Transition to CA2 Positive Transition (pulse mode)	–	1.0	μs	5a
T_{RS2}	Delay Time, CA1 Active Transition to CA2 Positive Transition (handshake mode)	–	2.0	μs	5b
T_{WHS}	Delay Time, Clock Positive Transition to CA2 or CB2 Negative Transition (write handshake)	–	1.0	μs	5c, 5d
T_{DS}	Delay Time, Peripheral Data Valid to CB2 Negative Transition	0	1.5	μs	5c, 5d
T_{RS3}	Delay Time, Clock Positive Transition to CA2 or CB2 Positive Transition (pulse mode)	–	1.0	μs	5c
T_{RS4}	Delay Time, CA1 or CB1 Active Transition to CA2 or CB2 Positive Transition (handshake mode)	–	2.0	μs	5d
T_{IL}	Set-up Time, Peripheral Data Valid to CA1 or CB1 Active Transition (input latching)	300	–	ns	5e
T_{SR1}	Shift-Out Delay Time – Time from ϕ_2 Falling Edge to CB2 Data Out	–	300	ns	5f
T_{SR2}	Shift-In Setup Time – Time from CB2 Data In to ϕ_2 Rising Edge	300	–	ns	5g
T_{IPW}	Pulse Width – PB6 Input Pulse	2	–	μs	5i
T_{ICW}	Pulse Width – CB1 Input Clock	2	–	μs	5h
I_{IPS}	Pulse Spacing – PB6 Input Pulse	2	–	μs	5i
I_{ICS}	Pulse Spacing – CB1 Input Pulse	2	–	μs	5h

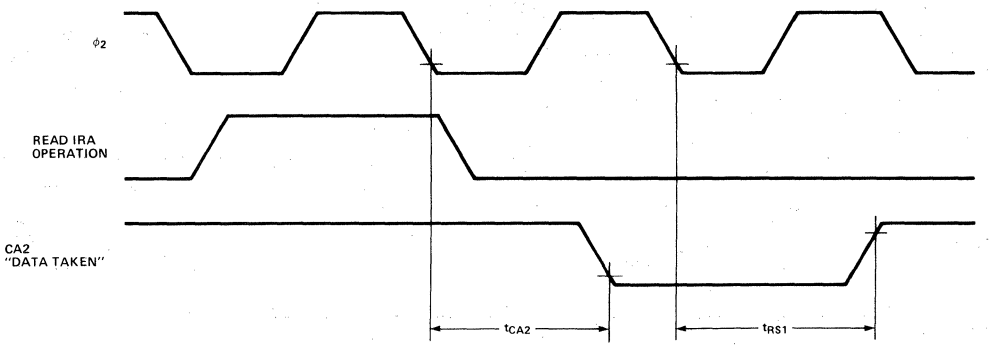


Figure 5a. CA2 Timing for Read Handshake, Pulse Mode

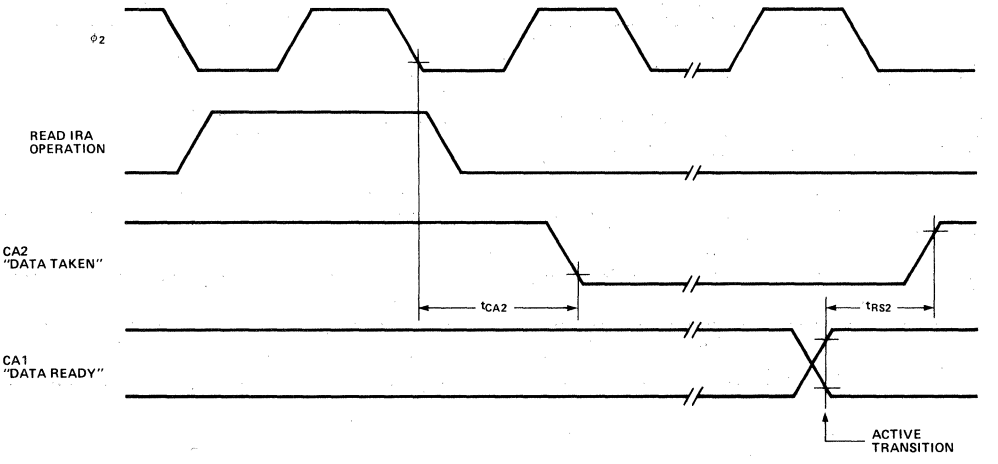


Figure 5b. CA2 Timing for Read Handshake, Handshake Mode

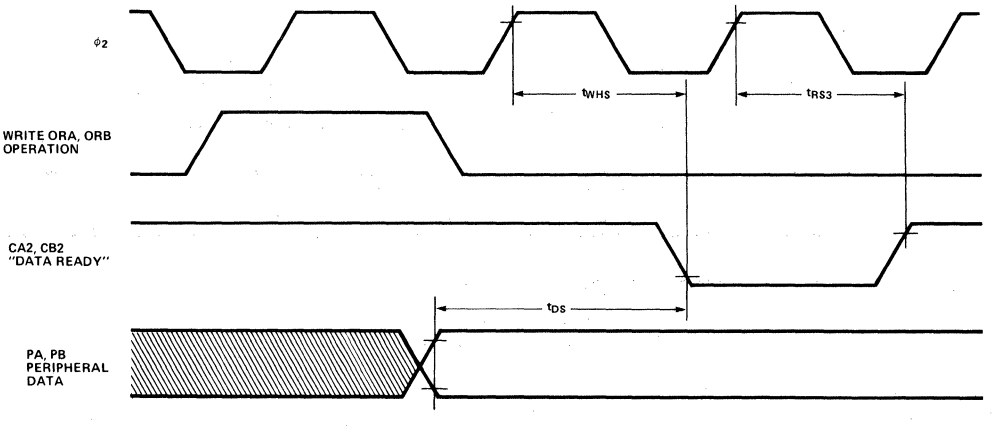


Figure 5c. CA2, CB2 Timing for Write Handshake, Pulse Mode

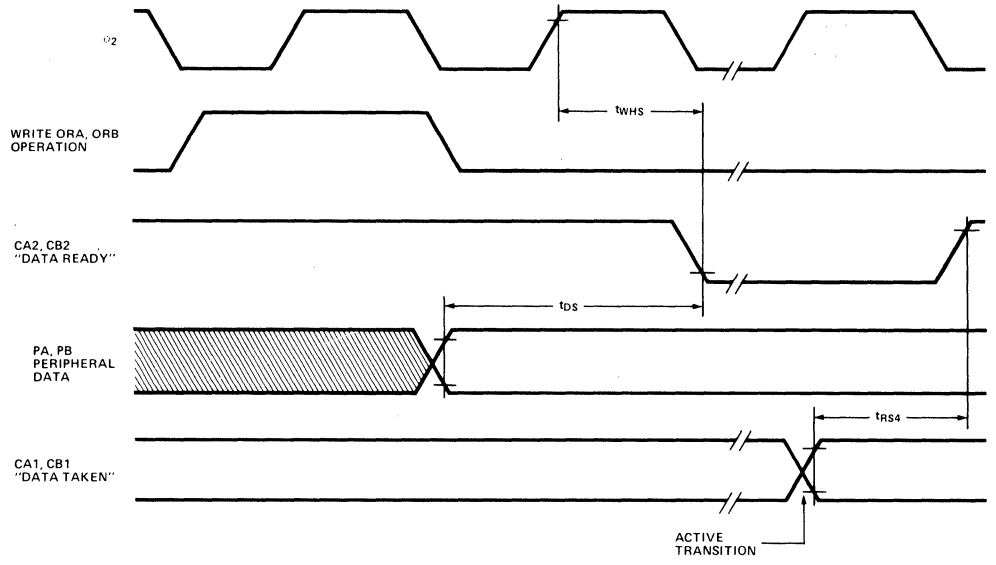


Figure 5d. CA2, CB2 Timing for Write Handshake, Handshake Mode

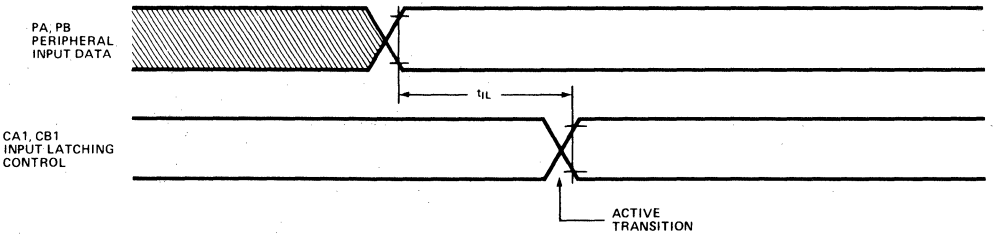


Figure 5e. Peripheral Data Input Latching Timing

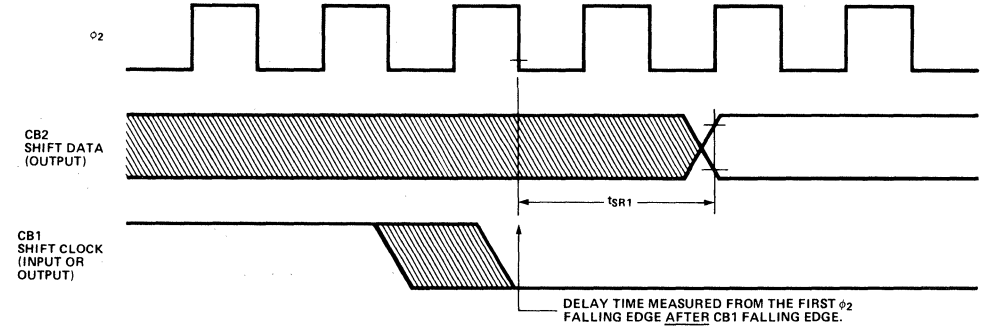


Figure 5f. Timing for Shift Out with Internal or External Shift Clocking

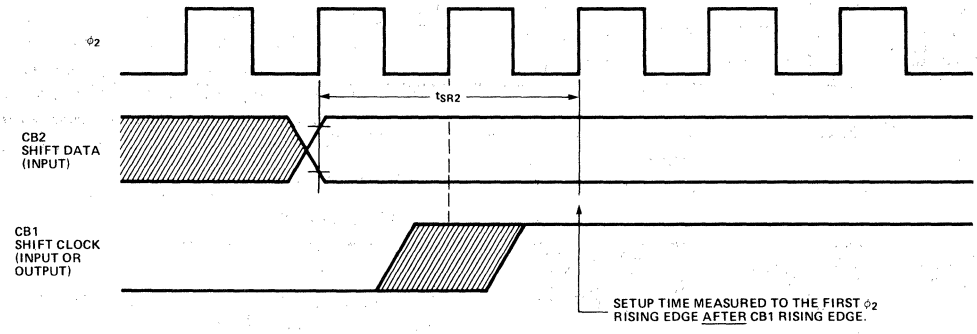


Figure 5g. Timing for Shift In with Internal or External Shift Clocking

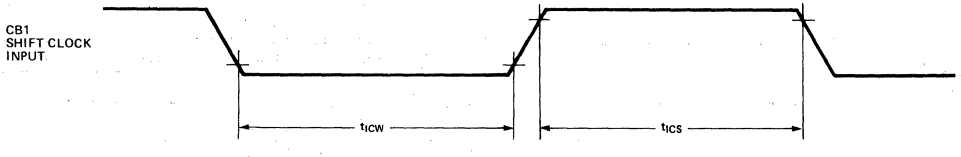


Figure 5h. External Shift Clock Timing

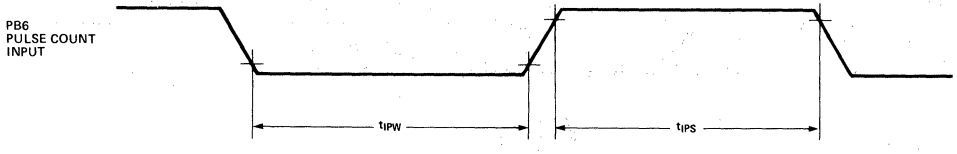


Figure 5i. Pulse Count Input Timing

PIN DESCRIPTIONS

$\overline{RES}$ (Reset)

The reset input clears all internal registers to logic 0 (except T1 and T2 latches and counters and the Shift Register). This places all peripheral interface lines in the input state, disables the timers, shift register, etc. and disables interrupting from the chip.

$\phi 2$ (Input Clock)

The input clock is the system $\phi 2$ clock and is used to trigger all data transfers between the system processor and the SY6522.

R/$\overline{W}$ (Read/Write)

The direction of the data transfers between the SY6522 and the system processor is controlled by the R/$\overline{W}$ line. If R/W is low, data will be transferred out of the processor into the selected SY6522 register (write operation). If R/W is high and the chip is selected, data will be transferred out of the SY6522 (read operation).

DB0–DB7 (Data Bus)

The eight bi-directional data bus lines are used to transfer data between the SY6522 and the system processor. During read cycles, the contents of the selected SY6522 register are placed on the data bus lines and transferred into the processor. During write cycles, these lines are high-impedance inputs and data is transferred from the processor into the selected register. When the SY6522 is unselected, the data bus lines are high-impedance.

CS1, $\overline{CS2}$ (Chip Selects)

The two chip select inputs are normally connected to processor address lines either directly or through decoding. The selected SY6522 register will be accessed when CS1 is high and $\overline{CS2}$ is low.

RS0–RS3 (Register Selects)

The four Register Select inputs permit the system processor to select one of the 16 internal registers of the SY6522, as shown in Figure 6.

Register Number	RS Coding				Register Desig.	Description	
	RS3	RS2	RS1	RS0		Write	Read
0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
2	0	0	1	0	DDRB	Data Direction Register "B"	
3	0	0	1	1	DDRA	Data Direction Register "A"	
4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
5	0	1	0	1	T1C-H	T1 High-Order Counter	
6	0	1	1	0	T1L-L	T1 Low-Order Latches	
7	0	1	1	1	T1L-H	T1 High-Order Latches	
8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
9	1	0	0	1	T2C-H	T2 High-Order Counter	
10	1	0	1	0	SR	Shift Register	
11	1	0	1	1	ACR	Auxiliary Control Register	
12	1	1	0	0	PCR	Peripheral Control Register	
13	1	1	0	1	IFR	Interrupt Flag Register	
14	1	1	1	0	IER	Interrupt Enable Register	
15	1	1	1	1	ORA/IRA	Same as Reg 1 Except No "Handshake"	

Figure 6. SY6522 Internal Register Summary

IRQ (Interrupt Request)

The Interrupt Request output goes low whenever an internal interrupt flag is set and the corresponding interrupt enable bit is a logic 1. This output is "open-drain" to allow the interrupt request signal to be "wire-or'ed" with other equivalent signals in the system.

PA0-PA7 (Peripheral A Port)

The Peripheral A port consists of 8 lines which can be individually programmed to act as inputs or outputs under control of a Data Direction Register. The polarity of output pins is controlled by an Output Register and input data may be latched into an internal register under control of the CA1 line. All of these modes of operation are controlled by the system processor through the internal control registers. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. Figure 7 illustrates the output circuit.

CA1, CA2 (Peripheral A Control Lines)

The two Peripheral A control lines act as interrupt inputs or as handshake outputs. Each line controls an internal interrupt flag with a corresponding interrupt enable bit. In addition, CA1 controls the latching of data on Peripheral A port input lines. CA1 is a high-impedance input only while CA2 represents one standard TTL load in the input mode. CA2 will drive one standard TTL load in the output mode.

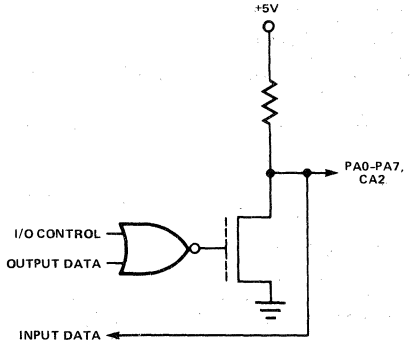


Figure 7. Peripheral A Port Output Circuit

PB0-PB7 (Peripheral B Port)

The Peripheral B port consists of eight bi-directional lines which are controlled by an output register and a data direction register in much the same manner as the PA port. In addition, the polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin. Peripheral B lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. In addition, they are capable of sourcing 1.0mA at 1.5VDC in the output mode to allow the outputs to directly drive Darlington transistor circuits. Figure 8 is the circuit schematic.

CB1, CB2 (Peripheral B Control Lines)

The Peripheral B control lines act as interrupt inputs or as handshake outputs. As with CA1 and CA2, each line controls an interrupt flag with a corresponding interrupt enable bit. In addition, these lines act as a serial port under control of the Shift Register. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. Unlike PB0-PB7, CB1 and CB2 cannot drive Darlington transistor circuits.

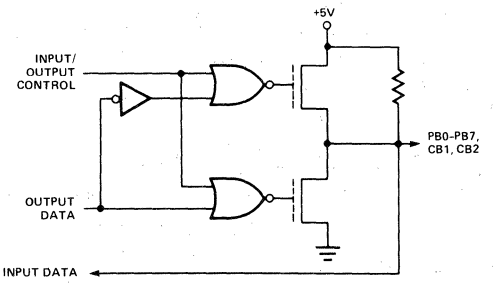


Figure 8. Peripheral B Port Output Circuit

FUNCTIONAL DESCRIPTION

Port A and Port B Operation

Each 8-bit peripheral port has a Data Direction Register (DDRA, DDRB) for specifying whether the peripheral pins are to act as inputs or outputs. A 0 in a bit of the Data Direction Register causes the corresponding peripheral pin to act as an input. A 1 causes the pin to act as an output.

Each peripheral pin is also controlled by a bit in the Output Register (ORA, ORB) and an Input Register (IRA, IRB). When the pin is programmed as an output, the voltage on the pin is controlled by the cor-

responding bit of the Output Register. A 1 in the Output Register causes the output to go high, and a "0" causes the output to go low. Data may be written into Output Register bits corresponding to pins which are programmed as inputs. In this case, however, the output signal is unaffected.

Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the levels on the PA pins. With input latching enabled, IRA will reflect the levels on the PA pins at the time the latching occurred (via CA1).

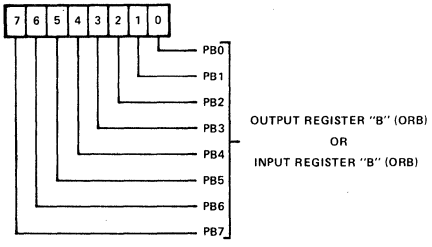
The IRB register operates similar to the IRA register. However, for pins programmed as outputs there is a difference. When reading IRA, the level on the pin determines whether a 0 or a 1 is sensed. When reading IRB, however, the bit stored in the output register, ORB, is the bit sensed. Thus, for outputs which have large loading effects and which pull an output "1" down or which pull an output "0" up, reading IRA may result in reading a "0" when a "1" was actually programmed, and reading a "1" when a "0" was programmed. Reading IRB, on the other hand, will read the "1" or "0" level actually programmed, no matter what the loading on the pin.

Figures 9, 10, and 11 illustrate the formats of the port registers. In addition, the input latching modes are selected by the Auxiliary Control Register (Figure 16.)

Handshake Control of Data Transfers

The SY6522 allows positive control of data transfers between the system processor and peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

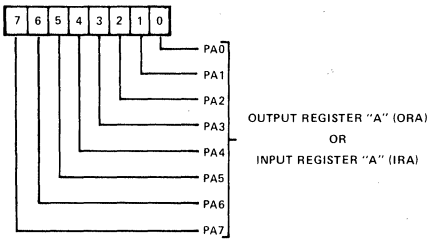
REG 0 – ORB/IRB



Pin Data Direction Selection	WRITE	READ
DDRB = "1" (OUTPUT)	MPU writes Output Level (ORB)	MPU reads output register bit in ORB. Pin level has no affect.
DDRB = "0" (INPUT) (Input latching disabled)	MPU writes into ORB, but no effect on pin level, until DDRB changed.	MPU reads input level on PB pin.
DDRB = "0" (INPUT) (Input latching enabled)		MPU reads IRB bit, which is the level of the PB pin at the time of the last CB1 active transition.

Figure 9. Output Register B (ORB), Input Register B (IRB)

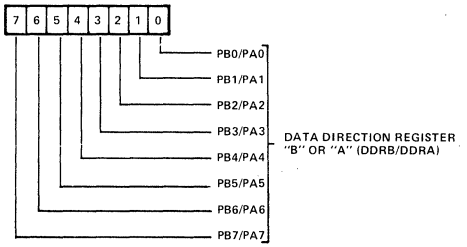
REG 1 – ORA/IRA



Pin Data Direction Selection	WRITE	READ
DDRA = "1" (OUTPUT) (Input latching disabled)	MPU writes Output Level (ORA).	MPU reads level on PA pin.
DDRA = "1" (OUTPUT) (Input latching enabled)		MPU reads IRA bit which is the level of the PA pin at the time of the last CA1 active transition.
DDRA = "0" (INPUT) (Input latching disabled)	MPU writes into ORA, but no effect on pin level, until DDRA changed.	MPU reads level on PA pin.
DDRA = "0" (INPUT) (Input latching enabled)		MPU reads IRA bit which is the level of the PA pin at the time of the last CA1 active transition.

Figure 10. Output Register A (ORA), Input Register A (IRA)

REG 2 (DDRB) AND REG 3 (DDRA)



"0" ASSOCIATED PB/PA PIN IS AN INPUT (HIGH-IMPEDANCE)

"1" ASSOCIATED PB/PA PIN IS AN OUTPUT, WHOSE LEVEL IS DETERMINED BY ORB/ORA REGISTER BIT.

Figure 11. Data Direction Registers (DDRB, DDRA)

Read Handshake

Positive control of data transfers from peripheral devices into the system processor can be accomplished very effectively using Read Handshaking. In this case, the peripheral device must generate the equivalent of a "Data Ready" signal to the processor signifying that valid data is present on the peripheral port. This signal normally interrupts the processor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

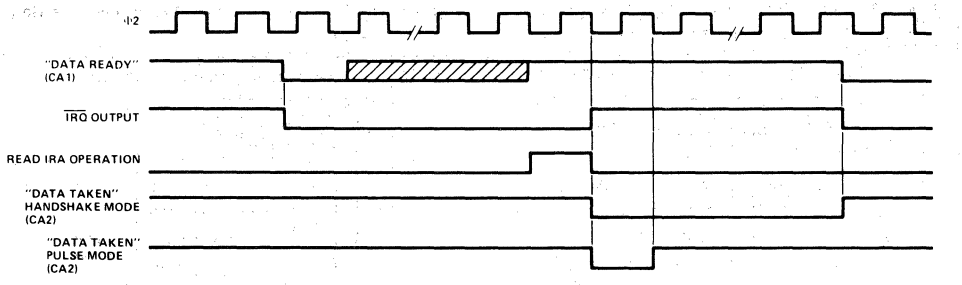


Figure 12. Read Handshake Timing (Port A, Only)

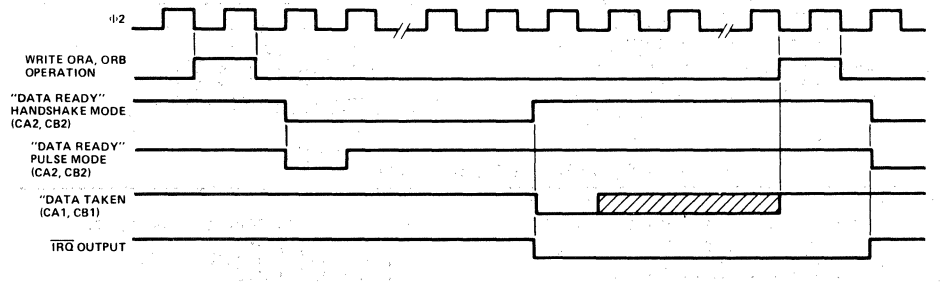


Figure 13. Write Handshake Timing

In the SY6522, automatic "Read" Handshaking is possible on the Peripheral A port only. The CA1 interrupt input pin accepts the "Data Ready" signal and CA2 generates the "Data Taken" signal. The "Data Ready" signal will set an internal flag which may interrupt the processor or which may be polled under program control. The "Data Taken" signal can either be a pulse or a level which is set low by the system processor and is cleared by the "Data Ready" signal. These options are shown in Figure 12 which illustrates the normal Read Handshaking sequence.

Write Handshake

The sequence of operations which allows handshaking data from the system processor to a peripheral device is very similar to that described for Read Handshaking. However, for Write Handshaking, the SY6522 generates the "Data Ready" signal and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the SY6522. CA2 or CB2 act as a "Data Ready" output in either the handshake mode or pulse mode and CA1 or CB1 accept the "Data Taken" signal from the peripheral device, setting the interrupt flag and cleaning the "Data Ready" output. This sequence is shown in Figure 13.

Selection of operating modes for CA1, CA2, CB1, and CB2 is accomplished by the Peripheral Control Register (Figure 14).

Timer Operation

Interval Timer T1 consists of two 8-bit latches and a 16-bit counter. The latches are used to store data which is to be loaded into the counter. After loading, the counter decrements at $\phi 2$ clock rate. Upon reaching zero, an interrupt flag will be set, and $\overline{IRQ}$ will go low if the interrupt is enabled. The timer will then disable any further interrupts, or will automatically transfer the contents of the latches into the counter and will continue to decrement. In addition, the timer may be programmed to invert the output signal on a peripheral pin each time it "times-out". Each of these modes is discussed separately below.

The T1 counter is depicted in Figure 15 and the latches in Figure 16.

REG 12 – PERIPHERAL CONTROL REGISTER

7	6	5	CB2 CONTROL
0	0	0	INPUT-NEGATIVE ACTIVE EDGE
0	0	1	INDEPENDENT INTERRUPT INPUT-NEG EDGE
0	1	0	INPUT-POSITIVE ACTIVE EDGE
0	1	1	INDEPENDENT INTERRUPT INPUT-POS EDGE
1	0	0	HANDSHAKE OUTPUT
1	0	1	PULSE OUTPUT
1	1	0	LOW OUTPUT
1	1	1	HIGH OUTPUT

CB1 INTERRUPT CONTROL
0 = NEGATIVE ACTIVE EDGE
1 = POSITIVE ACTIVE EDGE

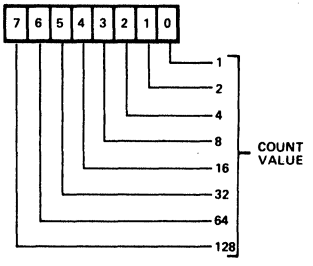
CA1 INTERRUPT CONTROL
0 = NEGATIVE ACTIVE EDGE
1 = POSITIVE ACTIVE EDGE

3	2	1	CA2 CONTROL
0	0	0	INPUT-NEGATIVE ACTIVE EDGE
0	0	1	INDEPENDENT INTERRUPT INPUT-NEG EDGE
0	1	0	INPUT-POSITIVE ACTIVE EDGE
0	1	1	INDEPENDENT INTERRUPT INPUT-POS EDGE
1	0	0	HANDSHAKE OUTPUT
1	0	1	PULSE OUTPUT
1	1	0	LOW OUTPUT
1	1	1	HIGH OUTPUT

Figure 14. CA1, CA2, CB1, CB2 Control

Two bits are provided in the Auxiliary Control Register (bits 6 and 7) to allow selection of the T1 operating modes. The four possible modes are depicted in Figure 17.

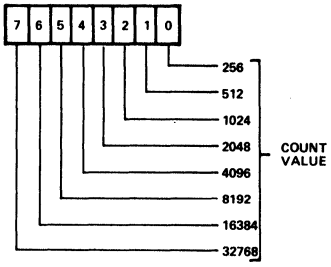
REG 4 – TIMER 1 LOW-ORDER COUNTER



WRITE – 8 BITS LOADED INTO T1 LOW-ORDER LATCHES. LATCH CONTENTS ARE TRANSFERRED INTO LOW-ORDER COUNTER AT THE TIME THE HIGH-ORDER COUNTER IS LOADED (REG 5).

READ – 8 BITS FROM T1 LOW-ORDER COUNTER TRANSFERRED TO MPU. IN ADDITION, T1 INTERRUPT FLAG IS RESET (BIT 6 IN INTERRUPT FLAG REGISTER).

REG 5 – TIMER 1 HIGH-ORDER COUNTER

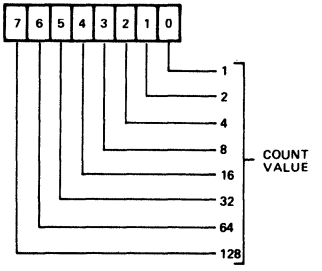


WRITE – 8 BITS LOADED INTO T1 HIGH-ORDER LATCHES. ALSO, AT THIS TIME BOTH HIGH AND LOW-ORDER LATCHES TRANSFERRED INTO T1 COUNTER. T1 INTERRUPT FLAG ALSO IS RESET.

READ – 8 BITS FROM T1 HIGH-ORDER COUNTER TRANSFERRED TO MPU.

Figure 15. T1 Counter Registers

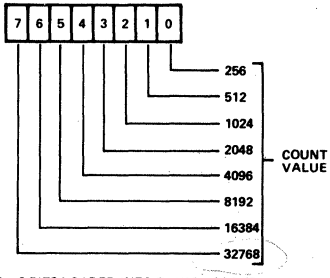
REG 6 – TIMER 1 LOW-ORDER LATCHES



WRITE – 8 BITS LOADED INTO T1 LOW-ORDER LATCHES. THIS OPERATION IS NO DIFFERENT THAT A WRITE INTO REG 4.

READ – 8 BITS FROM T1 LOW-ORDER LATCHES TRANSFERRED TO MPU. UNLIKE REG 4 OPERATION, THIS DOES NOT CAUSE RESET OF T1 INTERRUPT FLAG.

REG 7 – TIMER 1 HIGH-ORDER LATCHES

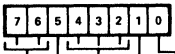


WRITE – 8 BITS LOADED INTO T1 HIGH-ORDER LATCHES. UNLIKE REG 4 OPERATION NO LATCH-TO-COUNTER TRANSFERS TAKE PLACE.

READ – 8 BITS FROM T1 HIGH-ORDER LATCHES TRANSFERRED TO MPU.

Figure 16. T1 Latch Registers

REG 11 – AUXILIARY CONTROL REGISTER



T1 TIMER CONTROL

7	6	OPERATION	PB7
0	0	TIMED INTERRUPT EACH TIME T1 IS LOADED	DISABLED
0	1	CONTINUOUS INTERRUPTS	
1	0	TIMED INTERRUPT EACH TIME T1 IS LOADED	ONE-SHOT OUTPUT
1	1	CONTINUOUS INTERRUPTS	SQUARE WAVE OUTPUT

T2 TIMER CONTROL

5	OPERATION
0	TIMED INTERRUPT
1	COUNT DOWN WITH PULSES ON PB6

PA

PB

LATCH ENABLE/DISABLE

0 = DISABLE
1 = ENABLE LATCHING

SHIFT REGISTER CONTROL

4	3	2	OPERATION
0	0	0	DISABLED
0	0	1	SHIFT IN UNDER CONTROL OF T2
0	1	0	SHIFT IN UNDER CONTROL OF 02
0	1	1	SHIFT IN UNDER CONTROL OF EXT. CLK
1	0	0	SHIFT OUT FREE-RUNNING AT T2 RATE
1	0	1	SHIFT OUT UNDER CONTROL OF T2
1	1	0	SHIFT OUT UNDER CONTROL OF 02
1	1	1	SHIFT OUT UNDER CONTROL OF EXT. CLK.

Figure 17. Auxiliary Control Register

Note: The processor does not write directly into the low order counter (T1C-L). Instead, this half of the counter is loaded automatically from the low order latch when the processor writes into the high order counter. In fact, it may not be necessary to write to the low order counter in some applications since the timing operation is triggered by writing to the high order counter.

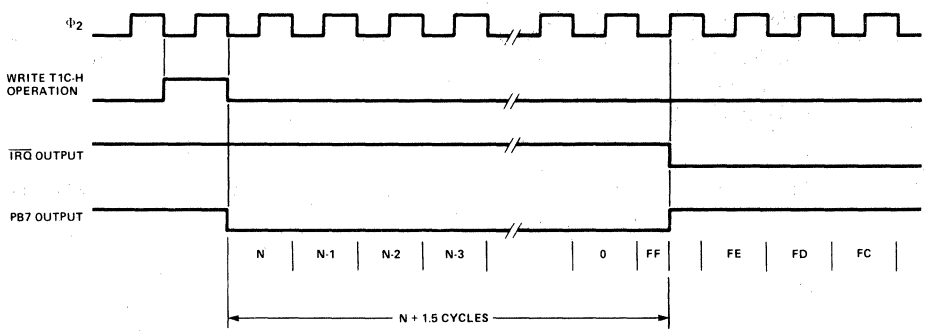


Figure 18. Timer 1 and Timer 2 One-Shot Mode Timing

Timer 1 One-Shot Mode

The interval timer one-shot mode allows generation of a single interrupt for each timer load operation. As with any interval timer, the delay between the "write T1C-H" operation and generation of the processor interrupt is a direct function of the data loaded into the timing counter. In addition to generating a single interrupt, Timer 1 can be programmed to produce a single negative pulse on the PB7 peripheral pin. With the output enabled (ACR7=1) a "write T1C-H" operation will cause PB7 to go low. PB7 will return high when Timer 1 times out. The result is a single programmable width pulse.

