SDK-85 System Design Kit User's Manual

Manual Order Number 9800451B

SDK-85 System Design Kit User's Manual

Manual Order Number 9800451B

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

A164/0280/25K FL

CONTENTS

Figure 1-1. SDK-85 System Design Kit

CHAPTER 1
DESCRIPTION

The MCS-85 System Design Kit (SDK-85) contains all the parts with which you can build a complete 8085 microcomputer system on a single board, and a library of MCS-85 literature to help you learn to use it. The finished computer has the following built-in features:

- High-performance, 3-MHz 8085A cpu (1.3 μs instruction cycle)
- Popular 8080A Instruction Set
- Direct Teletypewriter Interface
- Interactive LED Display
- Large Wire-Wrap Area for Custom-Designed Circuit
- System Monitor Software in ROM

You can assemble the kit in as little as 3 to 5 hours, depending upon your skill and experience at building electronic kits. Only a 5 Volt power source capable of delivering 1.3 Amperes is then needed to make the computer operate, using its built-in display and keyboard. If you wish to interface a Teletypewriter to the SDK-85, you will also need a −10 Volt power supply. After you have completed the basic kit, you may expand both memory and I/O by adding more RAM-I/O or ROM-I/O devices in the spaces provided for that purpose. Other spaces are allocated for bus expansion drivers and buffers that allow you to address and use external devices located either in the wire-wrap area of the board or off the board. You can, for example, access up to 64K of external memory via the expansion bus.

SDK-85 SPECIFICATIONS

Central Processor

CPU: 8085A

Instruction Cycle: 1.3 microsecond

T_{cy}: 330 ns

Memory

ROM: 2K bytes (expandable to 4K bytes) 8355 or 8755

RAM: 256 bytes (expandable to 512 bytes) 8155

Addressing: ROM 0000-07FF (expandable to 0FFF with an additional 8355 or 8755) RAM 2000-20FF (2800-28FF available with an additional 8155)

Input/Output

Parallel: 38 lines (expandable to 76 lines).

Serial: Through SID/SOD ports of 8085. Software generated baud rate.

Baud Rate: 110

Interfaces

Bus: All signals TTL compatible.

Parallel I/O: All signals TTL compatible.

Serial I/O: 20 mA current loop TTY.

Note: By populating the buffer area of the board, you have access to all bus signals which enable you to design custom system expansions into the kit's wire-wrap area.

Interrupts

Three Levels: (RST 7.5) - Keyboard Interrupt
(RST 6.5) - TTL Input
(INTR) - TTL Input

DMA

Hold Request: Jumper selectable. TTL compatible input.

Software

System Monitor: Preprogrammed 8755 or 8355 ROM

Addresses: 0000-07FF

I/O: Keyboard/Display or TTY (serial I/O)

Literature

Design Library (Provided with kit):

- SDK-85 User's Manual
- Microcomputer Systems Databook
- MCS-85 User's Manual
- 8080/8085 Assembly Language Programming Manual

Physical Characteristics

Width: 12.0 in.

Height: 10 in.

Depth 0.50 in.

Weight: approx. 12 oz.

Electrical Characteristics (DC Power Required)

V_{CC}: +5V ± 5% 1.3A

V_{TTY}: −10V ± 10% 0.3A

(V_{TTY} required only if teletypewriter is to be connected to the kit)

Environmental

Operating Temperature: 0-55°C

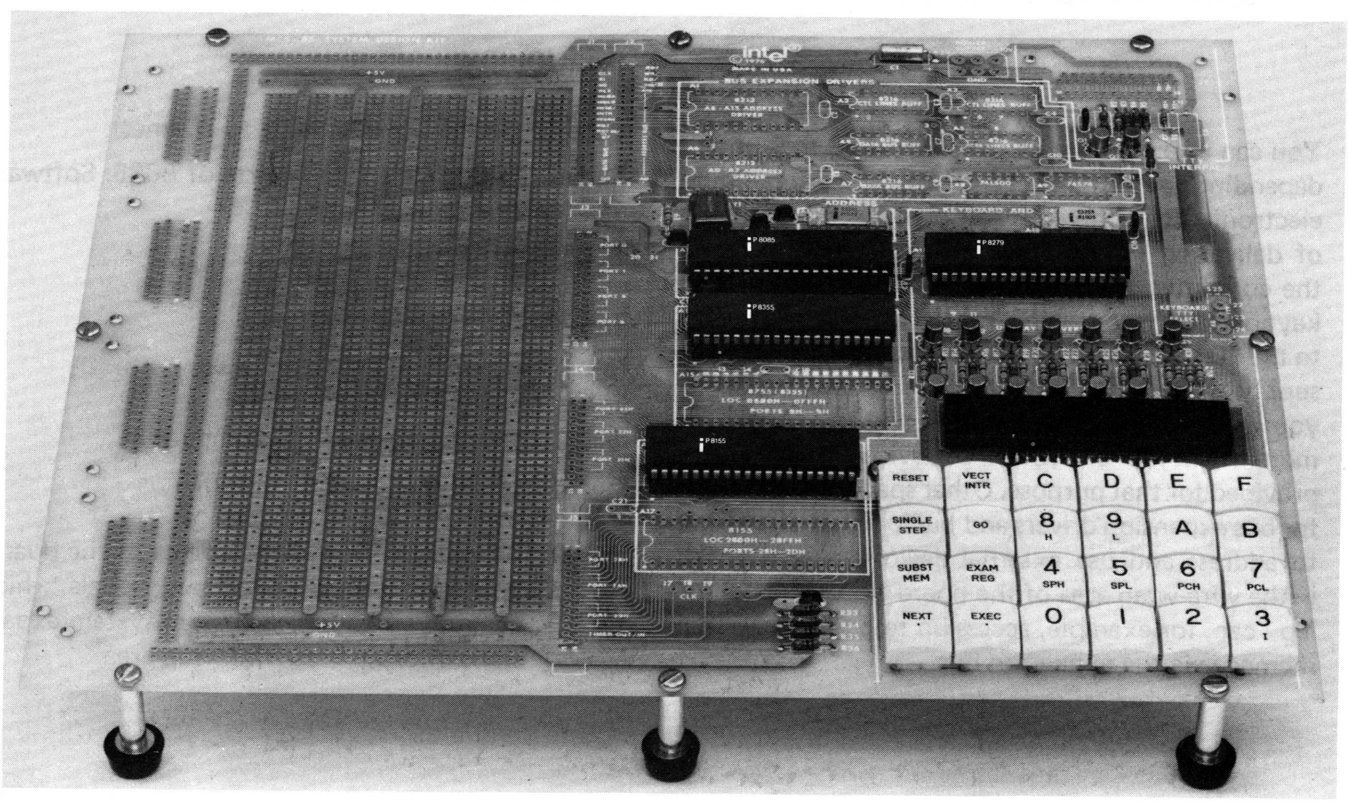


Figure 1-2. Finished Computer

CHAPTER 2
HOW TO ASSEMBLE THE KIT

2-1 GENERAL

Don't unpack your parts yet. Do a little reading first, and you may save yourself time and expense.

CAUTION

> The metal-oxide-semiconductor (MOS) devices in this kit are susceptible to static electricity. Do not remove them from the protective, black foam backing sheet until you have read the precautions and instructions in paragraph 2-4.

This manual was published only after the assembly of several kits by a number of persons of varying experience. In this chapter you will find virtually everything you need to know to put together your MCS-85 System Design Kit.

There are suggestions for laying out an efficient work area. All of the tools and materials you need are described in a checklist. There is a complete and detailed parts list. Basic assembly and soldering techniques are reviewed. Following the step-by-step assembly instructions in this chapter, you can't go wrong.

If you're an experienced kitbuilder, you already know that it's not a bad idea to read through this entire chapter first, before starting the job. That way, there won't be any surprises later. Take your time. Don't rush, and don't skip over quality-checking each step you perform. Desoldering, removing, and replacing just one DIP component because it was not oriented properly when first installed will cost you more time than double-checking **all** of them. Your objective is surely to produce a working computer, not to win a race.

2-2 GETTING ORGANIZED

Before starting work, it's a good idea to plan and organize your workplace. Be sure you have room to accommodate this book, lying open, and also the circuit board, along with tools and the hot soldering pencil. Unless you have the cordless, battery-powered soldering instrument, you'll want to arrange its cord out of the way to keep from accidentally pulling the soldering pencil off its holder. A muffin pan, an egg carton, or some small boxes could be used to sort parts into, if you don't have the traditional plastic, compartmented parts boxes. It might be helpful, too, to write the part values and reference designators on small cards as you sort them, and put these with the parts for quick identification. Arrange everything within comfortable reach, and you'll do the job quickly with little chance of errors.

2-3 SELECTING TOOLS AND MATERIALS

These tools and materials will be required to assemble the kit:

- ☐ Needle-nose pliers
- ☐ Small Phillips screwdriver
- ☐ Small diagonal cutters
- ☐ Soldering pencil, not more than 30 watts, with extra-small-diameter tip. (1/16 in. isn't too small.) You should also have a secure holder for it.
- ☐ Rosin-core solder, 60:40 (60% tin), small diameter (.05 in. or less) wire

  Note: Soldering paste is not needed. The solder will contain sufficient flux.

- ☐ Volt-Ohm-Milliammeter

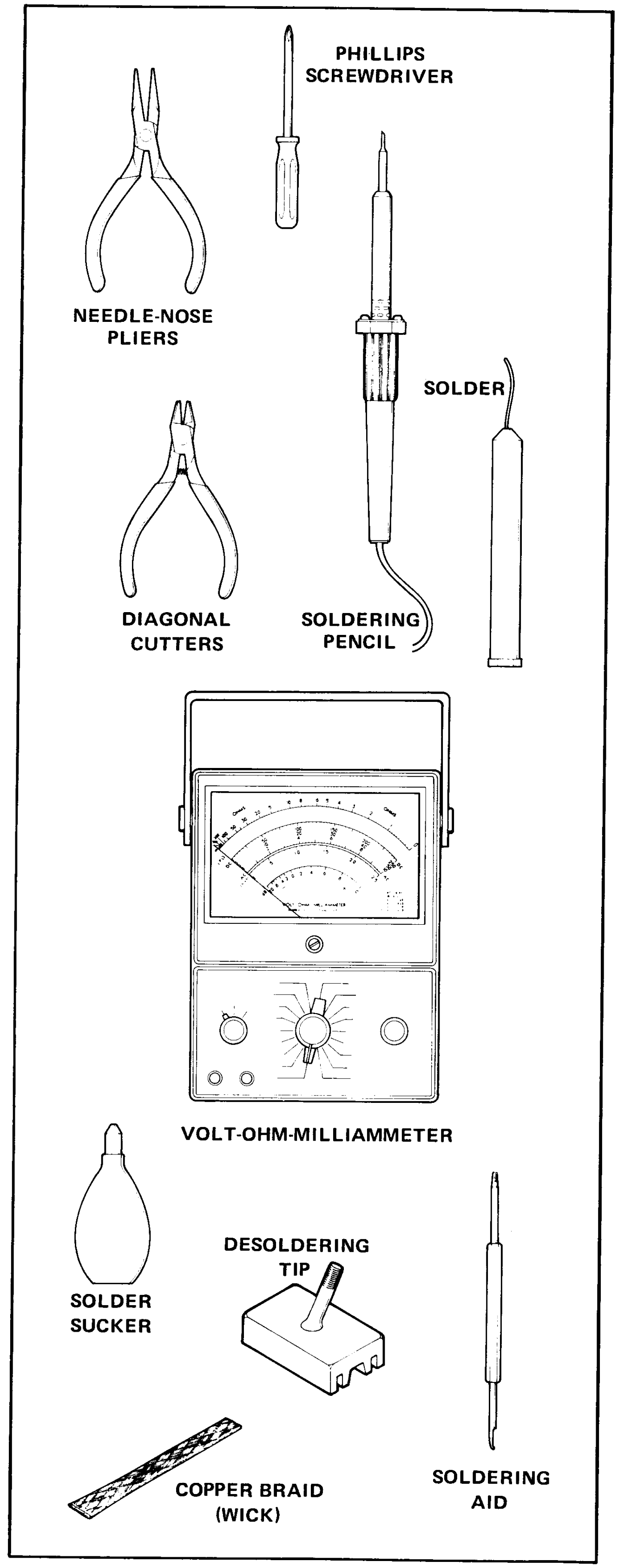
It is also useful to have the following:

- ☐ Soldering aid, with a small-tipped fork at one end and a reamer at the other, to help in coaxing component leads into holes and manipulating small parts.

If you should happen to make a soldering error and have to remove solder from joints, the job will be made much, much easier if you have the following:

- ☐ Solder sucking device, either the bulb variety (shown) or the pump variety
- ☐ Large-area desoldering tip for your soldering pencil, to spread heat over several leads of an IC device at the same time
- ☐ Length of copper braid to sop up solder like a sponge

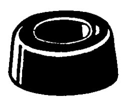
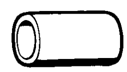
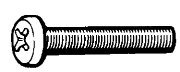
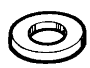
  Note: It is extremely difficult to remove DIP components using just a soldering pencil.



2-4 UNPACKING AND SORTING PARTS

The MCS-85 System Design Kit is shipped skin-packed on a card that includes a conductive backing to protect its metal-oxide-semiconductor (MOS) devices from static charge. Don't remove the four larger-size Intel devices from the foam backing until you have completed all of the instructions in this chapter and are ready to place them on the board. As a further protection against possible damage, these four devices are to be installed in sockets, rather than soldered on the board.

With a knife or sharp-pointed scissors, slit the film around the edges of the small-parts bags in the lower left corner of the skin-pack and remove them. First, open the bag of hardware and check to be sure you have:

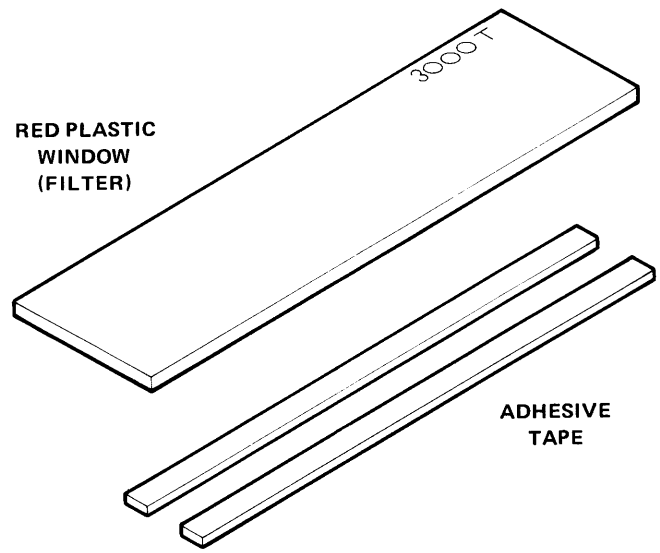
- ☐ 9 rubber feet 
- ☐ 9 Nylon spacers, 7/16 in. long 
- ☐ 9 screws, 3/4 in. long 
- ☐ 18 Nylon washers 
- ☐ 9 nuts 

CAUTION

Don't remove the other components from the skin-pack. The black foam backing is an electrically conductive material that protects the integrated-circuit devices from static electricity as well as from physical damage to their leads and ceramic substrates.

Underneath the two bags of small parts and hardware will be found:

- ☐ Red plastic window (covered with protective paper)
- ☐ Two strips of double-coated adhesive tape



Next, open the bag of electrical parts and sort them out by type and value. Give yourself plenty of unobstructed work space and try not to let tiny parts skitter away from you. The bag should yield the following:

Resistors, 1/4 Watt

□	8	24 Ohm (red-yellow-black)	R11, 14, 17, 20, 23, 26, 27, 30
□	1	47 Ohm (yellow-violet-black)	R5
□	1	200 Ohm (red-black-brown)	R33
□	6	270 Ohm (red-violet-brown)	R10, 13, 16, 19, 22, 25
□	2	1k (1,000) Ohm (brown-black-red)	R4, 31
□	1	1.6k Ohm (brown-blue-red)	R3
□	1	2.7k Ohm (red-violet-red)	R6
□	9	3k Ohm (orange-black-red)	R7, 9, 12, 15, 18, 21, 24, 28, 29
□	1	3.9k Ohm (orange-white-red)	R8
□	1	4.7k Ohm (yellow-violet-red)	R2
□	1	51k Ohm (green-brown-orange)	R32

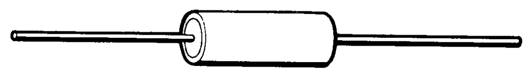
Resistor, 1/2 Watt

□	1	100 Ohm (brown-black-brown)	R1

Resistors, 1 Watt

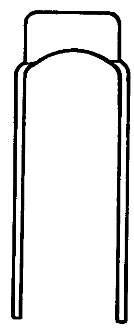
□	1	200 Ohm (red-black-brown)	R34
□	1	430 Ohm (yellow-orange-brown)	R35

Capacitor, tantalum



□	1	22 μf, 15V	C1

Capacitor, mono



□	2	1 μf, 25V	C5, 20

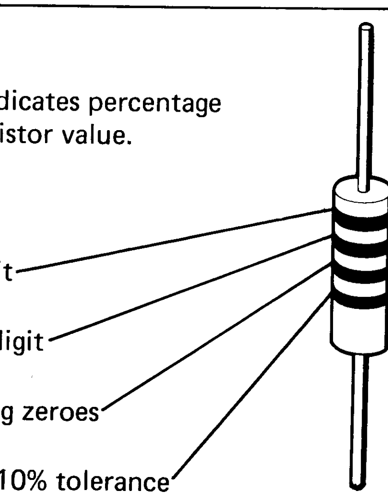
Resistor Color Code

Resistors are commonly identified by means of a code using color bands. Each color represents a number.

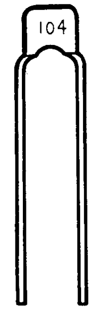
The first three bands employ the color code below:

Black	0	Green	5
Brown	1	Blue	6
Red	2	Violet	7
Orange	3	Gray	8
Yellow	4	White	9

The fourth band indicates percentage tolerance of the resistor value.

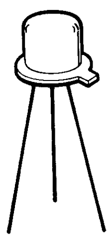


Capacitor, ceramic



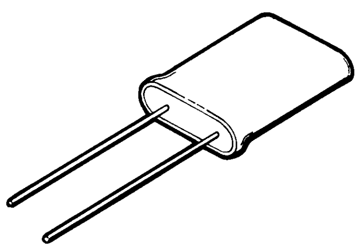
☐ 7 0.1 μf C11-16, 18

Transistor



☐ 16 2N2907 transistors Q1-16

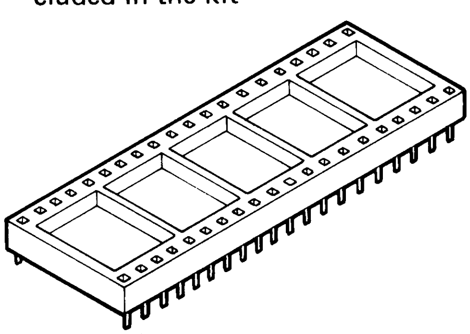
Crystal, clock



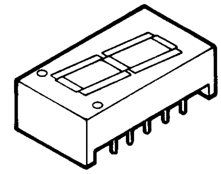
☐ 1 6.144 MHz Y1

Besides the small-parts bags, the skin-pack contains:

☐ 4 40-pin DIP (dual in-line package) sockets for the four large integrated circuits included in the kit

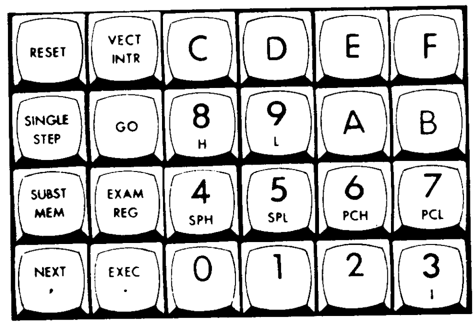


☐ 6 alphanumeric LED (light-emitting diode) displays



DS1-6

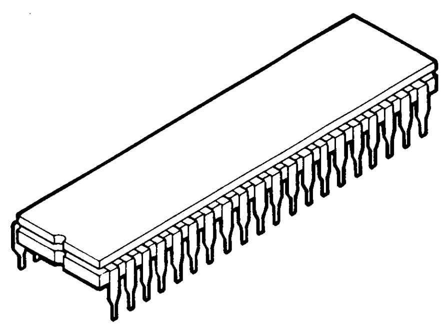
☐ 24 pushbutton switches, with keycaps labeled



S1-24

Note: It's a good idea to check all switches with the ohmmeter before installing. If one is bad, you'll save a lot of work.

Large, 40-pin ICs (integrated circuits)



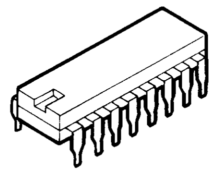
☐ 1 8085A microprocessor (cpu) A11

☐ 1 8355 (or 8755) ROM (read-only memory) with I/O (input/output) ports A14

☐ 1 8155 RAM (random-access, read-write memory) with I/O ports and timer A16

☐ 1 8279 keyboard/display interface A13

Small, 16-pin ICs



☐	1	8205 address decoder	A10
☐	1	74LS156 scan decoder	A12

CAUTION

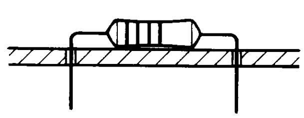
Large-scale integrated circuits are fragile! Dropping, twisting, or uneven pressure may break them. The discharge of static electricity can destroy them internally. Leave them embedded in the conductive-foam backing sheet until ready to install on the board. Never press down hard upon, twist, or bend the larger devices. Touch the exposed metal traces of the board with your hand before inserting one in its socket. The soldering of large devices directly on the circuit board is not recommended.

2-5 A REVIEW OF BASIC ASSEMBLY AND SOLDERING TECHNIQUES

The steps to producing a professional quality assembled circuit board are:

1. Have your work area organized before starting work, and keep it that way. (See paragraph 2-2.) Sort all parts into bins, cups, trays or boxes so they will be easily located by value when needed.

2. To prepare a part for soldering, bend its leads carefully with needle-nose pliers to make the part fit exactly the way you want it to.



It is good practice to orient color-coded resistors so that the codes are readily read, top-to-bottom or left-to-right, and to form the leads of parts with values printed on them so that the values are legible after assembly.

3. Fit each part in place and see that no undue stress is placed on the leads. Double-check and be sure you have the correct part inserted in the correct holes, properly oriented. Don't trim leads before soldering.

4. When ready to solder, be sure your soldering pencil is hot enough to melt solder quickly. Then turn the board face-down on your work surface. If necessary, hold the parts you are about to solder in place while turning it over so they won't fall out, and place something under the board to hold the parts in position while you solder on the back surface of the board. Some people prefer to crimp the leads to hold the parts in place. That's all right, too.

5. Bring the point of your soldering pencil into contact with the pad to be soldered, simultaneously also touching the lead.

6. At once, touch the end of the solder wire to the pad and lead, opposite the pencil tip. The amount of time required to melt the solder will depend upon the amount of foil surface there is on the board to carry away heat by conduction. The smallest pads will heat up in less than a second with a 25- or 30-watt pencil; large, ground-plane areas may require over five seconds.

7. The instant you see and feel the solder start to melt, withdraw the solder wire from the joint. Only a tiny drop of solder is needed to make a good joint.

8. The instant you see the solder draw into the hole, become shiny, and spread smoothly over the surface of both pad and lead, withdraw the soldering pencil. It will take only a moment for this to happen after step 7.

9. Don't reheat a joint unless there's something wrong with it: not enough solder, too much solder (causing a "bridge" to an adjacent pad or trace), or a "cold solder joint," which

appears dull on the surface or does not surround the lead completely and fill the hole.

Note: A little rosin from the solder core, remaining on the board, does no harm. Don't try to clean it off.

10. Clip off the excess length of lead that projects beyond the solder "bead," within 1/8 inch of the board. Save cut ends to use for strapping optional connections. (See paragraph 3-2.)

WARNING

Avoid eye injury when clipping excess lead ends. Hold lead end as you clip it, so it can't fly up in your face.

There are two important conditions that govern good soldering technique. They are:

1. Use no more heat than absolutely the minimum that will make a solid joint.
2. Use enough heat to cause solder to flow into the hole in the board and around the lead that's being soldered into it.

These conditions are both met simultaneously and easily only if you are careful, have the proper tools, and arrange your workplace so that the circuit board can lie flat while you apply steady, firm (but not hard) pressure with the soldering pencil without slipping. A small-diameter soldering tip is a **must!** Likewise, small-diameter solder wire is essential to achieving satisfactory results.

Note: Do not apply soldering paste to the work. Fluxing is not required in printed-circuit soldering, as the boards and component leads are plated or tinned to prevent oxidation of the copper.

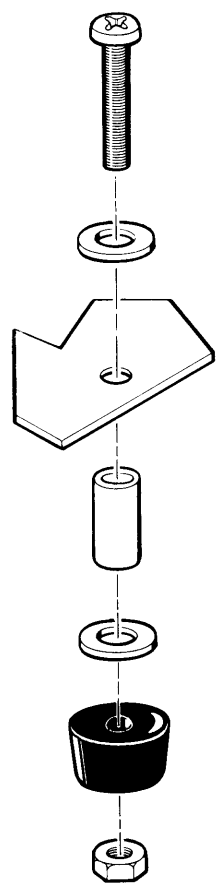
Always inspect carefully for cold solder joints, solder bridges, or (perish the thought!) lifted traces after each soldering operation. A good way to check for solder bridges is to hold the newly-soldered connection up to a light. If you can't see light between the soldered pad and any adjacent pads or traces that aren't supposed to be connected to it, it might be well to slip a solder-sucker or wick over the lead under examination, quickly remelt the solder and draw off the excess.

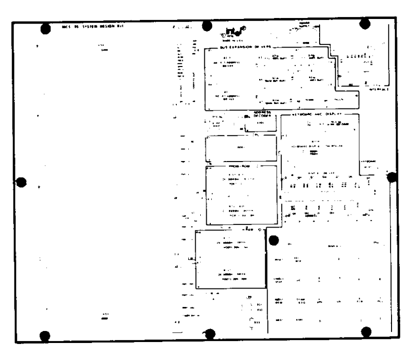
2-6 ASSEMBLY PROCEDURE

Follow these instructions in order and make a check mark in the box opposite each step when it is completed.

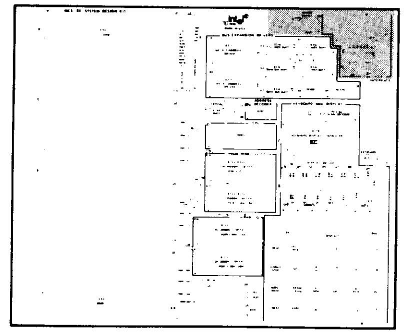
☐ First, place the board on your work surface, lettered side up.

☐ Install the nine rubber feet. Eight go around the edge of the board, and one goes near the middle of the board, to the left of the keyboard and display area. At each location, press a nut into the recess in a rubber foot, string a washer on a screw, and insert the screw through the hole in the board from the top. Place a spacer, then another washer on the screw, then place the nut and foot on the end of the screw, and tighten, with the screwdriver, just enough to hold the foot firmly.





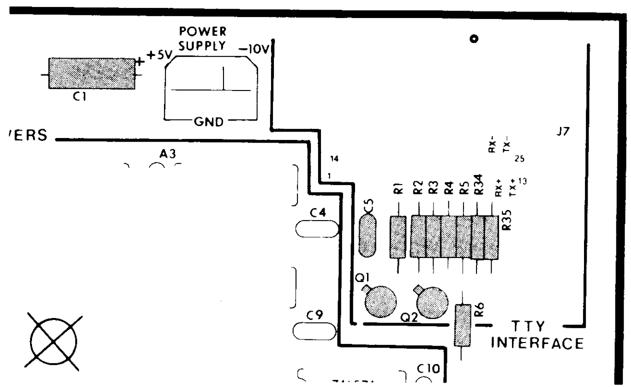
- ☐ Install capacitor C1 near the top edge of the board.
- ☐ Solder C1 in place. Clip excess lead ends.



WARNING

Avoid eye injury. Hold lead ends as you clip them so they can't fly up at you.

Assembly of TTY Interface Area—



- ☐ Install a 100 Ohm, 1/2 Watt resistor (brown-black-brown) at R1.
- ☐ Install a 4.7k Ohm resistor (yellow-violet-red) at R2.
- ☐ Install a 1.6k Ohm resistor (brown-blue-red) at R3.
- ☐ Install a 1k Ohm resistor (brown-black-red) at R4.
- ☐ Install a 47 Ohm resistor (yellow-violet-black) at R5.
- ☐ Install a 2.7k Ohm resistor (red-violet-red) at R6.
- ☐ Solder the six resistors in place, then clip their excess lead ends.
- ☐ Install a 1 uf capacitor at C5, and solder and clip it.
- ☐ Install a 200 Ohm, 1 Watt resistor (red-black-brown) at R34.
- ☐ Install a 430 Ohm resistor (yellow-orange-brown) at R35.
- ☐ Solder these two resistors in place, then clip their excess lead ends.
- ☐ Install transistors Q1 and Q2, and solder and clip them.

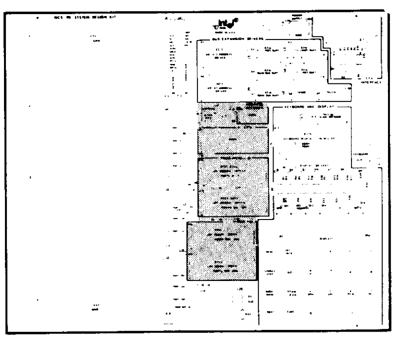
Assembly of Processing Area

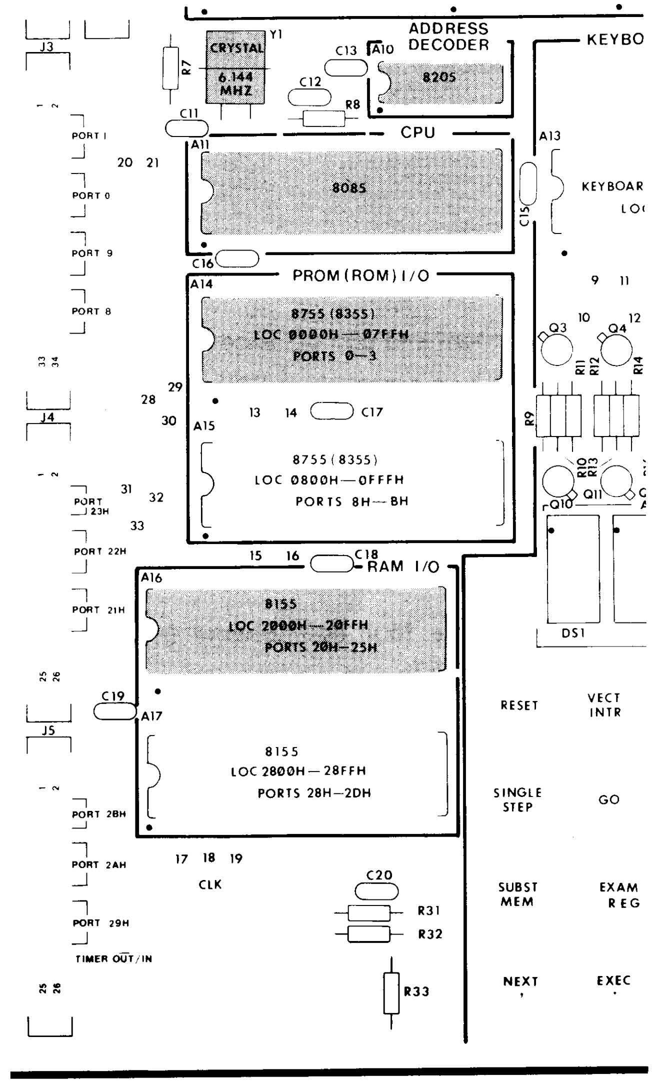
The processing area includes the clock crystal, address decoder, cpu, RAM-I/O and ROM-I/O areas, and related components.

- ☐ Install the crystal at Y1, with its leads bent so that the device lies flat on the board in the space outlined for it.
- ☐ Take a piece of scrap wire trimmed from a component previously mounted on the board. Bend it into the shape of a staple. Install it over the crystal, to hold it firmly in place.
- ☐ Solder the four connections just made.
- ☐ Install the 8205 address decoder at A10 and solder it.

Install three DIP sockets, crimping the corner leads of each to hold in place, at:

- ☐ A11, for the 8085 cpu.
- ☐ A14, for the PROM (ROM)-I/O device, an 8755 or 8355.
- ☐ A16, for the RAM-I/O device, an 8155.
- ☐ Solder the three sockets in, and check carefully for solder bridges.





☐ Install a 3k Ohm resistor (orange-black-red) at R7.

☐ Install a 3.9k Ohm resistor (orange-white-red) at R8.

☐ Solder these two resistors and clip off their lead ends.

Install three 0.1 uf ceramic capacitors at:

☐ C11

☐ C12

☐ C13

☐ Solder them and clip off excess lead length.

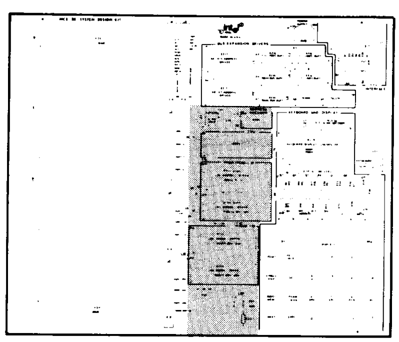
☐ Install a 1 uf capacitor at C20.

☐ Install a 1k resistor (brown-black-red) at R31.

☐ Install a 51k resistor (green-brown-orange) at R32.

☐ Install a 200 Ohm resistor (red-black-brown) at R33.

☐ Solder these four components in place and trim their leads.

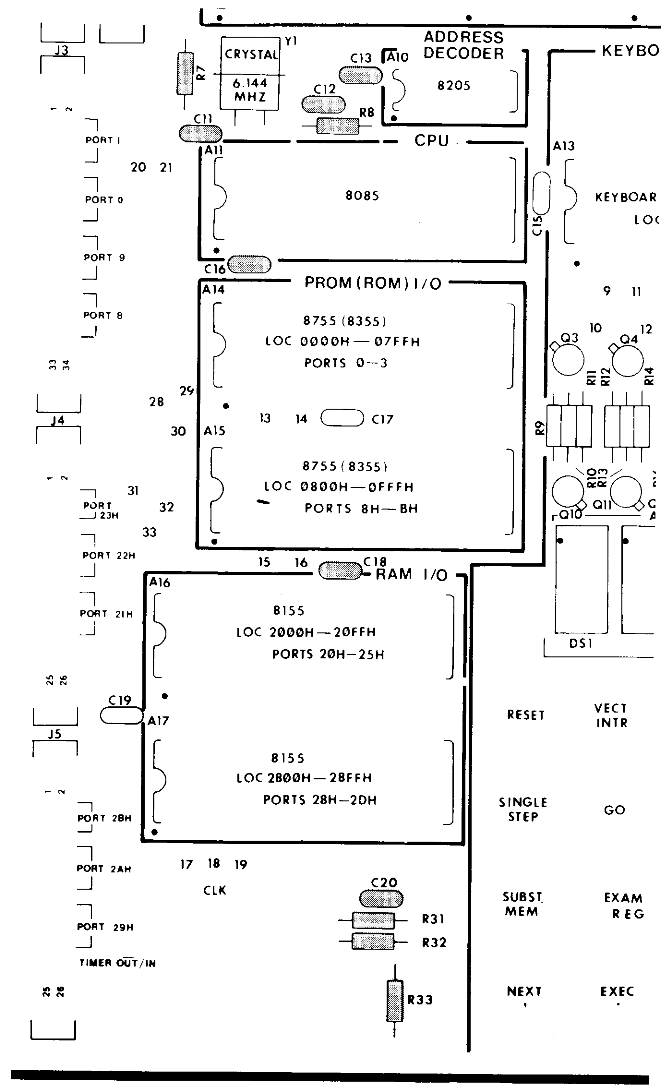


Install 0.1 uf ceramic capacitors at:

☐ C16

☐ C18

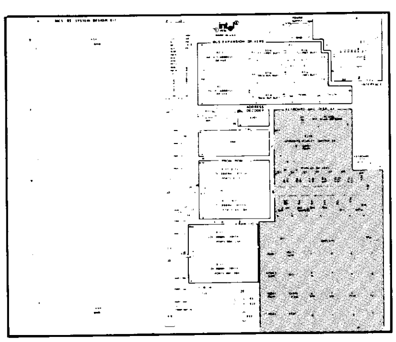
☐ Now solder the capacitors you have installed, and clip off their excess lead ends.

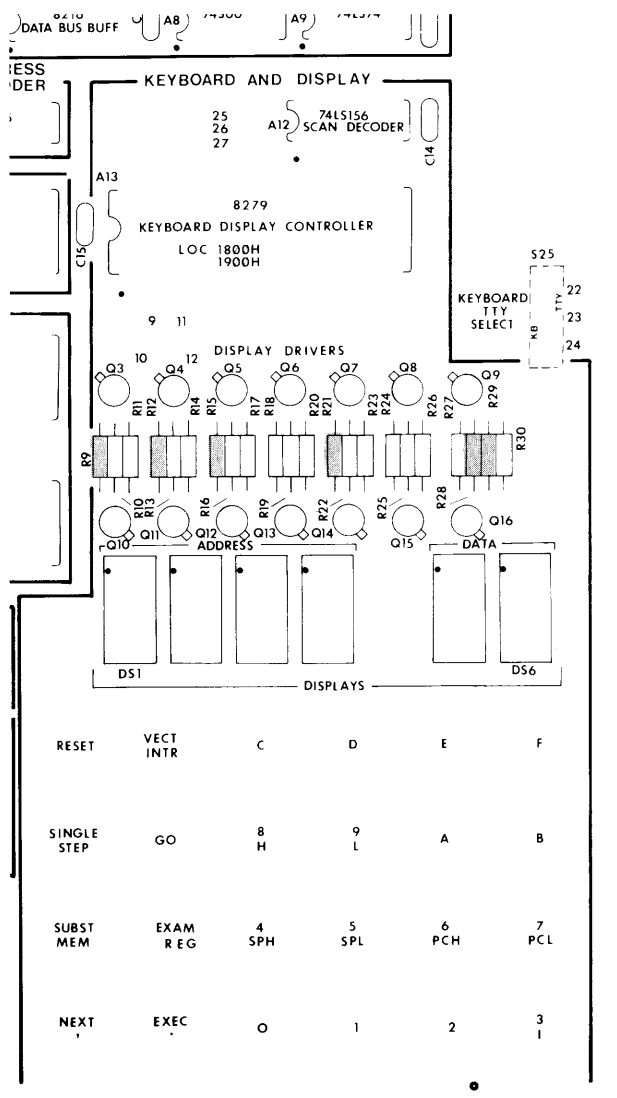


Assembly of Keyboard and Display Area

Find where the row of resistors, R9 through R30, go. Install eight 3k resistors (orange-black-red) at:

- ☐ R9
- ☐ R12
- ☐ R15
- ☐ R18
- ☐ R21
- ☐ R24
- ☐ R28 (Careful—the location pattern changes here!)
- ☐ R29
- ☐ Now solder all eight resistors in place and clip their excess lead ends.





