MICRO - MASTER®

MM-8000

8085 MICROPROCESSOR - BASIC SYSTEMS COURSE

COMPUTER THEORY - CONSTRUCTION
AND PROGRAMMING

TABLE OF CONTENTS

PARTS LIST

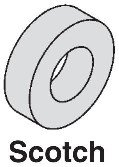
Qty	Description	Part #
	RESISTORS	
□ 9	150Ω 5% 1/4W	131500
□ 1	470Ω 5% 1/4W	134700
□ 1	510Ω 5% 1/4W	135100
□ 3	680Ω 5% 1/4W	136800
□ 1	1kΩ 5% 1/4W	141000
□ 8	1.2kΩ 5% 1/4W	141200
□ 3	2kΩ 5% 1/4W	142000
□ 14	3.9kΩ 5% 1/4W	143900
□ 8	6.8kΩ 5% 1/4W	146800
□ 14	10kΩ 5% 1/4W	151000
□ 5	47kΩ 5% 1/4W	154700
□ 2	68kΩ 5% 1/4W	156800
	CAPACITORS	
□ 1	20pF Discap	212080
□ 1	330pF Discap	223317
□ 10	0.1μF Discap	251010
□ 1	10μF 16V Electrolytic	271054
□ 2	100μF 25V Electrolytic	281045
	SEMICONDUCTORS	
□ 2	A70 Transistor	320070
□ 1	2N3904 Transistor	323904
□ 1	2816A or 9816A IC	332816
□ 1	LM-7805 IC	337805
□ 1	8085 IC	338085
□ 1	8155 IC (see note)	338155
□ 8	LED Diode Red	350002
□ 1	LED Diode Green	350010
□ 2	LED MAN71A / LTS72R / 312AR	350071

Qty	Description	Part #
	MISCELLANEOUS	
□ 1	74HC00 IC	39HC00
□ 1	74HC04 IC	39HC04
□ 1	74HCT573 IC	39T573
□ 1	Transformer Wall Type	440409
□ 1	PC Board	515030
□ 1	Switch PC Mount DPDT	541023
□ 15	Switch Slide Miniature SPDT	541102
□ 29	Swirch Dimple Dome Triangle	546101
□ 1	Heatsink Clip-on	615005
□ 1	Jack DC Power PC Mount	621080
□ 1	Plastic Case Black	623000
□ 4	Screw #8 self-tapping	642862
□ 4	IC Socket 14-pin	664014
□ 2	IC Socket 16-pin	664016
□ 1	IC Socket 20-pin	664020
□ 1	IC Socket 24-pin	664024
□ 2	IC Socket 40-pin	664040
□ 4	Clip PCBMnt	688000
□ 1	Label Keyboad	728000
□ 1	Label Case	728004
□ 1	Solder Tube Lead-free	9LF99

Note: The 8156 IC has been replaced with an 8155. The difference between the two are the active state of chip enable (CE). The CE on the 8155 is active low ($\overline{CE}$) and the 8156 is active high.

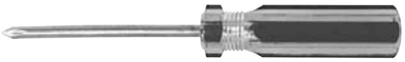
TOOLS REQUIRED

Long Nose Pliers



Scotch Tape
(1/2" wide)

1/4" Blade Screwdriver



Phillips Screwdriver
(small point size)

Diagonal Cutters

Desoldering Pump

VOM, VTVM or DMM Meter (optional)

Pencil Soldering Iron (25-40 watts) or Soldering Station

INTRODUCTION

The objective of this course is to present both the hardware and software necessary to take a few integrated circuits and build a completely functioning microprocessor system. No prior electronic or mathematical background is required to complete this course. If the material presented here is studied carefully,the student will acquire the necessary skills and knowledge to construct and program a complete working computer system. Figure 1 shows a block diagram of a basic computer system. The memory stores the information necessary for the operation of the computer. This information is stored as binary numbers made up of only ones and zeroes. These binary numbers are used for both data and instructions. The instructions are those numbers which specify the operation to be performed by the Central Processing Unit (CPU). The data numbers are those that are operated on by the CPU. The data may represent such things as dollars, weight, quantity, speed and many other physical properties.

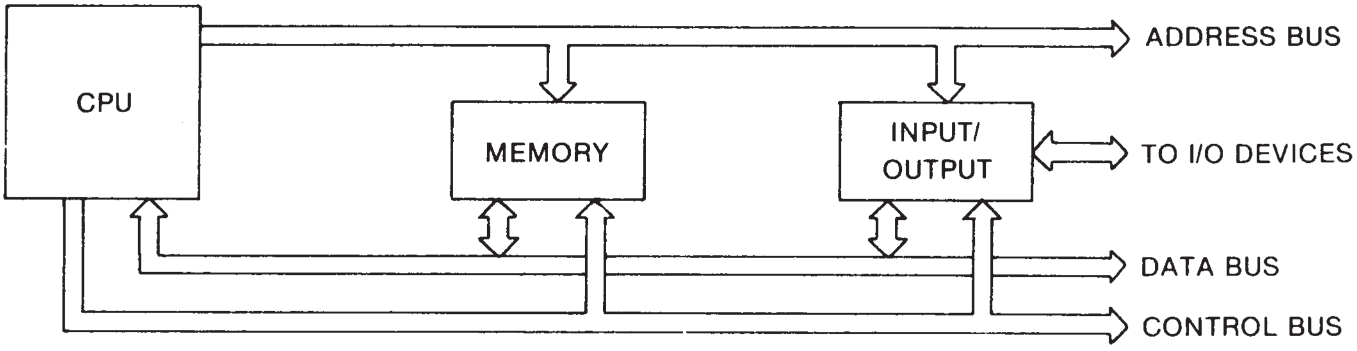


Figure 1

Each number in memory is stored at a specific address. To store a number in memory, the CPU places the address on the address bus (a bus is a wire or group of wires that connects inputs and outputs of different circuits) and the number on the data bus. The control bus is then used to instruct the memory circuit to store the number on the data bus at the address on the address bus. In a similar manner the data in memory can be recalled and sent back to the CPU for processing.

The Input/Output (I/O) section allows the CPU to communicate with outside devices. Information may enter the computer through the input section from devices such as a keyboard, disk drive or joystick. Through the output section,data may be transmitted to devices such as a video display, printer or disk.

The CPU consist of three main sections:

1. **The Arithmetic/Logic Unit (ALU).** The ALU performs arithmetic operations such as addition, subtraction and logical operations such as AND, OR or ROTATE.
2. **Registers.** The registers are temporary storage locations for data, addresses and instructions.
3. **Control.** The control section controls the transfer of information to and from memory and within the CPU.

A computer program is a sequence of instructions designed to perform a particular task. The computer performs these instructions in the order they are stored in memory. To do this,one of the CPU registers is always a program address counter which contains the address of the next instruction to be executed. The computer places the contents of the program address counter on the address bus and reads the number at that location. This number is interpreted as an instruction. The instruction is performed and the program address counter is incremented by one (or more if the instruction occupies more than one memory location). The next instruction in the list is then accessed and performed. An exception occurs when a branch instruction is executed. Branch instructions alter the normal sequence by instructing the computer to replace the contents of the program address counter with another address and to begin processing instructions at the new address. Branch instructions may be conditional or unconditional. Unconditional branches merely perform the address replacement. Conditional branches perform the address replacement only if some prior condition is met. Typical conditions for branching would be; if the contents of some register equals zero, or the result of the last subtraction was a negative number.

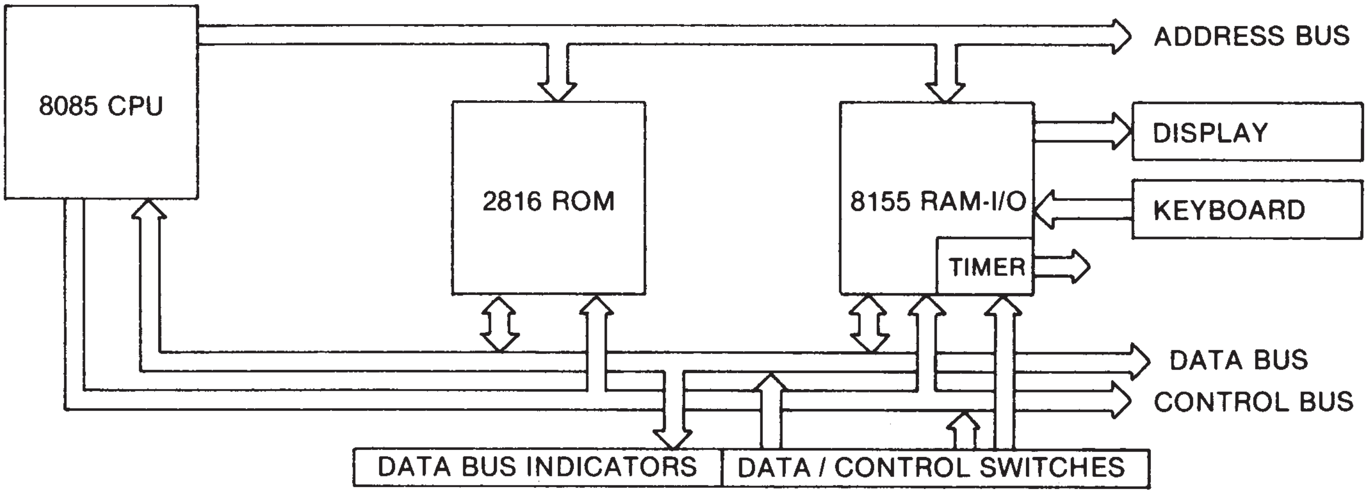


Figure 2

Figure 2 is a simplified block diagram of the MM-8000 system. The system utilizes the 8085 microprocessor as the Central Processor Unit. The 8085 is a single chip 8 bit microprocessor. The memory is in two parts. The 2816 integrated circuit provides 2K bytes of Read Only Memory (ROM). This memory is electronically erasable and may be written to at a slow rate by either the CPU or manually by data and control switches. The 8155 integrated circuit contains 256 bytes of Random Access Memory (RAM). The 8155 also contains the Input/Output (IO) section. This section consists of two 8 bit and one 6 bit I/O port. Inputs to the I/O section are from the MM-8000 keyboard consisting of 16 data and 8 function keys. Outputs from the I/O section are to two 7 segment LED displays. The data bus indicators consist of drive circuits and 8 LEDs which indicate the state of the data bus lines. The data/control switches are tied to the data and control busses and are used to write to or read from the ROM, RAM or IO ports. When reading, the data placed on the data bus by the 8155 or 2816 overrides that of the data switches.

The MM-8000 course consists of 14 lessons. In the first 13 lessons the MM-8000 PC board is built up step by step. As each section is added,its function is explained and experiments are run to demonstrate its operation.

In lesson 1 the binary and hexadecimal number systems used by computers are explained. In lesson 2 the data bus indicators and switches are installed and their operation described and demonstrated.

Lessons 3 thru 5 are concerned with the 8155 RAM-I/O integrated circuit. In lesson 3 the 8155 and several control switches are installed. These switches along with the switches and indicators installed in lesson 2 are used to manually write to and read from the 8156. The volatility of the RAM is also demonstrated. Lesson 4 adds the display. The data and control switches are used to write to both 7 segment LED displays via the 8155 I/O ports. The formation of hexadecimal numbers and letters is demonstrated. Lesson 5 adds the Timer (TIM) switch. This switch is used to manually step the 8155 timer. The operation of the timer in each of its 4 output modes is demonstrated.

Lesson 6 adds the 2816 ROM integrated circuit, the 573 transparent latch (not shown on the block diagram) and the control switches associated with the ROM. The working of the ROM and the necessity of the 573 latch in storing the low order address bits is explained. The data and control switches are used to write to and read from the ROM. The non-volatility of the ROM is demonstrated.

Lesson 7 is a detailed functional description of the 8085 microprocessor. The internal registers, flags, clock generation and interrupts are explained. Timing diagrams are given for different read and write memory cycles. Lesson 8 describes the instruction set of the 8085 microprocessor. The mnemonic and a description of each instruction is given. The description includes the binary code, timing information and the flags affected.

In lesson 9 the monitor program is described and MM-8000 system considerations such as memory allocation and PC board interface are discussed.

In lesson 10 thru 13,programs 1 thru 4 are manually loaded in ROM. Each program is a section of the Monitor program. As each program is loaded its functions are described and its operation is demonstrated. When program 4 is added the MM-8000 system is complete. Under Monitor program control,the keyboard may then be used to enter data into memory and to read and display any memory location.

Lesson 10 adds the 8085 microprocessor and auxiliary circuits. Program 1 is manually loaded into memory and its operation demonstrated. Program 1 turns on the display and copies a section of the Monitor program from ROM to RAM. In lesson 11 program 2 is loaded into memory. Program 2 alternately drives the two 7 segment displays at a rate determined by a delay constant in memory. The alternating display as well as the use of a "look up" table to generate the display characters is explained and demonstrated. In lesson 12 the keyboard is built up on the MM-8000 PC board and program 3 is loaded into memory. Program 3 adds the keyboard scan and data key processing sections of the Monitor program. Data entry and display are demonstrated. In lesson 13 program 4 is manually loaded into memory. Program 4 implements the 8 keyboard function keys. With these keys the Monitor mode may be changed, data may be stored to and read from memory and control may be transferred to other programs. Each of these functions is explained and demonstrated.

In lesson 14 the keyboard and the Monitor program are used to load and execute program 5. In program 5 two hexadecimal numbers entered via the keyboard,are added and the result displayed in the 7 segment displays.

At the completion of the MM-8000 course,the student will have learned the basic operations of the 8085 microprocessor. He will have manually operated each of the MM-8000 systems components, loaded the Monitor program and used it to load and run another computer program. He will have in his possession a fully operational microcomputer system and be ready to apply it in the fields of robotics, speech synthesis, communications, security systems, industrial controls, etc.

CONSTRUCTION

Introduction

The most important factor in assembling your MM-8000 Micro-Master® Kit is good soldering techniques. Using the proper soldering iron is of prime importance. A small pencil type soldering iron of 25 - 40 watts is recommended. **The tip of the iron must be kept clean at all times and well tinned.**

Safety Procedures

- Wear eye protection when soldering and during all phases of construction.
- Locate soldering iron in an area where you do not have to go around it or reach over it.
- **Do not hold solder in your mouth.** Wash your hands thoroughly after handling solder.
- Be sure that there is adequate ventilation present.

Assemble Components

In all of the following assembly steps, the components must be installed on the top side of the PC board unless otherwise indicated. The top legend shows where each component goes. The leads pass through the corresponding holes in the board and are soldered on the foil side.

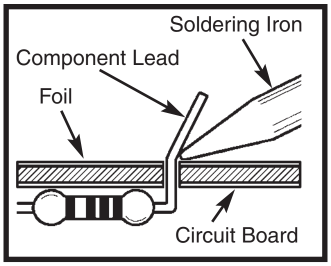
Use only rosin core solder.

DO NOT USE ACID CORE SOLDER!

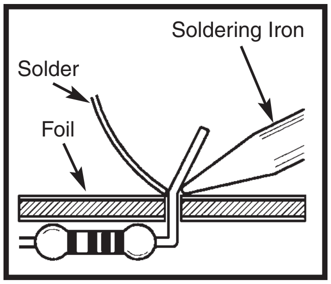
What Good Soldering Looks Like

A good solder connection should be bright, shiny, smooth, and uniformly flowed over all surfaces.

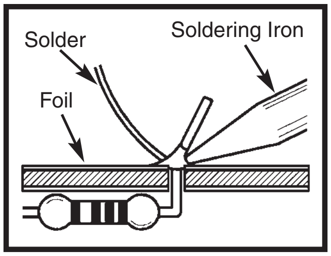
1. Solder all components from the copper foil side only. Push the soldering iron tip against both the lead and the circuit board foil.



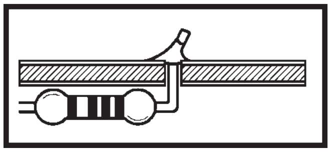
2. Apply a small amount of solder to the iron tip. This allows the heat to leave the iron and onto the foil. Immediately apply solder to the opposite side of the connection, away from the iron. Allow the heated component and the circuit foil to melt the solder.



3. Allow the solder to flow around the connection. Then, remove the solder and the iron and let the connection cool. The solder should have flowed smoothly and not lump around the wire lead.

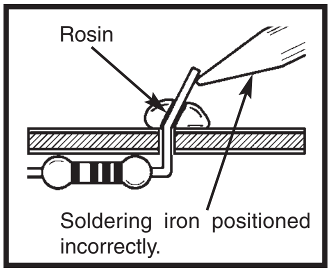


4. Here is what a good solder connection looks like.

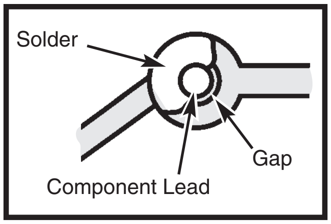


Types of Poor Soldering Connections

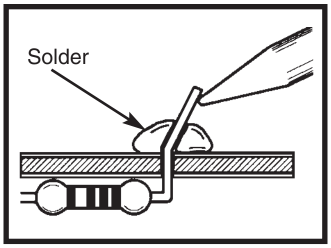
1. **Insufficient heat** - the solder will not flow onto the lead as shown.



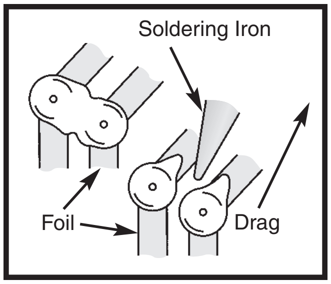
2. **Insufficient solder** - let the solder flow over the connection until it is covered. Use just enough solder to cover the connection.



3. **Excessive solder** - could make connections that you did not intend to between adjacent foil areas or terminals.



4. **Solder bridges** - occur when solder runs between circuit paths and creates a short circuit. This is usually caused by using too much solder. To correct this, simply drag your soldering iron across the solder bridge as shown.



TROUBLESHOOTING

It is rare that the components are defective. If a problem occurs in the test procedure of any lesson the following procedure should be followed:

1. Check solder joints to be sure that solder has flowed smoothly.
2. Check solder joints to be sure that solder has not flowed across to adjacent pads.
3. Check that all components are the correct value and in the proper place as specified in the Assembly Instructions.
4. Check that all LEDs and transistors are mounted with their flat side as indicated on the printed circuit board.
5. Check that all polarized capacitors are mounted with the plus side as indicated on the printed circuit board.
6. Check that all integrated circuits are mounted with the notch as indicated on the printed circuit board.
7. Most of the problems encountered in lessons 10 thru 14 are due to an improperly loaded program. Each program requires that all of the preceeding programs are operating properly. Carefully recheck each of the ROM locations of the current program as well as each location of all of the preceeding programs.

CAUTION

The MM-8000 should be built and tested lesson by lesson. The procedure at the end of each lesson is designed to be run on the system as it is configured at that time. In the final configuration all parts are installed except J10 and J11. NOT ALL THE PROCEDURES WILL RUN IN THIS CONFIGURATION. If the system is already in the final configuration the procedures may be run if the following steps are followed:

1. Start each procedure with the RESET switch up and all other switches down. Do not move the RESET switch down unless specificallly directed to do so.

2. For lessons 4 and 5, install J10 and remove the 8085. For lesson 5 only, remove J12 and install J11. The system should be returned to the final configuration at the completion of lesson 5.

LESSON 1
NUMBERS

Most of us have spent many years learning to count, add, subtract, multiply and divide in a numbering system based on the power of 10. This system uses 10 different symbols in various arrangements to represent different quantities. It is commonly believed that this system was developed because man could place his fingers in 10 different positions or states. It was found that in electronic circuits there are two convenient stable conditions or states, ON and OFF. A switch can be either open or closed. A transistor can be either on or off. Thus the natural numbering system for electronic circuits is the base 2 or binary system. This system has only two symbols, 0 and 1. Next let us look at how different quantities are represented in BASE 2, BASE 10 and BASE 16 numbering systems.

Counting in base 2 follows the same rules as in base 10. First use all the digits available, then start a new column to represent a higher quantity. If the numbering system had more than 10 symbols,the next column would not start until all the symbols were used. In Base 16 (Hexadecimal) we use the letters A, B, C, D, E, and F to represent the numbers 10 to 15. Table 1-1 compares counting to 16 in Base 10, Base 2 and Base 16.

BASE 10	BASE 2	BASE 16
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Table 1-1

In BASE 10, the number 16 means;
10^1 x 1 = 10
10^0 x 6 = $\underline{6}$ (Any number raised to power 0 = 1)
16

In the binary system the BASE 2 number 10000 means;
2^4 x 1 or 16 x 1 = 16 in BASE 10
2^3 x 0 or 8 x 0 = 0 in BASE 10
2^2 x 0 or 4 x 0 = 0 in BASE 10
2^1 x 0 or 2 x 0 = 0 in BASE 10
$\underline{2^0 \text{ x } 0}$ or 1 x 0 = $\underline{0}$ in BASE 10
10000 16 in BASE 10

The right hand side of the above table demonstrates how a binary number is converted to a BASE 10 number. To convert a BASE 10 number to the BASE 2 system divide the number by 2, recording the remainder each time, until the quotient becomes zero. The following example converts the BASE 10 number 16 to its binary equivalent.

DIVISION	Quotient	Remainder
16 2 =	8	0
8 2 =	4	0
4 2 =	2	0
2 2 =	1	0
1 2 =	0	1

The binary number is then equal to the reverse of the remainder column or 10000. You have just learned how to convert decimal numbers into binary numbers and vice versa. As you can see in Table 1-1 it only takes 1 digit in BASE 16 to represent 4 digits in BASE 2. By breaking the binary numbers into groups of 4 digits each, you can use Table 1-1 to convert numbers from binary to hexadecimal and back to binary. For example: 1111 0100 = F4 For this reason hexadecimal numbers are used as a short hand representation of binary numbers. The electronic circuits however, can only work with binary numbers. This is all the mathematics necessary to understand computer systems.

As you can see from the above example,a two digit number in BASE 10 (16) is a five digit number in BASE 2 (10000). The computer must keep track of, or remember 5 digits in its memory in order to work with this simple 2 digit BASE 10 number. This brings us to the first important part of every computer system, its memory.

LESSON 2
MEMORY

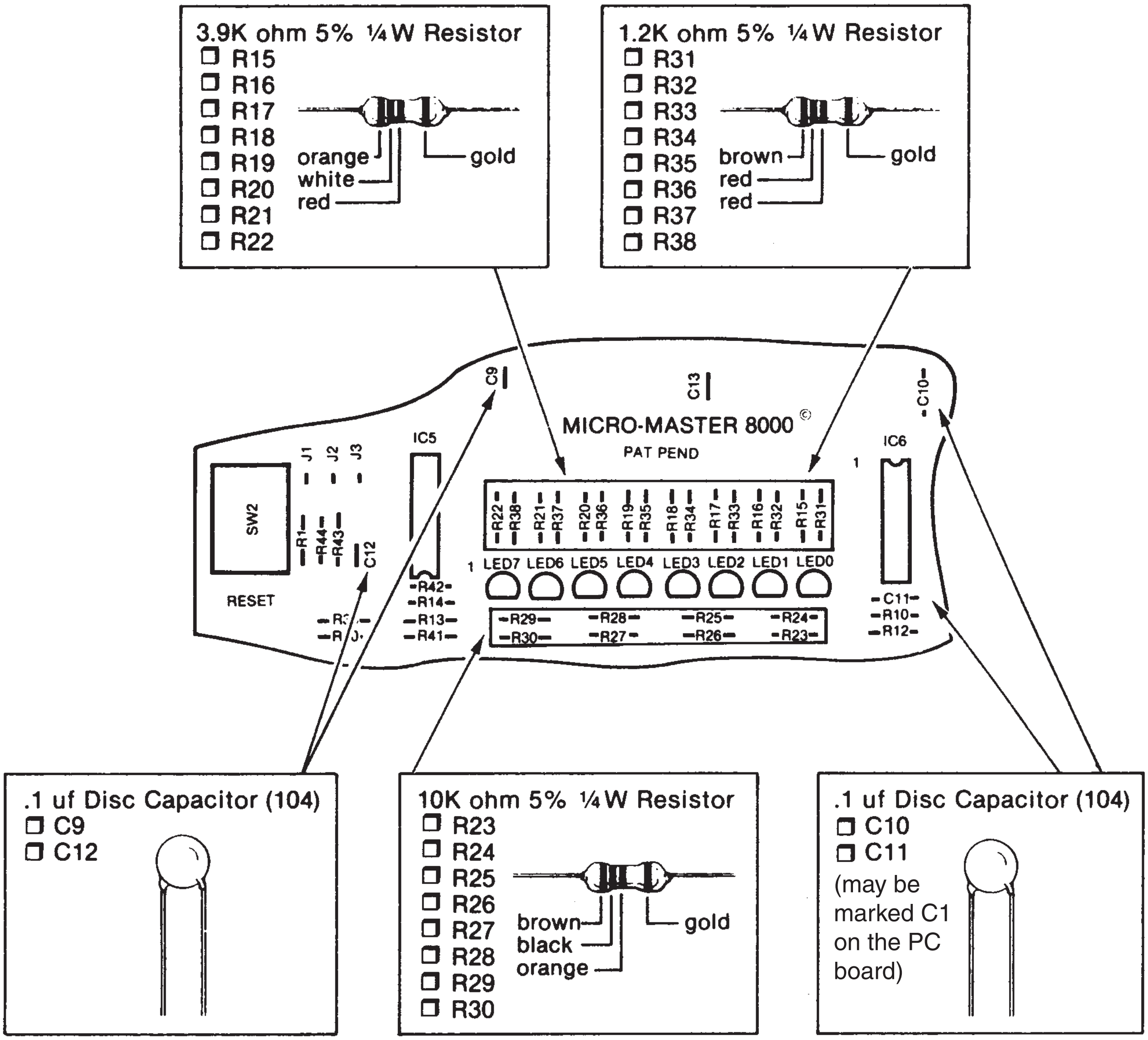
Websters dictionary defines the word memory as "the process of accepting, storing and recalling information". For a computer memory, this threefold process is best described as follows:

1. Data must have an input path to a memory location.
2. Once the data is accepted it will stay in that location.
3. If a path is opened back to a memory location, the data can be recalled.

In lesson 1 you learned that computer numbers were made up of a string of 1's and 0's. Even a medium size decimal number converted to binary will produce quite a long string of 1's and 0's. If we call each 1 or 0 a "bit", then we can break down a long string into little groups of 4 bits each. This group of 4 bits is called a "nibble". Each nibble can be represented by a single hexadecimal digit. Putting two nibbles together we get a group of 8 "bits" which is called a "byte". A microprocessor designed to transmit data one byte at a time is called an "8 bit microprocessor". In this way it takes only 8 wires to transmit the data to and from memory. Because these 8 wires carry data to and from memory like a "bus", they are called the "data bus". By connecting 8 switches to the data bus it is possible to change the data going to each memory location. Since the 8085 microprocessor also uses the data bus to transmit part of the memory address, the data bus is also referred to as the Address-Data Bus and abbreviated as AD0 thru AD7. When a switch places a high voltage on a "data bus", a 1 is placed on that data line. When a switch places a low voltage on a "data bus", a 0 is placed on that data line. Once the eight switches are set, the entire "byte" will be present on the data bus for processing. This is the method you will use to open a memory location and store data in that location. Each data line is also connected to a LED, (light emitting diode), through a driver circuit. When a "1" is present on the data line, the LED will be on. These LEDs are called the logic level indicators because they indicate the logic level of each data line in the data bus.

ASSEMBLY INSTRUCTIONS

Identify and install the following parts as shown in figure 2-1. Have you seen the solder tips page? Go back and read for helpful information. After soldering each part place a check in the box provided.



3.9K ohm 5% ¼W Resistor

- □ R15
- □ R16
- □ R17
- □ R18
- □ R19
- □ R20
- □ R21
- □ R22

1.2K ohm 5% ¼W Resistor

- □ R31
- □ R32
- □ R33
- □ R34
- □ R35
- □ R36
- □ R37
- □ R38

.1 uf Disc Capacitor (104)

- □ C9
- □ C12

10K ohm 5% ¼W Resistor

- □ R23
- □ R24
- □ R25
- □ R26
- □ R27
- □ R28
- □ R29
- □ R30

.1 uf Disc Capacitor (104)

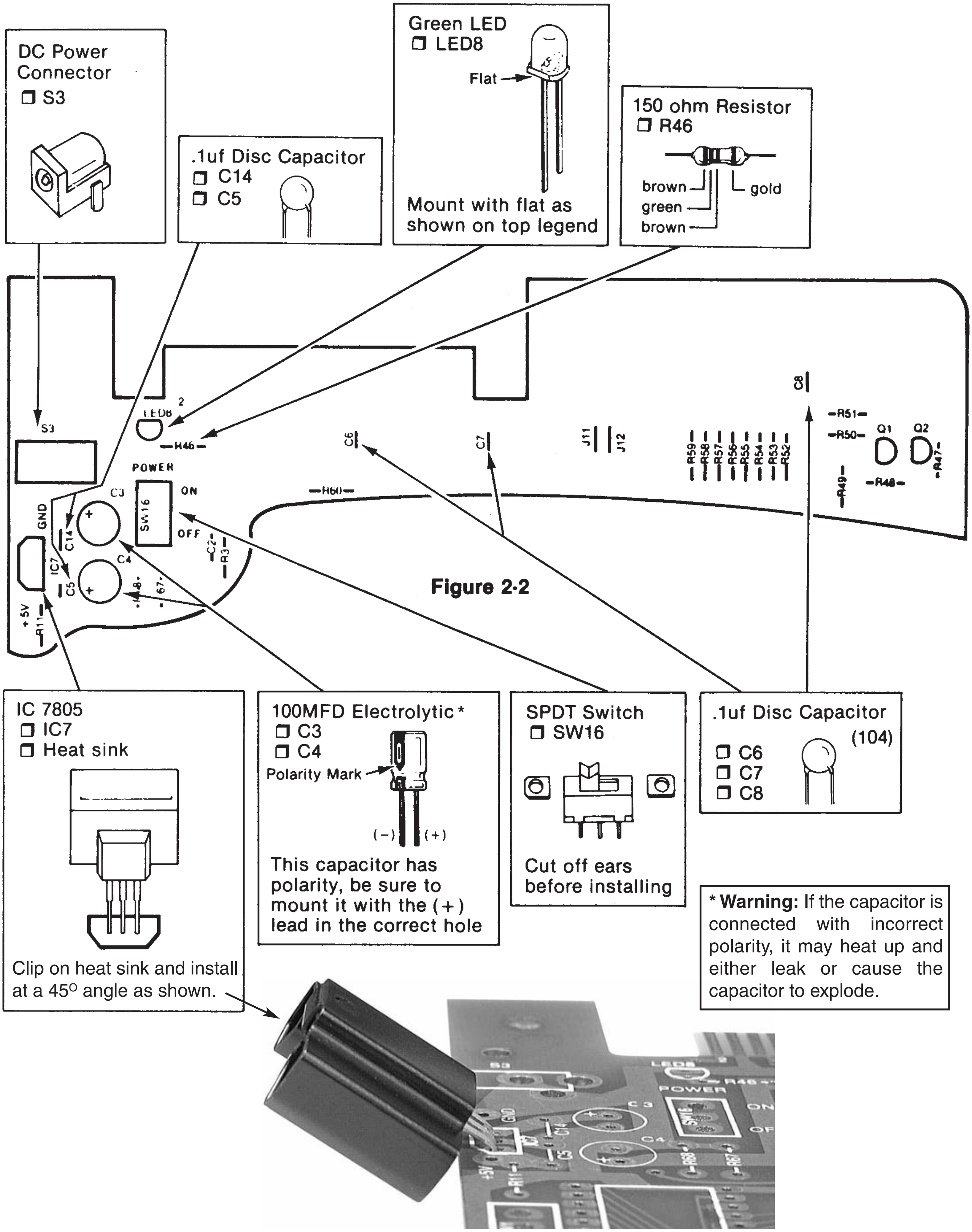
- □ C10
- □ C11

(may be marked C1 on the PC board)

Figure 2-1

ASSEMBLY INSTRUCTIONS

Identify and install the following parts as shown in figure 2-2.



DC Power Connector
□ S3

.1uf Disc Capacitor
□ C14
□ C5

Green LED
□ LED8
Flat
Mount with flat as shown on top legend

150 ohm Resistor
□ R46
brown
green
brown
gold

IC 7805
□ IC7
□ Heat sink
Clip on heat sink and install at a 45° angle as shown.

100MFD Electrolytic*
□ C3
□ C4
Polarity Mark
(−) (+)
This capacitor has polarity, be sure to mount it with the (+) lead in the correct hole

SPDT Switch
□ SW16
Cut off ears before installing

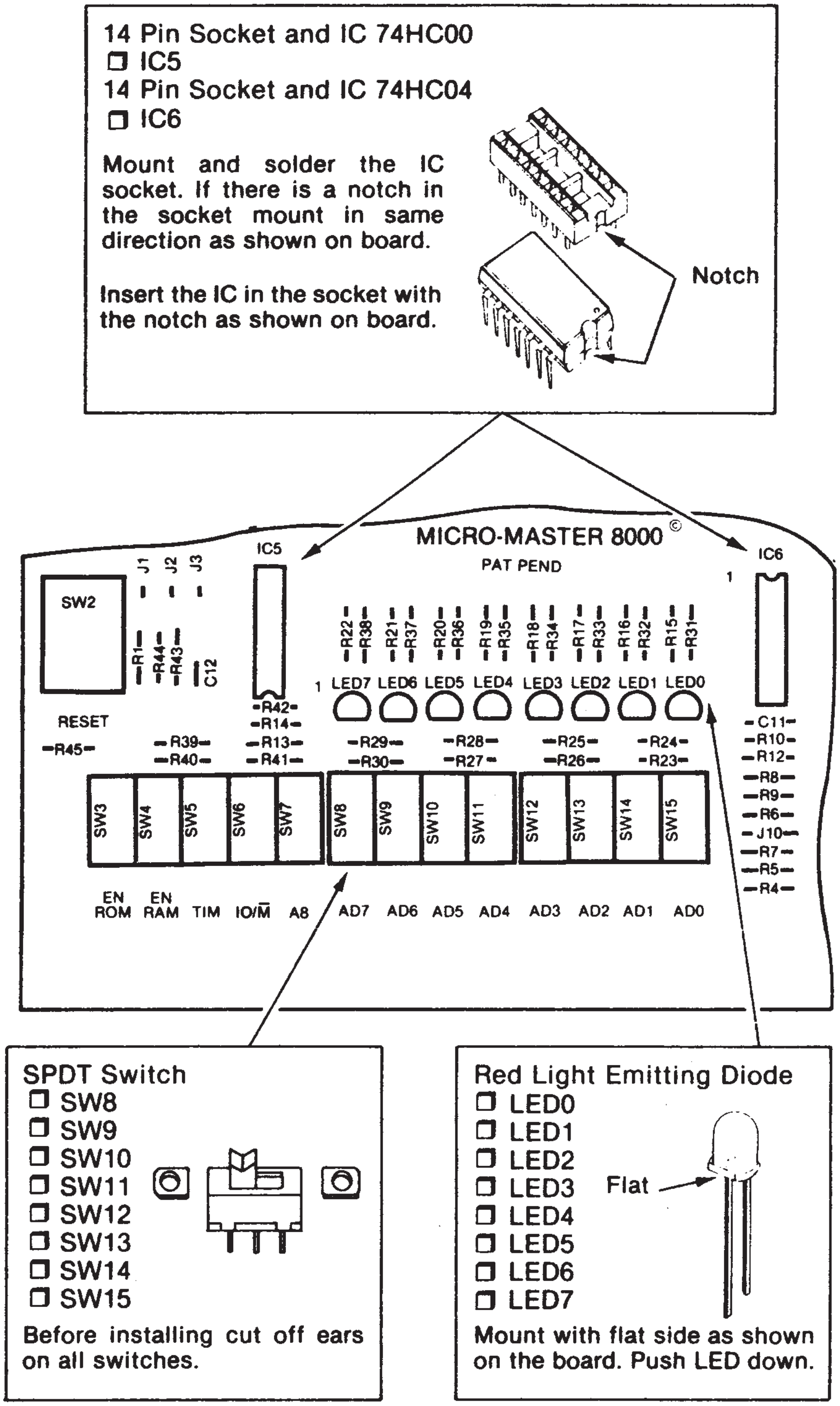
.1uf Disc Capacitor (104)
□ C6
□ C7
□ C8

Figure 2-2

*** Warning:** If the capacitor is connected with incorrect polarity, it may heat up and either leak or cause the capacitor to explode.

ASSEMBLY INSTRUCTIONS

Identify and install the following parts as shown in figure 2-3.



14 Pin Socket and IC 74HC00
□ IC5
14 Pin Socket and IC 74HC04
□ IC6

Mount and solder the IC socket. If there is a notch in the socket mount in same direction as shown on board.

Insert the IC in the socket with the notch as shown on board.

SPDT Switch
□ SW8
□ SW9
□ SW10
□ SW11
□ SW12
□ SW13
□ SW14
□ SW15

Before installing cut off ears on all switches.

Red Light Emitting Diode
□ LED0
□ LED1
□ LED2
□ LED3
□ LED4
□ LED5
□ LED6
□ LED7

Mount with flat side as shown on the board. Push LED down.

Figure 2-3

If unit will not function properly see trouble shooting guide on page VII.

CIRCUIT DESCRIPTION

Figure 2-5 is a simplified schematic of the MM-8000 DC power distribution system. Unregulated DC is brought in through connector S3 to the Power ON-OFF switch SW16. Switch SW16 applies the unregulated power to filter capacitors C3 and C14 and to voltage regulator IC7. The +5 volt output of the regulator is filtered by capacitors C4 thru C12. Capacitor C4 a 100uF capacitor, filters any low frequency noise, such as 60 hertz which may be present on the regulator output. Capacitors C5 thru C12 filter any high frequency noise which may be generated by the rapid switching from 1 to 0 and from 0 to 1 of the data bus and other lines throughout the MM-8000 system.

When power is applied current flows through resistor R46 and light emitting diode LED8 (green). The LED turns ON indicating that voltage is present on the 5 volt bus. Resistor R46 limits the current in LED8 to approximately 18ma.

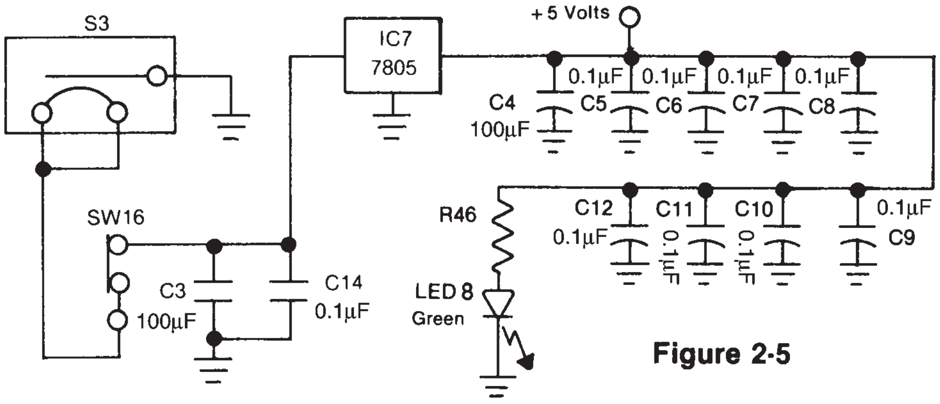


Figure 2-5

Light Emitting Diodes LED0 through LED7 are used to indicate the logic levels on the "Address Data" bus. LED0 through LED5 are driven by integrated circuit 6 (IC6). IC6 is a 74HC04 Hex inverter which consist of 6 independent inverter circuits each driving one LED logic level indicator. A typical circuit is shown in figure 2-6. When a 1 is present on the data bus the input to the inverter circuit is approximately 3.5 volts. Since this satisfies the requirements for a 1 on the input, the output of the inverter goes to the 0 state, which is approximately .2 volts DC. This causes 2.5 ma to flow through the corresponding LED and 1.2K resistor. The LED thus turns on indicating a logic 1 level on the corresponding data line. When a 0 is present on the data bus the input to the inverter circuit is approximately .2 volts DC. This causes the output of the inverter to go to +5 volts and no current can flow through the LED connected to this output. The LED remains dark indicating a 0 is present on that data line. The hex inverter thus provides a method of indicating the logic state on the first 6 address-data lines. Since a nand gate will be used later to make a debounce circuit,we will use ½ of the four nand gates in a 74HC00 to drive the remaining 2 logic level indicators.

Figure 2-6

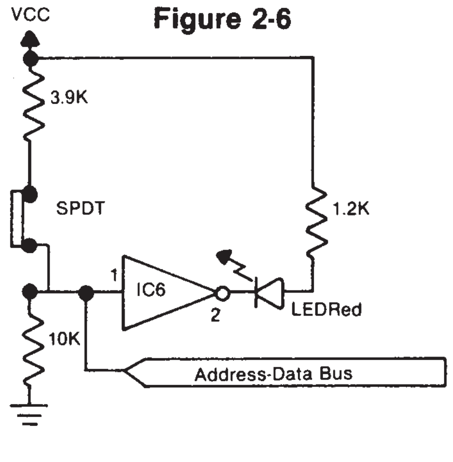
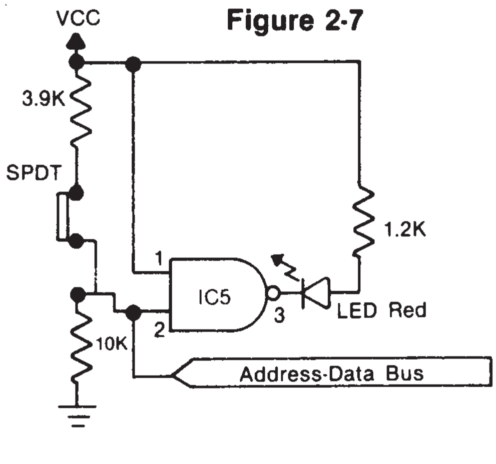


Figure 2-7



LED6 and LED7 are driven by ½ of intergrated circuit 5. A typical circuit is shown in figure 2-7. When one of the inputs of the nand gate is tied to +5 volts the circuit acts the same as the inverter circuit. By using IC5 as a logic level indicator and the debounce circuit not only is board space saved, but cost is also minimized.

A logic 1 or 0 is placed on the address-data bus by the SPDT switches as shown in figures 2-6 and 2-7. When the switch is in the down position (toward the edge of the PC board), the data bus line connected to that switch is brought to ground through the 10K resistor. This places a 0 on that bus line. When the switch is in the up position (away from the edge of the board) the combination of the 3.9K and the 10K resistors places a high voltage (a digital 1) on that bus line.

In this lesson only the switches are used to condition the address-data bus lines. In later experiments the microprocessor and memory circuits will be driving these lines. These circuits drive the address-data lines with a lower impedance than the 3.9K and 10K used here. They will therefore override the switches. In other words, when the microprocessor or memory circuits put a 1 or 0 on the address-data bus the lines will go to a 1 or 0 regardless of the data switch position.

FUNCTIONAL TEST

1. Put all switches in the down position.
2. Plug in the power supply and connect the output plug to connector S3.
3. Put the power switch (SW16) in the up position to turn power on. The green LED should light indicating the presence of voltage on the 5 volt bus. All red LEDs should be off.
4. When each data switch is put in the "1" position, the LED corresponding to that bit will turn on indicating a "1" is present on that data line.
5. Check that all eight switches and logic level indicators are working properly.
6. Turn power off.

LESSON 3
STORING AND READING DATA

There are many different ways of storing data in electronic circuits, but the two methods found in most computers are Random Access Memory (RAM) and Read Only Memory (ROM). The biggest difference between these two types of memory is the fact that RAM will lose the data stored in it if the power is turned off and ROM will not. The 8155 integrated circuit contains 256 memory locations. Each location is capable of storing 1 byte or 8 bits of data. Since this circuit can store 256 x 8 or 2048 bits of data, it is called a 2K memory (the K means times 1000).

Just like each house in a city has an address that allows us to find that house, each byte of data in the 8155 also has an address in order that we may recall that data later.

Data may be stored in any address by the following procedure:
1. Go to the proper address.
2. Put data byte on data bus.
3. Tell the memory circuit to accept the data.
4. Move on to the next address.

The 8155 integrated circuit uses the same 8 inputs for both data and addresses. How does the 8155 know what is data and what is an address? You tell it an address is on the data bus by pressing the "ALE" (Address Latch Enable) button. This lifts the ALE pin high, and the address is stored in a special address register inside the 8155. When the "WR" (Write) button is pushed the data on the data bus will be stored at the address location previously "latched" into the address register. Pressing the RD (Read) button will make the 8155 look up the data at the memory location pointed to by the special address register and send it out to the data bus. The following experiment will demonstrate these principals.