In the one-shot mode, writing into the high order latch has no effect on the operation of Timer 1. However, it will be necessary to assure that the low order latch contains the proper data before initiating the count-down with a "write T1C-H" operation. When the processor writes into the high order counter, the T1 interrupt flag will be cleared, the contents of the low order latch will be transferred into the low order counter, and the timer will begin to decrement at system clock rate. If the PB7 output is enabled, this signal will go low on the phase two following the write operation. When the counter reaches zero, the T1 interrupt flag will be set, the $\overline{IRQ}$ pin will go low (interrupt enabled), and the signal on PB7 will go high. At this time the counter will continue to decrement at system clock rate. This allows the system processor to read the contents of the counter to determine the time since interrupt. However, the T1 interrupt flag cannot be set again unless it has been cleared as described in this specification.

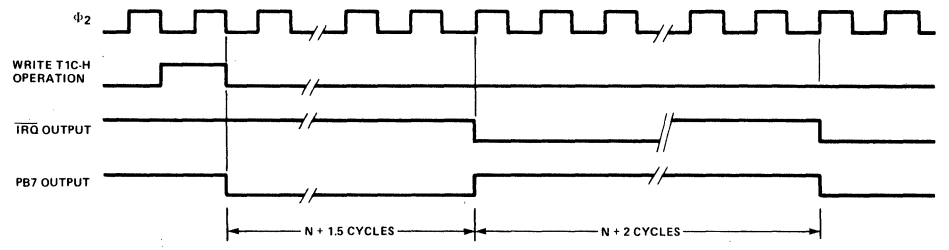
Timing for the SY6522 interval timer one-shot modes is shown in Figure 18.

Timer 1 Free-Run Mode

The most important advantage associated with the latches in T1 is the ability to produce a continuous series of evenly spaced interrupts and the ability to produce a square wave on PB7 whose frequency is not affected by variations in the processor interrupt response time. This is accomplished in the "free-running" mode.

In the free-running mode, the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. However, instead of continuing to decrement from zero after a time-out, the timer automatically transfers the contents of the latch into the counter (16 bits) and continues to decrement from there. The interrupt flag can be cleared by writing T1C-H, by reading T1C-L, or by writing directly into the flag as described later. However, it is not necessary to rewrite the timer to enable setting the interrupt flag on the next time-out.

All interval timers in the SY6522 are "re-triggerable". Rewriting the counter will always re-initialize the time-out period. In fact, the time-out can be prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the processor writes into the high order counter (T1C-H). However, by loading the latches only, the processor can access the timer during each down-counting operation without affecting the time-out in process. Instead, the data loaded into the latches will determine the length of the next time-out period. This capability is particularly valuable in the free-running mode with the output enabled. In this mode, the signal on PB7 is inverted and the interrupt flag is set with each time-out. By responding to the interrupts with new data for the latches, the processor can determine the period of the next half cycle during each half cycle of the output signal on PB7. In this manner, very complex waveforms can be generated. Timing for the free-running mode is shown in Figure 19.



Note: A precaution to take in the use of PB7 as the timer output concerns the Data Direction Register contents for PB7. Both DDRB bit 7 and ACR bit 7 must be 1 for PB7 to function as the timer output. If one is 1 and the other is 0, then PB7 functions as a normal output pin, controlled by ORB bit 7.

Figure 19. Timer 1 Free-Run Mode Timing

Timer 2 Operation

Timer 2 operates as an interval timer (in the "one-slot" mode only), or as a counter for counting negative pulses on the PB6 peripheral pin. A single control bit is provided in the Auxiliary Control Register to select between these two modes. This timer is comprised of a "write-only" low-order latch (T2L-L), a "read-only" low-order counter and a read/write high order counter. The counter registers act as a 16-bit counter which decrements at Φ2 rate. Figure 20 illustrates the T2 Counter Registers.

Timer 2 One-Shot Mode

As an interval timer, T2 operates in the "one-shot" mode similar to Timer 1. In this mode, T2 provides a single interrupt for each "write T2C-H" operation. After timing out, the counter will continue to decrement. However, setting of the interrupt flag will be disabled after initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H to enable setting of the interrupt flag. The interrupt flag is cleared by reading T2C-L or by writing T2C-H. Timing for this operation is shown in Figure 18.

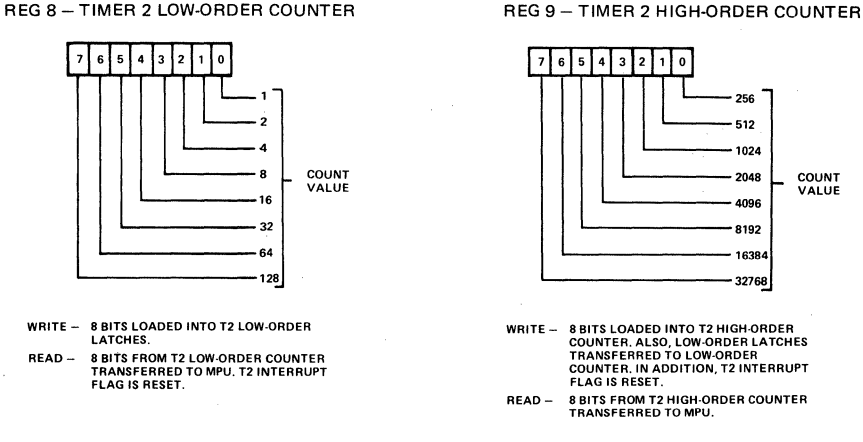


Figure 20. T2 Counter Registers

Timer 2 Pulse Counting Mode

In the pulse counting mode, T2 serves primarily to count a predetermined number of negative-going pulses on PB6. This is accomplished by first loading a number into T2. Writing into T2C-H clears the interrupt flag and allows the counter to decrement each time a pulse is applied to PB6. The interrupt flag will be set when T2 reaches zero. At this time the counter will continue to decrement with each pulse on PB6. However, it is necessary to rewrite T2C-H to allow the interrupt flag to set on subsequent down-counting operations. Timing for this mode is shown in Figure 21. The pulse must be low on the leading edge of $\Phi 2$.

Shift Register Operation

The Shift Register (SR) performs serial data transfers into and out of the CB2 pin under control of an internal modulo-8 counter. Shift pulses can be applied to the CB1 pin from an external source or, with the proper mode selection, shift pulses generated internally will appear on the CB1 pin for controlling external devices.

The control bits which select the various shift register operating modes are located in the Auxiliary Control Register. Figure 22 illustrates the configuration of the SR data bits and the SR control bits of the ACR.

Figures 23 and 24 illustrate the operation of the various shift register modes.

Interrupt Operation

Controlling interrupts within the SY6522 involves three principal operations. These are flagging the interrupts, enabling interrupts and signaling to the processor that an active interrupt exists within the chip. Interrupt flags are set by interrupting conditions which exist within the chip or on inputs to the chip. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the microprocessor must examine these flags in order from highest to lowest priority. This is accomplished by reading the flag register into the processor accumulator, shifting this register either right or left and then using conditional branch instructions to detect an active interrupt.

Associated with each interrupt flag is an interrupt enable bit. This can be set or cleared by the processor to enable interrupting the processor from the corresponding interrupt flag. If an interrupt flag is set to a logic 1 by an interrupting condition, and the corresponding interrupt enable bit is set to a 1, the Interrupt Request Output ($\overline{IRQ}$) will go low. $\overline{IRQ}$ is an "open-collector" output which can be "wire-or'ed" with other devices in the system to interrupt the processor.

In the SY6522, all the interrupt flags are contained in one register. In addition, bit 7 of this register will be read as a logic 1 when an interrupt exists within the chip. This allows very convenient polling of several devices within a system to locate the source of an interrupt.

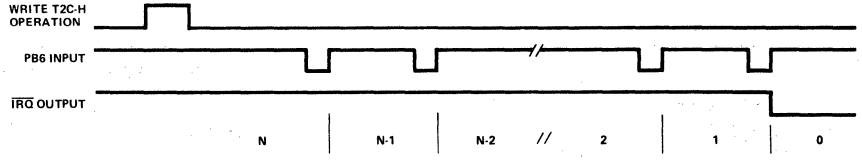
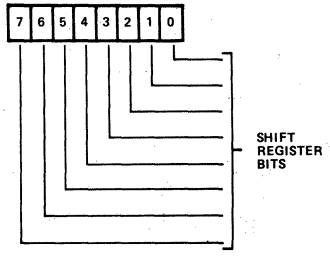


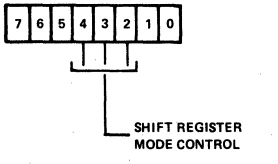
Figure 21. Timer 2 Pulse Counting Mode

REG 10 – SHIFT REGISTER



NOTES:
1. WHEN SHIFTING OUT, BIT 7 IS THE FIRST BIT OUT AND SIMULTANEOUSLY IS ROTATED BACK INTO BIT 0.
2. WHEN SHIFTING IN, BITS INITIALLY ENTER BIT 0 AND ARE SHIFTED TOWARDS BIT 7.

REG 11 – AUXILIARY CONTROL REGISTER



4	3	2	OPERATION
0	0	0	DISABLED
0	0	1	SHIFT IN UNDER CONTROL OF T2
0	1	0	SHIFT IN UNDER CONTROL OF Φ_2
0	1	1	SHIFT IN UNDER CONTROL OF EXT CLK
1	0	0	SHIFT OUT FREE-RUNNING AT T2 RATE
1	0	1	SHIFT OUT UNDER CONTROL OF T2
1	1	0	SHIFT OUT UNDER CONTROL OF Φ_2
1	1	1	SHIFT OUT UNDER CONTROL OF EXT CLK

Figure 22. SR and ACR Control Bits

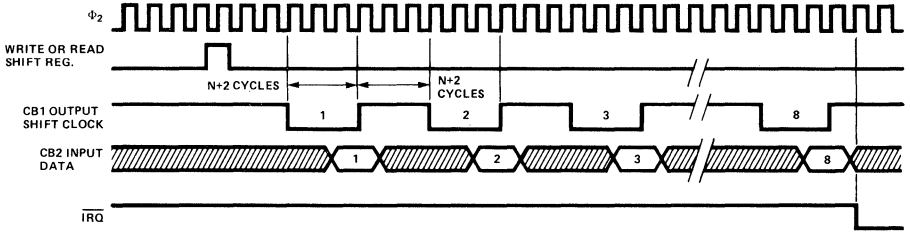
SR Disabled (000)

The 000 mode is used to disable the Shift Register. In this mode the microprocessor can write or read the SR, but the shifting operation is disabled and operation of CB1 and CB2 is controlled by the appropriate bits in the Peripheral Control Register (PCR). In this mode the SR Interrupt Flag is disabled (held to a logic 0).

Shift in Under Control of T2 (001)

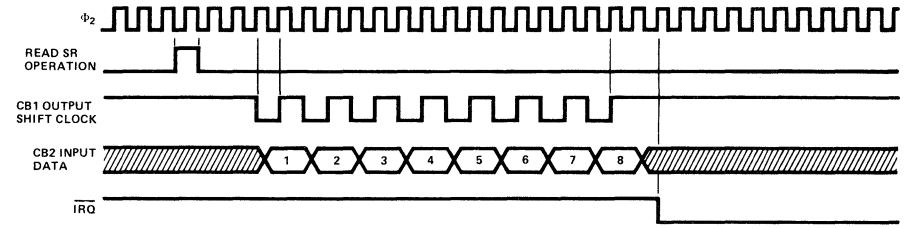
In the 001 mode the shifting rate is controlled by the low order 8 bits of T2. Shift pulses are generated on the CB1 pin to control shifting in external devices. The time between transitions of this output clock is a function of the system clock period and the contents of the low order T2 latch (N).

The shifting operation is triggered by writing or reading the shift register. Data is shifted first into the low order bit of SR and is then shifted into the next higher order bit of the shift register on the negative-going edge of each clock pulse. The input data should change before the positive-going edge of the CB1 clock pulse. This data is shifted into the shift register during the ϕ_2 clock cycle following the positive-going edge of the CB1 clock pulse. After 8 CB1 clock pulses, the shift register interrupt flag will be set and $\overline{IRQ}$ will go low.



Shift in Under Control of ϕ_2 (010)

In mode 010 the shift rate is a direct function of the system clock frequency. CB1 becomes an output which generates shift pulses for controlling external devices. Timer 2 operates as an independent interval timer and has no effect on SR. The shifting operation is triggered by reading or writing the Shift Register. Data is shifted first into bit 0 and is then shifted into the next higher order bit of the shift register on the trailing edge of each ϕ_2 clock pulse. After 8 clock pulses, the shift register interrupt flag will be set, and the output clock pulses on CB1 will stop.



Shift in Under Control of External CB1 Clock (011)

In mode 011 CB1 becomes an input. This allows an external device to load the shift register at its own pace. The shift register counter will interrupt the processor each time 8 bits have been shifted in. However, the shift register counter does not stop the shifting operation; it acts simply as a pulse counter. Reading or writing the Shift Register resets the Interrupt flag and initializes the SR counter to count another 8 pulses.

Note that the data is shifted during the first system clock cycle following the positive-going edge of the CB1 shift pulse. For this reason, data must be held stable during the first full cycle following CB1 going high.

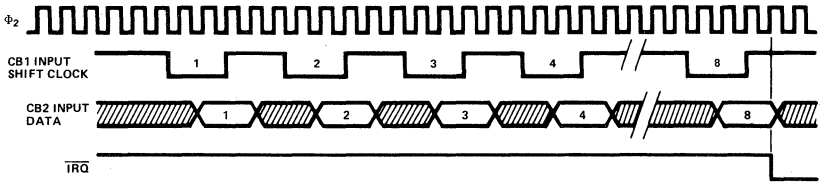
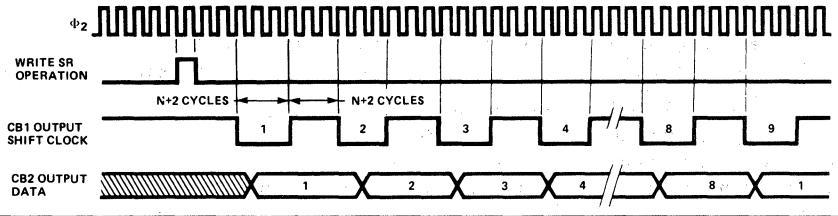


Figure 23. Shift Register Input Modes

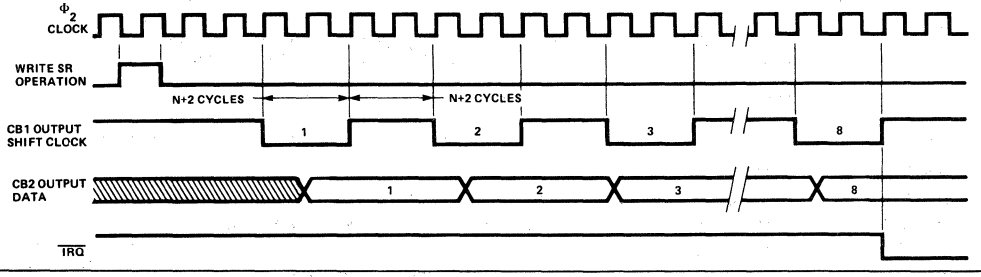
Shift Out Free-Running at T2 Rate (100)

Mode 100 is very similar to mode 101 in which the shifting rate is set by T2. However, in mode 100 the SR Counter does not stop the shifting operation. Since the Shift Register bit 7 (SR7) is recirculated back into bit 0, the 8 bits loaded into the shift register will be clocked onto CB2 repetitively. In this mode the shift register counter is disabled.



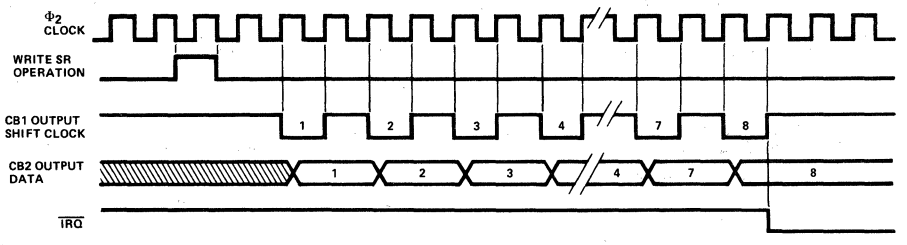
Shift Out Under Control of T2 (101)

In mode 101 the shift rate is controlled by T2 (as in the previous mode). However, with each read or write of the shift register the SR Counter is reset and 8 bits are shifted onto CB2. At the same time, 8 shift pulses are generated on CB1 to control shifting in external devices. After the 8 shift pulses, the shifting is disabled, the SR Interrupt Flag is set and CB2 remains at the last data level.



Shift Out Under Control of ϕ_2 (110)

In mode 110, the shift rate is controlled by the ϕ_2 system clock.



Shift Out Under Control of External CB1 Clock (111)

In mode 111 shifting is controlled by pulses applied to the CB1 pin by an external device. The SR counter sets the SR Interrupt flag each time it counts 8 pulses but it does not disable the shifting function. Each time the microprocessor writes or reads the shift register, the SR Interrupt flag is reset and the SR counter is initialized to begin counting the next 8 shift pulses on pin CB1. After 8 shift pulses, the interrupt flag is set. The microprocessor can then load the shift register with the next byte of data.

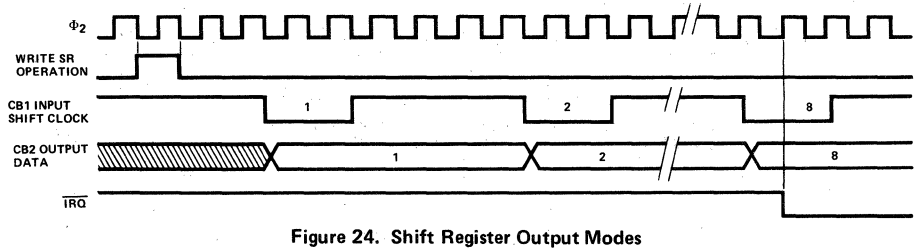


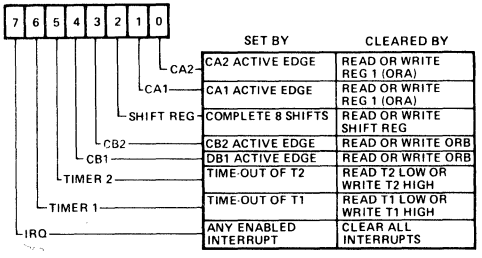
Figure 24. Shift Register Output Modes

The Interrupt Flag Register (IFR) and Interrupt Enable Register (IER) are depicted in Figures 25 and 26, respectively.

The IFR may be read directly by the processor. In addition, individual flag bits may be cleared by writing a "1" into the appropriate bit of the IFR. When the proper chip select and register signals are applied to the chip, the contents of this register are placed on the data bus. Bit 7 indicates the status of the IRQ output. This bit corresponds to the logic function: IRQ = IFR6 x IER6 + IFR5 x IER5 + IFR4 x IER4 + IFR3 x IER3 + IFR2 x IER2 + IFR1 x IER1 + IFR0 x IER0. Note: X = logic AND, + = Logic OR.

The IFR bit 7 is not a flag. Therefore, this bit is not directly cleared by writing a logic 1 into it. It can only be cleared by clearing all the flags in the register or by disabling all the active interrupts as discussed in the next section.

REG 13 – INTERRUPT FLAG REGISTER



	SET BY	CLEARED BY
CA2	CA2 ACTIVE EDGE	READ OR WRITE REG 1 (ORA)
CA1	CA1 ACTIVE EDGE	READ OR WRITE REG 1 (ORA)
SHIFT REG	COMPLETE 8 SHIFTS	READ OR WRITE SHIFT REG
CB2	CB2 ACTIVE EDGE	READ OR WRITE ORB
CB1	DB1 ACTIVE EDGE	READ OR WRITE ORB
TIMER 2	TIME-OUT OF T2	READ T2 LOW OR WRITE T2 HIGH
TIMER 1	TIME-OUT OF T1	READ T1 LOW OR WRITE T1 HIGH
IRQ	ANY ENABLED INTERRUPT	CLEAR ALL INTERRUPTS

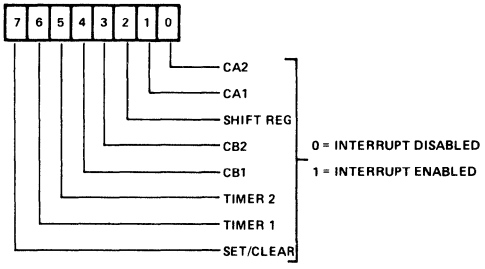
Figure 25. Interrupt Flag Register (IFR)

For each interrupt flag in IFR, there is a corresponding bit in the Interrupt Enable Register. The system processor can set or clear selected bits in this register to facilitate controlling individual interrupts without affecting others. This is accomplished by writing to address 1110 (IER address). If bit 7 of the data placed on the system data bus during this write operation is a 0, each 1 in bits 6 through 0 clears the corresponding bit in the Interrupt Enable Register. For each zero in bits 6 through 0, the corresponding bit is unaffected.

Setting selected bits in the Interrupt Enable Register is accomplished by writing to the same address with bit 7 in the data word set to a logic 1. In this case, each 1 in bits 6 through 0 will set the corresponding bit. For each zero, the corresponding bit will be unaffected. This individual control of the setting and clearing operations allows very convenient control of the interrupts during system operation.

In addition to setting and clearing IER bits, the processor can read the contents of this register by placing the proper address on the register select and chip select inputs with the R/W line high. Bit 7 will be read as a logic 0.

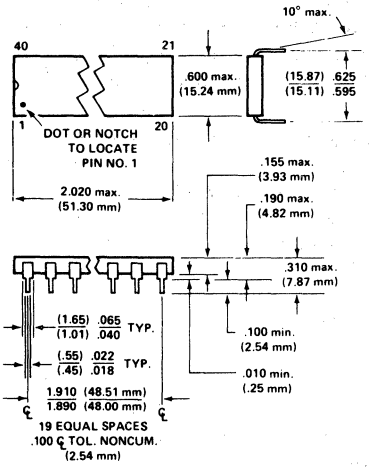
REG 14 – INTERRUPT ENABLE REGISTER



NOTES:
1. IF BIT 7 IS A "0", THEN EACH "1" IN BITS 0 - 6 DISABLES THE CORRESPONDING INTERRUPT.
2. IF BIT 7 IS A "1", THEN EACH "1" IN BITS 0 - 6 ENABLES THE CORRESPONDING INTERRUPT.
3. IF A READ OF THIS REGISTER IS DONE, BIT 7 WILL BE "0" AND ALL OTHER BITS WILL REFLECT THEIR ENABLE/DISABLE STATE.

Figure 26. Interrupt Enable Register (IER)

PACKAGE OUTLINE

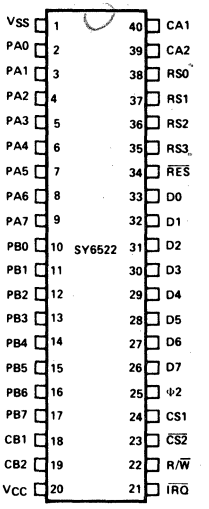


NOTE: Pin No. 1 is in lower left corner when symbolization is in normal orientation

ORDERING INFORMATION

Order Number	Package Type	Frequency Option
SYP 6522	Plastic	1 MHz
SYP 6522A	Plastic	2 MHz
SYC 6522	Ceramic	1 MHz
SYC 6522A	Ceramic	2 MHz

PIN CONFIGURATION



APPENDIX K

SY6532 DATA SHEET

RAM, I/O, Timer Array

SY6532

MICROPROCESSOR PRODUCTS

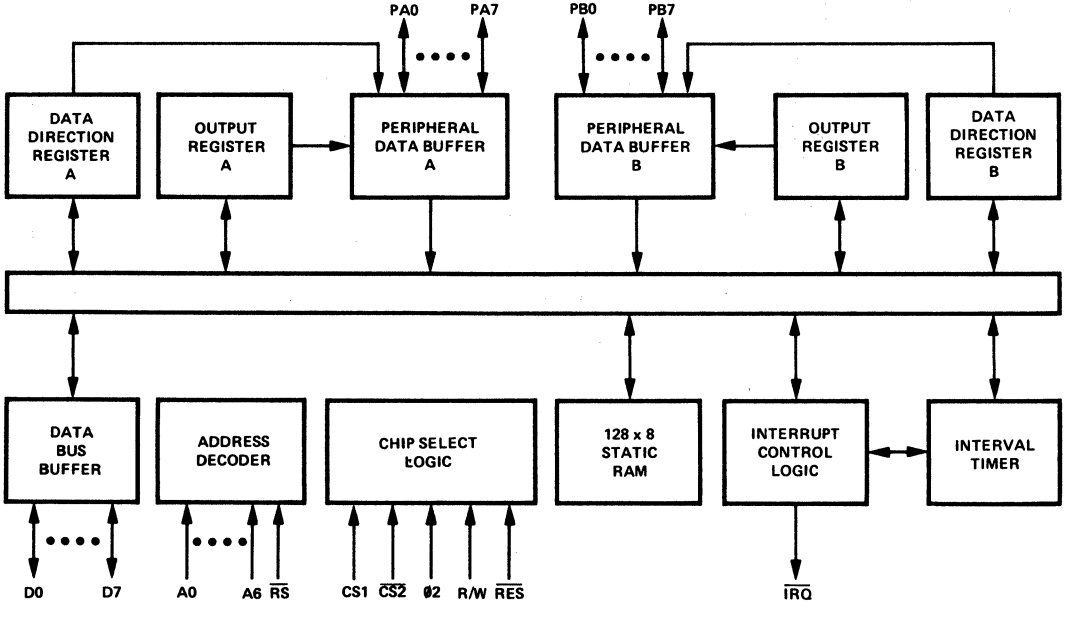
Synertek®

APRIL 1979

The SY6532 is designed to operate in conjuction with the SY6500 Microprocessor Family. It is comprised of a 128 x 8 static RAM, two software controlled 8 bit bi-directional data ports allowing direct interfacing between the microprocessor unit and peripheral devices, a software programmable interval timer with interrupt capable of timing in various intervals from 1 to 262,144 clock periods, and a programmable edge-detect interrupt circuit.

- 8 bit bi-directional Data Bus for direct communication with the microprocessor
- Programmable edge-sensitive interrupt
- 128 x 8 static RAM
- Two 8 bit bi-directional data ports for interface to peripherals
- Two programmable I/O Peripheral Data Direction Registers
- Programmable Interval Timer
- Programmable Interval Timer Interrupt
- TTL & CMOS compatible peripheral lines
- Peripheral pins with Direct Transistor Drive Capability
- High Impedance Three-State Data Pins

Figure 1. 6532 BLOCK DIAGRAM



MAXIMUM RATINGS

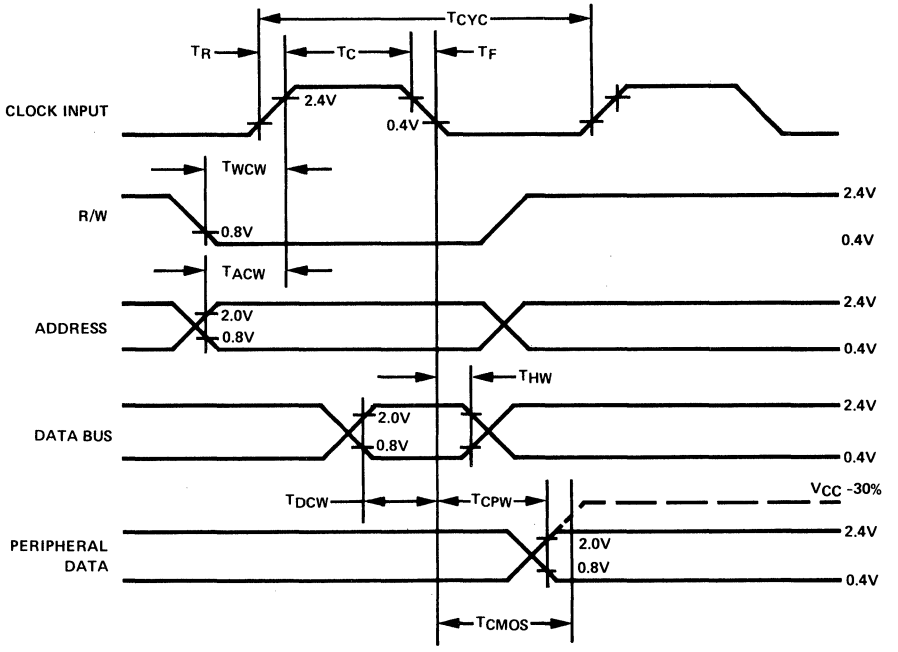
RATING	SYMBOL	VOLTAGE	UNIT
Supply Voltage	V_{CC}	-.3 to +7.0	V
Input/Output Voltage	V_{IN}	-.3 to +7.0	V
Operating Temperature Range	T_{OP}	0 to 70	°C
Storage Temperature Range	T_{STG}	-55 to +150	°C

ELECTRICAL CHARATERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0V, T_A = 25° C)

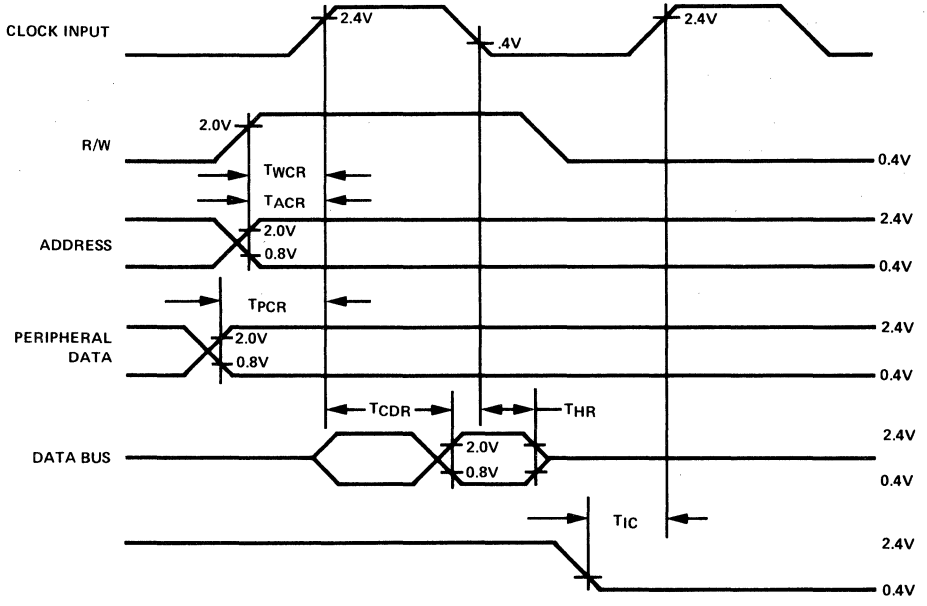
CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
Input High Voltage	V_{IH}	V_{SS} + 2.4		V_{CC}	V
Input Low Voltage	V_{IL}	V_{SS} - .3		V_{SS} + .4	V
Input Leakage Current; V_{IN} = V_{SS} + 5V AØ-A6, $\overline{RS}$, R/W, $\overline{RES}$, Ø2, CS1, $\overline{CS2}$	I_{IN}		1.0	2.5	μA
Input Leakage Current for High Impedance State (Three State); V_{IN} = .4V to 2.4V; DØ-D7	I_{TSI}		±1.0	±10.0	μA
Input High Current; V_{IN} = 2.4V PAØ-PA7, PBØ-PB7	I_{IH}	-100.	-300.		μA
Input Low Current; V_{IN} = .4V PAØ-PA7, PBØ-PB7	I_{IL}		-1.0	-1.6	MA
Output High Voltage V_{CC} = MIN, $I_{LOAD} \leq$ -100μA (PAØ-PA7, PBØ-PB7, DØ-D7) $I_{LOAD} \leq$ 3 MA (PBØ-PB7)	V_{OH}	 V_{SS} + 2.4 V_{SS} + 1.5			V
Output Low Voltage V_{CC} = MIN, $I_{LOAD} \leq$ 1.6MA	V_{OL}			V_{SS} + .4	V
Output High Current (Sourcing); $V_{OH} \geq$ 2.4V (PAØ-PA7, PBØ-PB7, DØ-D7) $\geq$ 1.5V Available for direct transistor drive (PBØ-PB7)	I_{OH}	 -100 3.0	 -1000 5.0		 μA MA
Output Low Current (Sinking); $V_{OL} \leq$ .4V	I_{OL}	1.6			MA
Clock Input Capacitance	C_{Clk}			30	pf
Input Capacitance	C_{IN}			10	pf
Output Capacitance	C_{OUT}			10	pf
Power Dissipation	I_{CC}		100	125	mA

All inputs contain protection circuitry to prevent damage due to high static charges. Care should be exercised to prevent unnecessary application of voltage outside the specification range.