Install six 270 Ohm resistors (red-violet-brown) at:

☐ R10

☐ R13

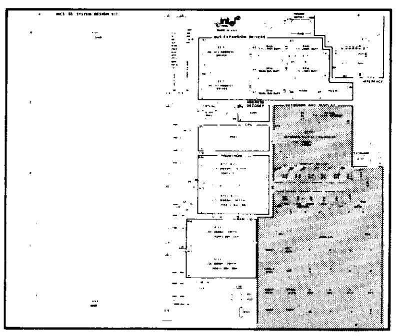
☐ R16

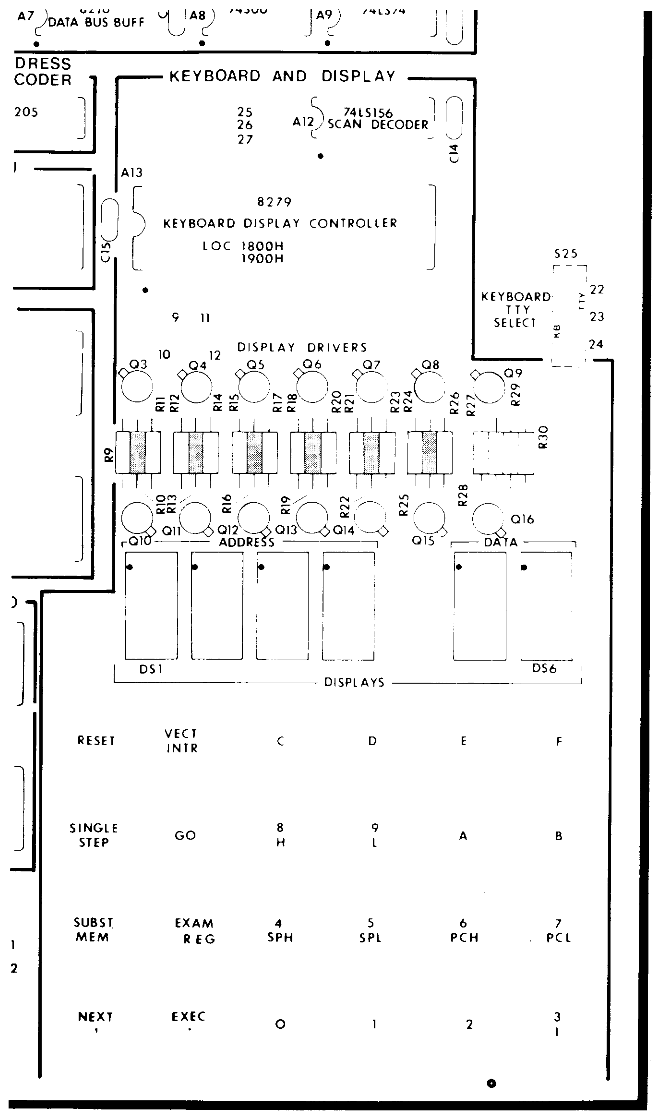
☐ R19

☐ R22

☐ R25

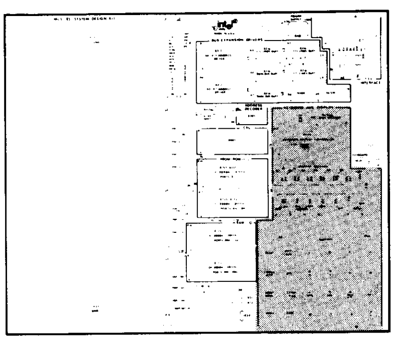
☐ Solder these six resistors and clip their excess lead ends.

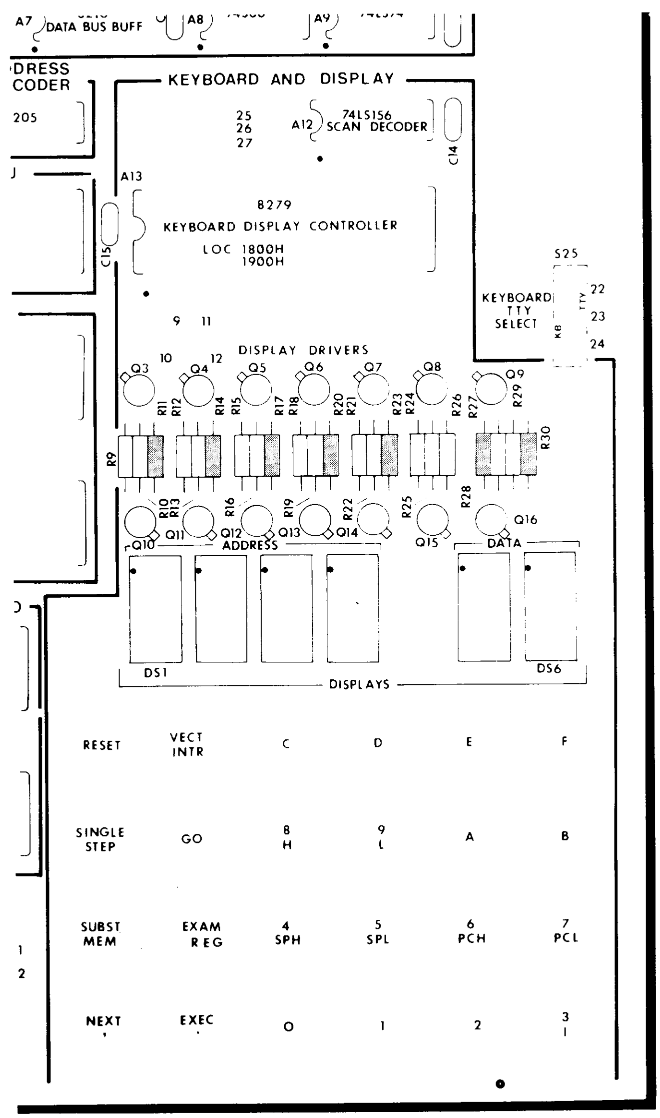




Install eight 24 Ohm resistors (red-yellow-black) at:

- ☐ R11
- ☐ R14
- ☐ R17
- ☐ R20
- ☐ R23
- ☐ R26
- ☐ R27 (Again, note the change in location pattern.)
- ☐ R30
- ☐ Solder these eight resistors and clip their excess lead ends.



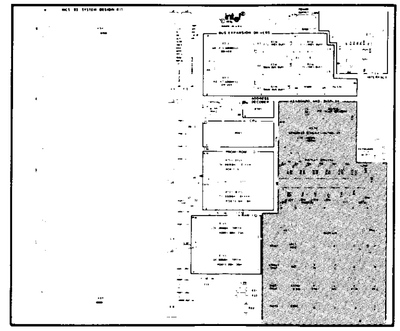


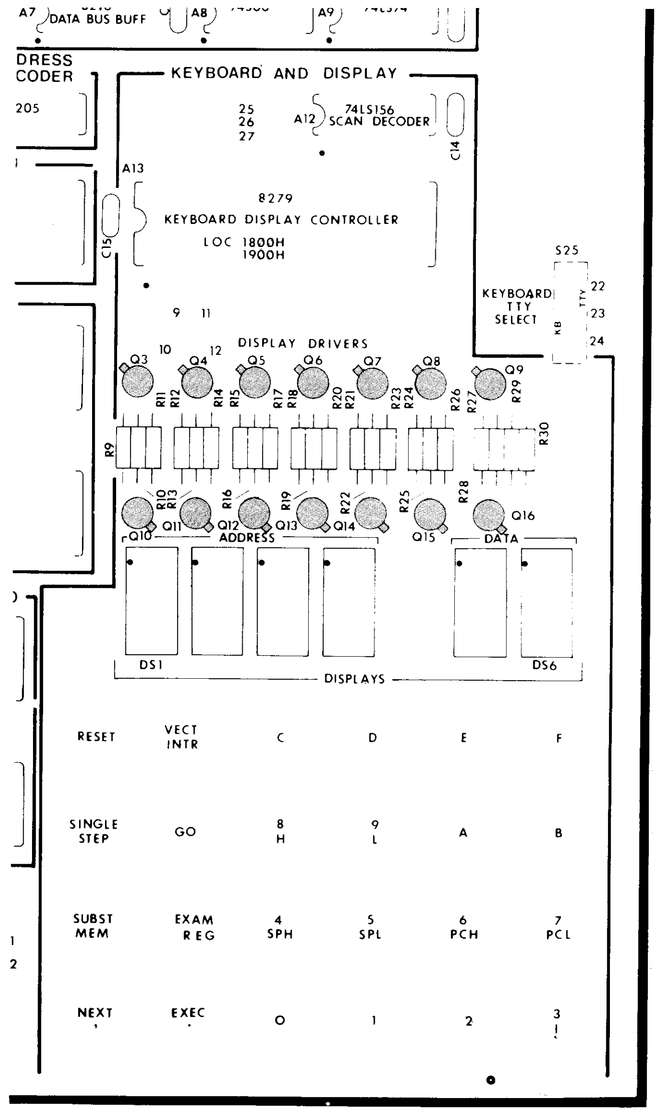
Install fourteen 2N2907 transistors in two rows. Position the seven transistors in the top row so that their indexing tabs point upward and to the left, at:

- ☐ Q3
- ☐ Q4
- ☐ Q5
- ☐ Q6
- ☐ Q7
- ☐ Q8
- ☐ Q9

Position the seven transistors in the bottom row so that their indexing tabs point down and to the right, at:

- ☐ Q10
- ☐ Q11
- ☐ Q12
- ☐ Q13
- ☐ Q14
- ☐ Q15
- ☐ Q16
- ☐ Press all of the transistors down to about 1/8 inch from the surface of the board. Let them stand approximately straight up. Then, turn the board over and solder all of their leads in place and trim the lead ends.





□ Install one of the 40-pin DIP sockets, for the 8279 Keyboard-Display Controller, at A13, and solder it in.

□ Install the 74LS156 scan decoder at A12, and solder it.

Be careful to orient the six alphanumeric LED displays so that the decimal points are even with the **bottom** of the digits and install at:

□ DS1

□ DS2

□ DS3

□ DS4

□ DS5

□ DS6

Note: If these components are provided with long, wirewrap leads, you will probably find it easiest to insert, solder, and clip them one at a time because of crowded quarters. The order shown above with the board turned bottomside up will be most convenient for you if you hold the soldering pencil in your left hand. If you solder right-handed, you may prefer to work from DS6 to DS1.

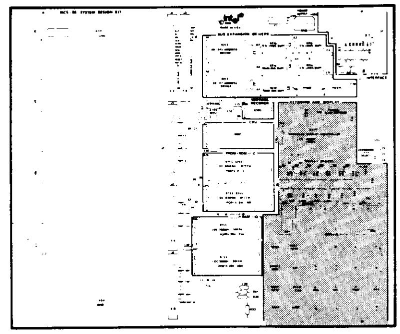
Note: Don't install the red filter over the display yet. It's a good idea to wait until after final assembly and checkout to do this, on the remote chance that you might have to remove one of the character displays.

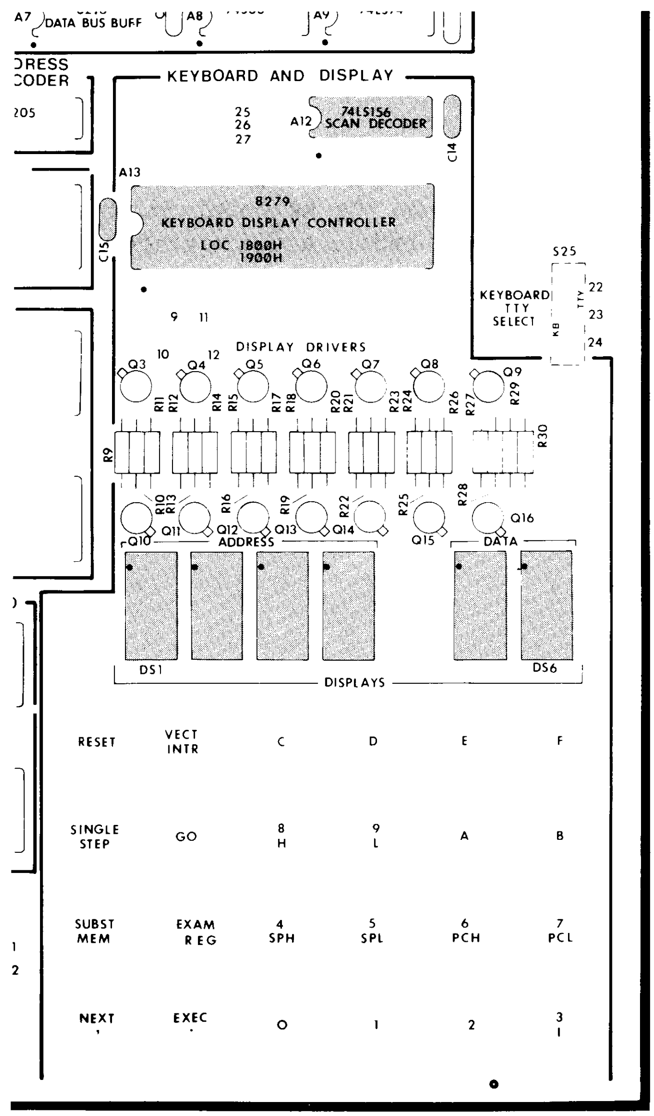
Install two 0.1 uf ceramic capacitors at:

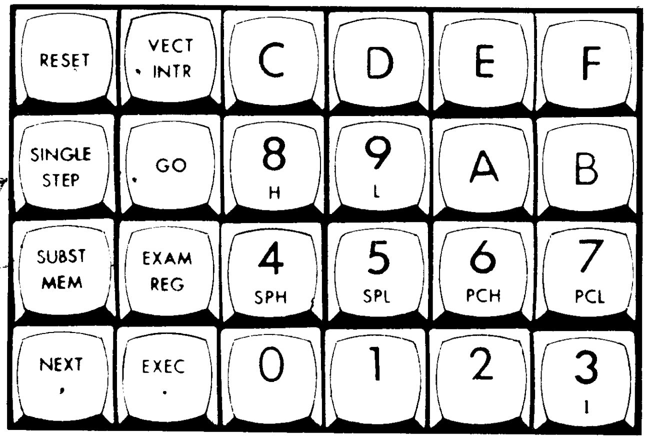
□ C14

□ C15

□ Solder the leads and clip them off close to the board.







☐ Install the twenty-four pushbutton switches that make up the keyboard. Be sure each button is rightside up and in its proper position before soldering.

The easiest method of doing this is to insert each button in its turn, bend its leads over on the back of the board to hold it in place, and go on until all buttons are in place, then solder all of them in one pass, with the board lying flat on the work surface and weighted down to make sure the switches are uniformly held firmly against the front surface of the board.

☐ RESET	☐ VECT INTR	☐ C	☐ D	☐ E	☐ F
☐ SINGLE STEP	☐ GO	☐ 8 H	☐ 9 L	☐ A	☐ B
☐ SUBST MEM	☐ EXAM REG	☐ 4 SPH	☐ 5 SPL	☐ 6 PCH	☐ 7 PCL
☐ NEXT ,	☐ EXEC .	☐ 0	☐ 1	☐ 2	☐ 3 I

☐ All soldered in place

CHAPTER 3
FINAL ASSEMBLY AND CHECKOUT

3-1 GENERAL

Now that most of the components are soldered on your circuit board, it's time to give your handiwork a quick visual check to make sure all of the devices are oriented correctly. The notched ends of the ICs should all be toward your left, and the decimal points of the LED displays should be at the bottom line of the characters.

It is recommended that the basic kit computer be checked out using the procedure in this chapter before adding any external options such as teletypewriter or expansion memory. It is well for you to have the assurance that you have a working cpu and display-keyboard before you add peripherals to your system. It is therefore recommended that you first wire the strapping options in Table 3-1 for the 8355 (or 8755) ROM-I/O that was furnished with the kit (and contains the SDK-85 System Monitor). Then install the strap in Table 3-2 for keyboard operation, and in Table 3-4 for the basic kit without expansion memory. (See paragraph 3-2.)

Paragraph 3-3 tells you how to hook up power to the MCS-85 System Design Kit, and paragraph 3-4 tells you how to start it up and see if it's working right. The subsequent paragraphs list the add-on options you can use without inventing any new circuitry on the board or off.

3-2 STRAPPING INSTRUCTIONS

The MCS-85 System Design Kit will accept 8355 or 8755 ROM-I/O devices at positions A14 and A15. These different devices are not completely electrically interchangeable, so you must make the strapping connections in Table 3-1, appropriate to the type of device in each socket.

To make a strapping connection (jumper), bend a short length of bare wire (such as the excess lead end cut from a resistor) to fit between the two holes you wish to strap together, insert the ends of the wire in the holes, and solder them. Then clip the remaining excess ends, just as you did with the components. When you install a jumper and solder it, be sure it doesn't touch any intervening traces or pads.

IMPORTANT: For normal operation of the SDK-85, it is *mandatory* to strap the following:

1. One of the three options in Table 3-1.
2. One of the two options in Table 3-2.
3. The two jumpers listed in Table 3-3.
4. Either basic kit operation or one of several expansion options listed in Table 3-4.

The keyboard-teletypewriter selection function may be done with a miniature printed circuit-board mount, single-pole, double-throw switch, S25, not furnished in the kit, or may be strapped with wire. Table 3-2 lists the connections. Table 3-3 lists keyboard strapping connections always made.

Tables 3-5 through 3-10 list all of the bus and port expansion connector pinouts. Table 3-11 lists suggested connector types.

3-3 POWER SUPPLY WIRING (See Figure 3-6.)

Connect a +5 Volt, regulated power supply with its positive output at the +5V POWER SUPPLY point on the board. A 6-pin Molex connector will fit the

(Text continues on page 3-4.)

TABLE 3-1
ROM/PROM STRAPPING

<table>
<tr><th>Device Location</th><th>8355 Figure 3-1</th><th>8755 Figure 3-2a</th><th>8755A Figure 3-2b</th></tr>
<tr><td>A14 (The SDK-85 Monitor ROM)</td><td rowspan="2">No Straps Required</td><td>Strap 28-29</td><td>Strap 29-30</td></tr>
<tr><td>A15</td><td>Strap 31-32</td><td>Strap 32-33</td></tr>
</table>

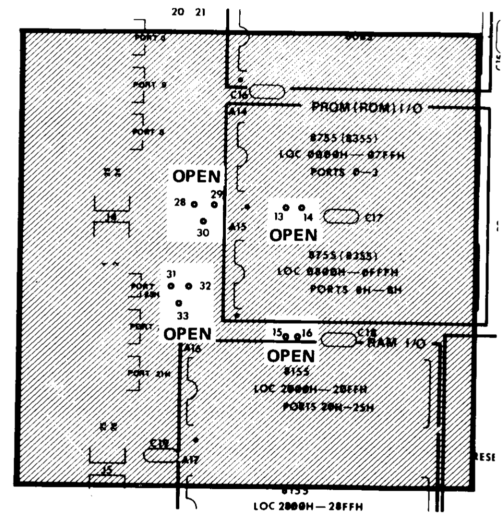


Figure 3-1 No Strapping Required for 8355 ROMs

TABLE 3-2
TELETYPEWRITER-KEYBOARD STRAPPING

TELETYPEWRITER Figure 3-3	KEYBOARD Figure 3-4
Strap 22-23	Strap 23-24

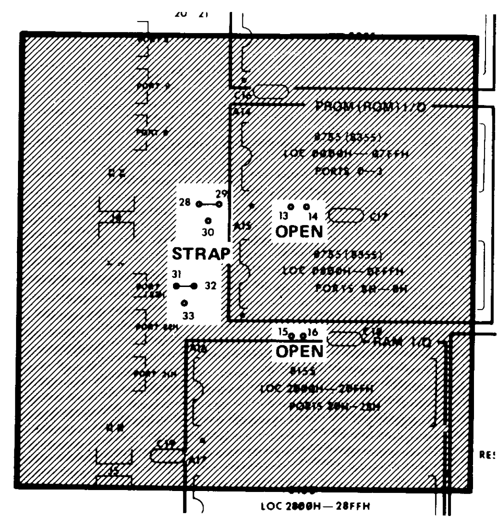


Figure 3-2a Strapping Connections for 8755 PROMS

TABLE 3-3
DISABLING UNUSED KEYBOARD CONTROLLER FUNCTIONS

Figure 3-5
Always strap 9-10.
Always strap 11-12.

Note: These two straps not usually removed, since the MCS-85 System Design Kit does not have SHIFT or CONTROL keys on its keyboard. These straps have no effect on operation of the corresponding key functions on a teletypewriter or other ASCII terminal that is connected to the TTY interface. They are provided for your use if you wish to modify the SDK-85's keyboard functions and replace its monitor software with your own.

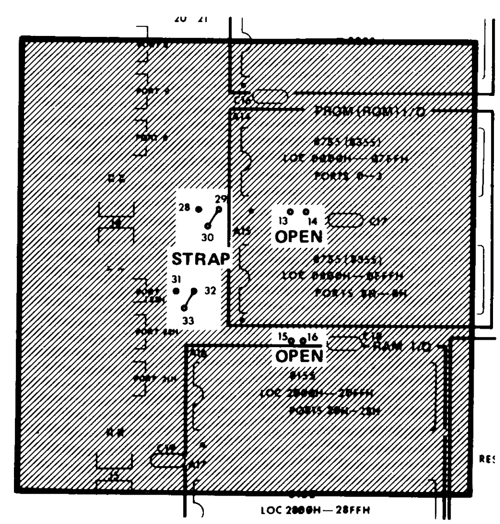


Figure 3-2b Strapping Connections for 8755A PROMS

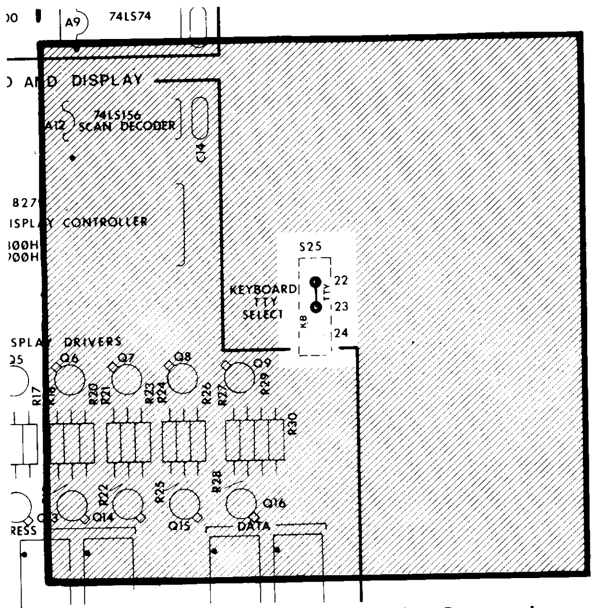


Figure 3-3 Teletypewriter Strapping Connection

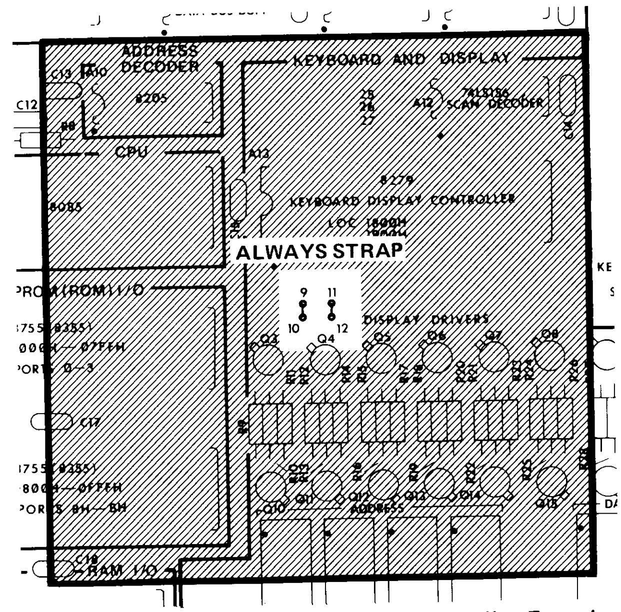


Figure 3-5 Disabling Unused Keyboard Controller Functions

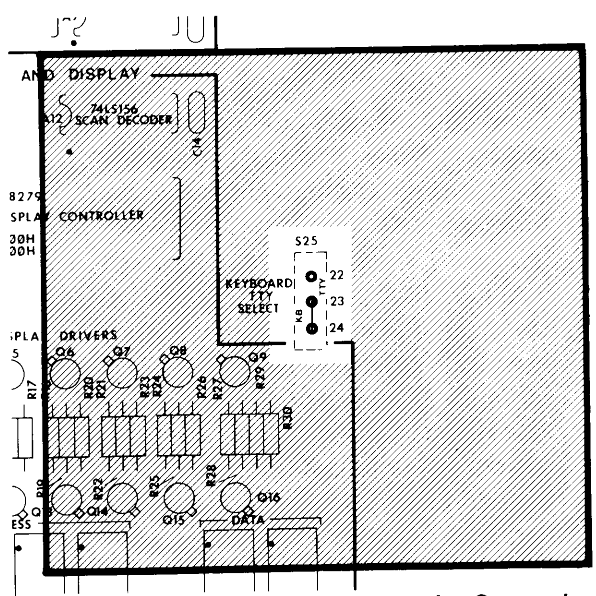


Figure 3-4 Keyboard-Display Strapping Connection

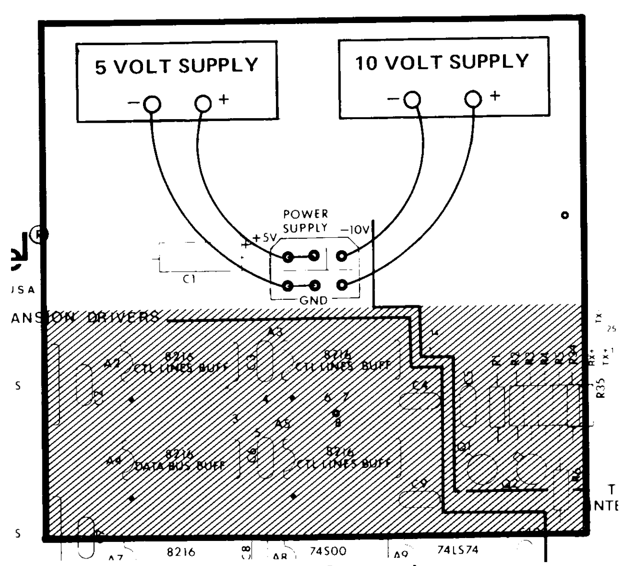


Figure 3-6 Power Supply Connections

hole pattern on the board (see p. 3-13 for the part number). If you are going to use a teletypewriter, connect a −10 Volt power supply with its negative output at the −10V point on the board. Connect the positive side of the −10 Volt power supply to the GND bus.

CAUTION

Do not turn on power until instructed to do so.

3-4 INSTALLING LARGE IC DEVICES

When you've finished all soldering operations on the board and are ready to fire it up, then it's time to plug in the large ICs. Once more, please make note of the precautions for handling these large MOS devices.

(Text continues on page 3-6.)

TABLE 3-4
BUS EXPANSION STRAPPING

FUNCTION	BASIC KIT WITHOUT EXPANSION MEMORY (Figure 3-7)	AUGMENTED KIT WITH EXPANSION MEMORY (Figure 3-8) (Also See Paragraph 3-7.)
RST 6.5	Strap 3-5	Strap 3-4 if no input is connected to J1-20. Leave 3, 4, and 5 not strapped if input is to be supplied for this restart function.
HOLD	Strap 6-8	Strap 7-8 if no input is connected to J1-14. Leave 6, 7, and 8 not strapped if input is to be supplied for this function.
INTR	Strap 20-21	Strap 20-21 if no input is connected to J1-18. Leave 20-21 not strapped if input is to be supplied for this function.
Memory Address Locations	Leave 25-26-27 unstrapped.	Strap 25-26 if all memory locations are external, i.e., addressed via bus expansion drivers.* (See Figure 3-9.) Strap 25-27 to enable the bus expansion drivers only when the upper 32K memory locations (8000H-FFFFH) are addressed. (See Figure 3-10.)

*Note: No devices may be installed in positions A13, A14, A15, A16, and A17 if this option is strapped.

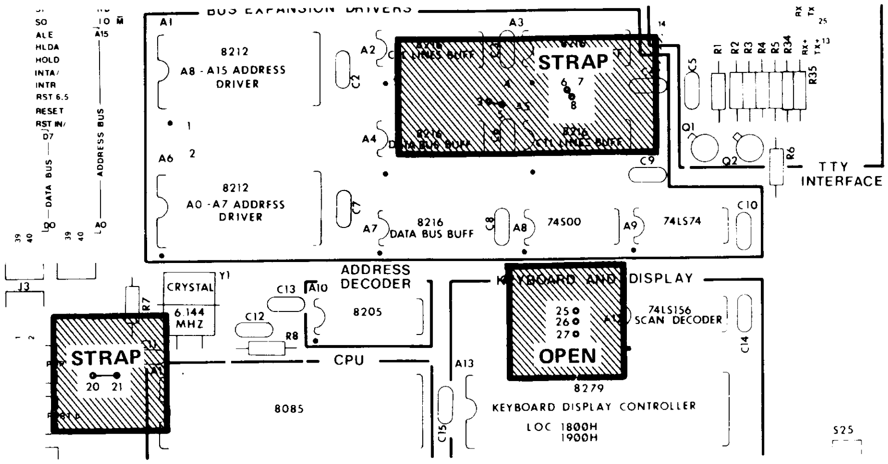


Figure 3-7 Strapping Required for Basic Kit (No Bus Expansion)

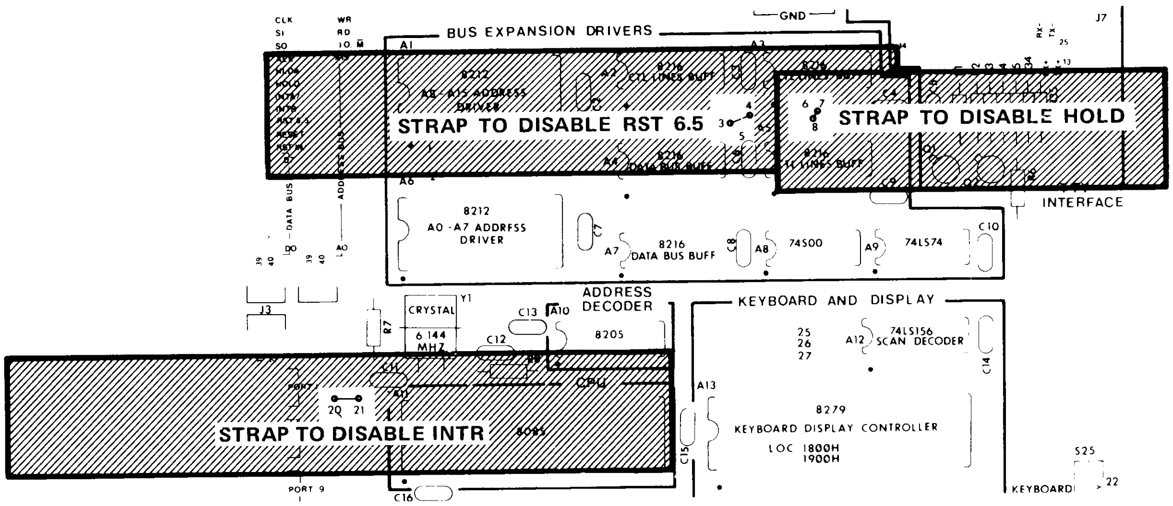


Figure 3-8 Strapping Options for Bus Expansion Control Lines

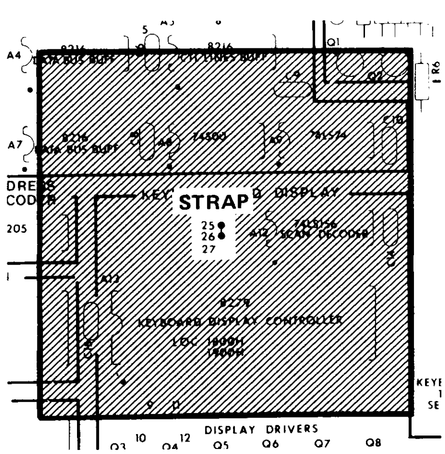


Figure 3-9 Strapping Options for all External Memory

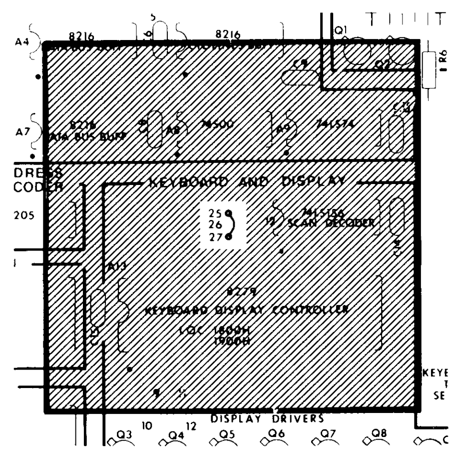


Figure 3-10 Strapping Options for Internal/External Memory

CAUTION

Large-scale integrated circuits are fragile! Dropping, twisting, or uneven pressure may break them. The discharge of static electricity can destroy them internally. Leave them embedded in the conductive-foam backing sheet until ready to install on the board. Never press down hard upon, twist, or bend the larger devices. Touch the exposed metal traces of the board with your hand before inserting one in its socket. The soldering of large devices directly on the circuit board is not recommended. If your Kit is provided with 8755 EPROM, do not remove the opaque sticker covering the window. Ultraviolet radiation including sunlight, can erase the monitor software contained in the device.

Inspect each IC to see that its leads are reasonably straight. (It's okay for the device to be a bit bow-legged.) The forked end of the soldering aid is a good tool for straightening bent leads. Carefully place an IC on its intended socket, oriented properly, with one row of its pins resting lightly in the socket holes. With your fingers or with the soldering aid, gently tease the other row of pins into their socket holes. Be sure no single pins have escaped. Once all pins have started, press down gently with fingers or with something flat to seat the device in its socket.

Each device must be oriented properly in its socket or it won't work. Every DIP device made has either a notch of some kind or a dot at one end. On the SDK-85 board, each notch or mark must face to the left. The markings on the board indicate this orientation. They also show which device type goes where. (See the pictorials on pages 2-9 and 2-15.)

3-5 STARTING THE FIRST TIME

Once you are certain that all parts are properly installed, the correct strapping options are soldered, and the power supplies connected, you are ready to start your MCS-85 System Design Computer. Clear the surface of your work table of any tools or wire that could come in contact with the underside of the circuit board and short it, and be sure there aren't any wire clippings on top of the board by accident.

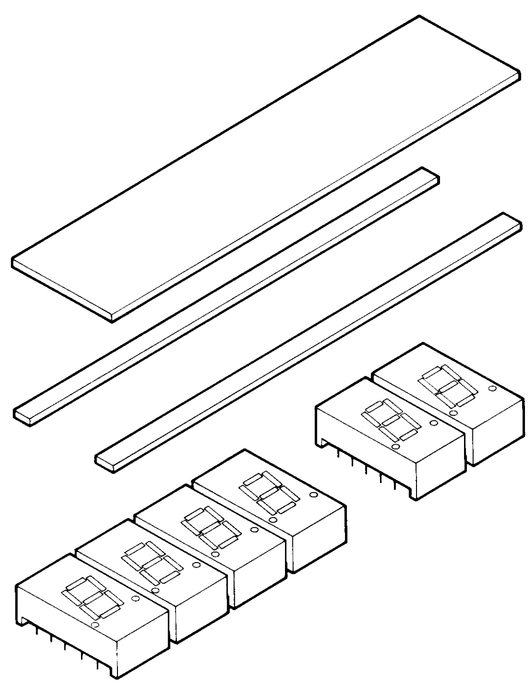
Peel the coverings from the red window and lay it on the display. (Don't stick it down yet.)

Energize the +5 Volt power supply.

Press the RESET button on the keyboard. The display should respond by reading out "– 80 85."

If the above readout appears, go on to Chapter 4 of this book and try out each button and function. Verify that each command produces the specified result, and that all segments of each 7-segment character display light.

Once you know the displays are all working right, peel the backing from the two strips of double-sided tape and use them to stick the red window in place.



3-6 WHAT IF IT DOESN'T?

If there is no response to the RESET command,

- ☐ Use the multimeter to check for the presence and proper polarity of +5 Volts on the board.
- ☐ Check all of the strapping connections, and be sure they are in the right places for the configuration you chose.
- ☐ Check carefully the seating of each and every pin of each of the four large ICs. Be sure no pins have accidentally bent over and missed the socket.
- ☐ Go back over the Chapter 2 assembly procedure and scan and check off all of the component values and all of the solder connections.
- ☐ Check the orientation of all semiconductor devices.
- ☐ Inspect for solder bridges or loose solder joints.

If all devices are properly soldered or firmly in their sockets and still there's no result, it can be presumed that there is a bad part somewhere. The keyboard switches can be checked using the multimeter, as mentioned in Chapter 2. If all switches are closing positively when pressed, and opening when released, further effective troubleshooting can be accomplished if you have a dual-trace oscilloscope of at least 5 MHz bandwidth, or a logic analyzer.

- ☐ Pin 37 of cpu A11 (8085) should show a clock output of 3.072 MHz (326 ns period). If it doesn't, there's something wrong with the 8085 or the crystal.
- ☐ Pin 30 of A11 should have a positive-going pulse about 160 ns wide every μs or so. This is the ALE pulse that indicates that the cpu is executing instructions.
- ☐ Pin 1 of address decoder A10 (8205) should pulse. If not, your 8085 is probably bad.
- ☐ If pin 1 of A10 pulses, check pin 15 of A10. If A10-15 doesn't follow A10-1, or has bad output voltage levels, the 8205 is either bad or installed wrong.
- ☐ If all else fails, call the Intel Service Hotline and describe the results of the foregoing procedure.

The numbers are:

(800) - 538-8014 when calling from out-
(800) - 538-8015 side California

(800) - 672-3507 California only

Note: The Service Hotline is available to provide limited support to help you get your kit running. If we can't help you over the phone, you may be directed to return your kit to us and we'll fix it for a flat fee and send it back to you. **The Service Hotline is available Monday through Friday, between 8 AM and 3:30 PM, Pacific time.**

IMPORTANT: The Service Hotline *is not* able to provide help to you in writing programs for your kit or in making hardware modifications. Please rely on the documentation provided with your kit for assistance.