ASSEMBLY INSTRUCTIONS

Unplug connector S3 to remove all power from the PC board. Identify and install the following parts as shown in figure 3-1. After soldering each part place a check in the box provided.

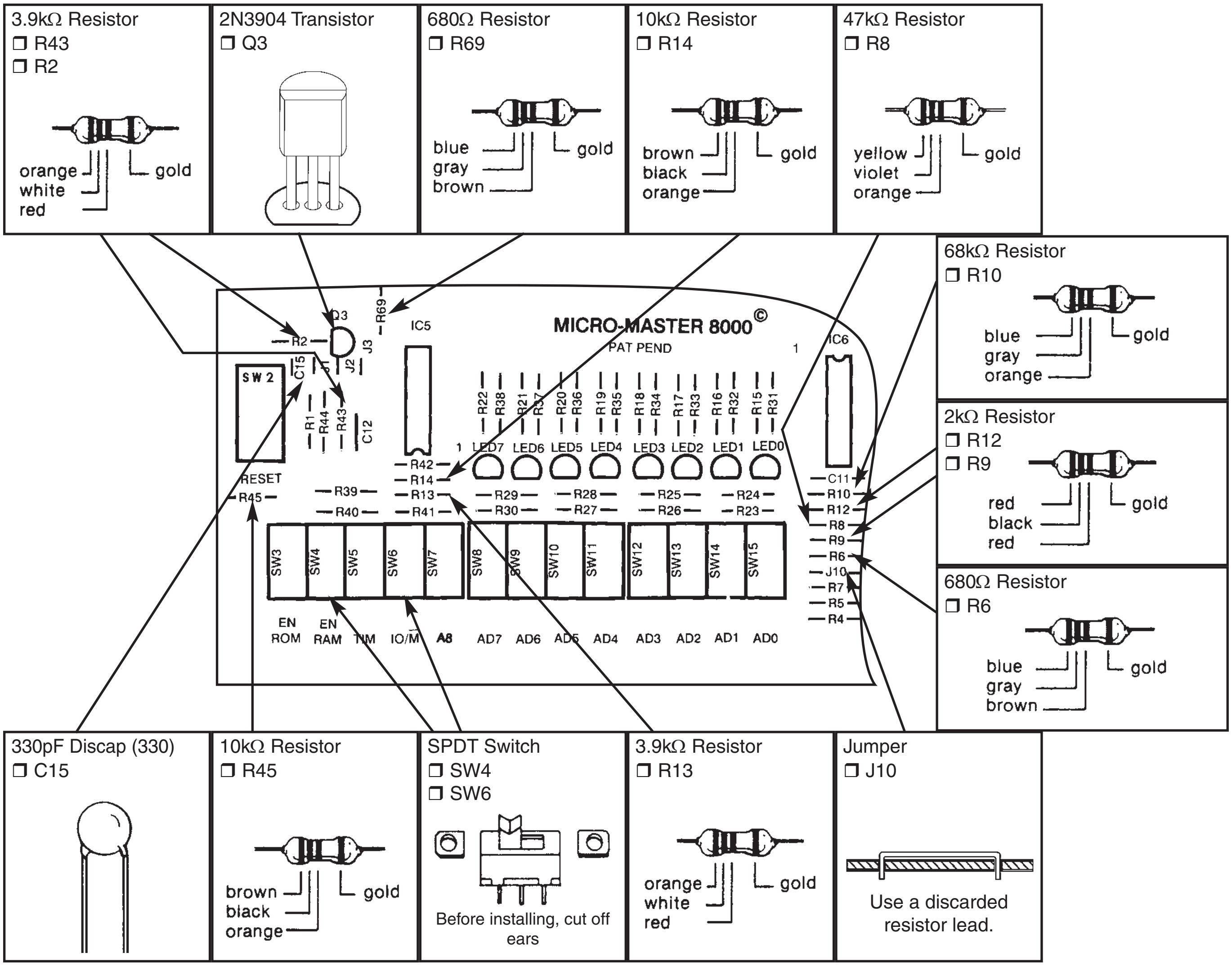
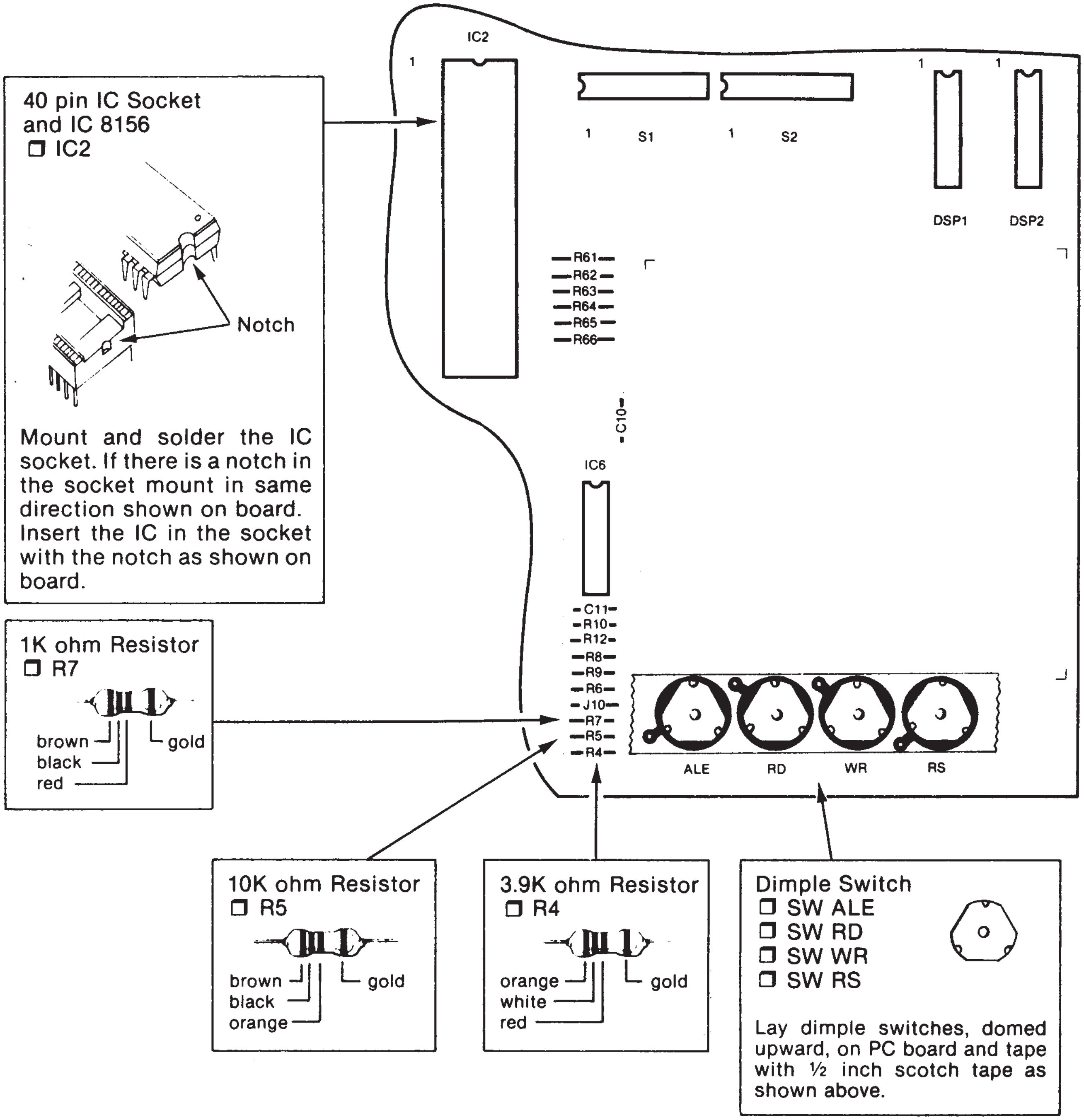


Figure 3-1

ASSEMBLY INSTRUCTIONS

Unplug connector S3 to remove all power from the PC board. Identify and install the following parts as shown in figure 3-2. After soldering each part place a check in the box provided.



40 pin IC Socket and IC 8156
❒ IC2

Mount and solder the IC socket. If there is a notch in the socket mount in same direction shown on board. Insert the IC in the socket with the notch as shown on board.

1K ohm Resistor
❒ R7

10K ohm Resistor
❒ R5

3.9K ohm Resistor
❒ R4

Dimple Switch
❒ SW ALE
❒ SW RD
❒ SW WR
❒ SW RS

Lay dimple switches, domed upward, on PC board and tape with ½ inch scotch tape as shown above.

If unit will not function properly see trouble shooting guide on page VII.

Figure 3-2

CIRCUIT DESCRIPTION

In this lesson the 8155 integrated circuit was installed on the address-data bus provided in lesson 2. The switch networks necessary for manual operation of the 8155 memory circuits were also installed. As shown in figure 3-3, these switch networks are:

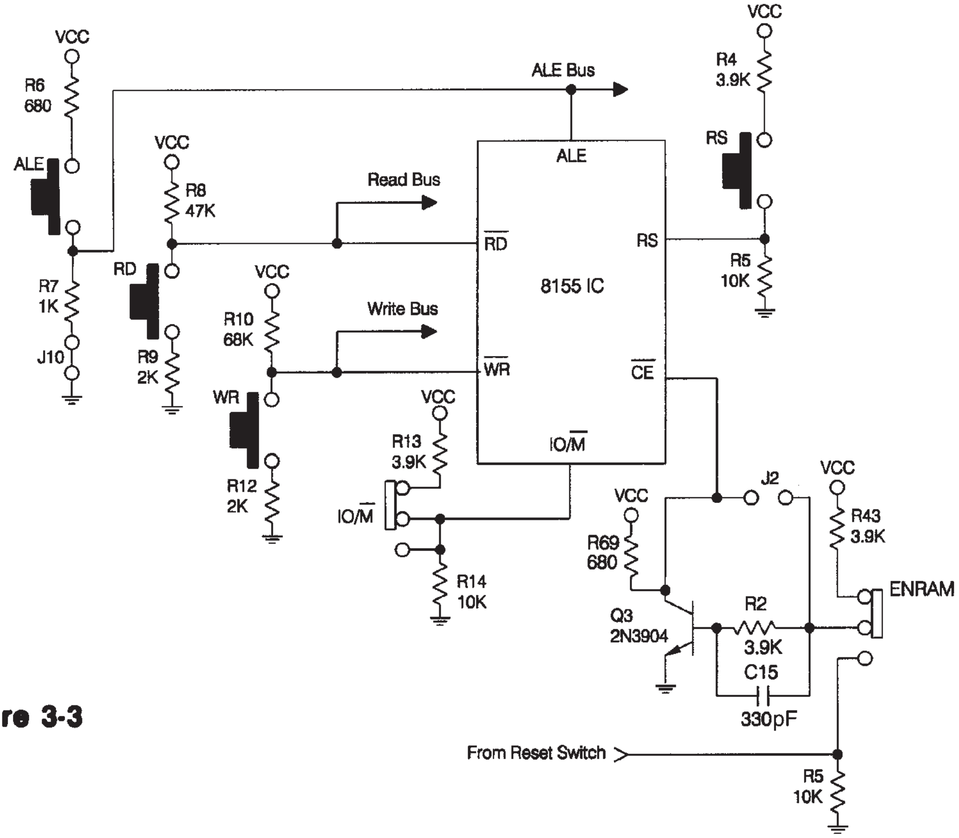


Figure 3-3

ADDRESS LATCH ENABLE (ALE)

The ALE line is normally held near ground potential (0 volts) by resistor R7 and jumper J10. When the ALE button is pressed the line is brought to a high voltage by the voltage divider network R6 and R7. In a later lesson,J10 will be removed to allow the 8085 microprocessor to drive the ALE line through resistor R7.

READ FROM MEMORY ($\overline{\text{RD}}$)

The $\overline{\text{RD}}$ (pronounced RD not) pin on the 8155 is held at a high voltage by resistor R8. The line over the RD indicates this pin is activated by a low voltage or a 0. When the RD button is pressed, the $\overline{\text{RD}}$ pin on the 8155 integrated circuit is brought to a low voltage by divider network R8 and R9. The data in the memory location at the previously latched address is now reproduced on the 8 data lines. The actual data in that memory location is not changed however. The data in each memory location will remain the same until it is overwritten or the power to the 8155 integrated circuit is removed. The IO/$\overline{\text{M}}$ switch must be in the $\overline{\text{M}}$ position in order to read memory.

WRITE TO MEMORY (WR)

WR operates much like RD except that when this key is pressed whatever is on the data lines will be written into the previously latched memory address. The data in that address at the time the WR key is pressed wil be written over and forever lost. The new data will remain in that address until it is written over or power is removed from the 8155 integrated circuit. The IO/$\overline{\text{M}}$ switch must be in the $\overline{\text{M}}$ position in order to write to memory.

INPUT OUTPUT OR MEMORY (IO/$\overline{M}$)

IO/$\overline{M}$ switch tells the 8155 integrated circuit whether the input-output ports (see lesson 4) or the memory gets the data. This switch operates electrically the same as the address-data switches.

ENABLE RAM (ENRAM)

This switch connected to the 8155 integrated circuit chip enable ($\overline{CE}$) pin through transistor Q3. When the switch is down (toward the edge of the board) the transistor is turned off. The $\overline{CE}$ pin is held high by resistor R69. In this state the 8155 is disabled and will ignore input data on all of its other pins. When the switch is up, the transistor turns on and the $\overline{CE}$ pin is held near ground. In this state, the 8155 is enabled and will respond to inputs on the other pins. In a later lesson, the reset switch will be used to allow address line A15 to control the $\overline{CE}$ input.

RESET (RS)

The reset key operates electrically the same as the ALE key. This key will be used in a later lesson.

PROCEDURE

1. Connect the power supply and set the power switch (SW16) to ON. The green LED should light indicating power is ON.
2. Put the ENRAM in the on position to enable the 8155. Put the IO/$\overline{M}$ switch in the off position to access the memory portion of the 8155.
3. Start at address 0000 0000 by putting all 8 data switches in the "0" (down) position.
4. Push the ALE switch to "latch" the address in the special address register.
5. Change the data switches to 1000 0001.
6. Press the WR button to write this data into the address location that has just been opened.
7. You can change the address by repeating steps 3 and 4. Data can be stored by repeating steps 5 and 6. Use this procedure to store the following data bytes in the first four memory locations.

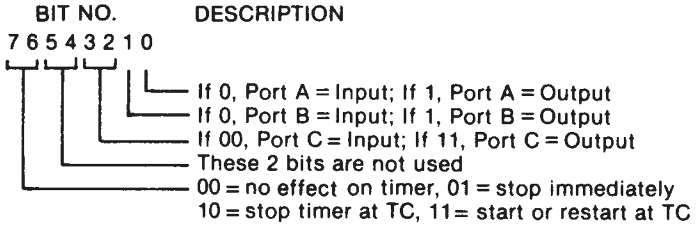
DECIMAL	ADDRESS	DATA
0	0000 0000	1000 0001
1	0000 0001	0100 0010
2	0000 0010	0010 0100
3	0000 0011	0001 1000

8. Check the data stored in each address by repeating steps 3 and 4 to return to each address.
9. Read the data at each address by pressing the RD button.
10. Put the power switch (SW16) down to turn power off.
11. Turn the power back on.
12. Repeat steps 8 and 9. Was the data lost?

5 volts must be present at all times or the data stored in the 8156 memory will be lost.

LESSON 4
REGISTERS AND PORTS

In lesson 3 the address of a memory location was "latched" into the 8155 before data was placed on the data bus. The read or write buttons were then pushed to perform the desired operation. The IO/M̄ (Input Output/Memory) switch was set in the M̄ position during this process, putting a low voltage on the IO/M̄ pin of the 8155. The line over the M is called a "NOT" bar and signifies that the Memory is activated when this pin is NOT high. Putting this switch in the IO position tells the 8155 to pass the data on the data bus to the Input/Output ports, timer control, or the Command Status Register. Before data can be transmitted or received through the ports the 8155 must be told which ports will be inputs and which ports will be outputs. Timer instructions must also be given to the 8155 if the timer is going to be used. (The timer will be discussed in another lesson). Instructions are given to the 8155 by setting "bits" to 1 or 0 in the COMMAND STATUS REGISTER. This register is located at address 0000 0000. (With the IO/M̄ switch in the IO position). Figure 4-1 shows how the Command Register operates.

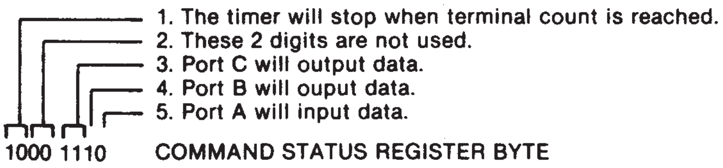


If 0, Port A = Input; If 1, Port A = Output
If 0, Port B = Input; If 1, Port B = Output
If 00, Port C = Input; If 11, Port C = Output
These 2 bits are not used
00 = no effect on timer, 01 = stop immediately
10 = stop timer at TC, 11= start or restart at TC

Figure 4-1

EXAMPLE

If the binary number 1000 1110 is written to address location 0000 0000 when the IO/M̄ switch is in the IO position, then the 8155 will function as follows:


1. The timer will stop when terminal count is reached.
2. These 2 digits are not used.
3. Port C will output data.
4. Port B will ouput data.
5. Port A will input data.

After the ports are set, data is transmitted through them by first going to the address of the desired port. Next,data is placed on the data bus if the port is configured as an output port. Data is put on the port pins if the port is configured as an input port. The addresses of each port can be found in figure 4-2.

ADDRESS	NAME
7654 3210	
XXXX X000	Command Register
XXXX X001	Port A
XXXX X010	Port B
XXXX X011	Port C

X-Indicates don't care,can be 1 or 0

Figure 4-2

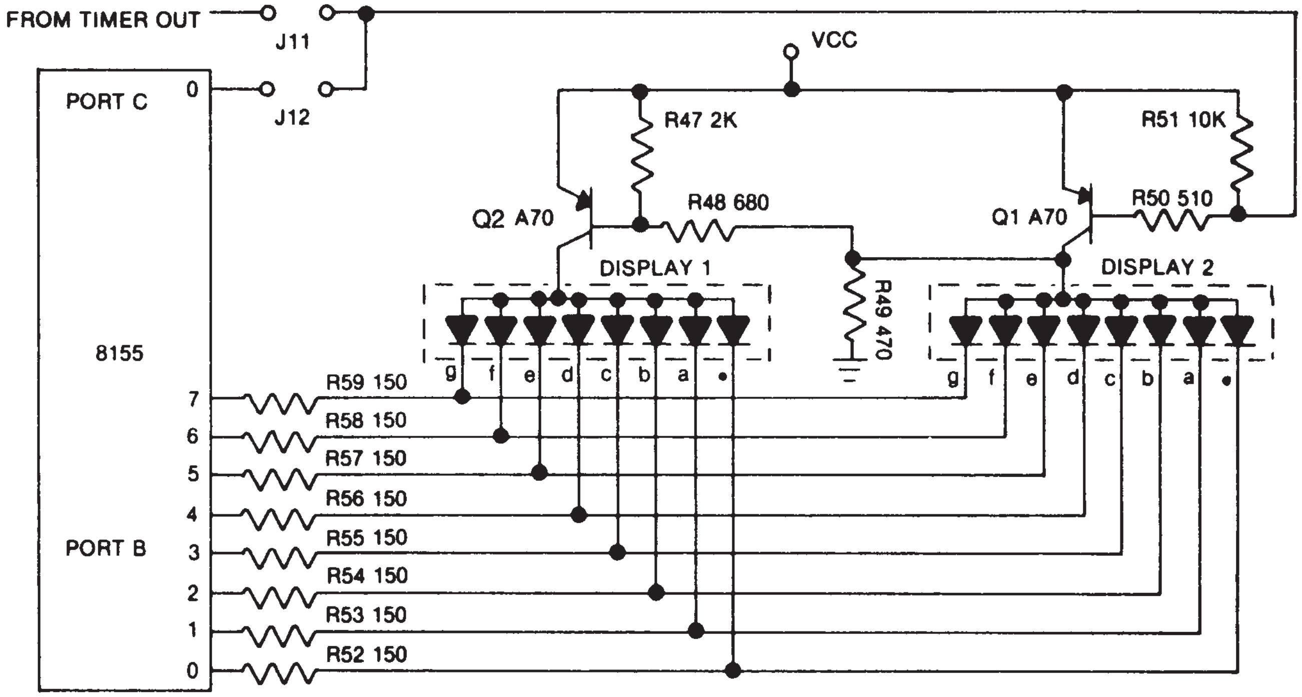
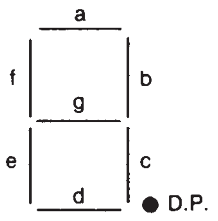


Figure 4-3

Both display 1 and display 2 are connected to port B as illustrated in figure 4-3. Only one display or digit is driven at a time. The display to be driven is controlled by bit 0 of port C. A 1 in this bit selects display 1, a 0 selects display 2. To light any segment of the display a zero,must be placed in the corresponding bit of port B (see figure 4-4).

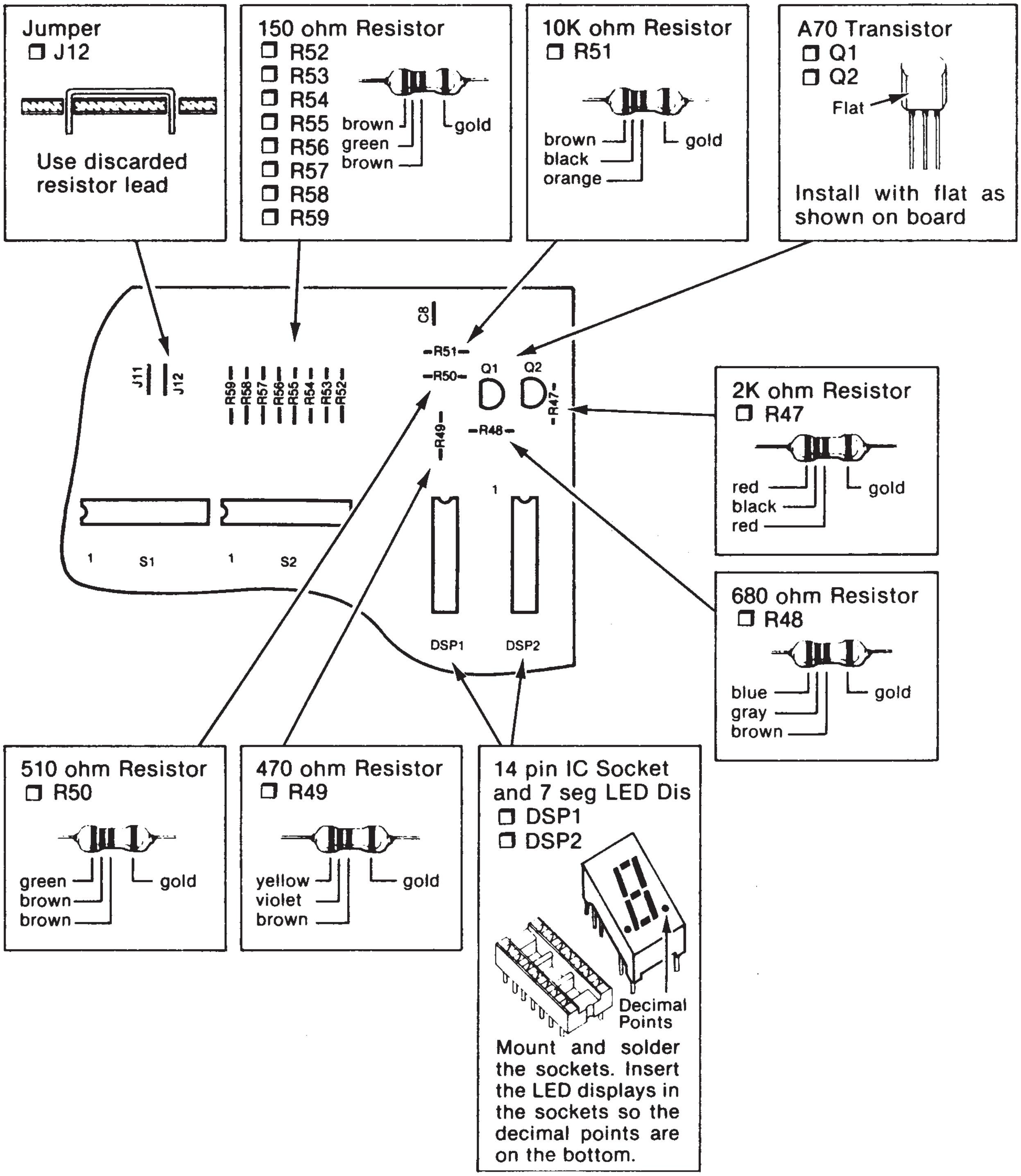


SEGMENT	PORT B BIT
DP	0
a	1
b	2
c	3
d	4
e	5
f	6
g	7

Figure 4-4

ASSEMBLY INSTRUCTIONS

Unplug connector S3 to remove all power from the PC board. Identify and install the following parts as shown in figure 4-5. After soldering each part place a check in the box provided



Jumper
□ J12

Use discarded resistor lead

150 ohm Resistor
□ R52
□ R53
□ R54
□ R55
□ R56
□ R57
□ R58
□ R59

10K ohm Resistor
□ R51

A70 Transistor
□ Q1
□ Q2

Install with flat as shown on board

2K ohm Resistor
□ R47

680 ohm Resistor
□ R48

510 ohm Resistor
□ R50

470 ohm Resistor
□ R49

14 pin IC Socket and 7 seg LED Dis
□ DSP1
□ DSP2

Mount and solder the sockets. Insert the LED displays in the sockets so the decimal points are on the bottom.

Figure 4-5

If unit will not function properly see trouble shooting guide on page VII.

CIRCUIT DESCRIPTION

Figure 4-3 shows the seven segment displays and the associated drive circuits added in lesson 4. To turn on a particular segment of a display, we must first enable that display by turning on the appropriate transistor Q1 or Q2, which ties the common anode point of the display to approximately 5 volts. Next we must switch the bit at Port B connected to that segment to a low voltage. This allows current to flow from the common anode point through the appropriate LED and 150 ohm resistor into Port B and to ground.

To enable Display 1, bit 0 of Port C is set to 1. The logic high output from bit 0 is coupled through J12 and places approximately 5 volts DC on the base of Q1, turning it off. This disables Display 2 and allows current to flow from the emitter to the base of Q2 through R48 and R49 to ground. Q2 turns on and enables Display 1. To enable Display 2, bit 0 of Port C is set to 0. The low voltage output from bit 0 allows current to flow from the emitter to the base of Q1, through R50 and J12 into bit 0 of Port C to ground. This turns Q1 on enabling Display 2. The 5 volts on the collector of Q1 is coupled through R48 to the base of Q2 turning it off. This disables Display 1.

TEST PROCEDURE

1. Connect the power supply and turn power on. Put the IO/$\overline{M}$ switch in the IO position (up). This tells the 8155 that data is to be sent to the internal registers and not to memory. Press the RS key to initialize the 8155. Put the ENRAM switch in the ON position.
2. Set data switches to 0000 0000. This is the address of the command status register.
3. Press the ALE key to latch the address in the 8155.
4. Set the data swtiches to 1000 1110.
5. Press the WR key to write the data into the command register. As in the example above this sets port A to input and ports B and C to outputs. Since the port outputs are initially 0 this lights all segments including the decimal point of display 2.
6. Change the data switches to 0000 0010. This is the address of port B.
7. Press the ALE switch to latch the address of port B into the 8155 address register.
8. Place the data to be transmitted on the data bus. Start with 0110 0001.
9. Press the WR key to transfer the data to port B. As shown in figure 4-4 , the 0's in bits 1,2,3,4 and 7 will make a 3 appear on display 2.
10. Change the data switches to all 0's. Did the data at port B change? Port B will only change when new data is written into the register at 0000 0010.
11. Repeat steps 8 and 9 to verify the seven segment display symbols shown in figure 4-6.

HEX	BINARY	7 SEGMENT
FF	1111 1111	BLANK
81	1000 0001	0
F3	1111 0011	1
49	0100 1001	2
61	0110 0001	3
33	0011 0011	4
25	0010 0101	5
05	0000 0101	6
F1	1111 0001	7
01	0000 0001	8
31	0011 0001	9
11	0001 0001	A
07	0000 0111	b
8D	1000 1101	C
43	0100 0011	d
0D	0000 1101	E
1D	0001 1101	F
FE	1111 1110	.

Figure 4-6

12. Set the data switches to 0000 0011. This is the address of port C.
13. Press the ALE key to latch the address into the 8155.
14. Set the data switches to 0000 0001. When this data is sent to port C display 2 will be disabled and display 1 will be enabled.
15. Press the WR key and the pattern on display 2 will jump to display 1.
16. Repeat steps 6 and 7 to set the address of Port B in the 8155.
17. Repeat steps 8 and 9 to verify that the data sent to port B will now appear on display 1.

READING AN OUTPUT PORT

Even if ports A, B or C are configured as output ports, their contents may be read in the same manner as input ports.

18. Set the data switches to 0000 0011 to address Port C.
19. Push the ALE button to latch the address into the 8155.
20. Push the RD button to read the contents of Port C. The data read is 1100 0001. The ones in bits 6 and 7 are due to the fact that Port C is a 6 bit register containing only bits 0-5. The 00 0001 in bits 0-5 is the data stored in step 14 and 15.
21. Set the data switches to 1111 1011. This is also the address of Port C. As shown in figure 4-2 bits 3-7 may be 1 or 0. Only bits 0-2 are decoded by the 8155.
22. Push the ALE button to latch the address into the 8155.
23. Set the data switches to 0000 0000.
24. Push the WR button to write this data into Port C. The 0 in bit 0 disables display 1 and enables display 2. The pattern in display 1 jumps to display 2.
25. Turn power off.

LESSON 5
THE TIMER

The 8156 timer circuit is a 14 bit programmable counter. The counter is programmed by setting the two bytes of the Count Length Register as shown in Figure 5-1. The Count Length Register bits T0 thru T13 are counted down by TIMER IN pulses. When the terminal count is reached, the cycle is complete and a pulse or square wave is present at the TIMER OUT pin of the 8155 integrated circuit. Bits M1 and M2 of the Count Length Register specify the timer mode as shown in Figure 5-2. The Count Length Register may be set to any value from 002H to 3FFFH (H = Hexadecimal Number).

7	6	5	4	3	2	1	0
T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0

LSB of count length reg.

7	6	5	4	3	2	1	0
M_2	M_1	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8

Timer Mode — MSB of count length reg.

Figure 5-1

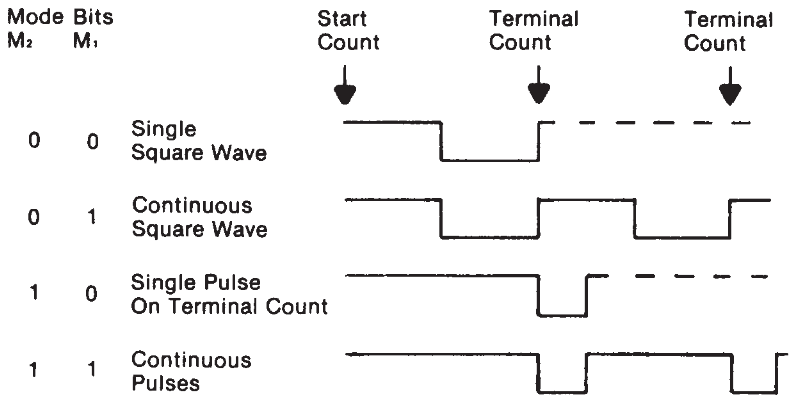


Figure 5-2

The LSB (Least Significant Byte) of the Count Length Register is at I/O address 04H. The MSB (Most Significant Byte) of the Count Length Register is at I/O address 05H. The Count Length Register is set in exactly the same manner as outputting to an I/O Port.

Bits 6 and 7 (TM2 and TM1) of the Command Status Register are used to stop and start the counter as shown in figure 5-3.

7	6	5	4	3	2	1	0
TM_2	TM_1	X	X	PC_2	PC_1	PB	PA

If 0, port A = Input
If 1, port A = Output

If 0, port B = Input
If 1, port B = Output

If 00, port C = Input
If 11, port C = Output

These 2 bits not used

00 = NOP-Do not affect counter operation.
01 = STOP- NOP if timer has not started, stop counting if timer is running.
10 = Stop after TC-stop immediately after present TC is reached (NOP if timer has not started).
11 = Start load mode and count length and start immediately after loading (if timer is not presently running). If timer is running start the new mode and count length after present TC is reached.

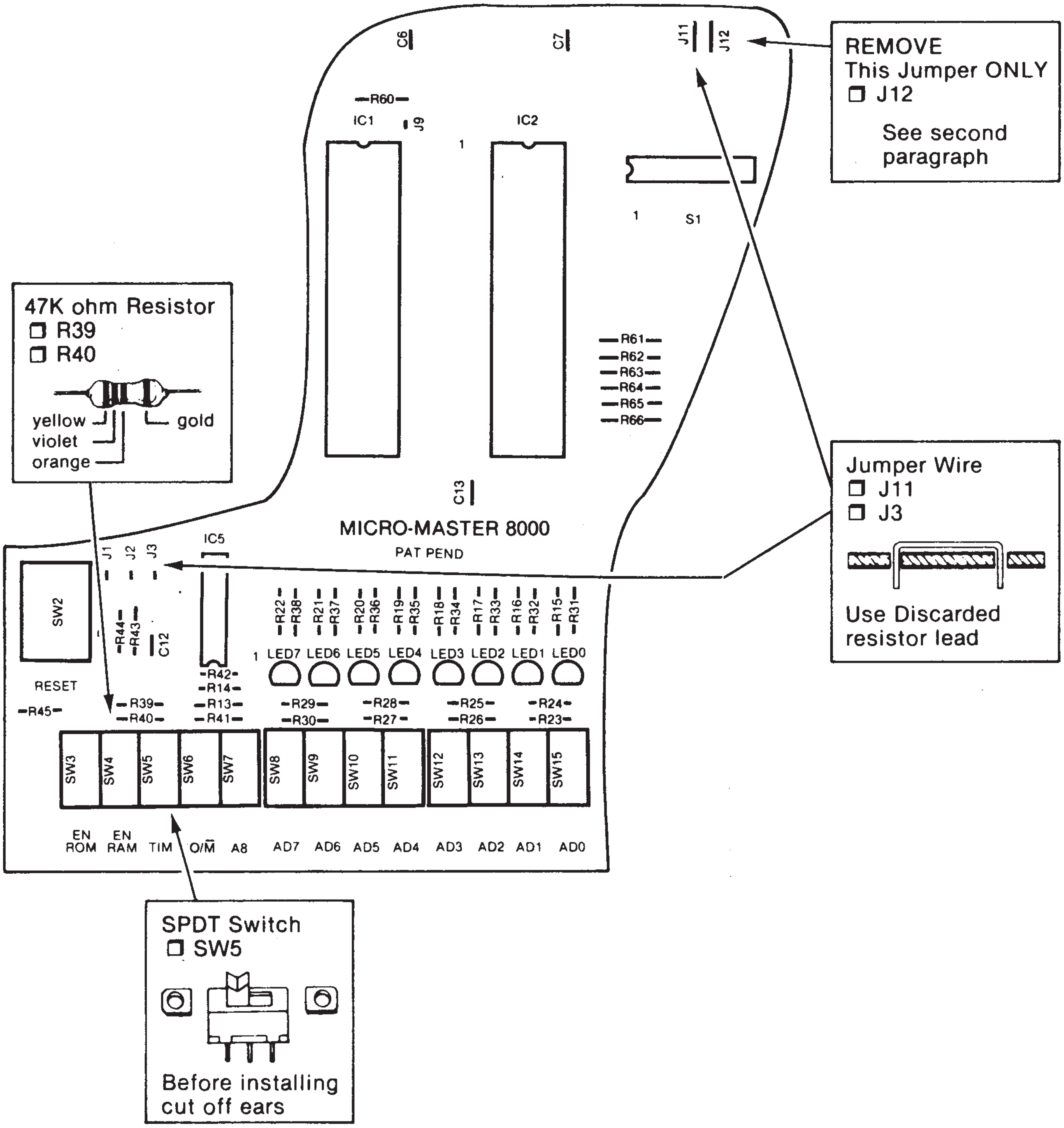
Figure 5-3

If the timer is not running when the Count Length Register is loaded, it will remain stopped until a new start command is issued. If the timer is running when a new mode or count length is loaded into the Count Length Register the timer will continue to run with the old mode or count length until a new start command is issued through the Command Status Register.

ASSEMBLY INSTRUCTIONS

Unplug connector S3 to remove all power from the PC board. Identify and install the following parts as shown in figure 5-4. After soldering each part, place a check in the box provided.

To remove jumper J12. Reset your desoldering pump and hold in one hand. With your soldering iron in the other hand heat one pad of J12, immediately put the tip of the pump in the melted solder and trigger the desoldering pump. Now remove the soldeŕ from the other pad. The holes should be clean of solder and the jumper wire free from the board.



REMOVE
This Jumper ONLY
□ J12

See second paragraph

47K ohm Resistor
□ R39
□ R40

Jumper Wire
□ J11
□ J3

Use Discarded resistor lead

SPDT Switch
□ SW5

Before installing cut off ears

Figure 5-4

If unit will not function properly see trouble shooting guide on page VII.

CIRCUIT DESCRIPTION

To clock the timer, a low to high transition must be applied to the $\overline{\text{TIMER IN}}$ pin of the 8155. In this lesson this will be done with the timer switch. Like any switch, when the TIM switch is moved from the down to the up position it is likely that the contact between the center and upper pins will be made and broken several times before the switch finally comes to rest. This phenomena is called switch bounce. If the switch output were applied directly to the TIMER IN pin, several low to high transitions would be received and counted each time the switch is thrown. Since we want only one transition each time the switch is thrown, the switch must be "debounced".

Debounce is accomplished by tying the TIM switch to a flip flop circuit as shown in Figure 5-5. Both pins 12 and 13 of IC5 must be high to make pin 11 go low. With the TIM switch in the down position, a low (ground) is applied to pin 13 causing pin 11 and pin 10 to be high. Pin 9 is biased high by resistor R39. With both pin 9 and 10 high pin 8 will be low. This applies a low to pin 12 and the TIMER IN through jumper J3. The low on pin 12 keeps pin 11 high even if the low is removed from pin 13. The flip flop is now in the zero state.

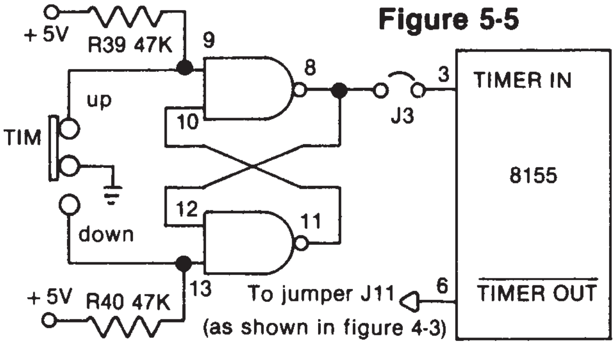


Figure 5-5

To clock the timer, the TIM switch is moved to the up position. This breaks the ground connection to pin 13 causing it to be biased high by resistor R40. Next, pin 9 is connected to ground causing pins 8 and 12 to go high. Since pin 13 is also high, pins 11 and 10 go low which forces pin 8 to stay high even if pin 9 goes high and low several times due to switch bounce. The flip flop is now in the one state and the low to high transition has been sent to the TIMER IN pin of the 8155 through jumper J3 to clock the TIMER.

In the same manner the flip flop action described above insures a clean transition on the TIMER INPUT when the switch is moved from up to down and the flip flop goes from the one state to the zero state.

To sense the state of the $\overline{\text{TIMER OUT}}$ jumper J12 is removed and jumper J11 is installed. The $\overline{\text{TIMER OUT}}$ line instead of bit 0 of Port C enables the Displays as shown in Figure 4-3. When $\overline{\text{TIMER OUT}}$ is active (low), Display 2 is on. When $\overline{\text{TIMER OUT}}$ is inactive (high), Display 1 is on.

PROCEDURE

SET UP

1. Connect the power supply and turn power on. Press the RS button to initialize the 8155.
2. Set the TIM switch down.
3. Set the ENRAM switch up. This tells the 8155 that the commands being sent on the data lines are for it and not for some other circuit.
4. Set the IO/$\overline{\text{M}}$ switch up. This directs the commands to the IO section of the 8155.

CONTINUOUS SQUARE WAVE:

1. Set the data switches to 0000 0000. This is the address of the Command Status Register.
2. Press the ALE button to load the address into the 8155 IC.
3. Set the data switches to 0100 0010. This will issue a stop command to the timer and set Port B to output.
4. Press the WR button to send data to Command Status Register.
5. Set the data switches to 0000 0100. This is the address of the LSB (Least Significant Byte) of the Count Length Register.
6. Press the ALE button to latch address into the 8155 IC.
7. Set the data switches to 0000 0110. This will set the LSB of the Count Length Register to 6.
8. Store the 6 in the LSB by pressing the WR button.
9. Set the data switches to 0000 0101. This is the address of the MSB (Most Significant Byte) of the Count Length Register.
10. Press the ALE button to latch the address into the 8155 IC.
11. Set the data switches to 0100 0000. This will set the MSB of the Count Status Register to zero and the timer mode to Continuous Square Wave.
12. Press the WR button to store the data.
13. Set the data switches to 0000 0000. This is the address of the Command Status Register.
14. Store the address by pressing ALE.
15. Set the data switches to 1100 0010. This keeps Port B an output port and starts the timer.
16. Store data by pressing WR button.
17. Clock the timer by moving the TIM switch up and down.
18. Repeat step 17 and observe that $\overline{\text{TIMER OUT}}$ switches from high (Display 1 on) to low (Display 2 on) on every third step. The first change requires 4 steps of the switch due to the internal state of the timer.
19. Set the data switches to 0000 0000. This is the address of the Command Status Register.
20. Latch in address by pressing ALE button.
21. Set the data switches to 0100 0010. This will issue a stop command to the timer.
22. Store the stop command by pressing the WR button. If the $\overline{\text{TIMER OUT}}$ line is low and the TIM switch is up, the $\overline{\text{TIMER OUT}}$ line will immediately go high (Display 1 on).
23. Repeat step 17. The $\overline{\text{TIMER OUT}}$ line, if not already high will go high (Display 1 on) the first time the TIM switch is in the up position. It will then remain high. The timer is now inactive following the stop command.

CONTINUOUS PULSES:

1. Set the data switches to 0000 0101. This is the address of the MSB of the Count Length Register.
2. Latch address by pressing the ALE button.
3. Set the data switches to 1100 0000. This will change the timer mode to continuous pulses.
4. Store data by pressing the WR button.
5. Clock the timer by moving the TIM switch up and down many times. Does the output change? No, a start command must be issued first.
6. Set the data switches to 0000 0000, the address of the Command Status Register.
7. Latch the address in by pressing the ALE button.
8. Issue a start command by setting the data switches to 1100 0010 and pressing the WR button.
9. Clock the timer by moving the TIM switch up and down many times. Observe that the $\overline{\text{TIMER OUT}}$ goes low (Display 2 on) on the 6th step and high (Display 1 on) on the 7th step. This pulse is repeated every 6th step.
10. Clock the timer until the $\overline{\text{TIMER OUT}}$ line has just gone high (Display 1 on), then procede to the next section.

STOP AFTER TERMINAL COUNT:

1. Set the data switches to 0000 0000, the Command Status Register, and press the ALE button to latch the address.
2. Set the data switches to 1000 0010. This will issue a STOP AFTER TC (TERMINAL COUNT) to the timer.
3. Press the WR button to store the command.
4. Clock the timer by moving the TIM switch up and down and observe that only one more pulse is output on the $\overline{\text{TIMER OUT}}$ line. After that, the $\overline{\text{TIMER OUT}}$ will remain high.

SINGLE SQUARE WAVE:

1. Set the data switches to 0000 0101. This is the address of the MSB of the Count Length Register. Latch the address by pressing the ALE button.
2. Set the data to 0000 0000. This will change the timer mode to Single Square Wave. Store command by pressing the WR button.
3. Set the data switches to 0000 0000, the address of the Command Status Register and latch in this address by pressing the ALE button.
4. Issue a start command to the timer by setting the data switches to 1100 0010 and pressing WR button.
5. Clock the timer by moving the TIM switch up and down. Observe that the $\overline{\text{TIMER OUT}}$ line goes low (Display 2 lit) on the 4th step and high (Display 1 lit) on the 7th step. After that the $\overline{\text{TIMER OUT}}$ line remains high.