WRITE TIMING CHARACTERISTICS



READ TIMING CHARACTERISTICS



WRITE TIMING CHARACTERISTICS

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
Clock Period	T_{CYC}	1			μS
Rise & Fall Times	TR, TF			25	NS
Clock Pulse Width	TC	470			NS
R/W valid before positive transition of clock	TWCW	180			NS
Address valid before positive transition of clock	TACW	180			NS
Data Bus valid before negative transition of clock	TDCW	300			NS
Data Bus Hold Time	THW	10			NS
Peripheral data valid after negative transition of clock	TCPW			1	μS
Peripheral data valid after negative transition of clock driving CMOS (Level = V_{CC} = 30%)	TCMOS			2	μS

READ TIMING CHARACTERISTICS

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
R/W valid after positive transition of clock	TWCR	180			NS
Address valid before positive transition of clock	TACR	180			NS
Peripheral data valid before positive transition of clock	TPCR	300			NS
Data Bus valid after positive transition of clock	TCDR			395	NS
Data Bus Hold Time	THR	10			NS
$\overline{IRQ}$ (Interval Timer Interrupt) valid before positive transition of clock	TIC	200			NS

Loading = 30 pf + 1 TTL load for PAØ-PA7, PBØ-PB7
= 130 pf + 1 TTL load for DØ-D7

INTERFACE SIGNAL DESCRIPTION

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 ($V_{IL} < 0.2$, $V_{IH} = Vcc \pm .2^{.3}$).

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 SY6532. A high on the R/W pin allows the processor to read (with proper addressing) the data supplied by the SY6532. A low on the R/W pin allows a write (with proper addressing) to the SY6532.

Interrupt Request ($\overline{IRQ}$)

The $\overline{IRQ}$ pin is an interrupt pin from the interrupt control logic. It will be normally high with a low indicating an interrupt from the SY6532. $\overline{IRQ}$ is an open-drain output, permitting several units to be wire-or'ed to the common $\overline{IRQ}$ microprocessor input pin. The $\overline{IRQ}$ pin may be activated by a transition on PA7 or timeout of the interval timer.

Data Bus (D0-D7)

The SY6532 has eight bi-directional data pins (D0-D7). These pins connect to the system's data lines and 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 SY6532 has 16 pins available for peripheral I/O operations. Each pin is individually programmable to act as either an input or an output. The 16 pins are divided into two 8-bit ports, PA0-PA7 and PB0-PB7. PA7 may also function as an interrupt input pin. This feature is described in another section. 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 SY6532 it receives data stored in the data register. The microprocessor will read correct information if the peripheral lines are greater than 2.4 volts for a "1" and less than 0.4 volts for a "0" as the peripheral pins are all TTL compatible. Pins PB0-PB7 are also capable of sourcing 3 ma at 1.5 v thus making them capable of direct transistor drive.

Address Lines (A0-A6)

There are 7 address pins. In addition to these, there is the $\overline{RS}$ pin. The above pins, A0-A6 and $\overline{RS}$, are always used as addressing pins. There are 2 additional pins which are used as CHIP SELECTS. They are pins CS1 and $\overline{CS2}$.

INTERNAL ORGANIZATION

A block diagram of the internal architecture is shown in Figure 1. The SY6532 is divided into four basic sections: RAM, I/O, Timer, and Interrupt Control. The RAM interfaces directly with the microprocessor through the system data bus and address lines. The I/O section consists of two 8-bit halves. Each half contains a Data Direction Register (DDR) and an I/O register.

RAM 128 Bytes (1024 Bits)

A 128 x 8 static RAM is contained on the SY6532. It is addressed by A0-A6 (Byte Select), $\overline{RS}$, CS1, and $\overline{CS2}$.

Internal Peripheral Registers

There are four 8-bit internal registers: two data direction registers and two output registers. The two data direction registers (A side and B side) control the direction of data into and out of the peripheral I/O pins. A logic zero in a bit of the data direction register (DDRA and DDRB) causes the corresponding pin of the I/O port to act as an input. A logic one causes the corresponding pin to act as an output. The voltage on any pin programmed as an output is determined by the corresponding bit in the output register (ORA and ORB).

Data is read directly from the PA pins during a peripheral read operation. Thus, for a PA pin programmed as an output, the data transferred into the processor will be the same as the data in the ORA only if the voltage on the pin is allowed to be $\geqslant$ 2.4 volts for a logic one and $\leqslant$ 0.4 volts for a zero. If the loading on the pin does not allow this, then the data resulting from the read operation may not match the contents of ORA.

The output buffers for the PB pins are somewhat different from the PA buffers. The PB buffers are push-pull devices which are capable of sourcing 3ma at 1.5 volts. This allows for these pins to directly drive transistor circuits. To assure that the processor will read the proper data when performing a peripheral read operation, logic is provided in the peripheral B port to permit the processor to read the contents of ORB, instead of the PB pins as is the case for the PA port.

Interval Timer

The timer section of the SY6532 contains three basic parts: preliminary divide down register, programmable 8-bit register and interrupt logic. These are illustrated in Figure 2.

The interval timer can be programmed to count up to 256 time intervals. Each time interval can be either 1T, 8T, 64T or 1024T increments, where T is the system clock period. When a full count is reached, and interrupt flag is set to a logic "1." After the interrupt flag is set the internal clock begins counting down to a maximum of -255T. Thus, after the interrupt flag is set, a Read of the timer will tell how long since the flag was set up to a maximum of 255T.

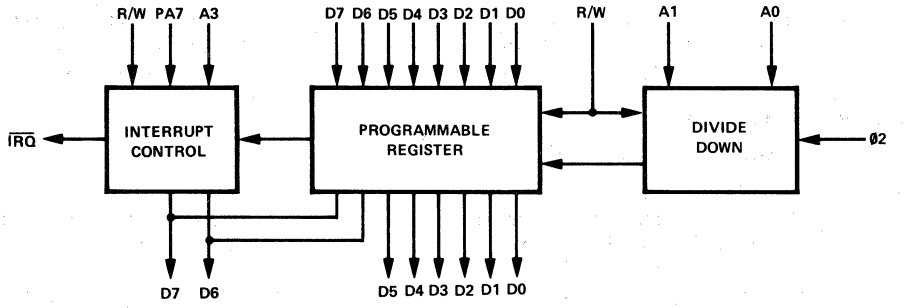
The 8-bit system Data Bus is used to transfer data to and from the Interval Timer. If a count of 52 time intervals were to be counted, the pattern 0 0 1 1 0 1 0 0 would be put on the Data Bus and written into the Interval Time register.

At the same time that data is being written to the Interval Timer, the counting intervals of 1, 8, 64, 1024T are decoded from address lines A0 and A1. During a Read or Write operation address line A3 controls the interrupt capability of $\overline{IRQ}$, i.e., $A_3 = 1$ enables $\overline{IRQ}$, $A_3 = 0$ disables $\overline{IRQ}$. In either case, when timeout occurs, bit 7 of the Interrupt Flag Register is set. This flag is cleared when the Timer register is either read from or written to by the processor. If $\overline{IRQ}$ is enabled by A3 and an interrupt occurs $\overline{IRQ}$ will go low. When the timer is read prior to the interrupt flag being set, the number of time intervals remaining will be read, i.e., 51, 50, 49, etc.

When the timer has counted down to 0 0 0 0 0 0 0 0 on the next count time an interrupt will occur and the counter will read 1 1 1 1 1 1 1 1. After interrupt, the timer register decrements at a divide by "1" rate of the system clock. If after interrupt, the timer is read and a value of 1 1 1 0 0 1 0 0 is read, the time since interrupt is 28T. The value read is in two's complement.

Value read	= 1 1 1 0 0 1 0 0
Complement	= 0 0 0 1 1 0 1 1
Add 1	= 0 0 0 1 1 1 0 0 = 28.

Figure 2. BASIC ELEMENTS OF INTERVAL TIMER

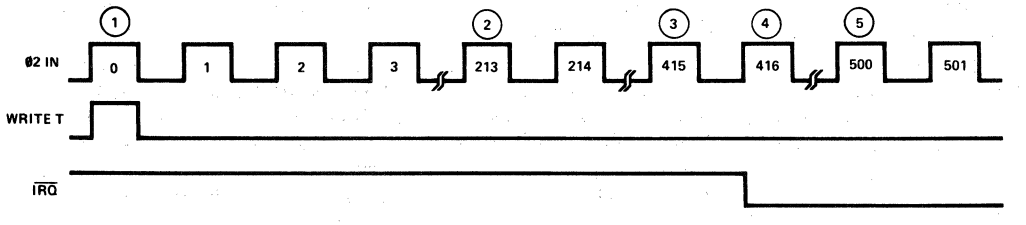


Thus, to arrive at the total elapsed time, merely do a two's complement add to the original time written into the timer. Again, assume time written as 0 0 1 1 0 1 0 0 (=52). With a divide by 8, total time to interrupt is (52 x 8) + 1 = 417T. Total elapsed time would be 416T + 28T = 444T, assuming the value read after interrupt was 1 1 1 0 0 1 0 0.

After interrupt, whenever the timer is written or read the interrupt is reset. However, the reading of the timer at the same time the interrupt occurs will not reset the interrupt flag.

Figure 3 illustrates an example of interrupt.

Figure 3. TIMER INTERRUPT TIMING



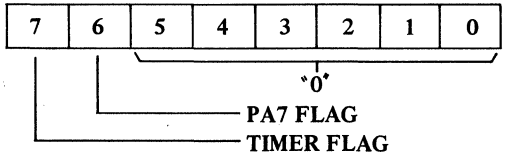
1. Data written into interval timers is $0\,0\,1\,1\,0\,1\,0\,0 = 52_{10}$
2. Data in Interval timer is $0\,0\,0\,1\,1\,0\,0\,1 = 25_{10}$
   $52 - \frac{213}{8} - 1 = 52\text{-}26\text{-}1 = 25$
3. Data in Interval timer is $0\,0\,0\,0\,0\,0\,0\,0 = 0_{10}$
   $52 - \frac{415}{8} - 1 = 52\text{-}51\text{-}1 = 0$
4. Interrupt has occurred at Ø2 pulse #416
   Data in Interval timer = 1 1 1 1 1 1 1 1
5. Data in Interval timer is 1 0 1 0 1 1 0 0
   two's complement is $0\,1\,0\,1\,0\,1\,0\,0 = 84_{10}$
   $84 + (52 \times 8) = 500_{10}$

When reading the timer after an interrupt, A3 should be low so as to disable the $\overline{IRQ}$ pin. This is done so as to avoid future interrupts until after another Write operation.

Interrupt Flag Register

The Interrupt Flag Register consists of two bits: the timer interrupt flag and the PA7 interrupt flag. When a read operation is performed on the Interrupt Flag Register, the bits are transferred to the processor on the data bus, as the diagram below, indicates.

Figure 4. INTERRUPT FLAG REGISTER



The PA7 flag is cleared when the Interrupt Flag Register is read. The timer flag is cleared when the timer register is either written or read.

ADDRESSING

Addressing of the SY6532 is accomplished by the 7 addressing pins, the $\overline{RS}$ pin and the two chip select pins CS1 and $\overline{CS2}$. To address the RAM, CS1 must be high with $\overline{CS2}$ and $\overline{RS}$ low. To address the I/O and Interval timer CS1 and $\overline{RS}$ must be high with $\overline{CS2}$ low. As can be seen to access the chip CS1 is high and $\overline{CS2}$ is low. To distinguish between RAM or I/O Timer the $\overline{RS}$ pin is used. When this pin is low the RAM is addressed, when high the I/O Interval timer section is addressed. To distinguish between timer and I/O address line A2 is utilized. When A2 is high the interval timer is accessed. When A2 is low the I/O section is addressed. Table 1 illustrates the chip addressing.

Edge Sense Interrupt

In addition to its use as a peripheral I/O line, the PA7 pin can function as an edge sensitive input. In this mode, an active transition on PA7 will set the internal interrupt flag (bit 6 of the Interrupt Flag Register). When this occurs, and providing the PA7 interrupt is enabled, the $\overline{IRQ}$ output will go low.

Control of the PA7 edge detecting logic is accomplished by performing a write operation to one of four addresses. The data lines for this operation are "don't care" and the addresses to be used are found in Figure 4.

The setting of the internal Interrupt flag by an active transition on PA7 is always enabled, no matter whether PA7 is set up as an input or an output.

The $\overline{RES}$ signal disables the PA7 interrupt and sets the active transition to the negative edge-detect state. During the reset operation, the interrupt flag may be set by a negative transition. It may, therefore, be necessary to clear the flag before its normal use as an edge detecting input is enabled. This can be achieved by reading the Interrupt Flag Register, as defined by Figure 4 immediately after reset.

I/O Register - Timer Addressing

Table 1 illustrates the address decoding for the internal elements and timer programming. Address line A2 distinguishes I/O registers from the timer. When A2 is low and $\overline{RS}$ is high, the I/O registers are addressed. Once the I/O registers are addressed, address lines A1 and A0 decode the desired register.

When the timer is selected A1 and A0 decode the "divide-by" matrix. This decoding is defined in Table 1. In addition, Address A3 is used to enable the interrupt flag to $\overline{IRQ}$.

Table 1 ADDRESSING DECODE

OPERATION	$\overline{RS}$	R/W	A4	A3	A2	A1	A0
Write RAM	0	0	–	–	–	–	–
Read RAM	0	1	–	–	–	–	–
Write DDRA	1	0	–	–	0	0	1
Read DDRA	1	1	–	–	0	0	1
Write DDRB	1	0	–	–	0	1	1
Read DDRB	1	1	–	–	0	1	1
Write Output Reg A	1	0	–	–	0	0	0
Read Output Reg A	1	1	–	–	0	0	0
Write Output Reg B	1	0	–	–	0	1	0
Read Output Reg B	1	1	–	–	0	1	0
Write Timer							
÷ 1T	1	0	1	(a)	1	0	0
÷ 8T	1	0	1	(a)	1	0	1
÷ 64T	1	0	1	(a)	1	1	0
÷ 1024T	1	0	1	(a)	1	1	1
Read Timer	1	1	–	(a)	1	–	0
Read Interrupt Flag	1	1	–	–	1	–	1
Write Edge Detect Control	1	0	0	–	1	(b)	(c)

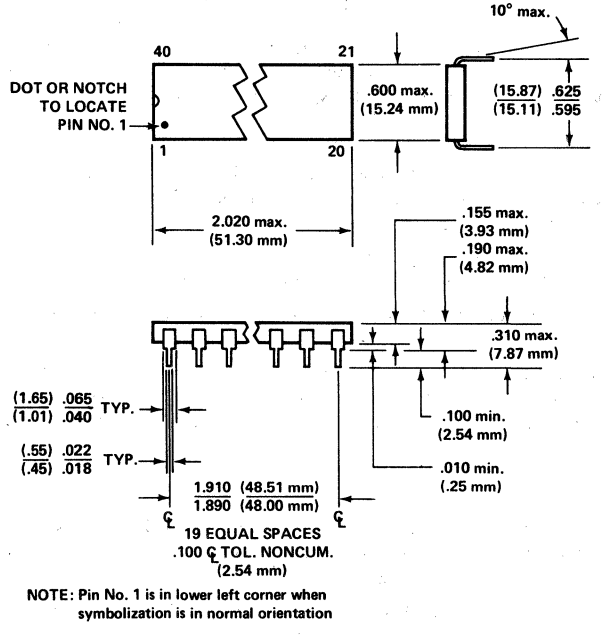
NOTES: – = Don't Care, "1" = High level (≥2.4V), "0" = Low level (≤0.4V)

(a) A3 = 0 to disable interrupt from timer to $\overline{IRQ}$
A3 = 1 to enable interrupt from timer to $\overline{IRQ}$

(b) A1 = 0 to disable interrupt from PA7 to $\overline{IRQ}$
A1 = 1 to enable interrupt from PA7 to $\overline{IRQ}$

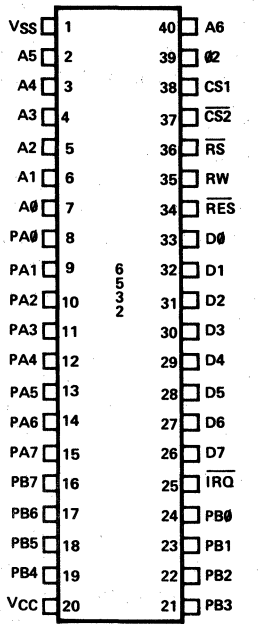
(c) A0 = 0 for negative edge-detect
A0 = 1 for positive edge-detect

PACKAGE OUTLINE



NOTE: Pin No. 1 is in lower left corner when symbolization is in normal orientation

PIN DESIGNATION



APPENDIX L

SY2114 RAM DATA SHEET

1024x4 Static Random Access Memory

SY2114

MEMORY PRODUCTS

SEPTEMBER 1978

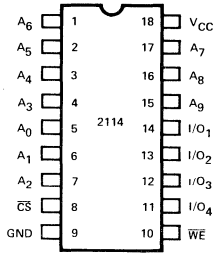
- 200 ns Maximum Access
- Low Operating Power Dissipation 0.1 mW/Bit
- No Clocks or Strobes Required
- Identical Cycle and Access Times
- Single +5V Supply
- Totally TTL Compatible: All Inputs, Outputs, and Power Supply
- Common Data I/O
- 400 mv Noise Immunity
- High Density 18 Pin Package

The SY2114 is a 4096-Bit static Random Access Memory organized 1024 words by 4-bits and is fabricated using Synertek's N-channel Silicon-Gate MOS technology. It is designed using fully DC stable (static) circuitry in both the memory array and the decoding and therefore requires no clock or refreshing to operate. Address setup times are not required and the data is read out nondestructively with the same polarity as the input data. Common Input/Output pins are provided to simplify design of the bus oriented systems, and can drive 1 TTL load.

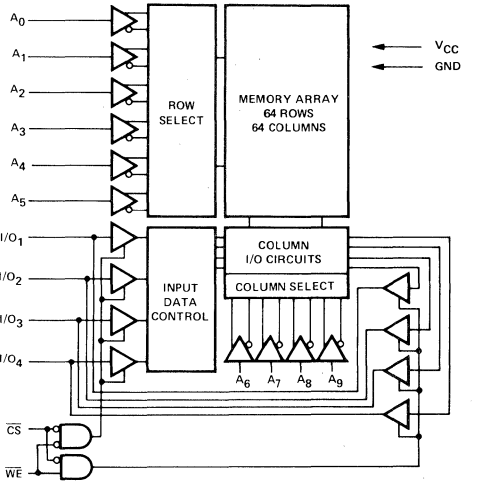
The SY2114 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives. It is totally TTL compatible in all respects: inputs, outputs, and the single +5V supply. A separate Chip Select ($\overline{CS}$) input allows easy selection of an individual device when outputs are or-tied.

The SY2114 is packaged in an 18-pin DIP for the highest possible density and is fabricated with N-channel, Ion Implanted, Silicon-Gate technology — a technology providing excellent performance characteristics as well as improved protection against contamination.

PIN CONFIGURATION



BLOCK DIAGRAM



ORDERING INFORMATION

Order Number	Package Type	Access Time	Supply Current (Max)	Temperature Range
SYC2114	Ceramic	450nsec	100mA	0°C to 70°C
SYP2114	Molded	450nsec	100mA	0°C to 70°C
SYC2114-3	Ceramic	300nsec	100mA	0°C to 70°C
SYP2114-3	Molded	300nsec	100mA	0°C to 70°C
SYC2114L	Ceramic	450nsec	70mA	0°C to 70°C
SYP2114L	Molded	450nsec	70mA	0°C to 70°C
SYC2114L-3	Ceramic	300nsec	70mA	0°C to 70°C
SYP2114L-3	Molded	300nsec	70mA	0°C to 70°C
SYC2114-2	Ceramic	200nsec	100mA	0°C to 70°C
SYP2114-2	Molded	200nsec	100mA	0°C to 70°C
SYC2114L-2	Ceramic	200nsec	70mA	0°C to 70°C
SYP2114L-2	Molded	200nsec	70mA	0°C to 70°C

SYNERTEK® INC. • P.O. BOX 552 • SANTA CLARA, CALIFORNIA 95052 • TEL. (408) 988-5600 • TWX: 910-338-0135

ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	-10°C to 80°C
Storage Temperature	−65°C to 150°C
Voltage on Any Pin with Respect to Ground	−0.5V to +7 V
Power Dissipation	1.0W

COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5V ± 5% (Unless otherwise specified)

Symbol	Parameter	2114-2 2114-3,2114 Min	2114-2 2114-3,2114 Max	2114L-2 2114L,2114L-3 Min	2114L-2 2114L,2114L-3 Max	Unit	Conditions
I_{LI}	Input Load Current (All input pins)		10		10	μA	V_{IN} = 0 to 5.25V
I_{LO}	I/O Leakage Current		10		10	μA	$\overline{CS}$ = 2.0V, $V_{I/O}$ = 0.4V to V_{CC}
I_{CC1}	Power Supply Current		95		65	mA	V_{CC} = 5.25V, $I_{I/O}$ = 0 mA, T_A = 25°C
I_{CC2}	Power Supply Current		100		70	mA	V_{CC} = 5.25V, $I_{I/O}$ = 0 mA, T_A = 0°C
V_{IL}	Input Low Voltage	-0.5	0.8	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4		0.4	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage	2.4	V_{CC}	2.4	V_{CC}	V	I_{OH} = -1.0 mA

CAPACITANCE T_A = 25°C, f = 1.0 MHz

Symbol	Test	Typ	Max	Units
$C_{I/O}$	Input/Output Capacitance		5	pF
C_{IN}	Input Capacitance		5	pF

NOTE: This parameter is periodically sampled and not 100% tested.

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5% (Unless Otherwise Specified)

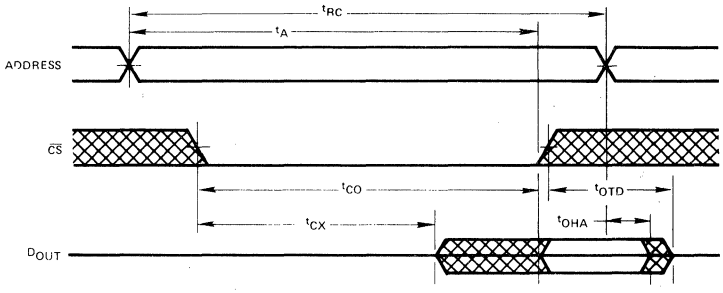
Symbol	Parameter	2114-2,2114L-2 Min.	2114-2,2114L-2 Max.	2114-3,2114L-3 Min.	2114-3,2114L-3 Max.	2114,2114L Min.	2114,2114L Max.	Unit
Read Cycle								
t_{RC}	Read Cycle Time	200		300		450		nsec
t_A	Access Time		200		300		450	nsec
t_{CO}	Chip Select to Output Valid		70		100		120	nsec
t_{CX}	Chip Select to Output Enabled	20		20		20		nsec
t_{OTD}	Chip Deselect to Output Off	0	60	0	80	0	100	nsec
t_{OHA}	Output Hold From Address Change	50		50		50		nsec
Write Cycle								
t_{WC}	Write Cycle Time	200		300		450		nsec
t_{AW}	Address to Write Setup Time	0		0		0		nsec
t_W	Write Pulse Width	120		150		200		nsec
t_{WR}	Write Release Time	0		0		0		nsec
t_{OTW}	Write to Output Off	0	60	0	80	0	100	nsec
t_{DW}	Data to Write Overlap	120		150		200		nsec
t_{DH}	Data Hold	0		0		0		nsec

A.C. Test Conditions

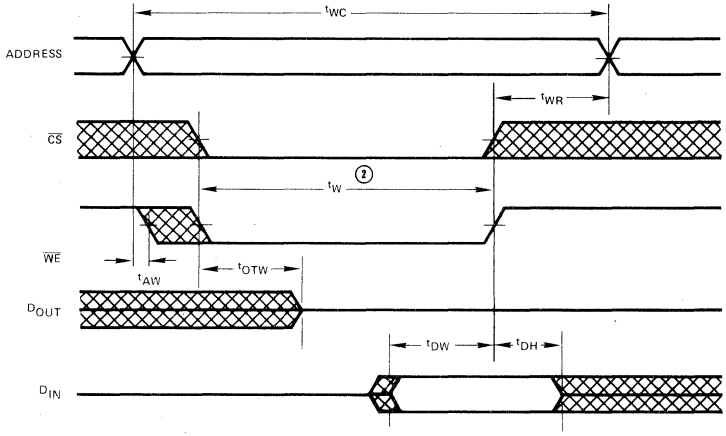
Input Pulse Levels . . . . . 0.8V to 2.0V
Input Rise and Fall Time . . . . . 10 n sec
Timing Measurement Levels: Input 1.5V
Output . . . . . 0.8 and 2.0V
Output Load . . . . . 1TTL Gate and 100pF

TIMING DIAGRAMS

Read Cycle ①



Write Cycle



NOTES:

① $\overline{WE}$ is high for a Read Cycle
② t_W is measured from the latter of $\overline{CS}$ or $\overline{WE}$ going low to the earlier of $\overline{CS}$ or $\overline{WE}$ going high.

DATA STORAGE

When $\overline{WE}$ is high, the data input buffers are inhibited to prevent erroneous data from being written into the array. As long as $\overline{WE}$ remains high, the data stored cannot be affected by the Address, Chip Select, or Data I/O logic levels or timing transitions.

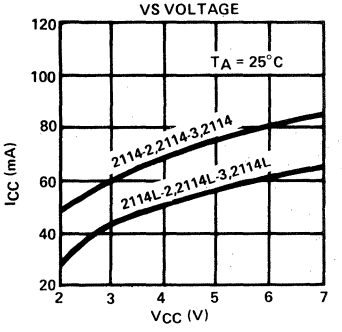
Data storage also cannot be affected by $\overline{WE}$, Addresses, or the I/O ports as long as $\overline{CS}$ is high. Either $\overline{CS}$ or $\overline{WE}$ or both can prevent extraneous writing due to signal transitions.

Data within the array can only be changed during Write time – defined as the overlap of CS low and $\overline{WE}$ low. The addresses must be properly established during the entire Write time plus t_{WR}.

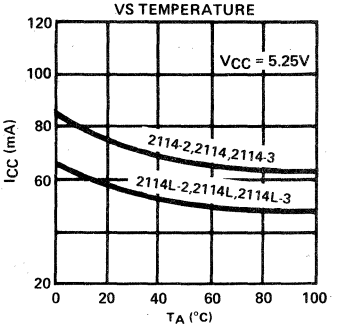
Internal delays are such that address decoding propagates ahead of data inputs and therefore no address setup time is required. If the Write time precedes the addresses, the data in previously addressed locations, or some other location, may be changed. Addresses must remain stable for the entire Write cycle but the Data Inputs may change. The data which is stable for t_{DW} at the end of the Write time will be written into the addressed location.

TYPICAL CHARACTERISTICS

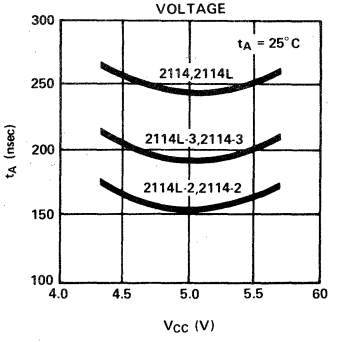
SUPPLY CURRENT VS VOLTAGE



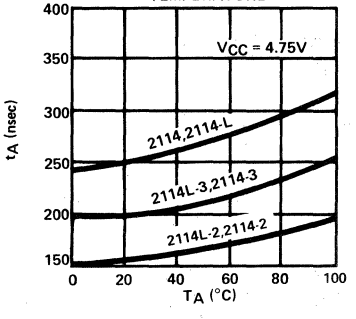
SUPPLY CURRENT VS TEMPERATURE



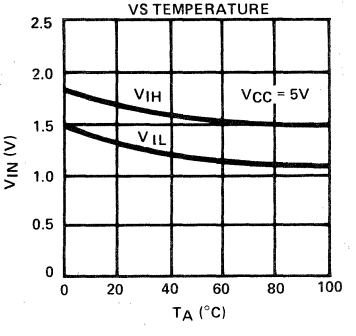
ACCESS TIME VS VOLTAGE



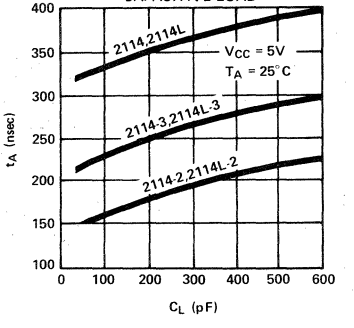
ACCESS TIME VS TEMPERATURE



INPUT VOLTAGE LIMITS VS TEMPERATURE

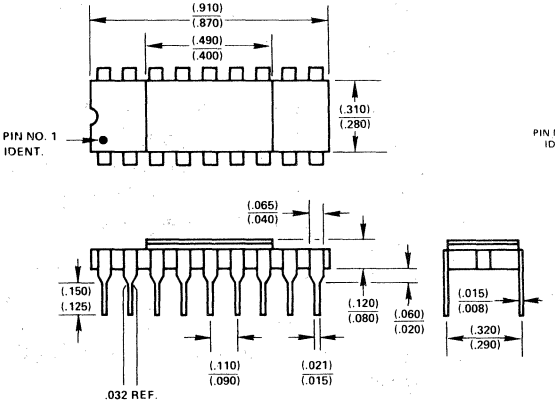


ACCESS TIME VS CAPACITIVE LOAD

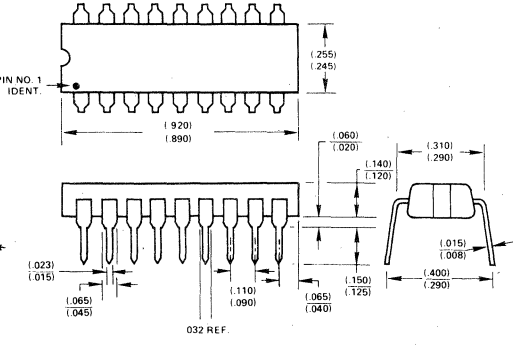


PACKAGE DIAGRAM

CERAMIC PACKAGE



MOLDED PACKAGE



APPENDIX M

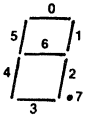
VERSION 1.1 MONITOR ENHANCEMENTS

SY1.1, the second release of the SUPERMON monitor program, includes several enhancements and fixes to SY1.0.

All entry point addresses in the main monitor program remain unchanged from version SY1.0. In the cassette routines, however, only LOADT and DUMPT can be relied upon to be at the old addresses.

The most useful enhancements come in the cassette software. First, the high speed tape leader time is now variable. It can be changed by modifying location TAPDEL, A630. TAPDEL is interpreted in units of 1.5 seconds of SYNC characters.

While reading a cassette file, its file ID will be displayed in non-decoded form on the leftmost LED. The display may be interpreted as follows:



where the numbers alongside the segments indicate the binary bit which corresponds to that segment.

Example:

File 01

File 02

File 54

File FE

File 03

During the SYNC search (while the "S." is lit on the display), the tape load operation may be interrupted by depressing the CR button on the hex keypad and turning off the recorder. The message Er 8C will be displayed.

The high speed write waveform is now variable. For information on using this feature, see technical note #72-SSC.

The error count after a paper tape load operation is displayed. The first digit of the error number indicates the number of invalid characters of checksums encountered in the tape, and the second digit indicates the number of memory errors (invalid readbacks). In each case, the digit 'F' indicates greater than or equal to 15 errors.

APPENDIX N

SYM I/O SOFTWARE

Using the SYM's character I/O routines in your software is easy and provides great flexibility.