TABLE 3-5
INTERFACE CONNECTOR J7
PIN ASSIGNMENTS

PIN	MARKING	ASSIGNMENT
1	–	Open
14	–	Open
2	–	Open
15	–	Open
3	–	Open
16	–	Open
4	–	Open
17	–	Open
5	–	Open
18	–	Open
6	–	Open
19	–	Open
7	–	Ground
20	–	Open
8	–	Open
21	–	Open
9	–	Open
22	–	Open
10	–	Open
23	–	Open
11	–	Open
24	RX–	Receive Return (–)
12	RX+	Receive (+)
25	TX–	Transmit Return (–)
13	TX+	Transmit (+)

3-7 CONNECTING A TELETYPEWRITER

If you wish to use a teletypewriter with your SDK-85 computer, connect it at Interface Connector J7 as shown in Table 3-5. You may use either a male connector or a female connector. (See Table 3-11.) Only four pins of this connector are assigned for Teletypewriter use; the remaining pins may be wire-wrapped to serve any function you choose.

TABLE 3-6
BUS EXPANSION CONNECTOR J1 PIN ASSIGNMENTS

ASSIGNMENT	PIN	PIN	MARKING	ASSIGNMENT	I/O
GND	1	2	—	OPEN	—
GND	3	4	CLK	Buffered CLK	O
GND	5	6	S1	Buffered S1	O
GND	7	8	S0	Buffered S0	O
GND	9	10	ALE	Buffered ALE	O
GND	11	12	HLDA	Buffered HLDA	O
GND	13	14	HOLD	Buffered HOLD	I
GND	15	16	INTA/	Buffered $\overline{\text{INTA}}$	O
GND	17	18	INTR	INTR	I
GND	19	20	RST 6.5	Buffered RST 6.5	I
GND	21	22	RST	Buffered RESET OUT	O
GND	23	24	RST IN/	$\overline{\text{RESET INPUT}}$	I
GND	25	26	D7	Buffered D7	I/O
GND	27	28	DATA BUS	Buffered D6	I/O
GND	29	30	DATA BUS	Buffered D5	I/O
GND	31	32	DATA BUS	Buffered D4	I/O
GND	33	34	DATA BUS	Buffered D3	I/O
GND	35	36	DATA BUS	Buffered D2	I/O
GND	37	38	DATA BUS	Buffered D1	I/O
GND	39	40	D0	Buffered D0	I/O

TABLE 3-7
BUS EXPANSION CONNECTOR J2 PIN ASSIGNMENTS

ASSIGNMENT	PIN	PIN	MARKING	ASSIGNMENT	I/O
GND	1	2	RDY	READY	I
GND	3	4	WR/	Buffered $\overline{WR}$	O
GND	5	6	RD/	Buffered $\overline{RD}$	O
GND	7	8	IO/$\overline{M}$	Buffered IO/$\overline{M}$	O
GND	9	10	A15	Buffered A15	O
GND	11	12		Buffered A14	O
GND	13	14		Buffered A13	O
GND	15	16		Buffered A12	O
GND	17	18		Buffered A11	O
GND	19	20		Buffered A10	O
GND	21	22		Buffered A9	O
GND	23	24	ADDRESS BUS	Buffered A8	O
GND	25	26		Buffered A7	O
GND	27	28		Buffered A6	O
GND	29	30		Buffered A5	O
GND	31	32		Buffered A4	O
GND	33	34		Buffered A3	O
GND	35	36		Buffered A2	O
GND	37	38		Buffered A1	O
GND	39	40	A0	Buffered A0	O

TABLE 3-8
I/O PORT CONNECTOR J3 PIN ASSIGNMENTS

ASSIGNMENT	PIN	PIN	MARKING	ASSIGNMENT
P1-6*	1	2		P1-7
P1-4	3	4	PORT 1	P1-5
P1-2	5	6		P1-3
P1-0	7	8		P1-1
P0-6	9	10		P0-7
P0-4	11	12	PORT 0	P0-5
P0-2	13	14		P0-3
P0-0	15	16		P0-1
P9-6	17	18		P9-7
P9-4	19	20	PORT 9	P9-5
P9-2	21	22		P9-3
P9-0	23	24		P9-1
P8-6	25	26		P8-7
P8-4	27	28	PORT 8	P8-5
P8-2	29	30		P8-3
P8-0	31	32		P8-1
GROUND	33	34		GROUND

***Note:**
1. Pn-m stands for PORT n Bit m (e.g. P9-6 means PORT 9H Bit 6).
2. Ports 0 & 1 are Ports A and B of 8355 (A14).
3. Ports 8 & 9 are Ports A and B of 8755 (A15).

TABLE 3-9
I/O PORT CONNECTOR J4 PIN ASSIGNMENTS

ASSIGNMENT	PIN	PIN	MARKING	ASSIGNMENT
P23H-4	1	2	PORT 23H	P23H-5
P23H-2	3	4	PORT 23H	P23H-3
P23H-0	5	6	PORT 23H	P23H-1
P22H-6	7	8	PORT 22H	P22H-7
P22H-4	9	10	PORT 22H	P22H-5
P22H-2	11	12	PORT 22H	P22H-3
P22H-0	13	14	PORT 22H	P22H-1
P21H-6	15	16	PORT 21H	P21H-7
P21H-4	17	18	PORT 21H	P21H-5
P21H-2	19	20	PORT 21H	P21H-3
P21H-0	21	22	PORT 21H	P21H-1
OPEN	23	24		OPEN
GROUND	25	26		GROUND

Note: Port 21H is Port A, Port 22H is Port B, Port 23H is Port C of 8155 (A16).

TABLE 3-10
I/O PORT AND TIMER CONNECTOR J5 PIN ASSIGNMENTS

ASSIGNMENT	PIN	PIN	MARKING	ASSIGNMENT
P2BH-4	1	2	PORT 2BH	P2BH-5
P2BH-2	3	4	PORT 2BH	P2BH-3
P2BH-0	5	6	PORT 2BH	P2BH-1
P2AH-6	7	8	PORT 2AH	P2AH-7
P2AH-4	9	10	PORT 2AH	P2AH-5
P2AH-2	11	12	PORT 2AH	P2AH-3
P2AH-0	13	14	PORT 2AH	P2AH-1
P29H-6	15	16	PORT 29H	P29H-7
P29H-4	17	18	PORT 29H	P29H-5
P29H-2	19	20	PORT 29H	P29H-3
P29H-0	21	22	PORT 29H	P29H-1
Timer $\overline{\text{OUT}}$	23	24	TIMER $\overline{\text{OUT}}$/IN	Timer In
GROUND	25	26		GROUND

Note: Port 29H is Port A, Port 2AH is Port B, Port 2BH is Port C of expansion RAM 8155 (A17).
Timer is on the same 8155 (A17).

TABLE 3-11
SUGGESTED CONNECTOR TYPES

REFERENCE DESIGNATION	FUNCTION	NO. OF PINS	MFR.	MFR'S. PART NO.
J1	Bus Expansion	40	Spectra Strip	800-576
J2	Bus Expansion	40	Spectra Strip	800-576
J3	I/O Ports	34	Spectra Strip	800-579
J4	I/O Ports	26	Spectra Strip	800-583
J5	I/O Ports and Timer	26	Spectra Strip	800-583
J6	Not Used			
J7	TTY Interface	25		
	Female } Optional		AMP	206584
	Male } Optional		AMP	206604
—	Power Supply	6	Molex	Model No. 1261
	Recepticle			03-09-1064
	Plug			03-09-2062

CHAPTER 4
OPERATING INSTRUCTIONS

4-1 WHAT IT DOES

The things you can do with the basic SDK-85 kit are:

- Examine the contents of all memory and register locations
- Deposit program steps or data in RAM or register locations
- Execute programs or subroutines upon command
- Reset (start) the monitor upon command
- Interrupt and start operation at a location you specify upon command

You may select either the keyboard and display on the board or a teletypewriter as the console device by operating a switch or by placing a jumper wire at the appropriate place on the board. (See Chapter 3.) Keyboard/display operation and teletypewriter operation are described separately in the following paragraphs.

Two of the keyboard buttons continue to function in teletypewriter mode, as well as in keyboard/display mode. These are the [RESET] and the [VECT INTR] keys.

4-2 THE BUTTONS AND DISPLAYS

Keyboard/display operation is done by pressing keys on the keypad. Responses are displayed either by echoing the key pressed or by prompting you with a message or prompt. When the [RESET] button is pressed, the monitor is ready to accept commands. For numeric arguments, the valid range is from 1 to 4 hexadecimal digits for address information and 1 to 2 hex digits for register and memory data.

Longer numbers may be entered, but such numbers will be evaluated modulo 2^{16} or 2^{8} respectively, i.e., only the last four or the last two digits entered will be accepted.

As noted, the number system being used in the SDK-85 is the hexadecimal, or base-16 number system. Table 4-1 lists the hexadecimal, decimal (base 10), and binary (base two) equivalents. The table also shows how each hex digit will appear in the seven-segment LED displays.

TABLE 4-1
NUMBER SYSTEMS

HEX	DECIMAL	BINARY	LED DISPLAY
0	0	0000	0
1	1	0001	1
2	2	0010	2
3	3	0011	3
4	4	0100	4
5	5	0101	5
6	6	0110	6
7	7	0111	7
8	8	1000	8
9	9	1001	9
A	10	1010	A
B	11	1011	b
C	12	1100	C
D	13	1101	d
E	14	1110	E
F	15	1111	F

Whenever the monitor expects a command, the display shows a dash ("–") at the left edge of the address field (possibly along with an error message). When the monitor expects a parameter, a decimal point will be displayed at the right edge of the field into which the argument will be placed. A parameter will be either an address or a byte of data which is used during the execution of a command.

In the descriptions of the command modes, upper case letters and numbers enclosed in boxes represent keyboard keys. Words or phrases in lower case enclosed in brackets "<>" describe the nature of the command parameters you may input.

The () in the Format Statement indicates an optional argument.

Reset:

The [RESET] key causes a hardware reset, and starts the monitor. The message "–80 85" will be displayed across the address and data field of the display if you are in display-keyboard mode. If in teletypewriter mode, the sign on message "SDK-85 VER X.X" will be printed. The monitor is ready to accept a command after a reset, and saves no information about the state of any user program before the reset.

Substitute Memory:

[SUBST MEM] <address> [NEXT] (<data>) [NEXT] (<data>) . . . [EXEC]

The substitute memory command allows you to read the contents of ROM memory and to examine and modify the contents of RAM memory locations.

The address argument denotes the contents of the memory address to be examined, and may be from 1 to 4 hex digits. If you enter longer numbers, only the last 4 digits entered are used). As soon as the number is terminated by the [NEXT] key, the contents of that location are shown in the data field, along with a decimal point at the right edge of the field. Entering a new number will cause that number to be displayed in the data field; however, the contents of the memory location will not be changed until an [EXEC] or [NEXT] key is pressed.

Pressing [NEXT] will place the contents displayed in the data field into the displayed memory address. Then the address and contents of the next higher memory location will automatically be shown. Pressing [EXEC] will place the contents displayed in the data field into the memory address displayed in the address field, and will also terminate the command.

Pressing [NEXT] while the address FFFF is being displayed will cause address 0000 to be displayed.

Whenever the command changes the contents of a memory location, it also verifies that the change has occurred correctly. If the contents of the location do not agree with what the new value should be (i.e., if the memory location is in ROM or is nonexistent), an error message is generated.

SUBSTITUTE MEMORY EXAMPLE 1

Using [SUBST MEM] to list the first few Monitor locations:

KEY	ADDR	DATA
[SUBST MEM]	.	
[0]	0000.	
[NEXT]	0000	3E.
[NEXT]	0001	00.
[NEXT]	0002	32.
[NEXT]	0003	00.
[EXEC]	–	

SUBSTITUTE MEMORY EXAMPLE 2

Using [SUBST MEM] to enter a small program:

KEY	ADDR	DATA
[SUBST MEM]	.	
[2]	0002.	
[0]	0020.	
[0]	0200.	
[0]	2000.	
[NEXT]	2000	**.
[3 I]	2000	03.
[E]	2000	3E.
[NEXT]	2001	**.
[4 SPH]	2001	04.
[7 PCL]	2001	47.
[NEXT]	2002	**.
[C]	2002	0C.
[F]	2002	CF.
[EXEC]	–	

NOTE: ** represents unpredictable values.

After loading the above program, use [SUBST MEM] again to go back and check locations 2000-2002 to see that they contain:

ADDRESS	DATA	CORRESPONDING 8085 ASSEMBLY LANGUAGE INSTRUCTIONS
2000	3E	MVI A, 47H
2001	47	
2002	CF	RST 1

This program will load the A register with the number 47 and jump back to the monitor.

Examine Registers:

[EXAM REG] <reg> [NEXT] (<data>) [NEXT] (<data>) . . . [EXEC]

The examine command allows you to display and modify the contents of the 8085 CPU registers. Pressing the [EXAM REG] key blanks both the address and data fields, and displays a decimal point at the right edge of the address field. At this point, you must press a register key (register names are denoted by legends on the keyboard). Any other key will generate an error response.

If a register key is pressed, the name of the register will appear in the address field, and the contents of the register will appear in the data field, along with a decimal point at the right hand edge. Entering a number will cause the number to be displayed in the data field; however, the contents of the register will not be changed until an [EXEC] or [NEXT] key is pressed.

Pressing [NEXT] will place the contents displayed in the data field into the register named in the address field, then will display the name and contents of the next register in sequence (See Table 4-2). Pressing [EXEC] will place the contents displayed in the data field in the register named in the address field, and will also terminate the command.

Pressing [NEXT] while register PCL is being displayed has the same effect as pressing [EXEC] .

The format for the I register is the lower 4 bits of the accumulator following execution of a RIM instruction. A "1" in an interrupt mask field denotes a masked condition. A "0" must be entered to use that interrupt.

The format for the I register is:

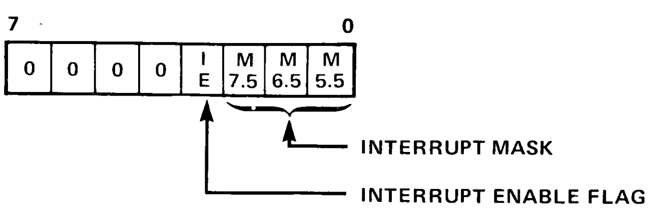
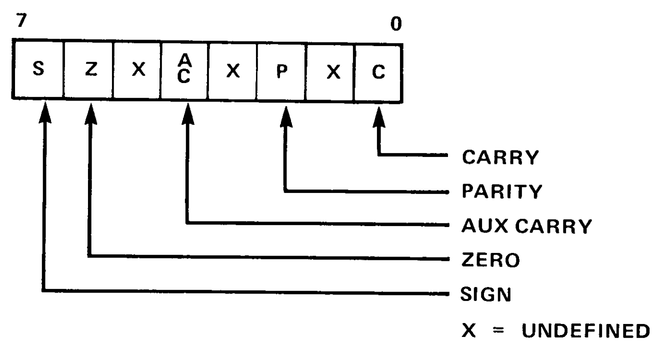


TABLE 4-2
REGISTER DISPLAY SEQUENCE

KEY/DISPLAY CODE	REGISTER
A	CPU register A
B	CPU register B
C	CPU register C
D	CPU register D
E	CPU register E
F	CPU flags byte
I	interrupt mask
H	CPU register H
L	CPU register L
SPH	most significant byte of stack pointer
SPL	least significant byte of stack pointer
PCH	most significant byte of program counter
PCL	least significant byte of program counter

The flag byte contains the 8085 CPU's condition flags.

The format for the flag byte is:



For more information about the 8085's flags and interrupt mask feature, consult the **MCS-85 User's Manual.**

EXAMINE REGISTER EXAMPLE 1

Using [EXAM REG] to initialize the 8085's stack pointer to 20C2:

KEY	ADDR	DATA
[EXAM REG]	.	
[4 SPH]	SPH	**.
[2]	SPH	02.
[0]	SPH	20.
[NEXT]	SPL	**.
[C]	SPL	0C.
[2]	SPL	C2.
[EXEC]	–	

EXAMINE REGISTER EXAMPLE 2

Using [EXAM REG] to examine the contents of the 8085's Registers:

KEY	ADDR	DATA
[EXAM REG]	.	
[A]	A	**.
[NEXT]	b	**.
[NEXT]	C	**.
[NEXT]	d	**.
[NEXT]	E	**.
[NEXT]	F	**.
[NEXT]	I	**.
[NEXT]	H	**.
[NEXT]	L	**.
[NEXT]	SPH	**.
[NEXT]	SPL	**.
[NEXT]	PCH	**.
[NEXT]	PCL	**.
[NEXT] or [EXEC]	–	

NOTE: ** represents the contents of the register whose name is in the address field of the display.

Go:

$\boxed{\text{GO}}$ (<address>) $\boxed{\text{EXEC}}$

Pressing the $\boxed{\text{GO}}$ key causes the contents of the program counter (PCH and PCL) to be displayed in the addressed field, along with a decimal point at the right edge of the field. The program counter is available for change, and any number entered (a number is optional) becomes the new contents of the program counter.

Pressing the $\boxed{\text{EXEC}}$ key transfers control of the CPU to the address in the address field (contents of the program counter). Before the transfer of control, the address and data display fields are cleared, and an 'E' is displayed at the left edge of the address field.

Pressing any other key but $\boxed{\text{EXEC}}$ generates an error message.

The monitor regains control of the CPU only after a $\boxed{\text{RESET}}$ or after execution of an RST 0, RST 1, or JMP 0 instruction in program.

IMPORTANT:

Note that because of the way the GO and SINGLE STEP commands are implemented in the Monitor, $\boxed{\text{GO}}$ and $\boxed{\text{SINGLE STEP}}$ will not work unless the 8085's stack pointer is pointing to an existing portion of RAM memory. If at any time these two commands don't seem to be working, set SPH to 20 and SPL to C2 using $\boxed{\text{EXAM REG}}$, then try it again. (Locations 20C2 to 20FF are reserved for the monitor program, therefore the stack pointer must be set to 20C2 or lower so as not to interfere with the monitor.)

GO COMMAND EXAMPLE

Now you can execute the program you entered in Example 2 of the [SUBST MEM] command. First, check to make sure the 3- location program is in memory, then the program will be executed.

KEY	ADDR	DATA	COMMENTS
[SUBST MEM]	.		
[2]	0002.		
[0]	0020.		
[0]	0200.		
[0]	2000.		
[NEXT]	2000	3E.	MVI A, 47
[NEXT]	2001	47.	
[NEXT]	2002	CF.	RST 1
[EXEC]	–		
[GO]	****.	**	
[2]	0002.		
[0]	0020.		
[0]	0200.		
[0]	2000.		
[EXEC]	– 80	85	

NOTE: **** denotes "don't care" values

Recall that this small program loads the A register with the number 47 and restarts the monitor. To verify that the A register now holds 47 and to get more practice using [EXAM REG], try the following sequence:

KEY	ADDR	DATA	COMMENT
[EXAM REG]	.		
[A]	A	47.	A reg now holds 47.
[0]	A	00.	
[EXEC]	–		Now A holds 0
[GO]	****.	**	
[2]	0002.		
[0]	0020.		Run the small Program again
[0]	0200.		
[0]	2000.		
[EXEC]	– 80	85	
[EXAM REG]	.		
[A]	A	47	Now A holds 47 again

Now try placing other values in location 2001 using [SUBST MEM] and use [GO] to execute the program again, seeing how those values are loaded into the A register after execution.

Single Step:

[SINGLE STEP] (<address>) [NEXT] [NEXT] . . . [EXEC]

Pressing the [SINGLE STEP] key causes the contents of the program counter (PCH and PCL) to be displayed in the address field of the display along with a decimal point at the right hand edge of the field. The data field contains the contents of the address denoted by the contents of the program counter. The program counter is made available for change, and any number entered (a number is optional) becomes the new contents of the program counter.

Pressing the [NEXT] key causes the CPU to execute the one instruction pointed to by the program counter. After execution the monitor regains control of the CPU, and the address and data fields show the new contents of the program counter (address of next instruction to execute) and contents of the byte addressed by the program counter, respectively. The decimal point is turned on at the right hand edge of the address field, indicating that the program counter is available again.

If the [EXEC] key is pressed, no instruction is executed. The address displayed in the address field is made the contents of the program counter and the single step command is terminated. You may now examine or modify registers and memory locations to verify program execution. Pressing the [SINGLE STEP] key takes you back to the single step mode, and subsequent pressing of the [NEXT] key allows you to continue, instruction by instruction, through your program.

Single stepping is implemented in the SDK-85 hardware by repeatedly interrupting the processor. Since interrupts cannot be recognized during the EI and DI instructions of the 8085, single step will not stop at either of these instructions.

SINGLE STEP EXAMPLE

Single stepping through the SDK-85 Monitor. This is what you should see on the display:

KEY	ADDR	DATA
[SINGLE STEP]	****.	**
[8 H]	0008.	
[NEXT]	000b.	E1
[NEXT]	000C.	22
[NEXT]	000F.	F5
[NEXT]	0010.	E1

To resume full speed operation at this point, do the following:

[EXEC]	–	
[GO]	0010.	E1
[EXEC]	– 80	85

Vector Interrupt:

The [VECT INTR] key is similar to the [GO] key in the respect that it takes control away from the monitor and gives it to another program. The interrupt key causes immediate recognition of RST 7.5 interrupt and control passes to location 3C in the monitor. This location contains an unconditional branch to instruction location 20CE in user RAM. You may place any instruction you wish in Locations 20CE thru 20D0 (e.g., a branch to a keyboard interrupt routine). The monitor does not regain control without specific action (a [RESET] command, or a RST 0, RST 1, or JMP 0 program instruction). In branching back to the monitor, unless the RST 1 instruction is executed, the monitor loses all past information about the user program.

Since an interrupt is recognized by the hardware, the monitor cannot clear the display; thus the display may remain unchanged after interrupt.

IMPORTANT: Two conditions must be satisfied for the Vector Interrupt feature to be enabled:

1. Interrupts must be enabled (by executing an EI instruction).

2. RST 7.5 must be unmasked (mask reset by the SIM instruction or by modifying the I-Register).

Program Debugging — The Use of Breakpoints

Along with the "cold start" reset caused when the [RESET] button is pressed, the monitor also implements a "warm start" procedure. Execution of an RST 1 instruction will cause the monitor to enter this "warm start" routine. The monitor will display the same message as a [RESET] ('–80 85'), but all registers and user memory will be preserved in the state they were in at the time of execution of the RST 1. No system reset or initialization will be performed.

By placing RST 1 instructions at key RAM locations where you want to examine the CPU status, you can break from your program and then examine and set memory locations and registers, or single-step a portion of your program.

To resume execution of the user program, press [GO]. The PC value of the next instruction appears in the address field of the display. Then press [EXEC] to continue execution.

Error Conditions — Illegal Key

If a key is pressed which is illegal in its context (e.g., a command key is pressed when the monitor is expecting a number), the command is aborted and an error message is generated. This message takes the form "–Err", displayed in the address field. The monitor is then ready to accept a command. The error message will be cleared when a command key is pressed. Therefore, you can cancel a command before you press [NEXT] or [EXEC] by pressing any illegal key instead.

Memory Substitution Errors

If the substitute memory command determines that the contents of a memory location were not changed correctly (i.e. location is in ROM or is nonexistent), the command is aborted and an error message is generated. This message also takes the form "–Err", displayed in the address field. The monitor is then ready to accept a new command. The error message will be cleared when a command key is pressed.

4-3 TELETYPEWRITER OPERATION

Console Commands

This portion of the SDK-85 monitor communicates via a teletypewriter (console). Operation consists of dialogue between the operator and the monitor in the monitor's command language. After you press the [RESET] button on the SDK-85 keypad, the monitor begins the dialogue by typing a sign-on message on the console ("MCS-85 Kit") and then requests a command by typing a prompt character ("."). Commands are in the form of a single alphabetic character specifying the command, followed by a list of numeric or alphabetic parameters. Numeric parameters are entered as hexadecimal numbers. The monitor recognizes the characters 0 through 9 and A through F as legal hexadecimal digits. Longer numbers may be entered, but only the last four digits will be retained.

The only command requiring an alphabetic parameter is the "X" command. The nature of such parameters will be discussed in the section explaining the command.

Use of the Monitor for Programming and Checkout

The monitor allows you to enter, check out, and execute small programs. It contains facilities for memory display and modification, 8085 CPU register display and modification, program loading from the console device, and program initiation with a breakpoint facility. In addition, the [VECT INTR] key on the keyboard may be used to initiate your own keyboard interrupt routine.

Command Structure

In the following paragraphs, the monitor command language is discussed. Each command is described, and examples of its use are included for clarity. Error conditions that may be encountered while operating the monitor are described on page 4-13.

The monitor requires each command to be terminated by a carriage return. With the exception of the "S" and 'X" commands, the command is not acted upon until the carriage return is sensed. Therefore, you may abort any command, before entering the carriage return, by typing any illegal character (such as RUBOUT).

Except where indicated otherwise, a single space is synonymous with the comma for use as a delimiter. Consecutive spaces or commas, or a space or comma immediately following the command letter, are illegal in all commands except the "X" command (see below).

Items enclosed in parentheses "( )" are optional.

Display Memory Command, D:

D <low address>, <high address>

Selected areas of addressable memory may be accessed and displayed by the D command. The D command produces a formatted listing of the memory contents between <low address> and <high address>, inclusive, on the console. Each line of the listing begins with the address of the first memory location displayed on that line, represented as 4 hexadecimal digits, followed by up to 16 memory locations, each one represented by 2 hexadecimal digits.

Program Execute Command, G:

G (<entry point>)

Control of the CPU is transferred from the monitor to the user program by means of the program execute command G. The entry point should be an address in RAM which contains an instruction in the program. If no entry point is specified, the monitor uses, as an address, the value on top of the stack when the monitor was entered.

G COMMAND EXAMPLE

G2000

Control is passed to location 2000.

D COMMAND EXAMPLE

D9, 26

0009	EF	20	E1	22	F2	20	F5									
0010	E1	22	ED	20	21	00	00	39	22	F4	20	21	ED	20	F9	C5
0020	D5	C3	3F	00	C3	57	01									

Insert Instructions into RAM, I:

I <address>
<data>

Single instructions, or an entire user program, are entered into RAM with the I command. After sensing the carriage return terminating the command line, the monitor waits for the user to enter a string of hexadecimal digits (0 to 9, A to F). Each digit in the string is converted into its binary value, and then loaded into memory, beginning at the starting address specified and continuing into sequential memory locations. Two hexadecimal digits are loaded into each byte of memory.

Separators between digits (spaces, commas, carriage returns) are ignored; illegal characters, however, will terminate the command with an error message (see page 4-13). The character ESC or ALT-MODE (which is echoed to the console as "$") terminates the digit string.

I COMMAND EXAMPLE 1

I2010
112233445566778899$

This command puts the following pattern into RAM:

2010 11 22 33 44 55 66 77 88 99

I COMMAND EXAMPLE 2

I2040
123456789$

This command puts the following pattern into RAM:

2040 12 34 56 78 90

Note that since an odd number of hexadecimal digits was entered initially, a zero was appended to the digit string.

Move Memory Command, M:

M <low address>, <high address>, <destination>

The M command moves the contents of memory between <low address> and <high address> inclusive, to the area of RAM beginning at <destination>. The contents of the source field remain undisturbed, unless the receiving field overlaps the source field.

The move operation is performed on a byte-by-byte basis, beginning at <low address>. Care should be taken if <destination> is between <low address> and <high address>. For example, if location 2010 contains 1A, the command M2010, 201F 2011 will result in locations 2010 to 2020 containing "1A1A1A . . .", and the original contents of memory will be lost.

The monitor will continue to move data until the source field is exhausted, or until it reaches address FFFF. If the monitor reaches FFFF without exhausting the source field, it will move data into this location, then stop.

M COMMAND EXAMPLE

M2010, 204F, 2050

64 bytes of memory are moved from 2010-204F to 2050-208F by this command.

Substitute Memory Command, S:

S <address> (<data>)

The S command allows you to examine and optionally modify memory locations individually. The command functions as follows:

1. Type an S, followed by the hexadecimal address of the first memory location you wish to examine, followed by a space or comma.

2. The contents of the location are displayed, followed by a dash (–).

3. To modify the contents of the location displayed, type in the new data, followed by a space, comma, or carriage return. If you do not wish to modify the location, type only the space, comma, or carriage return. The next higher memory location will automatically be displayed as in step (2).

4. Type a carriage return. The S command will be terminated.

S COMMAND EXAMPLE

S2050 AA- BB-CC 01-13 23-24

Location 2050, which contains AA, is unchanged, but location 2051 (which used to contain BB) now contains CC, 2052 (which used to contain 01) now contains 13, and 2053 (which used to contain 23) now contains 24.

Examine/Modify CPU Registers Command, X:

X (<register identifier>)

Display and modification of the CPU registers is accomplished via the X command. The X command uses <register identifier> to select the particular register to be displayed. A register identifier is a single alphabetic character denoting a register, as defined in Table 4-3.

TABLE 4-3
X COMMAND REGISTER IDENTIFIERS

IDENTIFIER CODE	REGISTER
A	Register A
B	Register B
C	Register C
D	Register D
E	Register E
F	Flags byte
I	Interrupt Mask
H	Register H
L	Register L
M	Registers H and L combined
S	Stack Pointer
P	Program Counter

The command operates as follows:

1. Type an X, followed by a register identifier or a carriage return.

2. The contents of the register are displayed (two hexadecimal digits for A, B, C, D, E, F, I, H, and L, four hexadecimal digits for M, S, & P), followed by a dash (–).

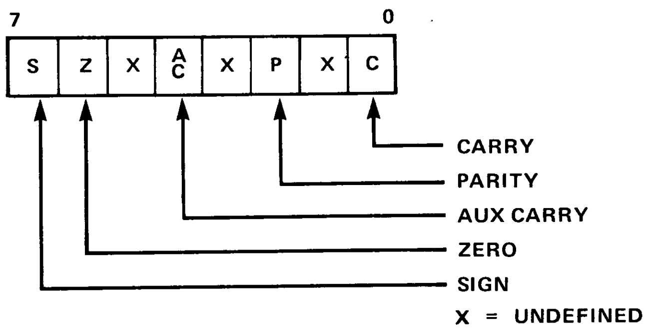
3. The register may be modified at this time by typing the new value, followed by a space, comma, or carriage return. If no modification is desired, type only the space, comma, or carriage return.

4. If a space or comma is typed in step (3), the next register in sequence will be displayed as in step 2 (unless P was just displayed which case the command is terminated). If a carriage return is entered in step 3, the X command is terminated.

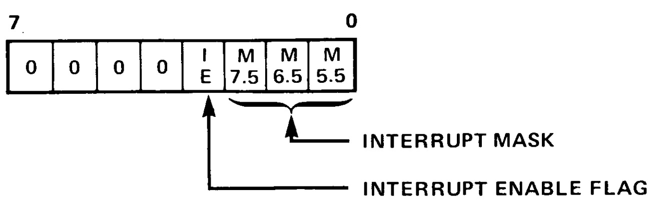
5. If a carriage return is typed in step (1) above, an annotated list of all registers and their contents is displayed.

   Note: The bits in the flag byte (F) and interrupt mask (I) are encoded as follows:

The format for the F register:



The format for the I register:



Note: For more information on the 8085's interrupt masks, please consult the **MCS-85 User's Manual.**

Program Debugging — Breakpoint Facility

The monitor treats the RST 1 instruction (CF) as a special sequence initiator. Upon execution of an RST 1 instruction the monitor will automatically save the complete CPU status and output the sign-on message "MCS-85 Kit" to the console. You may at that time display the contents of the CPU status register by initiating an "X" command. After examining the machine status and making any necessary changes you can resume execution of the program by inputting "G" and Carriage Return on the console. You can step through large portions of your program by inserting RST 1 instructions at key locations.

Error Conditions — Invalid Characters

Each character is checked as it is entered from the console. As soon as the monitor determines that the last character entered is illegal in its context, it aborts the command and issues an "*" to indicate the error.

INVALID CHARACTER EXAMPLE
D2000, 205G*
The character G was encountered in a parameter list where only hexadecimal digits and delimiters are valid.

Address Value Errors

Some commands require an address pair of the form <low address>, <high address>. If, on these commands, the value of <low address> is greater than or equal to the value of <high address>, the action indicated by the command will be performed on the data at low address only. Addresses are evaluated modulo 2^{16}. Thus, if a hexadecimal address greater than FFFF is entered, only the last 4 hex digits will be used. Another type of address error may occur when you specify a part of memory in a command which does not exist in the hardware configuration you are using.

In general, if a nonexistent portion of memory is specified as the source field for an instruction, the data fetched will be unpredictable. If a nonexistent portion of memory is given as the destination field in a command, the command has no effect.

CHAPTER 5
THE HARDWARE

5-1 OVERVIEW

This portion of the SDK-85 User's Manual should provide you with sufficient knowledge to write programs to exercise the basic system as well as providing capability to use the basic kit as a nucleus around which you can build larger systems.

Figure 5-1 is a functional block diagram of the SDK-85. The components enclosed in dashed boxes have places in the SDK-85 printed circuit board, but these are not needed for a minimum system and are not included in the kit. In addition, some control lines have been omitted from the block diagram for the sake of simplicity. The full SDK-85 schematic diagrams have been included in an appendix for your reference.

The text to follow describes each of the elements in the system:

5-2 SYSTEM COMPONENTS

The 8085 CPU & The System Buses

The 8085 CPU is an evolutionary enhancement of Intel's industry standard 8080A. It is 100% software compatible with the 8080A while offering the benefits of single power supply, higher integration, higher performance, and improved system timing.

The 8085 CPU is fully described in the Intel® **MCS-85™ User's Manual** so a detailed description will not be repeated here.

As the system block diagram shows, the 8085 derives its timing inputs directly from a crystal. In addition the 8085 drives the system with control signals available on-chip. No additional status decoding circuitry is required for most small- to medium-sized systems. The 8085 multiplexes its data bus with the low 8 bits of its address bus. The 8155 and 8355/8755 Memory I/O components in the kit are designed to be compatible with this bus structure, precluding the need for external bus latches.

Four vectored interrupt inputs are available in addition to the standard 8080A-type interrupt. There is also a serial input and serial output data line pair that is exercised under program control to provide the SDK-85's simple teletype I/O.

The basic clock frequency of the 8085 in the kit is 3.072 MHz (internally divided by 2 from the 6.144 MHz crystal input).

The 8155

The 8155 is a highly integrated chip designed for compatibility with the 8085's bus structure. It contains 256 bytes of static RAM memory, 22 programmable I/O lines, and a 14-bit timer/counter. The function of the 8155 is described in detail in the Intel **MCS-85 User's Manual.**

One 8155 is included with the SDK-85 kit and space for another has been provided on the circuit board. The RAM memory in the 8155 is available for storage of user programs as well as for temporary storage of information needed by system programs.

The 8155's timer is used by the SDK-85 monitor's Single Step routine to interrupt the processor following the execution of each instruction.

The 8355 & 8755

The 8355 and 8755 are two more chips specially designed for compatibility with 8085 systems. The 8355 contains 2048 bytes of mask programmed read only memory (ROM) and 16 I/O lines. The 8755 has an identical function and pinout to the 8355, but contains ultraviolet erasable and reprogrammable read only memory (EPROM) instead of the ROM.

The SDK-85 contains either one 8355 or one 8755 that is programmed with the system monitor. Space for a second 8755 or 8355 has been allocated on the PC board.

The 8279

The 8279 is a keyboard/display controller chip that handles the interface between the 8085 and the keypad and LED display on the SDK-85 board. The 8279 refreshes the display from an internal memory while scanning the keyboard to detect keyboard inputs. The 8279 is described in detail in the **MCS-85 User's Manual.**

The 8205

The basic SDK-85 also contains an 8205 chip (one-out-of-8 decoder) that decodes the 8085's memory address bits to provide chip enables for the 8155, the 8355/8755, and the 8279.

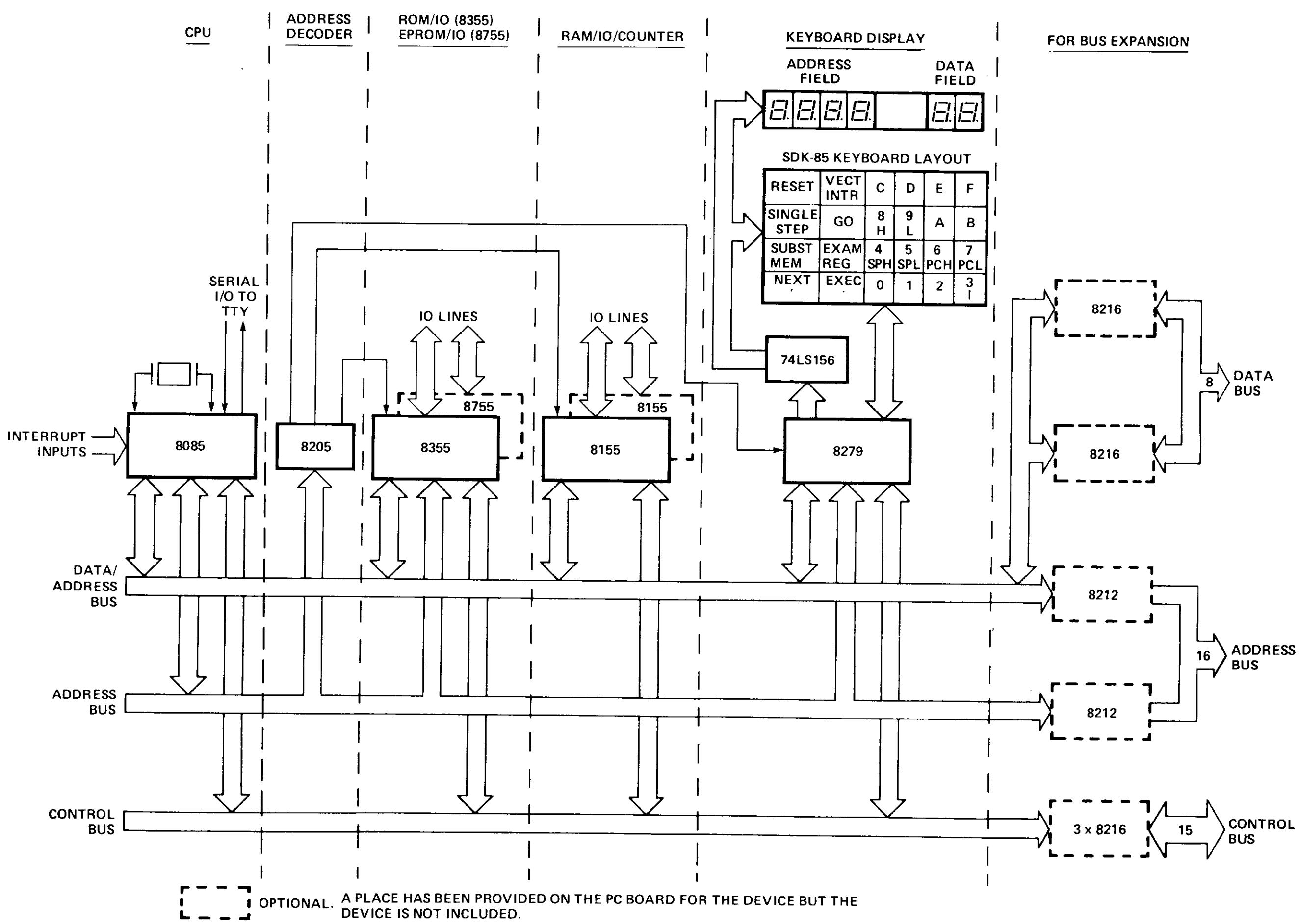


Figure 5-1 SDK-85 Functional Block Diagram

TABLE 5-1
8205 CHIP ENABLES

OUTPUT	ACTIVE ADDRESS RANGE	SELECTED DEVICE
CS0	0000-07FF	8755/8355 MONITOR ROM (A14)
CS1	0800-0FFF	8755/8355 EXPANSION ROM (A15)
CS2	1000-17FF	N/C
CS3	1800-1FFF	8279 KEYBOARD/DISPLAY CONTROLLER (A13)
CS4	2000-27FF	8155 BASIC RAM (A16)
CS5	2800-2FFF	8155 EXPANSION RAM (A17)
CS6	3000-37FF	N/C
CS7	3800-3FFF	N/C

AXX = IC# on schematic diagram in Appendix
N/C = not connected — avilable for user expansion

5-3 SDK-85 MEMORY ADDRESSING

Each memory/I/O chip in the basic SDK-85 System of Figure 5-1 is enabled by a signal coming from the 8205 address decoder. Table 5-1 lists each chip enable output accompanied by the address space over which it is active and the SDK-85 device that is selected.