SINGLE PULSE:

1. Repeat the above first four steps using 1000 0000 in step 2. This will put the timer mode in the Single Pulse mode.
2. Clock the timer by moving the TIM switch up and down. Observe that a pulse appears on the $\overline{\text{TIMER OUT}}$ line between the 6th and 7th step. After that the $\overline{\text{TIMER OUT}}$ line remains high (Display 1 lit).

CHANGING CYCLE LENGTH (FREQUENCY):

1. Repeat the Continuous Square Wave procedure but change the LSB of the Count Length Register in step 7 (be sure to keep the Count Length Register at 2 or greater). Note that the square wave output now corresponds to the length issued in step 7, except for the first cycle which is one step more. If an odd number is used in step 7, the high $\overline{\text{TIMER OUT}}$ length (Display 1 lit) will be one step longer than the low $\overline{\text{TIMER OUT}}$ length.
2. Turn power off.

LESSON 6 ROM

In lesson 3 we wrote to and read from the 8156 Random Access memory (RAM). When the power was turned off, the data in memory was lost. Much time would be lost if the computer programs were all stored in RAM and had to be reloaded by hand each time the power was turned on. We therefore want at least one program (called the monitor program) to reside in a non-volatile Read Only Memory (ROM). In ROM, data will be retained when power is turned off. Read Only Memories can be divided into two main types; Erasable and Nonerasabel ROMs. Erasable ROMs have some method of removing the programs stored on them (such as exposing them to ultraviolet light). Nonerasable ROMs are programmed once and connot be changed after programming. The Read Only Memory used in the MM-8000 system is the 2816 Electrically Erasable Programmable Read Only Memory (E^2 PROM). As the name implies, the 2816 is electrically reprogrammable in much the same way as the 8156 RAM. Due to the way the data is stored in the 2816, 10 ms (milliseconds) are required to complete the WRITE operation.

The 2816 contains over 2000 memory locations each capable of storing 1 byte or 8 bits (the total chip can store approximately 16,000 or 16K bits, hence the name 2816). As shown in figure 6-1, the input and output lines to the 2816 are as follows:

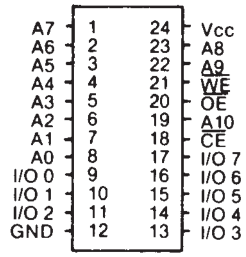


Figure 6-1

ADDRESS LINES (A0 thru A10)

These 11 address lines tell the 2816 which of the 2000 memory locations is to be written to or read from.

INPUT/OUTPUT LINES (I/O 0 thru I/O 7)

During a write operation, data on these lines is stored in the memory location on the address lines (A0 thru A10). Unlike the 8155, this integrated circuit uses different lines for address and data. During a read operation, the data specified by the address lines is placed on the I/O lines.

CHIP ENABLE ($\overline{CE}$)

The line over the CE is called a "not bar" and means the input is driven by a zero or is active low. A low on this line tells the 2816 that the commands present on the Write Enable ($\overline{WE}$) and Ouput Enable ($\overline{OE}$) are for this circuit and not for some other circuit.

OUTPUT ENABLE ($\overline{OE}$)

This line serves the same purpose as the $\overline{RD}$ line of the 8155. A low on the $\overline{OE}$ together with a low on the $\overline{CE}$ line and a high on the $\overline{WE}$ line gates the data stored in the specified address onto the Input/Output lines.

WRITE ENABLE ($\overline{WE}$)

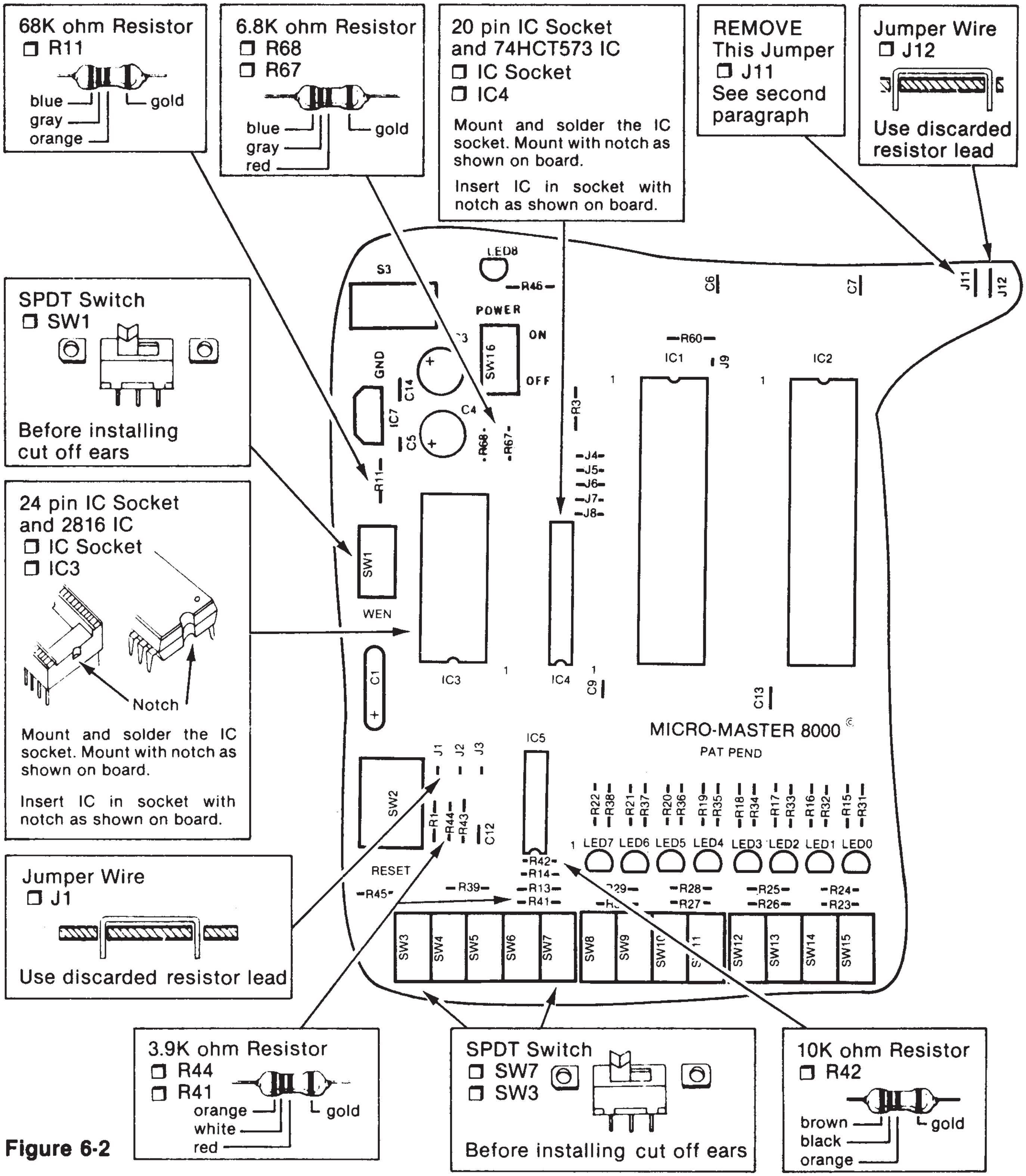
This line serves the same purpose as the $\overline{WR}$ of the 8155. A low on the $\overline{WE}$ line together with a low on the $\overline{CE}$ line and a high on the $\overline{OE}$ line stores the data present on the Input/Output lines into the specified address.

In lesson 3 the data bus was used for both address bytes and data bytes. First, the LSB (Least Significant Byte) of the address was placed on the data bus and latched into the internal address register of the 8155. Next, the data bus was used to transmit data to or from memory. The addressing of the 2816 differs in two significant ways. First, since the 2816 does not have an internal address register, an external address register must be provided. The MM-8000 uses the 74HCT573 Octal D-Type Transparent Latch for this purpose. Secondly three additional address lines A8, A9 and A10 of the MSB (Most Significant Byte) of the address are brought to the 2816. These are necessary to handle the larger memory capacity. With these exceptions, reading and writing to the 2816 is the same as to the 8155. The following experiment will demonstrate these facts.

ASSEMBLY INSTRUCTIONS

Unplug the power supply. Identify and install the following parts as shown in figure 6-2. After soldering each part place a check in the box provided.

To remove jumper J-11. Reset your desoldering pump and hold in one hand. With your soldering iron in the other hand heat one pad of J11, immediately put the tip of the pump in the melted solder and trigger the desoldering pump. Now remove the solder from the other pad. The holes should be clean of solder and the jumper wire free from the board.



68K ohm Resistor
□ R11

6.8K ohm Resistor
□ R68
□ R67

20 pin IC Socket and 74HCT573 IC
□ IC Socket
□ IC4

Mount and solder the IC socket. Mount with notch as shown on board.

Insert IC in socket with notch as shown on board.

REMOVE This Jumper
□ J11
See second paragraph

Jumper Wire
□ J12

Use discarded resistor lead

SPDT Switch
□ SW1

Before installing cut off ears

24 pin IC Socket and 2816 IC
□ IC Socket
□ IC3

Mount and solder the IC socket. Mount with notch as shown on board.

Insert IC in socket with notch as shown on board.

Jumper Wire
□ J1

Use discarded resistor lead

3.9K ohm Resistor
□ R44
□ R41

SPDT Switch
□ SW7
□ SW3

Before installing cut off ears

10K ohm Resistor
□ R42

Figure 6-2

If unit will not function properly see troubleshooting guide on page VII.

CIRCUIT DESCRIPTION

Figure 6-3 shows the components added in lesson 6. The 74HCT573 is an Octal D Type Transparent Latch which consist of 8 independent latches controlled by a control input (C). When C is high,the latches are transparent. A high on any D input will result in a high on the corresponding Q output. A low on a D input will likewise result in a low on the corresponding Q output. When C goes low, the data that was then present in the latch is retained on the Q outputs regardless of any later changes of the D input. The ALE bus is connected to the C input of the 74HCT573 integrated circuit. When the ALE key is pressed,the ALE bus goes high and the LSB of the address on the address-data bus is passed through the transparent latches to the address inputs A0 thru A7 of the 2816. When the ALE button is released,the address is retained at the A0 thru A7 inputs. The address-data bus is then free to pass data to or from the 2816 I/O pins. Address Lines A8, A9 and A10 are connected to the appropriate pins of the 2816. In a later lesson they will also be connected to the 8085 microprocessor. In this lesson,A9 and A10 are biased to a low voltage (0) through resistors R67 and R68. Line A8 is connected to the center of the A8 switch. When the switch is up,a high voltage (1) is applied to A8 through resistor R41. When the switch is down,a low voltage (0) is applied to A8 through resistor R42.

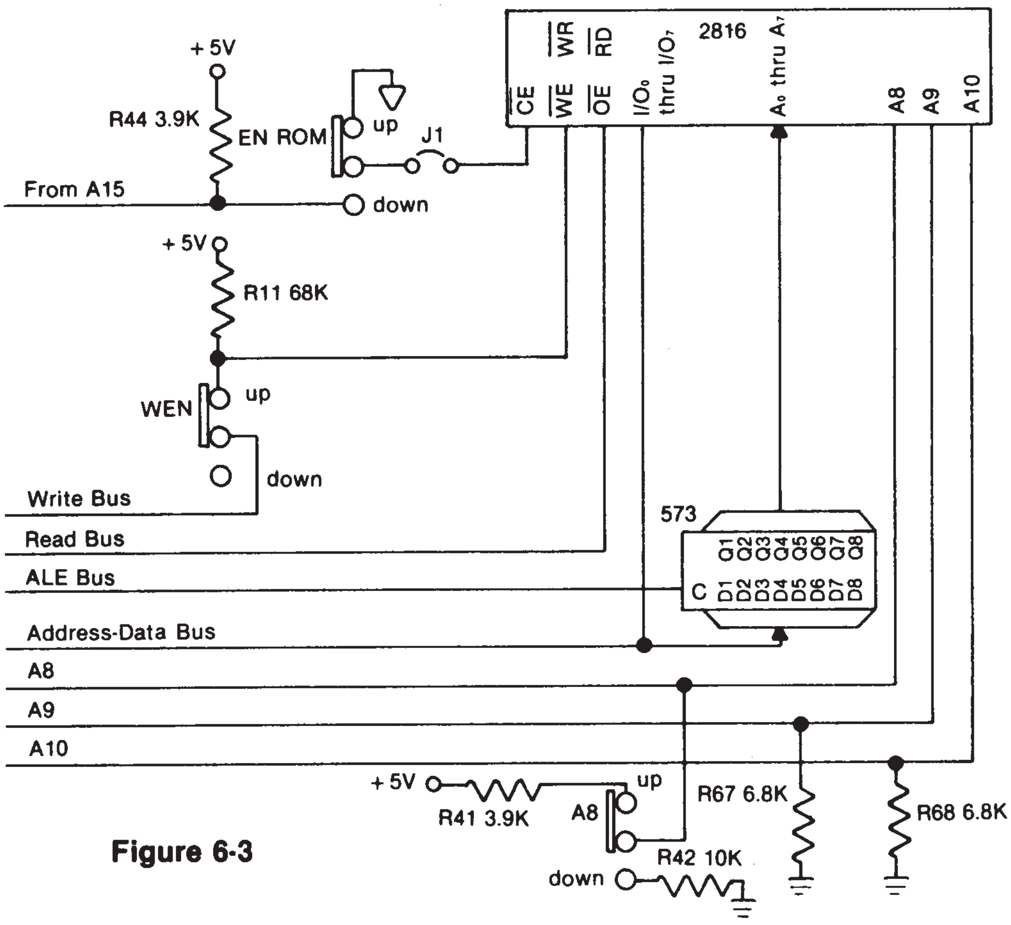


Figure 6-3

The Read bus is connected to the $\overline{OE}$ pin and is used to gate data from the 2816 onto the data bus. The $\overline{WE}$ input to the 2816 is tied to the top of the WEN switch. With the WEN switch in the up position,the $\overline{WE}$ input is connected to the write bus as shown in figure 3-3. The WR button can then be used to write data into the 2816 memory. Pushing the WR button ties the $\overline{WE}$ input to a low voltage (0) through the resistor network R10, R11 and R12. Releasing the WR button ties the $\overline{WE}$ input to a high voltage (1) through resistors R10 and R11. If the WEN switch is in the down position,the WR button is disconnected from the $\overline{WE}$ input. The $\overline{WE}$ input is tied to high voltage (1) through resistor R11 and writing is inhibited. To prevent accidental erasure of the 2816, the WEN switch should always be in the down position when power is turned ON or OFF.

The center of the Enable ROM (ENROM) switch is connected to the $\overline{CE}$ input of the 2816 through jumper wire J1. With the switch in the up position the $\overline{CE}$ is tied to a low voltage (ground). The 2816 will then respond to orders on the $\overline{WE}$ and $\overline{OE}$ lines. With the switch in the down position the $\overline{CE}$ input is tied to a high voltage (1) through resistor R44 and the 2816 is inhibited from responding to orders on the $\overline{WE}$ and $\overline{OE}$ lines. In a later lesson,address line A15 from the 8085 microprocessor will be used to enable and disable the 2816.

PROCEDURE

1. Put the ENROM and ENRAM switches in the down position to disable both ROM and RAM.
2. Put the WEN switch in the down position to insure that the 2816 is not written to during power turn on.
3. Connect the power supply and turn power ON.
4. Put the ENROM switch in the up position to enable the 2816 integrated circuit.
5. Put the WEN switch in the up position to allow writing to the 2816 integrated circuit.
6. Put the A8 switch in the down position in order to write data to memory locations 000H through 0FFH (where the H means Hexadecimal).
7. Set the data switches to 0000 0000.
8. Press the ALE button to latch the address on the data switches into the 74HCT573 Latch.
9. Set the data switches to 0000 0011.
10. Press the WR button to write this data to the 2816.
11. Repeat steps 7 thru 10 changing the addresses in step 7 and the data in step 9 in order to write the data shown in table 6-1 into the different address locations of the 2816.

Table 6-1

HEXADECIMAL ADDRESS	BINARY ADDRESS	DATA
0 00	000 0000 0000	0000 0011
0 01	000 0000 0001	0000 1100
0 02	000 0000 0010	0011 0000
0 03	000 0000 0011	1100 0000

12. Check the data stored in each address by repeating steps 7 and 8 to return to each address and then press the RD button to read the data stored at that address.
13. Put the WEN switch down to prevent further writing to the 2816 and insure that the 2816 will not be given an unwanted write command during power turn off or power turn on.
14. Turn power off.
15. Turn power on again.
16. Repeat step 12 to verify that the data stored on the 2816 was not lost when the power was removed.
17. Put the A8 switch up to change address line A8 to a 1. Since A9 and A10 are biased to 0. We are now set to write and read addresses 100H to 1FFH.
18. Put the WEN switch in the up position to allow writing to the 2816.
19. Repeat steps 7 through 10 changing the address in step 7 and the data in step 9 to write the data in table 6-2 to the 2816.

Table 6-2

HEXADECIMAL ADDRESS	BINARY ADDRESS	DATA
1 00	001 0000 0000	1000 0001
1 01	001 0000 0001	0100 0010
1 02	001 0000 0010	0010 0100
1 03	001 0000 0011	0001 1000

20. Repeat step 13.
21. Check the data stored in each address by repeating steps 7 and 8 to return to each address and then pushing the RD button.
22. Put the A8 switch down to return to addresses 000H through 0FFH and repeat step 12 to verify that the data of table 6-1 is still stored in these memory locations.
23. Turn power off.

LESSON 7

Lesson 7 describes the 8085A microprocessor. This description and Lesson 8 on the Instruction Set has been extracted from the Intel MSC-80/85TM Family Users Manual. If further information is desired, the Intel MCS-80/85TM Family Users Manual may be obtained from Intel Corporation Literature Department 3065 Bowers Ave. Santa Clara, California 95051.

8085A FUNCTIONAL DESCRIPTION

WHAT THE 8085A IS

The 8085A is an 8-bit general-purpose microprocessor that is very cost-effective in small systems because of its extraordinarily low hardware overhead requirements. At the same time it is capable of accessing up to 64K bytes of memory and has status lines for controlling large systems.

WHAT'S IN THE 8085A

In the 8085A microprocessor are contained the functions of clock generation, system bus control, and interrupt priority selection, in addition to execution of the instruction set. (See Figure 7-1.) The 8085A transfers data on an 8-bit, bi-directional 3-state bus (AD_0-AD_7) which is time-multiplexed so as to also transmit the eight lower-order address bits. An additional eight lines (A_8-A_{15}), expand the memory addressing capability to 16 bits, thereby allowing 64K bytes of memory to be accessed directly by the CPU. The 8085A CPU (central processing unit) generates control signals that can be used to select appropriate external devices and functions to perform READ and WRITE operations and also to select memory or I/O ports. The 8085A can address up to 256 different I/O locations. These addresses have the same numerical values (00 through FFH) as the first 256 memory addresses; they are distinguished by means of the IO/$\overline{M}$ output from the CPU. You may also choose to address I/O ports as memory locations (i.e., memory-map the I/O).

Registers

The 8085A, like the 8080, is provided with internal 8-bit registers and 16-bit registers. The 8085A has eight addressable 8-bit registers. Six of them can be used either as 8-bit registers or as 16-bit register pairs. Register pairs are treated as though they were single, 16-bit registers; the high-order byte of a pair is located in the first register and the low-order byte is located in the second. In addition to the register pairs, the 8085A contains two more 16-bit registers.

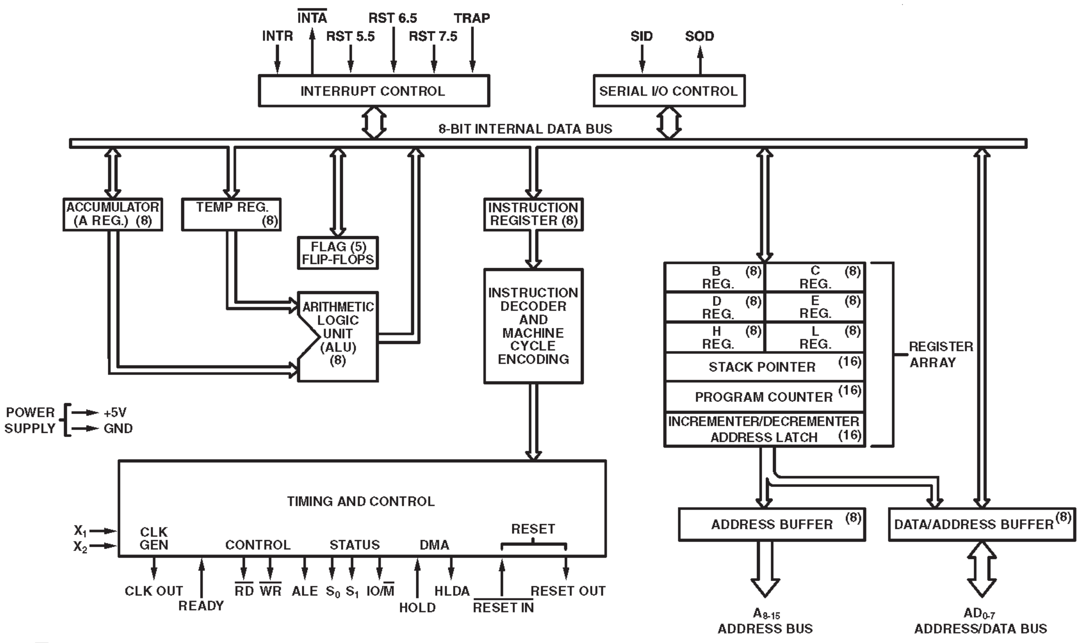


Figure 7-1 8085A CPU FUNCTIONAL BLOCK DIAGRAM

The 8085A's CPU registers are distinguished as follows:

The accumulator (ACC or A Register) is the focus of all of the accumulator instructions (Tables 8-1 and 8-2), which include arithmetic, logic, load and store, and I/O instructions. It is an 8-bit register only. (However, see Flags in this list).

The **program counter** (PC) always points to the memory location of the next instruction to be executed. It always contains a 16-bit address.

General-purpose registers BC, DE, and HL may be used as six 8-bit registers or as three 16-bit registers, interchangeably, depending on the instruction being performed. HL functions as a **data pointer** to reference memory addresses that are either the sources or the destinations in a number of instructions. A smaller number of instructions can use BC or DE for indirect addressing.

The **stack pointer** (SP) is a special data pointer that always points to the stack top (next available stack address). It is an indivisible 16-bit register.

The flag register contains five one-bit flags, each of which records processor status information and may also control processor operation. (See following paragraph).

Flags
The five flags in the 8085A CPU are shown:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

The carry flag (CY) is set and reset by arithmetic operations. Its status can be directly tested by a program. For example, the addition of two one-byte numbers can produce an answer that does not fit into one byte:

HEXIDECIMAL	BINARY
AEH	1 0 1 0 1 1 1 0
+74H	0 1 1 1 0 1 0 0
122H	1 0 0 1 0 0 0 1 0

Carry bit sets carry flag to 1

An addition operation that results in an overflow out of the high-order bit of the accumulator sets the carry flag. An addition operation that does not result in an overflow clears the carry flag. (SeeLesson 8 for further details.) The carry flag also acts as a 'borrow' flag for subtract operations.

The **auxiliary carry flag** (AC) indicates overflow out of bit 3 of the accumulator in the same way that the carry flag indicates overflow out of bit 7. This flag is commonly used in BCD (binary coded decimal) arithmetic.

The **sign flag** is set to the condition of the most significant bit of the accumulator following the execution of arithmetic or logic instructions. These instructions use bit 7 of data to represent the sign of the number of contained in the accumulator. This permits the manipulation of numbers in the range from -128 to + 127.

The **zero flag** is set if the result generated by certain instructions is zero. The zero flag is cleared if the result is not zero. A result that has a carry but has a zero answer byte in the accumulator will set both the carry flag and the zero flag. For example,

HEXADECIMAL	BINARY
A7H	1 0 1 0 0 1 1 1
+ 59H	+ 0 1 0 1 1 0 0 1
100H	1 0 0 0 0 0 0 0 0

Carry bit
Eight zero bits set zero flag to 1

Incrementing or decrementing certain CPU registers with a zero result will also set the zero flag.

The **parity flag** (P) is set to 1 if the parity (number of 1-bits) of the accumulator is even. If odd, it is cleared.

Stack

The stack pointer maintains the address of the last byte entered into the stack. The stack pointer can be initialized to use any portion of read-write memory as a stack. The stack pointer is decremented each time data is pushed onto the stack and is incremented each time data is popped off the stack (i.e., the stack grows downward in terms of memory address, and the stack 'top' is the lowest numerical address represented in the stack currently in use). Note that the stack pointer is always incremented or decremented by two bytes since all stack operations apply to register pairs.

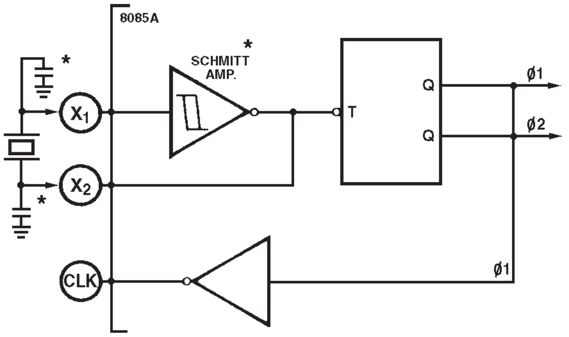
Arithmetic-Logic Unit (ALU)

The ALU contains the accumulator, the flag register, and some temporary registers that are inaccessible to the programmer.

Arithmetic, logic, and rotate operations are performed by the ALU. The results of these operations can be deposited in the accumulator, or they can be transferred to the internal data bus for use elsewhere.

Instruction Register and Decoder

During an instruction fetch, the first byte of an instruction (containing the opcode) is transferred from the internal bus to the 8-bit instruction register. (See Figure 7-1). The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, gated by timing signals, controls the registers, ALU, and data and address buffers. The outputs of the instruction decoder and internal clock generator generate the state and machine cycle timing signals.



*EXTERNAL CAPACITORS REQUIRED ONLY FOR CRYSTAL FREQUENCIES - 4MHz

Figure 7-2 8085A CLOCK LOGIC

Internal Clock Generator

The 8085A CPU incorporates a complete clock generator on its chip, so it requires only the addition of a quartz crystal to establish timing for its operation. (It will accept an external clock input at its X_1 input instead, however). A suitable crystal for the standard 8085A must be parallel-resonant at a fundamental of 6.25 MHz or less, twice the desired internal clock frequency. The 8085A-2 will operate with crystal of up to 10 MHz. The functions of the 8085A internal clock generator are shown in Figure 7-2. A Schmitt trigger is used interchangeably as oscillator or as input conditioner, depending upon whether a crystal or an external source is used. The clock circuitry generates two nonoverlapping internal clock signals $\phi_1$and ϕ_2 (see Figure 7-2).ϕ_1 andϕ_2 control the internal timing of the 8085A and are not directly available on the outside of the chip. The external pin CLK is a buffered, inverted version ofϕ_1. CLK is half the frequency of the crystal input signal, and may be used for clocking other devices in the system.

If the clock frequency requirement is not stringent enough to require a crystal, an RC network may be used. The RC network shown in Figure 10-3 gives a frequency of approximately 3MHz.

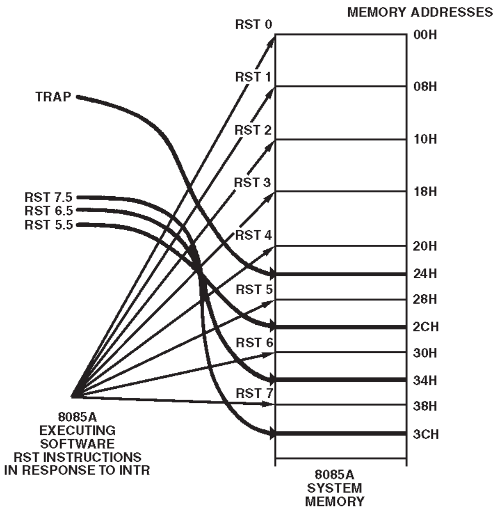


Figure 7-3 8085A HARDWARE AND SOFTWARE RST BRANCH LOCATIONS

Interrupts

The five hardware interrupt inputs provided in the 8085A are of three types. INTR is maskable (can be enabled or disabled by EI or DI software instructions), and causes the CPU to fetch in an RST instruction, externally placed on the data bus, which vectors a branch to any one of eight fixed memory locations (Restart addresses). (See Figure 7-3). INTR can also be controlled by the 8259 programmable interrupt controller, which generates CALL instructions instead of RSTs, and can thus vector operation of the CPU to a preprogrammed subroutine located anywhere in your systems's memory map. The RST 5.5, RST 6.5 and RST 7.5 hardware interrupts are different in function in that they are maskable through the use of the SIM instruction, which enables or disables these interrupts by clearing or setting corresponding mask flags based on data in the accumulator. (See Figure 7-4.) You may read the status of the interrupt mask previously set by performing a RIM instruction. Its execution loads into the accumulator the following information. (See Figure 7-5).

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware status.
- Current interrupt enable flag status (except that immediately following TRAP, the IE flag status **preceding** that interrupt is loaded).
- RST 5.5, 6.5, and 7.5 interrupts pending.

RST 5.5, 6.5, and 7.5 are also subject to being enabled or disabled by the EI and DI instructions, respectively. INTR, RST 5.5, and RST 6.5 are level-sensitive, meaning that these inputs may be acknowledged by the processor when they are held at a high level. RST 7.5 is edge-sensitive, meaning that an internal flip-flop in the 8085A registers the occurrence of an interrupt the instant a rising edge appears on the RST 7.5 input line. This input need not be held high; the flip-flop will remain set until it is cleared by one of three possible actions:

- The 8085A responds to the interrupt, and sends an internal reset signal to the RST 7.5 flip-flop. (See Figure 7-6A).
- The 8085A, before responding to the RST 7.5 interrupt, receives a $\overline{\text{RESET IN}}$ signal from an external source; this also activates the internal reset.
- The 8085A executes a SIM instruction, with accumulator bit 4 previously set to 1. (See Figure 7-4).

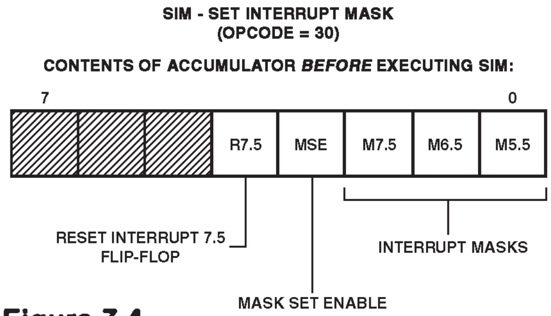


Figure 7-4 INTERRUPT MASKS SET USING SIM INSTRUCTION

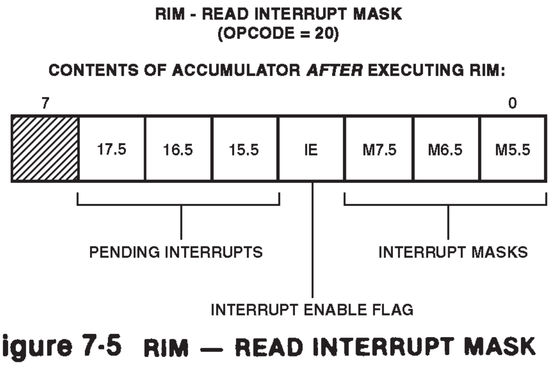


Figure 7-5 RIM — READ INTERRUPT MASK

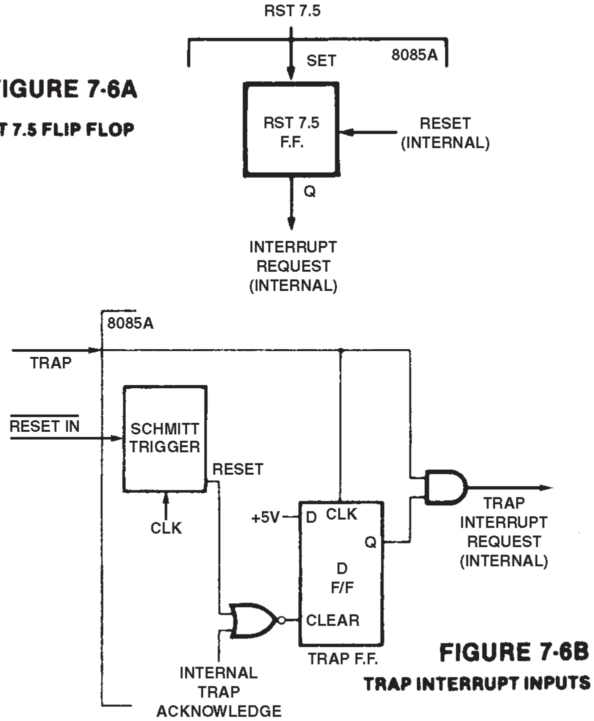


FIGURE 7-6A
RST 7.5 FLIP FLOP

FIGURE 7-6B
TRAP INTERRUPT INPUTS

Figure 7-6 RST 7.5 AND TRAP INTERRUPT INPUTS

The third type of hardware is TRAP. This input is not subject to any mask or interrupt enable/disable instruction. The receipt of a positive-going edge on the TRAP input triggers the processor's hardware interrupt sequence, but the pulse must be held high until acknowledged internally (see Figure 7-6B).

The sampling of all interrupts occurs on the descending edge of CLK, one cycle before the end of the instruction in which the interrupt input is activated. To be recognized, a valid interrupt must occur at least 160 ns before sampling time in the 8085A, or 150 ns in the 8085A-2. This means that to guarantee being recognized, RST 5.5 and 6.5 and TRAP need to be held on for at least 17 clock states plus 160 ns (150 for 8085A-2). assuming that the interrupt might arrive just barely too late to be acknowledged during a particular instruction, and that the following instruction might be an 18-state CALL. This timing assumes no WAIT or HOLD cycles are used.

The way interrupt masks are set and read is described in Lesson 8 under the RIM (read interrupt mask) and SIM (set interrupt mask) instruction listings. Interrupt functions and their priorities are shown in the table at right.

Name	Priority	Address (1) Branched to when interrupt occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled
RST 7.5	2	3CH	Rising edge (latched)
RST 6.5	3	34H	High level until sampled
RST 5.5	4	2CH	High level until sampled
INTR	5	(2)	High level until sampled

NOTES:

(1) In the case of TRAP and RST 5.5-7.5, the contents of the Program Counter are pushed onto the stack before the branch occurs.

(2) Depends on the instruction that is provided to the 8085A by the 8259 or other circuitry when the interrupt is acknowledged.

Serial Input and Output

The SID and SOD pins help to minimize chip count in small systems by providing for easy interface to a serial port using software for timing and coding and decoding of the data. Each time a RIM instruction is executed, the status of the SID pin is read into bit 7 of the accumulator. RIM is thus a dual-purpose instruction. (See Lesson 8). In similar fashion, SIM is used to latch bit 7 of the accumulator out of the SOD output via an internal flip-flop, providing that bit 6 of the accumulator is set to 1. (See Figure 7-7). SID and SOD timing will be described later.

SID can also be used as a general purpose TEST input and SOD can serve as a one-bit control output.

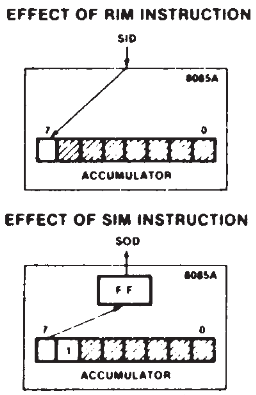
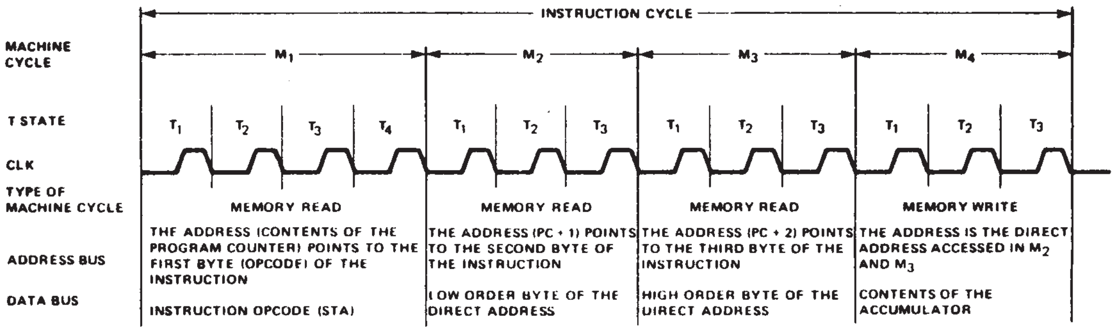


Figure 7-7
EFFECT OF RIM AND SIM INSTRUCTIONS ON SERIAL DATA LINES

Multiplexed Bus Cycle Timing

The execution of any 8085A program consists of a sequence of READ and WRITE operations, of which each transfers a byte of data between the 8085A and a particular memory or I/O address. These READ and WRITE operations are the only communication between the processor and the other components, and are all that is necessary to execute any instruction or program.



INSTRUCTION CYCLE
MACHINE CYCLE
M1
M2
M3
M4
T STATE
T1
T2
T3
T4
T1
T2
T3
T1
T2
T3
T1
T2
T3
CLK
TYPE OF MACHINE CYCLE
MEMORY READ
MEMORY READ
MEMORY READ
MEMORY WRITE
ADDRESS BUS
THE ADDRESS (CONTENTS OF THE PROGRAM COUNTER) POINTS TO THE FIRST BYTE (OPCODE) OF THE INSTRUCTION
THE ADDRESS (PC + 1) POINTS TO THE SECOND BYTE OF THE INSTRUCTION
THE ADDRESS (PC + 2) POINTS TO THE THIRD BYTE OF THE INSTRUCTION
THE ADDRESS IS THE DIRECT ADDRESS ACCESSED IN M2 AND M3
DATA BUS
INSTRUCTION OPCODE (STA)
LOW ORDER BYTE OF THE DIRECT ADDRESS
HIGH ORDER BYTE OF THE DIRECT ADDRESS
CONTENTS OF THE ACCUMULATOR

Figure 7-8 CPU TIMING FOR STORE ACCUMULATOR DIRECT (STA) INSTRUCTION

Each READ or WRITE operation of the 8085A is referred to as a machine cycle. The execution of each instruction by the 8085A consists of a sequence of from one to five machine cycles, and each machine cycle consists of a minimum of from three to six clock cycles (also referred to as T states). Consider the case of the Store Accumulator Direct (STA) instruction, shown in Figure 7-8. The STA instruction causes the contents of the accumulator to be stored at the direct address specified in the second and third bytes of the instruction. During the first machine cycle (M_1), the CPU puts the contents of the program counter (PC) on the address bus and performs a MEMORY READ cycle to read from memory the opcode of the next instruction (STA). The M_1 machine cycle is also referred to as the OPCODE FETCH cycle, since it fetches the operation code of the next instruction. In the fourth clock cycle (T_4) of M_1 the CPU interprets the data read in and recognizes it as the opcode of the STA instruction. At this point the CPU knows that it must do three more machine cycles (two MEMORY READs and one MEMORY WRITE) to complete the instruction.

The 8085A then increments the program counter so that it points to the next byte of the instruction and performs a MEMORY READ machine cycle (M_2) at address (PC + 1). The accessed memory places the addressed data on the data bus for the CPU. The 8085A temporarily stores this data (which is the low-order byte of the direct address) internally in the CPU. The 8085A again increments the program counter to location (PC + 2) and reads from memory (M_3) the next byte of data, which is the high-order byte of the direct address.

At this point, the 8085A has accessed all three bytes of the STA instruction, which it must now execute. The execution consists of placing the data accessed in M_2 and M_3 on the address bus, then placing the contents of the accumulator on the data bus, and then performing a MEMORY WRITE machine cycle (M_4). When M_4 is finished, the CPU will fetch (M_1) the first byte of the next instruction and continue from there.

State Transition Sequence

As the preceding example shows, the execution of an instruction consists of a series of machine cycles whose nature and sequence is determined by the opcode accessed in the M_1 machine cycle. While no one instruction cycle will consist of more than five machine cycles, every machine cycle will be one of the seven types listed in Figure 7-9. These seven types of machine cycles can be differentiated by the state of the three status lines (IO/$\overline{M}$,S_0,andS_1) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$).

MACHINE CYCLE		STATUS			CONTROL		
		IO/$\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
OPCODE FETCH	(OF)	0	1	1	0	1	1
MEMORY READ	(MR)	0	1	0	0	1	1
MEMORY WRITE	(MW)	0	0	1	1	0	1
I/O READ	(IOR)	1	1	0	0	1	1
I/O WRITE	(IOW)	1	0	1	1	0	1
INTR ACKNOWLEDGE	(INA)	1	1	1	1	1	0
BUS IDLE	(BI): DAD	0	1	0	1	1	1
	INA(RST/TRAP)	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

0 = Logic "0" 1 = Logic "1" TS = High Impedance X = Unspecified

Figure 7-9 8085A MACHINE CYCLE CHART

Most machine cycles consist of three T states, (cycles of the CLK output) with the exception of OPCODE FETCH, which normally has either four or six T states. The actual number of states required to perform any instruction depends on the instruction being executed, the particular machine cycle within the instruction cycle, and the number of WAIT and HOLD states inserted into each machine cycle through the use of the READY and HOLD inputs of the 8085A. The state transition diagram in Figure 7-10 illustrates how the 8085A proceeds in the course of a machine cycle. The state of various status and control signals, as well as the system buses, is shown in Figure 7-11 for each of the ten possible T states that the processor can be in.

Figure 7-10 also shows when the READY, HOLD, and interrupt signals are sampled, and how they modify the basic instruction sequence (T_1-T_6 and T_{WAIT}). As we shall see, the timings for each of the seven types of machine cycles are almost identical.

Figure 7-10 8085A CPU STATE TRANSISTION

NOTE: SYMBOL DEFINITION

T_X = CPU STATE T_X ALL CPU STATE TRANSISTIONS OCCUR ON THE FALLING EDGE OF CLK

◇X = A DECISION (X) THAT DETERMINES WHICH SEVERAL ALTERNATIVE PATHS TO FOLLOW.

$\boxed{X}$ = PERFORM THE ACTION X.

→ = FLOWLINE THAT INDICATES THE SEQUENCE OF EVENTS.

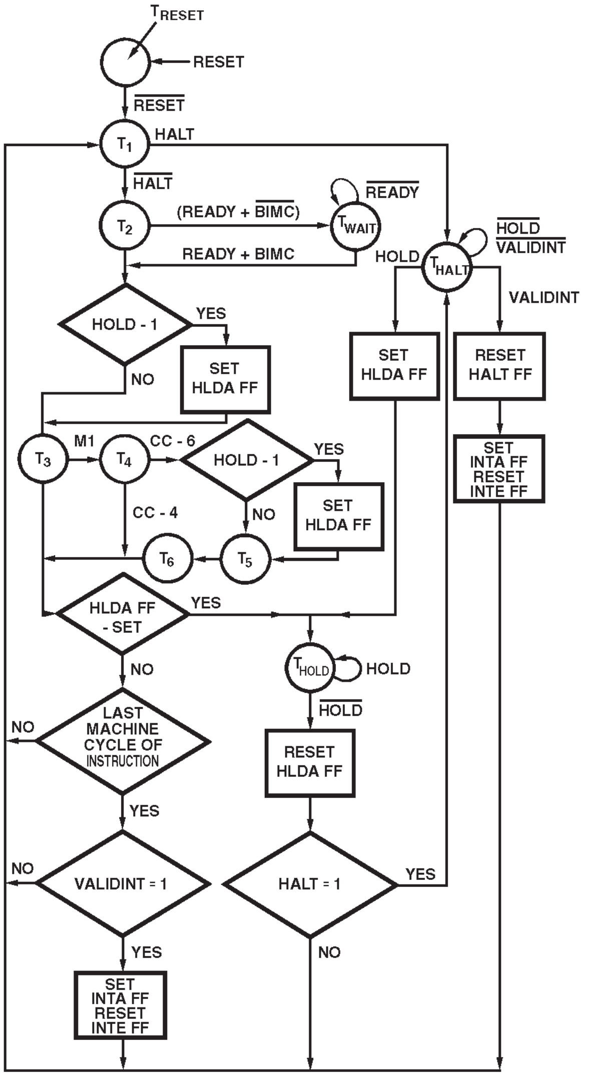
$\xrightarrow{X}$ = FLOWLINE THAT INDICATES THE SEQUENCE OF EVENTS IF CONDITION X IS TRUE.