The I/O devices to be discussed are:

- CRT An RS-232 serial ASCII terminal
- TTY A 20mA loop serial ASCII device
- HKB The 25 key ASCII keyboard on the SYM
- DSP The 6 digit 7 segment ASCII display on the SYM
- SCP The oscilloscope driver circuitry on the SYM
- UIN A user provided special input device
- UOUT A user provided special output device

The software modules are:

- VEC	I/O Vectors in Monitor RAM
- TIO	Serial Terminal drivers in ROM
- HDOUT, HKEY HKB and DSP	Drivers in ROM
- SCD	SCP drivers in RAM
- INCHR, OUTCHR	Character I/O subroutines in ROM
- UIO	User I/O drivers in RAM
- MON	The monitor program itself, exclusive of I/O

The easiest to use method of CHAR I/O is used by MON, this method is explained first.

Whenever the monitor wants to output a character, it places the ASCII value of the character in A and calls OUTCHR. The character is output and monitor resumes. That character may have gone out to a CRT, TTY, or both, DSP, SCP, or UOUT.

The vector OUTVEC controlled which device received the output by containing the address of the driver for the receiving device. If the output device is a CRT or TTY, then OUTVEC contains the address of ("points to") a driver called TOUT (Terminal OUT). When the DSP is the output device, OUTVEC points to HDOUT. The decision of which output driver to point to is made initially by monitor at sign on time. If, after RESET, a key on HKB is pressed, OUTVEC is left pointing to HDOUT (and INVEC pointing to HKEY). If a "1" or "Q" is pressed on the CRT, then OUTVEC is loaded with the address of TOUT. The I/O Vectors will not be changed again until RESET or the user loads them himself. If you wish to output characters to the device that you signed in on, then place the character in A and JSR to OUTCHR. Output will be disabled if the OUTPUT DISABLE FLAG (bit 6 of TECHO) is set.

When monitor needs to input a character, it calls INCHR. Control is vectored thru INVEC so that input is obtained from the device, CRT or HKB, that was signed in on. Several things occur to the character as it is being input. If it is lower case alphabetic, it is changed to upper case. Bit 7 is cleared. If the character is Control 0, ASCII ØF, then the OUTPUT DISABLE FLAG is toggled, and another character is obtained. The ASCII CHAR is returned in A. Both INCHR and OUTCHR are transparent to the X and Y registers.

When you connect a UIN or UOUT to the SYM, you must also write the I/O driver for it. If UIN is an unencoded keyboard, for instance, you must connect it to a 6522 VIA's I/O ports, and write the driver that will configure the VIA, scan the keys, debounce them, and return the ASCII character in A. To use this UIN, sign in on HKB or CRT, then change INVEC to point to the UIN input driver subroutine. The MON command SD is provided for this purpose. Now whenever MON or BASIC or a user program calls INCHR, the character will be obtained from UIN. Similarly, if the output device is to be an oscilloscope, then the scope driver subroutine provided in this manual should be installed in RAM and OUTVEC (if necessary) and SCNVEC should be pointed to this subroutine. All characters sent to OUTCHR will appear on the scope.

While HKB and DSP are in use, the following vectors are used: INVEC, OUTVEC, INSVEC (points to the routine that checks the input device for a BREAK condition), and SCNVEC (lights up the display or oscilloscope, etc. while waiting for an input key).

By vectoring the I/O on the SYM, any I/O devices can be used by all the software by calling INCHR and OUTCHR. If you wish to input lower case letters, then JSR INVEC directly or duplicate the code in bytes 8A1B thru 8A40 in RAM and change bytes 8A23 thru 8A2C to NOP's. Note that placing I/O routines in RAM can cause difficulties during Trace or Debug because the I/O routine used to print the Trace may itself get traced.

If your software needs special capability, you may choose to call MON I/O subroutines directly. For instance, lighting up the DSP to do a "video" game like ping-pong requires that you display some patterns that may not have ASCII equivalents. In this case the six locations of DISBUF should be poked to display the desired pattern and SCAND repeatedly called to light up the display. The subroutine ACCESS can be called to unwrite-protect the MON RAM. To control the game the HKB keys "4" and "CR" are especially useful. The subroutines KYSTAT and LRNKEY will tell you whether any key is down or not, and if so, which key is down, in ASCII. When an oscilloscope is used for output, the subroutine TEXT can do the bookkeeping for keeping the last 32 CHARS in SCPBUF ready for use.

Since all character I/O on the SYM can be in ASCII, even with the HKB and DSP, your software can use the HKB and DSP, CRT, TTY, SCP, or special I/O devices, without having to be rewritten. MON, for instance, doesn't even "know" whether the HKB and DSP or a CRT are being used for I/O.

If you have used other systems, maybe the KIM, for instance, some differences should be apparent. More I/O capability exists on the SYM than is typical. While no two systems handle I/O alike, most of the I/O routines accomplish the same task. In the KIM, the display can be lighted with hexadecimal characters, alas with no decimal points, by storing the hex in three bytes 00F9, 00FA, and 00FB. In the SYM the same thing is done by calling OUTBYT three times with the hex bytes in A. Or call OUTCHR up to 6 times, or store the 7 segment codes in DISBUF. To get a keycode from the HKB use: HKEY, GETKEY, GK, LRNKEY, and/or KYSTAT. Note that if you call GK, not only is the ASCII code returned in A, but X contains a "key code". The key codes for Ø thru F are ØØ thru ØF, the hex nibble equivalents, very handy. Since MON is 85% subroutines, the potential to save time and code by calling MON subroutines is unusually large. The program listing in the back of the manual provides the details of how all subroutines work.