Note that the 8279 is really an input/output device that is communicated with by the 8085 as though it were a series of memory locations.

The above chip enable table can be expanded to form a memory map that illustrates the active portions of the SDK memory (see Figure 5-2). Using the terminology of Figure 5-2, the basic SDK-85 with no additional memory/I/O chips provides the memory blocks marked MONITOR ROM and BASIC RAM. You must confine your programs to a subset of the space available in the BASIC RAM, the remainder of BASIC RAM being required for monitor storage locations. A list of the monitor-reserved RAM locations is provided in Table 5-2.

Note that RAM memory locations 20C2 through 20D0 are places for jump instructions pointing to the places in memory for the computer to go following the execution of an RST 5 instruction, an RST 6 instruction, an interrupt signal on the RST 6.5 input, etc. If you do not use any of these instructions or interrupt lines, then this RAM area is available for other programming.

When you add an expansion 8155 in the space provided on the SDK-85 board, the RAM locations shown in Figure 5-2 as EXPANSION RAM are made available for programming. The monitor reserves no space in the EXPANSION RAM, so all 256 locations are available for programming.

An extra 8355 or 8755 device when plugged into the appropriate spot on the board gives you program memory space in the area denoted EXPANSION ROM in the memory map.

The areas marked "FOLD BACK" in Figure 5-2 indicate address space that is unused, but unavailable for expansion, because these locations are multiple mappings of the basic locations.

TABLE 5-2
MONITOR-RESERVED RAM LOCATIONS

LOC.	CONTENTS	
20C2	User may place a JMP instr. to a RST 5 routine in locs 20C2 – 20C4.	Loaded by user
20C5	JMP to RST 6 routine	
20C8	JMP to RST 6.5 routine (hardwired user interrupt)	
20CB	JMP to RST 7 routine	
20CE	JMP to "VECT INTR" key routine	
20D1-20E8	Monitor Stack (temporary storage used by monitor)	
20E9	E Register	storage for user register images
20EA	D Register	
20EB	C Register	
20EC	B Register	
20ED	Flags	
20EE	A Register	
20EF	L Register	
20F0	H Register	
20F1	Interrupt Mask	
20F2	Prog. Cntr. – Low byte	
20F3	Prog. Cntr. – HI byte	
20F4	Stack Ptr. – Low byte	
20F5	Stack Ptr. – Hi byte	
20F6	Current Address	
20F8	Current Data	
20F9-20FC	Output buffer & Temp Locs.	
20FD	Register Pointer	
20FE	Input Buffer	
20FF	8155 Command/Status Register image	

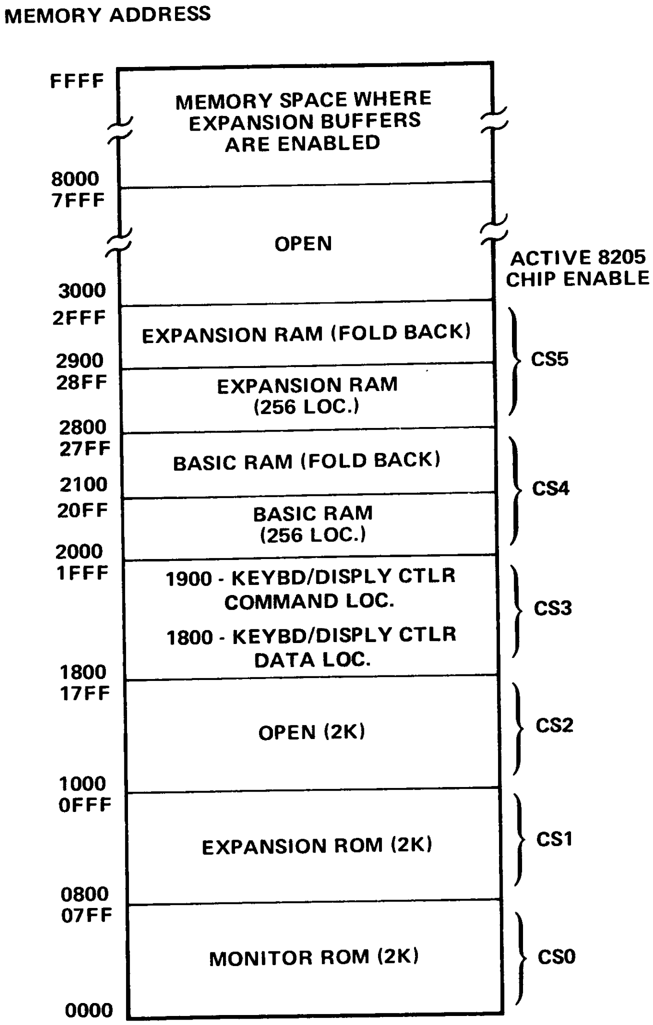


Figure 5-2 SDK-85 Memory Map

Any of the areas marked "OPEN" in Figure 5-2 are free for expansion. You may mount extra memory chips in the wire-wrap area of the SDK-85 board or on other circuit boards. The 8205 address decoder has 3 uncommitted chip select lines to allow the addition of three 2048-byte memory blocks without additional decoding circuitry.

If you want to expand on the basic SDK-85 you don't have to stick to the multiplexed-bus MCS 85 memory/I/O family. Mounting pads are present on the circuit board that accommodate an 8212 latch for address/data bus demultiplexing. To provide the current drive capability to operate much larger systems, spaces are also allocated for another 8212 to buffer the unmultiplexed half of the address and five 8216 buffer/drivers to buffer the data bus, and control signals. The function of these components is described in detail in the 8085 manual. The functional positioning of the optional latch, buffers, and drivers in the SDK-85 system structure is shown in Figure 5-1.

IMPORTANT:
As Figure 5-2 indicates, the optional expansion buffers leading to the SDK-85 board's prototyping area are enabled only over the address range 8000-FFFF. If you desire to use any of the "OPEN" expansion areas shown in Figure 5-2 (enabled by the 8205 chip selects), you will have to become familiar with the SDK-85 schematics at the back of this manual and implement custom modifications to the SDK-85 circuitry.

5-4 INPUT/OUTPUT PORT AND PERIPHERAL DEVICE ADDRESSING

As mentioned before, the 8155 and 8355/8755 that come with the SDK-85 Kit have on-board input/output ports. These ports are accessed using the IN and OUT instructions of the 8085. Each individual port being referenced has a unique 8-bit address. Table 5-3 contains all the port addresses for an expanded SDK-85 containing two 8155's and two 8355/8755's.

Please consult the **MCS-85 User's Manual** for the use of the various special purpose registers referred to in the table (Direction Registers, Command/Status Registers, etc.), and for complete instructions for exercising the memory-I/O chips (8155/8355/8755).

Hardware Note: The timer/counter of the first 8155 (RAM) is dedicated as a timer. It is hardwired to receive the 8085's system clock (3.072 MHz CLK) as its count input. This timer is used by the keyboard monitor's SINGLE STEP function, so you should beware of timer conflicts if you desire to count and use the SINGLE STEP function at the same time. (See paragraph 6-2.)

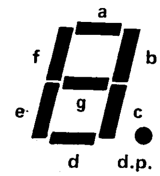
Accessing the 8279 Keyboard/Display Controller

As was mentioned in the memory addressing sections, the 8279 is a peripheral chip that is selected using memory-mapped I/O. Table 5-4 shows the two memory locations that are used to communicate with the 8279. Consult the **MCS-85 User's Manual** for detailed operating instructions.

TABLE 5-3
SDK-85 I/O PORT MAP

PORT	FUNCTION
00	Monitor ROM PORT A
01	Monitor ROM PORT B
02	Monitor ROM PORT A Data Direction Register
03	Monitor ROM PORT B Data Direction Register
08	Expansion ROM PORT A
09	Expansion ROM PORT B
0A	Expansion ROM PORT A Data Direction Register
0B	Expansion ROM PORT B Data Direction Register
20	BASIC RAM COMMAND/STATUS Register
21	BASIC RAM PORT A
22	BASIC RAM PORT B
23	BASIC RAM PORT C
24	BASIC RAM Low Order Byte of Timer Count
25	BASIC RAM High Order Byte of Timer Count
28	EXPANSION RAM COMMAND/STATUS Register
29	EXPANSION RAM PORT A
2A	EXPANSION RAM PORT B
2B	EXPANSION RAM PORT C
2C	EXPANSION RAM Low Order Byte of Timer Count
2D	EXPANSION RAM High Order Byte of Timer Count

The data format for character bytes being displayed by the 8279 is one bit corresponding to each of the seven LED segments plus one bit for the decimal point. Figure 5-3 shows the bit configuration.



MSB							LSB	
A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	← 8279 DISPLAY RAM OUTPUT DESIGNATION
d	c	b	a	d.p.	g	f	e	← SEGMENT

Figure 5-3 Data Format

The hardware is designed so that writing a zero into a bit position turns on the corresponding LED segment.

Example: a "4" would be represented as
1001 1001 = 99 (Hex)

These are six active LED displays available for use. They are configured in a four-place address field and a two-place data field as in Figure 5-4.

ADDRESS FIELD					DATA FIELD	
1	2	3	4		1	2

Figure 5-4 Display Configuration

TABLE 5-4
ACCESSING THE 8279
KEYBOARD DISPLAY CONTROLLER

LOCATION	READ/ WRITE	FUNCTION
1800	Read	Read Keyboard FIFO
	Write	Write Data to Display
1900	Read	Read Status Word
	Write	Write Command Word

The display digits are stored within the 8279 display RAM in the locations listed in Table 5-5.

TABLE 5-5

8279 DISPLAY RAM LOCATION	PURPOSE
0	Address digit 1
1	2
2	3
3	4
4	Data Digit 1
5	2
6	UNUSED
7	UNUSED

5-5 PROCESSOR INTERRUPT ALLOCATION

The 8085 has four Vector Interrupt input pins in addition to an 8080A-compatible interrupt input. The name of each interrupt and its function in the SDK-85 hardware is listed in Table 5-6.

The function of the on-chip interrupts is described in detail in the 8085 Manual.

TABLE 5-6
8085 ON-CHIP INTERRUPT ALLOCATION

INPUT	FUNCTION
RST 5.5	Dedicated to 8279
RST 6.5	Available User Interrupt
RST 7.5	VECT INTR button interrupt
TRAP	8155 Timer Interrupt
INTR	Available User Interrupt

5-6 THE SERIAL DATA INTERFACE

The SDK-85 has the capability of communicating with a teletype, using the 8085 serial input and serial output data lines (SID and SOD respectively) to send and receive the serial bit strings that encode data characters.

To send data to the teletype, the 8085 must toggle the SOD line in a set/reset fashion controlled by software timing routines in the SDK-85 monitor.

Input data is obtained by monitoring and timing changes in the level of the SID pin. Again, a monitor routine is called upon to do the job.

These teletype communications routines are accessible to the user.

Both subroutines communicate at a data rate of 110 baud, the standard rate for teletypewriters.

Since the 8085 serial input and output lines are designed for communicating with other integrated circuits, additional electronic circuitry is needed before they can be connected to a terminal. The TTY interface in the top right corner of the board allows the SDK-85 to be connected to any teletype that uses 20 mA "current-loop" input and output.

5-7 CONVERTER CIRCUIT FOR RS232C SERIAL PORT

If you are fortunate enough to have a CRT terminal that can operate at a 110-baud rate, and wish to use it with the SDK-85 computer, you may find that it is compatible only with "RS232c" voltage-level serial ports and not with current loops. If this is the case,

- ☐ Wire the MC1488 and MC1489 converter circuit (shown in Figure 5-5) into the wire-wrap area of the SDK-85 board.
- ☐ Remove R6, and connect the input line of the converter circuit to its lower pad. (You could put a switch in this line if you wanted to.)
- ☐ Open both the TTY and KEYBOARD jumpers, and connect the output line of the converter to the middle pad, which is strapping point 23. (If you are using a switch, one with a center off position could be used.)
- ☐ Connect your CRT as shown in Figure 5-5.
- ☐ Connect the 3 different voltages to the circuit.

5-8 ADDITIONAL INTERFACES

Additional interface considerations are discussed in Intel Application Note AP-29, which also describes a low-cost cassette tape-recorder interface, that can be added to your SDK-85 kit. AP-29 can be ordered by sending $1.00 to: Literature Department, Intel Corp., 3065 Bowers Ave., Santa Clara, Ca. 95051.

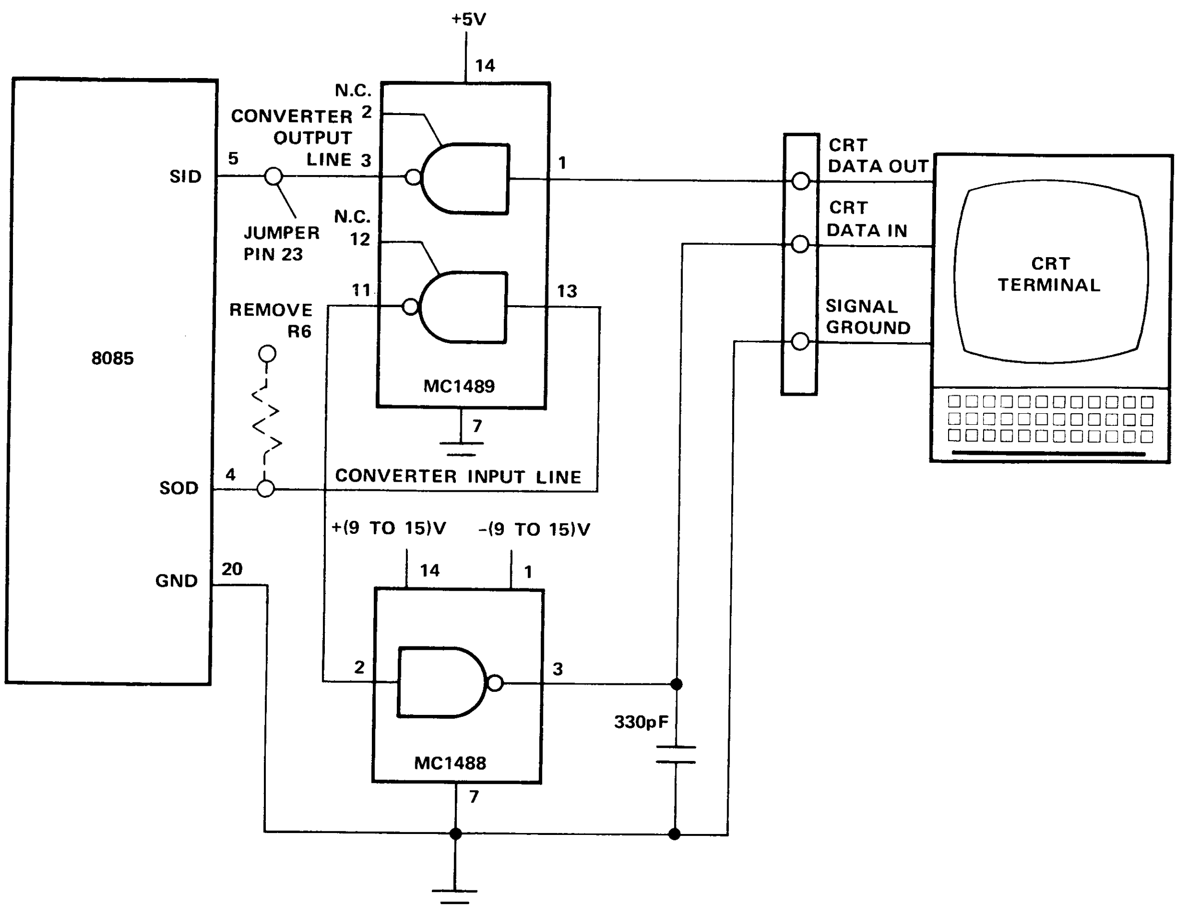


Figure 5-5 Modification for RS-232c Operation

CHAPTER 6
THE SOFTWARE

6-1 THE SDK-85 MONITOR

The SDK-85 monitor program provides utility functions employing either a teletypewriter or the kit's on-board keyboard and display as console. The program resides in 2k (k = 1024) bytes of the ROM memory, between location 0 and location 7FF. the routines that service each console device are independent; the two devices do not function simultaneously. You may select either the keyboard and display or the teletypewriter as the console device by actuating a switch (not furnished) or by changing strapping connections. Both can be used to perform substantially the same tasks. (See Chapter 4.)

6-2 PROGRAMMING HINTS

Stack Pointer

The 8085 makes use of a 16-bit internal register called the Stack Pointer to point to an area of memory called the stack. The 8085's stack is used for saving many things, such as memory addresses for returns from subroutines.

It is important always to define the stack pointer at the beginning of your program to avoid storing data in the wrong place. Locations 20C2 through 20D0 in RAM are reserved by the monitor for jump instructions when all interrupts are used. Thus, you should set the stack pointer initially at 20C2 (by the use of the program instruction LXI SP, 20C2H (31 C2 20), the keyboard command [EXAM REG] [4 SPH](20) [NEXT] (C2) [EXEC], or the teletypewriter "XS" command) in order to keep your own stack clear of data and programs you want to protect. If less than the full complement of interrupts is utilized, some or all of the unused space above 20C2 can be allocated to stack as described above. Remember that the stack must still occupy an unbroken string of contiguous memory locations.

RAM-I/O Command Status Register (CSR)

The basic 8155 command status register (port 20) is used to set up the on-chip I/O ports and timer. It can only be written to; it cannot be read. You can write to this register in your programs, but there is a precaution you should take: at any time when you write to the CSR in the basic RAM, you should also write the same pattern to RAM location 20FF. The reason is this: The [SINGLE STEP] command causes the monitor to change the CSR in order to set up the timer for execution of the command. If it is not told what value you previously put there (by saving the value in 20FF), that value will inevitably be overwritten and lost. Following each single step, the monitor reads location 20FF, logically ORs its timer command to the content of that location, and writes the CSR with the new command, thereby retrieving your previous configuration.

Access to Monitor Routines

You may "borrow" several of the SDK-85 monitor routines to simplify your programming task. Table 6-1 provides descriptions and calling addresses for these routines.

6-3 PROGRAMMING EXAMPLES

The programming examples presented at the end of this chapter demonstrate how to use the monitor routines to operate the keyboard and display.

TABLE 6-1
MONITOR ROUTINE CALLING ADDRESSES

Calling Address	Mnemonic	Description
07FD	CI	**Console Input** This routine returns a character (in ASCII code – see 8085/8080 reference card for codes) received from the teletype to the caller in the A register. The A register and CPU condition codes are affected by this operation
07FA	CO	**Console Output** This routine transmits a character (in ASCII code), passed from the caller in the C register, to the teletypewriter. The A and C registers, and the CPU condition codes are affected.
05EB	CROUT	**Carriage Return, Line Feed** CROUT sends carriage return and line feed characters to the teletype. The contents of the A, B, and C registers are destroyed and the CPU condition codes are affected.
06C7	NMOUT	**Hex Number Printer** NMOUT converts the 8-bit unsigned integer in the A register into 2 ASCII characters representing the 2 hex digits and prints the two digits on the teletypewriter. The contents of the A, B and C registers and the condition code flags are affected.
0363	UPDAD	**Update Address** Update address field of the display. The contents of the D-E register pair are displayed in the address field of the display. The contents of all the CPU registers and flags are affected.
036E	UPDDT	**Update Data** Update data field of the display. The contents of the A register are displayed in hex notation in the data field of the display. The contents of all of the CPU registers and flags are affected.
02E7	RDKBD	**Read Keyboard** This routine waits until a character is entered on the hex keypad and upon return places the value of the character in the A register. The A, H, and L registers and the flag flip flops are affected. NOTE: For RDKBD to work correctly, you must first: 1. Unmask RST 5.5 using the SIM instruction.
05F1	DELAY	**Time Delay** This routine takes the 16-bit contents of register pair DE and counts down to zero, then returns to the calling program. The A, D, and E registers and the flags are affected.

TABLE 6-1
MONITOR ROUTINE CALLING ADDRESSES (CONT'D)

Calling Address	Mnemonic	Description
02B7	OUTPT	**Output Characters to Display** The routine sends characters to the display with the parameters set up by registers A, B, H and L. Reg A = 0 = use address field = 1 = use data field Reg B = 0 = decimal point off = 1 = decimal point at right edge of field Reg HL = starting address of characters to to sent.

Character Displayed	Hexadecimal memory content pointed to by the HL register
0	00
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
A	0A
b	0B
C	0C
d	0D
E	0E
F	0F
H	10
L	11
P	12
I	13
r	14
S	05
Blank	15

PROGRAM EXAMPLE – RDKBD

After executing [GO] 2000, the program waits until a key is pressed. Then the value of the key is placed in the A register and the monitor is restarted. Use [EXAM REG] to see that the key value is now in the A register.

ADDRESS	DATA	SYMBOLIC	COMMENTS
2000	31	LXI SP, 20C2H	; define stack pointer
2001	C2		
2002	20		
2003	3E	MVI A, 08H	
2004	08		
2005	30	SIM	; unmask interrupt
2006	CD	CALL RDKBD	; read keyboard value
2007	E7		; into Reg A
2008	02		
2009	CF	RST 1	; break point, go back to monitor

PROGRAM EXAMPLE – UPDDT

Display FF in data field of display.

ADDRESS	DATA	SYMBOLIC	COMMENTS
2000	31	LXI SP, 20C2H	; define stack pointer
2001	C2		
2002	20		
2003	3E	MVI A, FFH	; load FF into Reg A
2004	FF		
2005	CD	CALL UPDDT	; output Reg A to data field
2006	6E		
2007	03		
2008	76	HLT	; HALT

To change the display value use [SUBST MEM] to vary the content of location 2004

PROGRAM EXAMPLE – RDKBD, UPDDT

Putting the two preceding examples together into one program causes the display to show the key value.

ADDRESS	DATA	SYMBOLIC	COMMENTS
2000	31C220	LXI SP, 20C2H	; define stack pointer
2003	3E08	MVI A, 08H	
2005	30	SIM	; unmask interrupt
2006	CDE702	LOOP: CALL RDKBD	; read keyboard value into Reg A
2009	CD6E03	CALL UPDDT	; output Reg A to data field
200C	C30620	JMP LOOP	; keep looping

PROGRAM EXAMPLE – ADD TWO NUMBERS IN HEX NOTATION

This program is an adaptation of the program above. The computer reads in two one-digit numbers using RDKBD. Then it adds them, and displays the sum (base 16) on the LED display using UPDDT.

ADDRESS	DATA		SYMBOLIC	COMMENTS
2000	31C220		LXI SP, 20C2H	; initialize stack pointer
2003	3E08		MVI A, 08H	
2005	30		SIM	; unmask interrupts
2006	CDE702	LOOP:	CALL RDKBD	; get first number
2009	47		MOV B,A	; save number in B reg.
200A	CDE702		CALL RDKBD	; get second number
200D	80		ADD B	; add the two numbers
200E	CD6E03		CALL UPDDT	; display the sum
2011	C30620		JMP LOOP	; keep looping

Note: for decimal (base10) addition of digits 0-9, insert the DAA instruction (opcode 27) between ADD B and CALL UPDDT in the above program.
Additional Suggestion: Try modifying this program to perform 2-digit decimal number addition. (Hint: use the 8085's RLC instruction.)

PROGRAM EXAMPLE – 4-DIGIT HEX COUNTER

This program displays a 4-digit hexadecimal (base 16) count in the address field of the display using the UPDAD routine from the monitor.

ADDRESS	DATA		SYMBOLIC	COMMENTS
2000	31C220		LXI SP 20C2	; initialize stack pointer
2003	13	LOOP:	INX D	; add 1 to the 16-bit count
2004	D5		PUSH D	; save the count in the stack
2005	CD6303		CALL UPDAD	; display the count
2008	110018		LXID, 1800	; set delay count
200B	CDF105		CALL DELAY	; wait out the delay
200E	D1		POP D	; restore the count to D & E regs
200F	C30320		JMP LOOP	; keep counting

PROGRAM EXAMPLE – DECIMAL COUNTER

The following program displays a count in the data field of the display. The count may be stopped by pressing the VECT INTR button. The count resumes when any other key (except RESET) is pressed. The "E" in the address field of the display signifies that a user program is executing.

ADDRESS	DATA	SYMBOLIC	COMMENTS
2000	31	LXI SP, 2080H	; INITIALIZE STACK POINTER.
2001	80		
2002	20		
2003	3E	MVI A, 08	; USE THE 8085's SIM INSTR TO
2004	08		; ENABLE THE VECT INTR BUTTON.
2005	30	SIM	
2006	FB	LOOP: EI	
2007	78	MOV A, B	
2008	3C	INR A	; INCREMENT AND ADJUST THE COUNT
2009	27	DAA	; FOR DECIMAL COUNTING.
200A	47	MOV B, A	
200B	C5	PUSH B	
200C	CD	CALL UPDDT	; DISPLAY COUNT IN DATA FIELD OF
200D	6E		; DISPLAY.
200E	03		
200F	16	MVI D, 18H	
2010	18		
2011	CD	CALL DELAY	; WAIT OUT A PROGRAMMABLE DELAY
2012	F1		; PERIOD BEFORE CONTINUING.
2013	05		
2014	C1	POP B	
2015	C3	JMP LOOP	; GO BACK TO THE BEGINNING.
2016	06		
2017	20		
—			
20CE	FB	EI	; CONTROL BRANCHES TO LOCATION
			; 20CE WHEN VECT INTR IS PRESSED.
20CF	76	HLT	; WAIT HERE FOR KEY DEPRESSION.
20D0	C9	RET	; RESUME THE COUNT.

To execute the program, type in GO 2000 EXEC.

Try to stop the count right at 00 using the VECT INTR key.

Change the speed of the count by using SUBST MEM to vary the contents of location 2010.

Additional Suggestions:

This counter can be turned into a digital stopwatch second counter by *inserting* the following instructions between DAA and MOV B, A in the above program:

200A	FE60	CPI A, 60	; check to see if count = 60
200C	C21020	JNZ 2010	; continue if count ≠ 60.
200F	AF	XRA A	; if count = 60 then set the count = 0

In addition, you will have to insert another MVI D and CALL DELAY before POP B and vary both delay counts in order to get exactly one second between counts on the LED display.

Additional Programming Idea: Expand on the digital stopwatch program by displaying hours and minutes in the address field of the LED display.

PROGRAM EXAMPLE – FLASH HELP

Load into Locations 2000 through 2007 (use the Substitute Memory command) the following data: 10, OE, 11, 12, 15, 15, 15, 15. Then load and execute the following program ([GO] 2010 [EXEC]). The display will flash "HELP".

ADDRESS	DATA	SYMBOLIC	COMMENTS
2010	31C220	LXI SP, 20C2H	; define stack pointer
2013	3E01	MVI A, 1	; use data field
2015	0600	MVI B, 0	; no decimal indicator
2017	210620	LXI H, 2006H	; use characters starting
			; at Location 2006
201A	CDB702	CALL OUTPT	; output the two characters
			; to data field
		DPY:	
201D	3E00	MVI A, 0	; use address field
201F	0600	MVI B, 0	; no decimal indicator
2021	210020	LXI H, 2000H	; use characters starting
			; at Location 2000
2024	CDB702	CALL OUTPT	; output the four characters
			; to address field
		;	
2027	11FFFF	LXI D, 0FFFFH*	; set up delay value
202A	CDF105	CALL DELAY	; time delay
		;	
202D	3E00	MOV A, 0	;
202F	0600	MOV B, 0	;
2031	210420	LXI H, 2004H	; output BLANKS to
2034	CDB702	CALL OUTPT	; Display
		;	
2037	11FFFF	LXI D, 0FFFFH	;
203A	CDF105	CALL DELAY	; time Delay
203D	C31D20	JMP DPY	; REPEAT

*Delay time proportional to value. Any number from 1 through FFFF may be chosen.

Additional Suggestions:

You may select any other 4-letter word from the characters on p. 6-3 and place the hex codes for those letters in memory locations 2000-2003. Then restart the program from location 2010 and your new word will flash on the display.

In addition, you may place the hex codes from p. 6-3 for a 2-letter word (like "HI") in memory locations 2004 and 2005, and the 2-letter word will flash in between the flashes of the 4-letter word.

PROGRAM EXAMPLE – USING THE 8155 AND 8355 DEVICE OUTPUT PORTS

ADDRESS	DATA		SYMBOLIC	COMMENTS
2000	31C220		LXI SP, 20C2H	; initialize stack pointer
2003	3E03		MVI A, 03	; put 8155 command in A reg.
2005	D320		OUT 20H	; program the 8155 CSR
2007	3EFF		MVI A, FF	; put 8355 DDR value in A reg.
2009	D302		OUT 02	; program PORT A DDR
200B	D303		OUT 03	; program PORT B DDR
200D	03	LOOP:	INXB	; increment 16-bit count
200E	79		MOV A, C	;
200F	D321		OUT 21	; send low byte of count
2011	D300		OUT 0	; to 8155 PORT A and
				; to 8355 port A
2013	78		MOV A, B	;
2014	D322		OUT 22	; send hi byte of
				; count to 8155 port B
2016	D301		OUT 01	; send hi byte of
				; count to 8355 port B
2018	C30D20		JMP LOOP	; loop back.

This program is an example showing how to configure the input/output ports of the 8155 and 8355 devices as output ports. The command register of the 8155 is loaded with the value 03 at the beginning of the program to signify that both 8155 ports A and B will be outputs. Likewise, both ports A and B of the 8355 are programmed to be outputs by writing all one's (FF) to both Data Direction Registers in the 8355.

The program increments a 16-bit binary count and sends the count out through the ports of the 8155 and 8355. If you have a logic probe or oscilloscope, you can look at the corresponding outputs on connector pads J3 and J4 on the SDK-85 PC board.