CC = NUMBER OF CLOCK CYCLES IN THE CURRENT MACHINE CYCLE.

BIMC = "BUS IDLE MACHINE CYCLE" = MACHINE CYCLE WHICH DOESN'T USE THE SYSTEM BUS.

VALIDINT = "VALID INTERRUPT" - AN INTERRUPT IS PENDING THAT IS BOTH ENABLED AND UNMASKED (MASKING ONLY APPLIES FOR RST 5.5, 6.5, AND 7.5 INPUTS).

HLDA FF = INTERNAL HOLD ACKNOWLEDGE FLIP FLOP. NOTE THAT THE 8085A SYSTEM BUSES ARE 3-STATED ONE CLOCK CYCLE AFTER THE HLDA FLIP FLOP IS SET.



Machine State	Status & Buses				Control		
	S1,S0	$IO/\overline{M}$	A_8-A_{15}	AD_0-AD_7	$\overline{RD}$,$\overline{WR}$	$\overline{INTA}$	ALE
T_1	X	X	X	X	1	1	1†
T_2	X	X	X	X	X	X	0
T_{WAIT}	X	X	X	X	X	X	0
T_3	X	X	X	X	X	X	0
T_4	1	0*	X	TS	1	1	0
T_5	1	0*	X	TS	1	1	0
T_6	1	0*	X	TS	1	1	0
T_{RESET}	X	TS	TS	TS	TS	1	0
T_{HALT}	0	TS	TS	TS	TS	1	0
T_{HOLD}	X	TS	TS	TS	TS	1	0

0=Logic "0" 1=Logic "1" TS=High Impedance X=Unspecified

† ALE not generated during 2nd and 3rd machine cycles of DAD instruction.

*** $IO/\overline{M}$ = 1 during T_4 - T_6 states of RST and INA cycles.**

Figure 7-11 8085A MACHINE STATE CHART

OPCODE FETCH (OF):

The OPCODE FETCH (OF) machine cycle is unique in that it has more than three clock cycles. This is because the CPU must interpret the opcode accessed in T_1, T_2, and T_3 before it can decide what to do next.

Figure 7-12 shows the timing relationships for an OF machine cycle. The particular instruction illustrated is DCX, whose timing for OF differs from other instructions in that it has six T states, while some instructions require only four T states for OF. In this discussion, as well as the following discussions, only the relative timing of the signals will be discussed; for the actual timings, refer to Intel MCS-80/85TM Family Users Manual.

The first thing that the 8085A does at the beginning of every machine cycle is to send out three status signals ($IO/\overline{M}$, S1, S0) that define what type of machine cycle is about to take place. The $IO/\overline{M}$ signal identifies the machine cycle as being either a memory reference or input/output operation. The S1 status signal identifies whether the cycle is a READ or WRITE operation. The S0 and S1 status signals can be used together (see Figure 7-9) to identify READ, WRITE, or OPCODE FETCH machine cycles as well as the HALT state. Referring to Figure 7-12, the 8085A will send out $IO/\overline{M}$ = 0, S1 = 1, S0 = 1 at the beginning of the machine cycle to identify it as a READ from a memory location to obtain an opcode; in other words, it identifies the machine cycle as an OPCODE FETCH cycle.

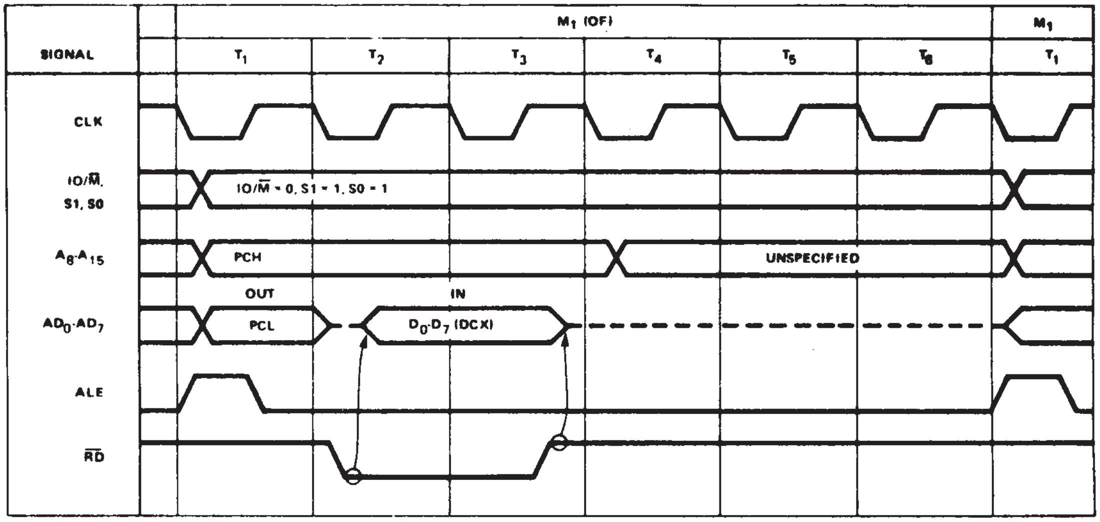


Figure 7-12 **OPCODE FETCH MACHINE CYCLE (OF DCX INSTRUCTION)**

The 8085A also sends out a 16-bit address at the beginning of every machine cycle to identify the particular memory location or I/O port that the machine cycle applies to. In the case of an OF cycle, the contents of the program counter is placed on the address bus. The high order byte (PCH) is placed on the A_8-A_{15} lines, where it will stay until at least T_4. The low order byte (PCL) is placed on the AD_0-AD_7 lines, whose three-state drivers are enabled if not found already on. Unlike the upper address lines, however, the information on the lower address lines will remain there for only one clock cycle, after which drivers will go to their high impedance state, indicated by a dashed line in Figure 7-12. This is necessary because the AD_0-AD_7 lines are time mulitplexed between the address and data buses. During T_1 of every machine cycle, AD_0-AD_7 output the lower 8-bits of address after which AD_0-AD_7 will either output the desired data for a WRITE operation or the drivers will float (as is the case for the OF cycle), allowing the external device to drive the lines for a READ operation.

Since the address information on AD_0,-AD_7 is of a transitory nature, it must be latched either internally in special multiplexed-bus components like the 8155 or externally in parts like the 8212 8-bit latch. The 8085A provides a special timing signal, ADDRESS LATCH ENABLE (ALE), to facilitate the latching of A_0,-A_7; ALE is present during T_1 of every machine cycle.

After the status signals and address have been sent out and the AD_0-AD_7 drivers have been disabled, the 8085A provides a low level on $\overline{RD}$ to enable the addressed memory device. The device will then start driving the AD_0-AD_7 lines; this is indicated by the dashed line turning into a solid line in Figure 7-12. After a period of time (which is the access time of the memory) valid data will be present on AD_0-AD_7. The 8085A during T_3 will load the memory data on AD_0-AD_7 into its instruction register and then raise $\overline{RD}$ to the high level, disabling the addressed memory device. At this point, the 8085A will have finished accessing the opcode of the instruction. Since this is the first machine cycle (M_1) of the instruction, the CPU will automatically step to T_4 as shown in Figure 7-10.

During T_4, the CPU will decode in the intruction register and decide whether to enter T_5 on the next clock or to start a new machine cycle and enter T_1. In the case of the DCX instruction shown in Figure 7-12, it will enter T_5 and then T_6 before going to T_1.

During T_5 and T_6 of DCX, the CPU will decrement the designated register. Since the A_8-A_{15} lines are driven by the address latch circuits, which are part of the incrementer/decrementer logic, the A_8-A_{15} lines may change during T_5 and T_6. Because the value of A_8-A_{15} can vary during T_4-T_6, it is most important that all memory and I/O devices on the system bus qualify their selection with $\overline{\text{RD}}$. If they don't use $\overline{\text{RD}}$, they may be spuriously selected. Moreover, with a linear selection technique (See Lesson 9), two or more devices could be simultaneously enabled, which could be potentially damaging. The generation of spurious addresses can also occur momentarily at address bus transitional periods in T_1. Therefore, the selection of all memory and I/O devices must be qualified with $\overline{\text{RD}}$ or $\overline{\text{WR}}$. Many new memory devices like the 8155 and 8355 have the RD input that internally is used to enable the data bus outputs, removing the need for externally qualifying the chip enable input with $\overline{\text{RD}}$.

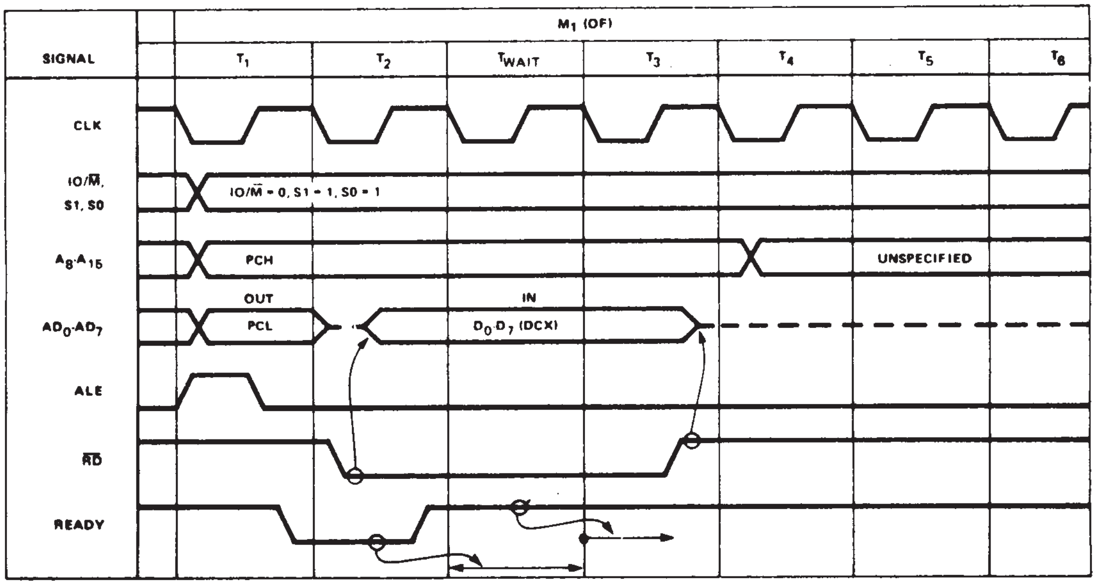


Figure 7-13 OPCODE FETCH MACHINE CYCLE WITH ONE WAIT STATE

Figure 7-13 is identical to Figure 7-12 with one exception, which is the use of the READY line. As we can see in Figure 7-10, when the CPU is in T_2, it examines the state of the READY line. If the READY line is high, the CPU will proceed to T_3 and finish executing the instruction. If the READY line is low, however, the CPU will enter T_{WAIT} and stay there indefinitely until READY goes high. When the READY line does go high, the CPU will exit T_{WAIT} and enter T_3, in order to complete the machine cycle. As shown in Figure 7-13, the external effect of using the READY line is to preserve the exact state of the processor signals at the end of T_2 for an integral number of clock periods, before finishing the machine cycle. This 'stretching' of the system timing has the further effect of increasing the allowable access time for memory or I/O devices. By inserting T_{WAIT} states, the 8085A can accommodate even the slowest of memories. Another common use of the READY line is to single-step the processor with a manual switch.

Read Cycle Timing

MEMORY READ (MR):

Figure 7-14 shows the timing of two successive MEMORY READ (MR) machine cycles, the first without a T_{WAIT} state and the second with one T_{WAIT} state. The timing during T_1-T_3 is absolutely identical to the OPCODE FETCH machined cycle, with the exception that the status sent out during T_1 is $IO/\overline{M} = 0$, $S1 = 1$, $S0 = 0$, identifying the cycles as a READ from a memory location. This differs from Figure 7-12 only in that $S0 = 1$ for an OF cycle, identifying that cycle as an OPCODE FETCH operation. Otherwise, the two cycles are identical during T_1-T_3.

A second difference occurs at the end of T_3. As shown in Figure 7-10, the CPU always goes to T_4 from T_3 during M_1 which is always an OF cycle. During all other machine cycles, the CPU will always go from T_3 to T_1 of the next machine cycle.

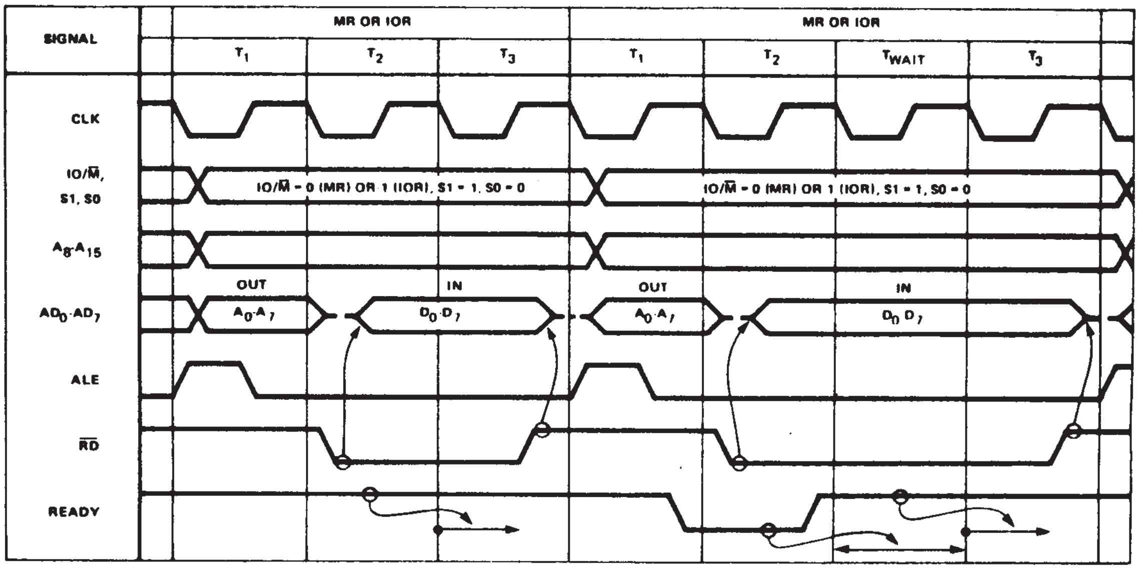


Figure 7-14 MEMORY READ (OR I/O READ) MACHINE CYCLES (WITH AND WITHOUT WAIT STATES)

The memory address used in the OF cycle is always the contents of the program counter, which points to the current instruction, while the address used in the MR cycle can have several possible origins. Also, the data read in during an MR cycle is placed in the appropriate register, not the instruction register.

I/O READ (IOR):

Figure 7-14 also shows the timing of two successive I/O READ (IOR) machine cycles, the first without a T_{WAIT} state. As is readily apparent, the timing of an IOR cycle is identical to the timing of an MR cycle, with the exception of $IO/\overline{M}=0$ for MR and $IO/\overline{M}=1$ for IOR; recall that $IO/\overline{M}$ status signal identifies the address of the current machine cycle as selecting either a memory location or an I/O port. The address used in the IOR cycle comes from the second byte (Port No.) of an INPUT instruction. Note that the I/O port address is duplicated onto both AD_0-AD_7 and A_8-A_{15}. The IOR cycle can occur only as the third machine cycle of an INPUT instruction.

Note that the READY signal can be used to generate T_{WAIT} states for I/O devices as well as memory devices. By gating the READY signal with the proper status lines, one could generate T_{WAIT} states for memory devices only or for I/O devices only. By gating in the address lines, one can further qualify T_{WAIT} state generation by the particular devices being accessed.

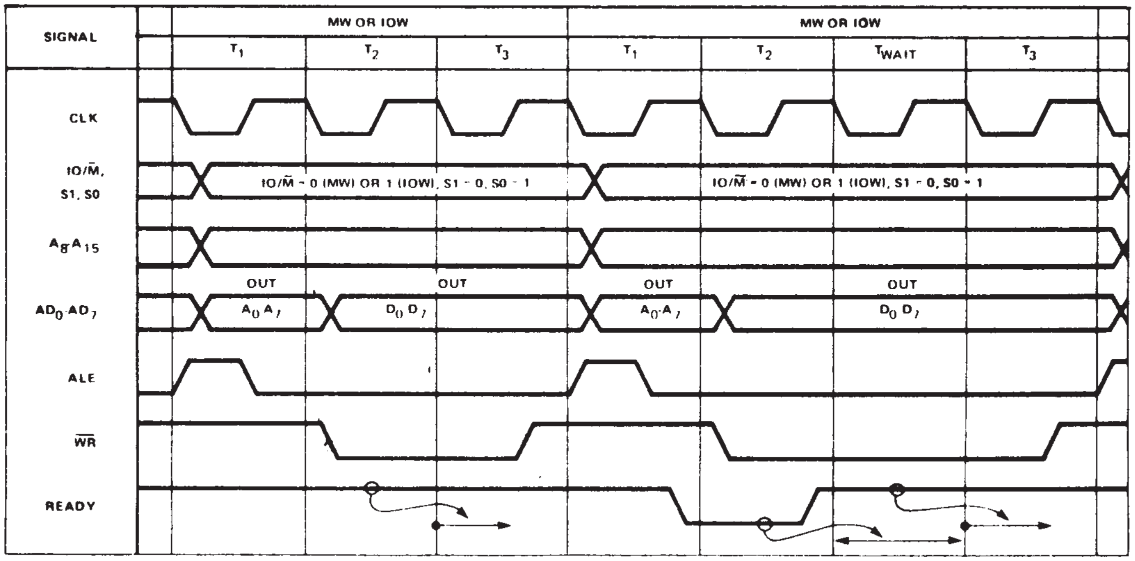


Figure 7-15 MEMORY WRITE (OR I/O WRITE) MACHINE CYCLES (WITH AND WITHOUT WAIT STATES)

WRITE Cycle Timing

MEMORY WRITE (MW):

Figure 7-15 shows the timing for two successive MEMORY WRITE (MW) machine cycles, the first without a T_{WAIT} state, and the second with one T_{WAIT} state. The 8085A sends out the status during T_1 in a similar fashion to the OF, MR and IOR cycles, except that IO/$\overline{M}$ = 0, S1 = 0, and S0 = 1, identifying the current machine cycle as being a WRITE operation to a memory location.

The address is sent out during T_1 in an identical manner to MR. However, at the end of T_1, there is a difference. While the AD_0-AD_7 drivers were disabled during T_2-T_3 of MR in expectation of the addressed memory device driving the AD_0-AD_7 lines, the drivers are not disabled for MW. This is because the CPU must provide the data to be written into the addressed memory location. The data is placed on AD_0-AD_7 at the start of T_2. The $\overline{WR}$ signal is also lowered at this time to enable the writing of the addressed memory device. During T_2, the READY line is checked to see if a T_{WAIT} state is required. If READY is low, T_{WAIT} states are inserted until READY goes high. During T_3, the $\overline{WR}$ line is raised, disabling the addressed memory device and thereby terminating the WRITE operation. The contents of the address and data lines are not changed until the next T_1, which directly follows.

Note that the data on AD_0-AD_7 is not guaranteed to be stable before the falling edge of $\overline{WR}$. The AD_0-AD_7 lines are guaranteed to be stable both before and after the rising edge of $\overline{WR}$.

I/O WRITE (IOW):

As Figure 7-15 shows, the timing for an I/O WRITE (IOW) machine cycle is the same as an MW machine cycle except that IO/$\overline{M}$ = 0 during the MW cycle and IO/$\overline{M}$ = 1 during the IOW cycle.

As with the IOR cycle discussed previously, the address used in an IOW cycle is the I/O port number which is duplicated on both the high and low bytes of the address bus. In the case of IOW, the port number comes from the second byte of an OUTPUT instruction as the instruction is executed.

Interrupt Acknowledge (INA) timing

Figures 7-16 and 7-17 (a continuation of 7-16) depict the course of action the CPU takes in response to a high level on the INTR line if the INTE FF (interrupt enable flip-flop) has been set by the EI instruction. The status of the TRAP and RST pins as well as INTR is sampled during the second clock cycle before $M_1 \bullet T_1$. If INTR was the only valid interrupt and if INTE FF is set, then the CPU will reset INTE FF and then enter and INTERRUPT ACKNOWLEDGE (INA) machine cycle. The INA cycle is identical to an OF cycle with two exceptions. $\overline{INTA}$ is sent out instead of $\overline{RD}$. Also, IO/$\overline{M}$ = 1 during INA, whereas IO/$\overline{M}$ = 0 for OF. Although the contents of the program counter are sent out on the address lines, the address lines can be ignored.

When $\overline{INTA}$ is sent out, the external interrupt logic must provide the opcode of an instruction to execute. The opcode is placed on the data bus and read in by the processor. If the opcode is the first byte of a multiple-byte instruction, additional $\overline{INTA}$ pulses will be provided by the 8085A to clock in the remaining bytes. RESTART and CALL instructions are the most logical choices, since they both force the processor to push the contents of the program counter onto the stack before jumping to a new location. In Figure 7-16 it is assumed that a CALL opcode is sent to the CPU during M_1. The CALL opcode could have been placed there by a device like the 8259 programmable interrupt controller.

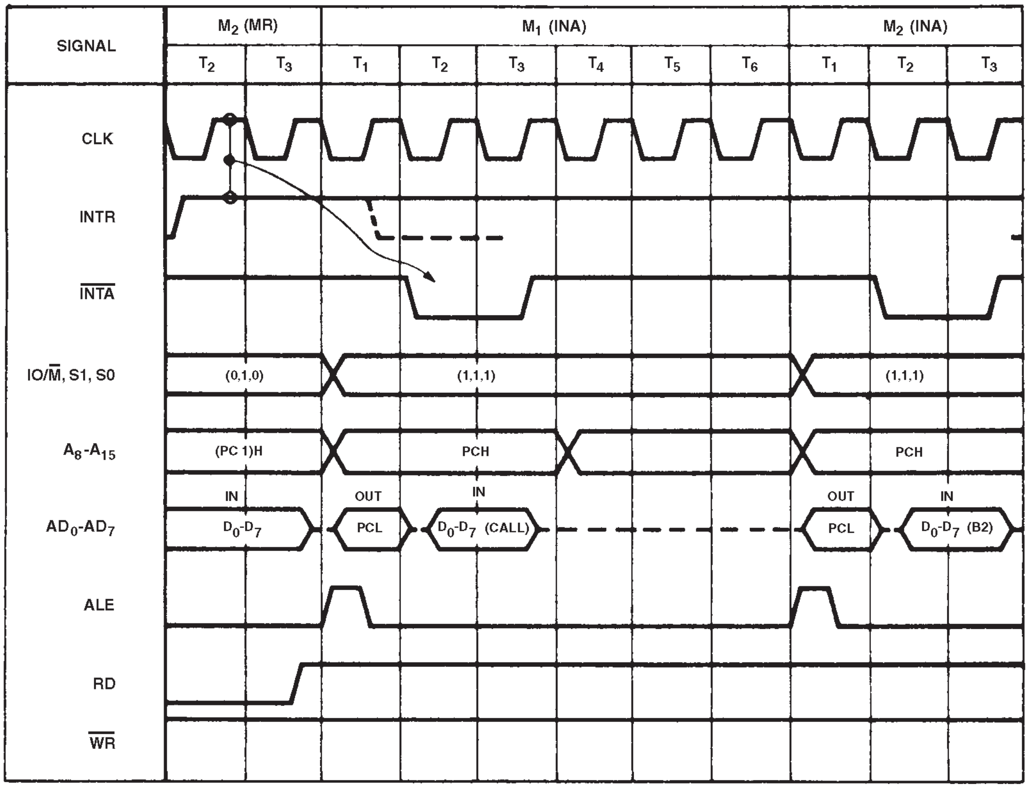


Figure 7-16 INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)

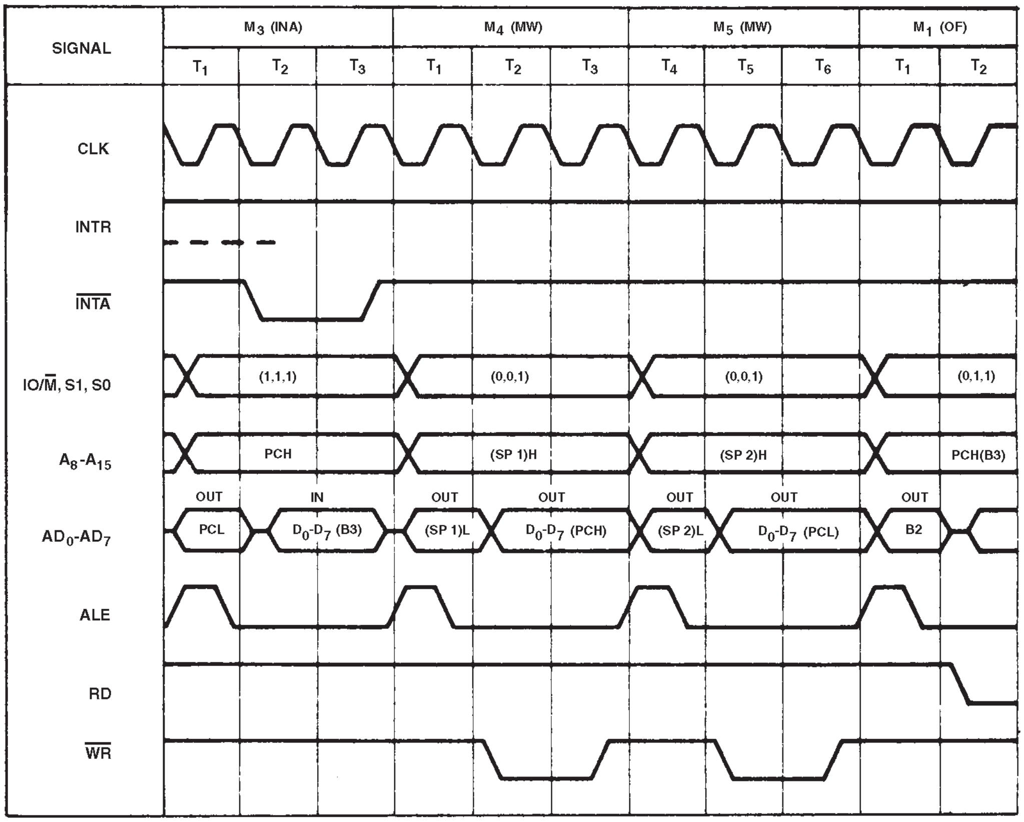


Figure 7-17 INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)

After receiving the opcode, the processor then decodes it and determines, in this case, that the CALL instruction requires two more bytes. The CPU therefore performs a second INA cycle (M_2) to access the second byte of the instruction from the 8259. The timing of this cycle is identical to M_1, except that it has only three T states. M_2 is followed by another INA cycle (M_3) to access the third byte of the CALL instruction from the 8259.

Now that the CPU has accessed the entire instruction used to acknowledge the interrupt, it will execute that instruction. Note that any instruction could be used (except EI or DI, the instructions which enable or diable interrupts), but the RESTART and CALL instructions are the most logical choices. Also notice that the CPU inhibited the incrementing of the program counter (PC) during the three INA cycles, so that the correct PC value can be pushed onto the stack during M_4 and M_5.

During M_4 and M_5, the CPU performs MEMORY WRITE machine cycles to write the upper and then lower bytes of the PC onto the top of the stack. The CPU then places the two bytes accessed in M_2 and M_3 into the lower and upper bytes of the PC. This has the effect of jumping the execution of the program to the location specified by the CALL instruction.

Bus Idle (BI) and HALT State

Most machine cycles of the 8085A are associated with either a READ or WRITE operation. There are two exceptions to this rule. The first exception takes place during M_2 and M_3 of the DAD instruction. The 8085A requires six internal T states to execute a DAD instruction, but it is not desirable to have M_1 be ten (four normal plus six extra) states long. Therefore, the CPU generates two extra machine cycles that do not access either the memory or the I/O. These cycles are referred to as BUS IDLE (BI) machine cycles. In the case of DAD, they are identical to MR cycles except that RD remains high and ALE is not generated. Note that READY is ignored during M_2 and M_3 of DAD.

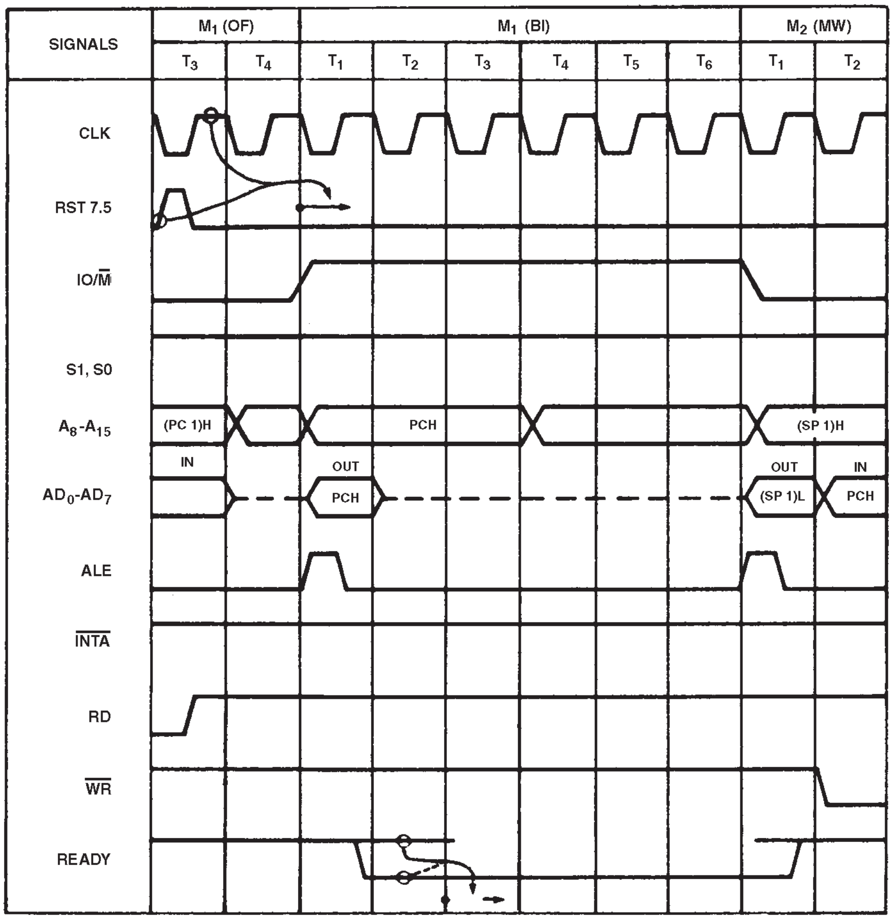


Figure 7-18 RST 7.5 BUS IDLE MACHINE CYCLE

The other time when the BUS IDLE machine cycle occurs is during the internal opcode generation for the RST or TRAP interrupts. Figure 7-18 illustrates the BI cycle generated in response to RST 7.5. Since this interrupt is rising-edge-triggered, it sets an internal latch; that latch is sampled at the falling edge of the next to the last T-state of the previous instruction. At this point the CPU must generate its own internal RESTART instruction which will (in subsequent machine cycles) cause the processor to push the program counter on the stack and to vector to location 3CH. To do this, it executes on OF machine cycle without issuing RD, generating the RESTART opcode instead. After M_1, the CPU continues execution normally in all respects except that the state of the READY line is ignored during the BI cycle.

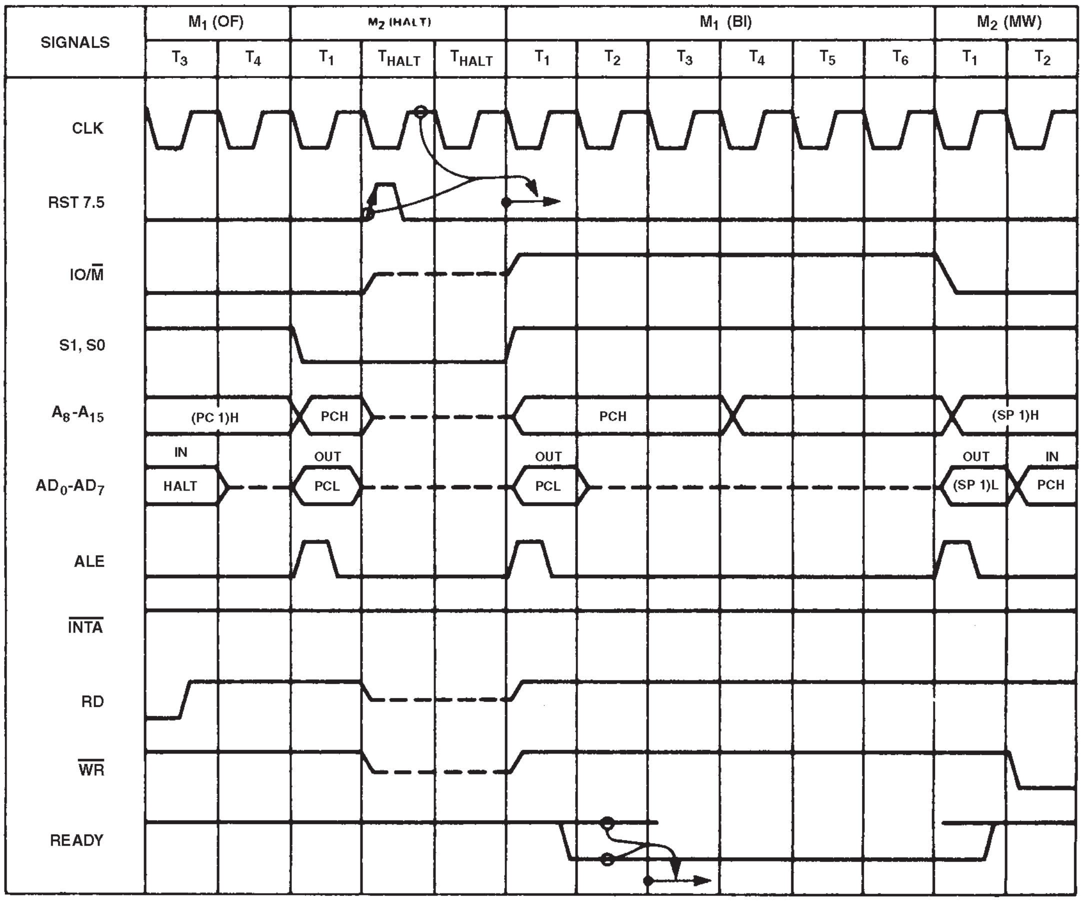


Figure 7-19 HALT STATE AND BUS IDLE MACHINE CYCLE RST 7.5 TERMINATES T_{HALT} STATE

Figure 7-19 illustrates the BI cycle generated in response to RST 7.5 when a HALT instruction has just been executed and the CPU is in the T_{HALT} state, with its various signals floating. There are only two ways the processor can completely exit the T_{HALT} state, as shown in Figure 7-10. The first way is for RESET to occur, which always forces the 8085A to T_{RESET}. The second way to exit T_{HALT} permanently is for a valid interrupt to occur, which will cause the CPU to disable further interrupts by resetting INTE FF, and to then proceed to $M_1 \bullet T_1$ of the next instruction. When the HOLD input is activated, the CPU will exit T_{HALT} for the duration of T_{HOLD} and then return to T_{HALT}.

In Figure 7-19 the RST 7.5 line is pulsed during T_{HALT}. Since RST 7.5 is a rising-edge-triggered interrupt, it will set an internal latch which is sampled during CLK = '1' of every T_{HALT} state (as well as during CLK = '1' two T states before any $M_1 \bullet T_1$). The fact that the latched interrupt was high (assuming that INTE FF = 1 and the RST 7.5 mask = 0) will force the CPU to exit the T_{HALT} state at the end of the next CLK period, and to enter $M_1 \bullet T_1$.

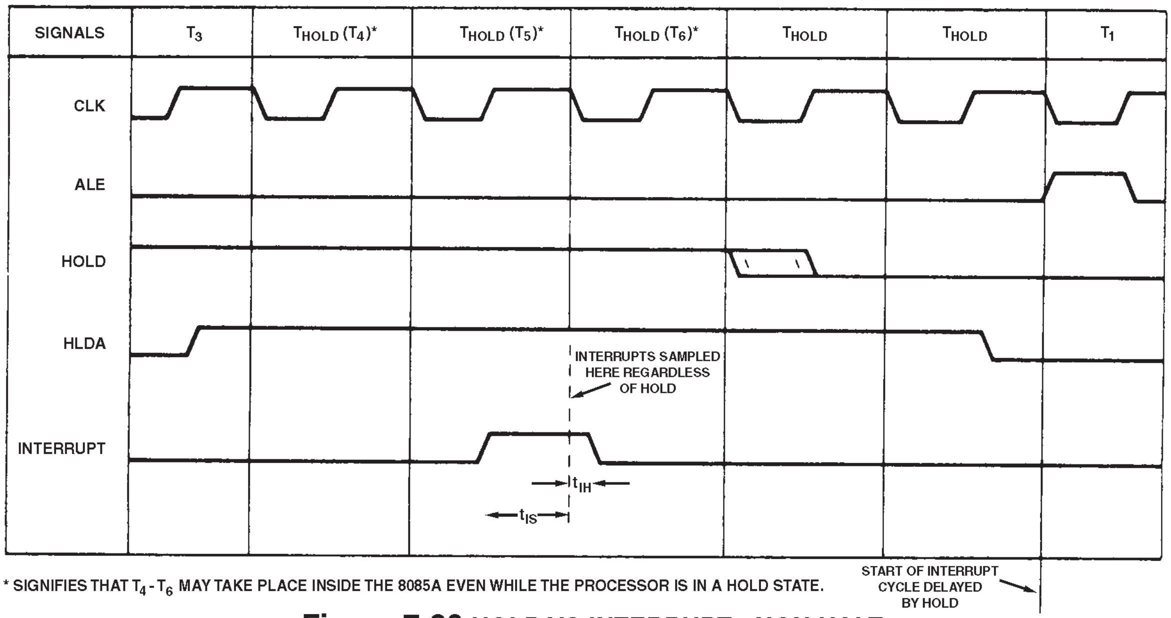
This completes our analysis of the timing of each of the seven types of machine cycles.

The 8085A uses the T_{HOLD} state to momentarily cease executing machine cycles, allowing external devices to gain control of the bus and perform DMA cycles. The processor internally latches the state of the HOLD line and the unmasked interrupts during CLK = '1' of every T_{HALT} state. If the internal latched HOLD signal is high during CLK'1' of any T_{HALT} state, the CPU will exit T_{HALT} and enter T_{HOLD} on the following CLK = '1'. As shown in Figure 7-20 this will occur even if a valid interrupt occurs simultaneously with the HOLD signal.

The state of the HOLD and the unmasked interrupt lines is latched internally during CLK = 1 of each T_{HOLD} state as well as during T_{HALT} states. If the internal latched HOLD signal is low during CLK = 1, the CPU will exit T_{HOLD} and enter T_{HALT} on the following CLK = 1.

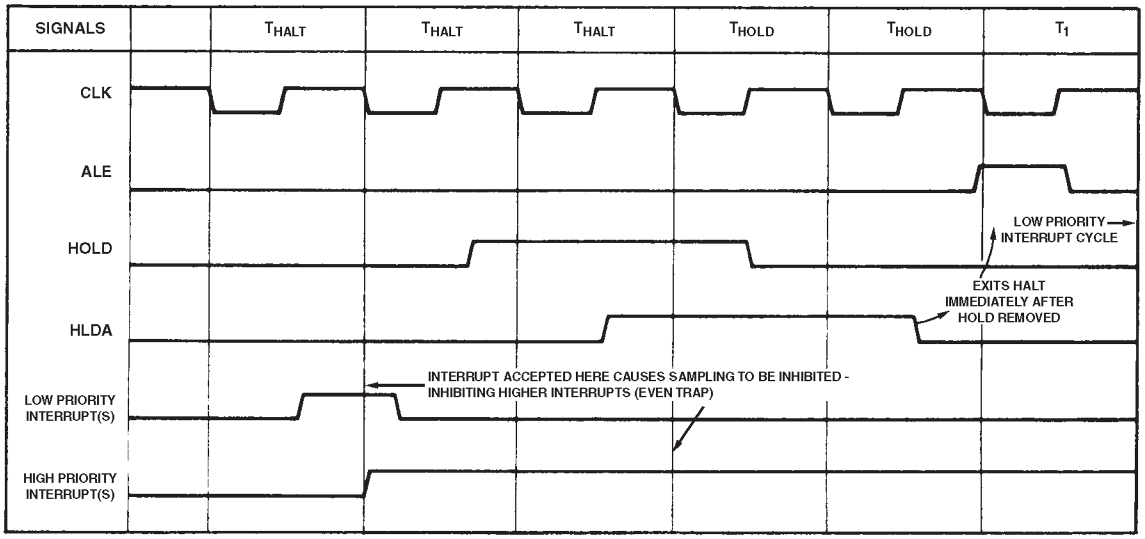
The 8085A accepts the first unmasked, enabled interrupt sampled; thereafter, all interrupt sampling is inhibited. The interrupt thus accepted will inevitably be executed when the CPU exits the HOLD state, even at the expense of holding off higher-priority interrupts (including TRAP). See Figure 7-21.

When the CPU is not in T_{HALT} or T_{HOLD}, it internally latches the HOLD line only during CLK = 1 of the last state before T_3 (T_2 or T_{WAIT}) and during CLK = 1 of the last state before T_5 (T_4 of a six T-state M_1). If the internal latched HOLD signal is high during the next CLK = 1, the CPU will enter T_{HOLD} after the following clock. When the CPU is not in T_{HALT} or T_{HOLD}, it will internally latch the state of the unmasked interupts only during CLK of the next to the last state before each $M_1 \bullet T_1$.



* SIGNIFIES THAT T_4 - T_6 MAY TAKE PLACE INSIDE THE 8085A EVEN WHILE THE PROCESSOR IS IN A HOLD STATE.

Figure 7-20 HOLD VS INTERRUPT - NON HALT


SIGNALS
THALT
THALT
THALT
THOLD
THOLD
T1
CLK
ALE
HOLD
HLDA
LOW PRIORITY INTERRUPT(S)
HIGH PRIORITY INTERRUPT(S)
LOW PRIORITY INTERRUPT CYCLE
EXITS HALT IMMEDIATELY AFTER HOLD REMOVED
INTERRUPT ACCEPTED HERE CAUSES SAMPLING TO BE INHIBITED - INHIBITING HIGHER INTERRUPTS (EVEN TRAP)

Power On and RESET IN

The 8085A employs a special internal circuit to increase its speed. This circuit, which is called a substrate bias generator, creates a negative voltage which is used to negatively bias the substrate. The circuit employs an oscillator and a charge pump which require a certain amount of time after POWER ON to stabilize. (See Figure 7-22).

Taking this circuit into account, the 8085A is not guaranteed to work until 10 ms after Vcc reaches 4.75V. For this reason, it is suggested that RESET IN be kept low during this period. Note that the 10 ms period does not include the time it takes for the power supply to reach its 4.75V level - which may be milliseconds in some systems. A simple RC network can satisfy this requirement.

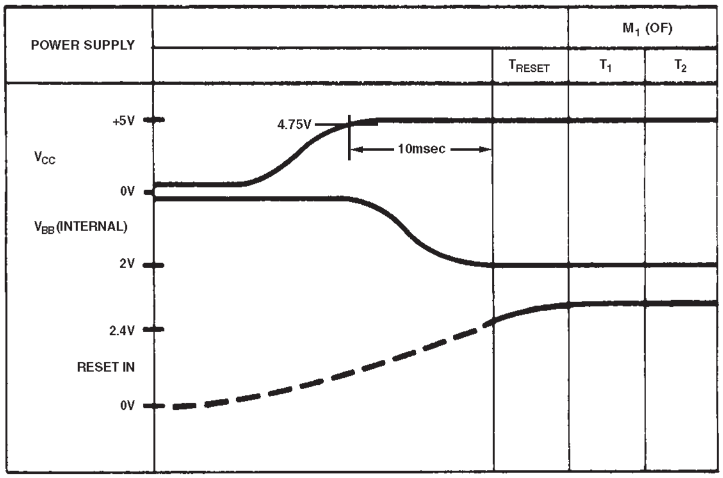
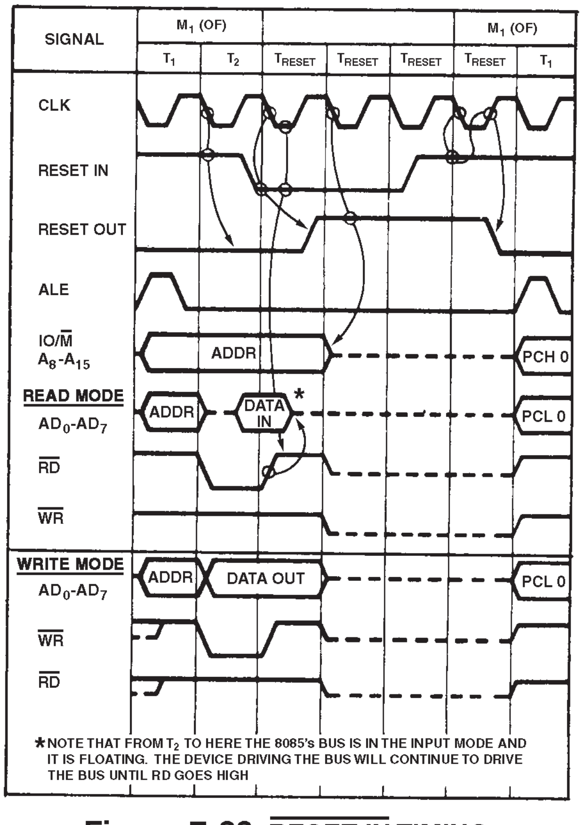


Figure 7-22 POWER-ON TIMING

The RESET IN line is latched every CLK = 1. This latched signal is recognized by the CPU during CLK = 1 of the next T state. (See Figure 7-23). If it is low, the CPU will issue RESET OUT and enter T_{HALT} for the next T state. RESET IN should be kept low for a minimum of three clock periods to ensure proper synchronization of the CPU. When the RESET IN signal goes high, the CPU will enter $M_1 \bullet T_1$ for the next T state. Note that the various signals and buses are floated in T_{RESET} as well as T_{HALT} and T_{HOLD}. For this reason, it is desirable to provide pull-up resistors for the main control signals (particularly WR).