SUPERMON Monitor Listing

```
LINE #  LOC     CODE        LINE

0002  0000                ;
0003  0000                ;*****
0004  0000                ;***** COPYRIGHT 1979 SYNERTEK SYSTEMS CORPORATION
0005  0000                ;***** VERSION 2  4/13/79  "SY1.1"
0006  0000                        *=$A600          ;SYS RAM (ECHOED AT TOP OF MEM)
0007  A600                SCPBUF  *=*+$20          ;SCOPE BUFFER LAST 32 CHRS
0008  A620                RAM     =*               ;DEFAULT BLK FILLS STARTING HERE
0009  A620                JTABLE  *=*+$10          ; BJUMPS - ABS ADDR, LO HI ORDER
0010  A630                TAPDEL  *=*+1            ;HS TAPE DELAY
0011  A631                KMBDRY  *=*+1            ;KIM TAPE READ BOUNDARY
0012  A632                HSBDRY  *=*+1            ;HS TAPE READ BOUNDARY
0013  A633                SCR3    *=*+1            ;RAM SCRATCH LOCS 3-F
0014  A634                SCR4    *=*+1
0015  A635                TAPET1  *=*+1            ;HS TAPE 1/2 BIT TIME
0016  A636                SCR6    *=*+1
0017  A637                SCR7    *=*+1
0018  A638                SCR8    *=*+1
0019  A639                SCR9    *=*+1
0020  A63A                SCRA    *=*+1
0021  A63B                SCRB    *=*+1
0022  A63C                TAPET2  *=*+1            ;HS TAPE 1/2 BIT TIME
0023  A63D                SCRD    *=*+1
0024  A63E                RC      =SCRD
0025  A63E                SCRE    *=*+1
0026  A63F                SCRF    *=*+1
0027  A640                DISBUF  *=*+5            ;DISPLAY BUFFER
0028  A645                RDIG    *=*+1            ;RIGHT MOST DIGIT OF DISPLAY
0029  A646                        *=*+3            ;NOT USED
0030  A649                PARNR   *=*+1            ;NUMBER OF PARMS RECEIVED
0031  A64A                ;
0032  A64A                ; 3 16 BIT PARMS, LO HI ORDER
0033  A64A                ; PASSED TO EXECUTE BLOCKS
0034  A64A                ;
0035  A64A                P3L     *=*+1
0036  A64B                P3H     *=*+1
0037  A64C                P2L     *=*+1
0038  A64D                P2H     *=*+1
0039  A64E                P1L     *=*+1
0040  A64F                P1H     *=*+1
0041  A650                PADBIT  *=*+1            ;PAD BITS FOR CARRIAGE RETURN
0042  A651                SDBYT   *=*+1            ;SPEED BYTE FOR TERMINAL I/O
0043  A652                ERCNT   *=*+1            ; ERROR COUNT  (MAX $FF)
0044  A653                ; BIT 7 = ECHO /NO ECHO, BIT 6 = CTL O TOGGLE SW
0045  A653                TECHO   *=*+1            ;TERMINAL ECHO FLAG
0046  A654                ; BIT7 =CRT IN, 6 =TTY IN, 5 = TTY OUT, 4 = CRT OUT
0047  A654                TOUTFL  *=*+1            ;OUTPUT FLAGS
0048  A655                KSHFL   *=*+1            ;KEYBOARD SHIFT FLAG
0049  A656                TV      *=*+1            ;TRACE VELOCITY (0=SINGLE STEP)
0050  A657                LSTCOM  *=*+1            ;STORE LAST MONITOR COMMAND
0051  A658                MAXRC   *=*+1            ;MAX REC LENGTH FOR MEM DUMP
0052  A659                ;
0053  A659                ; USER REG'S FOLLOW
0054  A659                ;
0055  A659                PCLR    *=*+1            ;PROG CTR
0056  A65A                PCHR    *=*+1
```

```
LINE # LOC      CODE        LINE

0057  A65B                SR      *=*+1           ;STACK
0058  A65C                FR      *=*+1           ;FLAGS
0059  A65D                AR      *=*+1           ;AREG
0060  A65E                XR      *=*+1           ;XREG
0061  A65F                YR      *=*+1           ;YREG
0062  A660                ;
0063  A660                ; I/O VECTORS FOLLOW
0064  A660                ;
0065  A660                INVEC   *=*+3           ;IN CHAR
0066  A663                OUTVEC  *=*+3           ;OUT CHAR
0067  A666                INSVEC  *=*+3           ;IN STATUS
0068  A669                URSVEC  *=*+3           ;UNRECOGNIZED SYNTAX VECTOR
0069  A66C                URCVEC  *=*+3           ;UNRECOGNIZED CMD/ERROR VECTOR
0070  A66F                SCNVEC  *=*+3           ;SCAN ON-BOARD DISPLAY
0071  A672                ;
0072  A672                ; TRACE, INTERRUPT VECTORS
0073  A672                ;
0074  A672                EXEVEC  *=*+2           ; EXEC CMD ALTERNATE INVEC
0075  A674                TRCVEC  *=*+2           ;TRACE
0076  A676                UBRKVC  *=*+2           ;USER BRK AFTER MONITOR
0077  A678                UBRKV   =UBRKVC
0078  A678                UIRQVC  *=*+2           ;USER NON-BRK IRQ AFTER MONITOR
0079  A67A                UIRQV   =UIRQVC
0080  A67A                NMIVEC  *=*+2           ;NMI
0081  A67C                RSTVEC  *=*+2           ;RESET
0082  A67E                IRQVEC  *=*+2           ;IRQ
0083  A680                ;
0084  A680                ;
0085  A680                ;I/O REG DEFINITIONS
0086  A680                PADA    =$A400 ORA      ;KEYBOARD/DISPLAY
0087  A680                PBDA    =$A402 ORB      ;SERIAL I/O
0088  A680                OR3A=$AC01              ;WP, DBON, DBOFF
0089  A680                DDR3A=OR3A+2            ;DATA DIRECTION FOR SAME
0090  A680                OR1B=$A000
0091  A680                DDR1B=$A002
0092  A680                PCR1    =$A00C          ; POR/TAPE REMOTE
0093  A680                ;
0094  A680                ; MONITOR MAINLINE
0095  A680                ;
0096  A680                      *=$8000
0097  8000  4C 7C 8B     MONITR JMP MONENT       ;INIT S, CLD, GET ACCESS
0098  8003  20 FF 80     WARM   JSR GETCOM       ;GET COMMAND + PARMS  (0-3)
0099  8006  20 4A 81            JSR DISPAT       ;DISPATCH CMD,PARMS TO EXEC BLKS
0100  8009  20 71 81            JSR ERMSG        ;DISP ER MSG IF CARRY SET
0101  800C  4C 03 80            JMP WARM         ;AND CONTINUE
0102  800F                ;
0103  800F                ; TRACE AND INTERRUPT ROUTINES
0104  800F                ;
0105  800F  08            IRQBRK PHP            ;IRQ OR BRK ?
0106  8010  48                   PHA
0107  8011  8A                   TXA
0108  8012  48                   PHA
0109  8013  BA                   TSX
0110  8014  BD 04 01             LDA $104,X      ;PICK UP FLAGS
0111  8017  29 10                AND #$10
```

```
LINE # LOC     CODE       LINE

0112  8019  F0 07          BEQ DETIRQ
0113  801B  68          DETBRK PLA              ;BRK
0114  801C  AA              TAX
0115  801D  68              PLA
0116  801E  28              PLP
0117  801F  6C F6 FF          JMP ($FFF6)
0118  8022  68          DETIRQ PLA              ;IRQ (NON BRK)
0119  8023  AA              TAX
0120  8024  68              PLA
0121  8025  28              PLP
0122  8026  6C F8 FF          JMP ($FFF8)
0123  8029  20 86 8B    SVIRQ  JSR ACCESS       ;SAVE REGS AND DISPLAY CODE
0124  802C  38              SEC
0125  802D  20 64 80          JSR SAVINT
0126  8030  A9 31            LDA #'1
0127  8032  4C 53 80          JMP IDISP
0128  8035  08          USRENT PHP              ;USER ENTRY
0129  8036  20 86 8B          JSR ACCESS
0130  8039  38              SEC
0131  803A  20 64 80          JSR SAVINT
0132  803D  EE 59 A6          INC PCLR
0133  8040  D0 03            BNE *+5
0134  8042  EE 5A A6          INC PCHR
0135  8045  A9 33            LDA #'3
0136  8047  4C 53 80          JMP IDISP
0137  804A  20 86 8B    SVBRK  JSR ACCESS
0138  804D  18              CLC
0139  804E  20 64 80          JSR SAVINT
0140  8051  A9 30            LDA #'0
0141  8053              ; INTRPT CODES  0 = BRK
0142  8053              ;               1 = IRQ
0143  8053              ;               2 = NMI
0144  8053              ;               3 = USER ENTRY
0145  8053  48          IDISP  PHA              ;OUT PC, INTRPT CODE (FROM A)
0146  8054  20 D3 80          JSR DBOFF         ;STOP NMI'S
0147  8057  20 4D 83          JSR CRLF
0148  805A  20 37 83          JSR OPCCOM
0149  805D  68              PLA
0150  805E  20 47 8A          JSR OUTCHR
0151  8061  4C 03 80          JMP WARM
0152  8064  8D 5D A6    SAVINT STA AR           ;SAVE USER REGS AFTER INTRPT
0153  8067  8E 5E A6          STX XR
0154  806A  8C 5F A6          STY YR
0155  806D  BA              TSX
0156  806E  D8              CLD
0157  806F  BD 04 01          LDA $104,X
0158  8072  69 FF            ADC #$FF
0159  8074  8D 59 A6          STA PCLR
0160  8077  BD 05 01          LDA $105,X
0161  807A  69 FF            ADC #$FF
0162  807C  8D 5A A6          STA PCHR
0163  807F  BD 03 01          LDA $103,X
0164  8082  8D 5C A6          STA FR
0165  8085  BD 02 01          LDA $102,X
0166  8088  9D 05 01          STA $105,X
```

```
LINE # LOC      CODE        LINE

0167   808B   BD 01 01            LDA $101,X
0168   808E   9D 04 01            STA $104,X
0169   8091   E8                  INX
0170   8092   E8                  INX
0171   8093   E8                  INX
0172   8094   9A                  TXS
0173   8095   E8                  INX
0174   8096   E8                  INX
0175   8097   8E 5B A6            STX SR
0176   809A   60                  RTS
0177   809B   20 86 8B     SVNMI   JSR ACCESS       ;TRACE IF TV NE 0
0178   809E   38                  SEC
0179   809F   20 64 80            JSR SAVINT
0180   80A2   20 D3 80            JSR DBOFF        ;STOP NMI'S
0181   80A5   AD 56 A6            LDA TV
0182   80A8   D0 05               BNE TVNZ
0183   80AA   A9 32               LDA #'2
0184   80AC   4C 53 80            JMP IDISP
0185   80AF   20 37 83     TVNZ    JSR OPCCOM       ;TRACE WITH DELAY
0186   80B2   AD 5D A6            LDA AR
0187   80B5   20 4A 83            JSR OBCRLF       ;DISPLAY ACC
0188   80B8   20 5A 83            JSR DELAY
0189   80BB   90 10               BCC TRACON       ;STOP IF KEY ENTERED
0190   80BD   4C 03 80            JMP WARM
0191   80C0   20 86 8B     TRCOFF  JSR ACCESS       ;DISABLE NMIS
0192   80C3   38                  SEC
0193   80C4   20 64 80            JSR SAVINT
0194   80C7   20 D3 80            JSR DBOFF
0195   80CA   6C 74 A6            JMP (TRCVEC)     ;AND GO TO SPECIAL TRACE
0196   80CD   20 E4 80     TRACON  JSR DBON         ;ENABLE NMI'S
0197   80D0   4C FD 83            JMP GO1ENT+3     ;AND RESUME (NO WRITE PROT)
0198   80D3   AD 01 AC     DBOFF   LDA OR3A         ;PULSE DEBUG OFF
0199   80D6   29 DF               AND #$DF
0200   80D8   09 10               ORA #$10
0201   80DA   8D 01 AC            STA OR3A
0202   80DD   AD 03 AC            LDA DDR3A
0203   80E0   09 30               ORA #$30
0204   80E2   D0 0F               BNE DBNEW-3      ;RELEASE FLIP FLOP SO KEY WORKS
0205   80E4   AD 01 AC     DBON    LDA OR3A         ;PULSE DEBUG ON
0206   80E7   29 EF               AND #$EF
0207   80E9   09 20               ORA #$20
0208   80EB   8D 01 AC            STA OR3A
0209   80EE   AD 03 AC            LDA DDR3A
0210   80F1   09 30               ORA #$30
0211   80F3   8D 03 AC            STA DDR3A
0212   80F6   AD 03 AC     DBNEW   LDA DDR3A        ;RELEASE FLIP FLOP
0213   80F9   29 CF               AND #$CF
0214   80FB   8D 03 AC            STA DDR3A
0215   80FE   60                  RTS
0216   80FF                ;
0217   80FF                ; GETCOM - GET COMMAND AND 0-3 PARMS
0218   80FF                ;
0219   80FF   20 4D 83     GETCOM  JSR CRLF
0220   8102   A9 2E               LDA #'.          ;PROMPT
0221   8104   20 47 8A            JSR OUTCHR
```

```
LINE # LOC     CODE        LINE

0222  8107  20 1B 8A    GETC1  JSR INCHR
0223  810A  F0 F3               BEQ GETCOM       ;CARRIAGE RETURN?
0224  810C  C9 7F               CMP #$7F         ;DELETE?
0225  810E  F0 F7               BEQ GETC1
0226  8110  C9 00               CMP #0           ;NULL?
0227  8112  F0 F3               BEQ GETC1
0228  8114                    ; L,S,U NEED TO BE HASHED 2 BYTES TO ONE
0229  8114  C9 53               CMP #'S
0230  8116  F0 1B               BEQ HASHUS
0231  8118  C9 55               CMP #'U
0232  811A  F0 17               BEQ HASHUS
0233  811C  C9 4C               CMP #'L
0234  811E  F0 0F               BEQ HASHL
0235  8120  8D 57 A6    STOCOM STA LSTCOM
0236  8123  20 42 83           JSR SPACE
0237  8126  20 08 82           JSR PSHOVE       ;ZERO PARMS
0238  8129  20 08 82           JSR PSHOVE
0239  812C  4C 20 82           JMP PARM        ;AND GO GET PARMS
0240  812F  A9 01       HASHL  LDA #$01         ;HASH LOAD CMDS TO ONE BYTE
0241  8131  10 02               BPL HASHUS+2
0242  8133  0A         HASHUS ASL A            ;HASH 'USER' CMDS TO ONE BYTE A
0243  8134  0A                 ASL A            ;U0 = $14 THRU U7 =$1B
0244  8135  8D 57 A6           STA LSTCOM
0245  8138  20 1B 8A           JSR INCHR        ;GET SECOND
0246  813B  F0 C2               BEQ GETCOM       ;CARRIAGE RETURN?
0247  813D  18                 CLC
0248  813E  6D 57 A6           ADC LSTCOM
0249  8141  29 0F               AND #$0F
0250  8143  09 10               ORA #$10
0251  8145  10 D9               BPL STOCOM
0252  8147  FF                 .BYT $FF,$FF,$FF ;NOT USED
0252  8148  FF
0252  8149  FF
0253  814A                    ;
0254  814A                    ;DISPATCH TO EXEC BLK 0PARM, 1PARM, 2PARM, OR 3PARM
0255  814A                    ;
0256  814A  C9 0D       DISPAT CMP #$0D         ;C/R IF OK ELSE URSVEC
0257  814C  D0 20               BNE HIPN
0258  814E  AD 57 A6           LDA LSTCOM
0259  8151  AE 49 A6           LDX PARNR
0260  8154  D0 03               BNE M12
0261  8156  4C 95 83           JMP BZPARM       ;0 PARM BLOCK
0262  8159  E0 01       M12     CPX #$01
0263  815B  D0 03               BNE M13
0264  815D  4C DA 84           JMP B1PARM       ;1PARM BLOCK
0265  8160  E0 02       M13     CPX #$02
0266  8162  D0 03               BNE M14
0267  8164  4C 19 86           JMP B2PARM      ;2 PARM BLOCK
0268  8167  E0 03       M14     CPX #$03
0269  8169  D0 03               BNE HIPN
0270  816B  4C 14 87           JMP B3PARM      ;3 PARM BLOCK
0271  816E  6C 6A A6    HIPN   JMP (URSVEC+1)  ;ELSE UNREC SYNTAX VECTOR
0272  8171                    ;
0273  8171                    ; ERMSG - PRINT ACC IN HEX IF CARRY SET
0274  8171                    ;
```

```
LINE # LOC      CODE        LINE

0275  8171  90 44        ERMSG  BCC M15
0276  8173  48                 PHA
0277  8174  20 4D 83            JSR CRLF
0278  8177  A9 45               LDA #'E
0279  8179  20 47 8A            JSR OUTCHR
0280  817C  A9 52               LDA #'R
0281  817E  20 47 8A            JSR OUTCHR
0282  8181  20 42 83            JSR SPACE
0283  8184  68                 PLA
0284  8185  4C FA 82            JMP OUTBYT
0285  8188                  ;
0286  8188                  ; SAVER - SAVE ALL REG'S + FLAGS ON STACK
0287  8188                  ; RETURN WITH F,A,X,Y UNCHANGED
0288  8188                  ; STACK HAS           FLAGS,A,X,Y PUSHED
0289  8188  08              SAVER  PHP                ;
0290  8189  48                     PHA                ;
0291  818A  48                     PHA                ;
0292  818B  48                     PHA
0293  818C  08                     PHP
0294  818D  48                     PHA
0295  818E  8A                     TXA
0296  818F  48                     PHA
0297  8190  BA                     TSX
0298  8191  BD 09 01               LDA $0109,X
0299  8194  9D 05 01               STA $0105,X
0300  8197  BD 07 01               LDA $0107,X
0301  819A  9D 09 01               STA $0109,X
0302  819D  BD 01 01               LDA $0101,X
0303  81A0  9D 07 01               STA $0107,X
0304  81A3  BD 08 01               LDA $0108,X
0305  81A6  9D 04 01               STA $0104,X
0306  81A9  BD 06 01               LDA $0106,X
0307  81AC  9D 08 01               STA $0108,X
0308  81AF  98                     TYA
0309  81B0  9D 06 01               STA $0106,X
0310  81B3  68                     PLA
0311  81B4  AA                     TAX
0312  81B5  68                     PLA
0313  81B6  28                     PLP
0314  81B7  60              M15    RTS
0315  81B8                  ; RESTORE EXCEPT A,F
0316  81B8  08              RESXAF PHP
0317  81B9  BA                     TSX
0318  81BA  9D 04 01               STA $0104,X
0319  81BD  28                     PLP
0320  81BE                  ; RESTORE EXCEPT F
0321  81BE  08              RESXF  PHP
0322  81BF  68                     PLA
0323  81C0  BA                     TSX
0324  81C1  9D 04 01               STA $0104,X
0325  81C4                  ; RESTORE ALL 100%
0326  81C4  68              RESALL PLA
0327  81C5  A8                     TAY
0328  81C6  68                     PLA
0329  81C7  AA                     TAX
```

```
LINE # LOC     CODE        LINE

0330  81C8  68              PLA
0331  81C9  28              PLP
0332  81CA  60              RTS
0333  81CB                 ;
0334  81CB                 ; MONITOR UTILITIES
0335  81CB                 ;
0336  81CB  C9 20        ADVCK  CMP #$20         ;SPACE?
0337  81CD  F0 02              BEQ M1
0338  81CF  C9 3E              CMP #'>         ;FWD ARROW?
0339  81D1  38          M1    SEC
0340  81D2  60              RTS
0341  81D3  20 FA 82     OBCMIN JSR OUTBYT       ;OUT BYTE, OUT COMMA, IN BYTE
0342  81D6  20 3A 83     COMINB JSR COMMA        ;OUT COMMA, IN BYTE
0343  81D9  20 1B 8A     INBYTE JSR INCHR
0344  81DC  20 75 82            JSR ASCNIB
0345  81DF  B0 14              BCS OUT4
0346  81E1  0A                 ASL A
0347  81E2  0A                 ASL A
0348  81E3  0A                 ASL A
0349  81E4  0A                 ASL A
0350  81E5  8D 33 A6           STA SCR3
0351  81E8  20 1B 8A           JSR INCHR
0352  81EB  20 75 82           JSR ASCNIB
0353  81EE  B0 11              BCS OUT2
0354  81F0  0D 33 A6           ORA SCR3
0355  81F3  18          GOOD  CLC
0356  81F4  60                RTS
0357  81F5  C9 3A        OUT4  CMP #':           ;COLON ?
0358  81F7  D0 05              BNE OUT1
0359  81F9  20 1B 8A           JSR INCHR
0360  81FC  D0 F5              BNE GOOD          ;CARRIAGE RETURN?
0361  81FE  B8          OUT1  CLV
0362  81FF  50 03              BVC CRCHK
0363  8201  2C 04 82     OUT2  BIT CRCHK
0364  8204  C9 0D        CRCHK CMP #$0D          ;CHECK FOR C/R
0365  8206  38                SEC
0366  8207  60                RTS
0367  8208  A2 10        PSHOVE LDX #$10         ;PUSH PARMS DOWN
0368  820A  0E 4A A6     PRM10 ASL P3L
0369  820D  2E 4B A6           ROL P3H
0370  8210  2E 4C A6           ROL P2L
0371  8213  2E 4D A6           ROL P2H
0372  8216  2E 4E A6           ROL P1L
0373  8219  2E 4F A6           ROL P1H
0374  821C  CA                 DEX
0375  821D  D0 EB              BNE PRM10
0376  821F  60                RTS
0377  8220  20 88 81     PARM  JSR SAVER         ;GET PARMS - RETURN ON C/R OR ERR
0378  8223  A9 00              LDA #0
0379  8225  8D 49 A6           STA PARNR
0380  8228  8D 33 A6           STA SCR3
0381  822B  20 08 82     PM1   JSR PSHOVE
0382  822E  20 1B 8A     PARFIL JSR INCHR
0383  8231  C9 2C              CMP #',         ;VALID DELIMITERS - ,
0384  8233  F0 04              BEQ M21
```

```
LINE #  LOC     CODE        LINE

0385   8235  C9 2D           CMP #'-
0386   8237  D0 11           BNE M22
0387   8239  A2 FF     M21   LDX #$FF
0388   823B  8E 33 A6         STX SCR3
0389   823E  EE 49 A6         INC PARNR
0390   8241  AE 49 A6         LDX PARNR
0391   8244  E0 03           CPX #$03
0392   8246  D0 E3           BNE PM1
0393   8248  F0 1D           BEQ M24
0394   824A  20 75 82  M22    JSR ASCNIB
0395   824D  B0 18           BCS M24
0396   824F  A2 04           LDX #4
0397   8251  0E 4A A6  M23    ASL P3L
0398   8254  2E 4B A6         ROL P3H
0399   8257  CA              DEX
0400   8258  D0 F7           BNE M23
0401   825A  0D 4A A6         ORA P3L
0402   825D  8D 4A A6         STA P3L
0403   8260  A9 FF           LDA #$FF
0404   8262  8D 33 A6         STA SCR3
0405   8265  D0 C7           BNE PARFIL
0406   8267  2C 33 A6  M24    BIT SCR3
0407   826A  F0 03           BEQ M25
0408   826C  EE 49 A6         INC PARNR
0409   826F  C9 0D     M25    CMP #$0D
0410   8271  18              CLC
0411   8272  4C B8 81         JMP RESXAF
0412   8275  C9 0D     ASCNIB CMP #$0D        ;C/R?
0413   8277  F0 19           BEQ M29
0414   8279  C9 30           CMP #'0
0415   827B  90 0C           BCC M26
0416   827D  C9 47           CMP #'G
0417   827F  B0 08           BCS M26
0418   8281  C9 41           CMP #'A
0419   8283  B0 08           BCS M27
0420   8285  C9 3A           CMP #':
0421   8287  90 06           BCC M28
0422   8289  C9 30     M26   CMP #'0
0423   828B  38              SEC             ;CARRY SET - NON HEX
0424   828C  60              RTS
0425   828D  E9 37     M27   SBC #$37
0426   828F  29 0F     M28   AND #$0F
0427   8291  18              CLC
0428   8292  60        M29   RTS
0429   8293  EE 4A A6  INCP3  INC P3L          ;INCREMENT P3 (16 BITS)
0430   8296  D0 03           BNE *+5
0431   8298  EE 4B A6         INC P3H
0432   829B  60              RTS
0433   829C  AE 4D A6  P2SCR  LDX P2H          ;MOVE P2 TO FE,FF
0434   829F  86 FF           STX $FF
0435   82A1  AE 4C A6         LDX P2L
0436   82A4  86 FE           STX $FE
0437   82A6  60              RTS
0438   82A7  AE 4B A6  P3SCR  LDX P3H          ;MOVE P3 TO FE,FF
0439   82AA  86 FF           STX $FF
```

```
LINE # LOC      CODE        LINE

0440   82AC   AE 4A A6            LDX P3L
0441   82AF   86 FE               STX $FE
0442   82B1   60                  RTS
0443   82B2   E6 FE        INCCMP INC $FE          ;INCREM FE,FF, COMPARE TO P3
0444   82B4   D0 14               BNE COMPAR
0445   82B6   E6 FF               INC $FF
0446   82B8   D0 10        WRAP   BNE COMPAR       ;TEST FOR WRAP AROUND
0447   82BA   2C BD 82            BIT EXWRAP
0448   82BD   60          EXWRAP RTS
0449   82BE   A5 FE        DECCMP LDA $FE          ;DECREM FE,FF AND COMPARE TO P3
0450   82C0   D0 06               BNE M32
0451   82C2   A5 FF               LDA $FF
0452   82C4   F0 F2               BEQ WRAP
0453   82C6   C6 FF               DEC $FF
0454   82C8   C6 FE        M32    DEC $FE
0455   82CA   20 88 81     COMPAR JSR SAVER        ;COMPARE FE,FF TO P3
0456   82CD   A5 FF               LDA $FF
0457   82CF   CD 4B A6            CMP P3H
0458   82D2   D0 05               BNE EXITCP
0459   82D4   A5 FE               LDA $FE
0460   82D6   CD 4A A6            CMP P3L
0461   82D9   B8          EXITCP CLV
0462   82DA   4C BE 81            JMP RESXF
0463   82DD   08          CHKSAD PHP              ;16 BIT CKSUM IN SCR6,7
0464   82DE   48                  PHA
0465   82DF   18                  CLC
0466   82E0   6D 36 A6            ADC SCR6
0467   82E3   8D 36 A6            STA SCR6
0468   82E6   90 03               BCC M33
0469   82E8   EE 37 A6            INC SCR7
0470   82EB   68          M33    PLA
0471   82EC   28                  PLP
0472   82ED   60                  RTS
0473   82EE   AD 59 A6     OUTPC  LDA PCLR         ;OUTPUT PC
0474   82F1   AE 5A A6            LDX PCHR
0475   82F4   48          OUTXAH PHA
0476   82F5   8A                  TXA
0477   82F6   20 FA 82            JSR OUTBYT
0478   82F9   68                  PLA
0479   82FA   48          OUTBYT PHA              ;OUTPUT 2 HEX DIGS FROM A
0480   82FB   48                  PHA
0481   82FC   4A                  LSR A
0482   82FD   4A                  LSR A
0483   82FE   4A                  LSR A
0484   82FF   4A                  LSR A
0485   8300   20 44 8A            JSR NBASOC
0486   8303   68                  PLA
0487   8304   20 44 8A            JSR NBASOC
0488   8307   68                  PLA
0489   8308   60                  RTS
0490   8309   29 0F        NIBASC AND #$0F          ;NIBBLE IN A TO ASCII IN A
0491   830B   C9 0A               CMP #$0A        ;LINE FEED
0492   830D   B0 04               BCS NIBALF
0493   830F   69 30               ADC #$30
0494   8311   90 02               BCC EXITNB
```

```
LINE # LOC      CODE        LINE

0495  8313  69 36       NIBALF ADC #$36
0496  8315  60          EXITNB RTS
0497  8316  20 4D 83    CRLFSZ JSR CRLF          ;PRINT CRLF, FF, FE
0498  8319  A6 FF       OUTSZ  LDX $FF
0499  831B  A5 FE              LDA $FE
0500  831D  4C F4 82           JMP OUTXAH
0501  8320  A9 3F       OUTQM  LDA #'?
0502  8322  4C 47 8A           JMP OUTCHR
0503  8325  20 3A 83    OCMCK  JSR COMMA         ;OUT COMMA, CKSUM LO
0504  8328  AD 36 A6           LDA SCR6
0505  832B  4C FA 82           JMP OUTBYT
0506  832E  A9 00       ZERCK  LDA #0            ;INIT CHECKSUM
0507  8330  8D 36 A6           STA SCR6
0508  8333  8D 37 A6           STA SCR7
0509  8336  60                RTS
0510  8337  20 EE 82    OPCCOM JSR OUTPC         ;PC OUT, COMMA OUT
0511  833A  48          COMMA  PHA               ;COMMA OUT
0512  833B  A9 2C              LDA #',
0513  833D  D0 06              BNE SPCP3
0514  833F  20 42 83    SPC2   JSR SPACE         ;2 SPACES OUT
0515  8342  48          SPACE  PHA               ;1 SPACE OUT
0516  8343  A9 20              LDA #$20       ;SPACE
0517  8345  20 47 8A    SPCP3  JSR OUTCHR
0518  8348  68                PLA
0519  8349  60                RTS
0520  834A  20 FA 82    OBCRLF JSR OUTBYT        ;BYTE OUT, CRLF OUT
0521  834D  48          CRLF   PHA
0522  834E  A9 0D              LDA #$0D
0523  8350  20 47 8A           JSR OUTCHR
0524  8353  A9 0A              LDA #$0A       ;LINE FEED
0525  8355  20 47 8A           JSR OUTCHR
0526  8358  68                PLA
0527  8359  60                RTS
0528  835A  AE 56 A6    DELAY  LDX TV            ;DELAY DEPENDS ON TV
0529  835D  20 88 81    DL1    JSR SAVER
0530  8360  A9 FF              LDA #$FF
0531  8362  8D 39 A6           STA SCR9
0532  8365  8D 38 A6           STA SCR8
0533  8368  0E 38 A6    DLY1   ASL SCR8          ;(SCR9,8)=FFFF-2**X
0534  836B  2E 39 A6           ROL SCR9
0535  836E  CA                DEX
0536  836F  D0 F7              BNE DLY1
0537  8371  20 03 89    DLY2   JSR IJSCNV        ;SCAN DISPLAY
0538  8374  20 86 83           JSR INSTAT       ;SEE IF KEY DOWN
0539  8377  B0 0A              BCS DLYO
0540  8379  EE 38 A6           INC SCR8         ;SCAN 2**X+1 TIMES
0541  837C  D0 03              BNE *+5
0542  837E  EE 39 A6           INC SCR9
0543  8381  D0 EE              BNE DLY2
0544  8383  4C BE 81    DLYO   JMP RESXF
0545  8386                    ; INSTAT - SEE IF KEY DOWN, RESULT IN CARRY
0546  8386                    ; KYSTAT, TSTAT RETURN IMMEDIATELY W/STATUS
0547  8386                    ; INSTAT WAITS FOR RELEASE
0548  8386  20 92 83    INSTAT JSR INJISV
0549  8389  90 06              BCC INST2
```

```
LINE #  LOC     CODE        LINE

0550   838B  20 92 83    INST1  JSR INJISV
0551   838E  B0 FB               BCS INST1
0552   8390  38                 SEC
0553   8391  60          INST2  RTS
0554   8392  6C 67 A6    INJISV JMP (INSVEC+1)
0555   8395              ;
0556   8395              ;
0557   8395              ; *** EXECUTE BLOCKS BEGIN HERE
0558   8395              ;
0559   8395              BZPARM=*
0560   8395              ; ZERO PARM COMMANDS
0561   8395              ;
0562   8395  C9 52       REGZ   CMP #'R          ;DISP REGISTERS
0563   8397  D0 5A               BNE GOZ        ;PC,S,F,A,X,Y
0564   8399  20 4D 83    RGBACK JSR CRLF
0565   839C  A9 50               LDA #'P
0566   839E  20 47 8A            JSR OUTCHR
0567   83A1  20 42 83            JSR SPACE
0568   83A4  20 EE 82            JSR OUTPC
0569   83A7  20 D6 81            JSR COMINB
0570   83AA  B0 13               BCS NH3
0571   83AC  8D 34 A6            STA SCR4
0572   83AF  20 D9 81            JSR INBYTE
0573   83B2  B0 0B               BCS NH3
0574   83B4  8D 59 A6            STA PCLR
0575   83B7  AD 34 A6            LDA SCR4
0576   83BA  8D 5A A6            STA PCHR
0577   83BD  90 09               BCC M34
0578   83BF  D0 02       NH3     BNE NOTCR
0579   83C1  18          EXITRG CLC
0580   83C2  60          EXRGP1 RTS
0581   83C3  20 CB 81    NOTCR  JSR ADVCK
0582   83C6  D0 FA               BNE EXRGP1
0583   83C8  A0 00       M34     LDY #0
0584   83CA  C8          M35     INY
0585   83CB  C0 06               CPY #6
0586   83CD  F0 CA               BEQ RGBACK
0587   83CF  20 4D 83            JSR CRLF
0588   83D2  B9 99 8F    NXTRG  LDA RGNAM-1,Y   ;GET REG NAME
0589   83D5              ; OUTPUT 3 SPACES TO LINE UP DISPLAY
0590   83D5  20 47 8A            JSR OUTCHR
0591   83D8  20 42 83            JSR SPACE
0592   83DB  20 3F 83            JSR SPC2
0593   83DE  B9 5A A6            LDA PCHR,Y
0594   83E1  20 D3 81            JSR OBCMIN
0595   83E4  B0 05               BCS M36
0596   83E6  99 5A A6            STA PCHR,Y
0597   83E9  90 DF               BCC M35
0598   83EB  F0 D4       M36     BEQ EXITRG
0599   83ED  20 CB 81            JSR ADVCK
0600   83F0  F0 D8               BEQ M35
0601   83F2  60                  RTS
0602   83F3  C9 47       GOZ     CMP #'G
0603   83F5  D0 20               BNE LPZB
0604   83F7  20 4D 83    GO2     JSR CRLF
```

```
LINE # LOC      CODE        LINE

0605  83FA  20 9C 8B    G01ENT JSR NACCES       ;WRITE PROT MONITOR RAM
0606  83FD  AE 5B A6           LDX SR           ;RESTORE REGS
0607  8400  9A                 TXS
0608  8401  AD 5A A6           LDA PCHR
0609  8404  48                 PHA
0610  8405  AD 59 A6           LDA PCLR
0611  8408  48           NR10  PHA
0612  8409  AD 5C A6           LDA FR
0613  840C  48                 PHA
0614  840D  AC 5F A6           LDY YR
0615  8410  AE 5E A6           LDX XR
0616  8413  AD 5D A6           LDA AR
0617  8416  40                 RTI
0618  8417  C9 11       LPZB  CMP #$11          ;LOAD PAPER TAPE
0619  8419  F0 03              BEQ *+5
0620  841B  4C A7 84           JMP DEFZ
0621  841E  20 88 81           JSR SAVER
0622  8421  20 4D 83           JSR CRLF
0623  8424  A9 00              LDA #0
0624  8426  8D 52 A6           STA ERCNT
0625  8429  20 2E 83     LPZ   JSR ZERCK
0626  842C  20 1B 8A     LP1   JSR INCHR
0627  842F  C9 3B              CMP #$3B          ;SEMI COLON
0628  8431  D0 F9              BNE LP1
0629  8433  20 A1 84           JSR LDBYTE
0630  8436  B0 56              BCS TAPERR
0631  8438  D0 09              BNE NUREC
0632  843A  AD 52 A6           LDA ERCNT         ;ERRORS ?
0633  843D  F0 01              BEQ *+3
0634  843F  38           EXITLP SEC
0635  8440  4C B8 81           JMP RESXAF
0636  8443  8D 3D A6     NUREC STA RC
0637  8446  20 A1 84           JSR LDBYTE
0638  8449  B0 43              BCS TAPERR
0639  844B  85 FF              STA $FF
0640  844D  20 A1 84           JSR LDBYTE
0641  8450  B0 D7              BCS LPZ
0642  8452  85 FE              STA $FE
0643  8454  20 A1 84     MORED JSR LDBYTE
0644  8457  B0 35              BCS TAPERR
0645  8459  A0 00              LDY #0
0646  845B  91 FE              STA ($FE),Y
0647  845D  D1 FE              CMP ($FE),Y
0648  845F  F0 0C              BEQ LPGD
0649  8461  AD 52 A6           LDA ERCNT
0650  8464  29 0F              AND #$0F
0651  8466  C9 0F              CMP #$0F
0652  8468  F0 03              BEQ *+5
0653  846A  EE 52 A6           INC ERCNT
0654  846D  20 B2 82     LPGD  JSR INCCMP
0655  8470  CE 3D A6           DEC RC
0656  8473  D0 DF              BNE MORED
0657  8475  20 D9 81           JSR INBYTE
0658  8478  B0 14              BCS TAPERR
0659  847A  CD 37 A6           CMP SCR7
```

```
LINE # LOC     CODE        LINE

0660  847D  D0 0C           BNE BADDY
0661  847F  20 D9 81        JSR INBYTE
0662  8482  B0 0A           BCS TAPERR
0663  8484  CD 36 A6        CMP SCR6
0664  8487  F0 A0           BEQ LPZ
0665  8489  D0 03           BNE TAPERR      ;(ALWAYS)
0666  848B  20 D9 81  BADDY  JSR INBYTE
0667  848E  AD 52 A6  TAPERR LDA ERCNT
0668  8491  29 F0           AND #$F0
0669  8493  C9 F0           CMP #$F0
0670  8495  F0 92           BEQ LPZ
0671  8497  AD 52 A6        LDA ERCNT
0672  849A  69 10           ADC #$10
0673  849C  8D 52 A6        STA ERCNT
0674  849F  D0 88           BNE LPZ
0675  84A1  20 D9 81  LDBYTE JSR INBYTE
0676  84A4  4C DD 82        JMP CHKSAD
0677  84A7  C9 44     DEPZ   CMP #'D         ;DEPOSIT, 0 PARM - USE (OLD)
0678  84A9  D0 03           BNE MEMZ
0679  84AB  4C E1 84        JMP NEWLN
0680  84AE  C9 4D     MEMZ   CMP #'M         ;MEM, 0 PARM - USE (OLD)
0681  84B0  D0 03           BNE VERZ
0682  84B2  4C 17 85        JMP NEWLOC
0683  84B5  C9 56     VERZ   CMP #'V         ;VERIFY, 0 PARM - USE (OLD)
0684  84B7  D0 0D           BNE L1ZB        ; ... DO 8 BYTES (LIKE VER 1 PARM)
0685  84B9  A5 FE           LDA $FE
0686  84BB  8D 4A A6        STA P3L
0687  84BE  A5 FF           LDA $FF
0688  84C0  8D 4B A6        STA P3H
0689  84C3  4C 9A 85        JMP VER1+4
0690  84C6  C9 12     L1ZB   CMP #$12        ;LOAD KIM, ZERO PARM
0691  84C8  D0 05           BNE L2ZB
0692  84CA  A0 00           LDY #0          ;MODE = KIM
0693  84CC  4C 78 8C  L1J    JMP LENTRY      ;GO TO CASSETTE ROUTINE
0694  84CF  C9 13     L2ZB   CMP #$13        ;LOAD HS, ZERO PARM
0695  84D1  D0 04           BNE EZPARM
0696  84D3  A0 80           LDY #$80        ;MODE = HS
0697  84D5  D0 F5           BNE L1J         ;(ALWAYS)
0698  84D7  6C 6D A6  EZPARM JMP (URCVEC+1) ;ELSE UNREC COMMAND
0699  84DA              B1PARM=*
0700  84DA              ;
0701  84DA              ; 1 PARAMETER COMMAND EXEC BLOCKS
0702  84DA              ;
0703  84DA  C9 44     DEP1   CMP #'D         ;DEPOSIT, 1 PARM
0704  84DC  D0 32           BNE MEM1
0705  84DE  20 A7 82        JSR P3SCR
0706  84E1  20 16 83  NEWLN  JSR CRLFSZ
0707  84E4  A0 00           LDY #0
0708  84E6  A2 08           LDX #8
0709  84E8  20 42 83  DEPBYT JSR SPACE
0710  84EB  20 D9 81        JSR INBYTE
0711  84EE  B0 11           BCS NH41
0712  84F0  91 FE           STA ($FE),Y
0713  84F2  D1 FE           CMP ($FE),Y     ;VERIFY
0714  84F4  F0 03           BEQ DEPN
```

```
LINE # LOC     CODE       LINE

0715  84F6  20 20 83          JSR OUTQM       ;TYPE "?" IF NG
0716  84F9  20 B2 82    DEPN  JSR INCCMP
0717  84FC  CA               DEX
0718  84FD  D0 E9            BNE DEPBYT
0719  84FF  F0 E0            BEQ NEWLN
0720  8501  F0 0B       NH41  BEQ DEPEC
0721  8503  C9 20            CMP #$20        ;SPACE = FWD
0722  8505  D0 4C            BNE DEPES
0723  8507  70 F0            BVS DEPN
0724  8509  20 42 83          JSR SPACE
0725  850C  10 EB            BPL DEPN
0726  850E  18          DEPEC CLC
0727  850F  60               RTS
0728  8510  C9 4D       MEM1  CMP #'M         ;MEMORY, 1 PARM
0729  8512  D0 65            BNE G01
0730  8514  20 A7 82          JSR P3SCR
0731  8517  20 16 83    NEWLOC JSR CRLFSZ
0732  851A  20 3A 83          JSR COMMA
0733  851D  A0 00             LDY #0
0734  851F  B1 FE             LDA ($FE),Y
0735  8521  20 D3 81          JSR OBCMIN
0736  8524  B0 11            BCS NH42
0737  8526  A0 00             LDY #0
0738  8528  91 FE             STA ($FE),Y
0739  852A  D1 FE             CMP ($FE),Y     ;VERIFY MEM
0740  852C  F0 03            BEQ NXTLOC
0741  852E  20 20 83          JSR OUTQM       ;TYPE ? AND CONTINUE
0742  8531  20 B2 82    NXTLOC JSR INCCMP
0743  8534  18               CLC
0744  8535  90 E0            BCC NEWLOC
0745  8537  F0 3E       NH42  BEQ EXITM1
0746  8539  50 04            BVC *+6
0747  853B  C9 3C            CMP #'<
0748  853D  F0 D8            BEQ NEWLOC
0749  853F  C9 20            CMP #$20      ;SPACE ?
0750  8541  F0 EE            BEQ NXTLOC
0751  8543  C9 3E            CMP #'>
0752  8545  F0 EA            BEQ NXTLOC
0753  8547  C9 2B            CMP #'+
0754  8549  F0 10            BEQ LOCP8
0755  854B  C9 3C            CMP #'<
0756  854D  F0 06            BEQ PRVLOC
0757  854F  C9 2D            CMP #'-
0758  8551  F0 16            BEQ LOCM8
0759  8553  38          DEPES SEC
0760  8554  60               RTS
0761  8555  20 BE 82    PRVLOC JSR DECCMP       ;BACK ONE BYT
0762  8558  18               CLC
0763  8559  90 BC            BCC NEWLOC
0764  855B  A5 FE       LOCP8 LDA $FE           ;GO FWD 8 BYTES
0765  855D  18               CLC
0766  855E  69 08            ADC #$08
0767  8560  85 FE            STA $FE
0768  8562  90 02            BCC M42
0769  8564  E6 FF            INC $FF
```

```
LINE # LOC     CODE        LINE

0770  8566  18           M42     CLC
0771  8567  90 AE                BCC NEWLOC
0772  8569  A5 FE        LOCM8   LDA $FE          ; GO BACKWD 8 BYTES
0773  856B  38                   SEC
0774  856C  E9 08                SBC #$08
0775  856E  85 FE                STA $FE
0776  8570  B0 02                BCS M43
0777  8572  C6 FF                DEC $FF
0778  8574  18           M43     CLC
0779  8575  90 A0                BCC NEWLOC
0780  8577  18           EXITM1  CLC
0781  8578  60                   RTS
0782  8579  C9 47        G01     CMP #'G          ; GO, 1 PARM (RTRN ADDR ON STK)
0783  857B  D0 19                BNE VER1       ; ...  PARM IS ADDR TO GO TO
0784  857D  20 4D 83             JSR CRLF
0785  8580  20 9C 8B             JSR NACCES       ;WRITE PROT MONITR RAM
0786  8583  A2 FF                LDX #$FF         ;PUSH RETURN ADDR
0787  8585  9A                   TXS
0788  8586  A9 7F                LDA #$7F
0789  8588  48                   PHA
0790  8589  A9 FF                LDA #$FF
0791  858B  48                   PHA
0792  858C  AD 4B A6             LDA P3H
0793  858F  48                   PHA
0794  8590  AD 4A A6             LDA P3L
0795  8593  4C 08 84             JMP NR10
0796  8596  C9 56        VER1    CMP #'V          ;VERIFY, 1 PARM (8 BYTES, CKSUM)
0797  8598  D0 1A                BNE JUMP1
0798  859A  AD 4A A6             LDA P3L
0799  859D  8D 4C A6             STA P2L
0800  85A0  18                   CLC
0801  85A1  69 07                ADC #$07
0802  85A3  8D 4A A6             STA P3L
0803  85A6  AD 4B A6             LDA P3H
0804  85A9  8D 4D A6             STA P2H
0805  85AC  69 00                ADC #0
0806  85AE  8D 4B A6             STA P3H
0807  85B1  4C 40 86             JMP VER2+4
0808  85B4  C9 4A        JUMP1   CMP #'J          ;JUMP (JUMP TABLE IN SYS RAM)
0809  85B6  D0 1F                BNE L11B
0810  85B8  AD 4A A6             LDA P3L
0811  85BB  C9 08                CMP #8           ; 0-7 ONLY VALID
0812  85BD  B0 26                BCS JUM2
0813  85BF  20 9C 8B             JSR NACCES       ;WRITE PROT SYS RAM
0814  85C2  0A                   ASL A
0815  85C3  A8                   TAY
0816  85C4  A2 FF                LDX #$FF         ;INIT STK PTR
0817  85C6  9A                   TXS
0818  85C7  A9 7F                LDA #$7F         ;PUSH COLD RETURN
0819  85C9  48                   PHA
0820  85CA  A9 FF                LDA #$FF
0821  85CC  48                   PHA
0822  85CD  B9 21 A6             LDA JTABLE+1,Y   ;GET ADDR FROM TABLE
0823  85D0  48                   PHA              ;PUSH ON STACK
0824  85D1  B9 20 A6             LDA JTABLE,Y
```

```
LINE # LOC     CODE        LINE

0825  85D4  4C 08 84          JMP NR10          ;LOAD UP USER REG'S AND RTI
0826  85D7  C9 12       L11B   CMP #$12          ;LOAD KIM FMT, 1 PARM
0827  85D9  D0 14              BNE L21B
0828  85DB  A0 00              LDY #0            ;MODE = KIM
0829  85DD  AD 4A A6     L11C   LDA P3L
0830  85E0  C9 FF              CMP #$FF          ; ID MUST NOT BE FF
0831  85E2  D0 02              BNE *+4
0832  85E4  38                 SEC
0833  85E5  60           JUM2   RTS
0834  85E6  20 08 82          JSR PSHOVE        ;FIX PARM POSITION
0835  85E9  20 08 82     L11D   JSR PSHOVE
0836  85EC  4C 78 8C          JMP LENTRY
0837  85EF  C9 13       L21B   CMP #$13          ;LOAD TAPE, HS FMT, 1 PARM
0838  85F1  D0 04              BNE WPR1B
0839  85F3  A0 80              LDY #$80          ;MODE = HS
0840  85F5  D0 E6              BNE L11C
0841  85F7  C9 57       WPR1B  CMP #'W           ;WRITE PROT USER RAM
0842  85F9  D0 1B              BNE E1PARM
0843  85FB  AD 4A A6          LDA P3L           ; FIRST DIG IS 1K ABOVE 0,
0844  85FE  29 11              AND #$11          ; SECOND IS 2K ABOVE 0
0845  8600  C9 08              CMP #8            ; THIRD IS 3K ABOVE 0.
0846  8602  2A                 ROL A
0847  8603  4E 4B A6          LSR P3H
0848  8606  2A                 ROL A
0849  8607  0A                 ASL A
0850  8608  29 0F              AND #$0F
0851  860A  49 0F              EOR #$0F          ;0 IS PROTECT
0852  860C  8D 01 AC          STA OR3A
0853  860F  A9 0F              LDA #$0F
0854  8611  8D 03 AC          STA DDR3A
0855  8614  18                 CLC
0856  8615  60                 RTS
0857  8616  4C 27 88     E1PARM JMP CALC3
0858  8619                B2PARM=*
0859  8619                ;
0860  8619                ; 2 PARAMETER EXEC BLOCKS
0861  8619                ;
0862  8619  C9 10       STD2   CMP #$10          ;STORE DOUBLE BYTE
0863  861B  D0 12              BNE MEM2
0864  861D  20 A7 82          JSR P3SCR
0865  8620  AD 4D A6          LDA P2H
0866  8623  A0 01              LDY #1
0867  8625  91 FE              STA ($FE),Y
0868  8627  88                 DEY
0869  8628  AD 4C A6          LDA P2L
0870  862B  91 FE              STA ($FE),Y
0871  862D  18                 CLC
0872  862E  60                 RTS
0873  862F  C9 4D       MEM2   CMP #'M           ;CONTINUE MEM SEARCH W/OLD PTR
0874  8631  D0 09              BNE VER2
0875  8633  AD 4C A6          LDA P2L
0876  8636  8D 4E A6          STA P1L
0877  8639  4C 08 88          JMP MEM3C
0878  863C  C9 56       VER2   CMP #'V           ;VERIFY MEM W/CHKSUMS , 2 PARM
0879  863E  D0 48              BNE L12B
```

```
LINE # LOC      CODE        LINE

0880   8640   20 9C 82            JSR P2SCR
0881   8643   20 2E 83            JSR ZERCK
0882   8646   20 16 83    VADDR   JSR CRLFSZ
0883   8649   A2 08               LDX #8
0884   864B   20 42 83    V2      JSR SPACE
0885   864E   A0 00               LDY #0
0886   8650   B1 FE               LDA ($FE),Y
0887   8652   20 DD 82            JSR CHKSAD
0888   8655   20 FA 82            JSR OUTBYT
0889   8658   20 B2 82            JSR INCCMP
0890   865B   70 11               BVS V1
0891   865D   F0 02               BEQ *+4
0892   865F   B0 0D               BCS V1
0893   8661   CA                  DEX
0894   8662   D0 E7               BNE V2
0895   8664   20 25 83    VOCK    JSR OCMCK
0896   8667   20 86 83            JSR INSTAT
0897   866A   90 DA               BCC VADDR
0898   866C   18                  CLC
0899   866D   60                  RTS
0900   866E   20 BE 82    V1      JSR DECCMP
0901   8671   E0 08               CPX #8
0902   8673   F0 03               BEQ *+5
0903   8675   E8                  INX
0904   8676   10 F6               BPL V1
0905   8678   20 25 83            JSR OCMCK
0906   867B   20 4D 83            JSR CRLF
0907   867E   20 42 83            JSR SPACE
0908   8681   AE 37 A6            LDX SCR7
0909   8684   20 F4 82            JSR OUTXAH
0910   8687   60                  RTS
0911   8688   C9 12       L12B    CMP #$12        ;LOAD KIM FMT TAPE, 2 PARMS
0912   868A   D0 0C               BNE SP2B
0913   868C   AD 4C A6    L12C    LDA P2L
0914   868F   C9 FF               CMP #$FF        ;ID MUST BE FF
0915   8691   D0 F4               BNE L12B-1      ;ERR
0916   8693   A0 00               LDY #0          ;MODE = HS
0917   8695   4C E9 85            JMP L11D
0918   8698   C9 1C       SP2B    CMP #$1C        ;SAVE PAPER TAPE, 2 PARMS
0919   869A   D0 75               BNE E2PARM
0920   869C   18                  CLC
0921   869D   20 88 81            JSR SAVER
0922   86A0   20 9C 82            JSR P2SCR
0923   86A3   20 FA 86    SP2C    JSR DIFFZ
0924   86A6   B0 03               BCS SP2D
0925   86A8   4C C4 81    SPEXIT  JMP RESALL
0926   86AB   20 4D 83    SP2D    JSR CRLF
0927   86AE   CD 58 A6            CMP MAXRC
0928   86B1   90 05               BCC SP2E
0929   86B3   AD 58 A6            LDA MAXRC
0930   86B6   B0 02               BCS SP2F
0931   86B8   69 01       SP2E    ADC #1
0932   86BA   8D 3D A6    SP2F    STA RC
0933   86BD   A9 3B               LDA #$3B        ;SEMI COLON
0934   86BF   20 47 8A            JSR OUTCHR
```

```
LINE #  LOC    CODE        LINE

0935  86C2  AD 3D A6           LDA RC
0936  86C5  20 F4 86           JSR SVBYTE
0937  86C8  A5 FF              LDA $FF
0938  86CA  20 F4 86           JSR SVBYTE
0939  86CD  A5 FE              LDA $FE
0940  86CF  20 F4 86           JSR SVBYTE
0941  86D2  A0 00        MORED2 LDY #$00
0942  86D4  B1 FE              LDA ($FE),Y
0943  86D6  20 F4 86           JSR SVBYTE
0944  86D9  20 86 83           JSR INSTAT       ;STOP IF KEY DEPRESSED
0945  86DC  B0 CA              BCS SPEXIT
0946  86DE  20 B2 82           JSR INCCMP
0947  86E1  70 C5              BVS SPEXIT
0948  86E3  CE 3D A6           DEC RC
0949  86E6  D0 EA              BNE MORED2
0950  86E8  AE 37 A6           LDX SCR7
0951  86EB  AD 36 A6           LDA SCR6
0952  86EE  20 F4 82           JSR OUTXAH
0953  86F1  18                 CLC
0954  86F2  90 AF              BCC SP2C
0955  86F4  20 DD 82    SVBYTE JSR CHKSAD
0956  86F7  4C FA 82           JMP OUTBYT
0957  86FA  20 2E 83    DIFFZ  JSR ZERCK
0958  86FD  AD 4A A6    DIFFL  LDA P3L
0959  8700  38                 SEC
0960  8701  E5 FE              SBC $FE
0961  8703  48                 PHA
0962  8704  AD 4B A6           LDA P3H
0963  8707  E5 FF              SBC $FF
0964  8709  F0 04              BEQ DIFF1
0965  870B  68                 PLA
0966  870C  A9 FF              LDA #$FF
0967  870E  60                 RTS
0968  870F  68           DIFF1  PLA
0969  8710  60           DIFFL2 RTS
0970  8711  4C 27 88    E2PARM JMP CALC3        ;MAY BE CALC OR EXEC
0971  8714               B3PARM=*
0972  8714               ;
0973  8714               ; 3 PARAMETER COMMAND EXECUTE BLOCKS
0974  8714               ;
0975  8714  C9 46        FILL3  CMP #'F          ;FILL MEM
0976  8716  D0 21              BNE BLK3
0977  8718  20 9C 82           JSR P2SCR
0978  871B  A9 00              LDA #0
0979  871D  8D 52 A6           STA ERCNT        ;ZERO ERROR COUNT
0980  8720  AD 4E A6           LDA P1L
0981  8723  A0 00        F1     LDY #0
0982  8725  91 FE              STA ($FE),Y
0983  8727  D1 FE              CMP ($FE),Y      ;VERIFY
0984  8729  F0 03              BEQ F3
0985  872B  20 C1 87           JSR BRTT         ;INC ERCNT (UP TO FF)
0986  872E  20 B2 82    F3     JSR INCCMP
0987  8731  70 7C              BVS B1
0988  8733  F0 EE              BEQ F1
0989  8735  90 EC              BCC F1
```

```
LINE # LOC      CODE        LINE

0990  8737  B0 76        F2      BCS B1          ;(ALWAYS)
0991  8739  C9 42        BLK3    CMP #'B         ;BLOCK MOVE (OVERLAP OK)
0992  873B  F0 03                BEQ *+5
0993  873D  4C CD 87              JMP S13B
0994  8740  A9 00                LDA #0
0995  8742  8D 52 A6              STA ERCNT
0996  8745  20 9C 82              JSR P2SCR
0997  8748  AD 4E A6              LDA P1L
0998  874B  85 FC                STA $FC
0999  874D  AD 4F A6              LDA P1H
1000  8750  85 FD                STA $FD
1001  8752  C5 FF                CMP $FF         ;WHICH DIRECTION TO MOVE?
1002  8754  D0 06                BNE *+8
1003  8756  A5 FC                LDA $FC
1004  8758  C5 FE                CMP $FE
1005  875A  F0 53                BEQ B1          ;16 BITS EQUAL THEN FINISHED
1006  875C  B0 14                BCS B2          ;MOVE DEC'NG
1007  875E  20 B7 87    BLP     JSR BMOVE       ;MOVE INC'NG
1008  8761  E6 FC                INC $FC
1009  8763  D0 02                BNE *+4
1010  8765  E6 FD                INC $FD
1011  8767  20 B2 82              JSR INCCMP
1012  876A  70 43                BVS B1
1013  876C  F0 F0                BEQ BLP
1014  876E  90 EE                BCC BLP
1015  8770  B0 3D                BCS B1
1016  8772  A5 FC        B2      LDA $FC         ;CALC VALS FOR MOVE DEC'NG
1017  8774  18                  CLC
1018  8775  6D 4A A6              ADC P3L
1019  8778  85 FC                STA $FC
1020  877A  A5 FD                LDA $FD
1021  877C  6D 4B A6              ADC P3H
1022  877F  85 FD                STA $FD
1023  8781  38                  SEC
1024  8782  A5 FC                LDA $FC
1025  8784  E5 FE                SBC $FE
1026  8786  85 FC                STA $FC
1027  8788  A5 FD                LDA $FD
1028  878A  E5 FF                SBC $FF
1029  878C  85 FD                STA $FD
1030  878E  20 A7 82              JSR P3SCR
1031  8791  AD 4C A6              LDA P2L
1032  8794  8D 4A A6              STA P3L
1033  8797  AD 4D A6              LDA P2H
1034  879A  8D 4B A6              STA P3H
1035  879D  20 B7 87    BLP1    JSR BMOVE       ;MOVE DEC'NG
1036  87A0  A5 FC                LDA $FC
1037  87A2  D0 02                BNE *+4
1038  87A4  C6 FD                DEC $FD
1039  87A6  C6 FC                DEC $FC
1040  87A8  20 BE 82              JSR DECCMP
1041  87AB  70 02                BVS B1
1042  87AD  B0 EE                BCS BLP1
1043  87AF  AD 52 A6    B1      LDA ERCNT       ;FINISHED, TEST ERCNT
1044  87B2  38                  SEC
```

```
LINE # LOC     CODE       LINE

1045  87B3  D0 01          BNE *+3
1046  87B5  18             CLC
1047  87B6  60             RTS
1048  87B7  A0 00       BMOVE LDY #0          ;MOVE 1 BYT + VER
1049  87B9  B1 FE          LDA ($FE),Y
1050  87BB  91 FC          STA ($FC),Y
1051  87BD  D1 FC          CMP ($FC),Y
1052  87BF  F0 0B          BEQ BRT
1053  87C1  AC 52 A6   BRTT LDY ERCNT       ;INC ERCNT, DONT PASS FF
1054  87C4  C0 FF          CPY #$FF
1055  87C6  F0 04          BEQ *+6
1056  87C8  C8             INY
1057  87C9  8C 52 A6       STY ERCNT
1058  87CC  60         BRT RTS
1059  87CD  C9 1D      S13B CMP #$1D         ;SAVE KIM FMT TAPE, 3 PARMS
1060  87CF  D0 15          BNE S23B
1061  87D1  A0 00          LDY #$0           ;MODE = KIM
1062  87D3  AD 4E A6   S13C LDA P1L
1063  87D6  D0 02          BNE *+4           ;ID MUST NOT = 0
1064  87D8  38             SEC
1065  87D9  60             RTS
1066  87DA  C9 FF          CMP #$FF          ;ID MUST NOT = FF
1067  87DC  D0 02          BNE *+4
1068  87DE  38         S1NG SEC
1069  87DF  60             RTS
1070  87E0  20 93 82       JSR INCP3         ;USE END ADDR + 1
1071  87E3  4C 87 8E       JMP SENTRY
1072  87E6  C9 1E      S23B CMP #$1E         ;SAVE HS FMT TAPE, 3 PARMS
1073  87E8  D0 04          BNE L23P
1074  87EA  A0 80          LDY #$80          ;MODE = HS
1075  87EC  D0 E5          BNE S13C          ;(ALWAYS)
1076  87EE  C9 13      L23P CMP #$13          ;LOAD HS, 3 PARMS
1077  87F0  D0 0F          BNE MEM3
1078  87F2  AD 4E A6       LDA P1L
1079  87F5  C9 FF          CMP #$FF          ;ID MUST BE FF
1080  87F7  D0 E5          BNE S1NG          ;ERR RETURN
1081  87F9  20 93 82       JSR INCP3         ;USE END ADDR + 1
1082  87FC  A0 80          LDY #$80          ;MODE = HS
1083  87FE  4C 78 8C       JMP LENTRY
1084  8801  C9 4D      MEM3 CMP #'M           ;MEM 3 SEARCH - BYTE
1085  8803  D0 22          BNE CALC3
1086  8805  20 9C 82       JSR P2SCR
1087  8808  AD 4E A6   MEM3C LDA P1L
1088  880B  A0 00          LDY #0
1089  880D  D1 FE          CMP ($FE),Y
1090  880F  F0 0B          BEQ MEM3E         ;FOUND SEARCH BYTE?
1091  8811  20 B2 82   MEM3D JSR INCCMP      ;NO, INC BUFFER ADDR
1092  8814  70 04          BVS MEM3EX        ;WRAP AROUND?
1093  8816  F0 F0          BEQ MEM3C
1094  8818  90 EE          BCC MEM3C
1095  881A  18         MEM3EX CLC
1096  881B  60             RTS               ;SEARCHED TO BOUND
1097  881C  20 17 85   MEM3E JSR NEWLOC      ;FOUND SEARCH BYTE
1098  881F  90 05          BCC MEM3F
1099  8821  C9 47          CMP #'G           ;ENTERED G?
```

```
LINE # LOC     CODE       LINE

1100  8823  F0 EC             BEQ MEM3D
1101  8825  38                SEC
1102  8826  60          MEM3F RTS
1103  8827  C9 43       CALC3 CMP #'C CALCULATE, 1, 2 OR 3 PARMS
1104  8829  D0 26             BNE EXE3          ;RESULT = P1+P2-P3
1105  882B  20 4D 83    C1    JSR CRLF
1106  882E  20 42 83          JSR SPACE
1107  8831  18                CLC
1108  8832  AD 4E A6          LDA P1L
1109  8835  6D 4C A6          ADC P2L
1110  8838  A8                TAY
1111  8839  AD 4F A6          LDA P1H
1112  883C  6D 4D A6          ADC P2H
1113  883F  AA                TAX
1114  8840  38                SEC
1115  8841  98                TYA
1116  8842  ED 4A A6          SBC P3L
1117  8845  A8                TAY
1118  8846  8A                TXA
1119  8847  ED 4B A6          SBC P3H
1120  884A  AA                TAX
1121  884B  98                TYA
1122  884C  20 F4 82          JSR OUTXAH
1123  884F  18                CLC
1124  8850  60                RTS
1125  8851  C9 45       EXE3  CMP #'E           ;EXECUTE FROM RAM, 1-3 PARMS
1126  8853  D0 57             BNE E3PARM
1127  8855                ; SEE IF VECTOR ALREADY MOVED
1128  8855  AD 62 A6          LDA INVEC+2       ;INVEC MOVED TO SCRA, SCRB
1129  8858                ; HI BYTE OF EXEVEC MUST BE DIFFERENT FROM INVEC
1130  8858  CD 73 A6          CMP EXEVEC+1      ;$FA, $FB USED AS RAM PTR
1131  885B  F0 15             BEQ PTRIN
1132  885D  8D 3B A6          STA SCRA+1        ;SAVE INVEC IN SCRA,B
1133  8860  AD 61 A6          LDA INVEC+1
1134  8863  8D 3A A6          STA SCRA
1135  8866  AD 72 A6          LDA EXEVEC        ;PUT ADDR OF RIN IN INVEC
1136  8869  8D 61 A6          STA INVEC+1
1137  886C  AD 73 A6          LDA EXEVEC+1
1138  886F  8D 62 A6          STA INVEC+2
1139  8872  AD 4B A6    PTRIN LDA P3H           ;INIT RAM PTR IN $FA, $FB
1140  8875  85 FB             STA $FB
1141  8877  AD 4A A6          LDA P3L
1142  887A  85 FA             STA $FA
1143  887C  18                CLC
1144  887D  60                RTS
1145  887E  20 88 81    RIN   JSR SAVER         ;GET INPUT FROM RAM
1146  8881  A0 00             LDY #$0           ;RAM PTR IN $FA, $FB
1147  8883  B1 FA             LDA ($FA),Y
1148  8885  F0 12             BEQ RESTIV        ;IF 00 BYTE, RESTORE INVEC
1149  8887  E6 FA             INC $FA
1150  8889  D0 02             BNE *+4
1151  888B  E6 FB             INC $FB
1152  888D  2C 53 A6          BIT TECHO         ;ECHO CHARS IN ?
1153  8890  10 03             BPL *+5
1154  8892  20 47 8A          JSR OUTCHR
```

```
LINE #  LOC    CODE        LINE

1155   8895  18              CLC
1156   8896  4C B8 81        JMP RESXAF
1157   8899  AD 3A A6  RESTIV LDA SCRA        ;RESTORE INVEC
1158   889C  8D 61 A6        STA INVEC+1
1159   889F  AD 3B A6        LDA SCRA+1
1160   88A2  8D 62 A6        STA INVEC+2
1161   88A5  18              CLC
1162   88A6  20 1B 8A        JSR INCHR
1163   88A9  4C B8 81        JMP RESXAF
1164   88AC  6C 6D A6  E3PARM JMP (URCVEC+1)  ;... ELSE UNREC CMD
1165   88AF                  ; ***
1166   88AF                  ; *** HEX KEYBOARD I/O
1167   88AF                  ; ***
1168   88AF  20 88 81  GETKEY JSR SAVER       ;FIND KEY
1169   88B2  20 CF 88        JSR GK
1170   88B5  C9 FE           CMP #$FE
1171   88B7  D0 13           BNE EXITGK
1172   88B9  20 CF 88        JSR GK
1173   88BC  8A              TXA
1174   88BD  0A              ASL A
1175   88BE  0A              ASL A
1176   88BF  0A              ASL A
1177   88C0  0A              ASL A
1178   88C1  8D 3E A6        STA SCRE
1179   88C4  20 CF 88        JSR GK
1180   88C7  8A              TXA
1181   88C8  18              CLC
1182   88C9  6D 3E A6        ADC SCRE
1183   88CC  4C B8 81  EXITGK JMP RESXAF
1184   88CF  A9 00     GK     LDA #0
1185   88D1  8D 55 A6        STA KSHFL
1186   88D4  20 03 89  GK1    JSR IJSCNV       ;SCAN KB
1187   88D7  F0 FB           BEQ GK1
1188   88D9  20 2C 89        JSR LRNKEY       ;WHAT KEY IS IT?
1189   88DC  F0 F6           BEQ GK1
1190   88DE  48              PHA
1191   88DF  8A              TXA
1192   88E0  48              PHA
1193   88E1  20 72 89        JSR BEEP
1194   88E4  20 23 89  GK2    JSR KEYQ
1195   88E7  D0 FB           BNE GK2          ;Z=1 IF KEY DOWN
1196   88E9  20 9B 89        JSR NOBEEP       ;DELAY (DEBOUNCE) W/O BEEP
1197   88EC  20 23 89        JSR KEYQ
1198   88EF  D0 F3           BNE GK2
1199   88F1  68              PLA
1200   88F2  AA              TAX
1201   88F3  68              PLA
1202   88F4  C9 FF           CMP #$FF         ;IF SHIFT, SET FLAG + GET NEXT KEY
1203   88F6  D0 07           BNE EXITG
1204   88F8  A9 19           LDA #$19
1205   88FA  8D 55 A6        STA KSHFL
1206   88FD  D0 D5           BNE GK1
1207   88FF  60        EXITG  RTS
1208   8900  20 C1 89  HDOUT  JSR OUTDSP       ;CHAR OUT, SCAN KB
1209   8903  6C 70 A6  IJSCNV JMP (SCNVEC+1)
```

```
LINE #  LOC   CODE        LINE

1210  8906  A9 09     SCAND  LDA #$9          ;SCAN DISPLAY FROM DISBUF
1211  8908  20 A5 89         JSR CONFIG
1212  890B  A2 05            LDX #5
1213  890D  A0 00     SC1    LDY #0
1214  890F  BD 40 A6         LDA DISBUF,X
1215  8912  8C 00 A4         STY PADA
1216  8915  8E 02 A4         STX PBDA
1217  8918  8D 00 A4         STA PADA
1218  891B  A0 10            LDY #$10
1219  891D  88       SC2    DEY
1220  891E  D0 FD            BNE SC2
1221  8920  CA               DEX
1222  8921  10 EA            BPL SC1
1223  8923  20 A3 89  KEYQ   JSR KSCONF       ; KEY DOWN ? (YES THEN Z=1)
1224  8926  AD 00 A4         LDA PADA
1225  8929  49 7F            EOR #$7F
1226  892B  60               RTS
1227  892C  29 3F     LRNKEY AND #$3F         ;DETERMINE WHAT KEY IS DOWN
1228  892E  8D 3F A6         STA SCRF
1229  8931  A9 05            LDA #$05
1230  8933  20 A5 89         JSR CONFIG
1231  8936  AD 02 A4         LDA PBDA
1232  8939  29 07            AND #$07
1233  893B  49 07            EOR #$07
1234  893D  D0 05            BNE LK1
1235  893F  2C 00 A4         BIT PADA
1236  8942  30 1A            BMI NOKEY
1237  8944  C9 04     LK1    CMP #$04
1238  8946  90 02            BCC LK2
1239  8948  A9 03            LDA #$03
1240  894A  0A       LK2    ASL A
1241  894B  0A               ASL A
1242  894C  0A               ASL A
1243  894D  0A               ASL A
1244  894E  0A               ASL A
1245  894F  0A               ASL A
1246  8950  18               CLC
1247  8951  6D 3F A6         ADC SCRF
1248  8954  A2 19            LDX #$19
1249  8956  DD D6 8B  LK3    CMP SYM,X
1250  8959  F0 05            BEQ FOUND
1251  895B  CA               DEX
1252  895C  10 F8            BPL LK3
1253  895E  E8       NOKEY  INX
1254  895F  60               RTS
1255  8960  8A       FOUND  TXA
1256  8961  18               CLC
1257  8962  6D 55 A6         ADC KSHFL
1258  8965  AA               TAX
1259  8966  BD EF 8B         LDA ASCII,X
1260  8969  60               RTS
1261  896A  20 23 89  KYSTAT JSR KEYQ         ;KEY DOWN? RETURN IN CARRY
1262  896D  18               CLC
1263  896E  F0 01            BEQ *+3
1264  8970  38               SEC
```

```
LINE #  LOC    CODE        LINE

1265  8971  60              RTS
1266  8972  20 88 81  BEEP   JSR SAVER       ;DELAY (BOUNCE) W/BEEP
1267  8975  A9 0D      BEEPP3 LDA #$0D
1268  8977  20 A5 89    BEEPP5 JSR CONFIG
1269  897A  A2 70              LDX #$70      ;DURATION CONSTANT
1270  897C  A9 08      BE1    LDA #8
1271  897E  8D 02 A4           STA PBDA
1272  8981  20 95 89           JSR BE2
1273  8984  A9 06              LDA #6
1274  8986  8D 02 A4           STA PBDA
1275  8989  20 95 89           JSR BE2
1276  898C  CA                 DEX
1277  898D  D0 ED              BNE BE1
1278  898F  20 A3 89           JSR KSCONF
1279  8992  4C C4 81           JMP RESALL
1280  8995  A0 1A      BE2    LDY #$1A
1281  8997  88        BE3    DEY
1282  8998  D0 FD              BNE BE3
1283  899A  60                 RTS
1284  899B  20 88 81  NOBEEP JSR SAVER       ;DELAY W/O BEEP
1285  899E  A9 01              LDA #$01
1286  89A0  4C 77 89           JMP BEEPP5    ;(BNE BEEPP5, $FF)
1287  89A3  A9 01      KSCONF LDA #$1        ;CONFIGURE FOR KEYBOARD
1288  89A5  20 88 81  CONFIG JSR SAVER       ;CONFIGURE I/O FROM TABLE VAL
1289  89A8  A0 01              LDY #$01
1290  89AA  AA                 TAX
1291  89AB  BD C8 8B  CON1   LDA VALSP2,X
1292  89AE  99 02 A4           STA PBDA,Y
1293  89B1  BD C6 8B           LDA VALS,X
1294  89B4  99 00 A4           STA PADA,Y
1295  89B7  CA                 DEX
1296  89B8  88                 DEY
1297  89B9  10 F0              BPL CON1
1298  89BB  4C C4 81           JMP RESALL
1299  89BE  20 AF 88  HKEY   JSR GETKEY      ;GET KEY FROM KB AND ECHO ON KB
1300  89C1  20 88 81  OUTDSP JSR SAVER       ;DISPLAY OUT
1301  89C4  29 7F              AND #$7F
1302  89C6  C9 07              CMP #$07      ;BELL?
1303  89C8  D0 03              BNE NBELL
1304  89CA  4C 75 89           JMP BEEPP3    ;YES - BEEP
1305  89CD  20 06 8A  NBELL  JSR TEXT        ;PUSH INTO SCOPE BUFFER
1306  89D0  C9 2C              CMP #$2C      ;COMMA?
1307  89D2  D0 0A              BNE OUD1
1308  89D4  AD 45 A6           LDA RDIG
1309  89D7  09 80              ORA #$80      ;TURN ON DECIMAL PT
1310  89D9  8D 45 A6           STA RDIG
1311  89DC  D0 25              BNE EXITOD
1312  89DE  A2 3A      OUD1   LDX #$3A
1313  89E0  DD EE 8B  OUD2   CMP ASCIM1,X
1314  89E3  F0 05              BEQ GETSGS
1315  89E5  CA                 DEX
1316  89E6  D0 F8              BNE OUD2
1317  89E8  F0 19              BEQ EXITOD
1318  89EA  BD 28 8C  GETSGS LDA SEGSM1,X    ;GET CORR SEG CODE FROM TABLE
1319  89ED  C9 F0              CMP #$F0
```

```
LINE #  LOC    CODE        LINE

1320   89EF   F0 12            BEQ EXITOD
1321   89F1   A2 00            LDX #0
1322   89F3   48               PHA
1323   89F4   BD 41 A6   OUD3   LDA DISBUF+1,X  ;SHOVE DOWN DISPLAY BUFFER
1324   89F7   9D 40 A6         STA DISBUF,X
1325   89FA   E8               INX
1326   89FB   E0 05            CPX #5
1327   89FD   D0 F5            BNE OUD3
1328   89FF   68               PLA
1329   8A00   8D 45 A6         STA RDIG
1330   8A03   4C C4 81   EXITOD JMP RESALL
1331   8A06   48         TEXT  PHA              ;UPDATE SCOPE BUFFER
1332   8A07   8A               TXA           ;SAVE X
1333   8A08   48               PHA
1334   8A09   A2 1E            LDX #$1E       ;PUSH DOWN 32 CHRS
1335   8A0B   BD 00 A6   TXTMOV LDA SCPBUF,X
1336   8A0E   9D 01 A6         STA SCPBUF+1,X
1337   8A11   CA               DEX
1338   8A12   10 F7            BPL TXTMOV
1339   8A14   68               PLA           ;RESTORE X
1340   8A15   AA               TAX
1341   8A16   68               PLA           ;RESTORE CHR
1342   8A17   8D 00 A6         STA SCPBUF      ;STORE CHR IN EMPTY SLOT
1343   8A1A   60               RTS
1344   8A1B               ;
1345   8A1B               ;***
1346   8A1B               ;*** TERMINAL I/O
1347   8A1B               ;***
1348   8A1B   20 88 81   INCHR  JSR SAVER        ;INPUT CHAR
1349   8A1E   20 41 8A          JSR INJINV
1350   8A21   29 7F            AND #$7F        ;DROP PARITY
1351   8A23   C9 61            CMP #$61        ;ALPHA?
1352   8A25   90 06            BCC INRT1
1353   8A27   C9 7B            CMP #$7B
1354   8A29   B0 02            BCS INRT1
1355   8A2B   29 DF            AND #$DF        ;CVRT TO UPPER CASE
1356   8A2D   C9 0F      INRT1  CMP #$0F        ;CTL O ?
1357   8A2F   D0 0B            BNE INRT2
1358   8A31   AD 53 A6         LDA TECHO
1359   8A34   49 40            EOR #$40        ;TOGGLE CTL O BIT
1360   8A36   8D 53 A6         STA TECHO
1361   8A39   18               CLC
1362   8A3A   90 E2            BCC INCHR+3     ;GO GET ANOTHER CHAR
1363   8A3C   C9 0D      INRT2  CMP #$0D        ;CARRIAGE RETURN?
1364   8A3E   4C B8 81          JMP RESXAF
1365   8A41   6C 61 A6   INJINV JMP (INVEC+1)
1366   8A44   20 09 83   NBASOC JSR NIBASC      ;NIBBLE TO ASCII, OUTCHR
1367   8A47   20 88 81   OUTCHR JSR SAVER
1368   8A4A   2C 53 A6          BIT TECHO       ;LOOK AT CTL O FLAG
1369   8A4D   70 03             BVS *+5
1370   8A4F   20 55 8A          JSR INJOUV
1371   8A52   4C C4 81          JMP RESALL
1372   8A55   6C 64 A6   INJOUV JMP (OUTVEC+1)
1373   8A58   20 88 81   INTCHR JSR SAVER        ;IN TERMINAL CHAR
1374   8A5B   A9 00             LDA #0
```

```
LINE #  LOC    CODE        LINE

1375  8A5D  85 F9           STA $F9
1376  8A5F  AD 02 A4  LOOK   LDA PBDA       ;FIND LEADING EDGE
1377  8A62  2D 54 A6        AND TOUTFL
1378  8A65  38              SEC
1379  8A66  E9 40           SBC #$40
1380  8A68  90 F5           BCC LOOK
1381  8A6A  20 E9 8A  TIN    JSR DLYH       ;TERMINAL BIT
1382  8A6D  AD 02 A4        LDA PBDA
1383  8A70  2D 54 A6        AND TOUTFL
1384  8A73  38              SEC
1385  8A74  E9 40           SBC #$40       ;OR BITS 6,7 (TTY,CRT)
1386  8A76  2C 53 A6        BIT TECHO      ;ECHO BIT?
1387  8A79  10 06           BPL DMY1
1388  8A7B  20 D4 8A        JSR OUT
1389  8A7E  4C 87 8A        JMP SAVE
1390  8A81  A0 07     DMY1  LDY #7
1391  8A83  88        TLP1  DEY
1392  8A84  D0 FD           BNE TLP1
1393  8A86  EA              NOP
1394  8A87  66 F9     SAVE  ROR $F9
1395  8A89  20 E9 8A        JSR DLYH
1396  8A8C  48              PHA            ;TIMING
1397  8A8D  B5 00           LDA 0,X
1398  8A8F  68              PLA
1399  8A90  90 D8           BCC TIN
1400  8A92  20 E9 8A        JSR DLYH
1401  8A95  18              CLC
1402  8A96  20 D4 8A        JSR OUT
1403  8A99  A5 F9           LDA $F9
1404  8A9B  49 FF           EOR #$FF
1405  8A9D  4C B8 81        JMP RESXAF
1406  8AA0  85 F9     TOUT  STA $F9        ;TERMINAL CHR OUT
1407  8AA2  20 88 81        JSR SAVER
1408  8AA5  20 E9 8A        JSR DLYH       ;DELAY 1/2 BIT TIME
1409  8AA8  A9 30           LDA #$30       ;SET FOR OUTPUT
1410  8AAA  8D 03 A4        STA PBDA+1     ;DATA DIRECTION
1411  8AAD  A5 F9           LDA $F9        ;RECOVER CHR DATA
1412  8AAF  A2 0B           LDX #$0B       ;START BIT,8DATA, 3STOPS
1413  8AB1  49 FF           EOR #$FF       ;INVERT DATA
1414  8AB3  38              SEC            ;START BIT
1415  8AB4  20 D4 8A  OUTC   JSR OUT       ;OUTPUT BIT FROM CARRY
1416  8AB7  20 E6 8A        JSR DLYF       ;WAIT FULL BIT TIME
1417  8ABA  A0 06           LDY #$06
1418  8ABC  88        PHAKE DEY
1419  8ABD  D0 FD           BNE PHAKE
1420  8ABF  EA              NOP
1421  8AC0  4A              LSR A
1422  8AC1  CA              DEX
1423  8AC2  D0 F0           BNE OUTC
1424  8AC4  A5 F9           LDA $F9
1425  8AC6  C9 0D           CMP #$0D       ;CARRIAGE RETURN?
1426  8AC8  F0 04           BEQ GOPAD      ;YES-PAD IT
1427  8ACA  C9 0A           CMP #$0A       ;PAD LINE FEED TOO
1428  8ACC  D0 03           BNE LEAVE
1429  8ACE  20 32 8B  GOPAD  JSR PAD
```

```
LINE #  LOC     CODE        LINE

1430  8AD1  4C C4 81  LEAVE  JMP RESALL
1431  8AD4  48        OUT    PHA               ;TERMINAL BIT OUT
1432  8AD5  AD 02 A4          LDA PBDA
1433  8AD8  29 0F             AND #$0F
1434  8ADA  90 02             BCC OUTONE
1435  8ADC  09 30             ORA #$30
1436  8ADE  2D 54 A6  OUTONE  AND TOUTFL        ;MASK OUTPUT
1437  8AE1  8D 02 A4          STA PBDA
1438  8AE4  68                PLA
1439  8AE5  60                RTS
1440  8AE6
1441  8AE6  20 E9 8A  DLYF   JSR DLYH           ;DELAY FULL
1442  8AE9  08        DLYH   PHP                ;DELAY HALF
1443  8AEA  48               PHA
1444  8AEB  8A               TXA
1445  8AEC  48               PHA
1446  8AED  98               TYA
1447  8AEE  AE 51 A6         LDX SDBYT
1448  8AF1  A0 03     DLYX   LDY #3
1449  8AF3  88        DLYY   DEY
1450  8AF4  D0 FD            BNE DLYY
1451  8AF6  CA               DEX
1452  8AF7  D0 F8            BNE DLYX
1453  8AF9  A8               TAY
1454  8AFA  68               PLA
1455  8AFB  AA               TAX
1456  8AFC  68               PLA
1457  8AFD  28               PLP
1458  8AFE  60               RTS
1459  8AFF  A9 00     BAUD   LDA #0             ;DETERMINE BAUD RATE ON PB7
1460  8B01  A8               TAY
1461  8B02  AD 02 A4  SEEK   LDA PBDA
1462  8B05  0A               ASL A
1463  8B06  B0 FA            BCS SEEK
1464  8B08  20 27 8B  CLEAR  JSR INK
1465  8B0B  90 FB            BCC CLEAR
1466  8B0D  20 27 8B  SET    JSR INK
1467  8B10  B0 FB            BCS SET
1468  8B12  8C 51 A6         STY SDBYT
1469  8B15  BD 63 8C  DEAF   LDA DECPTS,X
1470  8B18  CD 51 A6         CMP SDBYT
1471  8B1B  B0 07            BCS AGAIN
1472  8B1D  BD 69 8C         LDA STDVAL,X       ;LOAD CLOSEST STD VALUE
1473  8B20  8D 51 A6         STA SDBYT
1474  8B23  60               RTS
1475  8B24  E8        AGAIN  INX
1476  8B25  10 EE            BPL DEAF
1477  8B27  C8        INK    INY
1478  8B28  A2 1C            LDX #$1C
1479  8B2A  CA        INK1   DEX
1480  8B2B  D0 FD            BNE INK1
1481  8B2D  AD 02 A4         LDA PBDA
1482  8B30  0A               ASL A
1483  8B31  60               RTS
1484  8B32  AE 50 A6  PAD    LDX PADBIT         ;PAD CARRIAGE RETURN OR LF
```

```
LINE # LOC     CODE        LINE

1485  8B35  20 E6 8A    PAD1   JSR DLYF         ;WITH EXTRA STOP BITS
1486  8B38  CA                 DEX
1487  8B39  D0 FA              BNE PAD1
1488  8B3B  60                 RTS
1489  8B3C  20 A3 89    TSTAT  JSR KSCONF       ;SEE IF BREAK KEY DOWN
1490  8B3F  AD 02 A4           LDA PBDA
1491  8B42  2D 54 A6           AND TOUTFL
1492  8B45  38                 SEC
1493  8B46  E9 40              SBC #$40
1494  8B48  60                 RTS
1495  8B49  FF                 .BYT $FF         ;NOT USED
1496  8B4A              ; ***
1497  8B4A              ; *** RESET - TURN OFF POR, INIT SYS RAM, ENTER MONITOR
1498  8B4A              ; ***
1499  8B4A              ;
1500  8B4A  A2 FF       RESET  LDX #$FF
1501  8B4C  9A                 TXS              ;INIT STACK PTR
1502  8B4D  A9 CC       POR    LDA #$CC
1503  8B4F  8D 0C A0           STA PCR1         ;DISABLE POR, TAPE OFF
1504  8B52  A9 04              LDA #4
1505  8B54  48                 PHA
1506  8B55  28                 PLP              ;INIT F, DISABLE IRQ DURING DFTXFR
1507  8B56  20 86 8B           JSR ACCESS       ;UN WRITE PROT SYS RAM
1508  8B59  A2 5F       DFTXFR LDX #$5F         ;INIT SYS RAM (EXCPT SCPBUF)
1509  8B5B  BD A0 8F           LDA DFTBLK,X
1510  8B5E  9D 20 A6           STA RAM,X
1511  8B61  CA                 DEX
1512  8B62  10 F7              BPL DFTXFR+2
1513  8B64  A9 07       NEWDEV LDA #7           ;CHANGE DEVC/BAUD RATE
1514  8B66  20 47 8A           JSR OUTCHR       ;BEEP
1515  8B69  20 A3 89    SWITCH JSR KSCONF       ;KEYBOARD OR TERMINAL?
1516  8B6C  20 26 89    SWLP   JSR KEYQ+3
1517  8B6F  D0 0B              BNE MONENT
1518  8B71  2C 02 A4           BIT PBDA
1519  8B74  10 F6              BPL SWLP
1520  8B76  20 B7 8B           JSR VECSW        ;SWITCH VECTORS
1521  8B79  20 FF 8A           JSR BAUD
1522  8B7C  A2 FF       MONENT LDX #$FF         ;MONITOR ENTRY
1523  8B7E  9A                 TXS
1524  8B7F  D8                 CLD
1525  8B80  20 86 8B           JSR ACCESS       ;UNWRITE PROT MONITOR RAM
1526  8B83  4C 03 80           JMP WARM
1527  8B86  20 88 81    ACCESS JSR SAVER        ;UN WRITE PROT SYS RAM
1528  8B89  AD 01 AC           LDA OR3A
1529  8B8C  09 01              ORA #1
1530  8B8E  8D 01 AC    ACC1   STA OR3A
1531  8B91  AD 03 AC           LDA DDR3A
1532  8B94  09 01              ORA #1
1533  8B96  8D 03 AC           STA DDR3A
1534  8B99  4C C4 81           JMP RESALL
1535  8B9C  20 88 81    NACCES JSR SAVER        ;WRITE PROT SYS RAM
1536  8B9F  AD 01 AC           LDA OR3A
1537  8BA2  29 FE              AND #$FE
1538  8BA4  18                 CLC
1539  8BA5  90 E7              BCC ACC1
```

```
LINE # LOC      CODE        LINE

1540  8BA7  20 86 8B    TTY     JSR ACCESS        ;UN WRITE PROT RAM
1541  8BAA  A9 D5               LDA #$D5          ;110 BAUD
1542  8BAC  8D 51 A6           STA SDBYT
1543  8BAF  AD 54 A6           LDA TOUTFL
1544  8BB2  09 40              ORA #$40
1545  8BB4  8D 54 A6           STA TOUTFL
1546  8BB7  20 86 8B    VECSW   JSR ACCESS        ;UN WRITE PROT RAM
1547  8BBA  A2 08               LDX #$8
1548  8BBC  BD 6F 8C    SWLP2   LDA TRMTBL,X
1549  8BBF  9D 60 A6           STA INVEC,X
1550  8BC2  CA                  DEX
1551  8BC3  10 F7               BPL SWLP2
1552  8BC5  60                  RTS
1553  8BC6              ;
1554  8BC6              ;***
1555  8BC6              ;*** TABLES (I/O CONFIGURATIONS, KEY CODES, ASCII CODES)
1556  8BC6              ;***
1557  8BC6  00          VALS    .BYT $00,$80,$08,$37 ;KB SENSE, A=1
1557  8BC7  80
1557  8BC8  08
1557  8BC9  37
1558  8BCA  00                  .BYT $00,$7F,$00,$30 ;KB LRN, A=5
1558  8BCB  7F
1558  8BCC  00
1558  8BCD  30
1559  8BCE  00                  .BYT $00,$FF,$00,$3F ;SCAN DSP, A=9
1559  8BCF  FF
1559  8BD0  00
1559  8BD1  3F
1560  8BD2  00                  .BYT $00,$00,$07,$3F ;BEEP, A=D
1560  8BD3  00
1560  8BD4  07
1560  8BD5  3F
1561  8BD6              VALSP2 =VALS+2
1562  8BD6              SYM     =*                ;KEY CODES RETURNED BY LRNKEY
1563  8BD6              TABLE=*
1564  8BD6  01                  .BYT $01          ;0/U0
1565  8BD7  41                  .BYT $41          ;1/U1
1566  8BD8  81                  .BYT $81          ;2/U2
1567  8BD9  C1                  .BYT $C1          ;3/U3
1568  8BDA  02                  .BYT $02          ;4/U4
1569  8BDB  42                  .BYT $42          ;5/U5
1570  8BDC  82                  .BYT $82          ;6/U6
1571  8BDD  C2                  .BYT $C2          ;7/U7
1572  8BDE  04                  .BYT $04          ;8/JMP
1573  8BDF  44                  .BYT $44          ;9/VER
1574  8BE0  84                  .BYT $84          ;A/ASCII
1575  8BE1  C4                  .BYT $C4          ;B/BLK MOV
1576  8BE2  08                  .BYT $08          ;C/CALC
1577  8BE3  48                  .BYT $48          ;D/DEP
1578  8BE4  88                  .BYT $88          ;E/EXEC
1579  8BE5  C8                  .BYT $C8          ;F/FILL
1580  8BE6  10                  .BYT $10          ;CR/SD
1581  8BE7  50                  .BYT $50          ;-/+
1582  8BE8  90                  .BYT $90          ;>/<
```

```
LINE # LOC     CODE        LINE

1583  8BE9  D0                    .BYT $D0        ;SHIFT
1584  8BEA  20                    .BYT $20        ;GO/LP
1585  8BEB  60                    .BYT $60        ;REG/SP
1586  8BEC  A0                    .BYT $A0        ;MEM/WP
1587  8BED  00                    .BYT $00        ;L2/L1
1588  8BEE  40                    .BYT $40        ;S2/S1
1589  8BEF              ASCIM1 =*-1
1590  8BEF              ASCII  =*                 ;ASCII CODES AND HASH CODES
1591  8BEF  30                    .BYT $30        ;ZERO
1592  8BF0  31                    .BYT $31        ;ONE
1593  8BF1  32                    .BYT $32        ;TWO
1594  8BF2  33                    .BYT $33        ;THREE
1595  8BF3  34                    .BYT $34        ;FOUR
1596  8BF4  35                    .BYT $35        ;FIVE
1597  8BF5  36                    .BYT $36        ;SIX
1598  8BF6  37                    .BYT $37        ;SEVEN
1599  8BF7  38                    .BYT $38        ;EIGHT
1600  8BF8  39                    .BYT $39        ;NINE
1601  8BF9  41                    .BYT $41        ;A
1602  8BFA  42                    .BYT $42        ;B
1603  8BFB  43                    .BYT $43        ;C
1604  8BFC  44                    .BYT $44        ;D
1605  8BFD  45                    .BYT $45        ;E
1606  8BFE  46                    .BYT $46        ;F
1607  8BFF  0D                    .BYT $0D        ;CR
1608  8C00  2D                    .BYT $2D        ;DASH
1609  8C01  3E                    .BYT $3E        ;>
1610  8C02  FF                    .BYT $FF        ;SHIFT
1611  8C03  47                    .BYT $47        ;G
1612  8C04  52                    .BYT $52        ;R
1613  8C05  4D                    .BYT $4D        ;M
1614  8C06  13                    .BYT $13        ;L2
1615  8C07  1E                    .BYT $1E        ;S2
1616  8C08              ; KB UPPER CASE
1617  8C08  14                    .BYT $14        ;U0
1618  8C09  15                    .BYT $15        ;U1
1619  8C0A  16                    .BYT $16        ;U2
1620  8C0B  17                    .BYT $17        ;U3
1621  8C0C  18                    .BYT $18        ;U4
1622  8C0D  19                    .BYT $19        ;U5
1623  8C0E  1A                    .BYT $1A        ;U6
1624  8C0F  1B                    .BYT $1B        ;U7
1625  8C10  4A                    .BYT $4A        ;J
1626  8C11  56                    .BYT $56        ;V
1627  8C12  FE                    .BYT $FE        ;ASCII
1628  8C13  42                    .BYT $42        ;B
1629  8C14  43                    .BYT $43        ;C
1630  8C15  44                    .BYT $44        ;D
1631  8C16  45                    .BYT $45        ;E
1632  8C17  46                    .BYT $46        ;F
1633  8C18  10                    .BYT $10        ;SD
1634  8C19  2B                    .BYT $2B        ;+
1635  8C1A  3C                    .BYT $3C        ;<
1636  8C1B  00                    .BYT $00        ;SHIFT
1637  8C1C  11                    .BYT $11        ;LP
```

```
LINE # LOC     CODE        LINE

1638  8C1D  1C                    .BYT $1C        ;SP
1639  8C1E  57                    .BYT $57        ;W
1640  8C1F  12                    .BYT $12        ;L1
1641  8C20  1D                    .BYT $1D        ;S1
1642  8C21  2E                    .BYT $2E        ;.
1643  8C22  20                    .BYT $20        ;BLANK
1644  8C23  3F                    .BYT $3F        ;?
1645  8C24  50                    .BYT $50        ;P
1646  8C25  07                    .BYT $07        ;BELL
1647  8C26  53                    .BYT $53        ;S
1648  8C27  58                    .BYT $58        ;X
1649  8C28  59                    .BYT $59        ;Y
1650  8C29                 ; SEGMENT CODES FOR ON-BOARD DISPLAY
1651  8C29                 SEGSM1 =*-1
1652  8C29  3F                    .BYT $3F        ;ZERO
1653  8C2A  06                    .BYT $06        ;ONE
1654  8C2B  5B                    .BYT $5B        ;TWO
1655  8C2C  4F                    .BYT $4F        ;THREE
1656  8C2D  66                    .BYT $66        ;FOUR
1657  8C2E  6D                    .BYT $6D        ;FIVE
1658  8C2F  7D                    .BYT $7D        ;SIX
1659  8C30  07                    .BYT $07        ;SEVEN
1660  8C31  7F                    .BYT $7F        ;EIGHT
1661  8C32  67                    .BYT $67        ;NINE
1662  8C33  77                    .BYT $77        ;A
1663  8C34  7C                    .BYT $7C        ;B
1664  8C35  39                    .BYT $39        ;C
1665  8C36  5E                    .BYT $5E        ;D
1666  8C37  79                    .BYT $79        ;E
1667  8C38  71                    .BYT $71        ;F
1668  8C39  F0                    .BYT $F0        ;CR
1669  8C3A  40                    .BYT $40        ;DASH
1670  8C3B  70                    .BYT $70        ;>
1671  8C3C  00                    .BYT $00        ;SHIFT
1672  8C3D  6F                    .BYT $6F        ;G
1673  8C3E  50                    .BYT $50        ;R
1674  8C3F  54                    .BYT $54        ;M
1675  8C40  38                    .BYT $38        ;L2
1676  8C41  6D                    .BYT $6D        ;S2
1677  8C42  01                    .BYT $01        ;U0
1678  8C43  08                    .BYT $08        ;U1
1679  8C44  09                    .BYT $09        ;U2
1680  8C45  30                    .BYT $30        ;U3
1681  8C46  36                    .BYT $36        ;U4
1682  8C47  5C                    .BYT $5C        ;U5
1683  8C48  63                    .BYT $63        ;U6
1684  8C49  03                    .BYT $03        ;U7
1685  8C4A  1E                    .BYT $1E        ;J
1686  8C4B  72                    .BYT $72        ;V
1687  8C4C  77                    .BYT $77        ;A
1688  8C4D  7C                    .BYT $7C        ;B
1689  8C4E  39                    .BYT $39        ;C
1690  8C4F  5E                    .BYT $5E        ;D
1691  8C50  79                    .BYT $79        ;E
1692  8C51  71                    .BYT $71        ;F
```

```
LINE # LOC     CODE        LINE

1693  8C52  6D                    .BYT $6D        ;SD
1694  8C53  76                    .BYT $76        ;+
1695  8C54  46                    .BYT $46        ;<
1696  8C55  00                    .BYT $00        ;SHIFT
1697  8C56  38                    .BYT $38        ;LF
1698  8C57  6D                    .BYT $6D        ;SP
1699  8C58  1C                    .BYT $1C        ;W
1700  8C59  38                    .BYT $38        ;L1
1701  8C5A  6D                    .BYT $6D        ;S1
1702  8C5B  80                    .BYT $80        ;.
1703  8C5C  00                    .BYT $00        ;SPACE
1704  8C5D  53                    .BYT $53        ;?
1705  8C5E  73                    .BYT $73        ;P
1706  8C5F  49                    .BYT $49        ;BELL
1707  8C60  6D                    .BYT $6D        ;S
1708  8C61  64                    .BYT $64        ;X
1709  8C62  6E                    .BYT $6E        ;Y
1710  8C63  97              DECPTS .BYT $97,$3D,$1F,$10,$08,$00 ; TO DETERMINE BAUD RATE
1710  8C64  3D
1710  8C65  1F
1710  8C66  10
1710  8C67  08
1710  8C68  00
1711  8C69  D5 4C          STDVAL .DBY $D54C,$2410,$0601   ;STD VALS FOR BAUD RATES
1711  8C6B  24 10
1711  8C6D  06 01
1712  8C6F                 ; 110,300,600,1200,2400,4800 BAUD
1713  8C6F  4C 58 8A       TRMTBL JMP INTCHR       ;ALTERNATE VCTRS FOR TIO
1714  8C72  4C A0 8A              JMP TOUT
1715  8C75  4C 3C 8B              JMP TSTAT
1716  8C78                 ;
1717  8C78                 ;
1718  8C78                 ;***
1719  8C78                 ;*** DEFAULT TABLE
1720  8C78                 ;***
1721  8C78                        *=$8FA0
1722  8FA0                 DFTBLK=*
1723  8FA0  00 C0                 .WORD $C000     ;BASIC  *** JUMP TABLE
1724  8FA2  A7 8B                 .WORD TTY
1725  8FA4  64 8B                 .WORD NEWDEV
1726  8FA6  00 00                 .WORD $0000     ;PAGE ZERO
1727  8FA8  00 02                 .WORD $0200
1728  8FAA  00 03                 .WORD $0300
1729  8FAC  00 C8                 .WORD $C800
1730  8FAE  00 D0                 .WORD $D000
1731  8FB0  04                    .BYT $04        ;TAPE DELAY (9.0 SEC)
1732  8FB1  2C                    .BYT $2C        ;KIM TAPE BOUNDARY
1733  8FB2  46                    .BYT $46        ;HS TAPE BOUNDARY
1734  8FB3  00                    .BYT $00,$00    ;SCR3,SCR4
1734  8FB4  00
1735  8FB5  33                    .BYT $33        ;HS TAPE FIRST 1/2 BIT
1736  8FB6  00                    .BYT $00,$00    ;SCR6,SCR7
1736  8FB7  00
1737  8FB8  00                    .BYT $00,$00,$00,$00 ;SCR8-SCRB
1737  8FB9  00
```

```
LINE # LOC      CODE        LINE

1737  8FBA  00
1737  8FBB  00
1738  8FBC  5A                    .BYT $5A          ;HS TAPE SECOND 1/2 BIT
1739  8FBD  00                    .BYT $00,$00,$00 ;SCRD-SCRF
1739  8FBE  00
1739  8FBF  00
1740  8FC0  00                    .BYT $00,$00,$6D,$6E,$86,$06 ;DISP BUFFER (SY1.1)
1740  8FC1  00
1740  8FC2  6D
1740  8FC3  6E
1740  8FC4  86
1740  8FC5  06
1741  8FC6  00                    .BYT $00,$00,$00 ;NOT USED
1741  8FC7  00
1741  8FC8  00
1742  8FC9  00                    .BYT $00          ;PARNR
1743  8FCA  00 00                 .DBYT $0000,$0000,$0000 ;PARMS
1743  8FCC  00 00
1743  8FCE  00 00
1744  8FD0  01                    .BYT $01          ;PADBIT
1745  8FD1  4C                    .BYT $4C          ;SDBYT
1746  8FD2  00                    .BYT $00          ;ERCNT
1747  8FD3  80                    .BYT $80          ;TECHO
1748  8FD4  B0                    .BYT $B0          ;TOUTFL
1749  8FD5  00                    .BYT $00          ;KSHFL
1750  8FD6  00                    .BYT $00          ;TV
1751  8FD7  00                    .BYT $00          ;LSTCOM
1752  8FD8  10                    .BYT $10          ;MAXRC
1753  8FD9  4A 8B                 .WORD RESET       ;USER REG'S
1754  8FDB  FF                    .BYT $FF          ;STACK
1755  8FDC  00                    .BYT $00          ;FLAGS
1756  8FDD  00                    .BYT $00          ;A
1757  8FDE  00                    .BYT $00          ;X
1758  8FDF  00                    .BYT $00          ;Y
1759  8FE0                  ; VECTORS
1760  8FE0  4C BE 89              JMP HKEY          ;INVEC
1761  8FE3  4C 00 89              JMP HDOUT         ;OUTVEC
1762  8FE6  4C 6A 89              JMP KYSTAT        ;INSVEC
1763  8FE9  4C D1 81              JMP M1            ;UNRECOGNIZED SYNTAX (ERROR)
1764  8FEC  4C D1 81              JMP M1            ;UNRECOGNIZED COMMAND (ERROR)
1765  8FEF  4C 06 89              JMP SCAND         ;SCNVEC
1766  8FF2  7E 88                 .WORD RIN         ;IN PTR FOR EXEC FROM RAM
1767  8FF4  C0 80                 .WORD TRCOFF      ;USER TRACE VECTOR
1768  8FF6  4A 80                 .WORD SVBRK       ;BRK
1769  8FF8  29 80                 .WORD SVIRQ       ;USER IRQ
1770  8FFA  9B 80                 .WORD SVNMI       ;NMI
1771  8FFC  4A 8B                 .WORD RESET       ;RESET
1772  8FFE  0F 80                 .WORD IRQBRK      ;IRQ


1774  9000                  LENTRY =$8C78
1775  9000                  SENTRY =$8C78+$20F
1776  9000                  RGNAM  =$8F9A           ;REGISTER NAME PATCH
```

```
LINE # LOC      CODE        LINE

1778  9000                        .END

ERRORS = 0000 <0000>
```