APPENDIX A
MONITOR LISTING

ASM80 :F1:SDK85.SRC MACROFILE MOD85 XREF PRINT(:F1:SDK85.LST)


```
LOC  OBJ              SOURCE STATEMENT

                 ;***********************************************************************
                 ;
                 ;                       PROGRAM: SDK-85 MONITOR     VER 2.1
                 ;
                 ;                       COPYRIGHT (C) 1977
                 ;                       INTEL CORPORATION
                 ;                       3065 BOWERS AVENUE
                 ;                       SANTA CLARA, CALIFORNIA  95051
                 ;
                 ;***********************************************************************
                 ;
                 ; ABSTRACT
                 ; ========
                 ;
                 ; THIS PROGRAM IS A SMALL MONITOR FOR THE INTEL 8085 KIT AND
                 ; PROVIDES A MINIMUM LEVEL OF UTILITY FUNCTIONS FOR THE USER EMPLOYING
                 ; EITHER AN INTER-ACTIVE CONSOLE (I.E. TELETYPE) OR THE KIT'S
                 ; KEYBOARD/LED DISPLAY. THE KEYBOARD MONITOR ALLOWS THE USER TO PERFORM
                 ; SUCH FUNCTIONS AS MEMORY AND REGISTER MANIPULATION, PROGRAM LOADING,
                 ; PROGRAM EXECUTION, INTERRUPTION OF AN EXECUTING PROGRAM, AND
                 ; SYSTEM RESET.
                 ;
                 ; PROGRAM ORGANIZATION
                 ; ======= ============
                 ;
                 ; THE PROGRAM IS ORGANIZED AS FOLLOWS :-
                 ;       1) COLD START ROUTINE (RESET)
                 ;       2) WARM START - REGISTER SAVE ROUTINE
                 ;       3) INTERRUPT VECTORS
                 ;       4) KEYBOARD MONITOR
                 ;       5) TTY MONITOR
                 ;       6) LAYOUT OF RAM USAGE
                 ;
                 ; THE KEYBOARD MONITOR BEGINS WITH THE COMMAND RECOGNIZER, FOLLOWED BY
                 ; THE COMMAND ROUTINE SECTION, UTILITY ROUTINE SECTION AND MONITOR
                 ; TABLES. THE COMMAND AND UTILITY ROUTINES ARE IN ALPHABETICAL ORDER
                 ; WITHIN THEIR RESPECTIVE SECTIONS.
                 ; THROUGHOUT THE KEYBOARD MONITOR, A COMMENT FIELD BEGINNING
                 ; WITH "ARG - " INDICATES A STATEMENT WHICH LOADS A VALUE INTO
                 ; A REGISTER AS AN ARGUMENT FOR A FUNCTION. WHEN THE DESIRED VALUE
                 ; LIST OF KEYBOARD MONITOR ROUTINES
                 ; ==== == ======== ======= ========
                 ;
                 ; CMMND
                 ; -----
                 ; EXAM
                 ; GOCMD
                 ; SSTEP
                 ; SUBST
                 ; -----
                 ; CLEAR
                 ; CLDIS
                 ; CLDST
                 ; DISPC
                 ; ERR
                 ; GTHEX
                 ; HXDSP
                 ; ININT
                 ; INSDG
                 ; NXTRG
                 ; OUTPT
                 ; RDKBD
                 ; RETF
                 ; RETT
                 ; RGLOC
                 ; RSTOR
                 ; SETRG
                 ; UPDAD
                 ; UPDDT
                 ;
                         NAME    SDK85
                 ;
                 ;***********************************************************************
                 ;
                 ;                       SET CONDITIONAL ASSEMBLY FLAG
                 ;
                 ;***********************************************************************
                 ;
                 ;
0000             WAITS   SET     0          ;0=NO WAIT STATES
                                            ;1=A WAIT STATE IS GENERATED FOR EVERY M CYCLE
                                            ;THE APPROPRIATE DELAY TIME MUST BE USED FOR
                                                    ;TTY DELAY OR SET UP SINGLE
                                                    ;STEP TIMER FOR EACH CASE
                 ;
                 ;
                 ;***********************************************************************
                 ;
                 ;                       MONITOR EQUATES
                 ;
                 ;***********************************************************************
                 ;
2000             RAMST   EQU     2000H   ; START ADDRESS OF RAM - THIS PROGRAM ASSUMES
                 ; THAT 256 BYTES OF RANDOM ACCESS MEMORY BEGIN AT THIS ADDRESS.
                 ; THE PROGRAM USES STORAGE AT THE END OF THIS SPACE FOR VARIABLES,
                 ; SAVING REGISTERS AND THE PROGRAM STACK
                 ;
```

```
 LOC  OBJ         SOURCE STATEMENT

0017              RMUSE   EQU     23       ; RAM USAGE - CURRENTLY, 23 BYTES ARE USED FOR
                                           ; /SAVING REGISTERS AND VARIABLES
                  ;
0018              SKLN    EQU     24       ; MONITOR STACK USAGE - MAX OF 12 LEVELS
                  ;
000F              UBRLN   EQU     15       ; 5 USER BRANCHES - 3 BYTES EACH
                  ;
0000              ADFLD   EQU     0        ; INDICATES USE OF ADDRESS FIELD OF DISPLAY
0090              ADISP   EQU     90H      ; CONTROL CHARACTER TO INDICATE OUTPUT TO
                                           ; /ADDRESS FIELD OF DISPLAY
1900              CNTRL   EQU     1900H    ; ADDRESS FOR SENDING CONTROL CHARACTERS TO
                                           ; /DISPLAY CHIP
0011              COMMA   EQU     11H      ; COMMA FROM KEYBOARD
0000              CSNIT   EQU     0        ; INITIAL VALUE FOR COMMAND STATUS REGISTER
0020              CSR     EQU     20H      ; OUTPUT PORT FOR COMMAND STATUS REGISTER
0094              DDISP   EQU     94H      ; CONTROL CHARACTER TO INDICATE OUTPUT TO
                                           ; /DATA FIELD OF DISPLAY
0001              DOT     EQU     1        ; INDICATOR FOR DOT IN DISPLAY
1800              DSPLY   EQU     1800H    ; ADDRESS FOR SENDING CHARACTERS TO DISPLAY
0001              DTFLD   EQU     1        ; INDICATES USE OF DATA FIELD OF DISPLAY
0008              DTMSK   EQU     08H      ; MASK FOR TURNING ON DOT IN DISPLAY
0080              EMPTY   EQU     80H      ; HIGH ORDER 1 INDICATES EMPTY INPUT BUFFER
00CC              KBNIT   EQU     0CCH     ; CONTROL CHARACTER TO SET DISPLAY OUTPUT TO
                                           ; /ALL ONES DURING BLANKING PERIOD
0000              KMODE   EQU     0        ; CONTROL CHAR. TO SET KEYBOARD/DISPLAY MODE
                                           ; (2 KEY ROLLOVER, 8 CHARACTER LEFT ENTRY)
20E9              MNSTK   EQU     RAMST + 256 - RMUSE     ;START OF MONITOR STACK
0000              NODOT   EQU     0        ; INDICATOR FOR NO DOT IN DISPLAY
                  ;NUMC - DEFINED LATER    ; NUMBER OF COMMANDS
                  ;NUMRG - DEFINED LATER   ; NUMBER OF REGISTER SAVE LOCATIONS
0010              PERIO   EQU     10H      ; PERIOD FROM KEYBOARD
00FB              PRMPT   EQU     0FBH     ; PROMPT CHARACTER FOR DISPLAY (DASH)
0040              READ    EQU     40H      ; CONTROL CHARACTER TO INDICATE INPUT FROM
                                           ; /KEYBOARD
0025              TIMHI   EQU     25H      ; OUTPUT PORT FOR HIGH ORDER BYTE OF TIMER VALUE
0024              TIMLO   EQU     24H      ; OUTPUT PORT FOR LOW ORDER BYTE OF TIMER VALUE
0040              TMODE   EQU     40H      ; TIMER MODE - SQUARE WAVE, AUTO RELOAD
00C0              TSTRT   EQU     0C0H     ; START TIMER
000E              UNMSK   EQU     0EH      ; UNMASK INPUT INTERRUPT
20C2              USRBR   EQU     RAMST + 256 - (RMUSE + SKLN + UBRLN)      ; START OF USER
                                                           ; /BRANCH LOCATIONS
                          IF      1-WAITS ;TIMER VALUE FOR SINGLE STEP IF NO WAIT STATE
00C5              TIMER   EQU     197
                          ENDIF
                          IF      WAITS    ;TIMER VALUE FOR SINGLE STEP IF ONE WAIT STATE INSERTED
                  TIMER   EQU     237
                          ENDIF
                  ;
                  ;*********************************************************************
                  ;
                  ;                         MONITOR MACROS
                  ;
                  ;*********************************************************************
                  ;
                  TRUE    MACRO   WHERE    ; BRANCH IF FUNCTION RETURNS TRUE
                          JC      WHERE
                          ENDM
                  ;
                  FALSE   MACRO   WHERE    ; BRANCH IF FUNCTION RETURNS FALSE
                          JNC     WHERE
                          ENDM
                  ;
                  ;
                  ;*********************************************************************
                  ;
                  ; ***** "RESET" KEY ENTRY POINT - COLD START
                  ; ***** RST 0 ENTRY POINT
                  ;
0000 3E00                 MVI     A,KMODE  ; GET CONTROL CHARACTER
0002 320019               STA     CNTRL    ; SET KEYBOARD/DISPLAY MODE
0005 C3F101               JMP     CLDST    ; GO FINISH COLD START
                  CLDBK:                   ; THEN JUMP BACK HERE
                  ;
                  ; ***** RST 1 ENTRY POINT - WARM START
                  ;
0008                      ORG     8
                  ;       SAVE REGISTERS
0008 22EF20               SHLD    LSAV     ; SAVE H & L REGISTERS
000B E1                   POP     H        ; GET USER PROGRAM COUNTER FROM TOP OF STACK
000C 22F220               SHLD    PSAV     ; /AND SAVE IT
000F F5                   PUSH    PSW
0010 E1                   POP     H
0011 22ED20               SHLD    FSAV     ; SAVE FLIP/FLOPS & REGISTER A
0014 210000               LXI     H,0      ; CLEAR H & L
0017 39                   DAD     SP       ; GET USER STACK POINTER
0018 22F420               SHLD    SSAV     ; /AND SAVE IT
001B 21ED20               LXI     H,BSAV+1 ; SET STACK POINTER FOR SAVING
001E F9                   SPHL             ; /REMAINING REGISTERS
001F C5                   PUSH    B        ; SAVE B & C
0020 D5                   PUSH    D        ; SAVE D & E
0021 C33F00               JMP     RES10    ; LEAVE ROOM FOR VECTORED INTERRUPTS
                  ;
                  ; ***** TIMER INTERRUPT (TRAP) ENTRY POINT (RST 4.5)
0024                      ORG     24H
0024 C35701               JMP     STP25    ; BACK TO SINGLE STEP ROUTINE
                  ;
                  ; ***** RST 5 ENTRY POINT
                  ;
0028                      ORG     28H
```

```
LOC  OBJ         SEQ       SOURCE STATEMENT

0028 C3C220                  JMP     RSET5   ; BRANCH TO RST 5 LOCATION IN RAM
                       ;
                       ; ***** INPUT INTERRUPT ENTRY POINT (RST 5.5)
                       ;
002C                         ORG     2CH
002C C38E02                  JMP     ININT   ; BRANCH TO INPUT INTERRUPT ROUTINE
                       ;
                       ; ***** RST 6 ENTRY POINT
                       ;
0030                         ORG     30H
0030 C3C520                  JMP     RSET6   ; BRANCH TO RST 6 LOCATION IN RAM
                       ;
                       ; ***** HARD WIRED USER INTERRUPT ENTRY POINT (RST 6.5)
                       ;
0034                         ORG     34H
0034 C3C820                  JMP     RST65   ; BRANCH TO RST 6.5 LOCATION IN RAM
                       ;
                       ; ***** RST 7 ENTRY POINT
                       ;
0038                         ORG     38H
0038 C3CB20                  JMP     RSET7   ; BRANCH TO RST 7 LOCATION IN RAM
                       ;
                       ; ***** "VECTORED INTERRUPT" KEY ENTRY POINT (RST 7.5)
003C                         ORG     3CH
003C C3CE20                  JMP     USINT   ; BRANCH TO USER INTERRUPT LOCATION IN RAM
                       ;
                       RES10:  ; CONTINUE SAVING USER STATUS
003F 20                      RIM             ; GET USER INTERRUPT STATUS AND INTERRUPT MASK
0040 E60F                    ANI     0FH     ; KEEP STATUS & MASK BITS
0042 32F120                  STA     ISAV    ; SAVE INTERRUPT STATUS & MASK
0045 3E0E                    MVI     A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE
0047 30                      SIM
0048 F3                      DI              ; INTERRUPTS DISABLED WHILE MONITOR IS RUNNING
                                             ; (EXCEPT WHEN WAITING FOR INPUT)
0049 20                      RIM             ; TTY OR KEYBOARD MONITOR ?
004A 07                      RLC             ; IS TTY CONNECTED ?
004B DAFA03                  JC      GO      ; YES - BRANCH TO TTY MONITOR
                                             ; NO - ENTER KEYBOARD MONITOR
                       ;
                       ;*******************************************************************
                       ;
                       ;                         BEGINNING OF KEYBOARD MONITOR CODE
                       ;
                       ;*******************************************************************
                       ;
                       ;       OUTPUT SIGN-ON MESSAGE
004E AF                      XRA     A       ; ARG - USE ADDRESS FIELD OF DISPLAY
004F 0600                    MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0051 21A603                  LXI     H,SGNAD ; ARG - GET ADDRESS OF ADDRESS FIELD PORTION OF
                                             ; /SIGN-ON MESSAGE
0054 CDB702                  CALL    OUTPT   ; OUTPUT SIGN-ON MESSAGE TO ADDRESS FIELD
0057 3E01                    MVI     A,DTFLD ; ARG - USE DATA FIELD OF DISPLAY
0059 0600                    MVI     B,NODOT ; ARG - NO DOT IN DATA FIELD
005B 21AA03                  LXI     H,SGNDT ; ARG - GET ADDRESS OF DATA FIELD PORTION OF
                                             ; /SIGN-ON MESSAGE
005E CDB702                  CALL    OUTPT   ; OUTPUT SIGN-ON MESSAGE TO DATA FIELD
0061 3E80                    MVI     A,EMPTY
0063 32FE20                  STA     IBUFF   ; SET INPUT BUFFER EMPTY FLAG
                       ;
                       ;*******************************************************************
                       ;
                       ; FUNCTION: CMMND - COMMAND RECOGNIZER
                       ; INPUTS: NONE
                       ; OUTPUTS: NONE
                       ; CALLS: RDKBD,ERR,SUBST,EXAM,GOCMD,SSTEP
                       ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                       ;
                       CMMND:
0066 21E920                  LXI     H,MNSTK ; INITIALIZE MONITOR STACK POINTER
0069 F9                      SPHL
                                             ; OUTPUT PROMPT CHARACTER TO DISPLAY
006A 210019                  LXI     H,CNTRL ; GET ADDRESS FOR CONTROL CHARACTER
006D 3690                    MVI     M,ADISP ; OUTPUT CONTROL CHARACTER TO USE ADDRESS FIELD
006F 25                      DCR     H       ; ADDRESS FOR OUTPUT CHARACTER
0070 36FB                    MVI     M,PRMPT ; OUTPUT PROMPT CHARACTER
0072 CDE702                  CALL    RDKBD   ; READ KEYBOARD
0075 010400                  LXI     B,NUMC  ; COUNTER FOR NUMBER OF COMMANDS IN C
0078 217803                  LXI     H,CMDTB ; GET ADDRESS OF COMMAND TABLE
                       CMD10:
007B BE                      CMP     M       ; RECOGNIZE THE COMMAND ?
007C CA8700                  JZ      CMD15   ; YES - GO PROCESS IT
007F 23                      INX     H       ; NO - NEXT COMMAND TABLE ENTRY
0080 0D                      DCR     C       ; END OF TABLE ?
0081 C27B00                  JNZ     CMD10   ; NO - GO CHECK NEXT ENTRY
                                             ; YES - COMMAND UNKNOWN
0084 C31502                  JMP     ERR     ; DISPLAY ERROR MESSAGE AND GET ANOTHER COMMAND
                       CMD15:
0087 217C03                  LXI     H,CMDAD ; GET ADDRESS OF COMMAND ADDRESS TABLE
008A 0D                      DCR     C       ; ADJUST COMMAND COUNTER
                             ; COUNTER ACTS AS POINTER TO COMMAND ADDRESS TABLE
008B 09                      DAD     B       ; ADD POINTER TO TABLE ADDRESS TWICE BECAUSE
008C 09                      DAD     B       ; TABLE HAS 2 BYTE ENTRIES
008D 7E                      MOV     A,M     ; GET LOW ORDER BYTE OF COMMAND ADDRESS
008E 23                      INX     H
008F 66                      MOV     H,M     ; GET HIGH ORDER BYTE OF COMMAND ADDRESS IN H
0090 6F                      MOV     L,A     ; PUT LOW ORDER BYTE IN L
                             ; COMMAND ROUTINE ADDRESS IS NOW IN H & L
0091 E9                      PCHL            ; BRANCH TO ADDRESS IN H & L
                       ;
```

```
 LOC  OBJ              SOURCE STATEMENT

                 ;*******************************************************************
                 ;
                 ;                        COMMAND ROUTINES
                 ;
                 ;*******************************************************************
                 ;
                 ; FUNCTION: EXAM - EXAMINE AND MODIFY REGISTERS
                 ; INPUTS: NONE
                 ; OUTPUTS: NONE
                 ; CALLS: CLEAR,SETRG,ERR,RGNAM,RGLOC,UPDDT,GTHEX,NXTRG
                 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                 ;
                 EXAM:
0092 0601                MVI     B,DOT    ; ARG - DOT IN ADDRESS FIELD OF DISPLAY
0094 CDD701              CALL    CLEAR    ; CLEAR DISPLAY
0097 CD4403              CALL    SETRG    ; GET REGISTER DESIGNATOR FROM KEYBOARD AND
                                          ;/SET REGISTER POINTER ACCORDINGLY
                                          ; WAS CHARACTER A REGISTER DESIGNATOR?
                         FALSE   ERR      ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
009A D21502              JNC     ERR
                 EXM05:
009D CD0903              CALL    RGNAM    ; OUTPUT REGISTER NAME TO ADDRESS FIELD
00A0 CDFC02              CALL    RGLOC    ; GET REGISTER SAVE LOCATION IN H & L
00A3 7E                  MOV     A,M      ; GET REGISTER CONTENTS
00A4 32F820              STA     CURDT    ; STORE REGISTER CONTENTS AT CURRENT DATA
00A7 0601                MVI     B,DOT    ; ARG - DOT IN DATA FIELD
00A9 CD6B03              CALL    UPDDT    ; UPDATE DATA FIELD OF DISPLAY
00AC 0601                MVI     B,DTFLD  ; ARG - USE DATA FIELD OF DISPLAY
00AE CD2B02              CALL    GTHEX    ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?
                         FALSE   EXM10    ; NO - DO NOT UPDATE REGISTER CONTENTS
00B1 D2B800              JNC     EXM10
00B4 CDFC02              CALL    RGLOC    ; YES - GET REGISTER SAVE LOCATION IN H & L
00B7 73                  MOV     M,E      ; UPDATE REGISTER CONTENTS
                 EXM10:
00B8 FE10                CPI     PERIO    ; WAS LAST CHARACTER A PERIOD ?
00BA CAE901              JZ      CLDIS    ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
00BD FE11                CPI     COMMA    ; WAS LAST CHARACTER ',' ?
00BF C21502              JNZ     ERR      ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
00C2 CDA802              CALL    NXTRG    ; YES - ADVANCE REGISTER POINTER TO
                                          ;/NEXT REGISTER
                                          ; ANY MORE REGISTERS ?
                         TRUE    EXM05    ; YES - CONTINUE PROCESSING WITH NEXT REGISTER
00C5 DA9D00              JC      EXM05
00C8 C3E901              JMP     CLDIS    ; NO - CLEAR DISPLAY AND TERMINATE COMMAND
                 ;
                 ;*******************************************************************
                 ;
                 ; FUNCTION: GOCMD - EXECUTE USER PROGRAM
                 ; INPUTS: NONE
                 ; OUTPUTS: NONE
                 ; CALLS: DISPC,RDKBD,CLEAR,GTHEX,ERR,OUTPT
                 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                 ;
                 GOCMD:
00CB CD0002              CALL    DISPC    ; DISPLAY USER PROGRAM COUNTER
00CE CDE702              CALL    RDKBD    ; READ FROM KEYBOARD
00D1 FE10                CPI     PERIO    ; IS CHARACTER A PERIOD ?
00D3 CAEC00              JZ      G10      ; YES - GO EXECUTE THE COMMAND
                                          ; NO - ARG - CHARACTER IS STILL IN A
00D6 32FE20              STA     IBUFF    ; REPLACE CHARACTER IN INPUT BUFFER
00D9 0601                MVI     B,DOT    ; ARG - DOT IN ADDRESS FIELD
00DB CDD701              CALL    CLEAR    ; CLEAR DISPLAY
00DE 0600                MVI     B,ADFLD  ; ARG - USE ADDRESS FIELD
00E0 CD2B02              CALL    GTHEX    ; GET HEX DIGITS
00E3 FE10                CPI     PERIO    ; WAS LAST CHARACTER A PERIOD ?
00E5 C21502              JNZ     ERR      ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
00E8 EB                  XCHG             ; PUT HEX VALUE FROM GTHEX TO H & L
00E9 22F220              SHLD    PSAV     ; HEX VALUE IS NEW USER PC
                 G10:
00EC 0600                MVI     B,NODOT  ; YES - ARG - NO DOT IN ADDRESS FIELD
00EE CDD701              CALL    CLEAR    ; CLEAR DISPLAY
00F1 AF                  XRA     A        ; ARG - USE ADDRESS FIELD OF DISPLAY
00F2 0600                MVI     B,NODOT  ; ARG - NO DOT IN ADDRESS FIELD
00F4 21A203              LXI     H,EXMSG  ; GET ADDRESS OF EXECUTION MESSAGE IN H & L
00F7 CDB702              CALL    OUTPT    ; DISPLAY EXECUTION MESSAGE
00FA C31B03              JMP     RSTOR    ; RESTORE USER REGISTERS INCL. PROGRAM COUNTER
                                          ;/I.E. BEGIN EXECUTION OF USER PROGRAM
                 ;
                 ;*******************************************************************
                 ;
                 ; FUNCTION: SSTEP - SINGLE STEP (EXECUTE ONE USER INSTRUCTION)
                 ; INPUTS: NONE
                 ; OUTPUTS: NONE
                 ; CALLS: DISPC,RDKBD,CLEAR,GTHEX,ERR
                 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                 ;
                 SSTEP:
00FD CD0002              CALL    DISPC    ; DISPLAY USER PROGRAM COUNTER
0100 CDE702              CALL    RDKBD    ; READ FROM KEYBOARD
0103 FE10                CPI     PERIO    ; WAS CHARACTER A PERIOD ?
0105 CAE901              JZ      CLDIS    ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
0108 FE11                CPI     COMMA    ; WAS LAST CHARACTER ',' ?
010A CA2601              JZ      STP20    ; YES - GO SET TIMER
                         ; NO - CHARACTER FROM KEYBOARD WAS NEITHER PERIOD NOR COMMA
010D 32FE20              STA     IBUFF    ; REPLACE THE CHARACTER IN THE INPUT BUFFER
0110 0601                MVI     B,DOT    ; ARG - DOT IN ADDRESS FIELD
0112 CDD701              CALL    CLEAR    ; CLEAR DISPLAY
0115 0600                MVI     B,ADFLD  ; ARG - USE ADDRESS FIELD OF DISPLAY
0117 CD2B02              CALL    GTHEX    ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED ?
                         FALSE   ERR      ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
```

```
 LOC  OBJ         SEQ        SOURCE STATEMENT

011A D21502       396+        JNC     ERR
011D EB           397         XCHG            ; HEX VALUE FROM GTHEX TO H & L
011E 22F220       398         SHLD    PSAV    ; HEX VALUE IS NEW USER PC
0121 FE10         399         CPI     PERIO   ; WAS LAST CHARACTER FROM GTHEX A PERIOD ?
0123 CAE901       400         JZ      CLDIS   ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
                  401                         ; NO - MUST HAVE BEEN A COMMA
                  402 STP20:
0126 3AF120       403         LDA     ISAV    ; GET USER INTERRUPT MASK
0129 E608         404         ANI     08H     ; KEEP INTERRUPT STATUS
012B 32FD20       405         STA     TEMP    ; SAVE USER INTERRUPT STATUS
012E 2AF220       406         LHLD    PSAV    ; GET USER PC
0131 7E           407         MOV     A,M     ; GET USER INSTRUCTION
0132 FEF3         408         CPI     (DI)    ; DI INSTRUCTION ?
0134 C23B01       409         JNZ     STP21   ; NO
0137 AF           410         XRA     A       ; YES - RESET USER INTERRUPT STATUS
0138 C34201       411         JMP     STP22
                  412 STP21:
013B FEFB         413         CPI     (EI)    ; EI INSTRUCTION ?
013D C24501       414         JNZ     STP23   ; NO
0140 3E08         415         MVI     A,08H   ; YES - SET USER INTERRUPT STATUS
                  416 STP22:
0142 32FD20       417         STA     TEMP    ; SAVE NEW USER INTERRUPT STATUS
                  418 STP23:
0145 3E40         419         MVI     A,(TIMER SHR 8) OR TMODE ; HIGH ORDER BITS OF TIMER VALUE
                  420                                  ; /OR'ED WITH TIMER MODE
0147 D325         421         OUT     TIMHI
0149 3EC5         422         MVI     A,TIMER AND 0FFH ; LOW ORDER BITS OF TIMER VALUE
014B D324         423         OUT     TIMLO
014D 3AFF20       424         LDA     USCSR   ; GET USER IMAGE OF WHAT'S IN CSR
0150 F6C0         425         ORI     TSTRT   ; SET TIMER COMMAND BITS TO START TIMER
0152 D320         426         OUT     CSR     ; START TIMER
0154 C31B03       427         JMP     RSTOR   ; RESTORE USER REGISTERS
                  428 ;
                  429 STP25:                  ; BRANCH HERE WHEN TIMER INTERRUPTS AFTER
                  430                         ;/ONE USER INSTRUCTION
0157 F5           431         PUSH    PSW     ; SAVE PSW
0158 3AFF20       432         LDA     USCSR   ; GET USER IMAGE OF WHAT'S IN CSR
015B E63F         433         ANI     3FH     ; CLEAR 2 HIGH ORDER BITS
015D F640         434         ORI     40H     ; SET TIMER STOP BIT
015F D320         435         OUT     CSR     ; STOP TIMER
0161 F1           436         POP     PSW     ; RETRIEVE PSW
0162 22EF20       437         SHLD    LSAV    ; SAVE H & L
0165 E1           438         POP     H       ; GET USER PROGRAM COUNTER FROM TOP OF STACK
0166 22F220       439         SHLD    PSAV    ; SAVE USER PC
0169 F5           440         PUSH    PSW
016A E1           441         POP     H
016B 22ED20       442         SHLD    FSAV    ; SAVE FLIP/FLOPS AND A REGISTER
016E 210000       443         LXI     H,0     ; CLEAR H & L
0171 39           444         DAD     SP      ; GET USER STACK POINTER
0172 22F420       445         SHLD    SSAV    ; SAVE USER STACK POINTER
0175 21ED20       446         LXI     H,BSAV+1        ; SET MONITOR STACK POINTER FOR
0178 F9           447         SPHL            ;/SAVING REMAINING USER REGISTERS
0179 C5           448         PUSH    B       ; SAVE B & C
017A D5           449         PUSH    D       ; SAVE D & E
017B 20           450         RIM             ; GET USER INTERRUPT MASK
017C E607         451         ANI     07H     ; KEEP MASK BITS
017E 21FD20       452         LXI     H,TEMP  ; GET USER INTERRUPT STATUS
0181 B6           453         ORA     M       ; OR IT INTO MASK
0182 32F120       454         STA     ISAV    ; SAVE INTERRUPT STATUS & MASK
0185 3E0E         455         MVI     A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE
0187 30           456         SIM
0188 C3FD00       457         JMP     SSTEP   ; GO GET READY FOR ANOTHER INSTRUCTION
                  458 ;
                  459 ;*****************************************************************
                  460 ;
                  461 ; FUNCTION: SUBST -  SUBSTITUTE MEMORY
                  462 ; INPUTS: NONE
                  463 ; OUTPUTS: NONE
                  464 ; CALLS: CLEAR,GTHEX,UPDAD,UPDDT,ERR
                  465 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  466 ;
                  467 SUBST:
018B 0601         468         MVI     B,DOT   ; ARG - DOT IN ADDRESS FIELD
018D CDD701       469         CALL    CLEAR   ; CLEAR THE DISPLAY
0190 0600         470         MVI     B,ADFLD ; ARG - USE ADDRESS FIELD OF DISPLAY
0192 CD2B02       471         CALL    GTHEX   ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?
                  472         FALSE   ERR     ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
0195 D21502       473+        JNC     ERR
0198 EB           474         XCHG            ; ASSIGN HEX VALUE RETURNED BY GTHEX TO
0199 22F620       475         SHLD    CURAD   ; / CURRENT ADDRESS
                  476 SUB05:
019C FE11         477         CPI     COMMA   ; WAS ',' THE LAST CHARACTER FROM KEYBOARD?
019E C2CF01       478         JNZ     SUB15   ; NO - GO TERMINATE THE COMMAND
01A1 0600         479         MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
01A3 CD5F03       480         CALL    UPDAD   ; UPDATE ADDRESS FIELD OF DISPLAY
01A6 2AF620       481         LHLD    CURAD   ; GET CURRENT ADDRESS IN H & L
01A9 7E           482         MOV     A,M     ; GET DATA BYTE POINTED TO BY CURRENT ADDRESS
01AA 32F820       483         STA     CURDT   ; STORE DATA BYTE AT CURRENT DATA
01AD 0601         484         MVI     B,DOT   ; ARG - DOT IN DATA FIELD
01AF CD6B03       485         CALL    UPDDT   ; UPDATE DATA FIELD OF DISPLAY
01B2 0601         486         MVI     B,DTFLD ; ARG - USE DATA FIELD
01B4 CD2B02       487         CALL    GTHEX   ; GET HEX DIGITS - WERE ANY HEX DIGITS RECEIVED?
01B7 F5           488         PUSH    PSW     ; (SAVE LAST CHARACTER)
                  489         FALSE   SUB10   ; NO - LEAVE DATA UNCHANGED AT CURRENT ADDRESS
01B8 D2C401       490+        JNC     SUB10
01BB 2AF620       491         LHLD    CURAD   ; YES - GET CURRENT ADDRESS IN H & L
01BE 73           492         MOV     M,E     ; STORE NEW DATA AT CURRENT ADDRESS
                  493                 ; MAKE SURE DATA WAS ACTUALLY STORED IN CASE
                  494                 ;/CURRENT ADDRESS IS IN ROM OR IS NON-EXISTANT
01BF 7B           495         MOV     A,E     ; DATA TO A FOR COMPARISON
```

```
 LOC  OBJ         SOURCE STATEMENT

 01C0 BE                CMP     M         ; WAS DATA STORED CORRECTLY?
 01C1 C21502            JNZ     ERR       ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
                  SUB10:
 01C4 2AF620            LHLD    CURAD     ; INCREMENT CURRENT ADDRESS
 01C7 23                INX     H
 01C8 22F620            SHLD    CURAD
 01CB F1                POP     PSW       ; RETRIEVE LAST CHARACTER
 01CC C39C01            JMP     SUB05     ;
                  SUB15:
 01CF FE10              CPI     PERIO     ; WAS LAST CHARACTER '.' ?
 01D1 C21502            JNZ     ERR       ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
 01D4 C3E901            JMP     CLDIS     ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
                  ;
                  ;
                  ;*********************************************************************
                  ;
                  ;                            UTILITY ROUTINES
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: CLEAR - CLEAR THE DISPLAY
                  ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT IN ADDRESS FIELD OF DISPLAY
                  ;                      - 0 MEANS NO DOT
                  ; OUTPUTS: NONE
                  ; CALLS: OUTPT
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION:  CLEAR SENDS BLANK CHARACTERS TO BOTH THE ADDRESS FIELD
                  ;               AND THE DATA FIELD OF THE DISPLAY. IF THE DOT FLAG IS
                  ;               SET THEN A DOT WILL APPEAR AT THE RIGHT EDGE OF THE
                  ;               ADDRESS FIELD.
                  ;
                  CLEAR:
 01D7 AF                XRA     A         ; ARG - USE ADDRESS FIELD OF DISPLAY
                                          ; ARG - FLAG FOR DOT IN ADDR. FIELD IS IN B
 01D8 219A03            LXI     H,BLNKS   ; ARG - ADDRESS OF BLANKS FOR DISPLAY
 01DB CDB702            CALL    OUTPT     ; OUTPUT BLANKS TO ADDRESS FIELD
 01DE 3E01              MVI     A,DTFLD   ; ARG - USE DATA FIELD OF DISPLAY
 01E0 0600              MVI     B,NODOT   ; ARG - NO DOT IN DATA FIELD
 01E2 219A03            LXI     H,BLNKS   ; ARG - ADDRESS OF BLANKS FOR DISPLAY
 01E5 CDB702            CALL    OUTPT     ; OUTPUT BLANKS TO DATA FIELD
 01E8 C9                RET               ; RETURN
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: CLDIS - CLEAR DISPLAY AND TERMINATE COMMAND
                  ; INPUTS: NONE
                  ; OUTPUTS: NONE
                  ; CALLS: CLEAR
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION: CLDIS IS JUMPED TO BY COMMAND ROUTINES WISHING TO
                  ;              TERMINATE NORMALLY. CLDIS CLEARS THE DISPLAY AND
                  ;              BRANCHES TO THE COMMAND RECOGNIZER.
                  ;
                  CLDIS:
 01E9 0600              MVI     B,NODOT   ; ARG - NO DOT IN ADDRESS FIELD
 01EB CDD701            CALL    CLEAR     ; CLEAR THE DISPLAY
 01EE C36600            JMP     CMMND     ; GO GET ANOTHER COMMAND
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: CLDST - COLD START
                  ; INPUTS: NONE
                  ; OUTPUTS: NONE
                  ; CALLS: NOTHING
                  ; DESTROYS: A
                  ; DESCRIPTION:  CLDST IS JUMPED TO BY THE MAIN COLD START PROCEDURE,
                  ;               COMPLETES COLD START INITIALIZATION, AND JUMPS BACK
                  ;               TO THE MAIN COLD START PROCEDURE.
                  ;
                  CLDST:
 01F1 3ECC              MVI     A,KBNIT   ; GET CONTROL CHARACTER
 01F3 320019            STA     CNTRL     ; INITIALIZE KEYBOARD/DISPLAY BLANKING
 01F6 3E00              MVI     A,CSNIT   ; INITIAL VALUE OF COMMAND STATUS REGISTER
 01F8 D320              OUT     CSR       ; INITIALIZE CSR
 01FA 32FF20            STA     USCSR     ; INITIALIZE USER CSR VALUE
 01FD C30800            JMP     CLDBK     ; BACK TO MAIN PROCEDURE
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: DISPC - DISPLAY PROGRAM COUNTER
                  ; INPUTS: NONE
                  ; OUTPUTS: NONE
                  ; CALLS: UPDAD,UPDDT
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION: DISPC DISPLAYS THE USER PROGRAM COUNTER IN THE ADDRESS
                  ;              FIELD OF THE DISPLAY, WITH A DOT AT THE RIGHT EDGE
                  ;              OF THE FIELD. THE BYTE OF DATA ADDRESSED BY THE PROGRAM
                  ;              COUNTER IS DISPLAYED IN THE DATA FIELD OF THE DISPLAY.
                  ;
                  DISPC:
 0200 2AF220            LHLD    PSAV      ; GET USER PROGRAM COUNTER
 0203 22F620            SHLD    CURAD     ; MAKE IT THE CURRENT ADDRESS
 0206 7E                MOV     A,M       ; GET THE INSTRUCTION AT THAT ADDRESS
 0207 32F820            STA     CURDT     ; MAKE IT THE CURRENT DATA
 020A 0601              MVI     B,DOT     ; ARG - DOT IN ADDRESS FIELD
 020C CD5F03            CALL    UPDAD     ; UPDATE ADDRESS FIELD OF DISPLAY
 020F 0600              MVI     B,NODOT   ; ARG - NO DOT IN DATA FIELD
 0211 CD6B03            CALL    UPDDT     ; UPDATE DATA FIELD OF DISPLAY
 0214 C9                RET
                  ;
```

```
  LOC  OBJ          SEQ         SOURCE STATEMENT

                         ;*******************************************************************
                         ;
                         ; FUNCTION: ERR - DISPLAY ERROR MESSAGE
                         ; INPUTS: NONE
                         ; OUTPUTS: NONE
                         ; CALLS: OUTPT
                         ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                         ; DESCRIPTION:  ERR IS JUMPED TO BY COMMAND ROUTINES WISHING TO
                         ;               TERMINATE BECAUSE OF AN ERROR.
                         ;               ERR OUTPUTS AN ERROR MESSAGE TO THE DISPLAY AND
                         ;               BRANCHES TO THE COMMAND RECOGNIZER.
                         ;
                         ERR:
 0215 AF                         XRA     A       ; ARG - USE ADDRESS FIELD
 0216 0600                       MVI     B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
 0218 219E03                     LXI     H,ERMSG ; ARG - ADDRESS OF ERROR MESSAGE
 021B CDB702                     CALL    OUTPT   ; OUTPUT ERROR MESSAGE TO ADDRESS FIELD
 021E 3E01                       MVI     A,DTFLD ; ARG - USE DATA FIELD
 0220 0600                       MVI     B,NODOT ; ARG - NO DOT IN DATA FIELD
 0222 219A03                     LXI     H,BLNKS ; ARG - ADDRESS OF BLANKS FOR DISPLAY
 0225 CDB702                     CALL    OUTPT   ; OUTPUT BLANKS TO DATA FIELD
 0228 C36600                     JMP     CMMND   ; GO GET A NEW COMMAND
                         ;
                         ;*******************************************************************
                         ;
                         ; FUNCTION: GTHEX - GET HEX DIGITS
                         ; INPUTS: B - DISPLAY FLAG - 0 MEANS USE ADDRESS FIELD OF DISPLAY
                         ;                          - 1 MEANS USE DATA FIELD OF DISPLAY
                         ; OUTPUTS: A - LAST CHARACTER READ FROM KEYBOARD
                         ;          DE - HEX DIGITS FROM KEYBOARD EVALUATED MODULO 2**16
                         ;          CARRY - SET IF AT LEAST ONE VALID HEX DIGIT WAS READ
                         ;                - RESET OTHERWISE
                         ; CALLS: RDKBD,INSDG,HXDSP,OUTPT
                         ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                         ; DESCRIPTION:  GTHEX ACCEPTS A STRING OF HEX DIGITS FROM THE KEYBOARD,
                         ;               DISPLAYS THEM AS THEY ARE RECEIVED, AND RETURNS THEIR
                         ;               VALUE AS A 16 BIT INTEGER. IF MORE THAN 4 HEX DIGITS
                         ;               ARE RECEIVED, ONLY THE LAST 4 ARE USED. IF THE DISPLAY
                         ;               FLAG IS SET, THE LAST 2 HEX DIGITS ARE DISPLAYED IN THE
                         ;               DATA FIELD OF THE DISPLAY. OTHERWISE, THE LAST 4 HEX
                         ;               DIGITS ARE DISPLAYED IN THE ADDRESS FIELD OF THE
                         ;               DISPLAY. IN EITHER CASE, A DOT WILL BE DISPLAYED AT THE
                         ;               RIGHTMOST EDGE OF THE FIELD. A CHARACTER WHICH IS NOT
                         ;               A HEX DIGIT TERMINATES THE STRING AND IS RETURNED AS
                         ;               AN OUTPUT OF THE FUNCTION. IF THE TERMINATOR IS NOT
                         ;               A PERIOD OR A COMMA THEN ANY HEX DIGITS WHICH MAY HAVE
                         ;               BEEN RECEIVED ARE CONSIDERED TO BE INVALID. THE
                         ;               FUNCTION RETURNS A FLAG INDICATING WHETHER OR NOT ANY
                         ;               VALID HEX DIGITS WERE RECEIVED.
                         ;
                         GTHEX:
 022B 0E00                       MVI     C,0     ; RESET HEX DIGIT FLAG
 022D C5                         PUSH    B       ; SAVE DISPLAY AND HEX DIGIT FLAGS
 022E 110000                     LXI     D,0     ; SET HEX VALUE TO ZERO
 0231 D5                         PUSH    D       ; SAVE HEX VALUE
                         GTH05:
 0232 CDE702                     CALL    RDKBD   ; READ KEYBOARD
 0235 FE10                       CPI     10H     ; IS CHARACTER A HEX DIGIT?
 0237 D25502                     JNC     GTH20   ; NO - GO CHECK FOR TERMINATOR
                                                 ; YES - ARG - NEW HEX DIGIT IS IN A