*NOTE THAT FROM T_2 TO HERE THE 8085's BUS IS IN THE INPUT MODE AND IT IS FLOATING. THE DEVICE DRIVING THE BUS WILL CONTINUE TO DRIVE THE BUS UNTIL RD GOES HIGH

Figure 7-23 RESET IN TIMING

Specifically, the RESET IN signal causes the following actions:

RESETS	SETS
PROGRAM COUNTER	RST 5.5 MASK
INSTRUCTION REGISTER	RST 6.5 MASK
INTE FF	RST 7.5 MASK
RST 7.5 FF	
TRAP FF	
SOD FF	
MACHINE STATE FF's	
MACHINE CYCLE FF's	
INTERNALLY LATCHED FF's for HOLD,INTR, and READY	

RESET IN does not explicitly change the contents of the 8085A registers (A,B,C,D,E,H,L) and the condition flags, but due to RESET IN occurring at a random time during instruction execution, the results are indeterminate.

Following RESET, the 8085A will start executing instructions at location 0 with the interrupt system disabled, as shown in Figure 7-23.

Figure 7-23 also shows READ and WRITE operations being terminated by a RESET signal. Note that a RESET may prematurely terminate any READ or WRITE operation in process when the RESET occurs.

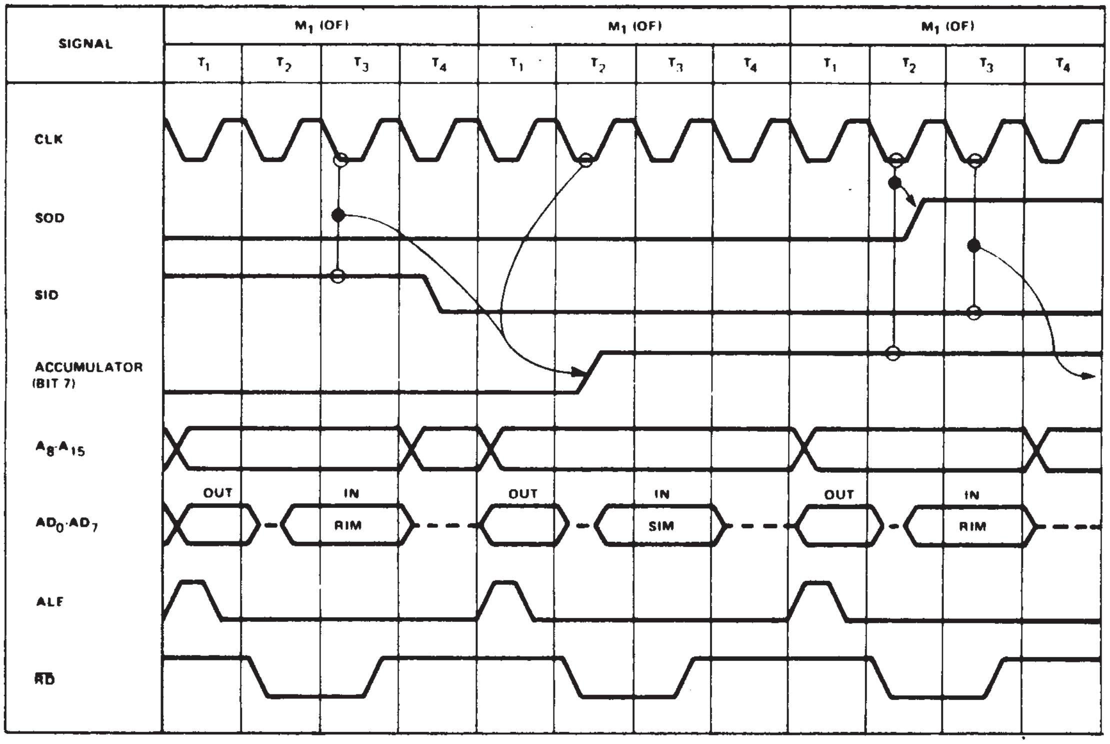
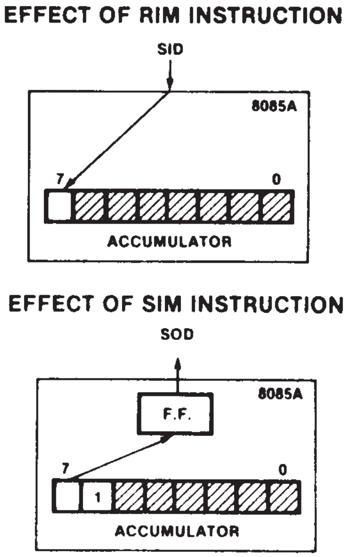


Figure 7-24 RELATIONSHIP OF SID AND SOD SIGNALS TO RIM AND SIM INSTRUCTIONS

SID and SOD Signals

Figure 7-24 shows the timing relationship of the SID and SOD signals to the RIM and SIM instructions. The 8085A has the ability to read the SID line into the accumulator bit 7 using RIM instructions. The state of the SID line is latched internally during $T_3 \bullet CLK = 0$ of the RIM instruction. Following this, the state of the interrupt pins and masks are also transferred directly to the accumulator.

The 8085A can set the SOD flip-flop from bit 7 of the accumulator using the SIM instruction. See Figure 7-25. The data is transferred from the accumulator bit 7 to SOD during $M_1 \bullet T_2 \bullet CLK = 0$ of the instruction following SIM, assuming that accumulator bit 6 is a 1. Accumulator bit 6 is a 'serial output enable' bit.


EFFECT OF RIM INSTRUCTION

EFFECT OF SIM INSTRUCTION

Figure 7-25

EFFECT OF RIM AND SIM INSTRUCTIONS

8085A FUNCTIONAL PIN DEFINITION

The following describes the function of each pin, see Figure 7-26.

A_8-A_{15} (Output, 3-state)

Address Bus: The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.

AD_0-AD_7 (Input/Output, 3-state)

Multiplexed Address/Data Bus: Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.

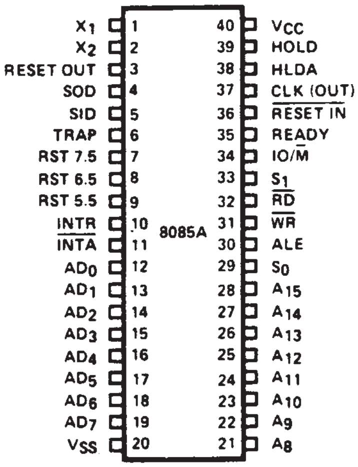


Figure 7-26 8085A Pinout Diagram

ALE (Output)

Address Latch Enable: It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-clip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.

S_0, S_1 and $IO/\overline{M}$ (Output)

Machine cycle status:

$IO/\overline{M}$	S_1	S_0	STATUS
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge
★	0	0	Halt
★	X	X	Hold
★	X	X	Reset

★ = 3-state (high impedance)
X = unspecified

S_1 can be used as an advanced $R/\overline{W}$ status. $IO/\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.

$\overline{RD}$ (Output, 3-state)

READ control: A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.

$\overline{WR}$ (Output, 3-state)

WRITE control: A low level on $\overline{WR}$ indicates the data on the Data Bus is to be writtten into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.

READY (Input)

If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an intergral number of clock cycles for READY to go high before completing the read or write cycle.

HOLD (Input)
HOLD indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data, $\overline{\text{RD}}$, $\overline{\text{WR}}$ and IO/$\overline{\text{M}}$ lines are 3-stated.

HLDA (Output)
HOLD ACKNOWLEDGE: Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low.

INTR (Input)
INTERRUPT REQUEST: is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

$\overline{\text{INTA}}$ (Output)
INTERRUPT ACKNOWLEDGE: Is used instead of (and has the same timing as) $\overline{\text{RD}}$ during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5, RST 6.5, RST 7.5 (Inputs)
RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inseted.

The priority of these interrupts is ordered as shown in Table 7-1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.

TRAP (Input)
Trap interrupt is a nonmaskable RESTART interrupt.It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. See Table 7-1.

$\overline{\text{RESET IN}}$ (Input)
Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{\text{RESET IN}}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay. The cpu is held in the reset condition as long as $\overline{\text{RESET IN}}$ is applied

RESET OUT (Output)
Indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.

X_1, X_2 (Input)

X_1 and X_2 are connected to a crystal, LC or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.

CLK (Output)

Clock Output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.

SID (Input)

Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (Output)

Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

V_{CC}

+5 volt supply.

V_{SS}

Ground Reference.

Table 7-1 INTERRUPT PRIORITY, RESTART ADDRESS, AND SENSITIVITY

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled.
RST 7.5	2	3CH	Rising edge (latched).
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note (2).	High level until sampled.

NOTES:

(1) The processor pushes the PC on the stack before branching to the indicated address.

(2) The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.

LESSON 8

WHAT THE INSTRUCTION SET IS

A computer, no matter how sophisticated, can do only what it is instructed to do. A program is a sequence of instructions, each of which is recognized by the computer and causes it to perform an operation. Once a program is placed in memory space that is accessible to your CPU, you may run that same sequence of instructions as often as you wish to solve the same problem or to do the same function. The set of instructions to which the 8085A CPU will respond is permanently fixed in the design of the chip.

Each computer instruction allows you to initiate the performance of a specific operation. The 8085A implements a group of instructions that move data between registers, between a register and memory, and between a register and an I/O port. It also has arithmetic and logic instructions, conditional and unconditional branch instructions, and machine control instructions. The CPU recognizes these instructions only when they are coded in binary form.

SYMBOLS AND ABBREVIATIONS:

The following symbols and abbreviations are used in the subsequent description of the 8085A instructions:

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r,r1,r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD = destination, SSS = source):

DDD or SSS	REGISTER NAME
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp One of the register pairs:

B represents the B,C pair with B as the high-order register and C as the low-order register;
D represents the D,E pair with D as the high-order register and E as the low-order register;
H represents the H,L pair with H as the high-order register and L as the low-order register;
SP represents the 16-bit stack pointer register.

RP The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh	The first (high order) register of a designated register pair.
rl	The second (low order) register of a designated register pair.
PC	16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).
SP	16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).
rm	Bit m of the register r (bits are number 7 through 0 from left to right).
	The condition flags:
Z	Zero
S	Sign
P	Parity
CY	Carry
AC	Auxiliary Carry
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
$\wedge$	Logical AND
$\veebar$	Exclusive OR
$\vee$	Inclusive OR
+	Addition
—	Two's complement subtraction
★	Multiplication
↔	"Is exchanged with"
——	The one's complement(e.g.,$(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

The instruction set encyclopedia is a detailed description of the 8085A instruction set. Each instruction is described in the following manner:
1.The instruction mnemonic and operand fields, are printed on the first line.
2. The name of the instruction is enclosed in parentheses following the mnemonic.
3. The next lines contain a symbolic description of what the instruction does.
4. This is followed by a narrative description of the operation of the instruction.
5. The boxes describe the binary codes that comprise the machine instruction
6. The last four lines contain information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a conditional jump, both times are listed, separated by a slash. Next, data addressing modes are listed if applicable. The last line lists any of the five flags that are affected by the execution of the instruction.

INSTRUCTION AND DATA FORMATS

Memory is organized in 8-bit bytes. Each byte has a unique location in physical memory. That location is described by one of a sequence of 16-bit binary addresses. The 8085A can address up to 64K (K = 1024, or2^{10};hence, 64K represents the decimal number 65,536) bytes of memory, which may consist of both random-access, read-write memory (RAM) and read-only memory (ROM), which is also random-access.

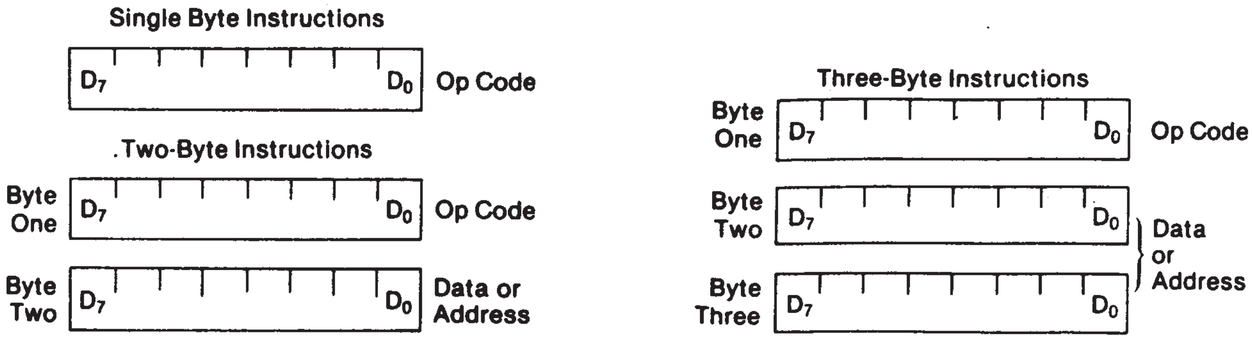
Data in the 8085A is stored in the form of 8-bit binary integers:

DATA WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MSB							LSB

When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the 8085A, BIT 0 is referred to as the **Least Significant Bit(LSB),** and BIT 7 (of an 8-bit number) is referred to as the **Most Significant Bit(MSB).**

An 8085A program instruction may be one, two or three bytes in length. Multiple-byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instruction. The exact instruction format will depend on the particular operation to be executed.



ADDRESSING MODES:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8085A has four different modes for addressing data stored in memory or in registers:

Direct — Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).

Register — The instruction specifies the register or register pair in which the data is located.

Reg Indirect — The instruction specifies a register pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair the low-order bits in the second).

Immediate — The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

Direct — The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)

Reg Indirect — The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

CONDITION FLAGS:

There are five condition flags associated with the execution of instructions on the 8085A. They are Zero, Sign, Parity, Carry, and Auxiliary Carry. Each is represented by a 1-bit register (or flip-flop) in the CPU. A flag is set by forcing the bit to 1; it is reset by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

Zero:	If the result of an instruction has the value 0, this flag is set; otherwise it is reset.
Sign:	If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.
Parity:	If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwse it is reset (i.e., if the result has odd parity).
Carry:	If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.
Aux Carry:	If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single-precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

INSTRUCTION SET ENCYCLOPEDIA

In the ensuing dozen pages, the complete 8085A instruction set is described, grouped in order under five different functional headings, as follows:

1. **Data Transfer Group** - Moves data between registers or between memory locations and registers. Includes moves, loads, stores, and exchanges. (See below.)
2. **Arithmetic Group** - Adds, subtracts, increments, or decrements data in registers or memory.
3. **Logic Group** - ANDs,ORs,XORs,compares, rotates, or complements data in registers or between memory and a register.
4. **Branch Group** - Initiates conditional or unconditional jumps, calls, returns, and restarts.
5. **Stack, I/O, and Machine Control Group** - Includes instructions for maintaining the stack, reading from input ports, writing to output ports, setting and reading interrupt masks, and setting and clearing flags.

Data Transfer Group

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected by any instruction in this group.**

MOV r1,r2 (Move Register)
(r1) ← (r2)
The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	none

MOV r,M (Move from memory)
(r) ← ((H)(L))
The content of the memory location, whose address is in registers H and L, is moved to register r.

0	1	D	D	D	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	none

MOV M,r (Move to memory)
((H)(L)) ← (r)
The content of register r is moved to the memory loction whose address is in registers H and L.

0	1	1	1	0	S	S	S

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	none

MVI r, data (Move Immediate)
(r) ← (byte 2)
The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	none

MVI M,data (Move to memory immediate)
((H)(L)) ← (byte 2)
The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles:	3
States:	10
Addressing:	immed./reg. indirect
Flags:	none

LXI rp,data 16 (Load register pair immediate)
(rh) ← (byte 3),
(rl) ← (byte 2)
Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

0	0	R	P	0	0	0	1
low-order data							
high-order data							

Cycles:	3
States:	10
Addressing:	immediate
Flags:	none

LDA addr (Load Accumulator direct)
(A) ← ((byte 3)(byte2))
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

Cycles:	4
States:	13
Addressing:	direct
Flags:	none

STA addr (Store Accumulator direct)
((byte 3)(byte2)) ← (A)
The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles:	4
States:	13
Addressing:	direct
Flags:	none

LHLD addr (Load H and L direct)
(L) ← ((byte 3)(byte 2))
(H) ← ((byte 3)(byte 2) + 1)
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

0	0	1	0	1	0	1	0
low-order addr							
high-order addr							

Cycles:	5
States:	16
Addressing:	direct
Flags:	none

SHLD addr (Store H and L direct)
((byte 3)(byte 2)) ← (L)
((byte 3)(byte 2) + 1) ← (H)
The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

Cycles:	5
States:	16
Addressing:	direct
Flags:	none

LDAX rp (load accumulator indirect)
(A) ← ((rp))
The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

0	0	R	P	1	0	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	none

STAX rp (Store accumulator indirect)
((rp)) ← (A)
The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

0	0	R	P	0	0	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	none

XCHG (Exchange H and L with D and E)
(H) ↔ (D)
(L) ↔ (E)
The contents of registers H and L are exchanged with the contents of registers D and E.

1	1	1	0	1	0	1	1

Cycles:	1
States:	4
Addressing:	register
Flags:	none

Arithmetic group
This group of instructions performs arithmetic operations on data in registers and memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxillary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add Register)
(A) ← (A) + (r)
The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	0	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

ADD M (Add memory)
(A) ← (A) + ((H)(L))
The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	0	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

ADI data (add immediate)
(A) ← (A) + (byte 2)
The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.

1	1	0	0	0	1	1	0
Data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

ADC r (Add Register with carry)
(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	1	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

ADC M (Add memory with carry)
(A) ← (A) + ((H)(L) + (CY)
The content of the memory location whose address is contained in the H and L resisters and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

1	0	0	0	1	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

ACI data (Add immediate with carry)
(A) ← (A) + (byte 2) + (CY)
The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

<table>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">data</td></tr>
</table>

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

SUB r (Subtract Register)
(A) ← (A) – (r)
The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

SUB M (Subtract memory)
(A) ← (A) – ((H)(L))
The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

SUI data (Subtract immediate)
(A) ← (A) – (byte 2)
The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

<table>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">data</td></tr>
</table>

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

SBB r (Subtract Register with borrow)
(A) ← (A) – (r) – (CY)
The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

SBB M (Subtract memory with borrow)
(A) ← (A) – ((H)(L)) – (CY)
The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

SBI data (Subtract immediate with borrow)
(A) ← (A) – (byte 2) – (CY)
The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	1	0	1	1	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

INR r (Increment Register)
(r) ← (r) + 1
The content of register r is incremented by one. Note condition flags **except CY** are affected.

0	0	D	D	D	1	0	0

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,AC

INR M (Increment memory)
((H)(L)) ← ((H)(L)) + 1
The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags **except CY** are affected.

0	0	1	1	0	1	0	0

Cycles:	3
States:	10
Addressing:	reg. indirect
Flags:	Z,S,P,AC

DCR r (Decrement Register)
(r) ← (r) – 1
The content of register r is decremented by one. Note: All condition flags **except CY** are affected.

0	0	D	D	D	1	0	1

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,AC

DCR M (Decrement memory)
((H)(L)) ← ((H)(L)) – 1
The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags **except CY** are affected.

0	0	1	1	0	1	0	1

Cycles:	3
States:	10
Addressing:	reg. indirect
Flags:	Z,S,P,AC

INX rp (Increment register pair)
(rh)(rl) ← (rh)(rl) + 1
The content of the register pair rp is incremented by one. Note: **No condition flags are affected.**

0	0	R	P	0	0	1	1

Cycles:	1
States:	6
Addressing:	register
Flags:	none

DCX rp (Decrement register pair)
(rh)(rl) ← (rh)(rl) – 1
The content of the register pair rp is decremented by one. Note: **No condition flags are affected.**

0	0	R	P	1	0	1	1

Cycles:	1
States:	6
Addressing:	register
Flags:	none

DAD rp (Add register pair to H and L)
(H)(L) ← (H)(L) + (rh)(rl)
The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add; otherwise it is reset.

0	0	R	P	1	0	0	1

Cycles:	3
States:	10
Addressing:	register
Flags:	CY

DAA (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the lease significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, **or** if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.

0	0	1	0	0	1	1	1

Cycles:	1
States:	4
Flags:	Z,S,P,CY,AC

Logical group

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

ANA r (AND Register)

(A)← (A)∧(r)

The content of register r is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

1	0	1	0	0	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

ANA M (AND) memory

(A)← (A)∧((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

1	0	1	0	0	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

ANI data (AND immediate)

(A)← (A)∧(byte 2)

The content of the second byte of the instruction is logically ANDed with the contents of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

1	1	1	0	0	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

XRA r (Exclusive OR Register)

(A)← (A)∀(r)

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	0	1	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

XRA M (Exclusive OR Memory)

(A)← (A)∀((H)(L))

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	0	1	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

XRI data (Exclusive OR immediate)

(A)← (A)∀(byte 2)

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	1	1	0	1	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

ORA r (OR Register)

$(A) \leftarrow (A) \vee (r)$

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	1	0	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

ORA M (OR memory)

$(A) \leftarrow (A) \vee ((H)(L)$

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	1	0	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

ORI data (OR immediate)

$(A) \leftarrow (A) \vee (\text{byte } 2)$

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	1	1	1	0	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

CMP r (Compare Register)

$(A) - (r)$

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A)** = (r). The CY flag is set to 1 if $(A) < (r)$.

1	0	1	1	1	S	S	S

Cycles:	1
States:	4
Addressing:	register
Flags:	Z,S,P,CY,AC

CMP M (Compare memory)

$(A) - ((H)(L))$

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if $(A) = ((H)(L))$. The CY flag is set to 1 if $(A) < ((H)(L))$.**

1	0	1	1	1	1	1	0

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	Z,S,P,CY,AC

CPI data (Compare immediate)

$(A) - (\text{byte } 2)$

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. **The Z flag is set to 1 if $(A) = (\text{byte } 2)$. The CY flag is set to 1 if $(A) < (\text{byte } 2)$.**

1	1.	1	1	1	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	Z,S,P,CY,AC

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n); (A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**

0	0	0	0	0	1	1	1

Cycles:	1
States:	4
Flags:	CY

RRC (Rotate right)

$(A_n) \leftarrow (A_{n+1}); (A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**

0	0	0	0	1	1	1	1

Cycles:	1
States:	4
Flags:	CY

RAL (Rotate left through carry)
$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$
$(A_0) \leftarrow (CY)$
The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**

0	0	0	1	0	1	1	1

Cycles:	1
States:	4
Flags:	CY

RAR (Rotate right through carry)
$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$
$(A_7) \leftarrow (CY)$
The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**

0	0	0	1	1	1	1	1

Cycles:	1
States:	4
Flags:	CY

CMA (Complement accumulator)
$(A) \leftarrow (\overline{A})$
The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**

0	0	1	0	1	1	1	1

Cycles:	1
States:	4
Flags:	none

CMC (Complement carry)
$(CY) \leftarrow (\overline{CY})$
The CY flag is complemented. **No other flags are affected.**

0	0	1	1	1	1	1	1

Cycles:	1
States:	4
Flags:	CY

STC (Set carry)
$(CY) \leftarrow 1$
The CY flag is set to 1. **No other flags are affected.**

0	0	1	1	0	1	1	1

Cycles:	1
States:	4
Flags:	CY

Branch Group
This group of instructions alter normal sequential program flow.

Condition flags are not affected by any instruction in this group.

The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ - not zero (Z = 0)	000
Z - zero(Z = 1)	001
NC - no carry (CY = 0)	010
C - carry (CY = 1)	011
PO - parity odd (P = 0)	100
PE - parity even (P = 1)	101
P - plus (S = 0)	110
M - minus (S = 1)	111

JMP addr (Jump)
(PC) ← (byte 3)(byte 2)
Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

<table>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td colspan="8">low-order addr</td></tr>
<tr><td colspan="8">high-order addr</td></tr>
</table>

Cycles:	3
States:	10
Addressing:	immediate
Flags:	none

J condition addr (Conditional jump)
If (CCC),
(PC) ← (byte 3)(byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

<table>
<tr><td>1</td><td>1</td><td>C</td><td>C</td><td>C</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">low-order addr</td></tr>
<tr><td colspan="8">high-order addr</td></tr>
</table>

Cycles:	2/3
States:	7/10
Addressing:	immediate
Flags:	none

CALL addr (Call)
((SP – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← (byte 3)(byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified by byte 3 and byte 2 of the current instruction.

<table>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td colspan="8">low-order addr</td></tr>
<tr><td colspan="8">high-order addr</td></tr>
</table>

Cycles:	5
States:	18
Addressing:	immediate/ reg. indirect
Flags:	none

C condition addr (Condition call)
If (CCC)
((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
((SP) ← (SP) – 2
(PC) ← (byte 3)(byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

<table>
<tr><td>1</td><td>1</td><td>C</td><td>C</td><td>C</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td colspan="8">low-order addr</td></tr>
<tr><td colspan="8">high-order addr</td></tr>
</table>

Cycles:	2/5
States:	9/18
Addressing:	immediate/ reg. indirect
Flags:	none

RET (Return)
(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of the register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

1	1	0	0	1	0	0	1

Cycles:	3
States:	10
Addressing:	reg. indirect
Flags:	none

R condition (Conditional return)
If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) = 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	0	0	0

Cycles:	1/3
States:	6/12
Addressing:	reg. indirect
Flags:	none

RST n (Restart)

$((SP)-1) \leftarrow (PCH)$
$((SP)-2) \leftarrow (PCL)$
$(SP) \leftarrow (SP)-2$
$(PC) \leftarrow 8 \star (NNN)$

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

1	1	N	N	N	1	1	1

Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Program Counter After Restart

PCHL (Jump H and L indirect-move H and L to PC)

$(PCH) \leftarrow (H)$
$(PCL) \leftarrow (L)$

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

1	1	1	0	1	0	0	1

Cycles: 1
States: 6
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

PUSH rp (Push)

$((SP)-1) \leftarrow (rh)$
$((SP)-2) \leftarrow (rl)$
$((SP) - (SP) \leftarrow 2$

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register or register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**

1	1	R	P	0	1	0	1

Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

$((SP)-1) \leftarrow (A)$
$((SP)-2)_0 \leftarrow (CY), ((SP)-2)_1 \leftarrow X$
$((SP)-2)_2 \leftarrow (P), ((SP)-2)_3 \leftarrow X$
$((SP)-2)_4 \leftarrow (AC), ((SP)-2)_5 \leftarrow X$
$((SP-2)_6 \leftarrow (Z), ((SP)-2)_7 \leftarrow S$
$(SP) \leftarrow (SP) - 2$ X: Undefined.

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.

1	1	1	1	0	1	0	1

Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

FLAG WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

X: undefined

POP rp (Pop)

$(rl) \leftarrow ((SP))$
$(rh) \leftarrow ((SP)+1)$
$(SP) \leftarrow (SP)+2$

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**

1	1	R	P	0	0	0	1

Cycles: 3
States: 10
Addressing: reg.indirect
Flags: none

POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
$(A) \leftarrow ((SP) + 1)$
$(SP) \leftarrow (SP) + 2$

The content of the memory location whose address is specified by the content of register SP is used to retore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

1	1	1	1	0	0	0	1

Cycles:	**3**
States:	**10**
Addressing:	**reg. indirect**
Flags:	**Z,S,P,CY,AC**

XTHL (Exchange stack top with H and L)

$(L) \leftrightarrow ((SP))$
$(H) \leftrightarrow ((SP) + 1)$

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

1	1	1	0	0	0	1	1

Cycles:	**5**
States:	**16**
Addressing:	**reg. indirect**
Flags:	**none**

SPHL (Move HL to SP)

$(SP) \leftarrow (H)(L)$

The contents of registers H and L (16 bits) are moved to register SP.

1	1	1	1	1	0	0	1

Cycles:	**1**
States:	**6**
Addressing:	**register**
Flags:	**none**

IN port (Input)

$(A) \leftarrow (data)$

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

1	1	0	1	1	0	1	1
port							

Cycles:	**3**
States:	**10**
Addressing:	**direct**
Flags:	**none**

OUT port (Output)

$(data) \leftarrow (A)$

The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.

1	1	0	1	0	0	1	1
port							

Cycles:	**3**
States:	**10**
Addressing:	**direct**
Flags:	**none**

EI (Enable interrupts)

The interrupt system is enabled **following the execution of the next instruction. Interrupts are not recognized during the EI instruction.**

NOTE: Placing an EI instruction on the bus in response to $\overline{INTA}$ during an INA cycle is prohibited.

1	1	1	1	1	0	1	1

Cycles:	1
States:	4
Flags:	none

DI (Disable interrupts)

The interrupt system is disabled **immediately following the execution of the DI instruction. Interrupts are not recognized during the DI instruction.**

1	1	1	1	0	0	1	1

Cycles:	1
States:	4
Flags:	none

NOTE: Placing a DI instruction on the bus in response to $\overline{INTA}$ during an INA cycle is prohibited.

HLT (Halt)
The processor is stopped. A second ALE is generated during the executive of HLT to strobe out the Halt cycle status information.

0	1	1	1	0	1	1	0

Cycles: 1+
States: 5
Flags: none

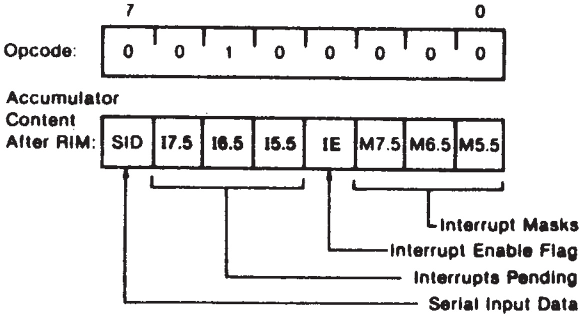
NOP (No op)
No operation is performed. The registers and flags are unaffected.

0	0	0	0	0	0	0	0

Cycles: 1
States: 4
Flags: none

RIM (Read Interrupt Masks)
The RIM instruction loads data into the accumulator relating to interrupts and the serial input. This data contains the following information:

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware interrupts (1 = mask disabled.
- Current interrupt enable flag status (1 = interrupts enabled) except immediately following a TRAP interrupt. (See below).
- Hardware interrupts pending (i.e., signal received but not yet serviced), on the RST 5.5, 6.5, and 7.5 lines.
- Serial input data.



Cycles: 1
States: 4
Flags: none

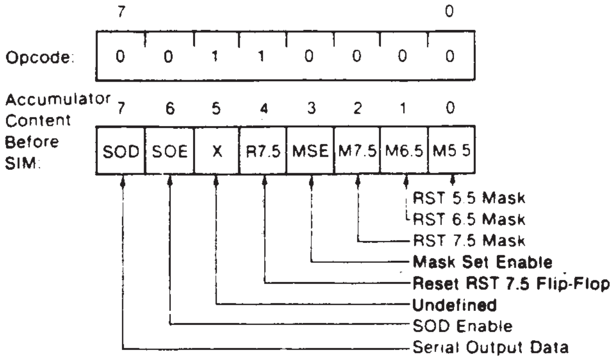
Immediately following a TRAP interrupt, the RIM instruction must be executed as a part of the service routine if you need to retrieve current interrupt status later. Bit 3 of the accumulator is (in this special case only) loaded with the interrupt enable (IE) flag status that existed prior to the TRAP interrupt. Following an RST 5.5, 6.5, 7.5, or INTR interrupt, the interrupt flag flip-flop reflects the current interrupt enable status. Bit 6 of the accumulator (I7.5) is loaded with the status of the RST 7.5 flip-flop, which is always set (edge-triggered) by an input on the RST 7.5 input line, even when that interrupt has been previously masked. (See SIM instruction.)

SIM (Set Interrupt, Masks)
The execution of the SIM instruction uses the contents of the accumulator (which must be previously loaded) to perform the following *functions:*

- Program the interrupt mask for the RST 5.5, 6.5, and 7.5 hardware interrupts.
- Reset the edge-triggered RST 7.5 input latch.
- Load the SOD output latch.

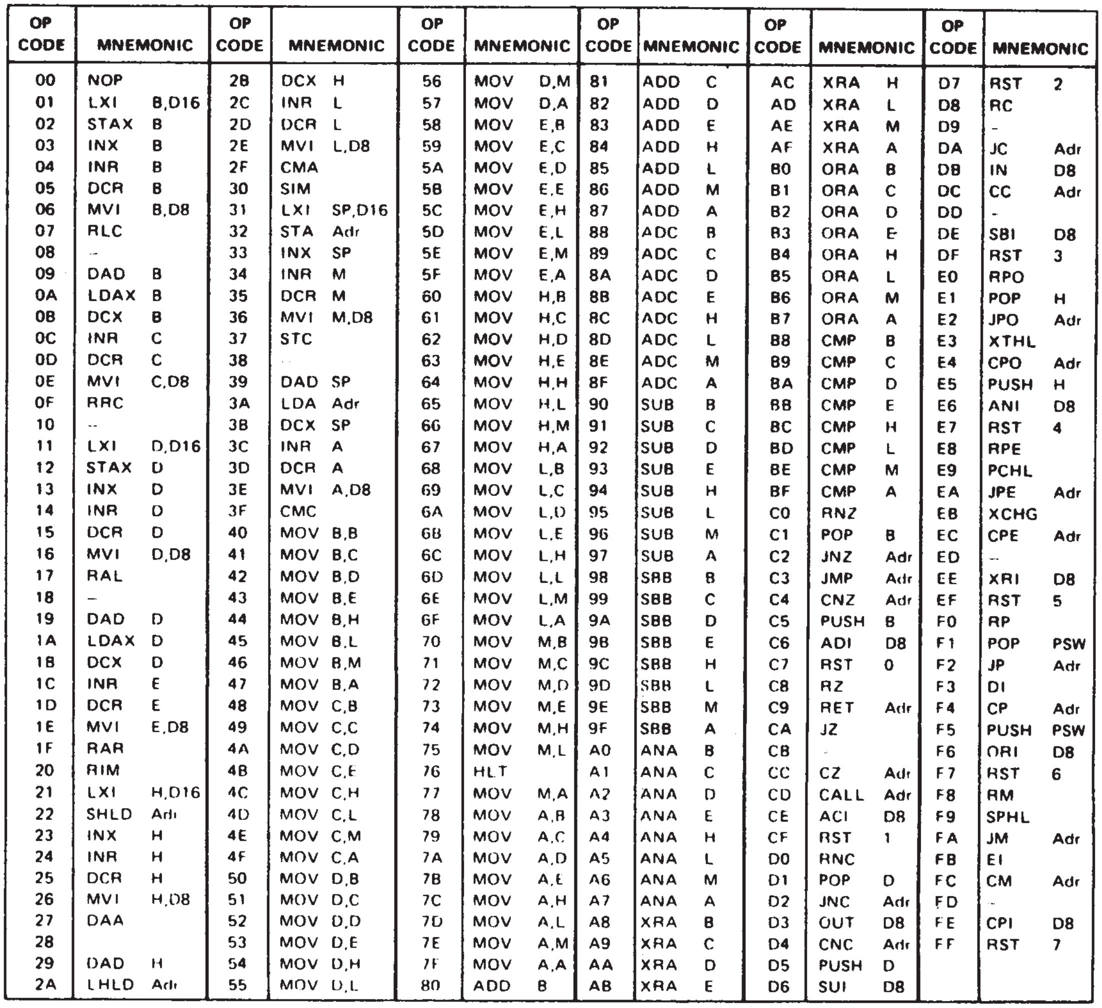
To program the interrupt masks, first set accumulator bit 3 to 1 and set to 1 any bits 0, 1, and 2, which disable interrupts RST 5.5, 6.5, and 7.5, respectively. Then do a SIM instruction. If accumulator bit 3 is 0 when the SIM instruction is executed, the interrupt mask register will not change. If accumulator bit 4 is 1 when the SIM instruction is executed, the RST 7.5 latch is then reset. RST 7.5 is distinguished by the fact that its latch is always set by a rising edge on the RST 7.5 input pin, even if the jump to service routine is inhibited by masking. This latch remains high until cleared by a $\overline{\text{RESET IN}}$, by a SIM instruction with accumulator bit 4 high, or by an internal processor acknowledge to an RST 7.5 interrupt subsequent to the removal of the mask (by a SIM instruction). The $\overline{\text{RESET IN}}$ signal always sets all three RST mask bits.

If accumulator bit 6 is at the 1 level when the SIM instruction is executed, the state of accumulator bit 7 is loaded into the SOD latch and thus becomes available for interface to an external device. The SOD latch is unaffected by the SIM instruction if bit 6 is 0. SOD is always reset by the $\overline{\text{RESET IN}}$ signal.



Cycles: 1
States: 4
Flags: none

Table 8-1 8085A CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE



OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	-
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	-
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	--	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	..	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	--	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	--
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	–	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	-	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	-
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28		53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity

D16 constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

Adr 16-bit address.

Table 8-2 - 8085A INSTRUCTION SET SUMMARY BY FUNCTIONAL GROUPING

		Instruction Code (1)							
Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MOVE, LOAD, AND STORE									
MOVr1 r2	Move register to register	0	1	D	D	D	S	S	S
MOV M.r	Move register to memory	0	1	1	1	0	S	S	S
MOV r.M	Move memory to register	0	1	D	D	D	1	1	0
MVI r	Move immediate to register	0	0	D	D	D	1	1	0
MVI M	Move immediate to memory	0	0	1	1	0	1	1	0
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1
STAX B	Store A indirect	0	0	0	0	0	0	1	0
STAX D	Store A indirect	0	0	0	1	0	0	1	0
LDAX B	Load A indirect	0	0	0	0	1	0	1	0
LDAX D	Load A indirect	0	0	0	1	1	0	1	0
STA	Store A direct	0	0	1	1	0	0	1	0
LDA	Load A direct	0	0	1	1	1	0	1	0
SHLD	Store H & L direct	0	0	1	0	0	0	1	0
LHLD	Load H & L direct	0	0	1	0	1	0	1	0
XCHG	Exchange D & E, H & L registers	1	1	1	0	1	0	1	1

8085A INSTRUCTION SET SUMMARY (Cont'd)

		Instruction Code (1)							
Mnemonic	**Description**	**D_7**	**D_6**	**D_5**	**D_4**	**D_3**	**D_2**	**D_1**	**D_0**
STACK OPS									
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1
POP B	POP register Pair B & C off stack	1	1	0	0	0	0	0	1
POP D	POP register Pair D & E off stack	1	1	0	1	0	0	0	0
POP H	POP register Pair H & L off stack	1	1	1	0	0	0	0	1
POP PSW	POP A and Flags off stack	1	1	1	1	0	0	0	1
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1
JUMP									
JMP	Jump unconditional	1	1	0	0	0	0	1	1
JC	Jump on carry	1	1	0	1	1	0	1	0
JNC	Jump on no carry	1	1	0	1	0	0	1	0
JZ	Jump on zero	1	1	0	0	1	0	1	0
JNZ	Jump on no zero	1	1	0	0	0	0	1	0
JP	Jump on positive	1	1	1	1	0	0	1	0
JM	Jump on minus	1	1	1	1	1	0	1	0
JPE	Jump on parity even	1	1	1	0	1	0	1	0
JPO	Jump on parity odd	1	1	1	0	0	0	1	0
PCHL	H & L to program counter	1	1	1	0	1	0	0	1
CALL									
CALL	Call unconditional	1	1	0	0	1	1	0	1
CC	Call on carry	1	1	0	1	1	1	0	0
CNC	Call on no carry	1	1	0	1	0	1	0	0
CZ	Call on zero	1	1	0	0	1	1	0	0
CNZ	Call on no zero	1	1	0	0	0	1	0	0
CP	Call on positive	1	1	1	1	0	1	0	0
CM	Call on minus	1	1	1	1	1	1	0	0
CPE	Call on parity even	1	1	1	0	1	1	0	0
CPO	Call on parity odd	1	1	1	0	0	1	0	0
RETURN									
RET	Return	1	1	0	0	1	0	0	1
RC	Return on carry	1	1	0	1	1	0	0	0
RNC	Return on no carry	1	1	0	1	0	0	0	0
RZ	Return on zero	1	1	0	0	1	0	0	0
RNZ	Return on no zero	1	1	0	0	0	0	0	0
RP	Return on positive	1	1	1	1	0	0	0	0
RM	Return on minus	1	1	1	1	1	0	0	0
RPE	Return on parity even	1	1	1	0	1	0	0	0
RPO	Return on parity odd	1	1	1	0	0	0	0	0
RESTART									
RST	Restart	1	1	A	A	A	1	1	1
INPUT/OUTPUT									
IN	Input	1	1	0	1	1	0	1	1
OUT	Output	1	1	0	1	0	0	1	1
INCREMENT AND DECREMENT									
INR r	Increment register	0	0	D	D	D	1	0	0
DCR r	Decrement register	0	0	D	D	D	1	0	1
INR M	Increment memory	0	0	1	1	0	1	0	0
DCR M	Decrement memory	0	0	1	1	0	1	0	1
INX B	Increment B & C registers	0	0	0	0	0	0	1	1
INX D	Increment D & E registers	0	0	0	1	0	0	1	1
INX H	Increment H & L registers	0	0	1	0	0	0	1	1

8085A INSTRUCTION SET SUMMARY (Cont'd)

		Instruction Code (1)							
Mnemonic	**Description**	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INCREMENT AND DECREMENT (cont'd)									
DCX B	Decrement B & C	0	0	0	0	1	0	1	1
DCX D	Decrement D & E	0	0	0	1	1	0	1	1
DCX H	Decrement H & L	0	0	1	0	1	0	1	1
ADD									
ADD r	Add register to A	1	0	0	0	0	S	S	S
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S
ADD M	Add memory to A	1	0	0	0	0	1	1	0
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0
ADI	Add immediate to A	1	1	0	0	0	1	1	0
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1
SUBTRACT									
SUB r	Subtract register from A	1	0	0	1	0	S	S	S
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0
LOGICAL									
ANA r	And register with A	1	0	1	0	0	S	S	S
XRA r	Exclusive OR register with A	1	0	1	0	1	S	S	S
ORA r	OR register with A	1	0	1	1	0	S	S	S
CMP r	Compare register with A	1	0	1	1	1	S	S	S
ANA M	And memory with A	1	0	1	0	0	1	1	0
XRA M	Exclusive OR memory with A	1	0	1	0	1	1	1	0
ORA M	OR memory with A	1	0	1	1	0	1	1	0
CMP M	Compare memory with A	1	0	1	1	1	1	1	0
ANI	And immediate with A	1	1	1	0	0	1	1	0
XRI	Exclusive OR immediate with A	1	1	1	0	1	1	1	0
ORI	OR immediate with A	1	1	1	1	0	1	1	0
CPI	Compare immediate with A	1	1	1	1	1	1	1	0
ROTATE									
RLC	Rotate A left	0	0	0	0	0	1	1	1
RRC	Rotate A right	0	0	0	0	1	1	1	1
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1
SPECIALS									
CMA	Complement A	0	0	1	0	1	1	1	1
STC	Set carry	0	0	1	1	0	1	1	1
CMC	Complement carry	0	0	1	1	1	1	1	1
DAA	Decimal adjust A	0	0	1	0	0	1	1	1
CONTROL									
EI	Enable interrupts	1	1	1	1	1	0	1	1
DI	Disable interrupt	1	1	1	1	0	0	1	1
NOP	No-operation	0	0	0	0	0	0	0	0
HLT	Halt	0	1	1	1	0	1	1	0
NEW 8085A INSTRUCTIONS									
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0

NOTES: 1 - DDS or SSS B 000, C 001, D 010, E011, H 100, L 101, Memory 110, A 111

2 - Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

LESSON 9

MONITOR PROGRAM

The Monitor program resides in ROM locations 0000 thru 0003 and ROM locations 0400 thru 012D. RAM locations 80DC thru 80FF are also used. If the RESET switch is down, the MM-8000 comes up running the Monitor program whenever the power is turned on. With the Monitor program running, the keyboard (discussed in Lesson 12) is used to enter data or new programs into memory. Any byte in memory may also be sent to the display. By using the GO button, control may be transferred from the Monitor to other programs.