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
ACCESS	8B86	1527	123 129 137 177 191 1507 1525 1540 1546
ACC1	8B8E	1530	1539
ADVCK	81CB	336	581 599
AGAIN	8B24	1475	1471
AR	A65D	59	152 186 616
ASCII	8BEF	1590	1259
ASCIM1	8BEE	1589	1313
ASCNIB	8275	412	344 352 394
BADDY	848B	666	660
BAUD	8AFF	1459	1521
BEEP	8972	1266	1193
BEEPP3	8975	1267	1304
BEEPP5	8977	1268	1286
BE1	897C	1270	1277
BE2	8995	1280	1272 1275
BE3	8997	1281	1282
BLK3	8739	991	976
BLP	875E	1007	1013 1014
BLP1	879D	1035	1042
BMOVE	87B7	1048	1007 1035
BRT	87CC	1058	1052
BRTT	87C1	1053	985
BZPARM	8395	559	261
B1	87AF	1043	987 990 1005 1012 1015 1041
B1PARM	84DA	699	264
B2	8772	1016	1006
B2PARM	8619	858	267
B3PARM	8714	971	270
CALC3	8827	1103	857 970 1085
CHKSAD	82DD	463	676 887 955
CLEAR	8B08	1464	1465
COMINB	81D6	342	569
COMMA	833A	511	342 503 732
COMPAR	82CA	455	444 446
CONFIG	89A5	1288	1211 1230 1268
CON1	89AB	1291	1297
CRCHK	8204	364	362 363
CRLF	834D	521	147 219 277 497 564 587 604 622 784 905 926 1105
CRLFSZ	8316	497	706 731 882
C1	882B	1105	****
DBNEW	80F6	212	204
DBOFF	80D3	198	146 180 194
DBON	80E4	205	196
DDR1B	A002	91	****
DDR3A	AC03	89	202 209 211 212 214 854 1531 1533
DEAF	8B15	1469	1476
DECCMP	82BE	449	761 899 1040
DECPTS	8C63	1710	1469
DELAY	835A	528	188
DEPBYT	84E8	709	718
DEPEC	850E	726	720
DEPES	8553	759	722
DEPN	84F9	716	714 723 725

```
SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

DEPZ     84A7        677   620
DEP1     84DA        703   ****
DETBRK   801B        113   ****
DETIRQ   8022        118   112
DFTBLK   8FA0       1722  1509
DFTXFR   8B59       1508  1512
DIFFL    86FD        958   ****
DIFFL2   8710        969   ****
DIFFZ    86FA        957   923
DIFF1    870F        968   964  1495
DISBUF   A640         27  1214  1323  1324
DISPAT   814A        256    99
DLYF     8AE6       1441  1416  1485
DLYH     8AE9       1442  1381  1395  1400  1408  1441
DLYO     8383        544   539
DLYX     8AF1       1448  1452
DLYY     8AF3       1449  1450
DLY1     8368        533   536
DLY2     8371        537   543
DL1      835D        529   ****
DMY1     8A81       1390  1387
ERCNT    A652         43   624   632   649   653   667   671   673   979   995 1043
                           1053  1057
ERMSG    8171        275   100
EXEVEC   A672         74  1130  1135  1137
EXE3     8851       1125  1104
EXITCP   82D9        461   458
EXITG    88FF       1207  1203
EXITGK   88CC       1183  1171
EXITLP   843F        634   ****
EXITM1   8577        780   745
EXITNB   8315        496   494
EXITOD   8A03       1330  1311  1317  1320
EXITRG   83C1        579   598
EXRGP1   83C2        580   582
EXWRAP   82BD        448   447
EZPARM   84D7        698   695
E1PARM   8616        857   842
E2PARM   8711        970   919
E3PARM   88AC       1164  1126
FILL3    8714        975   ****
FOUND    8960       1255  1250
FR       A65C         58   164   612
F1       8723        981   988   989
F2       8737        990   ****
F3       872E        986   984
GETCOM   80FF        219    98   223   246
GETC1    8107        222   225   227
GETKEY   88AF       1168  1299
GETSGS   89EA       1318  1314
GK       88CF       1184  1169  1172  1179
GK1      88D4       1186  1187  1189  1206
GK2      88E4       1194  1195  1198
GOOD     81F3        355   360
GOPAD    8ACE       1429  1426
GO2      83F7        604   ****
GOZ      83F3        602   563
```

```
SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

G01      8579        782   729
G01ENT   83FA        605   197
HASHL    812F        240   234
HASHUS   8133        242   230  232  241
HDOUT    8900       1208  1754
HIPN     816E        271   257  269
HKEY     89BE       1299  1753
IDISP    8053        145   127  136  184
IJSCNV   8903       1209   537 1186
INBYTE   81D9        343   572  657  661  666  675  710
INCCMP   82B2        443   654  716  742  889  946  986 1011 1091
INCHR    8A1B       1348   222  245  252  343  351  359  382  626 1162 1362
INCP3    8293        429  1070 1081
INJINV   8A41       1365  1349
INJISV   8392        554   548  550
INJOUV   8A55       1372  1370
INK      8B27       1477  1464 1466
INK1     8B2A       1479  1480
INRT1    8A2D       1356  1352 1354
INRT2    8A3C       1363  1357
INSTAT   8386        548   538  896  944
INST1    838B        550   551
INST2    8391        553   549
INSVEC   A666         67   554
INTCHR   8A58       1373  1713
INVEC    A660         65  1128 1133 1136 1138 1158 1160 1365 1549
IRQBRK   800F        105  1765
IRQVEC   A67E         82  ****
JTABLE   A620          9   822  824
JUMP1    85B4        808   797
JUM2     85E5        833   812
KEYQ     8923       1223  1194 1197 1261 1516
KSCONF   89A3       1287  1223 1278 1489 1515
KSHFL    A655         48  1185 1205 1257
KYSTAT   896A       1261  1755
LDBYTE   84A1        675   629  637  640  643
LEAVE    8AD1       1430  1428
LENTRY   8C78       1716   693  836 1083
LK1      8944       1237  1234
LK2      894A       1240  1238
LK3      8956       1249  1252
LOCM8    8569        772   758
LOCP8    855B        764   754
LOOK     8A5F       1376  1380
LPGD     846D        654   648
LPZ      8429        625   641  664  670  674
LPZB     8417        618   603
LP1      842C        626   628
LRNKEY   892C       1227  1188
LSTCOM   A657         50   235  244  248  258
L1J      84CC        693   697
L1ZB     84C6        690   684
L11B     85D7        826   809
L11C     85DD        829   840
L11D     85E9        835   917
L12B     8688        911   879  915
L12C     868C        913  ****
```