 023A D1                         POP     D       ; ARG - RETRIEVE HEX VALUE
 023B CD9F02                     CALL    INSDG   ; INSERT NEW DIGIT IN HEX VALUE
 023E C1                         POP     B       ; RETRIEVE DISPLAY FLAG
 023F 0E01                       MVI     C,1     ; SET HEX DIGIT FLAG
                                                 ;/(I.E. A HEX DIGIT HAS BEEN READ)
 0241 C5                         PUSH    B       ; SAVE DISPLAY AND HEX DIGIT FLAGS
 0242 D5                         PUSH    D       ; SAVE HEX VALUE
 0243 78                         MOV     A,B     ; TEST DISPLAY FLAG
 0244 0F                         RRC             ; SHOULD ADDRESS FIELD OF DISPLAY BE USED ?
 0245 D24902                     JNC     GTH10   ; YES - USE HEX VALUE AS IS
                                                 ; NO - ONLY LOW ORDER BYTE OF HEX VALUE SHOULD
                                                 ; /BE USED FOR DATA FIELD OF DISPLAY
 0248 53                         MOV     D,E     ; PUT LOW ORDER BYTE OF HEX VALUE IN D
                         GTH10:
                                                 ; ARG - HEX VALUE TO BE EXPANDED IS IN D & E
 0249 CD6C02                     CALL    HXDSP   ; EXPAND HEX VALUE FOR DISPLAY
                                                 ; ARG - ADDRESS OF EXPANDED HEX VALUE IN H & L
 024C 78                         MOV     A,B     ; ARG - PUT DISPLAY FLAG IN A
 024D 0601                       MVI     B,DOT   ; ARG - DOT IN APPROPRIATE FIELD
 024F CDB702                     CALL    OUTPT   ; OUTPUT HEX VALUE TO DISPLAY
 0252 C33202                     JMP     GTH05   ; GO GET NEXT CHARACTER
                         GTH20:                  ; LAST CHARACTER WAS NOT A HEX DIGIT
 0255 D1                         POP     D       ; RETRIEVE HEX VALUE
 0256 C1                         POP     B       ; RETRIEVE HEX DIGIT FLAG IN C
 0257 FE11                       CPI     COMMA   ; WAS LAST CHARACTER ',' ?
 0259 CA6702                     JZ      GTH25   ; YES - READY TO RETURN
 025C FE10                       CPI     PERIO   ; NO - WAS LAST CHARACTER '.' ?
 025E CA6702                     JZ      GTH25   ; YES - READY TO RETURN
                                                 ; NO - INVALID TERMINATOR - IGNORE ANY HEX DIGITS READ
 0261 110000                     LXI     D,0     ; SET HEX VALUE TO ZERO
 0264 C3F702                     JMP     RETF    ; RETURN FALSE
                         GTH25:
 0267 47                         MOV     B,A     ; SAVE LAST CHARACTER
 0268 79                         MOV     A,C     ; SHIFT HEX DIGIT FLAG TO
 0269 0F                         RRC             ;/CARRY BIT
 026A 78                         MOV     A,B     ; RESTORE LAST CHARACTER
 026B C9                         RET             ; RETURN
                         ;
                         ;*******************************************************************
                         ;
```

```
 LOC  OBJ         SEQ        SOURCE STATEMENT

                    ; FUNCTION: HXDSP - EXPAND HEX DIGITS FOR DISPLAY
                    ; INPUTS: DE - 4 HEX DIGITS
                    ; OUTPUTS: HL - ADDRESS OF OUTPUT BUFFER
                    ; CALLS: NOTHING
                    ; DESTROYS: A,H,L,F/F'S
                    ; DESCRIPTION: HXDSP EXPANDS EACH INPUT BYTE TO 2 BYTES IN A FORM
                    ;              SUITABLE FOR DISPLAY BY THE OUTPUT ROUTINES. EACH INPUT
                    ;              BYTE IS DIVIDED INTO 2 HEX DIGITS. EACH HEX DIGIT IS
                    ;              PLACED IN THE LOW ORDER 4 BITS OF A BYTE WHOSE HIGH
                    ;              ORDER 4 BITS ARE SET TO ZERO. THE RESULTING BYTE IS
                    ;              STORED IN THE OUTPUT BUFFER. THE FUNCTION RETURNS THE
                    ;              ADDRESS OF THE OUTPUT BUFFER.
                    ;
                    HXDSP:
026C 7A                     MOV     A,D      ; GET FIRST DATA BYTE
026D 0F                     RRC              ; CONVERT 4 HIGH ORDER BITS
026E 0F                     RRC              ; /TO A SINGLE CHARACTER
026F 0F                     RRC
0270 0F                     RRC
0271 E60F                   ANI     0FH
0273 21F920                 LXI     H,OBUFF ; GET ADDRESS OF OUTPUT BUFFER
0276 77                     MOV     M,A      ; STORE CHARACTER IN OUTPUT BUFFER
0277 7A                     MOV     A,D      ; GET FIRST DATA BYTE AND CONVERT 4 LOW ORDER
0278 E60F                   ANI     0FH      ; /BITS TO A SINGLE CHARACTER
027A 23                     INX     H        ; NEXT BUFFER POSITION
027B 77                     MOV     M,A      ; STORE CHARACTER IN BUFFER
027C 7B                     MOV     A,E      ; GET SECOND DATA BYTE
027D 0F                     RRC              ; CONVERT 4 HIGH ORDER BITS
027E 0F                     RRC              ; /TO A SINGLE CHARACTER
027F 0F                     RRC
0280 0F                     RRC
0281 E60F                   ANI     0FH
0283 23                     INX     H        ; NEXT BUFFER POSITION
0284 77                     MOV     M,A      ; STORE CHARACTER IN BUFFER
0285 7B                     MOV     A,E      ; GET SECOND DATA BYTE AND CONVERT LOW ORDER
0286 E60F                   ANI     0FH      ; /4 BITS TO A SINGLE CHARACTER
0288 23                     INX     H        ; NEXT BUFFER POSITION
0289 77                     MOV     M,A      ; STORE CHARACTER IN BUFFER
028A 21F920                 LXI     H,OBUFF ; RETURN ADDRESS OF OUTPUT BUFFER IN H & L
028D C9                     RET
                    ;
                    ;*********************************************************************
                    ;
                    ; FUNCTION: ININT - INPUT INTERRUPT PROCESSING
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: NOTHING
                    ; DESTROYS: NOTHING
                    ; DESCRIPTION:  ININT IS ENTERED BY MEANS OF AN INTERRUPT VECTOR (IV2C)
                    ;               WHEN THE READ KEYBOARD ROUTINE IS WAITING FOR A
                    ;               CHARACTER AND THE USER HAS PRESSED A KEY ON THE
                    ;               KEYBOARD (EXCEPT "RESET" OR "VECTORED INTERRUPT").
                    ;               ININT STORES THE INPUT CHARACTER IN THE INPUT BUFFER AND
                    ;               RETURNS CONTROL TO THE READ KEYBOARD ROUTINE.
                    ;
                    ININT:
028E E5                     PUSH    H        ; SAVE H & L
028F F5                     PUSH    PSW      ; SAVE F/F'S & REGISTER A
0290 210019                 LXI     H,CNTRL ; ADDRESS FOR CONTROL CHARACTER OUTPUT
0293 3640                   MVI     M,READ  ; OUTPUT CONTROL CHARACTER FOR READING
                                             ; /FROM KEYBOARD
0295 25                     DCR     H        ; ADDRESS FOR CHARACTER INPUT
0296 7E                     MOV     A,M      ; READ A CHARACTER
0297 E63F                   ANI     3FH      ; ZERO 2 HIGH ORDER BITS
0299 32FE20                 STA     IBUFF    ; STORE CHARACTER IN INPUT BUFFER
029C F1                     POP     PSW      ; RESTORE F/F'S & REGISTER A
029D E1                     POP     H        ; RESTORE H & L
029E C9                     RET
                    ;
                    ;*********************************************************************
                    ;
                    ; FUNCTION: INSDG - INSERT HEX DIGIT
                    ; INPUTS: A - HEX DIGIT TO BE INSERTED
                    ;         DE - HEX VALUE
                    ; OUTPUTS: DE - HEX VALUE WITH DIGIT INSERTED
                    ; CALLS: NOTHING
                    ; DESTROYS: A,F/F'S
                    ; DESCRIPTION: INSDG SHIFTS THE CONTENTS OF D & E LEFT 4 BITS
                    ;              (1 HEX DIGIT) AND INSERTS THE HEX DIGIT IN A IN THE LOW
                    ;              ORDER DIGIT POSITION OF THE RESULT. A IS ASSUMED TO
                    ;              CONTAIN A SINGLE HEX DIGIT IN THE LOW ORDER 4 BITS AND
                    ;              ZEROS IN THE HIGH ORDER  4 BITS.
                    ;
                    INSDG:
029F EB                     XCHG             ; PUT D & E IN H & L
02A0 29                     DAD     H        ; SHIFT H & L LEFT 4 BITS
02A1 29                     DAD     H
02A2 29                     DAD     H
02A3 29                     DAD     H
02A4 85                     ADD     L        ; INSERT LOW ORDER DIGIT
02A5 6F                     MOV     L,A
02A6 EB                     XCHG             ; PUT H & L BACK IN D & E
02A7 C9                     RET
                    ;
                    ;*********************************************************************
                    ;
                    ; FUNCTION: NXTRG - ADVANCE REGISTER POINTER TO NEXT REGISTER
                    ; INPUTS: NONE
                    ; OUTPUTS: CARRY - 1 IF POINTER IS ADVANCED SUCCESSFULLY
```

```
LOC  OBJ          SOURCE STATEMENT

                  ;                  - 0 OTHERWISE
                  ; CALLS: NOTHING
                  ; DESTROYS: A,F/F'S
                  ; DESCRIPTION:  IF THE REGISTER POINTER POINTS TO THE LAST REGISTER IN
                  ;               THE EXAMINE REGISTER SEQUENCE, THE POINTER IS NOT
                  ;               CHANGED AND THE FUNCTION RETURNS FALSE. IF THE REGISTER
                  ;               POINTER DOES NOT POINT TO THE LAST REGISTER THEN THE
                  ;               POINTER IS ADVANCED TO THE NEXT REGISTER IN THE SEQUENCE
                  ;               AND THE FUNCTION RETURNS TRUE.
                  ;
                  NXTRG:
02A8 3AFD20               LDA     RGPTR    ; GET REGISTER POINTER
02AB FE0C                 CPI     NUMRG-1  ; DOES POINTER POINT TO LAST REGISTER?
02AD D2F702               JNC     RETF     ; YES - UNABLE TO ADVANCE POINTER - RETURN FALSE
02B0 3C                   INR     A        ; NO - ADVANCE REGISTER POINTER
02B1 32FD20               STA     RGPTR    ; SAVE REGISTER POINTER
02B4 C3FA02               JMP     RETT     ; RETURN TRUE
                  ;
                  ;*******************************************************************
                  ;
                  ; FUNCTION: OUTPT - OUTPUT CHARACTERS TO DISPLAY
                  ; INPUTS: A - DISPLAY FLAG - 0 = USE ADDRESS FIELD
                  ;                            1 = USE DATA FIELD
                  ;           B - DOT FLAG - 1 = OUTPUT DOT AT RIGHT EDGE OF FIELD
                  ;                          0 = NO DOT
                  ;           HL - ADDRESS OF CHARACTERS TO BE OUTPUT
                  ; CALLS: NOTHING
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION:  OUTPT SENDS CHARACTERS TO THE DISPLAY. THE ADDRESS
                  ;               OF THE CHARACTERS IS RECEIVED AS AN ARGUMENT. EITHER
                  ;               2 CHARACTERS ARE SENT TO THE DATA FIELD, OR 4 CHARACTERS
                  ;               ARE SENT TO THE ADDRESS FIELD, DEPENDING ON THE
                  ;               DISPLAY FLAG ARGUMENT. THE DOT FLAG ARGUMENT DETERMINES
                  ;               WHETHER OR NOT A DOT (DECIMAL POINT) WILL BE SENT
                  ;               ALONG WITH THE LAST OUTPUT CHARACTER.
                  ;
                  OUTPT:
02B7 0F                   RRC              ; USE DATA FIELD ?
02B8 DAC202               JC      OUT05    ; YES - GO SET UP TO USE DATA FIELD
02BB 0E04                 MVI     C,4      ; NO - COUNT FOR ADDRESS FIELD
02BD 3E90                 MVI     A,ADISP  ; CONTROL CHARACTER FOR OUTPUT TO ADDRESS
                                           ; /FIELD OF DISPLAY
02BF C3C602               JMP     OUT10
                  OUT05:
02C2 0E02                 MVI     C,2      ; COUNT FOR DATA FIELD
02C4 3E94                 MVI     A,DDISP  ; CONTROL CHARACTER FOR OUTPUT TO DATA FIELD
                                           ; /OF DISPLAY
                  OUT10:
02C6 320019               STA     CNTRL
                  OUT15:
02C9 7E                   MOV     A,M      ; GET OUTPUT CHARACTER
02CA EB                   XCHG             ; SAVE OUTPUT CHARACTER ADDRESS IN D & E
02CB 218403               LXI     H,DSPTB  ; GET DISPLAY FORMAT TABLE ADDRESS
02CE 85                   ADD     L        ; USE OUTPUT CHARACTER AS A POINTER TO
02CF 6F                   MOV     L,A      ; /DISPLAY FORMAT TABLE
02D0 7E                   MOV     A,M      ; GET DISPLAY FORMAT CHARACTER FROM TABLE
02D1 61                   MOV     H,C      ; TEST COUNTER WITHOUT CHANGING IT
02D2 25                   DCR     H        ; IS THIS THE LAST CHARACTER ?
02D3 C2DC02               JNZ     OUT20    ; NO - GO OUTPUT CHARACTER AS IS
02D6 05                   DCR     B        ; YES - IS DOT FLAG SET ?
02D7 C2DC02               JNZ     OUT20    ; NO - GO OUTPUT CHARACTER AS IS
02DA F608                 ORI     DTMSK    ; YES - OR IN MASK TO DISPLAY DOT WITH
                                           ; /LAST CHARACTER
                  OUT20:
02DC 2F                   CMA              ; COMPLEMENT OUTPUT CHARACTER
02DD 320018               STA     DSPLY    ; SEND CHARACTER TO DISPLAY
02E0 EB                   XCHG             ; RETRIEVE OUTPUT CHARACTER ADDRESS
02E1 23                   INX     H        ; NEXT OUTPUT CHARACTER
02E2 0D                   DCR     C        ; ANY MORE OUTPUT CHARACTERS ?
02E3 C2C902               JNZ     OUT15    ; YES - GO PROCESS ANOTHER CHARACTER
02E6 C9                   RET              ; NO - RETURN
                  ;
                  ;*******************************************************************
                  ;
                  ; FUNCTION: RDKBD - READ KEYBOARD
                  ; INPUTS: NONE
                  ; OUTPUTS: A - CHARACTER READ FROM KEYBOARD
                  ; CALLS: NOTHING
                  ; DESTROYS: A,H,L,F/F'S
                  ; DESCRIPTION:  RDKBD DETERMINES WHETHER OR NOT THERE IS A CHARACTER IN
                  ;               THE INPUT BUFFER. IF NOT, THE FUNCTION ENABLES
                  ;               INTERRUPTS AND LOOPS UNTIL THE INPUT INTERRUPT
                  ;               ROUTINE STORES A CHARACTER IN THE BUFFER. WHEN
                  ;               THE BUFFER CONTAINS A CHARACTER, THE FUNCTION FLAGS
                  ;               THE BUFFER AS EMPTY AND RETURNS THE CHARACTER
                  ;               AS OUTPUT.
                  ;
                  RDKBD:
02E7 21FE20               LXI     H,IBUFF ; GET INPUT BUFFER ADDRESS
02EA 7E                   MOV     A,M     ; GET BUFFER CONTENTS
                                  ; HIGH ORDER BIT = 1 MEANS BUFFER IS EMPTY
02EB B7                   ORA     A       ; IS A CHARACTER AVAILABLE ?
02EC F2F302               JP      RDK10   ; YES - EXIT FROM LOOP
02EF FB                   EI              ; NO - READY FOR CHARACTER FROM KEYBOARD
02F0 C3E702               JMP     RDKBD
                  RDK10:
02F3 3680                 MVI     M,EMPTY ; SET BUFFER EMPTY FLAG
02F5 F3                   DI              ; RETURN WITH INTERRUPTS DISABLED
02F6 C9                   RET
```

```
  LOC  OBJ         SEQ        SOURCE STATEMENT

                   ;
                   ;**********************************************************************
                   ;
                   ; FUNCTION: RETF - RETURN FALSE
                   ; INPUTS: NONE
                   ; OUTPUTS: CARRY = 0 (FALSE)
                   ; CALLS: NOTHING
                   ; DESTROYS: CARRY
                   ; DESCRIPTION: RETF IS JUMPED TO BY FUNCTIONS WISHING TO RETURN FALSE.
                   ;              RETF RESETS CARRY TO 0 AND RETURNS TO THE CALLER OF
                   ;              THE ROUTINE INVOKING RETF.
                   ;
                   RETF:
  02F7 37                  STC              ; SET CARRY TRUE
  02F8 3F                  CMC              ; COMPLEMENT CARRY TO MAKE IT FALSE
  02F9 C9                  RET
                   ;
                   ;**********************************************************************
                   ;
                   ; FUNCTION: RETT - RETURN TRUE
                   ; INPUTS: NONE
                   ; OUTPUTS: CARRY = 1 (TRUE)
                   ; CALLS: NOTHING
                   ; DESTROYS: CARRY
                   ; DESCRIPTION: RETT IS JUMPED TO BY ROUTINES WISHING TO RETURN TRUE.
                   ;              RETT SETS CARRY TO 1 AND RETURNS TO THE CALLER OF
                   ;              THE ROUTINE INVOKING RETT.
                   ;
                   RETT:
  02FA 37                  STC              ; SET CARRY TRUE
  02FB C9                  RET
                   ;
                   ;**********************************************************************
                   ;
                   ; FUNCTION: RGLOC - GET REGISTER SAVE LOCATION
                   ; INPUTS: NONE
                   ; OUTPUTS: HL - REGISTER SAVE LOCATION
                   ; CALLS: NOTHING
                   ; DESTROYS: B,C,H,L,F/F'S
                   ; DESCRIPTION:  RGLOC RETURNS THE SAVE LOCATION OF THE REGISTER
                   ;               INDICATED BY THE CURRENT REGISTER POINTER VALUE.
                   ;
                   RGLOC:
  02FC 2AFD20              LHLD    RGPTR    ; GET REGISTER POINTER
  02FF 2600                MVI     H,0      ; /IN H & L
  0301 01ED03              LXI     B,RGTBL  ; GET REGISTER SAVE LOCATION TABLE ADDRESS
  0304 09                  DAD     B        ; POINTER INDEXES TABLE
  0305 6E                  MOV     L,M      ; GET LOW ORDER BYTE OF REGISTER SAVE LOC.
  0306 2620                MVI     H,(RAMST SHR 8) ; GET HIGH ORDER BYTE OF
                                            ; /REGISTER SAVE LOCATION
  0308 C9                  RET
                   ;
                   ;**********************************************************************
                   ;
                   ; FUNCTION: RGNAM - DISPLAY REGISTER NAME
                   ; INPUTS: NONE
                   ; OUTPUTS: NONE
                   ; CALLS: OUTPT
                   ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   ; DESCRIPTION: RGNAM DISPLAYS, IN THE ADDRESS FIELD OF THE DISPLAY,
                   ;              THE REGISTER NAME CORRESPONDING TO THE CURRENT
                   ;              REGISTER POINTER VALUE.
                   ;
                   RGNAM:
  0309 2AFD20              LHLD    RGPTR    ; GET REGISTER POINTER
  030C 2600                MVI     H,0
  030E 29                  DAD     H        ; MULTIPLY POINTER VALUE BY 4
  030F 29                  DAD     H        ;/(REGISTER NAME TABLE HAS 4 BYTE ENTRIES)
  0310 01B903              LXI     B,NMTBL  ; GET ADDRESS OF START OF REGISTER NAME TABLE
  0313 09                  DAD     B        ; ARG - ADD TABLE ADDRESS TO POINTER - RESULT IS
                                            ;/ADDRESS OF APPROPRIATE REGISTER NAME IN H & L
  0314 AF                  XRA     A        ; ARG - USE ADDRESS FIELD OF DISPLAY
  0315 0600                MVI     B,NODOT  ; ARG - NO DOT IN ADDRESS FIELD
  0317 CDB702              CALL    OUTPT    ; OUTPUT REGISTER NAME TO ADDRESS FIELD
  031A C9                  RET
                   ;
                   ;**********************************************************************
                   ;
                   ; FUNCTION: RSTOR - RESTOR USER REGISTERS
                   ; INPUTS: NONE
                   ; OUTPUTS: NONE
                   ; CALLS: NOTHING
                   ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   ; DESCRIPTION:  RSTOR RESTORES ALL CPU REGISTERS, FLIP/FLOPS,
                   ;               INTERRUPT STATUS, INTERRUPT MASK, STACK POINTER
                   ;               AND PROGRAM COUNTER FROM THEIR RESPECTIVE
                   ;               SAVE LOCATIONS IN MEMORY. BY RESTORING THE PROGRAM
                   ;               COUNTER, THE ROUTINE EFFECTIVELY TRANSFERS CONTROL TO
                   ;               THE ADDRESS IN THE PROGRAM COUNTER SAVE LOCATION.
                   ;
                   ;               THE TIMING OF THIS ROUTINE IS CRITICAL TO THE
                   ;               CORRECT OPERATION OF THE SINGLE STEP ROUTINE.
                   ;               IF ANY MODIFICATION CHANGES THE NUMBER OF CPU
                   ;               STATES NEEDED TO EXECUTE THIS ROUTINE THEN THE
                   ;               TIMER VALUE MUST BE ADJUSTED BY THE SAME NUMBER.
                   ;
                   ; ***** THIS IS ALSO THE ENTRY POINT FOR THE TTY MONITOR
                   ;       TO RESTORE REGISTERS.
                   ;
                   RSTOR:
```

```
  LOC  OBJ         SOURCE STATEMENT

 031B 3AF120              LDA     ISAV     ; GET USER INTERRUPT MASK
 031E F618                ORI     18H      ; ENABLE SETTING OF INTERRUPT MASK AND
                                           ; /RESET RST7.5 FLIP FLOP
 0320 30                  SIM              ; RESTORE USER INTERRUPT MASK
                  ;       RESTORE USER INTERRUPT STATUS
 0321 3AF120              LDA     ISAV     ; GET USER INTERRUPT MASK
 0324 E608                ANI     08H      ; SHOULD USER INTERRUPTS BE ENABLED ?
 0326 CA2D03              JZ      RSR05    ; NO - LEAVE INTERRUPTS DISABLED
 0329 FB                  EI               ; YES - ENABLE INTERRUPTS FOR USER PROGRAM
 032A C33103              JMP     RSR10
                  RSR05:
 032D 37                  STC              ; DUMMY INSTRUCTIONS - WHEN SINGLE STEP ROUTINE
 032E D23103              JNC     RSR10    ; /IS BEING USED, THE TIMER IS RUNNING AND
                                           ; /EXECUTE TIME FOR THIS ROUTINE MUST NOT
                                           ; /VARY.
                  RSR10:
 0331 21E920              LXI     H,MNSTK  ; SET MONITOR STACK POINTER TO START OF STACK
 0334 F9                  SPHL             ; /WHICH IS ALSO END OF REGISTER SAVE AREA
 0335 D1                  POP     D        ; RESTORE REGISTERS
 0336 C1                  POP     B
 0337 F1                  POP     PSW
 0338 2AF420              LHLD    SSAV     ; RESTORE USER STACK POINTER
 033B F9                  SPHL
 033C 2AF220              LHLD    PSAV
 033F E5                  PUSH    H        ; PUT USER PROGRAM COUNTER ON STACK
 0340 2AEF20              LHLD    LSAV     ; RESTORE H & L REGISTERS
 0343 C9                  RET              ; JUMP TO USER PROGRAM COUNTER
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: SETRG - SET REGISTER POINTER
                  ; INPUTS: NONE
                  ; OUTPUTS: CARRY - SET IF CHARACTER FROM KEYBOARD IS A REGISTER DESIGNATOR
                  ;                  RESET OTHERWISE
                  ; CALLS: RDKBD
                  ; DESTROYS: A,B,C,H,L,F/F'S
                  ; DESCRIPTION: SETRG READS A CHARACTER FROM THE KEYBOARD. IF THE
                  ;              CHARACTER IS A REGISTER DESIGNATOR, IT IS CONVERTED TO
                  ;              THE CORRESPONDING REGISTER POINTER VALUE, THE POINTER IS
                  ;              SAVED, AND THE FUNCTION RETURNS 'TRUE'. OTHERWISE, THE
                  ;              FUNCTION RETURNS 'FALSE'.
                  ;
                  SETRG:
 0344 CDE702              CALL    RDKBD    ; READ FROM KEYBOARD
 0347 FE10                CPI     10H      ; IS CHARACTER A DIGIT?
 0349 D2F702              JNC     RETF     ; NO - RETURN FALSE - CHARACTER IS NOT A
                                           ; /REGISTER DESIGNATOR
 034C D603                SUI     3        ; YES - TRY TO CONVERT REGISTER DESIGNATOR TO
                                           ; / INDEX INTO REGISTER POINTER TABLE
                                           ; WAS CONVERSION SUCCESSFUL?
 034E DAF702              JC      RETF     ; NO - RETURN FALSE
 0351 4F                  MOV     C,A      ; INDEX TO B & C
 0352 0600                MVI     B,0      ;
 0354 21AC03              LXI     H,RGPTB  ; GET ADDRESS OF REGISTER POINTER TABLE
 0357 09                  DAD     B        ; INDEX POINTS INTO TABLE
 0358 7E                  MOV     A,M      ; GET REGISTER POINTER FROM TABLE
 0359 32FD20              STA     RGPTR    ; SAVE REGISTER POINTER
 035C C3FA02              JMP     RETT     ; RETURN TRUE
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: UPDAD - UPDATE ADDRESS FIELD OF DISPLAY
                  ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT AT RIGHT EDGE OF FIELD
                  ;                        0 MEANS NO DOT
                  ; OUTPUTS: NONE
                  ; CALLS: HXDSP,OUTPT
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION: UPDAD UPDATES THE ADDRESS FIELD OF THE DISPLAY USING
                  ;              THE CURRENT ADDRESS.
                  ;
                  UPDAD:
 035F 2AF620              LHLD    CURAD    ; GET CURRENT ADDRESS
 0362 EB                  XCHG             ; ARG - PUT CURRENT ADDRESS IN D & E
 0363 CD6C02              CALL    HXDSP    ; EXPAND CURRENT ADDRESS FOR DISPLAY
                                           ; ARG - ADDRESS OF EXPANDED ADDRESS IS IN H & L
 0366 AF                  XRA     A        ; ARG - USE ADDRESS FIELD OF DISPLAY
                                           ; ARG - DOT FLAG IS IN B
 0367 CDB702              CALL    OUTPT    ; OUTPUT CURRENT ADDRESS TO ADDRESS FIELD
 036A C9                  RET
                  ;
                  ;*********************************************************************
                  ;
                  ; FUNCTION: UPDDT - UPDATE DATA FIELD OF DISPLAY
                  ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT AT RIGHT EDGE OF FIELD
                  ;                        0 MEANS NO DOT
                  ; OUTPUTS: NONE
                  ; CALLS: HXDSP,OUTDT
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION: UPDDT UPDATES THE DATA FIELD OF THE DISPLAY USING
                  ;              THE CURRENT DATA BYTE.
                  ;
                  UPDDT:
 036B 3AF820              LDA     CURDT    ; GET CURRENT DATA
 036E 57                  MOV     D,A      ; ARG - PUT CURRENT DATA IN D
 036F CD6C02              CALL    HXDSP    ; EXPAND CURRENT DATA FOR DISPLAY
                                           ; ARG - ADDRESS OF EXPANDED DATA IS IN H & L
 0372 3E01                MVI     A,DTFLD  ; ARG - USE DATA FIELD OF DISPLAY
                                           ; ARG - DOT FLAG IS IN B
 0374 CDB702              CALL    OUTPT    ; OUTPUT CURRENT DATA TO DATA FIELD
```

```
  LOC  OBJ          SEQ         SOURCE STATEMENT

 0377 C9            1093         RET
                    1094 ;
                    1095 ;*******************************************************************
                    1096 ;
                    1097 ;                           MONITOR TABLES
                    1098 ;
                    1099 ;*******************************************************************
                    1100 ;
                    1101 ; COMMAND TABLE
                    1102 ;   COMMAND CHARACTERS AS RECEIVED FROM KEYBOARD
                    1103 CMDTB:
 0378 12            1104          DB      12H      ; GO COMMAND
 0379 13            1105          DB      13H      ; SUBSTITUTE MEMORY COMMAND
 037A 14            1106          DB      14H      ; EXAMINE REGISTERS COMMAND
 037B 15            1107          DB      15H      ; SINGLE STEP COMMAND
 0004               1108 NUMC     EQU     $-CMDTB  ; NUMBER OF COMMANDS
                    1109 ;
                    1110 ;*******************************************************************
                    1111 ;
                    1112 ; COMMAND ROUTINE ADDRESS TABLE
                    1113 ; (MUST BE IN REVERSE ORDER OF COMMAND TABLE)
                    1114 CMDAD:
 037C FD00          1115          DW      SSTEP    ; ADDRESS OF SINGLE STEP ROUTINE
 037E 9200          1116          DW      EXAM     ; ADDRESS OF EXAMINE REGISTERS ROUTINE
 0380 8B01          1117          DW      SUBST    ; ADDRESS OF SUBSTITUTE MEMORY ROUTINE
 0382 CB00          1118          DW      GOCMD    ; ADDRESS OF GO ROUTINE
                    1119 ;
                    1120 ;*******************************************************************
                    1121 ;
                    1122 DSPTB:  ; TABLE FOR TRANSLATING CHARACTERS FOR OUTPUT
                    1123 ;
                    1124 ;                DISPLAY
                    1125 ;                FORMAT   CHARACTER
                    1126 ;                =======  =========
                    1127 ;
 0000               1128 ZERO     EQU     $ - DSPTB
 0384 F3            1129          DB      0F3H     ; 0
 0385 60            1130          DB       60H     ; 1
 0386 B5            1131          DB      0B5H     ; 2
 0387 F4            1132          DB      0F4H     ; 3
 0388 66            1133          DB       66H     ; 4
 0005               1134 FIVE     EQU     $ - DSPTB
 0005               1135 LETRS    EQU     $ - DSPTB
 0389 D6            1136          DB      0D6H     ; 5 AND S
 038A D7            1137          DB      0D7H     ; 6
 038B 70            1138          DB       70H     ; 7
 0008               1139 EIGHT    EQU     $ - DSPTB
 038C F7            1140          DB      0F7H     ; 8
 038D 76            1141          DB       76H     ; 9
 000A               1142 LETRA    EQU     $ - DSPTB
 038E 77            1143          DB       77H     ; A
 000B               1144 LETRB    EQU     $ - DSPTB
 038F C7            1145          DB      0C7H     ; B (LOWER CASE)
 000C               1146 LETRC    EQU     $ - DSPTB
 0390 93            1147          DB       93H     ; C
 000D               1148 LETRD    EQU     $ - DSPTB
 0391 E5            1149          DB      0E5H     ; D (LOWER CASE)
 000E               1150 LETRE    EQU     $ - DSPTB
 0392 97            1151          DB       97H     ; E
 000F               1152 LETRF    EQU     $ - DSPTB
 0393 17            1153          DB       17H     ; F
 0010               1154 LETRH    EQU     $ - DSPTB
 0394 67            1155          DB       67H     ; H
 0011               1156 LETRL    EQU     $ - DSPTB
 0395 83            1157          DB       83H     ; L
 0012               1158 LETRP    EQU     $ - DSPTB
 0396 37            1159          DB       37H     ; P
 0013               1160 LETRI    EQU     $ - DSPTB
 0397 60            1161          DB       60H     ; I
 0014               1162 LETRR    EQU     $ - DSPTB
 0398 05            1163          DB       05H     ; R (LOWER CASE)
 0015               1164 BLANK    EQU     $ - DSPTB
 0399 00            1165          DB       00H     ; BLANK
                    1166 ;
                    1167 ;*******************************************************************
                    1168 ;
                    1169 ; MESSAGES FOR OUTPUT TO DISPLAY
                    1170 ;
 039A 15            1171 BLNKS:  DB      BLANK,BLANK,BLANK,BLANK ; FOR ADDRESS OR DATA FIELD
 039B 15
 039C 15
 039D 15
 039E 15            1172 ERMSG:  DB      BLANK,LETRE,LETRR,LETRR ; ERROR MESSAGE FOR ADDR. FIELD
 039F 0E
 03A0 14
 03A1 14
 03A2 0E            1173 EXMSG:  DB      LETRE,BLANK,BLANK,BLANK ; EXECUTION MESSAGE
 03A3 15
 03A4 15
 03A5 15
                    1174                                     ; /FOR ADDRESS FIELD
 03A6 15            1175 SGNAD:  DB      BLANK,BLANK,EIGHT,ZERO  ; SIGN ON MESSAGE (ADDR. FIELD)
 03A7 15
 03A8 08
 03A9 00
 03AA 08            1176 SGNDT:  DB      EIGHT,FIVE              ; SIGN ON MESSAGE (DATA FIELD)
 03AB 05
                    1177 ;
                    1178 ;*******************************************************************
```

```
  LOC  OBJ         SEQ       SOURCE STATEMENT

                   1179 ;
                   1180 RGPTB:  ; REGISTER POINTER TABLE
                   1181                 ; THE ENTRIES IN THIS TABLE ARE IN THE SAME ORDER
                   1182                 ; AS THE REGISTER DESIGNATOR KEYS ON THE KEYBOARD.
                   1183                 ; EACH ENTRY CONTAINS THE REGISTER POINTER VALUE WHICH
                   1184                 ; CORRESPONDS TO THE REGISTER DESIGNATOR. REGISTER
                   1185                 ; POINTER VALUES ARE USED TO POINT INTO THE REGISTER
                   1186                 ; NAME TABLE (NMTBL) AND REGISTER SAVE LOCATION
                   1187                 ; TABLE (RGTBL).
                   1188 ;
  03AC 06          1189         DB      6       ; INTERRUPT MASK
  03AD 09          1190         DB      9       ; SPH
  03AE 0A          1191         DB      10      ; SPL
  03AF 0B          1192         DB      11      ; PCH
  03B0 0C          1193         DB      12      ; PCL
  03B1 07          1194         DB      7       ; H
  03B2 08          1195         DB      8       ; L
  03B3 00          1196         DB      0       ; A
  03B4 01          1197         DB      1       ; B
  03B5 02          1198         DB      2       ; C
  03B6 03          1199         DB      3       ; D
  03B7 04          1200         DB      4       ; E
  03B8 05          1201         DB      5       ; FLAGS
                   1202 ;
                   1203 ;*********************************************************************
                   1204 ;
                   1205 NMTBL:          ; REGISTER NAME TABLE
                   1206                 ; NAMES OF REGISTERS IN DISPLAY FORMAT
  03B9 15          1207         DB      BLANK,BLANK,BLANK,LETRA  ; A REGISTER
  03BA 15
  03BB 15
  03BC 0A
  03BD 15          1208         DB      BLANK,BLANK,BLANK,LETRB  ; B REGISTER
  03BE 15
  03BF 15
  03C0 0B
  03C1 15          1209         DB      BLANK,BLANK,BLANK,LETRC  ; C REGISTER
  03C2 15
  03C3 15
  03C4 0C
  03C5 15          1210         DB      BLANK,BLANK,BLANK,LETRD  ; D REGISTER
  03C6 15
  03C7 15
  03C8 0D
  03C9 15          1211         DB      BLANK,BLANK,BLANK,LETRE  ; E REGISTER
  03CA 15
  03CB 15
  03CC 0E
  03CD 15          1212         DB      BLANK,BLANK,BLANK,LETRF  ; FLAGS
  03CE 15
  03CF 15
  03D0 0F
  03D1 15          1213         DB      BLANK,BLANK,BLANK,LETRI  ; INTERRUPT MASK
  03D2 15
  03D3 15
  03D4 13
  03D5 15          1214         DB      BLANK,BLANK,BLANK,LETRH  ; H REGISTER
  03D6 15
  03D7 15
  03D8 10
  03D9 15          1215         DB      BLANK,BLANK,BLANK,LETRL  ; L REGISTER
  03DA 15
  03DB 15
  03DC 11
  03DD 15          1216         DB      BLANK,LETRS,LETRP,LETRH  ; STACK POINTER HIGH ORDER BYTE
  03DE 05
  03DF 12
  03E0 10
  03E1 15          1217         DB      BLANK,LETRS,LETRP,LETRL  ; STACK POINTER LOW ORDER BYTE
  03E2 05
  03E3 12
  03E4 11
  03E5 15          1218         DB      BLANK,LETRP,LETRC,LETRH  ; PROGRAM COUNTER HIGH BYTE
  03E6 12
  03E7 0C
  03E8 10
  03E9 15          1219         DB      BLANK,LETRP,LETRC,LETRL  ; PROGRAM COUNTER LOW BYTE
  03EA 12
  03EB 0C
  03EC 11
                   1220 ;
                   1221 ;*********************************************************************
                   1222 ;
                   1223 ; REGISTER SAVE LOCATION TABLE
                   1224 ; ADDRESSES OF SAVE LOCATIONS OF REGISTERS IN THE ORDER IN WHICH
                   1225 ; THE REGISTERS ARE DISPLAYED BY THE EXAMINE COMMAND
                   1226 ;
                   1227 RGTBL:
  03ED EE          1228         DB      ASAV AND 0FFH   ; A REGISTER
  03EE EC          1229         DB      BSAV AND 0FFH   ; B REGISTER
  03EF EB          1230         DB      CSAV AND 0FFH   ; C REGISTER
  03F0 EA          1231         DB      DSAV AND 0FFH   ; D REGISTER
  03F1 E9          1232         DB      ESAV AND 0FFH   ; E REGISTER
  03F2 ED          1233         DB      FSAV AND 0FFH   ; FLAGS
  03F3 F1          1234         DB      ISAV AND 0FFH   ; INTERRUPT MASK
  03F4 F0          1235         DB      HSAV AND 0FFH   ; H REGISTER
  03F5 EF          1236         DB      LSAV AND 0FFH   ; L REGISTER
  03F6 F5          1237         DB      SPHSV AND 0FFH  ; STACK POINTER HIGH ORDER BYTE
  03F7 F4          1238         DB      SPLSV AND 0FFH  ; STACK POINTER LOW ORDER BYTE
  03F8 F3          1239         DB      PCHSV AND 0FFH  ; PROGRAM COUNTER HIGH ORDER BYTE
```

```
 LOC  OBJ         SOURCE STATEMENT

 03F9 F2                 DB      PCLSV AND 0FFH  ; PROGRAM COUNTER LOW ORDER BYTE
 000D             NUMRG  EQU     ($ - RGTBL)     ; NUMBER OF ENTRIES IN
                                                 ; /REGISTER SAVE LOCATION TABLE
                  ;
                  ;*********************************************************************
                  ;*********************************************************************
                  ;
                  ;                         SDK-85 TTY MONITOR
                  ;
                  ;*********************************************************************
                  ;*********************************************************************
                  ;
                  ;
                  ; ABSTRACT
                  ; ========
                  ;
                  ; THIS PROGRAM WAS ADAPTED, WITH FEW CHANGES, FROM THE SDK-80 MONITOR.
                  ; THIS PROGRAM RUNS ON THE 8085 BOARD AND IS DESIGNED TO PROVIDE
                  ; THE USER WITH A MINIMAL MONITOR.  BY USING THIS PROGRAM,
                  ; THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD
                  ; A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS
                  ; ALREADY IN MEMORY.  THE MONITOR ALSO PROVIDES THE USER WITH
                  ; ROUTINES FOR PERFORMING CONSOLE I/O.
                  ;
                  ;
                  ; PROGRAM ORGANIZATION
                  ; ======= ============
                  ;
                  ; THE LISTING IS ORGANIZED IN THE FOLLOWING WAY.  FIRST THE COMMAND
                  ; RECOGNIZER, WHICH IS THE HIGHEST LEVEL ROUTINE IN THE PROGRAM.
                  ; NEXT THE ROUTINES TO IMPLEMENT THE VARIOUS COMMANDS.  FINALLY,
                  ; THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK.  WITHIN
                  ; EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL
                  ; ORDER, BY ENTRY POINT OF THE ROUTINE.
                  ;
                  ; MACROS USED IN THE TTY MONITOR ARE DEFINED IN THE KEYBOARD MONITOR.
                  ;
                  ; LIST OF FUNCTIONS
                  ; ==== == =========
                  ;
                  ;       GETCM
                  ;       -----
                  ;
                  ;       DCMD
                  ;       GCMD
                  ;       ICMD
                  ;       MCMD
                  ;       SCMD
                  ;       XCMD
                  ;       -----
                  ;
                  ;       CI
                  ;       CNVBN
                  ;       CO
                  ;       CROUT
                  ;       DELAY
                  ;       ECHO
                  ;       ERROR
                  ;       FRET
                  ;       GETCH
                  ;       GETHX
                  ;       GETNM
                  ;       HILO
                  ;       NMOUT
                  ;       PRVAL
                  ;       REGDS
                  ;       RGADR
                  ;       SRET
                  ;       STHF0
                  ;       STHLF
                  ;       VALDG