PARAMETER	RAM ADDRESS
Mode L	80FA
Mode H	80FB
Display Data-DDA	80FC
Display Address Low-DAL	80FD
Display Address High-DAH	80FE

Table 9-1

MONITOR PROGRAM MODE	MODE H	MODE L
Data-DA	80	FC
Address Low-AL	80	FD
Address High-AH	80	FE

Table 9-2

The Monitor program operates by controlling the parameters shown in Table 9-1. There are three modes of operation, Data (DA), Address Low (AL), and Address High (AH). The active mode is determined by the code in the MODE L byte. As shown in Table 9-2, the MODE L byte is set to the low order byte of the address of DDA (FC) for DA mode, DAL (FD) for AL mode, and DAH (FE) for AH mode. The MODE H byte always contains 80 which is the high order address byte of DDA, DAL and DAH. The mode is set by pressing the corresponding keyboard button. Each mode operates as follows.

DATA MODE

In DA mode the MODE H and MODE L bytes contain the address of the DDA byte 80FC. The two hexadecimal digits of DDA are displayed in Display 1 and Display 2. No decimal point is lit indicating that the DDA byte is being displayed. When one of the keyboard data keys (0 thru F) is pressed, the data is entered into the low order digit of DDA and displayed in Display 2. The digit that was in the low order position is shifted to the high order position and displayed in Display 1.

ADDRESS LOW MODE

In AL mode the mode H and mode L bytes contain the address of the DAL byte 80FD. Data from the keyboard is now entered into DAL in the same manner as into DDA. The display is the same except that the decimal point of Display 2 is lit indicating that the DAL byte is being displayed.

ADDRESS HIGH MODE

In AH mode the MODE H and MODE L bytes contain the address of the DAH byte 80FE. Operation in the AH mode corresponds to AL mode except that the decimal point of Display 1 is lit indicating that the DAH byte is being displayed.

When the keyboard Store (ST) button is pressed, the data in byte DDA is stored at the address specified by DAH and DAL. This address is then incremented by one and the Monitor is put in DA mode displaying the data byte just stored.

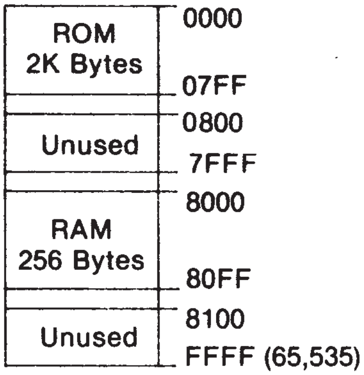
When the keyboard button Read (RD) is pressed, the contents of the address specified by DAH and DAL is read and placed in the DDA byte. The address (DAH and DAL) is then incremented by one and the Monitor is put in the DA mode displaying the byte just read.

When the keyboard GO button is pressed, control is transferred to the instruction whose address is specified by DAH and DAL.

The keyboard X1 and X2 buttons are reserved for future use. At present, if the X1 or X2 buttons are pressed the Monitor program is restarted.

MEMORY ALLOCATION

During each machine cycle,the 8085 sends out 16 bits of address which give 2^{16} or 65,536 directly addressable memory locations. The MM-8000 system contains 2K bytes of ROM and 256 bytes of RAM. There are therefore, many unused locations. To assign ROM and RAM to specific areas in the 65K memory field, address line A15 is used as a chip select. This is called linear selection. When the RESET, ENROM and ENRAM switches are down, address line A15 drives the $\overline{\mathrm{CE}}$ input of the ROM and the CE input of the RAM. The ROM is thus selected for addresses below 8000 and the RAM is selected for address 8000 and above. Since the RAM contains only 256 bytes, only address lines A0 thru A7 are needed. Lines A8 thru A14 are not decoded and may be either 1 or 0. For convenience,we will define A8 thru A14 as zeros. The RAM thus occupies address space 8000 thru 80FF (see figure 9-1). Similarly the ROM occupies address space 0000 thru 07FF.



Memory Map

Figure 9-1

INPUT-OUTPUT

The MM-8000 system uses an I/O mapped I/O, that is the IO/$\overline{\mathrm{M}}$ line rather than a memory address line is used to distinguish between I/O cycles and memory cycles. During Input and Output instructions,the second byte (port byte) of the instruction is sent out on the Address-Data bus and repeated on address lines A8 - A15 (see figure 9-2). To enable the 8156, bit 7 of the port byte (repeated on A15) must be a 1. As shown in lesson 4, bit 7 does not affect the port selected.

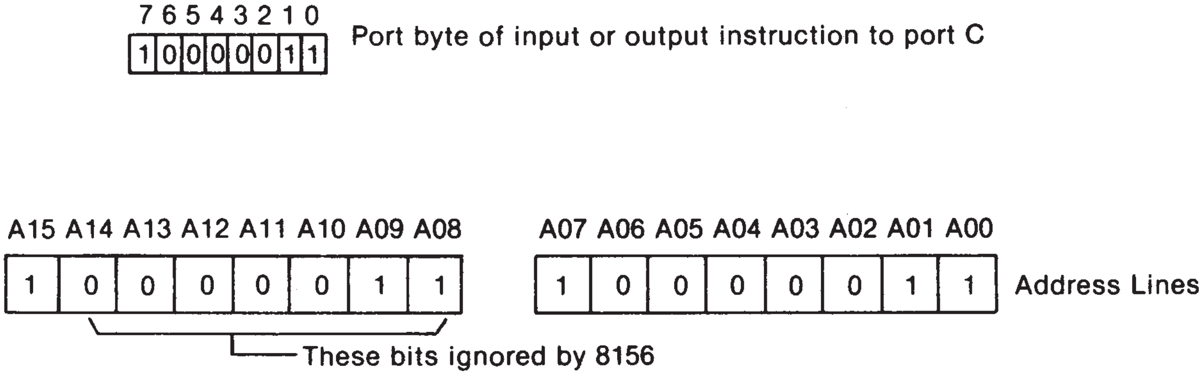

7 6 5 4 3 2 1 0
1 0 0 0 0 0 1 1
Port byte of input or output instruction to port C
A15 A14 A13 A12 A11 A10 A09 A08
1 0 0 0 0 0 1 1
A07 A06 A05 A04 A03 A02 A01 A00
1 0 0 0 0 0 1 1
Address Lines
These bits ignored by 8156

Figure 9-2

EXPANSION

To provide the MM-8000 system with memory and I/O expansion capability the necessary signal lines are brought to a 44 pin edgeboard connector (see table 9-3) and two 16 pin IC connectors (see table 9-4).

PIN No.	SIGNAL NAME	SOURCE or DESTINATION
1	INTR	IC1-10
3	RST 5.5	IC1-9
5	RST 6.5	IC1-8
7	RST 7.5	IC1-7
9	TRAP	IC1-6
11	A8	IC1-21
13	A9	IC1-22
15	A10	IC1-23
17	A11	IC1-24
19	A12	IC1-25
21	A13	IC1-26
23	A14	IC1-27
25	A15	IC1-28
27	KEY	
29	GROUND	
31	ALE	R7
33	$\overline{\text{WD}}$	IC1-31
35	$\overline{\text{RD}}$	IC1-32
37	CE RAM	IC2-8
39	IO/$\overline{\text{M}}$	IC1-34
41	$\overline{\text{TIMER OUT}}$	IC2-6
43	TIMER IN	IC2-3

PIN No.	SIGNAL NAME	SOURCE or DESTINATION
2	V UNREG.	
4	GROUND	
6	$\overline{\text{CE}}$ ROM	IC3-18
8	$\overline{\text{INTA}}$	IC1-11
10	SID	IC1-5
12	SOD	IC1-4
14	RST OUT	IC1-3
16	S0	IC1-29
18	S1	IC1-33
20	READY	IC1-35
22	CLK OUT	IC1-37
24	HLDA	IC1-38
26	HOLD	IC1-39
28	KEY	
30	AD0	IC1-12
32	AD1	IC1-13
34	AD2	IC1-14
36	AD3	IC1-15
38	AD4	IC1-16
40	AD5	IC1-17
42	AD6	IC1-18
44	AD7	IC1-19

Table 9-3

CONNECTOR S1		
PIN No.	SIGNAL NAME	SOURCE / DESTINATION
1	OPEN	
2	OPEN	
3	OPEN	
4	OPEN	
5	OPEN	
6	OPEN	
7	OPEN	
8	OPEN	
9	+5V	
10	GROUND	
11	PORT C-5	IC2-5
12	PORT C-4	IC2-2
13	PORT C-3	IC2-1
14	PORT C-2	IC2-39
15	PORT C-1	IC2-38
16	PORT C-0	IC2-37

CONNECTOR S2		
PIN No.	SIGNAL NAME	SOURCE / DESTINATION
1	PORT A-7	IC2-28
2	PORT A-6	IC2-27
3	PORT A-5	IC2-26
4	PORT A-4	IC2-25
5	PORT A-3	IC2-24
6	PORT A-2	IC2-23
7	PORT A-1	IC2-22
8	PORT A-0	IC2-21
9	PORT B-0	IC2-29
10	PORT B-1	IC2-30
11	PORT B-2	IC2-31
12	PORT B-3	IC2-32
13	PORT B-4	IC2-33
14	PORT B-5	IC2-34
15	PORT B-6	IC2-35
16	PORT B-7	IC2-36

Table 9-4

LESSON 10

PROGRAM 1

Program 1 performs the initialization section of the Monitor program. The 36 ROM locations, 010A thru 012D, are copied to RAM locations 80DC thru 80FF. These locations contain a delay routine of approximately 10 milliseconds and will be used later when writing to ROM. When running this routine in RAM, only RAM address locations are accessed. In lesson 6 it was shown that writing to ROM required a 10ms delay. The ROM can not be accessed during this delay time. Therefore, after writing to ROM, the Monitor program runs the 10ms delay routine in RAM before returning to ROM. The initialization section of the Monitor program also configures Port A as an input port and Ports B and C as output ports.

Most of the address locations below 0040 are reserved as Trap and Restart locations. The Monitor program uses locations 0040 thru 005F for look up tables. Since the 8085 microprocessor always starts running at location 0000, a Jump instruction at 0000 immediately moves the program to location 0060 to start the copy routine.

A program listing of programs 1 thru 4 may be found in Appendix 1. All data and addresses in the program listing are in hexadecimal unless stated otherwise. Refer to Table 1-1 for a binary to hexadecimal conversion chart. Refer to figure 10-1 for a flow chart and to Appendix 2 for a detailed description of Program 1.

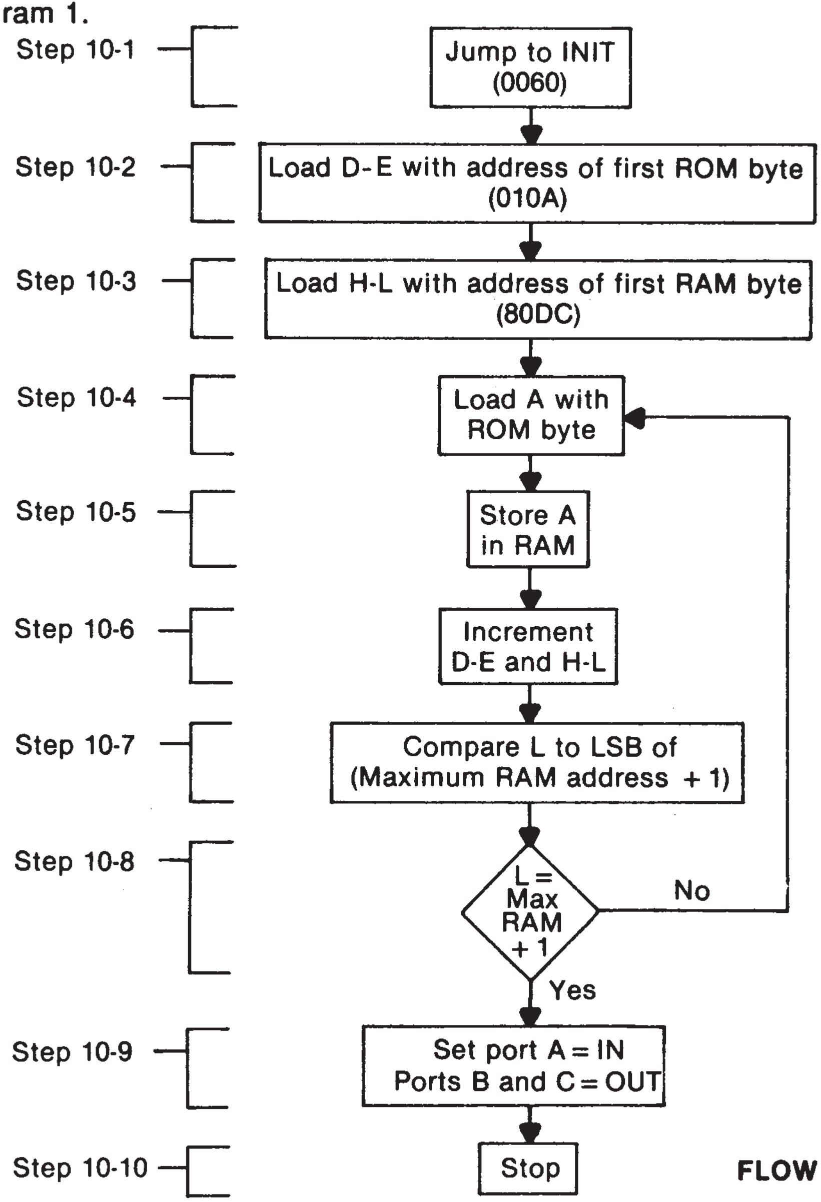


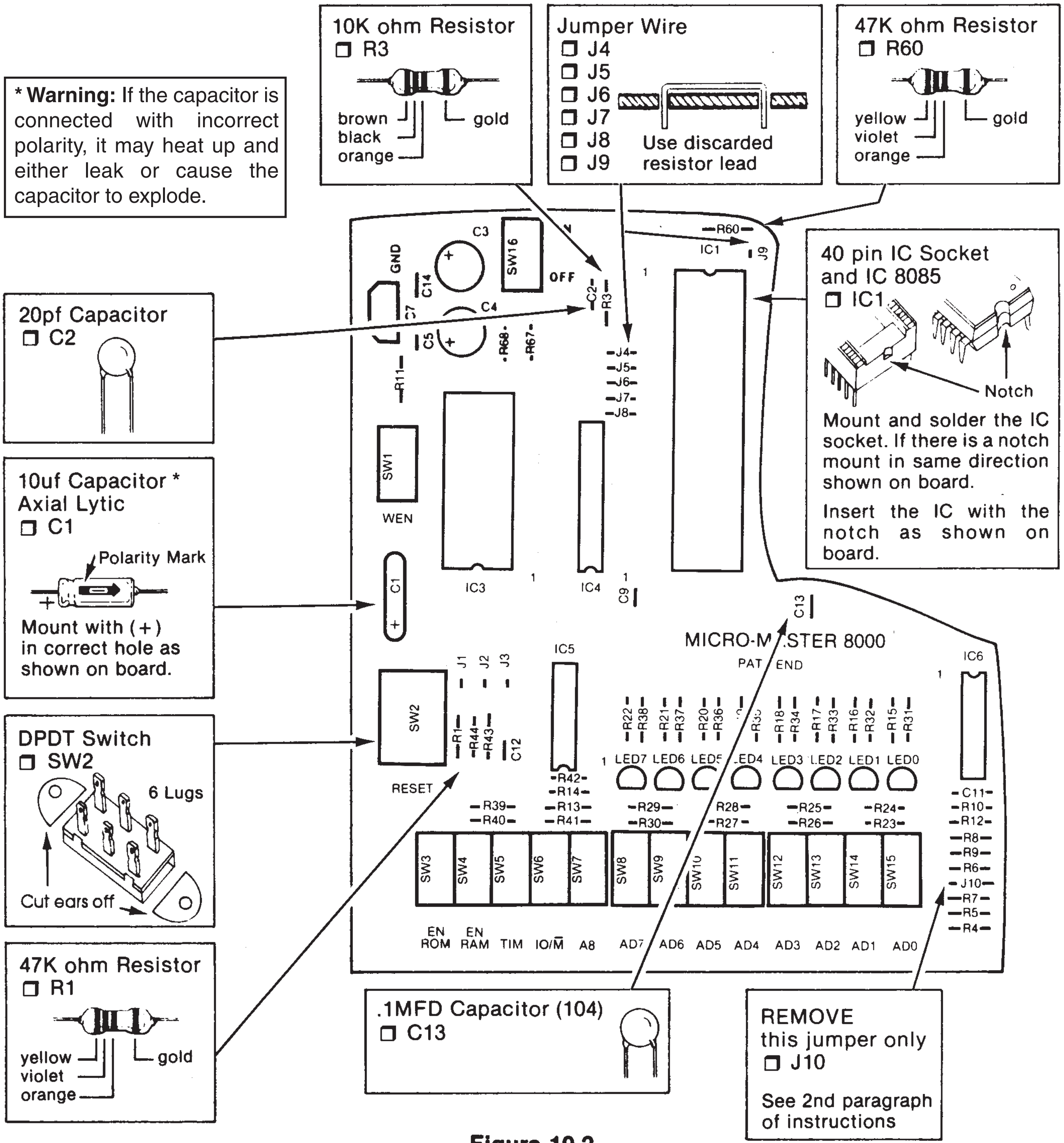
Figure 10-1

FLOW CHART PROGRAM 1

ASSEMBLY INSTRUCTIONS

Unplug connector S3 to remove all power from the PC board. Identify and install the following parts as shown in figure 10-2. After soldering each part place a check in the box provided.

To remove jumper J10, reset your desoldering pump and hold in one hand. With your soldering iron in the other hand,heat one pad of J10, immediately put the tip of the pump in the melted solder and trigger the desoldering pump. Now remove the solder from the other pad. The holes should be clean of solder and the jumper wire free from the board.



10K ohm Resistor
□ R3

Jumper Wire
□ J4
□ J5
□ J6
□ J7
□ J8
□ J9

Use discarded resistor lead

47K ohm Resistor
□ R60

* **Warning:** If the capacitor is connected with incorrect polarity, it may heat up and either leak or cause the capacitor to explode.

20pf Capacitor
□ C2

40 pin IC Socket and IC 8085
□ IC1

Mount and solder the IC socket. If there is a notch mount in same direction shown on board.

Insert the IC with the notch as shown on board.

10uf Capacitor *
Axial Lytic
□ C1

Mount with (+) in correct hole as shown on board.

DPDT Switch
□ SW2

Cut ears off

47K ohm Resistor
□ R1

.1MFD Capacitor (104)
□ C13

REMOVE
this jumper only
□ J10

See 2nd paragraph of instructions

Figure 10-2

If unit will not function properly see trouble shooting guide on page VII.

CIRCUIT DESCRIPTION

Refer to figure 10-3 for a simplified schematic of the parts added in lesson 10. Addition of the 8085 integrated circuit allows microprocessor control of the Data Bus AD0 thru AD7, the Address Bus A8 thru A15, and the Control Buses consisting of RESET OUT, IO/$\overline{M}$, $\overline{RD}$, $\overline{WR}$, and ALE. Jumper J10 is removed to allow the 8085 to drive the ALE Bus through resistor R7. When the 8085 is RESET, the ALE output is held at a low voltage allowing the ALE button to control the bus as in previous lessons.

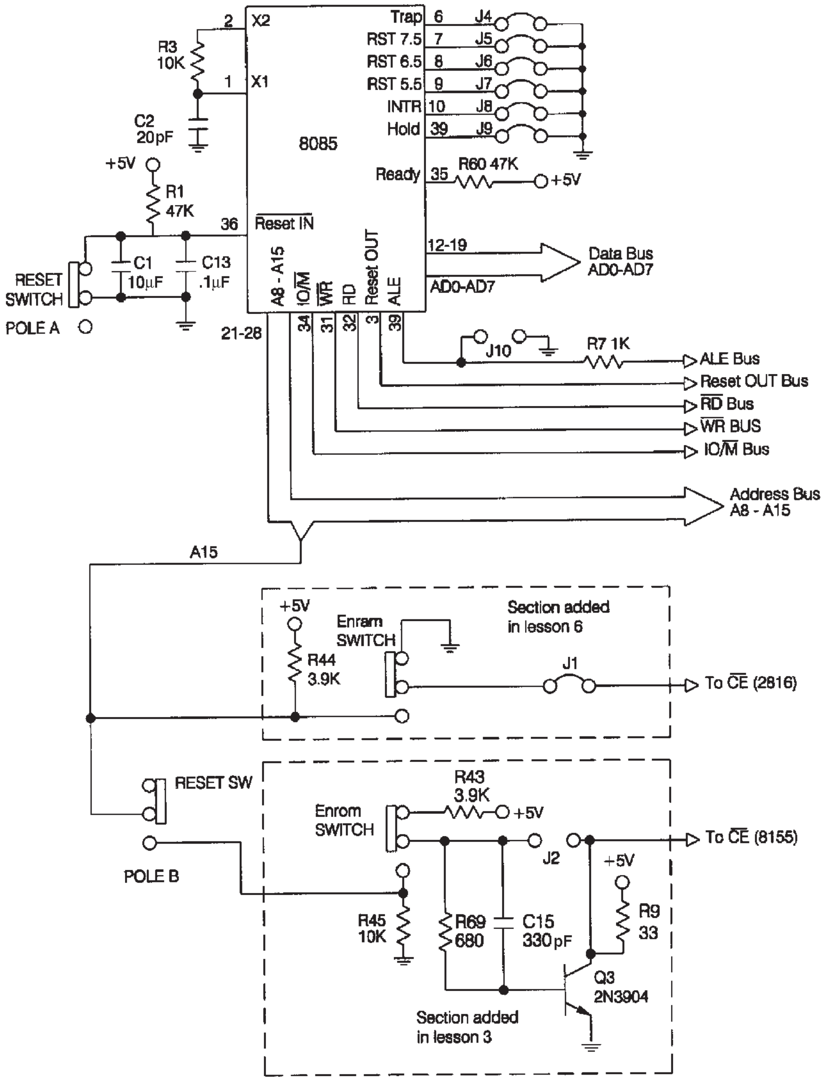


Figure 10-3

To avoid any unwanted inputs, jumpers J4 thru J9 bias the TRAP, RST7.5, RST6.5, RST5.5, INTR and HOLD lines to ground,and resistor R60 biases the READY line to + 5 volts. Resistor R3 and capacitor C2 are connected to the X1 and X2 inputs of the 8085 and set the clock frequency to approximately 1.5Mhz.

When the RESET switch is up, the $\overline{\text{RESET IN}}$ input of the 8085 is tied to ground (0) through pole A of the switch and the 8085 is in the Reset State. When the switch is put down a high voltage (1) is applied to the $\overline{\text{RESET IN}}$ input through R1 and the 8085 starts running at address 0000. If the RESET switch is down when the power is turned ON, the application of the high voltage to the $\overline{\text{RESET IN}}$ is delayed by capacitors C1 and C13 to allow the system to stabilize before starting the 8085.

When the RESET switch is up and the 8085 is in the reset state, the address lines, including A15, are in the high impedance state which effectively disconnects them from the Address Bus. In this state the ENRAM and ENROM switches operate as described in Lessons 3 and 6. Before the RESET switch is put down,the ENRAM and ENROM must be in the down (off) position for proper operation of the MM-8000 system. When the RESET switch is put down,the address lines go to the low impedance state. Address line A15 drives the $\overline{CE}$ input of the 2816 through the ENROM switch and J1. Address line A15 also drives the CE input of the 8156 through pole B of the RESET switch, the ENRAM switch, and J2. As on the Data Bus lines the 3.9K pull up resistor R44 and the 10K pull down resistor R45 are overridden by the low impedance A15 drive circuit in the 8085.

PROCEDURE

1. Put the reset switch up and all other switches down.
2. Connect the power supply and turn power ON.
3. Put the ENROM switch up to enable the ROM.
4. Put the WEN switch up to enable writing to the ROM.
5. Set the data switches to 0000 0000 and press the ALE button to latch this address into the 74HCT573.
6. Set the data switches to 1100 0011 (C3 in HEX) and push the WR button to load the first instruction of the program into ROM.
7. Repeat steps 5 and 6 changing the address and data to load the remainder of program 1 as listed in Appendix 1 address locations 0001, 0002 and 0060 thru 0074.
8. Put the A8 switch up to access memory locations 0100 thru 01FF.
9. Repeat steps 5 and 6 changing the address and data to load the program codes listed in Appendix 1 at locations 010A thru 012D. This is a section of the Monitor program which will be used in a later program.
10. Put the WEN switch down to disable further writing to ROM.
11. Put the A8 switch down to return to addresses 0000 thru 00FF.
12. Set the data switches to 0000 0000 and press the ALE button to latch the address into the 74HCT573. Push the RD button to verify the data stored in that address.
13. Repeat step 12 (changing address) to verify that the data stored at locations 0001, 0002 and 0060 thru 0074 is stored correctly.
14. Put the A8 switch up to return to addresses 0100 thru 01FF.
15. Repeat step 12 (changing address) to verify that the data stored at locations 010A thru 012D is stored correctly.
16. Put the ENROM switch down to disable the ROM.
17. Put the A8 switch down to return to addresses 0000 thru 00FF.
18. Put the RESET switch down. The MM-800C now performs the copy operation, sets Port A to input and Ports B and C to outputs and then stops at location 0074. As in lesson 4, the port outputs are initially zero. This lights all segments including the decimal point of Display 2.
19. Put the RESET switch up to return to manual operation. Note that the RESET line to the 8156 returns the IO Ports to inputs which turns off Display 2.
20. Put the ENRAM switch up to enable RAM.
21. Set the data switches to 1101 1100 (DC in Hex) and press the ALE button to latch the address into the 8155. Then push the RD button to verify that the first byte was copied correctly from ROM.
22. Repeat step 21 changing the address from DD thru FF to verify that all 36 bytes were copied correctly.
23. Put the ENRAM switch down to disable the RAM.
24. Turn OFF power.

LESSON 11

For Program 2 the Delay and Display sections of the Monitor program are added to Program 1. Figure 11-1 shows a flow chart of Program 2. To link the program sections,the HLT instruction at location 0074 in Program 1 is replaced by the JMP instruction at location 0074 in Program 2 (see appendix 1).

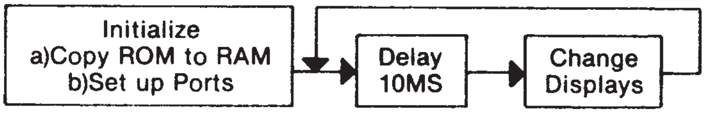
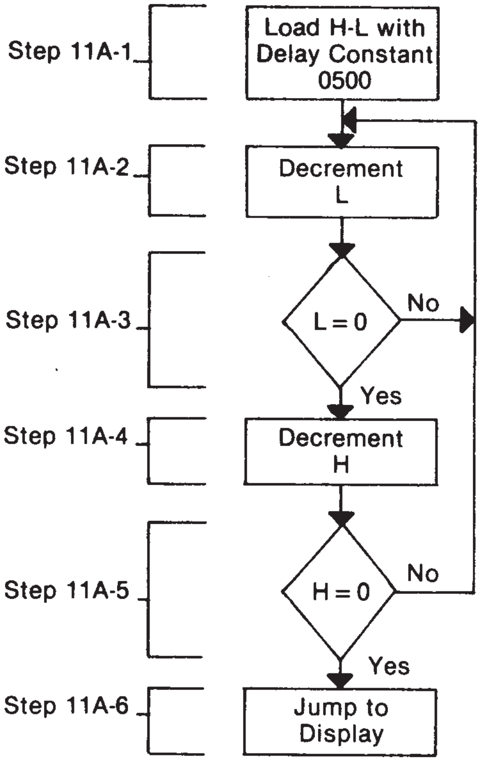


Figure 11-1

DELAY SECTION

The Delay section resides in RAM locations 80EC thru 80F9. This is part of the 36 bytes copied from ROM to RAM in Program 1. A flow chart of the Delay section is shown in figure 11-2. The delay is accomplished by first loading the H-L register pair with a delay constant. Next,the L register is repeatedly decremented and checked for zero (steps 11A-2 and 11A-3). When the L register is zero,the H register is decremented once and the L register is again decremented to zero. When the H register reaches zero,the delay is complete.

Figure 11-2



DISPLAY SECTION

The Display section resides in ROM locations 0077 thru 00A4 . RAM locations 80FA thru 80FE are used to store the parameters MODE L, MODE H, DDA, DAL and DAH. The initial value of these parameters, shown in table 11-1, are copied to RAM from ROM at the start of the program. As was shown in lesson 4,the data output on Port B goes to both Display 1 (left display) and Display 2 (right display). Bit 0 of Port C controls which display utilizes the data. Only one display is on at a time. The Monitor program displays the low order digit, bits 0 thru 3 of a particular byte in the right display and the high order digit, bits 4 thru 7 in the left display. The Display section of the Monitor inputs Port C and senses which digit is currently being displayed. The data in Port B is changed to display the other digit and bit 0 of Port C is complemented to light the other display. As shown in figure 11-1, Program 2 then delays and repeats the display switch. If this is done fast enough,it will appear like both displays are lit simultaneously.

Table 11-1

ROM ADDRESS	LOCATION NAME	RAM ADDRESS	INITIAL VALUE FROM ROM
0128	MODE L	80FA	FC
0129	MODE H	80FB	80
012A	DDA	80FC	00
012B	DAL	80FD	00
012C	DAH	80FE	00
012D	ENKP	80FF	FF

Initial value of Monitor Program Parameters

If for example, the digit 9 binary 1001 is to be displayed with the decimal point lit,the bit pattern 0011 0000 must be output to Port B. To translate the 1001 to 0011 0000,the Pattern Table shown in appendix 1 is used. The bit patterns required to display 0 thru F with the decimal point lit are stored in locations 0040 thru 004F. If as in the example above,a 9 is to be displayed with the decimal point,the 9 is added to the base address 0040 to get 0049. The bit pattern contained in location 0049 is then output to Port B.

If the decimal point is not wanted,bit 0 is set to 1 before outputting . To do this,a decimal point byte is generated containing a 1 in bit 0 for all conditions other than AL mode and Display 2 or AH mode and Display 1. Bits 1 thru 7 are zeros (see appendix 2 Figure A2-1d). The decimal point byte is merged with the Pattern Table byte before outputting. See figure 11-3 for a flow chart of the display section.

See appendix 2 for a detailed description of Program 2.

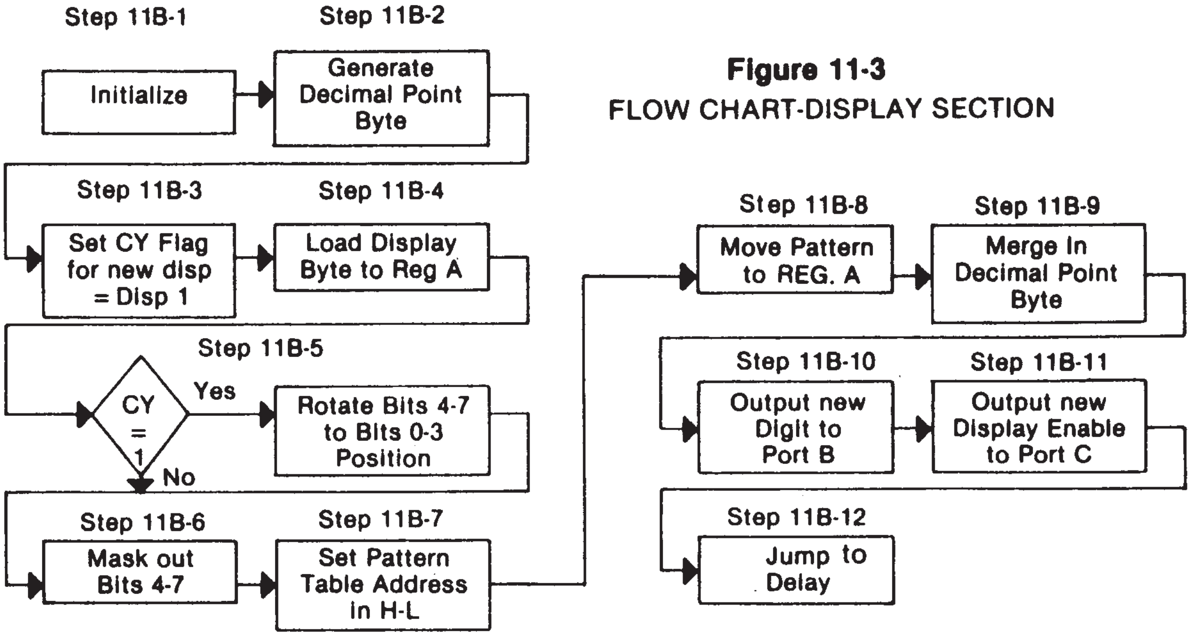

Step 11B-1
Initialize
Step 11B-2
Generate Decimal Point Byte
Step 11B-3
Set CY Flag for new disp = Disp 1
Step 11B-4
Load Display Byte to Reg A
Step 11B-5
CY = 1
Yes
No
Rotate Bits 4-7 to Bits 0-3 Position
Step 11B-6
Mask out Bits 4-7
Step 11B-7
Set Pattern Table Address in H-L
Step 11B-8
Move Pattern to REG. A
Step 11B-9
Merge In Decimal Point Byte
Step 11B-10
Output new Digit to Port B
Step 11B-11
Output new Display Enable to Port C
Step 11B-12
Jump to Delay

Figure 11-3
FLOW CHART-DISPLAY SECTION

PROCEDURE

1. Set the RESET switch up and all other switches down.
2. Connect the Power Supply and turn power ON.
3. Put the ENROM switch up to enable the ROM.
4. Put the WEN switch up to allow writing to the ROM.
5. Use the usual procedure (set the data switches to the address and push the ALE button, then set the data switches to the code to be written and push the WR button) to copy Program 2 to the ROM. Program 2 is listed in appendix 1 locations 0074 thru 00A4.
6. Use the usual procedure to copy the Pattern Table, addresses 0040 thru 004F as listed in appendix 1.
7. Put the A8 switch up to access memory locations 0100 thru 01FF.
8. Use the usual procedure to store FF at address location 011C. ROM address 011C is copied to RAM address 80EE and becomes the high order byte of the delay constant. The value FF gives a delay of approximately one half second.
9. Put the WEN switch down to prevent further writing to the ROM.
10. Use the usual procedure (set the data switches to the address to be verified and push the ALE button, then the RD button to read the data) to verify the program codes listed in Appendix 1 at locations 010A thru 012D. REMEMBER - ADDRESS LOCATION 011C HAS BEEN CHANGED TO FF.
11. Put the A8 switch down to return to memory locations 0000 thru 00FF.
12. Use the usual procedure to verify Programs 1 and 2 at locations 0000 thru 0002 and locations 0060 thru 00A4.
13. Use the usual procedure to verify the Pattern table stored in step 6.
14. Put the ENROM switch down to disable the ROM.
15. Put the RESET switch down to run Program 2. Since the program is initialized to Data (DA) Mode with the Display Data (DDA) location set to 00, both displays will show zeros with no decimal point. Display 1 and Display 2 will alternate at approx. ½ second intervals as determined by the delay constant set in step 8. As in Program 1, the port outputs are initially zero. This lights all segments including the decimal point of Display 2 during the first delay period.
16. Put the RESET switch up to stop the program.

CHANGE OF DELAY CONSTANT

17. Put the ENROM and WEN switches up to enable writing to the ROM.
18. Put the A8 switch up to access locations 0100 thru 01FF.
19. Use the usual procedure to store 05 at address location 011C. This address will be copied to RAM address 80EE and become the high order byte of the delay constant. The value 05 gives a delay of approximately 10MS.
20. Put the WEN switch down to prevent further writing to the ROM.
21. Use the usual procedure to verify the 05 at location 011C.
22. Put the ENROM switch down to disable the ROM. Like the data switches the A8 switch is overridden by the 8085 and may be left up.
23. Put the RESET switch down to run the program. Display 1 and Display 2 will alternate at approx. 10MS intervals making it appear that they are both on continuously.
24. Put the RESET switch up to stop the program.

SETTING AL MODE

25. Put the ENROM and WEN switches up to enable writing to the ROM.
26. Use the usual procedure to store FD in address 0128 and 12 in address 012B. See table 11-1. The FD in location 0128 is copied to RAM address 80FA and sets the program to AL mode. The 12 in location 012B is copied to DAL in RAM address 80FD and will be displayed.
27. Put the WEN switch down to prevent further writing to the ROM.
28. Use the usual procedure to verify the FD at address 0128 and 12 at address 012B.
29. Put the ENROM switch down to disable the ROM.
30. Put the RESET switch down to run the program. The display will show the 12 in the DAL byte. The Display 2 decimal point will be lit indicating the program is in AL mode.
31. Put the RESET switch up to stop the program.

SETTING AH MODE

32. Repeat steps 25 thru 31. In step 26 store FE in address 0128 to put the program in AH mode. Store 34 in address 012C to be displayed. In step 30 the display will show the 34 in the DAH byte and the display 1 decimal point will be lit indicating the program is in AH mode.

CHECKING THE PATTERN TABLE

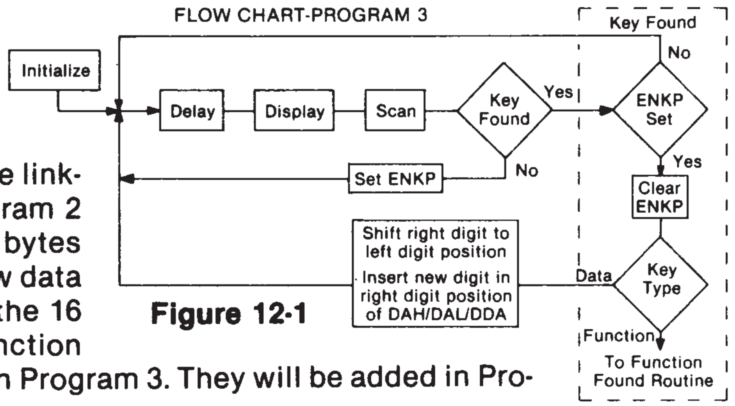
33. Repeat steps 25 thru 31. In step 26 store the numbers 56, 78 etc. in location 012C to verify the remaining bytes of the Pattern table (see appendix 1). The Display 1 decimal point will be on in each case indicating the program is in AH mode.

RESTORING

34. Repeat steps 25 thru 31. In step 26 restore the MODE L, DDA, DAL and DAH bytes to their original values as in table 11-1. In step 30, 00 will be displayed with no decimal point as in step 23.
35. Turn power off.

LESSON 12

Program 3 adds the Scan (locations 00A2 to 00D3), Key Found (locations 00D4 to 00E1) and Data Found (locations 00E2 to 00F0) sections of the Monitor program to program 2. Figure 12-1 shows a flow chart of Program 3. The Display and Scan sections are linked by replacing the JMP instruction in Program 2 at locations 00A2 thru 00A4 with the first 3 bytes of the Scan routine. The added sections allow data to be entered into the Monitor program via the 16 data keys shown in figure 12-2. The 8 function keys are scanned but will not be processed in Program 3. They will be added in Program 4.



FLOW CHART-PROGRAM 3
Initialize
Delay
Display
Scan
Key Found
Yes
No
Set ENKP
Shift right digit to left digit position
Insert new digit in right digit position of DAH/DAL/DDA
Key Found
No
ENKP Set
Yes
Clear ENKP
Key Type
Data
Function
To Function Found Routine

Figure 12-1

When a key is pressed,it is likely to be held down for several hundred milliseconds. During this time the keyboard will be scanned many times. The Enable Key Processing (ENKP) byte insures that the key will be processed only once each time it is pushed. If the Scan routine finds a key pushed and the ENKP byte set (FF), the program will first clear the ENKP byte to 00 before processing the key and looping back to the delay routine. As long as the ENKP remains zero,further key processing is inhibited. Scanning continues at 10MS intervals. When the key is released and a scan routine is executed without finding a key, the ENKP byte is again set to FF to enable the next key input. The 10MS interval between scans serves to debounce the keys.

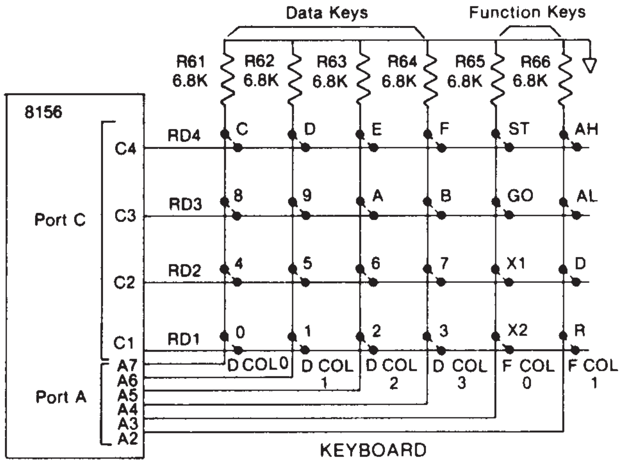
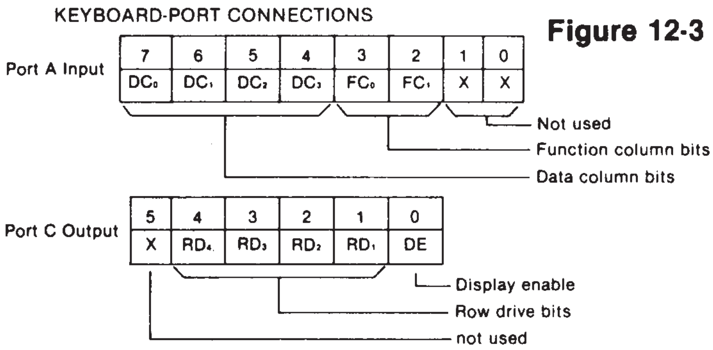


Figure 12-2

SCAN ROUTINE

The scan routine checks each of the 16 data and 8 function keys shown in figure 12-2 to see if it is pressed. The RST, ALE, WR, RD and RS keys are not scanned. The 24 scanned keys are arranged differently on the printed circuit board but are shown in figure 12-2 as 4 rows and 6 columns to illustrate the matrix nature of the interconnection. When a key is pressed,a row line is connected to a column line. For example if data key 5 is pressed, Row Driver 2 (RD2) is connected to Data Column 1 (D COL1).As shown in figure 12-3,Port C is configured as an output port and drives the 4 Row Driver lines. Port A is configured as an input port and senses the 6 column lines. In the example above,when key 5 is pushed a high voltage on RD2 will be sensed as a 1 on D COL1. All other column lines will be 0.


KEYBOARD-PORT CONNECTIONS

Port A Input

7	6	5	4	3	2	1	0
DC_0	DC_1	DC_2	DC_3	FC_0	FC_1	X	X

Port C Output

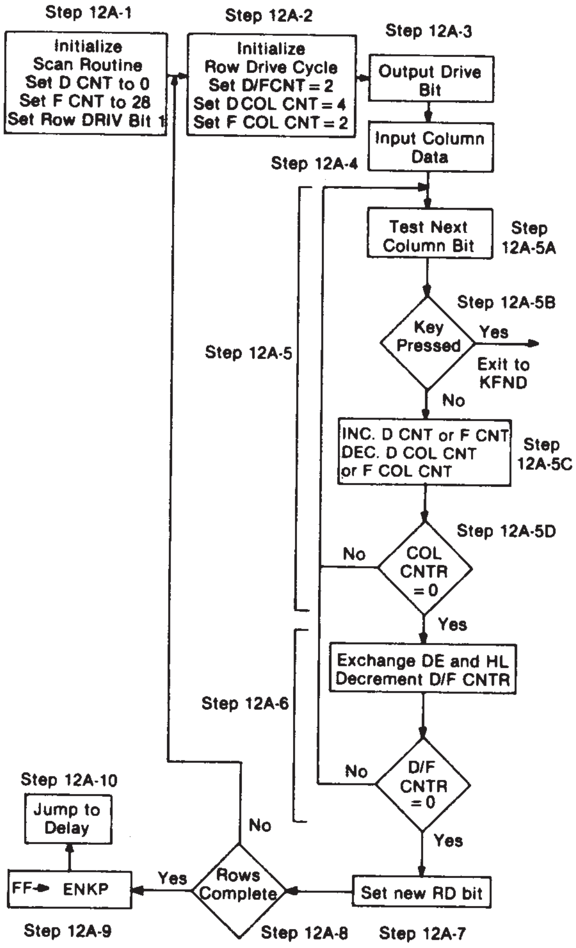
5	4	3	2	1	0
X	RD_4	RD_3	RD_2	RD_1	DE

Figure 12-3

Scanning is started by activating RD1. The column lines are then input to the A register through Port A. These column lines are then sensed sequentially from D COL0 thru F COL1 see figure 12-2. This Row Drive cycle is then repeated for RD2, RD3 and RD4. Refer to figure 12-4 for a flow chart of the scan routine. If a key is found to be pressed the program immediately exits the Scan routine and goes to the Key Found (KFND) routine. Figure 12-5 shows the 8085 register usage. The B register is used to store the current Row Drive bit. Initially bit 1 is set and then output to Port C to drive RD1. As each Row Drive cycle is complete the B register is shifted left one bit to obtain the next Row Drive bit. When the bit is shifted into bit 5 scanning is complete.