SYMBOL	VALUE	LINE DEFINED		CROSS-REFERENCES					
L2ZB	84CF	694	691						
L21B	85EF	837	827						
L23P	87EE	1076	1073						
MAXRC	A658	51	927	929					
MEMZ	84AE	680	678						
MEM1	8510	728	704						
MEM2	862F	873	863						
MEM3	8801	1084	1077						
MEM3C	8808	1087	877	1093	1094				
MEM3D	8811	1091	1100						
MEM3E	881C	1097	1090						
MEM3EX	881A	1095	1092						
MEM3F	8826	1102	1098						
MONENT	8B7C	1522	97	1517					
MONITR	8000	97	****						
MORED	8454	643	656						
MORED2	86D2	941	949						
M1	81D1	339	337	1757					
M12	8159	262	260						
M13	8160	265	263						
M14	8167	268	266						
M15	81B7	314	275						
M21	8239	387	384						
M22	824A	394	386						
M23	8251	397	400						
M24	8267	406	393	395					
M25	826F	409	407						
M26	8289	422	415	417					
M27	828D	425	419						
M28	828F	426	421						
M29	8292	428	413						
M32	82C8	454	450						
M33	82EB	470	468						
M34	83C8	583	577						
M35	83CA	584	597	600					
M36	83EB	598	595						
M42	8566	770	768						
M43	8574	778	776						
NACCES	8B9C	1535	605	785	813				
NBASOC	8A44	1366	485	487	591				
NBELL	89CD	1305	1303						
NEWDEV	8B64	1513	1725						
NEWLN	84E1	706	679	719					
NEWLOC	8517	731	682	744	748	763	771	779	1097
NH3	83BF	578	570	573					
NH41	8501	720	711						
NH42	8537	745	736						
NIBALF	8313	495	492						
NIBASC	8309	490	1366						
NMIVEC	A67A	80	****						
NOBEEP	899B	1284	1196						
NOKEY	895E	1253	1236						
NOTCR	83C3	581	578						
NR10	8408	611	795	825					
NUREC	8443	636	631						
NXTLOC	8531	742	740	750	752				
NXTRG	83D2	588	****						

```
SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

OBCMIN   81D3         341   594  735
OBCRLF   834A         520   187
OCMCK    8325         503   895  904
OPCCOM   8337         510   148  185
OR1B     A000          90  ****
OR3A     AC01          88    89  198  201  205  208  852 1528 1530 1536
OUD1     89DE        1312  1307
OUD2     89E0        1313  1316
OUD3     89F4        1323  1327
OUT      8AD4        1431  1388 1402 1415
OUTBYT   82FA         479   284  341  477  505  520  888  956
OUTC     8AB4        1415  1423
OUTCHR   8A47        1367   150  221  279  281  502  517  523  525  566  589
                           934 1154 1514
OUTDSP   89C1        1300  1208
OUTONE   8ADE        1436  1434
OUTPC    82EE         473   510  568
OUTQM    8320         501   715  741
OUTSZ    8319         498  ****
OUTVEC   A663          66  1372
OUTXAH   82F4         475   500  908  952 1122
OUT1     81FE         361   358
OUT2     8201         363   353
OUT4     81F5         357   345
PAD      8B32        1484  1429
PADA     A400          86  1215 1217 1224 1235 1294
PADBIT   A650          41  1484
PAD1     8B35        1485  1487
PARFIL   822E         382   405
PARM     8220         377   239
PARNR    A649          30   259  379  389  390  408
PBDA     A402          87  1216 1231 1271 1274 1292 1376 1382 1410 1432 1437
                           1461 1481 1490 1518
PCHR     A65A          56   134  162  474  576  593  596  608
PCLR     A659          55   132  159  473  574  610
PCR1     A00C          92  1503
PHAKE    8ABC        1418  1419
PM1      822B         381   392
POR      8B4D        1502  ****
PRM10    820A         368   375
PRVLOC   8555         761   756
PSHOVE   8208         367   237  238  381  834  835
PTRIN    8872        1139  1131
P1H      A64F          40   373  999 1111
P1L      A64E          39   372  876  980  997 1062 1078 1087 1108
P2H      A64D          38   371  433  804  865 1033 1112
P2L      A64C          37   370  435  799  869  875  913 1031 1109
P2SCR    829C         433   880  922  977  996 1086
P3H      A64B          36   369  398  431  438  457  688  792  803  806  847
                           962 1021 1034 1119 1139
P3L      A64A          35   368  397  401  402  429  440  460  686  794  798
                           802  810  829  843  958 1018 1032 1116 1141
P3SCR    82A7         438   705  730  864 1030
RAM      A620           8  1510
RC       A63D          24   636  655  932  935  948
RDIG     A645          28  1308 1310 1329
REGZ     8395         562  ****
```

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
RESALL	81C4	326	925 1279 1298 1330 1371 1430 1534
RESET	8B4A	1500	1746 1764
RESTIV	8899	1157	1148
RESXAF	81B8	316	411 1156 1163 1183 1364 1405
RESXF	81BE	321	462 544 635
RGBACK	8399	564	586
RIN	887E	1145	1759
RSTVEC	A67C	81	****
SAVE	8A87	1394	1389
SAVER	8188	289	377 455 529 621 921 1145 1168 1266 1284 1288 1300 1348 1367 1373 1407 1527 1535
SAVINT	8064	152	125 131 139 179 193
SCAND	8906	1210	1758
SCNVEC	A66F	70	1209
SCPBUF	A600	7	1335 1336 1342
SCRA	A63A	20	1132 1134 1157 1159
SCRB	A63B	21	****
TAPET2	A63C	22	****
SCRD	A63D	23	24
SCRE	A63E	25	1178 1182
SCRF	A63F	26	1228 1247
TAPDEL	A630	10	****
KMBDRY	A631	11	****
HSBDRY	A632	12	****
SCR3	A633	13	350 354 380 388 404 406
SCR4	A634	14	571 575
TAPET1	A635	15	****
SCR6	A636	16	466 467 504 507 663 951
SCR7	A637	17	469 508 659 907 950
SCR8	A638	18	532 533 540
SCR9	A639	19	531 534 542
SC1	890D	1213	1222
SC2	891D	1219	1220
SDBYT	A651	42	1447 1468 1470 1473 1542
SEEK	8B02	1461	1463
SEGSM1	8C28	1651	1318
SENTRY	8E87	1717	1071
SET	8B0D	1466	1467
SPACE	8342	515	236 282 514 567 709 724 884 906 1106
SPCP3	8345	517	513
SPC2	833F	514	592
SPEXIT	86A8	925	945 947
SP2B	8698	918	912
SP2C	86A3	923	954
SP2D	86AB	926	924
SP2E	86B8	931	928
SP2F	86BA	932	930
SR	A65B	57	175 606
STDVAL	8C69	1711	1472
STD2	8619	862	****
STOCOM	8120	235	251
SVBRK	804A	137	1761
SVBYTE	86F4	955	936 938 940 943
SVIRQ	8029	123	1762
SVNMI	809B	177	1763
SWITCH	8B69	1515	****
SWLP	8B6C	1516	1519