                  ;       VALDL
                  ;       -----
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ;                         MONITOR EQUATES
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
 001B             BRCHR   EQU     1BH     ; CODE FOR BREAK CHARACTER (ESCAPE)
 07FA             BRTAB   EQU     07FAH   ; LOCATION OF START OF BRANCH TABLE IN ROM
 000D             CR      EQU     0DH     ; CODE FOR CARRIAGE RETURN
 001B             ESC     EQU     1BH     ; CODE FOR ESCAPE CHARACTER
 000F             HCHAR   EQU     0FH     ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
 00FF             INVRT   EQU     0FFH    ; MASK TO INVERT HALF BYTE FLAG
 000A             LF      EQU     0AH     ; CODE FOR LINE FEED
 0000             LOWER   EQU     0       ; DENOTES LOWER HALF OF BYTE IN ICMD
                  ;LSGNON EQU     ---     ; LENGTH OF SIGNON MESSAGE - DEFINED LATER
                  ;MNSTK  EQU     ---     ; START OF MONITOR STACK - DEFINED IN
                                          ; /KEYBOARD MONITOR
                  ;NCMDS  EQU     ---     ; NUMBER OF VALID COMMANDS - DEFINED LATER
 000F             NEWLN   EQU     0FH     ; MASK FOR CHECKING MEMORY ADDR DISPLAY
 007F             PRTY0   EQU     07FH    ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
                  ;RAMST  EQU     ---     ; START ADDRESS OF RAM - DEFINED IN
                                          ; KEYBOARD MONITOR
```

```
 LOC  OBJ        SOURCE STATEMENT

                ;RTABS  EQU     ---      ; SIZE OF ENTRY IN RTAB TABLE
 0080           SSTRT   EQU     80H      ; SHIFTED START BIT
 0040           STOPB   EQU     40H      ; STOP BIT
 00C0           STRT    EQU     0C0H     ; UNSHIFTED START BIT
 001B           TERM    EQU     1BH      ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
 00FF           UPPER   EQU     0FFH     ; DENOTES UPPER HALF OF BYTE IN ICMD

                ;DELAY VALUES IF NO WAIT STATE
                ;
                        IF      1-WAITS
 048C           IBTIM   EQU     1164     ;INTER-BIT TIME DELAY
 048C           OBTIM   EQU     1164     ;OUTPUT INTER-BIT TIME DELAY
 1230           TIM4    EQU     4656     ;4 BIT TIME DELAY
 0246           WAIT    EQU     582      ;DELAY UNTIL READY TO SAMPLE BITS
                        ENDIF
                ;
                ;DELAY VALUES IF ONE WAIT STATE
                ;
                        IF      WAITS
                IBTIM   EQU     930      ;INTER-BIT DELAY
                OBTIM   EQU     930      ;OUTPUT INTER-BIT TIME DELAY
                TIM4    EQU     3720     ;4 BIT TIME DELAY
                WAIT    EQU     465      ;DELAY UNTIL READY TO SAMPLE BITS
                        ENDIF
                ;
                ;
                ;****************************************************************
                ;
                ;
                ;                       RESTART ENTRY POINT
                ;
                ;
                ;****************************************************************
                ;
                ;
                ;
                ;************************************************************
                ;
                ;
                ;                       PRINT SIGNON MESSAGE
                ;
                ;
                ;************************************************************
                ;
                ;
                GO:
 03FA 218C07            LXI     H,SGNON ; GET ADDRESS OF SIGNON MESSAGE
 03FD 0614              MVI     B,LSGNON        ; COUNTER FOR CHARACTERS IN MESSAGE
                MSGL:
 03FF 4E                MOV     C,M      ; FETCH NEXT CHAR TO C REG
 0400 CDC405            CALL    CO       ; SEND IT TO THE CONSOLE
 0403 23                INX     H        ; POINT TO NEXT CHARACTER
 0404 05                DCR     B        ; DECREMENT BYTE COUNTER
 0405 C2FF03            JNZ     MSGL     ; RETURN FOR NEXT CHARACTER
                ;
                ;
                ;****************************************************************
                ;
                ;
                ;                       COMMAND RECOGNIZING ROUTINE
                ;
                ;
                ;****************************************************************
                ;
                ; FUNCTION: GETCM
                ; INPUTS: NONE
                ; OUTPUTS: NONE
                ; CALLS: GETCH,ECHO,ERROR
                ; DESTROYS: A,B,C,H,L,F/F'S
                ; DESCRIPTION: GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
                ;              AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
                ;              CHARACTER TABLE.  IF SUCCESSFUL, THE ROUTINE
                ;              CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
                ;              A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
                ;              IS TRANSFERRED TO THIS ROUTINE.  IF THE CHARACTER
                ;              DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
                ;              THE ERROR HANDLER.
                ;
                GETCM:
 0408 21E920            LXI     H,MNSTK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
 040B F9                SPHL             ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
 040C 0E2E              MVI     C,'.'    ; PROMPT CHARACTER TO C
 040E CDF805            CALL    ECHO     ; SEND PROMPT CHARACTER TO USER TERMINAL
 0411 C31404            JMP     GTC03    ; WANT TO LEAVE ROOM FOR RST BRANCH
                GTC03:
 0414 CD1F06            CALL    GETCH    ; GET COMMAND CHARACTER TO A
 0417 CDF805            CALL    ECHO     ; ECHO CHARACTER TO USER
 041A 79                MOV     A,C      ; PUT COMMAND CHARACTER INTO ACCUMULATOR
 041B 010600            LXI     B,NCMDS ; C CONTAINS LOOP AND INDEX COUNT
 041E 21AE07            LXI     H,CTAB  ; HL POINTS INTO COMMAND TABLE
                GTC05:
 0421 BE                CMP     M        ; COMPARE TABLE ENTRY AND CHARACTER
 0422 CA2D04            JZ      GTC10    ; BRANCH IF EQUAL - COMMAND RECOGNIZED
 0425 23                INX     H        ; ELSE, INCREMENT TABLE POINTER
 0426 0D                DCR     C        ; DECREMENT LOOP COUNT
 0427 C22104            JNZ     GTC05    ; BRANCH IF NOT AT TABLE END
 042A C31106            JMP     ERROR    ; ELSE, COMMAND CHARACTER IS ILLEGAL
                GTC10:
 042D 21A007            LXI     H,CADR  ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
                                         ; /OF COMMAND ROUTINE ADDRESSES
```

```
 LOC  OBJ         SEQ        SOURCE STATEMENT

 0430 09          1440          DAD     B        ; ADD WHAT IS LEFT OF LOOP COUNT
 0431 09          1441          DAD     B        ; ADD AGAIN - EACH ENTRY IN CADR IS 2 BYTES LONG
 0432 7E          1442          MOV     A,M      ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
 0433 23          1443          INX     H        ; POINT TO NEXT BYTE IN TABLE
 0434 66          1444          MOV     H,M      ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
 0435 6F          1445          MOV     L,A      ; PUT LSP OF ADDRESS OF TABLE ENTRY INTO L
 0436 E9          1446          PCHL             ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE
                  1447 ;
                  1448 ;
                  1449 ;*******************************************************************
                  1450 ;
                  1451 ;
                  1452 ;              COMMAND IMPLEMENTING ROUTINES
                  1453 ;
                  1454 ;
                  1455 ;*******************************************************************
                  1456 ;
                  1457 ;
                  1458 ; FUNCTION: DCMD
                  1459 ; INPUTS: NONE
                  1460 ; OUTPUTS: NONE
                  1461 ; CALLS: ECHO,NMOUT,HILO,GETCM,CROUT,GETNM
                  1462 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1463 ; DESCRIPTION: DCMD IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND
                  1464 ;
                  1465 DCMD:
 0437 0E02        1466          MVI     C,2      ; GET 2 NUMBERS FROM INPUT STREAM
 0439 CD5B06      1467          CALL    GETNM
 043C D1          1468          POP     D        ; ENDING ADDRESS TO DE
 043D E1          1469          POP     H        ; STARTING ADDRESS TO HL
                  1470 DCM05:
 043E CDEB05      1471          CALL    CROUT    ; ECHO CARRIAGE RETURN/LINE FEED
 0441 7C          1472          MOV     A,H      ; DISPLAY ADDRESS OF FIRST LOCATION IN LINE
 0442 CDC706      1473          CALL    NMOUT
 0445 7D          1474          MOV     A,L      ; ADDRESS IS 2 BYTES LONG
 0446 CDC706      1475          CALL    NMOUT
                  1476 DCM10:
 0449 0E20        1477          MVI     C,' '
 044B CDF805      1478          CALL    ECHO     ; USE BLANK AS SEPARATOR
 044E 7E          1479          MOV     A,M      ; GET CONTENTS OF NEXT MEMORY LOCATION
 044F CDC706      1480          CALL    NMOUT    ; DISPLAY CONTENTS
 0452 CDA006      1481          CALL    HILO     ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
                  1482                           ; /GREATER THAN OR EQUAL TO ENDING ADDRESS
                  1483          FALSE   DCM15    ; IF NOT, MORE TO DISPLAY
 0455 D25E04      1484+         JNC     DCM15
 0458 CDEB05      1485          CALL    CROUT    ; CARRIAGE RETURN/LINE FEED TO END LINE
 045B C30804      1486          JMP     GETCM    ; ALL DONE
                  1487 DCM15:
 045E 23          1488          INX     H        ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
 045F 7D          1489          MOV     A,L      ; GET LOW ORDER BITS OF NEW ADDRESS
 0460 E60F        1490          ANI     NEWLN    ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
                  1491                           ; /START OF NEW LINE
 0462 C24904      1492          JNZ     DCM10    ; NO - NOT AT END OF LINE
 0465 C33E04      1493          JMP     DCM05    ; YES - START NEW LINE WITH ADDRESS
                  1494 ;
                  1495 ;
                  1496 ;*******************************************************************
                  1497 ;
                  1498 ;
                  1499 ; FUNCTION: GCMD
                  1500 ; INPUTS: NONE
                  1501 ; OUTPUTS: NONE
                  1502 ; CALLS: ERROR,GETHX,RSTTF
                  1503 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1504 ; DESCRIPTION: GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.
                  1505 ;
                  1506 GCMD:
 0468 CD2606      1507          CALL    GETHX    ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
                  1508          FALSE   GCM05    ; BRANCH IF NO NUMBER PRESENT
 046B D27D04      1509+         JNC     GCM05
 046E 7A          1510          MOV     A,D      ; ELSE, GET TERMINATOR
 046F FE0D        1511          CPI     CR       ; SEE IF CARRIAGE RETURN
 0471 C21106      1512          JNZ     ERROR    ; ERROR IF NOT PROPERLY TERMINATED
 0474 21F220      1513          LXI     H,PSAV ; WANT NUMBER TO REPLACE SAVE PGM COUNTER
 0477 71          1514          MOV     M,C
 0478 23          1515          INX     H
 0479 70          1516          MOV     M,B
 047A C38304      1517          JMP     GCM10
                  1518 GCM05:
 047D 7A          1519          MOV     A,D      ; IF NO STARTING ADDRESS, MAKE SURE THAT
 047E FE0D        1520          CPI     CR       ; /CARRIAGE RETURN TERMINATED COMMAND
 0480 C21106      1521          JNZ     ERROR    ; ERROR IF NOT
                  1522 GCM10:
 0483 C31B03      1523          JMP     RSTOR    ; RESTORE REGISTERS AND BEGIN EXECUTION
                  1524                           ; (RSTOR IS IN KEYBOARD MONITOR)
                  1525 ;
                  1526 ;
                  1527 ;*******************************************************************
                  1528 ;
                  1529 ;
                  1530 ; FUNCTION: ICMD
                  1531 ; INPUTS: NONE
                  1532 ; OUTPUTS: NONE
                  1533 ; CALLS: ERROR,ECHO,GETCH,VALDL,VALDG,CNVBN,STHLF,GETNM,CROUT
                  1534 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1535 ; DESCRIPTION: ICMD IMPLEMENTS THE INSERT CODE INTO MEMORY (I) COMMAND.
                  1536 ;
                  1537 ICMD:
 0486 0E01        1538          MVI     C,1
```

```
LOC  OBJ            SEQ       SOURCE STATEMENT

0488 CD5B06                   CALL    GETNM   ; GET SINGLE NUMBER FROM INPUT STREAM
048B 3EFF                     MVI     A,UPPER
048D 32FD20                   STA     TEMP    ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
0490 D1                       POP     D       ; ADDRESS OF START TO DE
                    ICM05:
0491 CD1F06                   CALL    GETCH   ; GET A CHARACTER FROM INPUT STREAM
0494 4F                       MOV     C,A
0495 CDF805                   CALL    ECHO    ; ECHO IT
0498 79                       MOV     A,C     ; PUT CHARACTER BACK INTO A
0499 FE1B                     CPI     TERM    ; SEE IF CHARACTER IS A TERMINATING CHARACTER
049B CAC704                   JZ      ICM25   ; IF SO, ALL DONE ENTERING CHARACTERS
049E CD7907                   CALL    VALDL   ; ELSE, SEE IF VALID DELIMITER
                              TRUE    ICM05   ; IF SO SIMPLY IGNORE THIS CHARACTER
04A1 DA9104                   JC      ICM05
04A4 CD5E07                   CALL    VALDG   ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
                              FALSE   ICM20   ; IF NOT, BRANCH TO HANDLE ERROR CONDITION
04A7 D2C104                   JNC     ICM20
04AA CDBB05                   CALL    CNVBN   ; CONVERT DIGIT TO BINARY
04AD 4F                       MOV     C,A     ; MOVE RESULT TO C
04AE CD3F07                   CALL    STHLF   ; STORE IN APPROPRIATE HALF WORD
04B1 3AFD20                   LDA     TEMP    ; GET HALF BYTE FLAG
04B4 B7                       ORA     A       ; SET F/F'S
04B5 C2B904                   JNZ     ICM10   ; BRANCH IF FLAG SET FOR UPPER
04B8 13                       INX     D       ; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
                    ICM10:
04B9 EEFF                     XRI     INVRT   ; TOGGLE STATE OF FLAG
04BB 32FD20                   STA     TEMP    ; PUT NEW VALUE OF FLAG BACK
04BE C39104                   JMP     ICM05   ; PROCESS NEXT DIGIT
                    ICM20:
04C1 CD3407                   CALL    STHF0   ; ILLEGAL CHARACTER
04C4 C31106                   JMP     ERROR   ; MAKE SURE ENTIRE BYTE FILLED THEN ERROR
                    ICM25:
04C7 CD3407                   CALL    STHF0   ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
04CA CDEB05                   CALL    CROUT   ; ADD CARRIAGE RETURN
04CD C30804                   JMP     GETCM
                    ;
                    ;
                    ;*******************************************************************
                    ;
                    ;
                    ; FUNCTION: MCMD
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: GETCM,HILO,GETNM
                    ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    ; DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.
                    ;
                    MCMD:
04D0 0E03                     MVI     C,3
04D2 CD5B06                   CALL    GETNM   ; GET 3 NUMBERS FROM INPUT STREAM
04D5 C1                       POP     B       ; DESTINATION ADDRESS TO BC
04D6 E1                       POP     H       ; ENDING ADDRESS TO HL
04D7 D1                       POP     D       ; STARTING ADDRESS TO DE
                    MCM05:
04D8 E5                       PUSH    H       ; SAVE ENDING ADDRESS
04D9 62                       MOV     H,D
04DA 6B                       MOV     L,E     ; SOURCE ADDRESS TO HL
04DB 7E                       MOV     A,M     ; GET SOURCE BYTE
04DC 60                       MOV     H,B
04DD 69                       MOV     L,C     ; DESTINATION ADDRESS TO HL
04DE 77                       MOV     M,A     ; MOVE BYTE TO DESTINATION
04DF 03                       INX     B       ; INCREMENT DESTINATION ADDRESS
04E0 78                       MOV     A,B
04E1 B1                       ORA     C       ; TEST FOR DESTINATION ADDRESS OVERFLOW
04E2 CA0804                   JZ      GETCM   ; IF SO, CAN TERMINATE COMMAND
04E5 13                       INX     D       ; INCREMENT SOURCE ADDRESS
04E6 E1                       POP     H       ; ELSE, GET BACK ENDING ADDRESS
04E7 CDA006                   CALL    HILO    ; SEE IF ENDING ADDR>=SOURCE ADDR
                              FALSE   GETCM   ; IF NOT, COMMAND IS DONE
04EA D20804                   JNC     GETCM
04ED C3D804                   JMP     MCM05   ; MOVE ANOTHER BYTE
                    ;
                    ;
                    ;*******************************************************************
                    ;
                    ;
                    ; FUNCTION: SCMD
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: GETHX,GETCM,NMOUT,ECHO
                    ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    ; DESCRIPTION: SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.
                    ;
                    SCMD:
04F0 CD2606                   CALL    GETHX   ; GET A NUMBER, IF PRESENT, FROM INPUT
04F3 C5                       PUSH    B
04F4 E1                       POP     H       ; GET NUMBER TO HL - DENOTES MEMORY LOCATION
                    SCM05:
04F5 7A                       MOV     A,D     ; GET TERMINATOR
04F6 FE20                     CPI     ' '     ; SEE IF SPACE
04F8 CA0005                   JZ      SCM10   ; YES - CONTINUE PROCESSING
04FB FE2C                     CPI     ','     ; ELSE, SEE IF COMMA
04FD C20804                   JNZ     GETCM   ; NO - TERMINATE COMMAND
                    SCM10:
0500 7E                       MOV     A,M     ; GET CONTENTS OF SPECIFIED LOCATION TO A
0501 CDC706                   CALL    NMOUT   ; DISPLAY CONTENTS ON CONSOLE
0504 0E2D                     MVI     C,'-'
0506 CDF805                   CALL    ECHO    ; USE DASH FOR SEPARATOR
0509 CD2606                   CALL    GETHX   ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
```

```
 LOC  OBJ         SEQ         SOURCE STATEMENT


                  1638          FALSE   SCM15    ; IF NO VALUE PRESENT, BRANCH
050C D21005       1639+         JNC     SCM15
050F 71           1640          MOV     M,C      ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
                  1641 SCM15:
0510 23           1642          INX     H        ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
0511 C3F504       1643          JMP     SCM05
                  1644 ;
                  1645 ;
                  1646 ;************************************************************
                  1647 ;
                  1648 ;
                  1649 ; FUNCTION: XCMD
                  1650 ; INPUTS: NONE
                  1651 ; OUTPUTS: NONE
                  1652 ; CALLS: GETCH,ECHO,REGDS,GETCM,ERROR,RGADR,NMOUT,CROUT,GETHX
                  1653 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1654 ; DESCRIPTION: XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)
                  1655 ;               COMMAND.
                  1656 ;
                  1657 XCMD:
0514 CD1F06       1658          CALL    GETCH    ; GET REGISTER IDENTIFIER
0517 4F           1659          MOV     C,A
0518 CDF805       1660          CALL    ECHO     ; ECHO IT
051B 79           1661          MOV     A,C
051C FE0D         1662          CPI     CR
051E C22705       1663          JNZ     XCM05    ; BRANCH IF NOT CARRIAGE RETURN
0521 CDEA06       1664          CALL    REGDS    ; ELSE, DISPLAY REGISTER CONTENTS
0524 C30804       1665          JMP     GETCM    ; THEN TERMINATE COMMAND
                  1666 XCM05:
0527 4F           1667          MOV     C,A      ; GET REGISTER IDENTIFIER TO C
0528 CD1B07       1668          CALL    RGADR    ; CONVERT IDENTIFIER INTO RTAB TABLE ADDR
052B C5           1669          PUSH    B
052C E1           1670          POP     H        ; PUT POINTER TO REGISTER ENTRY INTO HL
052D 0E20         1671          MVI     C,' '
052F CDF805       1672          CALL    ECHO     ; ECHO SPACE TO USER
0532 79           1673          MOV     A,C
0533 32FD20       1674          STA     TEMP     ; PUT SPACE INTO TEMP AS DELIMITER
                  1675 XCM10:
0536 3AFD20       1676          LDA     TEMP     ; GET TERMINATOR
0539 FE20         1677          CPI     ' '      ; SEE IF A BLANK
053B CA4305       1678          JZ      XCM15    ; YES - GO CHECK POINTER INTO TABLE
053E FE2C         1679          CPI     ','      ; NO - SEE IF COMMA
0540 C20804       1680          JNZ     GETCM    ; NO - MUST BE CARRIAGE RETURN TO END COMMAND
                  1681 XCM15:
0543 7E           1682          MOV     A,M
0544 B7           1683          ORA     A        ; SET F/F'S
0545 C24E05       1684          JNZ     XCM18    ; BRANCH IF NOT AT END OF TABLE
0548 CDEB05       1685          CALL    CROUT    ; ELSE, OUTPUT CARRIAGE RETURN LINE FEED
054B C30804       1686          JMP     GETCM    ; AND EXIT
                  1687 XCM18:
054E E5           1688          PUSH    H        ; PUT POINTER ON STACK
054F 5E           1689          MOV     E,M
0550 1620         1690          MVI     D,RAMST SHR 8    ; ADDRESS OF SAVE LOCATION FROM TABLE
0552 23           1691          INX     H
0553 46           1692          MOV     B,M      ; FETCH LENGTH FLAG FROM TABLE
0554 D5           1693          PUSH    D        ; SAVE ADDRESS OF SAVE LOCATION
0555 D5           1694          PUSH    D
0556 E1           1695          POP     H        ; MOVE ADDRESS TO HL
0557 C5           1696          PUSH    B        ; SAVE LENGTH FLAG
0558 7E           1697          MOV     A,M      ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
0559 CDC706       1698          CALL    NMOUT    ; DISPLAY IT
055C F1           1699          POP     PSW      ; GET BACK LENGTH FLAG
055D F5           1700          PUSH    PSW      ; SAVE IT AGAIN
055E B7           1701          ORA     A        ; SET F/F'S
055F CA6705       1702          JZ      XCM20    ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
0562 2B           1703          DCX     H        ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
0563 7E           1704          MOV     A,M
0564 CDC706       1705          CALL    NMOUT    ; DISPLAY THEM
                  1706 XCM20:
0567 0E2D         1707          MVI     C,'-'
0569 CDF805       1708          CALL    ECHO     ; USE DASH AS SEPARATOR
056C CD2606       1709          CALL    GETHX    ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
                  1710          FALSE   XCM30    ; NO - GO CHECK FOR NEXT REGISTER
056F D28705       1711+         JNC     XCM30
0572 7A           1712          MOV     A,D
0573 32FD20       1713          STA     TEMP     ; ELSE, SAVE THE TERMINATOR FOR NOW
0576 F1           1714          POP     PSW      ; GET BACK LENGTH FLAG
0577 E1           1715          POP     H        ; PUT ADDRESS OF SAVE LOCATION INTO HL
0578 B7           1716          ORA     A        ; SET F/F'S
0579 CA7E05       1717          JZ      XCM25    ; IF 8 BIT REGISTER, BRANCH
057C 70           1718          MOV     M,B      ; SAVE UPPER 8 BITS
057D 2B           1719          DCX     H        ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
                  1720 XCM25:
057E 71           1721          MOV     M,C      ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
                  1722 XCM27:
057F 110300       1723          LXI     D,RTABS  ; SIZE OF ENTRY IN RTAB TABLE
0582 E1           1724          POP     H        ; POINTER INTO REGISTER TABLE RTAB
0583 19           1725          DAD     D        ; ADD ENTRY SIZE TO POINTER
0584 C33605       1726          JMP     XCM10    ; DO NEXT REGISTER
                  1727 XCM30:
0587 7A           1728          MOV     A,D      ; GET TERMINATOR
0588 32FD20       1729          STA     TEMP     ; SAVE IN MEMORY
058B D1           1730          POP     D        ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
058C D1           1731          POP     D        ; /OF SAVE LOCATION
058D C37F05       1732          JMP     XCM27    ; GO INCREMENT REGISTER TABLE POINTER
                  1733 ;
                  1734 ;
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                  ;*****************************************************************
                  ;
                  ;
                  ;                         UTILITY ROUTINES
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: CI
                  ; INPUTS: NONE
                  ; OUTPUTS: A - CHARACTER FROM TTY
                  ; CALLS: DELAY
                  ; DESTROYS: A,F/F'S
                  ; DESCRIPTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
                  ;              TTY AND THEN RETURNS THE CHARACTER, VIA THE A
                  ;              REGISTER, TO THE CALLING ROUTINE.  THIS ROUTINE
                  ;              IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.
                  ;
                  CI:
0590 F3                   DI
0591 D5                   PUSH    D         ; SAVE DE
                  CI05:
0592 20                   RIM               ; GET INPUT BIT
0593 17                   RAL               ; INTO CARRY WITH IT
0594 DA9205               JC      CI05      ; BRANCH IF NO START BIT
0597 114602               LXI     D,WAIT    ; WAIT UNTIL MIDDLE OF BIT
059A CDF105               CALL    DELAY
059D C5                   PUSH    B         ; SAVE BC
059E 010800               LXI     B,8       ; B<--0, C<--# BITS TO RECEIVE
                  CI10:
05A1 118C04               LXI     D,IBTIM
05A4 CDF105               CALL    DELAY     ; WAIT UNTIL MIDDLE OF NEXT BIT
05A7 20                   RIM               ; GET THE BIT
05A8 17                   RAL               ; INTO CARRY
05A9 78                   MOV     A,B       ; GET PARTIAL RESULT
05AA 1F                   RAR               ; SHIFT IN NEXT DATA BIT
05AB 47                   MOV     B,A       ; REPLACE RESULT
05AC 0D                   DCR     C         ; DEC COUNT OF BITS TO GO
05AD C2A105               JNZ     CI10      ; BRANCH IF MORE LEFT
05B0 118C04               LXI     D,IBTIM   ; ELSE, WANT TO WAIT OUT STOP BIT
05B3 CDF105               CALL    DELAY
05B6 78                   MOV     A,B       ; GET RESULT
05B7 C1                   POP     B
05B8 D1                   POP     D         ; RESTORE SAVED REGISTERS
05B9 FB                   EI
05BA C9                   RET               ; THAT'S IT
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: CNVBN
                  ; INPUTS: C - ASCII CHARACTER '0'-'9' OR 'A'-'F'
                  ; OUTPUTS: A - 0 TO F HEX
                  ; CALLS: NOTHING
                  ; DESTROYS: A,F/F'S
                  ; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
                  ;              CNVBN INTO ITS CORRESPONDING BINARY VALUE.  CNVBN
                  ;              DOES NOT CHECK THE VALIDITY OF ITS INPUT.
                  ;
                  CNVBN:
05BB 79                   MOV     A,C
05BC D630                 SUI     '0'       ; SUBTRACT CODE FOR '0' FROM ARGUMENT
05BE FE0A                 CPI     10        ; WANT TO TEST FOR RESULT OF 0 TO 9
05C0 F8                   RM                ; IF SO, THEN ALL DONE
05C1 D607                 SUI     7         ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
05C3 C9                   RET               ; SO RETURN AFTER SUBTRACTING BIAS OF 7
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: CO
                  ; INPUTS: C - CHARACTER TO OUTPUT TO TTY
                  ; OUTPUTS: C - CHARACTER OUTPUT TO TTY
                  ; CALLS: DELAY
                  ; DESTROYS: A,F/F'S
                  ; DESCRIPTION: CO SENDS ITS INPUT ARGUMENT TO THE TTY.
                  ;
                  CO:
05C4 F3                   DI
05C5 C5                   PUSH    B         ; SAVE BC
05C6 D5                   PUSH    D         ; SAVE DE
05C7 3EC0                 MVI     A,STRT    ; START BIT MASK
05C9 0607                 MVI     B,7       ; B WILL COUNT BITS TO SEND
                  CO05:
05CB 30                   SIM               ; SEND A BIT
05CC 118C04               LXI     D,OBTIM   ; WAIT FOR TTY TO HANDLE IT
05CF CDF105               CALL    DELAY
05D2 79                   MOV     A,C       ; PICK UP BITS LEFT TO SEND
05D3 1F                   RAR               ; LOW ORDER BIT TO CARRY
05D4 4F                   MOV     C,A       ; PUT REST BACK
05D5 3E80                 MVI     A,SSTRT   ; SHIFTED ENABLE BIT
05D7 1F                   RAR               ; SHIFT IN DATA BIT
05D8 EE80                 XRI     80H       ; COMPLEMENT DATA BIT
05DA 05                   DCR     B         ; DEC COUNT
05DB F2CB05               JP      CO05      ; SEND IF MORE BITS NEED TO BE SENT
05DE 3E40                 MVI     A,STOPB   ; ELSE, SEND STOP BITS
05E0 30                   SIM
```

```
  LOC  OBJ          SEQ         SOURCE STATEMENT

 05E1 113012                 LXI     D,TIM4   ; WAIT 4 BIT TIME (FAKE PARITY + 3 STOP BITS)
 05E4 CDF105                 CALL    DELAY
 05E7 D1                     POP     D
 05E8 C1                     POP     B        ; RESTORE SAVED REGISTERS
 05E9 FB                     EI
 05EA C9                     RET              ; ALL DONE
                        ;
                        ;
                        ;***************************************************************
                        ;
                        ;
                        ; FUNCTION CROUT
                        ; INPUTS: NONE
                        ; OUTPUTS: NONE
                        ; CALLS: ECHO
                        ; DESTROYS: A,B,C,F/F'S
                        ; DESCRIPTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE
                        ;              FEED) TO THE CONSOLE.
                        ;
                        CROUT:
 05EB 0E0D                   MVI     C,CR
 05ED CDF805                 CALL    ECHO
 05F0 C9                     RET
                        ;
                        ;
                        ;***************************************************************
                        ;
                        ;
                        ; FUNCTION: DELAY
                        ; INPUTS: DE - 16 BIT INTEGER DENOTING NUMBER OF TIMES TO LOOP
                        ; OUTPUTS: NONE
                        ; CALLS: NOTHING
                        ; DESTROYS: A,D,E,F/F'S
                        ; DESCRIPTION: DELAY DOES NOT RETURN TO CALLER UNTIL INPUT ARGUMENT
                        ;              IS COUNTED DOWN TO 0.
                        ;
                        DELAY:
 05F1 1B                     DCX     D        ; DECREMENT INPUT ARGUMENT
 05F2 7A                     MOV     A,D
 05F3 B3                     ORA     E
 05F4 C2F105                 JNZ     DELAY    ; IF ARGUMENT NOT 0, KEEP GOING
 05F7 C9                     RET
                        ;
                        ;
                        ;***************************************************************
                        ;
                        ;
                        ; FUNCTION: ECHO
                        ; INPUTS: C - CHARACTER TO ECHO TO TERMINAL
                        ; OUTPUTS: C - CHARACTER ECHOED TO TERMINAL
                        ; CALLS: CO
                        ; DESTROYS: A,B,F/F'S
                        ; DESCRIPTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA
                        ;              THE MONITOR, SENDS THAT CHARACTER TO THE USER
                        ;              TERMINAL.  A CARRIAGE RETURN IS ECHOED AS A CARRIAGE
                        ;              RETURN LINE FEED, AND AN ESCAPE CHARACTER IS ECHOED AS $.
                        ;
                        ECHO:
 05F8 41                     MOV     B,C      ; SAVE ARGUMENT
 05F9 3E1B                   MVI     A,ESC
 05FB B8                     CMP     B        ; SEE IF ECHOING AN ESCAPE CHARACTER
 05FC C20106                 JNZ     ECH05    ; NO - BRANCH
 05FF 0E24                   MVI     C,'$'    ; YES - ECHO AS $
                        ECH05:
 0601 CDC405                 CALL    CO       ; DO OUTPUT THROUGH MONITOR
 0604 3E0D                   MVI     A,CR
 0606 B8                     CMP     B        ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RETURN
 0607 C20F06                 JNZ     ECH10    ; NO - NO NEED TO TAKE SPECIAL ACTION
 060A 0E0A                   MVI     C,LF     ; YES - WANT TO ECHO LINE FEED, TOO
 060C CDC405                 CALL    CO
                        ECH10:
 060F 48                     MOV     C,B      ; RESTORE ARGUMENT
 0610 C9                     RET
                        ;
                        ;
                        ;***************************************************************
                        ;
                        ;
                        ; FUNCTION: ERROR
                        ; INPUTS: NONE
                        ; OUTPUTS: NONE
                        ; CALLS: ECHO,CROUT,GETCM
                        ; DESTROYS: A,B,C,F/F'S
                        ; DESCRIPTION: ERROR PRINTS THE ERROR CHARACTER (CURRENTLY AN ASTERISK)
                        ;              ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
                        ;              AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.
                        ;
                        ERROR:
 0611 0E2A                   MVI     C,'*'
 0613 CDF805                 CALL    ECHO     ; SEND * TO CONSOLE
 0616 CDEB05                 CALL    CROUT    ; SKIP TO BEGINNING OF NEXT LINE
 0619 C30804                 JMP     GETCM    ; TRY AGAIN FOR ANOTHER COMMAND
                        ;
                        ;
                        ;***************************************************************
                        ;
                        ;
                        ; FUNCTION: FRET
                        ; INPUTS: NONE
```

```
 LOC  OBJ          SEQ        SOURCE STATEMENT

                   ; OUTPUTS: CARRY - ALWAYS 0
                   ; CALLS: NOTHING
                   ; DESTROYS: CARRY
                   ; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
                   ;              INDICATE FAILURE ON RETURN.  FRET SETS THE CARRY
                   ;              FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
                   ;              CALLER OF THE ROUTINE INVOKING FRET.
                   ;
                   FRET:
061C 37                    STC             ; FIRST SET CARRY TRUE
061D 3F                    CMC             ; THEN COMPLEMENT IT TO MAKE IT FALSE
061E C9                    RET             ; RETURN APPROPRIATELY
                   ;
                   ;
                   ;***********************************************************************
                   ;
                   ;
                   ; FUNCTION: GETCH
                   ; INPUTS: NONE
                   ; OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
                   ; CALLS: CI
                   ; DESTROYS: A,C,F/F'S
                   ; DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
                   ;              TO THE CALLING PROGRAM.
                   ;
                   GETCH:
061F CD9005                CALL    CI      ; GET CHARACTER FROM TERMINAL
0622 E67F                  ANI     PRTY0   ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
0624 4F                    MOV     C,A     ; PUT VALUE IN C REGISTER FOR RETURN
0625 C9                    RET
                   ;
                   ;
                   ;***********************************************************************
                   ;
                   ;
                   ; FUNCTION: GETHX
                   ; INPUTS: NONE
                   ; OUTPUTS: BC - 16 BIT INTEGER
                   ;          D - CHARACTER WHICH TERMINATED THE INTEGER
                   ;          CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
                   ;                - 0 IF FIRST CHARACTER IS DELIMITER
                   ; CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR
                   ; DESTROYS: A,B,C,D,E,F/F'S
                   ; DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
                   ;              STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
                   ;              INTEGER.  IF MORE THAN 4 HEX DIGITS ARE ENTERED,
                   ;              ONLY THE LAST 4 ARE USED.  THE NUMBER TERMINATES WHEN
                   ;              A VALID DELIMITER IS ENCOUNTERED.  THE DELIMITER IS
                   ;              ALSO RETURNED AS AN OUTPUT OF THE FUNCTION.  ILLEGAL
                   ;              CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
                   ;              ERROR INDICATION.  IF THE FIRST (VALID) CHARACTER
                   ;              ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
                   ;              GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
                   ;              OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
                   ;              OF BC ARE UNDEFINED.
                   ;
                   GETHX:
0626 E5                    PUSH    H       ; SAVE HL
0627 210000                LXI     H,0     ; INITIALIZE RESULT
062A 1E00                  MVI     E,0     ; INITIALIZE DIGIT FLAG TO FALSE
                   GHX05:
062C CD1F06                CALL    GETCH   ; GET A CHARACTER
062F 4F                    MOV     C,A
0630 CDF805                CALL    ECHO    ; ECHO THE CHARACTER
0633 CD7907                CALL    VALDL   ; SEE IF DELIMITER
                           FALSE   GHX10   ; NO - BRANCH
0636 D24506        +       JNC     GHX10
0639 51                    MOV     D,C     ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
063A E5                    PUSH    H
063B C1                    POP     B       ; MOVE RESULT TO BC
063C E1                    POP     H       ; RESTORE HL
063D 7B                    MOV     A,E     ; GET FLAG
063E B7                    ORA     A       ; SET F/F'S
063F C23207                JNZ     SRET    ; IF FLAG NON-0, A NUMBER HAS BEEN FOUND
0642 CA1C06                JZ      FRET    ; ELSE, DELIMITER WAS FIRST CHARACTER
                   GHX10:
0645 CD5E07                CALL    VALDG   ; IF NOT DELIMITER, SEE IF DIGIT
                           FALSE   ERROR   ; ERROR IF NOT A VALID DIGIT, EITHER
0648 D21106        +       JNC     ERROR
064B CDBB05                CALL    CNVBN   ; CONVERT DIGIT TO ITS BINARY VALUE
064E 1EFF                  MVI     E,0FFH  ; SET DIGIT FLAG NON-0
0650 29                    DAD     H       ; *2
0651 29                    DAD     H       ; *4
0652 29                    DAD     H       ; *8
0653 29                    DAD     H       ; *16
0654 0600                  MVI     B,0     ; CLEAR UPPER 8 BITS OF BC PAIR
0656 4F                    MOV     C,A     ; BINARY VALUE OF CHARACTER INTO C