Five counters are used to control the scan:

DATA COUNTER (D CNT): The D CNT generates the hexadecimal code of the data key pressed. It is initially set to 0 and incremented each time a Data Column is tested and the key is found to be open (D COL line at a low voltage). Whenever a column is tested, either Data or Function, and found to be at a low voltage, the E register is incremented. As shown in figure 12-5, when Data Columns are tested the D CNT is in the E register and the Function Counter (F CNT) is in the L register. When Function Columns are tested the positions are reversed. The F CNT is in the E register and the D CNT in the L register.

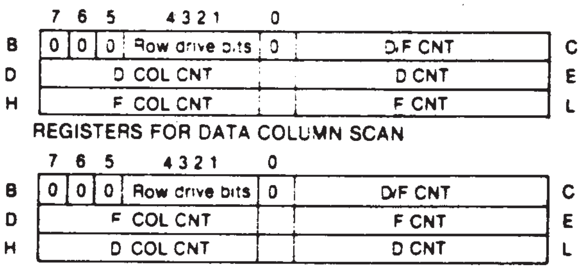


FLOW CHART-SCAN ROUTINE

Figure 12-4

FUNCTION COUNTER (F CNT): The F CNT indicates which function key is pressed in a manner simular to the D CNT. Since function keys are not implemented in Program 3, it will not be used here. Program 4 will implement the function keys.

DATA COLUMN COUNTER (D COL CNT): The D COL CNT keeps track of the number of Data Columns tested during each Row Drive cycle. Whenever a column is tested, either Data or Function,and found to be at a low voltage (0),the D register is decremented. As shown in figure 12-5 the D COL CNT is always paired with the D CNT. It is in the D register when Data Columns are scanned and in the H register when Function Columns are scanned. The D COL CNT is set to 04 at the start of each Row Drive cycle. When it has been decremented to zero the Data Column scan is complete.



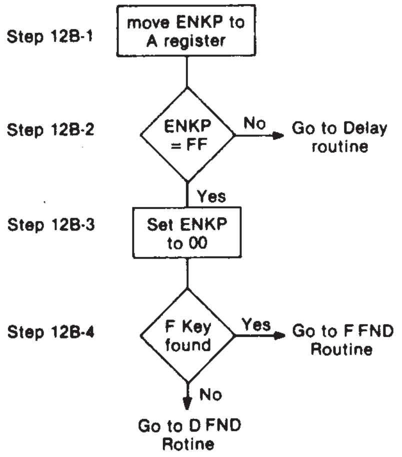
REGISTERS FOR DATA COLUMN SCAN

REGISTERS FOR FUNCTION COLUMN SCAN

Figure 12-5

FUNCTION COLUMN COUNTER (F COL CNT): The F COL CNT serves a purpose similar to the D COL CNT. It is in the D register when Function Columns are scanned and in the H register when Data Columns are scanned. It is set to 02 at the start of each Row Drive cycle and decremented for each Function Column tested. When zero,it signals the program that Function Column scanning is complete.

DATA/FUNCTION COUNTER (D/F CNT): The D/F CNT resides in the C register. It signals which column section, Data or Function is being scanned. The D/F CNT is set to 02 at the start of each Row Drive cycle. It is decremented to 01 at the completion of the Data Cloumn scan and to 00 at the completion of the Function Column scan. When zero,the D/F CNT signals the program that the Row Drive cycle is complete. The Key Found (KFND) routine also uses the D/F CNT to determine whether the key found was a data or a function key.

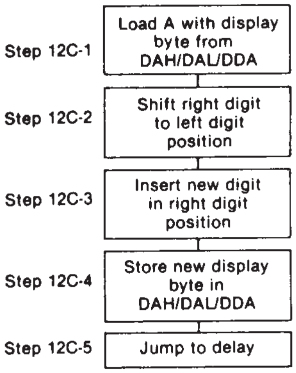


FLOW CHART-KEY FOUND ROUTINE

Figure 12-6

KEY FOUND ROUTINE: The Key Found KFND routine senses the state of the ENKP byte. If key processing is not enabled,control is transferred back to the Delay rountine. If key processing is enabled,the ENKP byte is cleared to disable the keyboard until the key is released. The D/F CNT is then sensed to determine if a data or function key was pressed. If a data key was pressed, control is transferred to the DFND routine. In Program 4 control,will be transferred to the F FND routine if a function key was pressed. Refer to figure 12-6 for a flow chart of the Key Found routine.

DATA FOUND ROUTINE: The Data Found (D FND) routine processes the data keys. The data key code is entered into the DDA, DAL or DAH byte as specified by the Mode H and Mode L byte. The right digit of the byte is shifted to the left digit position and the new key code is inserted in the right digit position. Refer to figure 12-7 for a flow chart of the D FND routine.


Step 12C-1
Load A with display byte from DAH/DAL/DDA
Step 12C-2
Shift right digit to left digit position
Step 12C-3
Insert new digit in right digit position
Step 12C-4
Store new display byte in DAH/DAL/DDA
Step 12C-5
Jump to delay

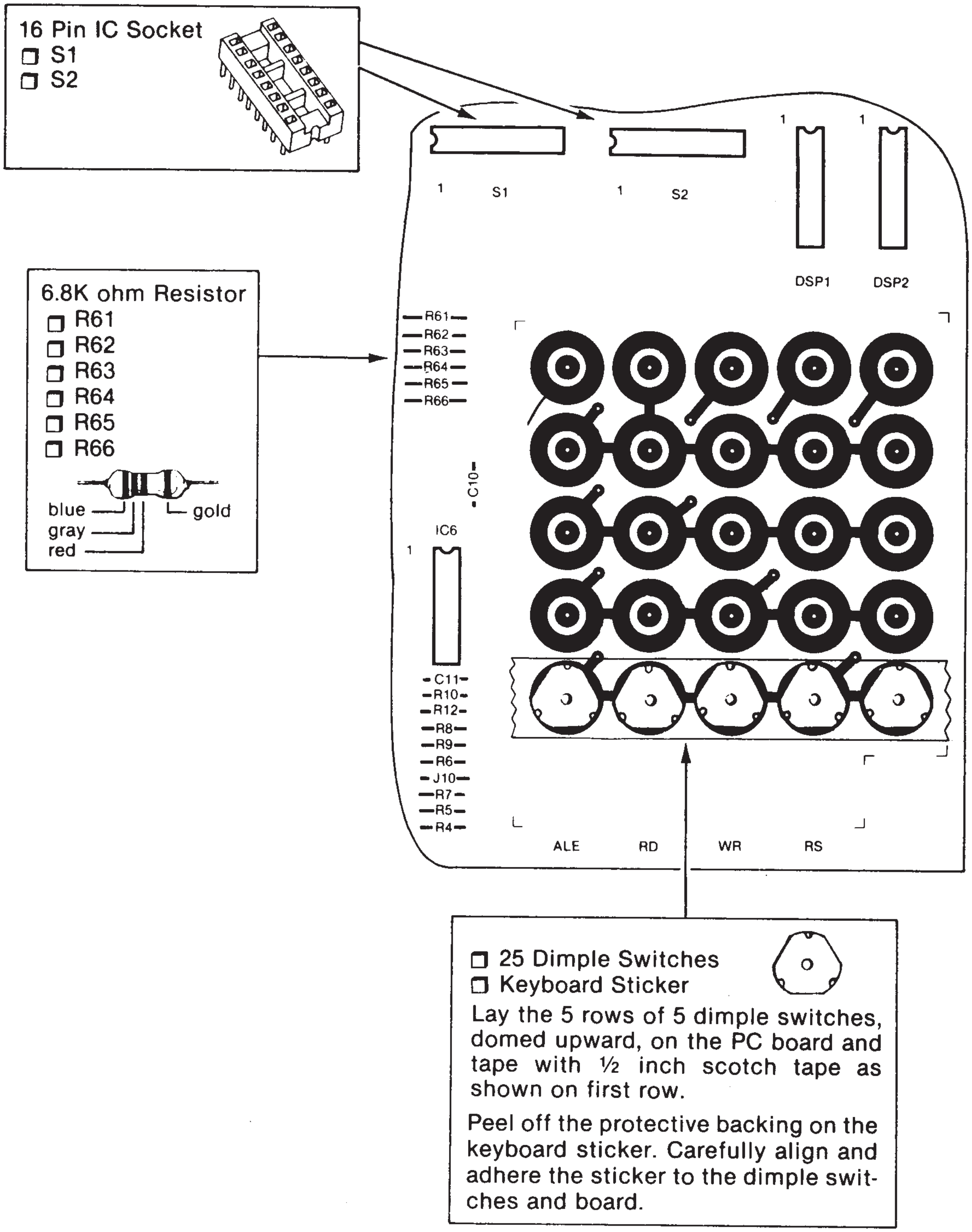
FLOW CHART-DATA FOUND ROUTINE

Figure 12-7

See appendix 2 for a detailed description of Program 3.

ASSEMBLY INSTRUCTIONS

Unplug the power supply. Identify and assemble the following parts as shown in figure 12-8. Place a check in the box provided after each part has been completed.



16 Pin IC Socket
□ S1
□ S2

6.8K ohm Resistor
□ R61
□ R62
□ R63
□ R64
□ R65
□ R66

□ 25 Dimple Switches
□ Keyboard Sticker

Lay the 5 rows of 5 dimple switches, domed upward, on the PC board and tape with ½ inch scotch tape as shown on first row.

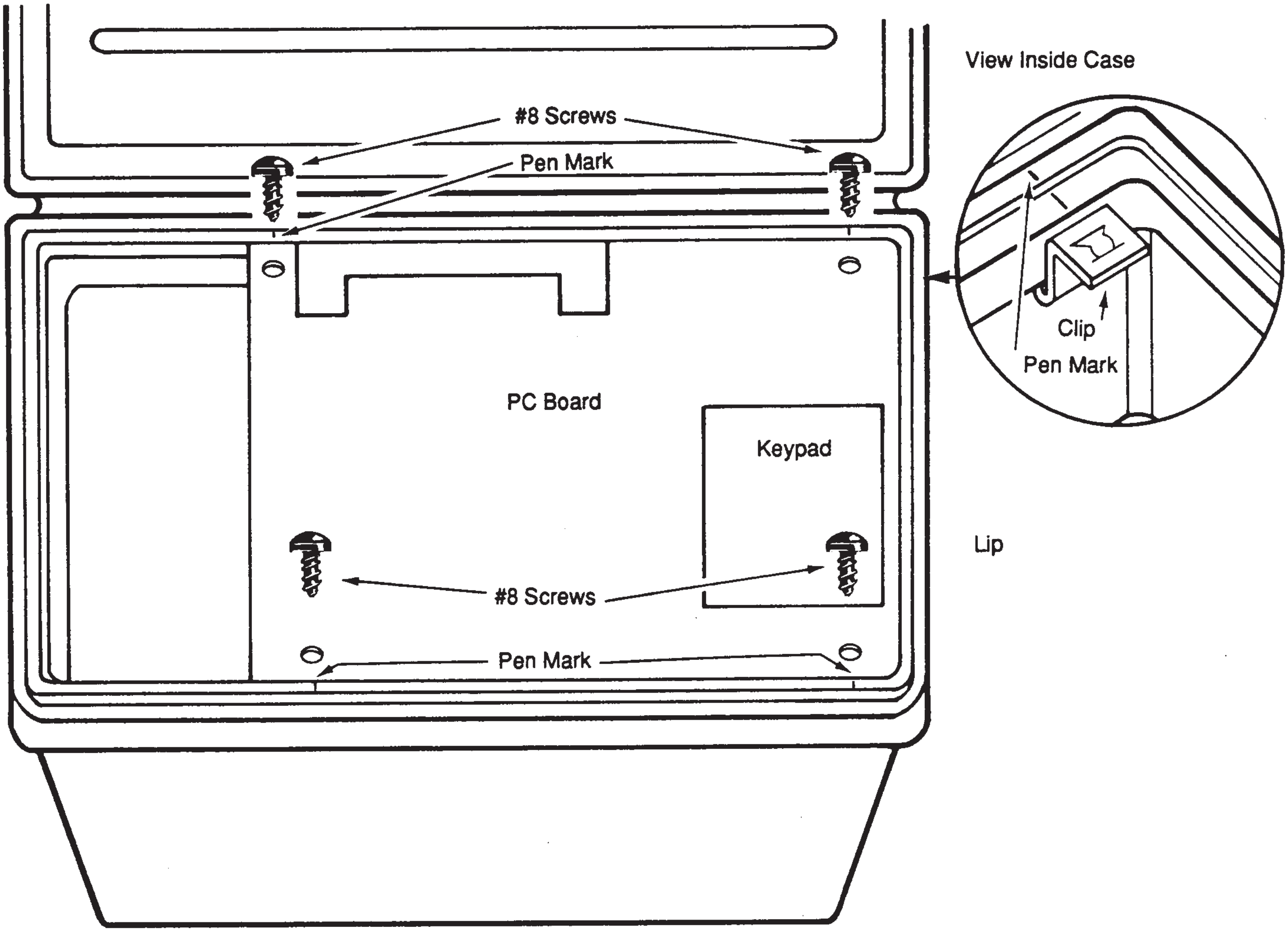
Peel off the protective backing on the keyboard sticker. Carefully align and adhere the sticker to the dimple switches and board.

Figure 12-8

If unit will not function properly see trouble shooting guide on page VII.

FINAL ASSEMBLY

□ Place the PC board in the case so its right edge is flush with the lip on the right side of the case, see figure below. Draw a line on the lip of the case across from the 4 holes in the PC board. Remove the PC board from the case and install the 4 clips so the hole in the clip lines up with the mark made on the lip. Place the PC board back in the case. Be sure the 4 holes in the PC board line up with the hole in the clips. If the holes don't line up, move the clip so that it will. Hold the PC board in place with four #8 screws.



□ Peel off the protective backing on the Micro Master label. Carefully align with the edges of the indentation on the top cover of the case and press in place.

CIRCUIT DESCRIPTION

A simplified schematic of the Restart (RST) switch circuitry is shown in figure 12-9. With the Reset switch up, the $\overline{\text{RESET IN}}$ pin of the 8085 is at ground and the RST switch has no effect. When the Reset switch is down and the 8085 is running, pressing the RST switch places a ground (0) on the $\overline{\text{RESET IN}}$ pin and stops the 8085. Releasing the RST switch ungrounds the $\overline{\text{RESET IN}}$ pin. This has the same effect as putting pole A of the RESET switch down as described in lesson 10.

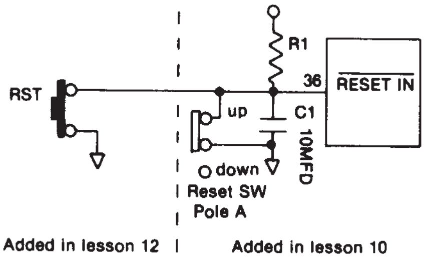


Figure 12-9

Figure 12-2 shows a simplified schematic of the 24 scanned keys added in lesson 12. Each key is at the intersection of a row and a column line. Pressing the key connects the row to the column line. Port C is configured as an output port. Placing a 1 in Port C, bits C1 thru C4 places a high voltage on the corresponding Row Drive line RD1 thru RD4. Only one Row Drive line is made high at a time. Port A is configured as an input port. The column line Data Column 0 (D COL0) thru Function Column 1 (F COL1), are input to Port A on lines A7 thru A2. The column lines are held to a low voltage (0) thru resistors R61 thru R66. A column line is driven to a high (1) voltage only when a switch connects it to an active (high) Row Drive line.

PROCEDURE

1. Set the RESET switch up and all other switches down.
2. Connect the Power Supply and turn power ON.
3. Put the ENROM switch up to enable the ROM.
4. Put the WEN switch up to enable writing to the ROM.
5. Use the usual procedure to copy Program 3 to the ROM. Program 3 is listed in Appendix 1 locations 00A2 thru 00F0.
6. Put the WEN switch down to prevent further writing to the ROM.
7. Use the usual procedure to verify programs 1, 2 and 3 at locations 0000 thru 0002 and 0060 thru 00F0.
8. Use the usual procedure to verify the Pattern Table at locations 0040 thru 004F.
9. Put the A8 switch up to access memory locations 0100 thru 01FF.
10. Use the usual procedure to verify address locations 010A thru 012D.
11. Put the EROM switch down to disable the ROM.
12. Put the RESET switch down to run the program. As in lesson 11, the program starts in Data Mode with DDA set to 00. The display therefore shows 00 with no decimal point.
13. Push the 1 button. The display changes to 01.
14. Push the 2 button. The 1 in the right digit is shifted to the left digit and the 2 is inserted in the right digit.
15. Push buttons 3 thru F. Note that each is inserted in the right digit and the previous digit shifted to the left digit.
16. Push the F button twice to obtain FF in the display.
17. Put the RESET switch up and then down again. The display is now 00. The FF displayed in step 16 was in the DDA byte in RAM not in ROM. The program is still initialized to display 00.
18. Repeat step 16.
19. Push and release the RST button. Note that it has the same effect as putting the RESET switch up and then down again.
20. Put the RESET switch up.
21. Turn power OFF.

LESSON 13

Program 4 implements the eight function keys AH, AL, DA, R, ST, GO, X1 and X2. To do this,the Function Found (F FND) routine and the eight function routines, Function Address High (FAH) thru Function X2 (FX2),are added to program 3. With these additions the Monitor program is complete. A flow chart of the complete Monitor program is shown in Appendix 3.

The Function Found (F FND) routine (in locations 00F1 thru 00FF) directs the Monitor program to the starting address of the selected function routine. The function routines (in locations 0100 thru 0109 and 80DC thru 80EB) perform the tasks required by the particular function key and then loop the program back to the Delay routine.

Function Found Routine (F FND)

The Monitor program is directed to the proper Function routine by means of the Function Table shown in appendix 1. The Function Table contains the starting address of each of the Function routines. The addresses appear in the table in the same order as the associated keys are scanned. Two byte are required for each address. The low order byte appears first followed immediately by the high order byte. There are therefore two bytes between addresses.

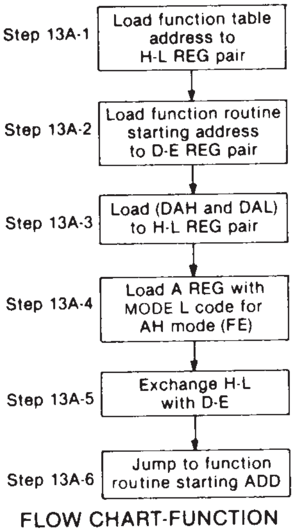
In lesson 11 the hexadecimal data code was used to access the proper address in the Pattern Table. In a similar manner, the F CNT is used to access the proper address in the Function Table. At the start of the scan routine, the F CNT is initialized to 28(H), (see lesson 12). This is the base address (50 (H)) of the Function Table shifted right 1 bit. Shifting right 1 bit is equivalent to division by 2. 50H/2 = 28(H). As each function key is scanned the F CNT is incremented by 1. Column 1 of Table 13-1 lists the function keys in the order they are scanned. Columns 2 and 3 list the binary and hexadecimal values of the F CNT when the corresponding key is found. Since each address in the Function Table takes 2 locations,the count generated in the F CNT (and the initial value) must be multiplied by 2 to obtain the correct Function Table address. To multiply, the F CNT is shifted left 1 bit. The binary and hexadecimal result of the shift is shown in columns 4 and 5 of Table 13-1. This result is the required low order byte of the Function Table address.

FUNCTION	F CNT INTITIAL		F CNT SHIFTED LEFT [F CNT X2]	
	BINARY	HEX	BINARY	HEX
X2	0010 1000	28	0101 0000	50
R	0010 1001	29	0101 0010	52
X1	0010 1010	2A	0101 0100	54
D	0010 1011	2B	0101 0110	56
GO	0010 1100	2C	0101 1000	58
AL	0010 1101	2D	0101 1010	5A
ST	0010 1110	2E	0101 1100	5C
AH	0010 1111	2F	0101 1110	5E

FUNCTION TABLE ADDRESS GENERATION

Table 13-1

See Figure 13-1 for a flow chart of the F FND routine. The F FND routine first generates the Function Table address in the H-L register pair. The Function Routine starting address from the Function Table is then loaded into the D-E register pair. To facilitate the execution of the Function routines the DAH and DAL bytes are put in the H-L register pair and the MODE L code for AH mode (FE) is put in the A register. The H-L and D-E register pairs are then interchanged and control is transferred to the Function routine. When the program enters the Function routines,the DAH and DAL bytes are in the D-E register pair.


Step 13A-1
Load function table address to H-L REG pair
Step 13A-2
Load function routine starting address to D-E REG pair
Step 13A-3
Load (DAH and DAL) to H-L REG pair
Step 13A-4
Load A REG with MODE L code for AH mode (FE)
Step 13A-5
Exchange H-L with D-E
Step 13A-6
Jump to function routine starting ADD

FLOW CHART-FUNCTION FOUND ROUTINE

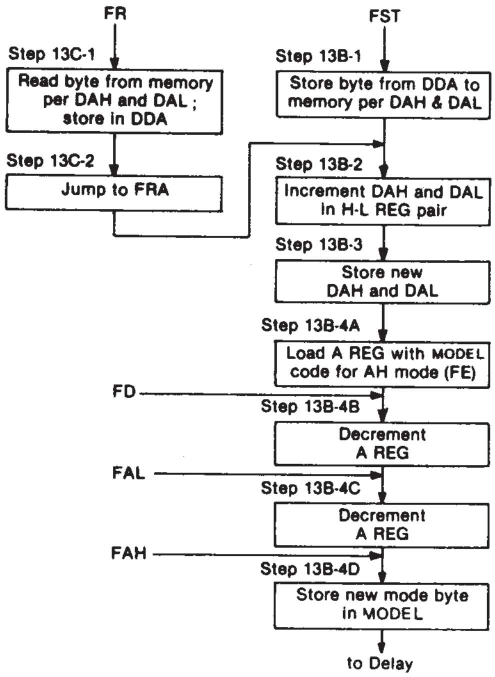
Figure 13-1

Function Store Routine (FST)

Refer to figure 13-2 for a flow chart of FST routine. The FST routine stores the byte in DDA to the memory location specified by DAH and DAL. The address in DAH and DAL is then incremented by one and the program is placed in DA mode. To facilitate the execution of the FAH, FAL and FDA routines the program is placed in DA mode by first setting the A register to the MODE L code for AH mode (FE). The A register is then decremented twice to the MODE L code for DA mode (FC) and stored in MODE L.

Function Read Routine (FR)

Refer to figure 13-2 for a flow chart of the FR routine. The FR routine reads the byte at the address specified by DAH and DAL and stores it in DDA to be displayed. Control is then transferred to the FST routine to increment DAH and DAL and place the program in DA mode.


FR
FST
Step 13C-1
Read byte from memory per DAH and DAL ; store in DDA
Step 13C-2
Jump to FRA
Step 13B-1
Store byte from DDA to memory per DAH & DAL
Step 13B-2
Increment DAH and DAL in H-L REG pair
Step 13B-3
Store new DAH and DAL
Step 13B-4A
Load A REG with MODEL code for AH mode (FE)
FD
Step 13B-4B
Decrement A REG
FAL
Step 13B-4C
Decrement A REG
FAH
Step 13B-4D
Store new mode byte in MODEL
to Delay

FLOW CHART-FUNCTION ROUTINES

Figure 13-2

Function Routines FAH, FAL and FDA

The only task required of the Function Address High (FAH), Function Address Low (FAL) and Function Data (FDA) routines is to set the program mode by storing the appropriate byte in MODE L. Before transfering control to the Function routines, the Function Found routine places the MODE L code for AH mode (FE) in the A register. When the AH button is pushed, control is transferred to the instruction labled FAH at location 80E9 in the FST routine (see appendix 1 and figure 13-2). This instruction stores the AH code (FE) in the MODE L byte. When the AL button is pushed control is transferred to the instruction labled FAL. The (FE) in the A register is then decremented to the code for AL mode (FD) before being stored in the MODE L byte. When the DA button is pressed, control is transferred to the instruction labled FDA and the A register is decremented twice before being stored in the MODE L byte.

Function GO Routine

The Function GO (FGO) routine transfers control to the instruction whose address is specified by DAH and DAL.

Function Routines FX1 and FX2

The Function X1 (FX1) and Function X2 (FX2) routines are reserved for future use. Presently, the Function table contains 0000 as a starting address of the FX1 and FX2 routines. They will therefore reinitialize the Monitor program.

See appendix 2 for a detailed description of Program 4.

PROCEDURE

1. Put the RESET switch up and all other switches down.
2. Connect the Power Supply and turn power ON.
3. Put the ENROM switch up to enable the ROM.
4. Put the WEN switch up to allow writing to the ROM.
5. Use the usual procedure to store the Function Table and the Function Found section of program 4 at locations 0050 thru 005F and 00F1 thru 00FF. See appendix 1.
6. Put the A8 switch up to access memory locations 0100 to 01FF.
7. Use the usual procedure to store the Function GO and Function Read sections of program 4 at locations 0100 thru 0109.
8. Put the WEN switch down to prevent further writing to the ROM.
9. Use the usual procedure to verify the codes stored in step 7 and ROM locations 010A thru 012D.
10. Put the A8 switch down to access memory locations 0000 thru 00FF.
11. Use the usual procedure to verify the remainder of the Monitor program at locations 0000 thru 0002 and 0040 thru 00FF.
12. Put the ENROM switch down to disable the ROM.

CHECKING MODES

13. Put the RESET switch down to run the program. As in lessons 11 and 12,the program starts in Data (DA) mode with DDA set to 00. The display therefore shows 00 with no decimal point.
14. Push the 1 button twice. The DDA byte now contains 11 and 11 is displayed.
15. Push the AL button. The program is now in Address Low (AL) mode which lights the right (display 2) decimal point. Since the DAL byte is initialized to 00,the display shows 00.
16. Push the 2 button twice. The DAL byte is now 22 and the display shows 22.
17. Push the AH button. The program is now in Address High (AH) mode which lights the left (display 1) decimal point. Since the DAH byte is initialized to 00,the display shows 00.
18. Push the 3 button twice. The DAH byte is now 33 and the display shows 33.
19. Push the DA button. The program returns to Data (DA) mode. No decimal point is lit. The display shows the 11 in byte DDA.
20. Push the AL button. The program returns to AL mode. The right decimal point is lit. The display shows the 22 in byte DAL.
21. Push the AH button. The program returns to AH mode. The left decimal point is lit. The display shows the 33 in byte DAH.

STORING-READING IN RAM

22. Push the AH button to put the monitor program in AH mode.
23. Push the 8 and then the 0 button to store 80 in DAH. If DAH already contains the desired number, it need not be reloaded.
24. Push the AL button to put the Monitor program in AL mode.
25. Push the 0 button twice to store 00 in DAL. If DAL already contains the desired number, it need not be reloaded.
26. Push the DA button to put the Monitor program in DA mode.
27. Push the 1 and then the 2 button to store 12 in DDA. If DDA already contains the desired number, it need not be reloaded.
28. Push the ST button to store the 12 from DDA to address 8000 as specified by DAH and DAL. The address in DAH and DAL is incremented by 1.
29. Push the AL button to put the program in AL mode. Note that DAL has been incrementvd to 01. When the ST button is pushed again,the data will be stored in address 8001.
30. Push the DA button to put the program in DA mode.

31. Repeat steps 27 and 28 to complete the following table.

ADDRESS	DATA
8000	12
8001	34
8002	56
8003	78
8004	9A
8005	BC
8006	DE
8007	F0

32. Push the AL button to put the program in AL mode. Note that DAL is 08.
33. Repeat steps 22 thru 25 to return DAH and DAL to 8000.
34. Push the R button to verify the first address of the table. The 12 at address 8000 will be read and stored in DDA. The program will be put in DA mode to diaplay DDA. The address in DAH and DAL will be incremented by 1.
35. Repeat step 34 to verify the remainder of the table.

STORING-READING IN ROM

36. Put the WEN switch up to allow writing to ROM.
37. Repeat steps 22 thru 25 setting DAH and DAL to 012E. This is the byte after the last byte of the Monitor program.
38. Push the DA button to put the program in DA mode.
39. Repeat steps 27 and 28 to store the following:

ADDRESS	DATA
012E	11
012F	22
0130	33

40. Put the WEN switch down to prevent further writing to ROM.
41. Repeat steps 22 thru 25 setting DAH and DAL back to 012E.
42. Repeat step 34 to verify the table stored in step 39.

CHECKING THE GO FUNCTION

43. Repeat steps 22 thru 25 to set DAH and DAL to 8000.
44. Push the DA button to put the program in DA mode.
45. Repeat steps 27 and 28 to store the following table:

ADDRESS	MNEMONIC	CODE
8000	JMP	C3
8001		00
8002		80

This is a jump instruction which unconditionally transfers control back to itself. Once executed, it will repeat indefinitely.

46. Repeat steps 22 thru 25 to return DAH and DAL to 8000.
47. Repeat step 34 to verify the table stored in step 45.
48. Repeat steps 22 thru 25 to again return DAH and DAL to 8000.
49. At this point the program is in AL mode displaying DAL (00). Push the GO button. Control is transferred to the jump instruction at location 8000 which repeats indefinitely. The Monitor program is no longer running. Depending on its state when the GO button was pushed,the display contains either the left or the right zero. The other display is blank.
50. Push the RST button to restart the Monitor program.

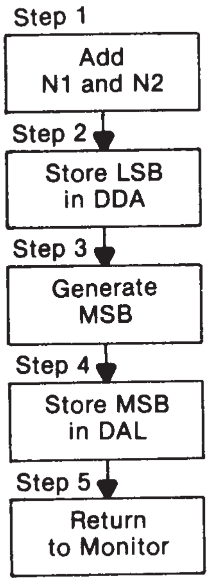
CHECKING X1 AND X2

51. Push the 1 button to set the display to 01.
52. Push the X1 button. The program is reinitialized. The display shows 00.
53. Repeat steps 51 and 52 using X2. The X2 button has the same effect as X1.
54. Turn power off.

LESSON 14

With the monitor program complete, new programs may be entered into memory via the keyboard. The programs are executed by entering their starting address in DAH and DAL and pushing the GO button. These programs may then run indefinitely or, when completed, may stop the microprocessor or return to the monitor program. Program 5 will demonstrate this.

Program 5 adds the two single byte numbers N1 and N2 previously stored in memory at locations 8000 and 8001. The LSB (least significant byte) of the sum is placed in DDA. The carry out from bit 7 generates the MSB (most significant byte) which is placed in DAL. The Monitor program is then reentered at the SET DA instruction at location 80E5. (see Appendix 1). This places the Monitor in DA mode displaying the LSB of the sum in DDA. The MSB may be displayed by pushing the AL button. Refer to Figure 14-1 for a flow chart and to Table 14-1 for a program list of Program 5.



FLOW CHART
PROGRAM 5

Figure 14-1

PROGRAM 5

	ADDRESS	MNEMONIC	CODE	COMMENTS
Step 1	8000	N1		
	8001	N2		
	8002	LXI HL	21	Load H-L with address of N1
	8003		00	
	8004		80	
	8005	MOV AM	7E	Move N1 to A register
	8006	INX HL	23	Increment HL to address of N2
	8007	ADD M	86	N1 + N2 (LSB)→A register
Step 2	8008	LXI HL	21	Load HL register with address of DDA
	8009		FC	
	800A		80	
	800B	MOV MA	77	Store LSB in DDA
Step 3	800C	MVI A	3E	0 to A register
	800D		00	
	800E	ADC A	8F	Generate MSB; 0 + 0 + CY→A
Step 4	800F	INX HL	23	Increment to address of DAL
	8010	MOV MA	77	Store MSB to DAL
Step 5	8011	JMP	C3	Return to monitor at SET DA [80E5]
	8012	SET DA	E5	
	8013		80	

Table 14-1

STEP 1.
The LXI HL (Load register pair immediate) instruction at location 8002 loads the H-L register pair with the address of N1 as specified in locations 8003 and 8004. The MOV AM (Move from memory) instruction at location 8005 loads the A register with N1 as specified in the H-L register pair. The INX HL (Increment register pair) instruction at location 8006 steps the H-L register pair to the address of N2. The ADD M (Add memory) instruction at location 8007 adds N1 in the A register to N2 as specified by the H-L register pair. The result is placed in the A register. The flag is set if there is a carry out of bit 7. The A register now contains the LSB of the sum.

STEP 2.
The LXI HL (Load register pair immediate) instruction at location 8008 loads the HL register pair with the address of DDA as specified in locations 8009 and 800A. The MOV MA (Move to memory) instruction at location 800B stores the LSB of the sum from the A register to DDA as specified by the H-L register pair.

STEP 3.
The MVI A (Move immediate) instruction at location 800C moves the 00 at location 800D to the A register. The ADC A (Add Register with carry) instruction at location 800E adds the A register and the CY bit to the A register and places the sum in the A register. Since the A register is zero the sum is simply the CY bit. The CY bit was last set or cleared by the ADD M instruction in Step 1. The A register therefore contains 01 if there was a carry out from the addition of N1 and N2 or 00 if there was no carry. This is the MSB of the sum.

STEP 4.
The INX HL (Increment register pair) instruction at location 800F steps the H-L register pair to the address of DAL. The MOV M A (Move to memory) instruction at location 8010 stores the MSB to DAL as specified by the H-L register pair.

STEP 5.
The JMP (Jump) instruction at location 8011 transfers control to the SET DA instruction at location 80E5 in the Monitor program (See Appendix 1). Entry at this point places the Monitor in DA mode displaying the LSB of the sum of N1 and N2.

PROCEDURE

1. Put the RESET switch up and all other switches down.
2. Connect the Power Supply and turn power ON.
3. Put the RESET switch down to start the Monitor program.
4. Push the AH button to put the program in AH mode.
5. Push the 8 and then the 0 buttons to store 80 in DAH.
6. Push the AL button to put the program in AL mode.
7. Push the 0 and then the 2 button to store 02 in DAL.
8. Push the DA button to put the program in DA mode.
9. Push the 2 and then the 1 button to store 21in DDA.
10. Push the ST button to store the 21 in DDA to the address 8002 as contained in DAH and DAL.
11. Repeat steps 9 and I0 to store the remainder of Program 5 as listed in Table 14-1.
12. Repeat steps 4 thru 7 to set DAH and DAL back to 8002.
13. Push R to verify the 21 at address 8002.
14. Repeat step 13 to verify the remainder ot Table 14-1.

ADDING NUMBERS

15. Repeat steps 4 thru 7 setting DAH and DAL to 8000.
16. Push the DA button to put the program in DA mode.
17. Push the 1 and then the 7 button to store 17 (H) in DDA.
18. Push the ST button to store the 17 (H) in DDA to address 8000as contained in DAH and DAL. This sets N1 to 17 (H) (decimal 23).
19. Push the 3 and then the 2 button to store 32 (H) in DDA.
20. Push the ST button to store the 32 (H) in DDA to address 8001. This sets N2 to 32 (H) (decimal 50).
21. After storing the 32 (H) in location 8001, the DAH and DAL address is incremented to 8002 which is the starting address of Program 5. Push the GO button to run the program. Program 5 places the LSB of the sum in DDA and the MSB in DAL. Control is then returned to the Monitor program. The Monitor is placed in DA mode which displays the LSB of the sum. 49(H) = 17(H) + 32(H) (in decimal 73 = 23 + 50)
22. Push the AL button to display DAL. The MSB is 00(H) indicating there was no carry.
23. Repeat steps 15 thru 22 storing 3A(H) in address 8000 and E1(H) in 8001. In step 21 the LSB of the sum is 1B(H). In step 22 the MSB is 01(H). 11B(H)=3A(H) + E1(H) (in decimal 283 = 58+225).
24. Turn power off.

APPENDIX 1

PROGRAM 1

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
START	10-1	0000	JMP	C3	Jump to INIT
		0001	INIT	60	
		0002		00	
INIT	10-2	0060	LXI DE	11	Load D-E with 1st ROM address to be copied to RAM
		0061		0A	
		0062		01	
	10-3	0063	LXI HL	21	Load H-L with 1st RAM address
		0064		DC	
		0065		80	
COPY	10-4	0066	LDAX DE	1A	Move byte from ROM to A
	10-5	0067	MOV MA	77	Store byte in RAM
	10-6	0068	INX HL	13	Increment H-L and D-E
		0069	INX DE	23	
	10-7	006A	MOV AL	7D	Set Z flag if RAM adress excedes address to be copied to
		006B	CPI	FE	
		006C		00	
	10-8	006D	JNZ	C2	If Z flag not set, go back and copy next byte
		006E	copy	66	
		006F		00	
	10-9	0070	MVI A	3E	Set:
		0071		0E	Port A to input
		0072	OUT	D3	Port B to output
		0073		80	Port C to output
	10-10★	0074	HLT	76	Stop

See last page of Appendix 1 for pattern table locations 0040-004F and for function table locations 0050-005F

★ This instruction is used in Program 1 only. It is replaced by location 0074 of Program 2.

Program 2

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
Link		0074	JMP	C3	This instruction links programs 1 and 2.
		0075	delay	EC	It replaces the HLT at 0074 in program 1.
		0076		80	
DISP	11B-1	0077	LHLD	2A	Load MODE H and MODE L to H-L register pair
		0078		FA	
		0079		80	
		007A	INP	DB	Input port C
		007B		83	
		007C	XRI	EE	Complement bit 0 to change display.
		007D		01	Other port C bits unaffected
		007E	MOV BA	47	Save new port C in REG B
	11B-2a	007F	XRA L	AD	C0 = 1 For
		0080	MOV CA	4F	[DA∧DISP1]V[AL∧DISP2]V[AH∧DISP1]
	11B-2b	0081	MOV AL	7D	A0 = 1 For AL V AH
		0082	RRC	OF	
		0083	ADC H	8C	
	11B-2c	0084	ANA C	A1	A0 = 1 For [AL∧DISP2]V[AH∧DISP1]
	11B-2d	0085	CMA	2F	Complement A
	11B-2e	0086	ANI	E6	Mask out all but A0
		0087		01	= 1 For DAV[AL∧DISP1]V[AH∧DISP2]
		0088	NOP	00	
		0089	MOV CA	4F	Save decimal point byte in REG C
	11B-3	008A	MOV AB	78	Set C8 flag for left digit to be output
		008B	RRC	0F	
	11B-4	008C	MOV AM	7E	Fetch display byte
	11B-5	008D	JNC	D2	If DISP2 (right) digit is to be output go to RDIG.
		008E	RDIG	94	Do not interchange digits
		008F		00	
		0090	RRC	0F	Interchange left and right digits
		0091	RRC	0F	
		0092	RRC	0F	
		0093	RRC	0F	
RDIG	11B-6	0094	ANI	E6	Mask out bits A4-A7
		0095		0F	
	11B-7	0096	ORI	F6	Merge in base address of pattern table
		0097		40	
		0098	MOV LA	6F	Store pattern table address (low) in L REG
		0099	MVI H	26	Store pattern table address (high) in H REG
		009A		00	
	11B-8	009B	MOV AM	7E	Fetch pattern to A REG
	11B-9	009C	ORA C	B1	Merge in decimal point byte
	11B-10	009D	OUT B	D3	Output pattern to port B
		009E		82	
	11B-11	009F	MOV AB	78	Fetch new port C to A
		00A0	OUT C	D3	Output new port C to port C
		00A1		83	
	11B-12★	00A2	JMP	C3	Jump to delay
		00A3	delay	EC	
		00A4		80	

★ This instruction is used in program 2 only. It is replaced by locations 00A2 thru 00A4 of program 3.