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
SWLP2	8BBC	1548	1551
SYM	8BD6	1562	1249
S1NG	87DE	1068	1080
S13B	87CD	1059	993
S13C	87D3	1062	1075
S23B	87E6	1072	1060
TABLE	8BD6	1563	****
TAPERR	848E	667	630 638 644 658 662 665
TECHO	A653	45	1152 1358 1360 1368 1386
TEXT	8A06	1331	1305
TIN	8A6A	1381	1399
TLP1	8A83	1391	1392
TOUT	8AA0	1406	1714
TOUTFL	A654	47	1377 1383 1436 1543 1545
TRACON	80CD	196	189
TRCOFF	80C0	191	1760
TRCVEC	A674	75	195
TRMTBL	8C6F	1713	1548
TSTAT	8B3C	1489	1715
TTY	8BA7	1540	1724
TV	A656	49	181 528
TVNZ	80AF	185	182
TXTMOV	8A0B	1335	1338
UBRKV	A676	77	****
UBRKVC	A676	76	77
UIRQV	A678	79	****
UIRQVC	A678	78	79
URCVEC	A66C	69	698 1164
URSVEC	**A669**	**68**	**271**
USRENT	8035	128	****
VADDR	8646	882	897
VALS	8BC6	1557	1293 1561
VALSP2	8BC8	1561	1291
VECSW	8BB7	1546	1520
VERZ	84B5	683	681
VER1	8596	796	689 783
VER2	863C	878	807 874
VOCK	8664	895	****
V1	866D	**900**	890 892 903
V2	864B	884	894
WARM	8003	98	101 151 190 1526
WPR1B	85F7	841	838
WRAP	82B8	446	452
XR	A65E	60	153 615
YR	A65F	61	154 614
ZERCK	832E	506	625 881 957

```
LINE #  LOC     CODE       LINE

0002  0000              ;****** VERSION 2 4/13/79 "SY1.1"
0003  0000              ;****** COPYRIGHT 1978 SYNERTEK SYSTEMS CORPORATION
0004  0000              ;******
0005  0000              BDRY    =$F8            ;0/1 BDRY FOR READ TIMING
0006  0000              OLD     =$F9            ;HOLD PREV INPUT LEVEL IN GETTR
0007  0000              CHAR    =$FC            ;CHAR ASSY AND DISASSY
0008  0000              MODE    =$FD            ;BIT7=1 IS HS, 0 IS KIM
0009  0000                                      ;... BIT6=1 - IGNORE DATA
0010  0000              BUFADL  =$FE            ;RUNNING BUFFER ADR
0011  0000              BUFADH  =$FF
0012  0000              TAPDEL  =$A630          ;HI SPEED TAPE DELAY
0013  0000              KMBDRY  =$A631          ;KIM READ BDRY
0014  0000              HSBDRY  =$A632          ;HS READ BDRY
0015  0000              TAPET1  =$A635          ;HS FIRST 1/2 BIT
0016  0000              TAPET2  =$A63C          ;HS SECOND 1/2 BIT
0017  0000              SCR6    =$A636          ;SCR 6
0018  0000              SCR7    =$A637          ;SCR 7
0019  0000              SCR8    =$A638          ;SCR 8
0020  0000              SCR9    =$A639          ;SCR 9

0022  0000                      *=$A64A
0023  A64A              EAL     *=*+1           ;P3L - END ADDR +1 (LO)
0024  A64B              EAH     *=*+1           ;P3H -  (HI)
0025  A64C              SAL     *=*+1           ;P2L - START ADDR  (LO)
0026  A64D              SAH     *=*+1           ;P2H -  (HI)
0027  A64E              ID      *=*+1           ;P1L -  ID

0029  A64F              EOT     = $04
0030  A64F              SYN     = $16
0031  A64F              TPBIT   =%1000          ;BIT 3 IS ENABLE/DISABLE TO DECODER
0032  A64F              FRAME   =$FF            ;ERROR MSG # FOR FRAME ERROR
0033  A64F              CHECK   =$CC            ;ERROR # FOR CHECKSUM ERROR
0034  A64F              LSTCHR  =$2F            ;LAST CHAR NOT '/'
0035  A64F              NONHEX  =$FF            ;NON HEX CHAR IN KIM REC

0037  A64F              ACCESS  =$8B86          ;UNWRITE PROTECT SYSTEM RAM
0038  A64F              P2SCR   =$829C          ;MOVE P2 TO $FF,$FE IN PAGE ZERO
0039  A64F              ZERCK   =$832E          ;MOVE ZERO TO CHECK SUM
0040  A64F              CONFIG  =$89A5          ;CONFIGURE I/O

0042  A64F              ; I/O - TAPE ON/OFF IS CB2 ON VIA 1 (A000)
0043  A64F              ;       TAPE IN IS PB6 ON VIA 1 (A000)
0044  A64F              ;       TAPE OUT IS CODE 7 TO DISPLAY DECODER, THRU 6532,
0045  A64F              ;            PB0-PB3 (A400)

0047  A64F              VIAACR  =$A00B
0048  A64F              VIAPCR  =$A00C          ;CONTROL CB2 TAPE ON/OFF, POR
0049  A64F              TPOUT   =$A402
0050  A64F              TAPOUT  =TPOUT
0051  A64F              DDROUT  =$A403
0052  A64F              TAPIN   =$A000
0053  A64F              DDRIN   =$A002
0054  A64F              TIMER   =$A406          ;6532 TIMER READ
0055  A64F              TIM8    =$A415          ;6532 TIMER SET (8US)
0056  A64F              DDRDIG  =$A401
```

```
LINE # LOC      CODE        LINE

0057   A64F                DIG     =$A400

0059   A64F                ; LOADT ENTER W/ID IN PARM 2, MODE IN [Y]

0061   A64F                        *=$8C78
0062   8C78   20 A9 8D     LOADT   JSR START       ;INITIALIZE
0063   8C7B   20 52 8D     LOADT2  JSR SYNC        ;GET IN SYNC
0064   8C7E   20 E1 8D     LOADT4  JSR RDCHTX
0065   8C81   C9 2A                CMP #'*         ;START OF DATA?
0066   8C83   F0 06                BEQ LOAD11
0067   8C85   C9 16                CMP #SYN        ;NO - SYN?
0068   8C87   D0 F2                BNE LOADT2      ;IF NOT, RESTART SYNC SEARCH
0069   8C89   F0 F3                BEQ LOADT4      ;IF YES, KEEP LOOKING FOR *

0071   8C8B   06 FD       LOAD11  ASL MODE        ;GET MODE IN A, CLEAR BIT6
0072   8C8D   6A                   ROR A
0073   8C8E   85 FD                STA MODE
0074   8C90   20 26 8E             JSR RDBYTX      ;READ ID BYTE ON TAPE
0075   8C93   8D 00 A4             STA DIG         ;DISPLAY ON LED (NOT DECODED)
0076   8C96   CD 4E A6             CMP ID          ;COMPARE WITH REQUESTED ID
0077   8C99   F0 29                BEQ LOADT5      ;LOAD IF EQUAL
0078   8C9B   AD 4E A6             LDA ID          ;COMPARE WITH 0
0079   8C9E   C9 00                CMP #0
0080   8CA0   F0 22                BEQ LOADT5      ;IF 0, LOAD ANYWAY
0081   8CA2   C9 FF                CMP #$FF        ;COMPARE WITH FF
0082   8CA4   F0 07                BEQ LOADT6      ;IF FF, USE REQUEST SA TO LOAD

0084   8CA6   24 FD                BIT MODE        ;UNWANTED RECORD, KIM OR HS?
0085   8CA8   30 16                BMI HWRONG
0086   8CAA   4C 7B 8C             JMP LOADT2      ;IF KIM, RESTART SEARCH

0088   8CAD                ; SA (&EA IF USED) COME FROM REQUEST, DISCARD TAPE VALUES
0089   8CAD                ;   (BUFAD ALREADY SET TO SA BY 'START')
0090   8CAD                ;
0091   8CAD   20 74 8E     LOADT6  JSR RDCHK       ;GET SAL FROM TAPE
0092   8CB0   20 74 8E             JSR RDCHK       ;GET SAH FROM TAPE
0093   8CB3   24 FD                BIT MODE        ;HS OR KIM?
0094   8CB5   10 52                BPL LOADT7      ;IF KIM, START READING DATA
0095   8CB7   20 74 8E             JSR RDCHK       ;HS, GET EAH, EAL FROM TAPE
0096   8CBA   20 74 8E             JSR RDCHK       ; ... BUT IGNORE
0097   8CBD   4C DE 8C             JMP LT7H        ;START READING HS DATA

0099   8CC0                ; SA (& EA IF USED) COME FROM TAPE, SA REPLACES BUFAD

0101   8CC0   A9 C0       HWRONG  LDA #$C0        ;READ THRU TO GET TO NEXT REC
0102   8CC2   85 FD                STA MODE        ;BUT DON'T CHECK CKSUM, NO FRAME ERR

0104   8CC4   20 74 8E     LOADT5  JSR RDCHK       ;GET SAL FROM TAPE
0105   8CC7   85 FE                STA BUFADL      ;PUT IN BUF START L
0106   8CC9   20 74 8E             JSR RDCHK       ;SAME FOR SAH
0107   8CCC   85 FF                STA BUFADH
0108   8CCE                ;(SAL - H STILL HAVE REQUEST VALUE)
0109   8CCE   24 FD                BIT MODE        ;HS OR KIM?
0110   8CD0   10 37                BPL LOADT7      ;IF KIM, START READING RECORD
0111   8CD2   20 74 8E             JSR RDCHK       ;HS, GET & SAVE EAL,EAH
```

```
LINE # LOC     CODE        LINE

0112  8CD5  8D 4A A6          STA EAL
0113  8CD8  20 74 8E          JSR RDCHK
0114  8CDB  8D 4B A6          STA EAH

0116  8CDE                 ; READ HS DATA

0118  8CDE  20 E5 8D    LT7H  JSR RDBYTH      ;GET NEXT BYTE
0119  8CE1  A6 FE            LDX BUFADL      ;CHECK FOR END OF DATA + 1
0120  8CE3  EC 4A A6          CPX EAL
0121  8CE6  D0 07            BNE LT7HA
0122  8CE8  A6 FF            LDX BUFADH
0123  8CEA  EC 4B A6          CPX EAH
0124  8CED  F0 14            BEQ LT7HB
0125  8CEF  20 77 8E    LT7HA JSR CHKT        ;NOT END. UPDATE CHECKSUM
0126  8CF2  24 FD            BIT MODE        ;WRONG RECORD?
0127  8CF4  70 04            BVS LT7HC       ;IF SO, DONT STORE BYTE
0128  8CF6  A0 00            LDY #0          ;STORE BYTE
0129  8CF8  91 FE            STA (BUFADL),Y
0130  8CFA  E6 FE      LT7HC INC BUFADL      ;BUMP BUFFER ADDR
0131  8CFC  D0 E0            BNE LT7H
0132  8CFE  E6 FF            INC BUFADH      ;CARRY
0133  8D00  4C DE 8C          JMP LT7H

0135  8D03  C9 2F      LT7HB CMP #'/         ;EA, MUST BE "/"
0136  8D05  D0 29            BNE LCERR       ;LAST CHAR NOT '/'
0137  8D07  F0 15            BEQ LOADT8      ;(ALWAYS)

0139  8D09                 ; READ KIM DATA

0141  8D09  20 2A 8E    LOADT7 JSR RDBYT
0142  8D0C  B0 26            BCS LDT7A       ;NONHEX OR LAST CHAR?
0143  8D0E  20 77 8E          JSR CHKT        ;UPDATE CHECKSUM (PACKED BYTE)
0144  8D11  A0 00            LDY #0          ;STORE BYTE
0145  8D13  91 FE            STA (BUFADL),Y
0146  8D15  E6 FE            INC BUFADL      ;BUMP BUFFER ADR
0147  8D17  D0 F0            BNE LOADT7      ;CARRY?
0148  8D19  E6 FF            INC BUFADH
0149  8D1B  4C 09 8D          JMP LOADT7

0151  8D1E                 ; TEST CHECKSUM & FINISH

0153  8D1E                 LOADT8 =*
0154  8D1E  20 26 8E    LT8A  JSR RDBYTX      ;CHECK SUM
0155  8D21  CD 36 A6          CMP SCR6
0156  8D24  D0 16            BNE CKERR
0157  8D26  20 26 8E          JSR RDBYTX
0158  8D29  CD 37 A6          CMP SCR7
0159  8D2C  D0 0E            BNE CKERR       ;CHECK SUM ERROR
0160  8D2E  F0 11            BEQ OKEXIT      ;(ALWAYS)

0162  8D30  A9 2F      LCERR LDA #LSTCHR     ;LAST CHAR IS NOT '/'
0163  8D32  D0 0A            BNE NGEXIT      ;(ALWAYS)

0165  8D34  C9 2F      LDT7A CMP #'/         ;LAST OR NONHEX?
0166  8D36  F0 E6            BEQ LOADT8      ;LAST
```

```
LINE # LOC      CODE        LINE

0167  8D38                  FRERR                       ;FRAMING ERROR
0168  8D38  A9 FF            NHERR  LDA #NONHEX         ;KIM ONLY, NON HEX CHAR READ
0169  8D3A  D0 02                   BNE NGEXIT          ;(ALWAYS)

0171  8D3C  A9 CC            CKERR  LDA #CHECK         ;CHECKSUM ERROR

0173  8D3E  38                NGEXIT SEC                ;ERROR INDICATOR TO MONITOR IS CARRY
0174  8D3F  B0 01                   BCS EXIT            ;(ALWAYS)

0176  8D41  18                OKEXIT CLC                ;NO ERROR

0178  8D42  24 FD             EXIT   BIT MODE
0179  8D44  50 08                   BVC EX10            ;READING WRONG REC?
0180  8D46  A0 80                   LDY #$80
0181  8D48  4C 78 8C                JMP LOADT           ;RESTART SEARCH

0183  8D4B  68                USRREQ PLA                ;USER REQUESTS EXIT
0184  8D4C  68                       PLA
0185  8D4D  38                       SEC
0186  8D4E  A2 CC             EX10   LDX #$CC
0187  8D50  D0 69                   BNE STTC            ;STOP TAPE, RETURN
0188  8D52  AD 02 A0          SYNC   LDA DDRIN           ;CHANGE DATA DIRECTION
0189  8D55  29 BF                   AND #$BF
0190  8D57  8D 02 A0                STA DDRIN
0191  8D5A  A9 00                   LDA #0
0192  8D5C  8D 0B A0                STA VIAACR
0193  8D5F  AD 31 A6                LDA KMBDRY          ;SET UP BOUNDARY
0194  8D62  24 FD                   BIT MODE
0195  8D64  10 03                   BPL SY100
0196  8D66  AD 32 A6                LDA HSBDRY
0197  8D69  85 F8             SY100  STA BDRY
0198  8D6B  A9 6D                   LDA #$6D
0199  8D6D  8D 00 A4                STA DIG             ;INDICATE NO SYNC ON LEDS
0200  8D70  A5 FD                   LDA MODE            ;TURN ON OUT OF SYNC MODE
0201  8D72  09 40                   ORA #$40            ;BIT6
0202  8D74  85 FD                   STA MODE
0203  8D76  A9 7F             SYNC5  LDA #$7F           ;TEST FOR CR DOWN ON HKB
0204  8D78  8D 01 A4                STA DDRDIG
0205  8D7B  2C 00 A4                BIT DIG
0206  8D7E  10 CB                   BPL USRREQ          ;CR KEY DOWN - EXIT (ERROR)
0207  8D80  20 9F 8D                JSR SYNBIT
0208  8D83  66 FC                   ROR CHAR
0209  8D85  A5 FC                   LDA CHAR
0210  8D87  C9 16                   CMP #SYN
0211  8D89  D0 EB                   BNE SYNC5
0212  8D8B  A2 0A             SYNC10 LDX #10            ;NOW MAKE SURE CAN GET 10 SYNS
0213  8D8D  20 E1 8D                JSR RDCHTX
0214  8D90  C9 16                   CMP #SYN
0215  8D92  D0 E2                   BNE SYNC5
0216  8D94  CA                       DEX
0217  8D95  D0 F6                   BNE SYNC10+2
0218  8D97  8E 00 A4                STX DIG             ;TURN OFF DISPLAY
0219  8D9A  CA                       DEX                 ;X=$FF
0220  8D9B  8E 01 A4                STX DDRDIG
0221  8D9E  60                       RTS
```

```
LINE # LOC     CODE        LINE

0222  8D9F                ;SYNBIT - GET BIT IN SYN SEARCH. IF HS, ENTER WITH
0223  8D9F                ;  TIMER STARTED BY PREV BIT. BIT RETURNED IN CARRY.

0225  8D9F  24 FD         SYNBIT BIT MODE          ;KIM OR HS?
0226  8DA1  10 69                BPL RDBITK        ;KIM
0227  8DA3  20 CA 8D      SYB10  JSR GETTR         ;HS
0228  8DA6  B0 22                BCS GETTR         ;IF SHORT, GET NEXT TRANS
0229  8DA8  60                    RTS               ;BIT IS ZERO

0231  8DA9  84 FD         START  STY MODE          ;MODE PARM PASSED IN [Y]
0232  8DAB  20 86 8B             JSR ACCESS        ;FIX BASIC WARM START BUG
0233  8DAE  A9 09                LDA #9
0234  8DB0  20 A5 89             JSR CONFIG        ;PARTIAL I/O CONFIGURATION
0235  8DB3  20 2E 83             JSR ZERCK         ;ZERO THE CHECK SUM
0236  8DB6  20 9C 82             JSR P2SCR         ;MOVE SA TO FE,FF IN PAGE ZERO
0237  8DB9  A2 EC                LDX #$EC
0238  8DBB  8E 0C A0      STTC   STX VIAPCR        ;TAPE ON
0239  8DBE  60                    RTS

0241  8DBF                ; GETTR - GET TRANSITION TIME FROM 6532 CLOCK
0242  8DBF                ; DESTROYS A,Y

0244  8DBF  A9 00         KGETTR LDA #0            ;KIM GETTR - GET FULL CYCLE
0245  8DC1  85 F9                STA OLD           ;FORCE GETTR POLARITY
0246  8DC3  AD 00 A0      KG100  LDA TAPIN         ;WAIT TIL INPUT LO
0247  8DC6  29 40                AND #$40
0248  8DC8  D0 F9                BNE KG100

0250  8DCA  A0 FF         GETTR  LDY #$FF
0251  8DCC  AD 00 A0      NOTR   LDA TAPIN
0252  8DCF  29 40                AND #$40
0253  8DD1  C5 F9                CMP OLD
0254  8DD3  F0 F7                BEQ NOTR          ;NO CHANGE
0255  8DD5  85 F9                STA OLD
0256  8DD7  AD 06 A4             LDA TIMER
0257  8DDA  8C 15 A4             STY TIM8          ;RESTART CLOCK
0258  8DDD  18                    CLC
0259  8DDE  65 F8                ADC BDRY
0260  8DE0  60                    RTS

0262  8DE1  24 FD         RDCHTX BIT MODE          ;READ HS OR KIM CHARACTER
0263  8DE3  10 7A                BPL RDCHT         ;KIM

0265  8DE5                ; RDBYTH - READ HS BYTE
0266  8DE5                ; Y DESTROYED, BYTE RETURNED IN CHAR AND A
0267  8DE5                ; TIME FROM ONE CALL TO NEXT MUST BE LESS THAN
0268  8DE5                ;    START BIT TIME (TIMER STILL RUNNING)

0270  8DE5  8E 38 A6      RDBYTH STX SCR8          ;SAVE X
0271  8DE8  A2 08                LDX #8
0272  8DEA  20 CA 8D             JSR GETTR         ;GET START BIT TIME
0273  8DED  B0 14                BCS RDBH90        ;IF NOT 0, FRAMING ERR
0274  8DEF  20 CA 8D      RDBH10 JSR GETTR         ;GET BIT IN CARRY
0275  8DF2  90 04                BCC RDASSY
0276  8DF4  20 CA 8D             JSR GETTR         ;BIT IS ONE, WAIT HALF CYC
```

```
LINE # LOC     CODE        LINE

0277  8DF7  38              SEC             ;MAKE SURE "1"
0278  8DF8  66 FC       RDASSY ROR CHAR
0279  8DFA  CA              DEX
0280  8DFB  D0 F2            BNE RDBH10
0281  8DFD  A5 FC            LDA CHAR        ;GET IN ACC
0282  8DFF  AE 38 A6          LDX SCR8        ;RESTORE X
0283  8E02  60              RTS
0284  8E03  24 FD       RDBH90 BIT MODE        ;NO ERR IF NOT IN SYNC
0285  8E05  70 F8            BVS RDBH90-4    ;OR READING WRONG REC
0286  8E07  68              PLA             ;FIX STACK
0287  8E08  68              PLA
0288  8E09  4C 38 8D          JMP FRERR

0290  8E0C              ; RDBITK - READ KIM BIT - X,Y,A DESTROYED, BIT RETURNED IN C

0292  8E0C  20 BF 8D   RDBITK JSR KGETTR       ;WAIT FOR LF
0293  8E0F  B0 FB            BCS RDBITK
0294  8E11  20 BF 8D          JSR KGETTR      ;GET SECOND
0295  8E14  B0 F6            BCS RDBITK
0296  8E16  A2 00            LDX #0
0297  8E18  E8         RDB100 INX             ;COUNT LF FULL CYCLES
0298  8E19  20 BF 8D          JSR KGETTR
0299  8E1C  90 FA            BCC RDB100
0300  8E1E  20 BF 8D          JSR KGETTR      ;GET SECOND
0301  8E21  90 F5            BCC RDB100
0302  8E23  E0 08            CPX #$08        ;GET BIT TO CARRY
0303  8E25  60              RTS

0305  8E26  24 FD       RDBYTX BIT MODE        ;READ HS OR KIM BYTE
0306  8E28  30 BB            BMI RDBYTH      ;HS

0308  8E2A  20 5F 8E    RDBYT  JSR RDCHT       ;READ KIM BYTE INTO CHAR AND A
0309  8E2D  C9 2F            CMP #'/         ;READ ONE CHAR IF LAST
0310  8E2F  F0 2C            BEQ PACKT3      ;SET CARRY AND RETURN
0311  8E31  20 3C 8E          JSR PACKT
0312  8E34  B0 26            BCS RDRTN       ;NON HEX CHAR?
0313  8E36  AA              TAX             ;SAVE MSD
0314  8E37  20 5F 8E          JSR RDCHT
0315  8E3A  86 FC            STX CHAR        ;MOVE MSD TO CHAR
0316  8E3C              ; AND FALL INTO PACKT AGAIN

0318  8E3C              ;PACKT - ASCII HEX TO 4 BITS
0319  8E3C              ;INPUT IN A, OUTPUT IN CHAR AND A, CARRY SET = NON HEX

0321  8E3C  C9 30       PACKT  CMP #$30        ;LT "0"?
0322  8E3E  90 1D            BCC PACKT3
0323  8E40  C9 47            CMP #$47        ;GT "F" ?
0324  8E42  B0 19            BCS PACKT3
0325  8E44  C9 40            CMP #$40        ;A-F?
0326  8E46  F0 15            BEQ PACKT3      ;40 NOT VALID
0327  8E48  90 03            BCC PACKT1
0328  8E4A  18              CLC
0329  8E4B  69 09            ADC #9
0330  8E4D  2A           PACKT1 ROL A           ;GET LSD INTO LEFT NIBBLE
0331  8E4E  2A              ROL A
```

```
LINE #  LOC    CODE        LINE

0332  8E4F  2A            ROL A
0333  8E50  2A            ROL A
0334  8E51  A0 04         LDY #4
0335  8E53  2A         PACKT2 ROL A          ;ROTATE 1 BIT AT A TIME INTO CHAR
0336  8E54  26 FC            ROL CHAR
0337  8E56  88              DEY
0338  8E57  D0 FA            BNE PACKT2
0339  8E59  A5 FC            LDA CHAR        ;GET INTO ACCUM ALSO
0340  8E5B  18              CLC             ;OK
0341  8E5C  60         RDRTN  RTS
0342  8E5D  38         PACKT3 SEC             ;NOT HEX
0343  8E5E  60              RTS

0345  8E5F            ; RDCHT - READ KIM CHAR
0346  8E5F            ; PRESERVES X, RETURNS CHAR IN CHAR (W/PARITY)
0347  8E5F            ; AND A (W/O PARITY)

0349  8E5F  8A         RDCHT  TXA             ;SAVE X
0350  8E60  48              PHA
0351  8E61  A9 FF            LDA #$FF        ;USE A TO COUNT BITS (BY SHIFTING)
0352  8E63  48         KBITS  PHA             ;SAVE COUNTER
0353  8E64  20 0C 8E          JSR RDBITK
0354  8E67  66 FC            ROR CHAR
0355  8E69  68              PLA
0356  8E6A  0A              ASL A
0357  8E6B  D0 F6            BNE KBITS       ;DO 8 BITS
0358  8E6D  68              PLA             ;RESTORE X
0359  8E6E  AA              TAX
0360  8E6F  A5 FC            LDA CHAR
0361  8E71  2A              ROL A
0362  8E72  4A              LSR A           ;DROP PARITY
0363  8E73  60              RTS

0365  8E74            ; RDCHK - READ ONE BYT, INCLUDE IN CKSUM

0367  8E74  20 26 8E   RDCHK  JSR RDBYTX      ;FALL INTO CHKT

0369  8E77            ; CHKT - UPDATE CHECK SUM FROM BYTE IN A
0370  8E77            ; DESTROYS Y

0372  8E77  A8         CHKT   TAY             ;SAVE ACCUM
0373  8E78  18              CLC
0374  8E79  6D 36 A6          ADC SCR6
0375  8E7C  8D 36 A6          STA SCR6
0376  8E7F  90 03            BCC CHKT10
0377  8E81  EE 37 A6          INC SCR7        ;BUMP HI BYTE
0378  8E84  98         CHKT10 TYA             ;RESTORE A
0379  8E85  60              RTS


0381  8E86  FF              .BYT $FF          ;NOT USED
0382  8E87                  *=$8E87           ;KEEP OLD ENTRY POINT
0383  8E87  20 A9 8D   DUMPT  JSR START       ;INIT VIA & CKSUM, SA TO BUFAD & STAR
0384  8E8A  A9 07            LDA #7          ;CODE FOR TAPE OUT
0385  8E8C  8D 02 A4          STA TAPOUT      ;BIT 3 USED FOR HI/LO
```

```
LINE #  LOC     CODE        LINE

0386  8E8F  A2 01           LDX #1          ;KIM DELAY CONSTANT (OUTER)
0387  8E91  A4 FD           LDY MODE        ;128 KIM, 0 HS
0388  8E93  10 03           BPL DUMPT1      ;KIM - DO 128 SYNS
0389  8E95  AE 30 A6        LDX TAPDEL      ;HS INITIAL DELAY (OUTER)
0390  8E98  8A        DUMPT1 TXA
0391  8E99  48               PHA
0392  8E9A  A9 16      DMPT1A LDA #SYN
0393  8E9C  20 0A 8F          JSR OUTCTX
0394  8E9F  88                DEY
0395  8EA0  D0 F8             BNE DMPT1A      ;INNER LOOP (HS OR KIM)
0396  8EA2  68                PLA
0397  8EA3  AA                TAX
0398  8EA4  CA                DEX
0399  8EA5  D0 F1             BNE DUMPT1
0400  8EA7  A9 2A             LDA #'*         ;WRITE START
0401  8EA9  20 0A 8F          JSR OUTCTX

0403  8EAC  AD 4E A6          LDA ID          ;WRITE ID
0404  8EAF  20 3F 8F          JSR OUTBTX

0406  8EB2  AD 4C A6          LDA SAL         ;WRITE SA
0407  8EB5  20 3C 8F          JSR OUTBCX
0408  8EB8  AD 4D A6          LDA SAH
0409  8EBB  20 3C 8F          JSR OUTBCX

0411  8EBE
0412  8EBE  24 FD             BIT MODE        ;KIM OR HS?
0413  8EC0  10 0C             BPL DUMPT2

0415  8EC2  AD 4A A6          LDA EAL         ;HS. WRITE EA
0416  8EC5  20 3C 8F          JSR OUTBCX
0417  8EC8  AD 4B A6          LDA EAH
0418  8ECB  20 3C 8F          JSR OUTBCX

0420  8ECE  A5 FE      DUMPT2 LDA BUFADL      ;CHECK FOR LAST BYTE
0421  8ED0  CD 4A A6          CMP EAL
0422  8ED3  D0 25             BNE DUMPT4
0423  8ED5  A5 FF             LDA BUFADH
0424  8ED7  CD 4B A6          CMP EAH
0425  8EDA  D0 1E             BNE DUMPT4

0427  8EDC  A9 2F             LDA #'/         ;LAST. WRITE "/"
0428  8EDE  20 0A 8F          JSR OUTCTX
0429  8EE1  AD 36 A6          LDA SCR6        ;WRITE CHECK SUM
0430  8EE4  20 3F 8F          JSR OUTBTX
0431  8EE7  AD 37 A6          LDA SCR7
0432  8EEA  20 3F 8F          JSR OUTBTX

0434  8EED  A9 04             LDA #EOT        ;WRITE TWO EOT'S
0435  8EEF  20 3F 8F          JSR OUTBTX
0436  8EF2  A9 04             LDA #EOT
0437  8EF4  20 3F 8F          JSR OUTBTX

0439  8EF7              DT3E   = * (SET "OK" MARK)
0440  8EF7  4C 41 8D          JMP OKEXIT
```

```
LINE # LOC     CODE        LINE

0442  8EFA  A0 00       DUMPT4 LDY #0          ;GET BYTE
0443  8EFC  B1 FE              LDA (BUFADL),Y
0444  8EFE  20 3C 8F           JSR OUTBCX       ;WRITE IT W/CHK SUM
0445  8F01  E6 FE              INC BUFADL       ;BUMP BUFFER ADDR
0446  8F03  D0 C9              BNE DUMPT2
0447  8F05  E6 FF              INC BUFADH       ;CARRY
0448  8F07  4C CE 8E           JMP DUMPT2
0449  8F0A  24 FD       OUTCTX BIT MODE         ;HS OR KIM?
0450  8F0C  10 48              BPL OUTCHT       ;KIM

0452  8F0E              ;  OUTBTH - NO CLOCK
0453  8F0E              ; A,X DESTROYED
0454  8F0E              ; MUST RESIDE ON ONE PAGE - TIMING CRITICAL
0455  8F0E  A2 09       OUTBTH LDX #9           ;8 BITS + START BIT
0456  8F10  8C 39 A6           STY SCR9
0457  8F13  85 FC              STA CHAR
0458  8F15  AD 02 A4           LDA TAPOUT       ;GET PREV LEVEL
0459  8F18  46 FC       GETBIT LSR CHAR
0460  8F1A  49 08              EOR #TPBIT
0461  8F1C  8D 02 A4           STA TAPOUT       ;INVERT LEVEL
0462  8F1F              ; *** HERE STARTS FIRST HALF CYCLE
0463  8F1F  AC 35 A6           LDY TAPET1
0464  8F22  88          A416   DEY              ;TIME FOR THIS LOOP IS 5Y-1
0465  8F23  D0 FD              BNE A416
0466  8F25  90 12              BCC NOFLIP       ;NOFLIP IF BIT ZERO
0467  8F27  49 08              EOR #TPBIT       ;BIT IS ONE - INVERT OUTPUT
0468  8F29  8D 02 A4           STA TAPOUT
0469  8F2C              ; *** END OF FIRST HALF CYCLE
0470  8F2C  AC 3C A6    B416   LDY TAPET2
0471  8F2F  88          B416B  DEY              ;LENGTH OF LOOP IS 5Y-1
0472  8F30  D0 FD              BNE B416B
0473  8F32  CA                 DEX
0474  8F33  D0 E3              BNE GETBIT       ;GET NEXT BIT (LAST IS 0 START BIT)
0475  8F35  AC 39 A6           LDY SCR9         ; (BY 9 BIT LSR)
0476  8F38  60                 RTS
0477  8F39  EA          NOFLIP NOP              ;TIMING
0478  8F3A  90 F0              BCC B416         ;(ALWAYS)
0479  8F3C              ;
0480  8F3C  20 77 8E    OUTBCX JSR CHKT         ;WRITE HS OR KIM BYTE & CKSUM
0481  8F3F  24 FD       OUTBTX BIT MODE         ;WRITE HS OR KIM BYTE
0482  8F41  30 CB              BMI OUTBTH       ;HS

0484  8F43              ;OUTBTC - OUTPUT ONE KIM BYTE

0486  8F43              OUTBTC =*
0487  8F43  A8          OUTBT  TAY              ;SAVE DATA BYTE
0488  8F44  4A                 LSR A
0489  8F45  4A                 LSR A
0490  8F46  4A                 LSR A
0491  8F47  4A                 LSR A
0492  8F48  20 4B 8F           JSR HEXOUT       ;MORE SIG DIGIT
0493  8F4B              ; FALL INTO HEXOUT

0495  8F4B  29 0F       HEXOUT AND #$0F         ;CVT LSD OF [A] TO ASCII, OUTPUT
0496  8F4D  C9 0A              CMP #$0A
```

```
LINE # LOC     CODE        LINE

0497  8F4F  18                CLC
0498  8F50  30 02             BMI HEX1
0499  8F52  69 07             ADC #$07
0500  8F54  69 30       HEX1  ADC #$30

0502  8F56              ; OUTCHT - OUTPUT ASCII CHAR (KIM)
0503  8F56              ; CLOCK NOT USED
0504  8F56              ; X,Y PRESERVED
0505  8F56              ; MUST RESIDE ON ONE PAGE - TIMING CRITICAL

0507  8F56  8E 38 A6    OUTCHT STX SCR8        ;PRESERVE X
0508  8F59  8C 39 A6           STY SCR9        ;DITTO Y
0509  8F5C  85 FC              STA CHAR
0510  8F5E  A9 FF              LDA #$FF        ;USE FF W/SHIFTS TO COUNT BITS
0511  8F60  48          KIMBIT PHA             ;SAVE BIT CTR
0512  8F61  AD 02 A4           LDA TPOUT       ;GET CURRENT OUTPUT LEVEL
0513  8F64  46 FC              LSR CHAR        ;GET DATA BIT IN CARRY
0514  8F66  A2 12              LDX #18         ;ASSUME 'ONE'
0515  8F68  B0 02              BCS HF
0516  8F6A  A2 24              LDX #36         ;BIT IS ZERO
0517  8F6C  A0 19        HF    LDY #25
0518  8F6E  49 08              EOR #TPBIT      ;INVERT OUTPUT
0519  8F70  8D 02 A4           STA TPOUT
0520  8F73  88          HFP1  DEY              ;PAUSE FOR 138 USEC
0521  8F74  D0 FD              BNE HFP1
0522  8F76  CA                DEX             ;COUNT HALF CYCS OF HF
0523  8F77  D0 F3              BNE HF
0524  8F79  A2 18        LF    LDX #24         ;ASSUME BIT IS ONE
0525  8F7B  B0 02              BCS LF20
0526  8F7D  A2 0C              LDX #12         ;BIT IS ZERO
0527  8F7F  A0 27        LF20  LDY #39
0528  8F81  49 08              EOR #TPBIT     ;INVERT OUTPUT
0529  8F83  8D 02 A4           STA TPOUT
0530  8F86  88          LFP1  DEY              ;PAUSE FOR 208 USEC
0531  8F87  D0 FD              BNE LFP1
0532  8F89  CA                DEX             ;COUNT HALF CYCS
0533  8F8A  D0 F3              BNE LF20
0534  8F8C  68                PLA             ;RESTORE BIT CTR
0535  8F8D  0A                ASL A           ;DECREMENT IT
0536  8F8E  D0 D0              BNE KIMBIT      ;FF SHIFTED 8X = 00
0537  8F90  AE 38 A6           LDX SCR8
0538  8F93  AC 39 A6           LDY SCR9
0539  8F96  98                TYA             ;RESTORE DATA BYTE
0540  8F97  60                RTS


0542  8F98  FF                .BYT $FF,$FF    ;NOT USED
0542  8F99  FF



0544  8F9A              ; REGISTER NAME PATCH
0545  8F9A                    *=$8F9A
0546  8F9A  53                .BYT 'S'
```

```
LINE # LOC     CODE        LINE

0547  8F9B  46                    .BYT 'F'
0548  8F9C  41                    .BYT 'A'
0549  8F9D  58                    .BYT 'X'
0550  8F9E  59                    .BYT 'Y'
0551  8F9F  01                    .BYT $01
0552  8FA0                ;
0553  8FA0                       .END
```

ERRORS = 0000 <0000>

```
SYMBOL TABLE

SYMBOL    VALUE

A416      8F22    ACCESS    8B86    B416      8F2C    B416B     8F2F
BDRY      00F8    BUFADH    00FF    BUFADL    00FE    CHAR      00FC
CHECK     00CC    CHKT      8E77    CHKT10    8E84    CKERR     8D3C
CONFIG    89A5    DDRDIG    A401    DDRIN     A002    DDROUT    A403
DIG       A400    DMPT1A    8E9A    DT3E      8EF7    DUMPT     8E87
DUMPT1    8E98    DUMPT2    8ECE    DUMPT4    8EFA    EAH       A64B
EAL       A64A    EOT       0004    EX10      8D4E    EXIT      8D42
FRAME     00FF    FRERR     8D38    GETBIT    8F18    GETTR     8DCA
HEX1      8F54    HEXOUT    8F4B    HF        8F6C    HFP1      8F73
HSBDRY    A632    HWRONG    8CC0    ID        A64E    KBITS     8E63
KG100     8DC3    KGETTR    8DBF    KIMBIT    8F60    KMBDRY    A631
LCERR     8D30    LDT7A     8D34    LF        8F79    LF20      8F7F
LFP1      8F86    LOAD11    8C8B    LOADT     8C78    LOADT2    8C7B
LOADT4    8C7E    LOADT5    8CC4    LOADT6    8CAD    LOADT7    8D09
LOADT8    8D1E    LSTCHR    002F    LT7H      8CDE    LT7HA     8CEF
LT7HB     8D03    LT7HC     8CFA    LT8A      8D1E    MODE      00FD
NGEXIT    8D3E    NHERR     8D38    NOFLIP    8F39    NONHEX    00FF
NOTR      8DCC    OKEXIT    8D41    OLD       00F9    OUTBCX    8F3C
OUTBT     8F43    OUTBTC    8F43    OUTBTH    8F0E    OUTBTX    8F3F
OUTCHT    8F56    OUTCTX    8F0A    P2SCR     829C    PACKT     8E3C
PACKT1    8E4D    PACKT2    8E53    PACKT3    8E5D    RDASSY    8DF8
RDB100    8E18    RDBH10    8DEF    RDBH90    8E03    RDBITK    8E0C
RDBYT     8E2A    RDBYTH    8DE5    RDBYTX    8E26    RDCHK     8E74
RDCHT     8E5F    RDCHTX    8DE1    RDRTN     8E5C    SAH       A64D
SAL       A64C    SCR6      A636    SCR7      A637    SCR8      A638
SCR9      A639    START     8DA9    STTC      8DBB    SY100     8D69
SYB10     8DA3    SYN       0016    SYNBIT    8D9F    SYNC      8D52
SYNC10    8D8B    SYNC5     8D76    TAPDEL    A630    TAPET1    A635
TAPET2    A63C    TAPIN     A000    TAPOUT    A402    TIM8      A415
TIMER     A406    TPBIT     0008    TPOUT     A402    USRREQ    8D4B
VIAACR    A00B    VIAPCR    A00C    ZERCK     832E
END OF ASSEMBLY

<
```

Synertek Systems Corporation

150 S. WOLFE ROAD, SUNNYVALE, CALIFORNIA 94086 • TEL. (408) 988-5600 • TWX: 910-338-0135