PROGRAM 3

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
Scan	12A-1	00A2	MVI E	1E	Set D CNT to 00 in E register
		00A3		00	
		00A4	MVI L	2E	Set F CNT to 28 in L register
		00A5		28	
		00A6	MVI B	06	Set RDI = 1 register B = 02
		00A7		02	
New DRV	12A-2	00A8	MVI C	0E	Set D/F CNT = 02 .
		00A9		02	
		00AA	MVI D	16	Set D COL CNT = 04 in D register
		00AB		04	
		00AC	MVI H	26	Set F COL CNT = 02 in H register
		00AD		02	
	12A-3	00AE	INP C	DB	Input port C
		00AF		83	
		00B0	ANI	E6	Mask out row drive field bits 1-4
		00B1		E1	
		00B2	ORA B	B0	Merge in new row drive bit
		00B3	OUT C	D3	Output new row drive bit to port C
		00B4		83	
	12A-4	00B5	INP A	DB	Input columns from port A
		00B6		81	
TST key	12A-5a	00B7	RLC	07	Set CY flag if current key is pushed
	12A-5b	00B8	JC	DA	If key is pushed go to K FND
		00B9	K FND	D4	
		00BA		00	
	12A-5c	00BB	INR E	1C	Key not found; set up for next key. Increment
		00BC	DCR D	15	D CNT (F CNT) Decrement D COL CNT (F COL CNT)
	12A-5d	00BD	JNZ	C2	If D COL CNT (F COL CNT) is not zero
		00BE	TST key	B7	go back and test next column
		00BF		00	
	12A-6	00C0	XCHG	EB	Exchange register pairs D-E and H-L
		00C1	DCR C	0D	Decrement D/F CNTR
		00C2	JNZ	C2	If D/F CNT not zero go back and test F COL0 and F COL
		00C3	TST key	B7	
		00C4		00	
	12A-7	00C5	MOV AB	78	Column scan is complete
		00C6	RLC	07	Set up next row drive bit in register B.
		00C7	MOV BA	47	
	12A-8	00C8	SUI	D6	Compare row drive bit to bit 5.
		00C9		20	If equal, Z flag is set. Scan is complete
		00CA	JNZ	C2	If scan not complete go back and output
		00CB	New DRV	A8	new row drive bit
		00CC		00	
	12A-9	00CD	CMA	2F	Here scan is complete with no key found.
		00CE	STA	32	Set FF to ENKP
		00CF		FF	
		00D0		80	
	12A-10	00D1	JMP	C3	Jump to delay. Start new program cycle
		00D2	delay	EC	
		00D3		80	
K FND	12B-1	00D4	LXI HL	21	Load H-L register pair with
		00D5		FF	address of ENKP byte
		00D6		80	
		00D7	MOV AM	7E	Load ENKP to A register

PROGRAM 3 Con't

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
	12B-2	00D8	RRC	0F	Set C8 flag if ENKP is set
		00D9	JNC	D2	Loop back to delay if ENKP not set
		00DA	delay	EC	
		00DB		80	
	12B-3	00DC	MVI M	36	Clear ENKP to 00
		00DD		00	
	12B-4	00DE	DCR C	0D	Decrement D/F CNT
		00DF	JZ	CA	Jump to F FND if key was function key
		00E0	F FND	F1	(D/F CNT went to 00)
		00E1		00	
D FND	12C-1	00E2	LHLD	2A	Load H-L register pair with MODE H and MODE L
		00E3		FA	
		00E4		80	
		00E5	MOV AM	7E	Move DDA/DAL/DAH to reg pair
	12C-2	00E6	RLC	07	Shift right digit to left digit position
		00E7	RLC	07	
		00E8	RLC	07	
		00E9	RLC	07	
	12C-3	00EA	ANI	E6	Clear right digit
		00EB		F0	
		00EC	ORA E	B3	Insert new digit in right digit position
	12C-4	00ED	MOV MA	77	Store new byte in DDA/DAL/DAH
	12C-5	00EE	JMP	C3	Jump to delay
		00EF	delay	EC	
		00F0		80	

PROGRAM 4

NAME	STEP	ADDRESS	MNOMONIC	CODE	COMMENTS
F FND	13A-1	00F1	MOV AE	7B	Move F CNT to A. Multiply by 2 to get low order byte
		00F2	RLC	07	of function table address, store IN L.
		00F3	MOV LA	6F	
		00F4	MVI H	26	Set high order byte of function table address (00) to H
		00F5		00	
	13A-2	00F6	MOV EM	5E	Move function routine starting address
		00F7	INX HL	23	from function table to D-E register pair
		00F8	MOV DM	56	
	13A-3	00F9	LHLD	2A	Move (DAH and DAL) to H-L register pair
		00FA		FD	
		00FB		80	
	13A-4	00FC	MVI A	3E	Move address of DAH (FE) to A register
		00FD		FE	
	13A-5	00FE	XCHG	EB	Move function routine address to H-L reg pair
	13A-6	00FF	PCHL	E9	Move H-L to PC; jump to function routine
FGO	13B-1	0100	XCHG	EB	Move DAH and DAL to H-L reg pair
	13B-2	0101	PCHL	E9	Jump to address specified by DAH and DAL
FR	13C-1	0102	XCHG	EB	Move (DAH and DAL) back to H-L from D-E
		0103	MOV AM	7E	Load A with byte to be read per H-L
		0104	STA	32	Store byte to be read in DDA to be displayed
		0105		FC	
		0106		80	
	13C-2	0107	JMP	C3	Jump to FRA to complete read function
		0108	FRA	E1	
		0109		80	

PROGRAM SEGMENT COPIED ROM TO RAM

ADDRESS	CODE	Copied to RAM Location
010A	EB	80DC
010B	3A	80DD
010C	FC	80DE
010D	80	80DF
010E	77	80E0
010F	23	80E1
0110	22	80E2
0111	FD	80E3
0112	80	80E4
0113	3E	80E5
0114	FE	80E6
0115	3D	80E7
0116	3D	80E8
0117	32	80E9
0118	FA	80EA
0119	80	80EB
011A	21	80EC
011B	00	80ED
011C	05	80EE
011D	2D	80EF
011E	C2	80F0
011F	EF	80F1
0120	80	80F2
0121	25	80F3
0122	C2	80F4
0123	EF	80F5
0124	80	80F6
0125	C3	80F7
0126	77	80F8
0127	00	80F9
0128	FC	80FA
0129	80	80FB
012A	00	80FC
012B	00	80FD
012C	00	80FE
012D	FF	80FF

PROGRAM 4
COPIED FROM ROM

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
FST	13B-1	80DC	XCHG	EB	Move (DAH and DAL) back to H-L from D-E
		80DD	LDA	3A	Load A reg with byte to be stored (from DDA)
		80DE		FC	
		80DF		80	
		80E0	MOV MA	77	Store byte per H-L reg pair
FRA	13B-2	80E1	INX HL	23	Increment (DAH and DAL) in H-L
	13B-3	80E2	SHLD	22	Store new DAH and DAL
		80E3		FD	
		80E4		80	
SET DA	13B-4a	80E5	MVI A	3E	Store MODE L code for AH mode (FE) in A register
		80E6		FE	
FDA	13b-4B	80E7	DCR A	3D	Decrement MODE L code
FAL	13B-4c	80E8	DCR A	3D	Decrement MODE L code
FAH	13B-4d	80E9	STA	32	Store new MODE L code in MODE L
		80EA		FA	
		80EB		80	

PROGRAM 2
COPIED FROM ROM

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
Delay	11A-1	80EC	LXI HL	21	Load H-L register pair with delay constant
		80ED		00	
		80EE		05	
D Loop	11A-2	80EF	DCR L	2D	Decrement L
	11A-3	80F0	JNZ	C2	If L is not zero go back and decrement again
		80F1	D loop	EF	
		80F2		80	
	11A-4	80F3	DCR H	25	Decrement H
	11A-5	80F4	JNZ	C2	If H is not zero go back and recycle L
		80F5	D loop	EF	
		80F6		80	
	11A-6	80F7	JMP	C3	Delay is complete
		80F8	display	77	go to display
		80F9		00	

PARAMETERS
COPIED FROM ROM

NAME	STEP	ADDRESS	MNEMONIC	CODE	COMMENTS
MODE L		80FA		FC	Code shown is initial value
MODE H		80FB		80	copied from ROM
DDA		80FC		00	
DAL		80FD		00	
DAH		80FE		00	
ENKP		80FF		FF	

PATTERN TABLE

ADDRESS	DISPLAY	CODE HEX	CODE BINARY 7654	3210
0040	0	80	1000	0000
0041	1	F2	1111	0010
0042	2	48	0100	1000
0043	3	60	0110	0000
0044	4	32	0011	0010
0045	5	24	0010	0100
0046	6	04	0000	0100
0047	7	F0	1111	0000
0048	8	00	0000	0000
0049	9	30	0011	0000
004A	A	10	0001	0000
004B	B	06	0000	0110
004C	C	8C	1000	1100
004D	D	42	0100	0010
004E	E	0C	0000	1100
004F	F	1C	0001	1100

FUNCTION TABLE

ADDRESS	FUNCTION	CODE starting address
0050	FX2	00
0051		00
0052	FR	02
0053		01
0054	FX1	00
0055		00
0056	FDA	E7
0057		80
0058	FGO	00
0059		01
005A	FAL	E8
005B		80
005C	FST	DC
005D		80
005E	FAH	E9
005F		80

APPENDIX 2

The following is a detailed description of programs 1 thru 4. See Appendix 1 for program listings.

PROGRAM 1
Refer to Figure 10-1 for a flow chart of Program 1.

STEP 10-1
The JMP (Jump) instruction at location 0000 loads the Program Counter with the address contained in the next two bytes of the instruction. The 60 at address 0001 is put in the low order byte and the 00 at address 0002 is put in the high order byte of the Program Counter. The next instruction executed then comes from address 0060. In other words, control is transferred to location 0060.

STEP I0-2
The LXI DE (Load Register Pair Immediate) instruction at location 0060 loads the register pair D-E with the number contained in the next two bytes of the instruction. The low order byte 0A from location 0061 is put in the E Register. The high order byte 01 from location 0062 is put in the D Register. The number 0I0A is the address of the first ROM byte to be copied to RAM.

STEP 10-3
As in step 2 the LXI HL (Load Register Pair Immediate) instruction at location 0063 loads the number DC to the L Register and 80 to the H Register. The number 80DC is the address of the first byte in RAM to be copied to.

STEP 10-4
To transfer a byte from one memory location to another, the byte must first be loaded into the A Register and then transferred to the new memory location. The LDAX DE (Load Accumulator Indirect) instruction at memory location 0066 loads the A Register with the byte from memory location 010A which is specified by the D-E register pair.

STEP 10-5
The MOV MA (Move to Memory) instruction at location 0067 transfers the number in the A Register to memory location 80DC as specified by the H-L register pair.

STEP 10-6
The transfer of the first byte is now complete. To transfer the second and successive bytes the D-E and H-L register pairs are increased by one and steps 4 and 5 are repeated. The INX HL (Increment Register Pair) instruction at location 0068 increases the H-L register pair by one. The INX DE (Increment Register Pair) instruction at location 0069 increases the D-E register pair by one.

STEP 10-7
If the last byte copied was to location 80FF, the copy process is complete and the H-L register pair has been incremented to 8100. Since copying started at 80DC it is only necessary to check the L Register to see if it is 00. If it is the program proceeds to step 8. If it is not the program loops back to step 4 and continues copying data from ROM to RAM. The MOV AL (Move Register) instruction at location 006A moves the contents of the L Register to the A Register. The CPI (Compare Immediate) instruction at location 006B subtracts the next byte of the instruction (00 at location 006C) from the A Register. If the A Register is zero the result of the subtraction is zero and the Z flag is set. The Z flag is thus set when copying is complete.

STEP 10-8. If the Z flag is not set the JNZ (Jump Not Zero) instruction at location 006D transfers control to location 0066 as specified by the next two bytes of the program and copying continues. If the Z flag is set control is transferred to the instruction following the JNZ instruction.

STEP 10-9
The MVI A (Move Immediate) instruction at location 0070 loads the A Register with the next byte of the instruction, in this case the OE at location 0071. In lesson 4 it was shown that when the code OE is sent to the Command Status Register of the 8155 Port A is made an input port and ports B and C are made output ports. The OUT (Output instruction at location 0072 transfers the contents of the A Register to the I/O port specified by the next byte of the program. In this case the 80 at location 0073 specifiesthe Command Status register. The transfer is accomplished in exactly the same manner as was done manually in lesson 4. First the 80 is put on the data bus and latched with the ALE line. Next the OE in the A Register is put on the data bus and transferred to the I/O section of the 8155 by the IO/$\overline{M}$ and $\overline{WR}$ lines. During the execution of the OUT instruction the low order address 80 is repeated on the high order address lines A8 thru A15. The MM-8000 uses the 1 appearing on line A15 to enable the 8155.

STEP 10-10
The HLT (Halt) command at location 0074 stops the 8085.

PROGRAM 2

DELAY ROUTINE

Refer to figure 11-2 for a flow chart of the Delay Routine.

STEP 11A-1
The LXI HL (Load register pair immediate) instruction at location 80EC loads the delay constant 0500 from locations 80ED and 80EE to the H-L register pair, 00 in the L register and 05 in the H register.

STEP 11A-2
The DCR L (Decrement Register) instruction at location 80EF decreases the L register by one. The first time the instruction is executed, the L register steps from binary 0000 0000 to 1111 1111, next to 1111 1110 and so on down to 0000 0000. When the register is stepped to zero, the Z flag is set.

STEP 11A-3
If the Z flag is not set, the JNZ (Jump not zero) instruction at location 80FO transfers control back to theDCR L instruction at location 80EF as specified in locations 80F1 and 80F2. If the Z flag is set control is transferred to the next instruction in the list.

STEP 11A-4
The DCR H (Decrement Register) instruction at location 80F3 decreases the H register by one. If the register goes to zero the Z flag is set.

STEP 11A-5
If the Z flag is not set the JNZ (Jump not zero) instruction at location 80F4 transfers control to the DCR L instruction at address 80EF as specified in locations 80F5 and 80F6. If the Z flag is set control is transferred to the next instruction in the list.

STEP 11A-6
Here the delay is complete. The JMP (Jump) instruction at location 80F7 unconditionally transfers control to the instruction at address 0077 as specified in locations 80F8 and 80F9.

DISPLAY ROUTINE

Refer to Figure 11-3 for a flow chart of the Display Routine.

STEP 11B-1

The LHLD (Load H and L direct) instruction at location 0077 loads the L register with the contents of the memory location specified in locations 0078 and 0079. The contents of the next memory location is loaded in the H register. In this case the contents of MODE L and MODE H from addresses 80FA and 80FB are loaded to the L and H registers. The MODE bytes will be used to access the byte to be displayed and to generate the decimal point byte. The INP (Input) instruction at location 7A loads the A register with the data placed on the data bus by the port specified in the second byte of the instruction. Here the 83 at location 007B specifies Port C which is presently configured as an output port. As was demonstrated in lesson 4, the data being output by Port C is thus loaded into the A register. The XRI (Exclusive OR immediate) instruction at location 007C performs an exclusive or between the A register and the data in the second byte of the instruction (here 01). The result is placed in the A register. Since bits 1 thru 7 of the second byte of the instruction are 0, these bits of the A register are unchanged. The 1 in bit 0 of the second byte of the instruction complements bit 0 of the A register. The MOV BA (Move Register) instruction at location 007E stores the data in register A in register B. Later this data will be output to Port C. Since bit 0 drives the displays the new Port C will light the opposite display.

STEP 11B-2

In step 11B-2 a decimal point byte is generated which contains a 0 in bit 0 if the decimal point is to be lit and a 1 if the decimal point is to be unlit. Bits 1 thru 7 are 0. Figure A2-1d shows this final decimal point byte which will be merged with a byte from the pattern table and output to Port B to drive the display. To arrive at the final decimal point byte an interim byte is generated with a 1 instead of a 0 in bit 0 if the decimal point is to be lit. The conditions for lighting the decimal point are shown in Figure A2-1b. Step 11B-2a generates a byte with bit 0 a 1 for the conditions shown in Figure A2-1c. The unwanted 1 in DA mode is masked out by steps 11B-2b and 11B-2c. Step 11B-2d complements bit 0. Step 11B-2e masks out bits 1 thru 7 and saves the final decimal point byte in register C.

a

	MODE H	MODE L
	76543210	76543210
Data (DA) Mode	10000000	11111100
Address Low (AL) Mode	10000000	11111101
Address High (AH) Mode	10000000	11111110

b

	MODE		
	DA	AL	AH
Display 1 (left)	0	0	1
Display 2 (right)	0	1	0

c

	MODE		
	DA	AL	AH
Display 1 (left)	1	0	1
Display 2 (right)	0	1	0

d

	Final decimal point byte
	76543210
Decimal point lit	00000000
Decimal point out	00000001

DECIMAL POINT BYTE GENERATION

Figure A2-1

STEP 11B-2a

The XRA L (Exclusive OR Register) instruction at location 007F performs an exclusive OR operation between the new Port C byte in the A register and the MODE L byte in the L register. The result is placed in the A register. The MODE L byte is shown in the Figure A2-1a. Since a 1 in bit 0 of the new Port C byte drives Display 1 the result of the exclusive OR for bit 0 is shown in Figure A2-1c. This result is stored in Register C by the MOV CA (Move Register) instruction at location 0080.

STEP 11B-2b
Step 11B-2b generates a 1 in A0 for modes AL or AH. A0 is 0 for DA mode. The MOV AL (Move Register) instruction at location 0081 loads the A register with the MODE L byte in register L. The RRC (Rotate right) instruction at location 0082 shifts each bit of the A register one place to the right. A0 is shifted to the CY bit. In DA mode A0 and the CY bit are both 0. In AL or AH mode either A0 or CY bit is a 1. The ADC H (Add Register with carry) instruction at location 0083 adds the contents of the A register, H register and CY bit. The result is placed in the A register. (Note the H register is used here only because H0 is a 0. The fact that H contains the MODE H byte has no significance). Since H0 is 0 the sum in bit 0 is 1 for AL and AH modes and 0 for DA mode.

STEP 11B-2c
The ANA C (AND Register) instruction at location 0084 AND's the A register with the byte (corresponding to Figure A2-1c) saved in step 11B-2a. Since A0 contains a 0 in DA mode, the unwanted 1 in DA mode is eliminated. Bit 0 now corresponds to Figure A2-1b.

STEP 11B-2d
The CMA (Complement accumulator) instruction at location 0085 complements the A register, all ones are changed to 0 and all zeros to 1. Bit 0 now corresponds to the final decimal point byte in Figure A2-1d.

STEP 11B-2e
The ANI (AND immediate) instruction at location 0086 ANDS the A register with the next byte of the instruction (the 01 at location 0087). The result is placed in the A register. This leaves bit 0 unchanged and sets bits 1 thru 7 to 0.The NOP (NO op) instruction at location 0088 has no effect on the program. The MOV CA (Move Register) instruction at location 0089 saves the final decimal point byte in register C.

STEP 11B-3
The MOV AB (Move Register) instruction at location 008A loads the A register with the new Port C byte in register B. If bit 0 is a 1, the left (high order) digit of the display byte will be output to Display 1. If bit 0 is 0 the right (low order) digit will go to Display 2. The RRC (Rotate right) instruction at location 008B shifts each bit of the A register one place to the right. A0 is shifted to A7 and the CY bit. The CY bit is thus set if the left (high order) digit is to be output.

STEP 11B-4
The MOV AM (Move from Memory) instruction at location 008C loads the A register with the contents of the memory address in the H and L registers. Since these registers contain the Mode H and Mode L bytes as shown in Figure A2-1a, the byte to be displayed (DDA, DAL or DAH) is loaded into the A register. The CY bit is not changed

STEP 11B-5
Step 11B-5 places the digit to be displayed in the low order position (Bits 0-3). If the CY bit is 0, the JNC (Jump no carry) instruction at location 008D transfers control to the instruction labled RDIG (Right digit) at loction 0094. The low order digit remains in A0-A3. If the CY bit is a 1, control is transferred to the RRC (Rotate right) instruction at location 0090. The four RRC instructions at 0090 thru 0093 shift the high order digit into the low order position A0-A4.

STEP 11B-6
The ANI (AND immediate) instruction at location 0094 performs an AND between the A register and the OF in the second byte of the instruction. The result is placed in the A register. This sets A7-A4 to 0 and leaves A3-A0 unchanged.

STEP 11B-7
Step 11B-7 sets the appropriate Pattern Table address in the H-L register pair. The ORI (OR immediate) at location 0096 OR'S the A register and the 40 at location 0097. The result is placed in the A register. This is the low order byte of the Pattern Table address. The MOV LA (Move Register) instruction at location 0098 stores this byte in the L register. The MVI H (Move immediate) instruction at location 0099 stores the high order byte of the Pattern Table address (00 from location 009A) in the H register.

STEP 11B-8
The MOV AM (Move from memory) instruction at location 009B loads the A register with contents of the memory location whose address is in the H-L register pair. This is the digit pattern (with decimal point lit) to be displayed.

STEP 11B-9
The ORA C (OR Register) instruction at location 009C OR's the digit pattern in the A register with the final decimal point byte in the C register. The result is put in the A register.

STEP 11B-10
The OUT B (Output) instruction at location 009D outputs the new digit pattern in the A register to Port B as specified by the 82 in location 009E.

STEP 11B-11
The MOV AB (Move register) instruction at location 009F loads the A register with the New Port C byte in register B. The OUT C (Output) instruction at location 00AO outputs the New Port C byte to Port C as specified by the 83 in location 00A1.

STEP 11B-12
The JMP (Jump) instruction at location 00A2 transfers control to the delay routine at address 80EC. The Program 2 loop shown in Figure 11-1 is then repeated.

PROGRAM 3

SCAN ROUTINE

Refer to Figure 12-4 for a flow chart of the Scan routine.

STEP 12A-1
Step 12A-1 initializes the scan routine. The MVI E (Move immediate) instruction at location 00A2 loads the E register with the contents of the next byte of the instruction. This sets the D CNT to 00.

In the same way the MVI L instruction at location 00A4 sets the F CNT, in the L register to 28. This will not be used in Program 3. The MVI B instruction at location 00A6 loads 02 into the B register. The one in bit 1 is the initial Row Drive bit.

STEP 12A-2
Step 12A-2 initializes the Row Drive cycle consisting of steps 2 thru 8. The MVI C instruction at location 00A8 sets the D/F CNT in register C to 02. The MVI D instruction at location 00AA sets the D COL CNT in register D to 04. The MVI H instruction at location 00AC sets the F COL CNT in register H to 02.

STEP 12A-3
The INP C (Input) instruction at location OOAE loads the A register with the contents of Port C as specified by the 83 at location 00AF. The ANI (AND immediate) instruction at location 00B0 performs an AND between the E1 at location 00B1 and the A register. The result is placed in the A register. Since the byte E1 contains zeros in bits 1 thru 4, the Row Drive field is cleared. The ORA B (OR Register) instruction at location 00B2 merges in the new Row Drive bit from register B. The OUT C (Output) instruction at location 00B3 outputs the A register containing the new Row Drive bit to Port C as specified by the 83 in location 00B4.

STEP 12A-4
The INP A (Input) instruction at location 00B5 inputs Port A to the A register. This loads D COL O thru F COL I into A register bits A7 thru A2.

STEP 12A-5
Step 12A-5 is a program loop which scans either the 4 Data Column bits or the 2 Function Column bits. Immediately after inputting Port A, the D and E registers contain the D COL CNT and D CNT respectively. Data Column bits D COL O thru D COL 3 are in A register bits A7 thru A4. With the Row Drive bit RD1 active the data keys will be scanned as follows:

STEP 12A-5A
The RLC (Rotate left) instruction at location 00B7 shifts the A register left one bit. Bit A7 is shifted to A0 and to the CY bit. The CY bit is thus set if data key 0 is pressed.

STEP 12A-5B
If the CY bit is set the JC (Jump carry) instruction at location 00B8 transfers control to the Key Found (K FND) routine as specified in locations 00B9 and 00BA. Scanning is terminated. The D CNT contains 0, the hexadecimal code of the data key depressed. The D/F CNT in register C contains 02 indicating that scanning was terminated during a Data Column scan and the E register contains the date key code rather than function key information. If the CY bit is not set control is transferred to the next instruction.

STEP 12A-5C
The INR E (Increment Register) instruction at location 00BB steps the D CNT in register E to 01 which is the hexadecimal code of the next key to be tested. The DCR D (Decrement Register) instruction at location 00BC decrements the D COL CNT to 03. After D COL 3 is tested on the 4th pass through the loop the D COL CNT is decremented to 00 and the Z flag is set.

STEP 12A-5D
If the Z flag is not set the JNZ (Jump not zero) instruction at location 00BD transfers control back to the RLC instruction at location 00B7 as specified in locations 00BE and 00BF. On this next pass through the loop D COL 1 is shifted into the CY bit to test data key 1. When all four data keys are checked the Z flag is set in step 12A-5C and control is transferred out of the loop to the next instruction.

STEP 12A-6
The XCHG (Exchange H-L with D-E) instruction at location 00C0 interchanges the H-L register pair with the D-E register pair. After the Data Column scan described above the D COL CNT-D CNT pair is put in the H-L register pair and F COL CNT-F CNT goes to the D-E register pair ready for a Function Column scan. After the Function Column scan the registers are again interchanged. The DCR C (Decrement Register) instruction at location 00C1 decrements the D/F CNT in the C register. After a Data Column scan it is decremented to 01. After a Function Column scan it is decremented to 00 and the Z flag is set. When the D/F CNT is decremented to 01, the Z flag is not set and the JNZ (Jump not zero) instruction at location 00C2 transfers control back to the RLC instruction at location 00B7. The program then performs the Function Column scan. If the Z flag is set the column scan is complete and control is transferred to the next instruction in the list.

STEP 12A-7
The MOV AB (Move Register) instruction at location 00C5 transfers the contents of the B register containing the current Row Drive bit to the A register. The RLC (Rotate left) instruction at location 00C6 shifts the Row Drive bit left one position. The MOV BA (Move Register) instruction at location 00C7 stores the new Row Drive bit back in the B register.

STEP 12A-8
Bit 4 is the last Row Drive bit used. When scanning is complete Step 12A-7 shifts the Row Drive bit to bit 5. The SUI (Subtract immediate) instruction at location 00C8 subtracts the byte 20, in location 00C9, from the A register. The result is placed in the A register. If the Row Drive bit was in bit 5 the result of the subtraction is zero and the Z flag is set indicating scanning is complete. If the scanning is not complete the JNZ (Jump not zero) instruction at location 00CA transfers control back to the MVI C instruction at location 00A8 starting a new Row Drive cycle. If scanning is complete, control is tranferred to the next instruction.

STEP 12A-9
At Step 12A-9 scanning is complete with no key found. The A register is zero as a result of the SUI instruction at location 00C8. The CMA (Complement A Register) instruction at location 00CD complements each bit of the A register. The result is FF. The STA (Store A Register direct) instruction at location 00CE stores the A register in the memory location specified by locations 00CF and 00D0. This sets the ENKP byte at location 80FF.

STEP 12A-10
The JMP (Jump) instruction at location 00D1 transfers control to the delay routine at location 80EC.

KEY FOUND ROUTINE
Refer to Figure 12-6 for a flow chart of the Key Found routine.

STEP 12B-1
The LX1 HL (Load register pair immediate) instruction at location 00D4 loads the H-L register pair with the address of the ENKP byte 80FF. The MOV AM (Move from memory) instruction at location 00D7 loads the A register with the ENKP byte as specified by the H-L register pair.

STEP 12B-2
The RRC (Rotate right) instruction at location 00D8 shifts the A register one bit to the right. A0 is moved to A7 and the CY bit. The CY bit is set if key processing is enabled(ENKP = FF).If key processing is not enabled the JNC (Jump no carry) instruction at location 00D9 transfers control to the delay routine at address 80EC. If key processing is enabled control is transferred to the next instruction.

STEP 12B-3
The MVI M (Move to memory immediate) instruction at location 00DC stores the next byte of the instruction in the memory location specified by the H-L register pair. The ENKP byte is thus cleared to 00.

STEP 12B-4
The DCR C (Decrement Register) instruction at location 00DE decrements the D/F CNT. If the key found is a function key the D/F CNT goes to zero and sets the Z flag. If the Z flag is set the JZ (Jump zero) instruction at location 00DF transfers control to the Function Found (F FND) routine at location 00F1. This routine is not implemented in Program 3. If the Z flag is not set the key found is a data key and control is transferred to the next instruction at address 00E2. This is the start of the Data Found (D FND) routine.

DATA FOUND ROUTINE

Refer to Figure 12-7 for a flow chart of the Data Found routine.

STEP 12C-1

The LHLD (Load H and L direct) instruction at location 00E2 loads the H and L registers with the MODE H and MODE L bytes as specified by locations 00E3 and 00E4. The MOV AM (Move from memory) instruction at location 00E5 loads the A register with the DDA, DAL or DAH byte as specified by the Mode bytes.

STEP 12C-2

The four RLC (Rotate left) instructions at locations 00E6 thru 00E9 shift the right digit of the A register bits A0 thru A3 to the left digit, A4 thru A7. At the same time the left digit is rotated into the right digit position and must be cleared before inserting the new key code.

STEP 12C-3

The ANI (AND immediate) instruction at location 00EA ANDS the A register with the F0 byte in location 00EB. The result is placed in the A register. This clears the right digit. The ORA E (OR Register) instruction at location 00EC OR'S the left digit in the A register with the new data code in the E register and places the result in the A register.

STEP 12C-4

The MOV MA (Move to memory) instruction at location 00ED stores the new byte back in DDA, DAL or DAH as specified by the MODE bytes in the H-L register pair.

STEP 12C-5

The JMP (Jump) instruction at location 00EE transfers control to the delay routine.

PROGRAM 4

FUNCTION FOUND ROUTINE

See Figure 13-1 for a flow chart of the F FND Routine.

STEP 13A-1

The MOV AE (Move Register) instruction at location 00F1 loads the A register with F CNT from the E register. The RLC (Rotate left) instruction at location 00F2 shifts the A register left l bit. A7 (a zero) is shifted into A0. This multiplies the F CNT by 2. The MOV LA (Move Register) instruction at location 00F3 transfers F CNT X2, the low order byte of the Function Table address, to the L register. The MVI H (move immediate) instruction at location 00F4 loads the 00 from location 00F5 to the H register. This is the high order byte of the Function Table address.

STEP 13A-2

The MOV EM (Move from memory) instruction at location 00F6 loads the E register with the low order byte of the Function Routine starting address as specified by the H-L register pair. The INX HL (Increment register pair) instruction at location 00F7 increments the Function Table address in the H-L register pair. The H-L register pair now specifies the high order byte of the Function Routine starting address. The MOV DM (Move from memory) instruction at location 00F8 loads the D register with the high order byte of the Function Routine starting address as specified by the H-L register pair.

STEP 13A-3
The LHLD (Load H and L direct) instruction at location 00F9 loads the H and L registers with the DAH and DAL bytes as specified by locations 00FA and 00FB.

STEP 13A-4
The MVI A (Move immediate instruction at location 00FC loads the A register with the number FE from location 00FD. This is the MODE L code specifying AH Mode.

STEP 13A-5
The XCHG (Exchange H-L with D-E) instruction at location 00FE interchanges the H-L register pair with the D-E register pair. This puts DAH and DAL in the D-E register pair and the Function Routine starting address in the H-L register pair.

STEP 13A-6
The PCHL (move H and L to PC - Jump H and L indirect) instruction at location 00FF transfers the H-L register pair to the Program Counter (PC). Control is therefore transferred to the Function Routine starting address.

FUNCTION STORE ROUTINE
Refer to Figure 13-2 for a flow chart of the Function Store routine (FST).

STEP 13B-1
The XCHG (Exchange H-L with D-E) instruction at location 80DC moves DAH and DAL to the H-L register pair from the D-E register pair. The DAH and DAL bytes were placed in D-E by Steps 13A-3 and 13A-5 of the Function Found routine. The LDA (Load A Register direct) instruction at address 80DD loads the A register with the DDA byte from the address 80FC as specified in locations 80DE and 80DF. The MOV MA (Move to memory)instruction at location 80E0 moves the DDA byte from the A register to the memory address specified by DAH and DAL in the H-L register pair. The DAH and DAL address was entered via the keyboard and may specify an address in ROM. For this reason the remainder of the Function Store routine and the following Delay routine reside in RAM instead of ROM.

STEP 13B-2
The INX HL (Increment register pair) instruction at location 80E1 increments DAH and DAL in the H-L register pair.

STEP 13B-3
The SHLD (Store H and L direct) instruction at location 80E2 stores the L register with DAL in location 80FD as specified in location 80E3 and 80E4. The H register with DAH is stored in the next location, 80FE.

STEP 13B-4
Step 13B-4 sets the MODE L byte to FC which puts the monitor program in DA Mode. To do this the A register is first set to FE, the MODE L code for AH Mode, and then decremented twice to specify DA Mode before storing in the MODE L byte. This is done to facilitate execution of the FDA, FAL and FAH routines as will be explained below.

STEP 13B-4A
The MVI A (Move immediate) instruction at location 80E5 loads the A register with FE from location 80E6.

STEPS 13-4B and 13-4C
The two DCR A(Decrement Register) instruction at loctions 80E7 and 80E8 each decrement the A register one count.

STEP 13B-4D
The STA (Store A register direct) instruction at location 80E9 moves the MODE L code from the A register to the MODE L byte as specified in locations 80EA and 80EB.

FUNCTION READ ROUTINE

Refer to Figure 13-2 for a flow chart of the Function Read routine (FR).

STEP 13C-1
The XCHG (Exchange H-L with D-E) intruction at location 0102 moves DAH and DAL to the H-L register pair from the D-E register pair. The DAH and DAL bytes were placed in D-E by Steps 13A-3 and 13A-5 of the Function found routine. The MOV AM (Move from memory) instruction at location 0103 loads the A register with the byte specified by DAH and DAL. The STA (Store accumulator direct) instruction at location 0104 stores the A register in DDA as specified by locations 0105 and 0106. The remainder of the FR routine task is to increment DAH and DAL and to put the program in DA Mode. This is done in Steps 13B-2 thru 13B-4D of the FST routine.

STEP 13C-2
To perform the remaining tasks, the JMP (Jump) instruction at location 0107 transfers control to the instruction labled FRA at location 80E1 in the FST routine.

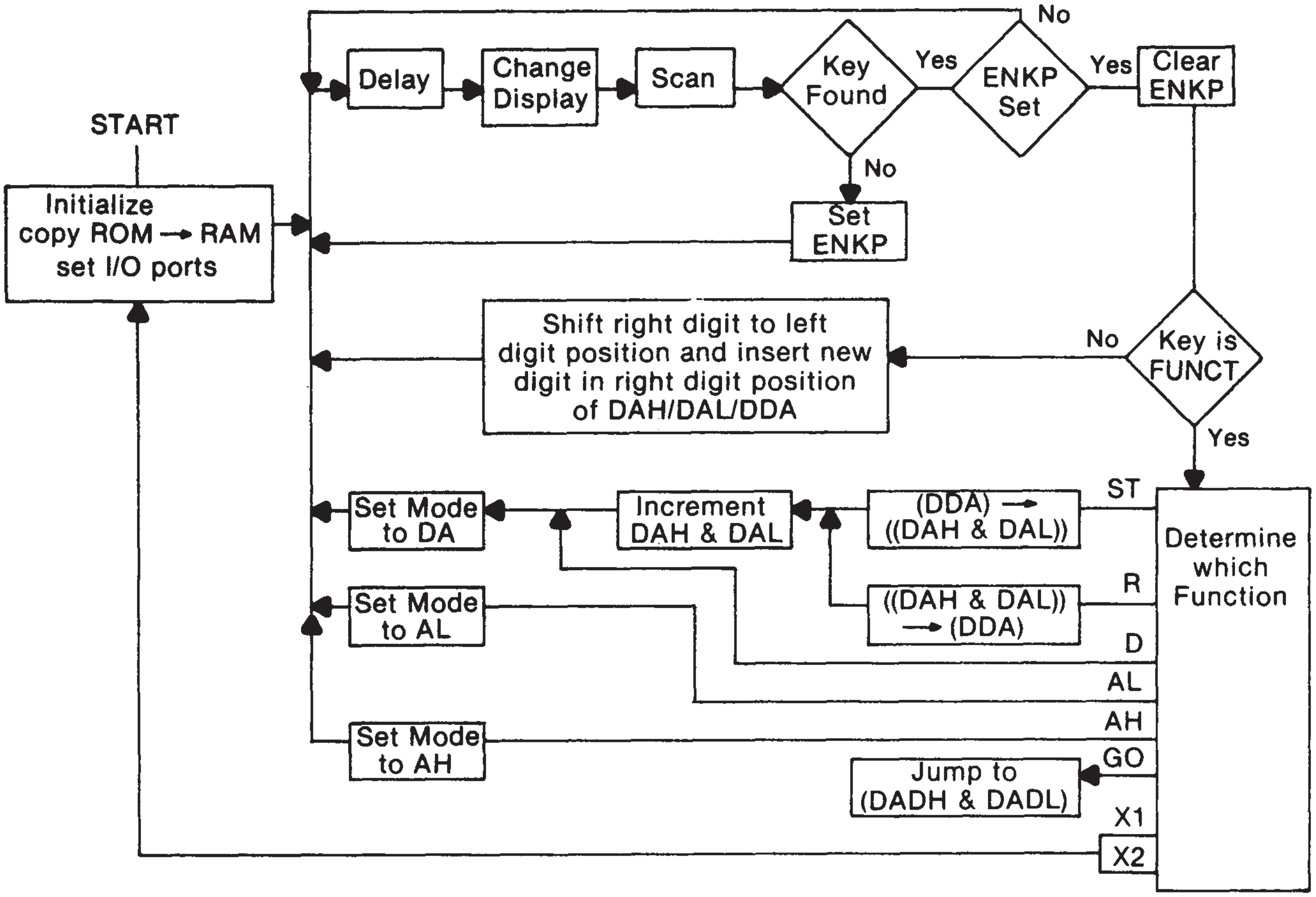
FUNCTION GO ROUTINE

STEP 13D-1
The XCHG (Exchange H-L with D-E) instruction at location 0100 moves DAH and DAL from the D-E register pair to the H-L register pair. The DAH and DAL bytes were placed in D-E by Steps 13A-3 and 13A-5 of the Function Found routine.

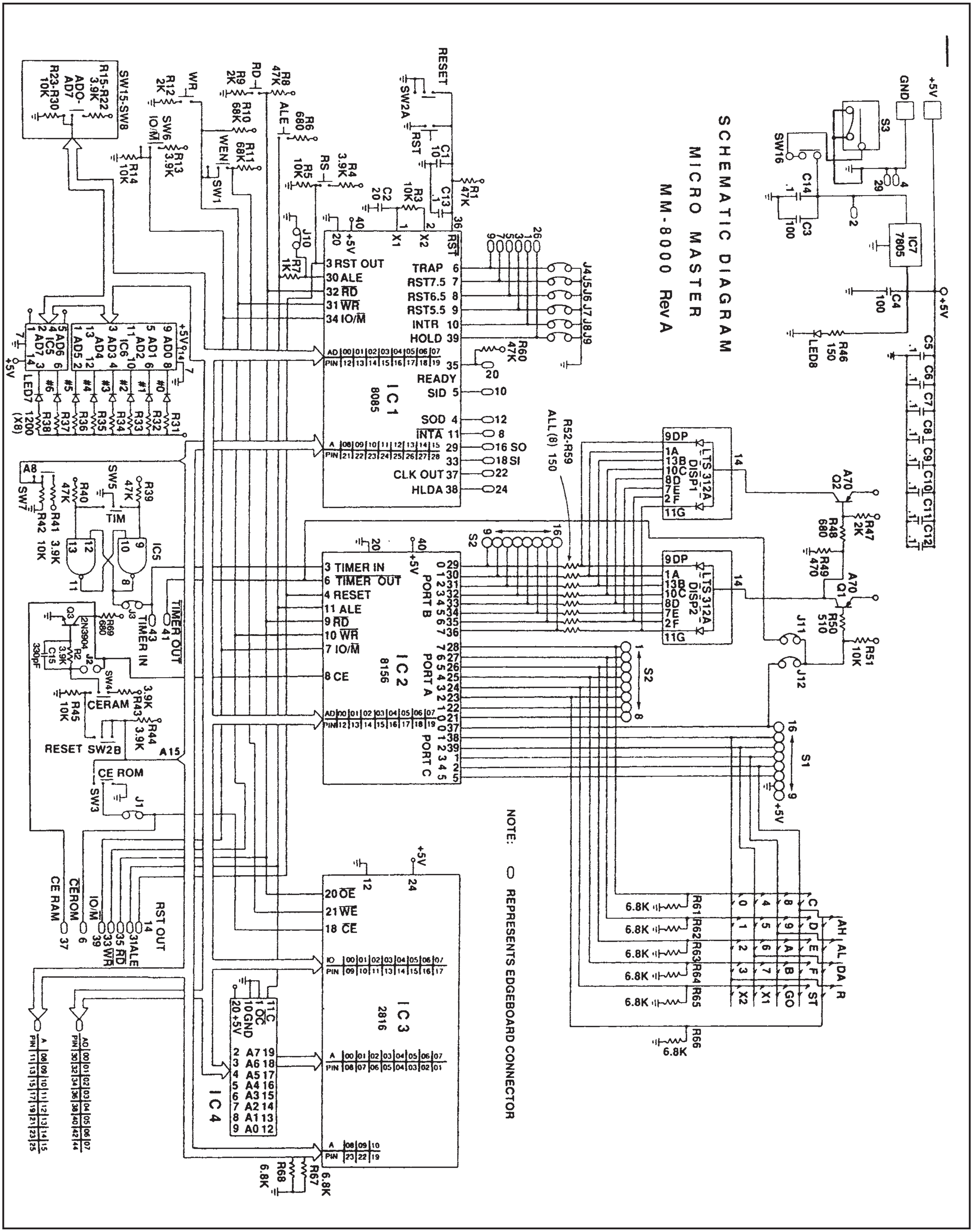
STEP 13D-2
The PCHL (Jump H and L indirect - move H and L to PC) instruction at location 0101 transfers the H-L register pair to the Program Counter. Control is therefore transferred to the instruction specified by DAH and DAL.

APPENDIX 3

Monitor Program Flow Chart

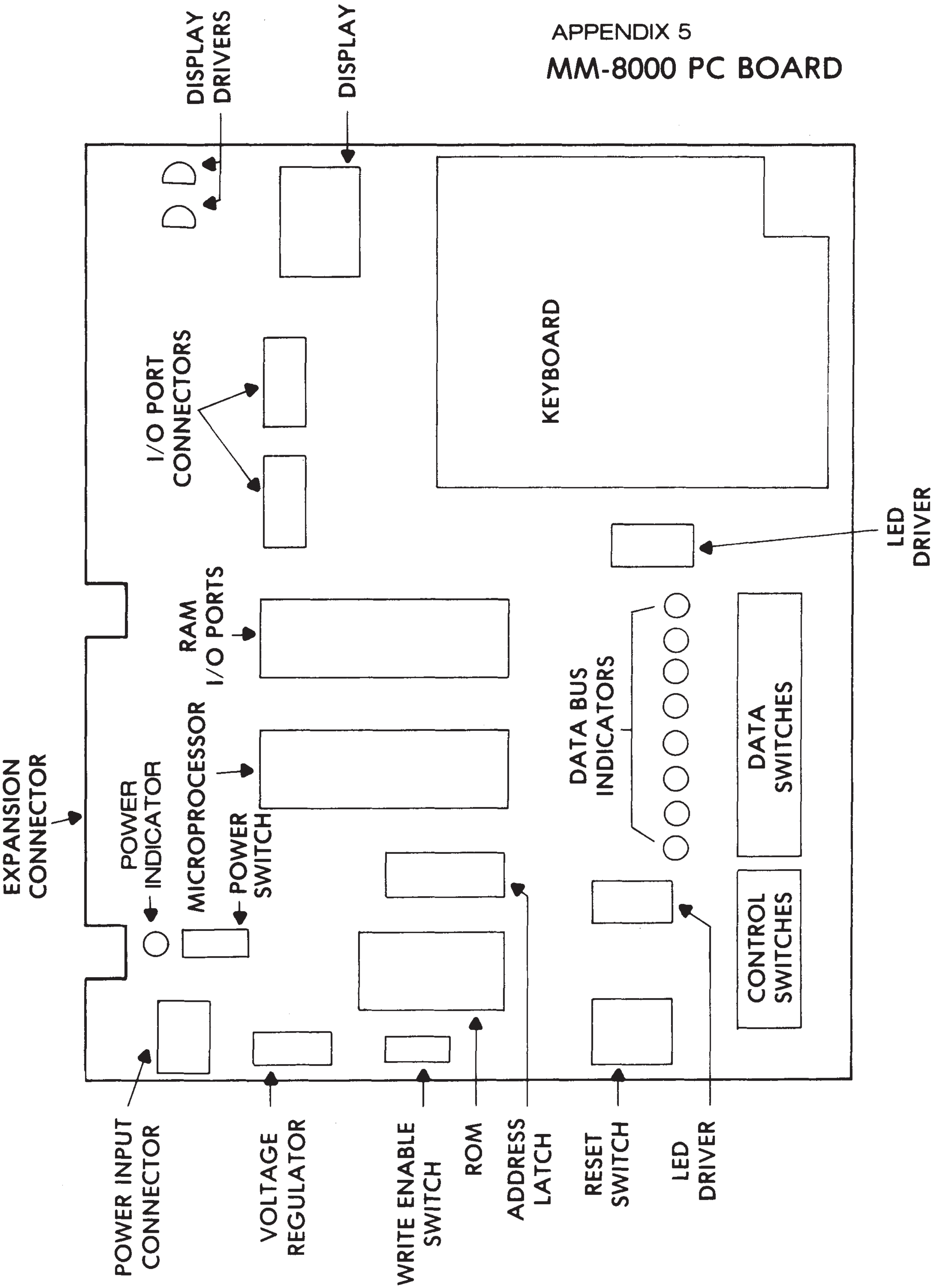


APPENDIX 4


SCHEMATIC DIAGRAM
MICRO MASTER
MM-8000 Rev A
+5V
GND
S3
SW16
C14 .1
C3 100
IC7 7805
C4 100
R46 150
LED8
C5 C6 C7 C8 C9 C10 C11 C12 .1
RESET
SW2A
RST
C1 10
C13 .1
R1 47K
R8 47K
RD
R9 2K
R10 68K
R11 68K
WEN
SW1
ALE
R6 680
RS
R4 3.9K
R5 10K
C2 20
R3 10K
WR
R12 2K
SW6 IO/M
R13 3.9K
R14 10K
SW15-SW8
R15-R22 3.9K
AD0-AD7
R23-R30 10K
J10
R7 1K
IC 1
8085
RST OUT
ALE
RD
WR
IO/M
TRAP
RST7.5
RST6.5
RST5.5
INTR
HOLD
READY
SID
SOD
INTA
CLK OUT
HLDA
16 SO
18 SI
J4 J5 J6 J7 J8 J9
R60 47K
IC5
IC6
LED7 1200 (X8)
R31-R38
A8
SW7
R42 10K
R41 3.9K
R40 47K
SW5
R39 47K
TIM
J3
TIMER OUT 41
TIMER IN 43
Q3 2N3904
R69 680
C15 330pF
R2 3.9K
J2
SW4
CERAM
R45 10K
R43 3.9K
R44 3.9K
RESET SW2B
A15
CE ROM
SW3
J1
IC 2
8156
TIMER IN
TIMER OUT
RESET
CE
PORT B
PORT A
PORT C
S2
R52-R59 ALL (8) 150
LTS 312A DISP1
LTS 312A DISP2
9DP 1A 13B 10C 8D 7E 2F 11G
A70 Q2
R47 2K
R48 680
R49 470
A70 Q1
R50 510
R51 10K
J11
J12
S1
C D E F ST
8 9 A B GO
4 5 6 7 X1
0 1 2 3 X2
AH AL DA R
R61 R62 R63 R64 R65 R66 6.8K
IC 3
2816
OE
WE
CE
IC 4
1C 1OC 10GND 20+5V
A7 A6 A5 A4 A3 A2 A1 A0
R67 6.8K
R68 6.8K
RST OUT 14
31ALE
35 RD
33 WR
IO/M 39 6
CEROM 6
CE RAM 37
NOTE: ○ REPRESENTS EDGEBOARD CONNECTOR

APPENDIX 5

MM-8000 PC BOARD



Elenco® Electronics, Inc.
150 Carpenter Avenue
Wheeling, IL 60090
(847) 541-3800
Website: www.elenco.com
e-mail: elenco@elenco.com