0657 09                    DAD     B       ; ADD THIS VALUE TO PARTIAL RESULT
0658 C32C06                JMP     GHX05   ; GET NEXT CHARACTER
                   ;
                   ;
                   ;*****************************************************************
                   ;
                   ;
                   ; FUNCTION: GETNM
                   ; INPUTS: C - COUNT OF NUMBERS TO FIND IN INPUT STREAM
                   ; OUTPUTS: TOP OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON TOP
                   ;                         OF STACK)
                   ; CALLS: GETHX,HILO,ERROR
                   ; DESTROYS: A,B,C,D,E,H,L,F/F'S
```

```
 LOC  OBJ           SOURCE STATEMENT

                    ; DESCRIPTION: GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1
                    ;              AND 3, INCLUSIVE,  IN THE INPUT
                    ;              STREAM AND RETURNS THEIR VALUES ON THE STACK.  IF 2
                    ;              OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
                    ;              LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
                    ;              SECOND NUMBERS WILL BE SET EQUAL.  THE LAST NUMBER
                    ;              REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
                    ;              OR AN ERROR INDICATION WILL RESULT.
                    ;
                    GETNM:
 065B 2E03                  MVI     L,3     ; PUT MAXIMUM ARGUMENT COUNT INTO L
 065D 79                    MOV     A,C     ; GET THE ACTUAL ARGUMENT COUNT
 065E E603                  ANI     3       ; FORCE TO MAXIMUM OF 3
 0660 C8                    RZ              ; IF 0, DON'T BOTHER TO DO ANYTHIING
 0661 67                    MOV     H,A     ; ELSE, PUT ACTUAL COUNT INTO H
                    GNM05:
 0662 CD2606                CALL    GETHX   ; GET A NUMBER FROM INPUT STREAM
                            FALSE   ERROR   ; ERROR IF NOT THERE - TOO FEW NUMBERS
 0665 D21106                JNC     ERROR
 0668 C5                    PUSH    B       ; ELSE, SAVE NUMBER ON STACK
 0669 2D                    DCR     L       ; DECREMENT MAXIMUM ARGUMENT COUNT
 066A 25                    DCR     H       ; DECREMENT ACTUAL ARGUMENT COUNT
 066B CA7706                JZ      GNM10   ; BRANCH IF NO MORE NUMBERS WANTED
 066E 7A                    MOV     A,D     ; ELSE, GET NUMBER TERMINATOR TO A
 066F FE0D                  CPI     CR      ; SEE IF CARRIAGE RETURN
 0671 CA1106                JZ      ERROR   ; ERROR IF SO - TOO FEW NUMBERS
 0674 C36206                JMP     GNM05   ; ELSE, PROCESS NEXT NUMBER
                    GNM10:
 0677 7A                    MOV     A,D     ; WHEN COUNT 0, CHECK LAST TERMINATOR
 0678 FE0D                  CPI     CR
 067A C21106                JNZ     ERROR   ; ERROR IF NOT CARRIAGE RETURN
 067D 01FFFF                LXI     B,0FFFFH        ; HL GETS LARGEST NUMBER
 0680 7D                    MOV     A,L     ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT
 0681 B7                    ORA     A       ; CHECK FOR 0
 0682 CA8A06                JZ      GNM20   ; IF YES, 3 NUMBERS WERE INPUT
                    GNM15:
 0685 C5                    PUSH    B       ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFFH
 0686 2D                    DCR     L
 0687 C28506                JNZ     GNM15
                    GNM20:
 068A C1                    POP     B       ; GET THE 3 ARGUMENTS OUT
 068B D1                    POP     D
 068C E1                    POP     H
 068D CDA006                CALL    HILO    ; SEE IF FIRST >= SECOND
                            FALSE   GNM25   ; NO - BRANCH
 0690 D29506                JNC     GNM25
 0693 54                    MOV     D,H
 0694 5D                    MOV     E,L     ; YES - MAKE SECOND EQUAL TO THE FIRST
                    GNM25:
 0695 E3                    XTHL            ; PUT FIRST ON STACK - GET RETURN ADDR
 0696 D5                    PUSH    D       ; PUT SECOND ON STACK
 0697 C5                    PUSH    B       ; PUT THIRD ON STACK
 0698 E5                    PUSH    H       ; PUT RETURN ADDRESS ON STACK
                    GNM30:
 0699 3D                    DCR     A       ; DECREMENT RESIDUAL COUNT
 069A F8                    RM              ; IF NEGATIVE, PROPER RESULTS ON STACK
 069B E1                    POP     H       ; ELSE, GET RETURN ADDR
 069C E3                    XTHL            ; REPLACE TOP RESULT WITH RETURN ADDR
 069D C39906                JMP     GNM30   ; TRY AGAIN
                    ;
                    ;
                    ;*****************************************************************
                    ;
                    ;
                    ; FUNCTION: HILO
                    ; INPUTS: DE - 16 BIT INTEGER
                    ;         HL - 16 BIT INTEGER
                    ; OUTPUTS: CARRY - 0 IF HL<DE
                    ;                - 1 IF HL>=DE
                    ; CALLS: NOTHING
                    ; DESTROYS: F/F'S
                    ; DESCRIPTION: HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE.  THE
                    ;              INTEGERS ARE TREATED AS UNSIGNED NUMBERS.  THE CARRY
                    ;              BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.
                    ;
                    HILO:
 06A0 C5                    PUSH    B       ; SAVE BC
 06A1 47                    MOV     B,A     ; SAVE A IN B REGISTER
 06A2 E5                    PUSH    H       ; SAVE HL PAIR
 06A3 7A                    MOV     A,D     ; CHECK FOR DE = 0000H
 06A4 B3                    ORA     E
 06A5 CAC106                JZ      HIL05   ; WE'RE AUTOMATICALLY DONE IF IT IS
 06A8 23                    INX     H       ; INCREMENT HL BY 1
 06A9 7C                    MOV     A,H     ; WANT TO TEST FOR 0 RESULT AFTER
 06AA B5                    ORA     L       ; /INCREMENTING
 06AB CAC106                JZ      HIL05   ; IF SO, HL MUST HAVE CONTAINED 0FFFFH
 06AE E1                    POP     H       ; IF NOT, RESTORE ORIGINAL HL
 06AF D5                    PUSH    D       ; SAVE DE
 06B0 3EFF                  MVI     A,0FFH  ; WANT TO TAKE 2'S COMPLEMENT OF DE CONTENTS
 06B2 AA                    XRA     D
 06B3 57                    MOV     D,A
 06B4 3EFF                  MVI     A,0FFH
 06B6 AB                    XRA     E
 06B7 5F                    MOV     E,A
 06B8 13                    INX     D       ; 2'S COMPLEMENT OF DE TO DE
 06B9 7D                    MOV     A,L
 06BA 83                    ADD     E       ; ADD HL AND DE
 06BB 7C                    MOV     A,H
 06BC 8A                    ADC     D       ; THIS OPERATION SETS CARRY PROPERLY
 06BD D1                    POP     D       ; RESTORE ORIGINAL DE CONTENTS
```

```
 LOC  OBJ            SEQ        SOURCE STATEMENT

06BE 78                         MOV     A,B      ; RESTORE ORIGINAL CONTENTS OF A
06BF C1                         POP     B        ; RESTORE ORIGINAL CONTENTS OF BC
06C0 C9                         RET              ; RETURN WITH CARRY SET AS REQUIRED
                          HIL05:
06C1 E1                         POP     H        ; IF HL CONTAINS 0FFFFH, THEN CARRY CAN
06C2 78                         MOV     A,B      ;  /ONLY BE SET TO 1
06C3 C1                         POP     B        ; RESTORE ORIGINAL CONTENTS OF REGISTERS
06C4 C33207                     JMP     SRET     ; SET CARRY AND RETURN
                          ;
                          ;
                          ;*******************************************************************
                          ;
                          ;
                          ; FUNCTION: NMOUT
                          ; INPUTS: A - 8 BIT INTEGER
                          ; OUTPUTS: NONE
                          ; CALLS: ECHO,PRVAL
                          ; DESTROYS: A,B,C,F/F'S
                          ; DESCRIPTION: NNMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER IN THE
                          ;              A REGISTER INTO 2 ASCII CHARACTERS.  THE ASCII CHARACTERS
                          ;              ARE THE ONES REPRESENTING THE 8 BITS.  THESE TWO
                          ;              CHARACTERS ARE SENT TO THE CONSOLE AT THE CURRENT PRINT
                          ;              POSITION OF THE CONSOLE.
                          ;
                          NMOUT:
06C7 E5                         PUSH    H        ; SAVE HL - DESTROYED BY PRVAL
06C8 F5                         PUSH    PSW      ; SAVE ARGUMENT
06C9 0F                         RRC
06CA 0F                         RRC
06CB 0F                         RRC
06CC 0F                         RRC              ; GET UPPER 4 BITS TO LOW 4 BIT POSITIONS
06CD E60F                       ANI     HCHAR    ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
06CF 4F                         MOV     C,A
06D0 CDE206                     CALL    PRVAL    ; CONVERT LOWER 4 BITS TO ASCII
06D3 CDF805                     CALL    ECHO     ; SEND TO TERMINAL
06D6 F1                         POP     PSW      ; GET BACK ARGUMENT
06D7 E60F                       ANI     HCHAR    ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
06D9 4F                         MOV     C,A
06DA CDE206                     CALL    PRVAL
06DD CDF805                     CALL    ECHO
06E0 E1                         POP     H        ; RESTORE SAVED VALUE OF HL
06E1 C9                         RET
                          ;
                          ;
                          ;*******************************************************************
                          ;
                          ;
                          ; FUNCTION;  PRVAL
                          ; INPUTS: C - INTEGER, RANGE 0 TO F
                          ; OUTPUTS: C - ASCII CHARACTER
                          ; CALLS: NOTHING
                          ; DESTROYS: B,C,H,L,F/F'S
                          ; DESCRIPTION: PRVAL CONVERTS A NUMBER IN THE RANGE 0 TO F HEX TO
                          ;              THE CORRESPONDING ASCII CHARACTER, 0-9,A-F.  PRVAL
                          ;              DOES NOT CHECK THE VALIDITY OF ITS INPUT ARGUMENT.
                          ;
                          PRVAL:
06E2 21B407                     LXI     H,DIGTB  ; ADDRESS OF TABLE
06E5 0600                       MVI     B,0      ; CLEAR HIGH ORDER BITS OF BC
06E7 09                         DAD     B        ; ADD DIGIT VALUE TO HL ADDRESS
06E8 4E                         MOV     C,M      ; FETCH CHARACTER FROM MEMORY
06E9 C9                         RET
                          ;

                          ;*******************************************************************
                          ;
                          ;
                          ; FUNCTION: REGDS
                          ; INPUTS: NONE
                          ; OUTPUTS: NONE
                          ; CALLS: ECHO,NMOUT,ERROR,CROUT
                          ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                          ; DESCRIPTION: REGDS DISPLAYS THE CONTENTS OF THE REGISTER SAVE
                          ;              LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE.  THE
                          ;              DISPLAY IIS DRIVEN FROM A TABLE, RTAB, WHICH CONTAINS
                          ;              THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS,
                          ;              AND LENGTH (8 OR 16 BITS).
                          ;
                          REGDS:
06EA 21C407                     LXI     H,RTAB   ; LOAD HL WITH ADDRESS OF START OF TABLE
                          REG05:
06ED 4E                         MOV     C,M      ; GET PRINT SYMBOL OF REGISTER
06EE 79                         MOV     A,C
06EF B7                         ORA     A        ; TEST FOR 0 - END OF TABLE
06F0 C2F706                     JNZ     REG10    ; IF NOT END, BRANCH
06F3 CDEB05                     CALL    CROUT    ; ELSE, CARRIAGE RETURN/LINE FEED TO END
06F6 C9                         RET              ; /DISPLAY
                          REG10:
06F7 CDF805                     CALL    ECHO     ; ECHO CHARACTER
06FA 0E3D                       MVI     C,'='
06FC CDF805                     CALL    ECHO     ; OUTPUT EQUALS SIGN, I.E. A=
06FF 23                         INX     H        ; POINT TO START OF SAVE LOCATION ADDRESS
0700 5E                         MOV     E,M      ; GET LSP OF SAVE LOCATION ADDRESS TO E
0701 1620                       MVI     D,RAMST SHR 8    ; PUT MSP OF SAVE LOC ADDRESS INTO D
0703 23                         INX     H        ; POINT TO LENGTH FLAG
0704 1A                         LDAX    D        ; GET CONTENTS OF SAVE ADDRESS
0705 CDC706                     CALL    NMOUT    ; DISPLAY ON CONSOLE
0708 7E                         MOV     A,M      ; GET LENGTH FLAG
0709 B7                         ORA     A        ; SET SIGN F/F
070A CA1207                     JZ      REG15    ; IF 0, REGISTER IS 8 BITS
```

```
LOC  OBJ         SEQ        SOURCE STATEMENT

070D 1B                   DCX     D         ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
070E 1A                   LDAX    D         ; GET LOWER 8 BITS
070F CDC706               CALL    NMOUT     ; DISPLAY THEM
                  REG15:
0712 0E20                 MVI     C,' '
0714 CDF805               CALL    ECHO
0717 23                   INX     H         ; POINT TO START OF NEXT TABLE ENTRY
0718 C3ED06               JMP     REG05     ; DO NEXT REGISTER
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: RGADR
                  ; INPUTS: C - CHARACTER DENOTING REGISTER
                  ; OUTPUTS: BC - ADDRESS OF ENTRY IN RTAB CORRESPONDING TO REGISTER
                  ; CALLS: ERROR
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION: RGADR TAKES A SINGLE CHARACTER AS INPUT.  THIS CHARACTER
                  ;              DENOTES A REGISTER.  RGADR SEARCHES THE TABLE RTAB
                  ;              FOR A MATCH ON THE INPUT ARGUMENT.  IF ONE OCCURS,
                  ;              RGADR RETURNS THE ADDRESS OF THE ADDRESS OF THE
                  ;              SAVE LOCATION CORRESPONDING TO THE REGISTER.  THIS
                  ;              ADDRESS POINTS INTO RTAB.  IF NO MATCH OCCURS, THEN
                  ;              THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS
                  ;              PASSED TO THE ERROR ROUTINE.
                  ;
                  RGADR:
071B 21C407               LXI     H,RTAB    ; HL GETS ADDRESS OF TABLE START
071E 110300               LXI     D,RTABS   ; DE GET SIZE OF A TABLE ENTRY
                  RGA05:
0721 7E                   MOV     A,M       ; GET REGISTER IDENTIFIER
0722 B7                   ORA     A         ; CHECK FOR TABLE END (IDENTIFIER IS 0)
0723 CA1106               JZ      ERROR     ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL
0726 B9                   CMP     C         ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT
0727 CA2E07               JZ      RGA10     ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR
072A 19                   DAD     D         ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY
072B C32107               JMP     RGA05     ; TRY AGAIN
                  RGA10:
072E 23                   INX     H         ; IF A MATCH, INCREMENT TABLE POINTER TO
072F 44                   MOV     B,H       ; /SAVE LOCATION ADDRESS
0730 4D                   MOV     C,L       ; RETURN THIS VALUE
0731 C9                   RET
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: SRET
                  ; INPUTS: NONE
                  ; OUTPUTS: CARRY = 1
                  ; CALLS: NOTHING
                  ; DESTROYS: CARRY
                  ; DESCRIPTION: SRET IS JUMPED TO BY ROUTINES WISHING TO RETURN SUCCESS.
                  ;              SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE
                  ;              CALLER OF THE ROUTINE INVOKING SRET.
                  ;
                  SRET:
0732 37                   STC               ; SET CARRY TRUE
0733 C9                   RET               ; RETURN APPROPRIATELY
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: STHF0
                  ; INPUTS: DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
                  ; OUTPUTS: NONE
                  ; CALLS: STHLF
                  ; DESTROYS: A,B,C,H,L,F/F'S
                  ; DESCRIPTION: STHF0 CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF
                  ;              IT IS SET TO LOWER.  IF SO, STHF0 STORES A 0 TO
                  ;              PAD OUT THE LOWER HALF OF THE ADDRESSED BYTE;
                  ;              OTHERWISE, THE ROUTINE TAKES NO ACTION.
                  ;
                  STHF0:
0734 3AFD20               LDA     TEMP      ; GET HALF BYTE FLAG
0737 B7                   ORA     A         ; SET F/F'S
0738 C0                   RNZ               ; IF SET TO UPPER, DON'T DO ANYTHING
0739 0E00                 MVI     C,0       ; ELSE, WANT TO STORE THE VALUE 0
073B CD3F07               CALL    STHLF     ; DO IT
073E C9                   RET
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ; FUNCTION: STHLF
                  ; INPUTS: C - 4 BIT VALUE TO BE STORED IN HALF BYTE
                  ;           DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
                  ; OUTPUTS: NONE
                  ; CALLS: NOTHING
                  ; DESTROYS: A,B,C,H,L,F/F'S
                  ; DESCRIPTION: STHLF TAKES THE 4 BIT VALUE IN C AND STORES IT IN
                  ;              HALF OF THE BYTE ADDRESSED BY REGISTERS DE.  THE
                  ;              HALF BYTE USED (EITHER UPPER OR LOWER) IS DENOTED
                  ;              BY THE VALUE OF THE FLAG IN TEMP.  STHLF ASSUMES
                  ;              THAT THIS FLAG HAS BEEN PREVIOUSLY SET
                  ;              (NOMINALLY BY ICMD).
                  ;
```

```
 LOC  OBJ         SEQ        SOURCE STATEMENT

                  2334 STHLF:
073F D5           2335          PUSH    D
0740 E1           2336          POP     H       ; MOVE ADDRESS OF BYTE INTO HL
0741 79           2337          MOV     A,C     ; GET VALUE
0742 E60F         2338          ANI     0FH     ; FORCE TO 4 BIT LENGTH
0744 4F           2339          MOV     C,A     ; PUT VALUE BACK
0745 3AFD20       2340          LDA     TEMP    ; GET HALF BYTE FLAG
0748 B7           2341          ORA     A       ; CHECK FOR LOWER HALF
0749 C25207       2342          JNZ     STH05   ; BRANCH IF NOT
074C 7E           2343          MOV     A,M     ; ELSE, GET BYTE
074D E6F0         2344          ANI     0F0H    ; CLEAR LOWER 4 BITS
074F B1           2345          ORA     C       ; OR IN VALUE
0750 77           2346          MOV     M,A     ; PUT BYTE BACK
0751 C9           2347          RET
                  2348 STH05:
0752 7E           2349          MOV     A,M     ; IF UPPER HALF, GET BYTE
0753 E60F         2350          ANI     0FH     ; CLEAR UPPER 4 BITS
0755 47           2351          MOV     B,A     ; SAVE BYTE IN B
0756 79           2352          MOV     A,C     ; GET VALUE
0757 0F           2353          RRC
0758 0F           2354          RRC
0759 0F           2355          RRC
075A 0F           2356          RRC             ; ALIGN TO UPPER 4 BITS
075B B0           2357          ORA     B       ; OR IN ORIGINAL LOWER 4 BITS
075C 77           2358          MOV     M,A     ; PUT NEW CONFIGURATION BACK
075D C9           2359          RET
                  2360 ;
                  2361 ;
                  2362 ;*****************************************************************
                  2363 ;
                  2364 ;
                  2365 ; FUNCTION: VALDG
                  2366 ; INPUTS: C - ASCII CHARACTER
                  2367 ; OUTPUTS: CARRY - 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
                  2368 ;                - 0 OTHERWISE
                  2369 ; CALLS: NOTHING
                  2370 ; DESTROYS: A,F/F'S
                  2371 ; DESCRIPTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
                  2372 ;              AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
                  2373 ;              (0-9,A-F), AND FAILURE OTHERWISE.
                  2374 ;
                  2375 VALDG:
075E 79           2376          MOV     A,C
075F FE30         2377          CPI     '0'     ; TEST CHARACTER AGAINST '0'
0761 FA1C06       2378          JM      FRET    ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
0764 FE39         2379          CPI     '9'     ; ELSE, SEE IF IN RANGE '0'-'9'
0766 FA3207       2380          JM      SRET    ; CODE BETWEEN '0' AND '9'
0769 CA3207       2381          JZ      SRET    ; CODE EQUAL '9'
076C FE41         2382          CPI     'A'     ; NOT A DIGIT - TRY FOR A LETTER
076E FA1C06       2383          JM      FRET    ; NO - CODE BETWEEN '9' AND 'A'
0771 FE47         2384          CPI     'G'
0773 F21C06       2385          JP      FRET    ; NO - CODE GREATER THAN 'F'
0776 C33207       2386          JMP     SRET    ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE
                  2387 ;
                  2388 ;
                  2389 ;*****************************************************************
                  2390 ;
                  2391 ;
                  2392 ; FUNCTION: VALDL
                  2393 ; INPUTS: C - CHARACTER
                  2394 ; OUTPUTS: CARRY - 1 IF INPUT ARGUMENT VALID DELIMTER
                  2395 ;                - 0 OTHERWISE
                  2396 ; CALLS: NOTHING
                  2397 ; DESTROYS: A,F/F'S
                  2398 ; DESCRIPTION: VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID
                  2399 ;              DELIMITER CHARACTER (SPACE, COMMA, CARRIAGE RETURN) AND
                  2400 ;              FAILURE OTHERWISE.
                  2401 ;
                  2402 VALDL:
0779 79           2403          MOV     A,C
077A FE2C         2404          CPI     ','     ; CHECK FOR COMMA
077C CA3207       2405          JZ      SRET
077F FE0D         2406          CPI     CR      ; CHECK FOR CARRIAGE RETURN
0781 CA3207       2407          JZ      SRET
0784 FE20         2408          CPI     ' '     ; CHECK FOR SPACE
0786 CA3207       2409          JZ      SRET
0789 C31C06       2410          JMP     FRET    ; ERROR IF NONE OF THE ABOVE
                  2411 ;
                  2412 ;
                  2413 ;*****************************************************************
                  2414 ;
                  2415 ;
                  2416 ;                       MONITOR TABLES
                  2417 ;
                  2418 ;
                  2419 ;*****************************************************************
                  2420 ;
                  2421 ;
                  2422 SGNON:                   ; SIGNON MESSAGE
078C 0D           2423          DB      CR,LF,'SDK-85   VER 2.1',CR,LF
078D 0A
078E 53444B2D
0792 38352020
0796 20564552
079A 20322E31
079E 0D
079F 0A
0014              2424 LSGNON  EQU     $-SGNON ; LENGTH OF SIGNON MESSAGE
                  2425 ;
                  2426 CADR:                    ; TABLE OF ADDRESSES OF COMMAND ROUTINES
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

 07A0 0000          2427         DW       0       ; DUMMY
 07A2 1405          2428         DW       XCMD
 07A4 F004          2429         DW       SCMD
 07A6 D004          2430         DW       MCMD
 07A8 8604          2431         DW       ICMD
 07AA 6804          2432         DW       GCMD
 07AC 3704          2433         DW       DCMD
                    2434 ;
                    2435 CTAB:                        ; TABLE OF VALID COMMAND CHARACTERS
 07AE 44            2436         DB       'D'
 07AF 47            2437         DB       'G'
 07B0 49            2438         DB       'I'
 07B1 4D            2439         DB       'M'
 07B2 53            2440         DB       'S'
 07B3 58            2441         DB       'X'
 0006               2442 NCMDS   EQU      $-CTAB  ; NUMBER OF VALID COMMANDS
                    2443 ;
                    2444 DIGTB:                       ; TABLE OF PRINT VALUES OF HEX DIGITS
 07B4 30            2445         DB       '0'
 07B5 31            2446         DB       '1'
 07B6 32            2447         DB       '2'
 07B7 33            2448         DB       '3'
 07B8 34            2449         DB       '4'
 07B9 35            2450         DB       '5'
 07BA 36            2451         DB       '6'
 07BB 37            2452         DB       '7'
 07BC 38            2453         DB       '8'
 07BD 39            2454         DB       '9'
 07BE 41            2455         DB       'A'
 07BF 42            2456         DB       'B'
 07C0 43            2457         DB       'C'
 07C1 44            2458         DB       'D'
 07C2 45            2459         DB       'E'
 07C3 46            2460         DB       'F'
                    2461 ;
                    2462 RTAB:                        ; TABLE OF REGISTER INFORMATION
 07C4 41            2463         DB       'A'     ; REGISTER IDENTIFIER
 07C5 EE            2464         DB       ASAV AND 0FFH   ; ADDRESS OF REGISTER SAVE LOCATION
 07C6 00            2465         DB       0       ; LENGTH FLAG - 0=8 BITS, 1=16 BITS
 0003               2466 RTABS   EQU      $-RTAB  ; SIZE OF AN ENTRY IN THIS TABLE
 07C7 42            2467         DB       'B'
 07C8 EC            2468         DB       BSAV AND 0FFH
 07C9 00            2469         DB       0
 07CA 43            2470         DB       'C'
 07CB EB            2471         DB       CSAV AND 0FFH
 07CC 00            2472         DB       0
 07CD 44            2473         DB       'D'
 07CE EA            2474         DB       DSAV AND 0FFH
 07CF 00            2475         DB       0
 07D0 45            2476         DB       'E'
 07D1 E9            2477         DB       ESAV AND 0FFH
 07D2 00            2478         DB       0
 07D3 46            2479         DB       'F'
 07D4 ED            2480         DB       FSAV AND 0FFH
 07D5 00            2481         DB       0
 07D6 49            2482         DB       'I'
 07D7 F1            2483         DB       ISAV AND 0FFH
 07D8 00            2484         DB       0
 07D9 48            2485         DB       'H'
 07DA F0            2486         DB       HSAV AND 0FFH
 07DB 00            2487         DB       0
 07DC 4C            2488         DB       'L'
 07DD EF            2489         DB       LSAV AND 0FFH
 07DE 00            2490         DB       0
 07DF 4D            2491         DB       'M'
 07E0 F0            2492         DB       HSAV AND 0FFH
 07E1 01            2493         DB       1
 07E2 53            2494         DB       'S'
 07E3 F5            2495         DB       SSAV+1 AND 0FFH
 07E4 01            2496         DB       1
 07E5 50            2497         DB       'P'
 07E6 F3            2498         DB       PSAV+1 AND 0FFH
 07E7 01            2499         DB       1
 07E8 00            2500         DB       0       ; END OF TABLE MARKERS
 07E9 00            2501         DB       0
                    2502 ;
 07FA               2503         ORG      BRTAB   ; BRANCH TABLE FOR USER ACCESSIBLE ROUTINES
                    2504 ;
 07FA C3C405        2505         JMP      CO      ; TTY CONSOLE OUTPUT
 07FD C39005        2506         JMP      CI      ; TTY CONSOLE  INPUT
                    2507
                    2508 ;*******************************************************************
                    2509 ;
                    2510 ; IN THE FOLLOWING LOCATIONS, THE USER MAY PLACE JUMP INSTRUCTIONS TO
                    2511 ; ROUTINES FOR HANDLING THE FOLLOWING:-
                    2512 ;       A) RST 5,6 & 7 INSTRUCTIONS
                    2513 ;       B) HARDWIRED USER INTERRUPT (RST 6.5)
                    2514 ;       C) KEYBOARD "VECTORED INTERRUPT" KEY (RST 7.5)
                    2515 ;
 20C2               2516         ORG      USRBR   ; START OF USER BRANCH LOCATIONS
                    2517 ;
 20C2 00            2518 RSET5:  DB       0,0,0   ; JUMP TO RST 5 ROUTINE
 20C3 00
 20C4 00
 20C5 00            2519 RSET6:  DB       0,0,0   ; JUMP TO RST 6 ROUTINE
 20C6 00
 20C7 00
 20C8 00            2520 RST65:  DB       0,0,0   ; JUMP TO RST 6.5 (HARDWIRED USER INTERRUPT)
 20C9 00
 20CA 00
```

```
  LOC  OBJ          SEQ        SOURCE STATEMENT

 20CB 00            2521 RSET7:  DB      0,0,0    ; JUMP TO RST 7 ROUTINE
 20CC 00
 20CD 00
 20CE 00            2522 USINT:  DB      0,0,0    ; JUMP TO "VECTORED INTERRUPT" KEY ROUTINE
 20CF 00
 20D0 00
                    2523 ;
                    2524 ;*****************************************************************
                    2525 ;
                    2526 ; SPACE IS RESERVED HERE FOR THE MONITOR STACK
                    2527 ;
                    2528 ;*****************************************************************
                    2529 ;
 20E9               2530         ORG     MNSTK    ; START OF MONITOR STACK
                    2531 ;
                    2532 ;       SAVE LOCATIONS FOR USER REGISTERS
                    2533 ;
 20E9 00            2534 ESAV:   DB      0        ; E REGISTER
 20EA 00            2535 DSAV:   DB      0        ; D REGISTER
 20EB 00            2536 CSAV:   DB      0        ; C REGISTER
 20EC 00            2537 BSAV:   DB      0        ; B REGISTER
 20ED 00            2538 FSAV:   DB      0        ; FLAGS
 20EE 00            2539 ASAV:   DB      0        ; A REGISTER
 20EF 00            2540 LSAV:   DB      0        ; L REGISTER
 20F0 00            2541 HSAV:   DB      0        ; H REGISTER
 20F1 00            2542 ISAV:   DB      0        ; INTERRUPT MASK
                    2543 PSAV:                    ; PROGRAM COUNTER
 20F2 00            2544 PCLSV:  DB      0        ;   LOW ORDER BYTE
 20F3 00            2545 PCHSV:  DB      0        ;   HIGH ORDER BYTE
                    2546 SSAV:                    ; STACK POINTER
 20F4 00            2547 SPLSV:  DB      0        ;   LOW ORDER BYTE
 20F5 00            2548 SPHSV:  DB      0        ;   HIGH ORDER BYTE
                    2549 ;
                    2550 ;*****************************************************************
                    2551 ;
                    2552 ; MONITOR STORAGE LOCATIONS
                    2553 ;
 20F6 0000          2554 CURAD:  DW      0        ; CURRENT ADDRESS
 20F8 00            2555 CURDT:  DB      0        ; CURRENT DATA
 0004               2556 OBUFF:  DS      4        ; OUTPUT BUFFER
                    2557 TEMP:                    ; TEMPORARY LOCATION FOR TTY MONITOR
                    2558                          ; TEMPORARY LOCATION FOR SINGLE STEP ROUTINE
 20FD 00            2559 RGPTR:  DB      0        ; REGISTER POINTER
 20FE 00            2560 IBUFF:  DB      0        ; INPUT BUFFER
 20FF 00            2561 USCSR:  DB      0        ; USER SHOULD STORE IMAGE OF CSR HERE EACH TIME
                    2562                          ; /CSR IS CHANGED. OTHERWISE, SINGLE STEP
                    2563                          ; /ROUTINE WILL DESTROY CSR CONTENTS.
                    2564         END

PUBLIC SYMBOLS


EXTERNAL SYMBOLS


USER SYMBOLS

ADFLD  A 0000   ADISP  A 0090   ASAV   A 20EE   BLANK  A 0015   BLNKS  A 039A   BRCHR  A 001B   BRTAB  A 07FA
BSAV   A 20EC   CADR   A 07A0   CI     A 0590   CI05   A 0592   CI10   A 05A1   CLDBK  A 0008   CLDIS  A 01E9
CLDST  A 01F1   CLEAR  A 01D7   CMD10  A 007B   CMD15  A 0087   CMDAD  A 037C   CMDTB  A 0378   CMMND  A 0066
CNTRL  A 1900   CNVBN  A 05BB   CO     A 05C4   CO05   A 05CB   COMMA  A 0011   CR     A 000D   CROUT  A 05EB
CSAV   A 20EB   CSNIT  A 0000   CSR    A 0020   CTAB   A 07AE   CURAD  A 20F6   CURDT  A 20F8   DCM05  A 043E
DCM10  A 0449   DCM15  A 045E   DCMD   A 0437   DDISP  A 0094   DELAY  A 05F1   DIGTB  A 07B4   DISPC  A 0200
DOT    A 0001   DSAV   A 20EA   DSPLY  A 1800   DSPTB  A 0384   DTFLD  A 0001   DTMSK  A 0008   ECH05  A 0601
ECH10  A 060F   ECHO   A 05F8   EIGHT  A 0008   EMPTY  A 0080   ERMSG  A 039E   ERR    A 0215   ERROR  A 0611
ESAV   A 20E9   ESC    A 001B   EXAM   A 0092   EXM05  A 009D   EXM10  A 00B8   EXMSG  A 03A2   FALSE  + 0001
FIVE   A 0005   FRET   A 061C   FSAV   A 20ED   G10    A 00EC   GCM05  A 047D   GCM10  A 0483   GCMD   A 0468
GETCH  A 061F   GETCM  A 0408   GETHX  A 0626   GETNM  A 065B   GHX05  A 062C   GHX10  A 0645   GNM05  A 0662
GNM10  A 0677   GNM15  A 0685   GNM20  A 068A   GNM25  A 0695   GNM30  A 0699   GO     A 03FA   GOCMD  A 00CB
GTC03  A 0414   GTC05  A 0421   GTC10  A 042D   GTH05  A 0232   GTH10  A 0249   GTH20  A 0255   GTH25  A 0267
GTHEX  A 022B   HCHAR  A 000F   HIL05  A 06C1   HILO   A 06A0   HSAV   A 20F0   HXDSP  A 026C   IBTIM  A 048C
IBUFF  A 20FE   ICM05  A 0491   ICM10  A 04B9   ICM20  A 04C1   ICM25  A 04C7   ICMD   A 0486   ININT  A 028E
INSDG  A 029F   INVRT  A 00FF   ISAV   A 20F1   KBNIT  A 00CC   KMODE  A 0000   LETRA  A 000A   LETRB  A 000B
LETRC  A 000C   LETRD  A 000D   LETRE  A 000E   LETRF  A 000F   LETRH  A 0010   LETRI  A 0013   LETRL  A 0011
LETRP  A 0012   LETRR  A 0014   LETRS  A 0005   LF     A 000A   LOWER  A 0000   LSAV   A 20EF   LSGNON A 0014
MCM05  A 04D8   MCMD   A 04D0   MNSTK  A 20E9   MSGL   A 03FF   NCMDS  A 0006   NEWLN  A 000F   NMOUT  A 06C7
NMTBL  A 03B9   NODOT  A 0000   NUMC   A 0004   NUMRG  A 000D   NXTRG  A 02A8   OBTIM  A 048C   OBUFF  A 20F9
OUT05  A 02C2   OUT10  A 02C6   OUT15  A 02C9   OUT20  A 02DC   OUTPT  A 02B7   PCHSV  A 20F3   PCLSV  A 20F2
PERIO  A 0010   PRMPT  A 00FB   PRTY0  A 007F   PRVAL  A 06E2   PSAV   A 20F2   RAMST  A 2000   RDK10  A 02F3
RDKBD  A 02E7   READ   A 0040   REG05  A 06ED   REG10  A 06F7   REG15  A 0712   REGDS  A 06EA   RES10  A 003F
RETF   A 02F7   RETT   A 02FA   RGA05  A 0721   RGA10  A 072E   RGADR  A 071B   RGLOC  A 02FC   RGNAM  A 0309
RGPTB  A 03AC   RGPTR  A 20FD   RGTBL  A 03ED   RMUSE  A 0017   RSET5  A 20C2   RSET6  A 20C5   RSET7  A 20CB
RSR05  A 032D   RSR10  A 0331   RST65  A 20C8   RSTOR  A 031B   RTAB   A 07C4   RTABS  A 0003   SCM05  A 04F5
SCM10  A 0500   SCM15  A 0510   SCMD   A 04F0   SETRG  A 0344   SGNAD  A 03A6   SGNDT  A 03AA   SGNON  A 078C
SKLN   A 0018   SPHSV  A 20F5   SPLSV  A 20F4   SRET   A 0732   SSAV   A 20F4   SSTEP  A 00FD   SSTRT  A 0080
STH05  A 0752   STHF0  A 0734   STHLF  A 073F   STOPB  A 0040   STP20  A 0126   STP21  A 013B   STP22  A 0142
STP23  A 0145   STP25  A 0157   STRT   A 00C0   SUB05  A 019C   SUB10  A 01C4   SUB15  A 01CF   SUBST  A 018B
TEMP   A 20FD   TERM   A 001B   TIM4   A 1230   TIMER  A 00C5   TIMHI  A 0025   TIMLO  A 0024   TMODE  A 0040
TRUE   + 0000   TSTRT  A 00C0   UBRLN  A 000F   UNMSK  A 000E   UPDAD  A 035F   UPDDT  A 036B   UPPER  A 00FF
USCSR  A 20FF   USINT  A 20CE   USRBR  A 20C2   VALDG  A 075E   VALDL  A 0779   WAIT   A 0246   WAITS  A 0000
XCM05  A 0527   XCM10  A 0536   XCM15  A 0543   XCM18  A 054E   XCM20  A 0567   XCM25  A 057E   XCM27  A 057F
XCM30  A 0587   XCMD   A 0514   ZERO   A 0000

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ADFLD   105#   358    393    470
ADISP   106#   269    835
ASAV    1228   2464   2539#
BLANK   1164#  1171   1171   1171   1171   1172   1173   1173   1173   1175   1175   1207   1207   1207   1208   1208
        1208   1209   1209   1209   1210   1210   1210   1211   1211   1211   1212   1212   1212   1213   1213   1213
        1214   1214   1214   1215   1215   1215   1216   1217   1218   1219
BLNKS   530    534    615    1171#
BRCHR   1324#
BRTAB   1325#  2503
BSAV    184    446    1229   2468   2537#
CADR    1438   2426#
CI      1754#  1960   2506
CI05    1757#  1760
CI10    1765#  1774
CLDBK   169#   571
CLDIS   331    339    386    400    507    549#
CLDST   168    565#
CLEAR   310    357    366    392    469    527#   551
CMD10   275#   280
CMD15   277    283#
CMDAD   284    1114#
CMDTB   274    1103#  1108
CMMND   264#   552    617
CNTRL   108#   167    268    567    754    843
CNVBN   1556   1796#  2013
CO      1390   1815#  1899   1904   2505
CO05    1821#  1832
COMMA   110#   332    387    477    680
CR      1326#  1511   1520   1662   1855   1900   2058   2063   2406   2423   2423
CROUT   1471   1485   1572   1685   1854#  1925   2219
CSAV    1230   2471   2536#
CSNIT   111#   568
CSR     112#   426    435    569
CTAB    1429   2435#  2442
CURAD   475    481    491    499    501    587    1065   2554#
CURDT   320    483    589    1086   2555#
DCM05   1470#  1493
DCM10   1476#  1492
DCM15   1484   1487#
DCMD    1465#  2433
DDISP   113#   840
DELAY   1762   1767   1776   1824   1836   1871#  1875
DIGTB   2191   2444#
DISPC   350    383    585#
DOT     115#   309    321    356    391    468    484    590    674
DSAV    1231   2474   2535#
DSPLY   116#   860
DSPTB   847    1122#  1128   1134   1135   1139   1142   1144   1146   1148   1150   1152   1154   1156   1158   1160
        1162   1164
DTFLD   117#   248    323    486    532    613    1090
DTMSK   118#   856
ECH05   1896   1898#
ECH10   1902   1905#
ECHO    1422   1426   1478   1546   1636   1660   1672   1708   1856   1892#  1924   1997   2168   2173   2222   2224
        2239
EIGHT   1139#  1175   1176
EMPTY   119#   253    891
ERMSG   611    1172#
ERR     282    315    333    361    396    473    497    506    608#
ERROR   1436   1512   1521   1569   1922#  2012   2052   2059   2064   2267
ESAV    1232   2477   2534#
ESC     1327#  1894
EXAM    308#   1116
EXM05   316#   338
EXM10   326    329#
EXMSG   369    1173#
FALSE   156#   314    325    395    472    489    1483   1508   1554   1607   1638   1710   1999   2011   2051   2078
FIVE    1134#  1176
FRET    1942#  2008   2378   2383   2385   2410
FSAV    180    442    1233   2480   2538#
G10     353    364#
GCM05   1509   1518#
GCM10   1517   1522#
GCMD    1506#  2432
GETCH   1425   1544   1658   1959#  1995
GETCM   1418#  1486   1573   1603   1608   1631   1665   1680   1686   1926
GETHX   1507   1623   1637   1709   1990#  2050
GETNM   1467   1539   1588   2043#
GHX05   1994#  2022
GHX10   2000   2009#
GNM05   2049#  2060
GNM10   2056   2061#
GNM15   2069#  2072
GNM20   2068   2073#
GNM25   2079   2082#
GNM30   2087#  2092
GO      233    1385#
GOCMD   349#   1118
GTC03   1423   1424#
GTC05   1430#  1435
GTC10   1432   1437#
GTH05   651#   676
GTH10   665    669#
GTH20   654    677#
GTH25   681    683    687#
GTHEX   324    359    394    471    487    646#
HCHAR   1328#  2165   2170
HIL05   2115   2119   2137#
HILO    1481   1606   2077   2109#
HSAV    1235   2486   2492   2541#
HXDSP   671    709#   1067   1088
IBTIM   1350#  1766   1775
IBUFF   254    355    390    760    883    2560#
```

```
ICM05  1543#  1552   1566
ICM10  1561   1563#
ICM20  1555   1567#
ICM25  1549   1570#
ICMD   1537#  2431
ININT   202    751#
INSDG   657    779#
INVRT  1329#  1564
ISAV    226    403    454    994    999   1234   2483   2542#
KBNIT   120#   566
KMODE   122#   166
LETRA  1142#  1207
LETRB  1144#  1208
LETRC  1146#  1209   1218   1219
LETRD  1148#  1210
LETRE  1150#  1172   1173   1211
LETRF  1152#  1212
LETRH  1154#  1214   1216   1218
LETRI  1160#  1213
LETRL  1156#  1215   1217   1219
LETRP  1158#  1216   1217   1218   1219
LETRR  1162#  1172   1172
LETRS  1135#  1216   1217
LF     1330#  1903   2423   2423
LOWER  1331#
LSAV    175    437   1019   1236   2489   2540#
LSGNON 1387   2424#
MCM05  1592#  1609
MCMD   1586#  2430
MNSTK   124#   265   1010   1419   2530
MSGL   1388#  1393
NCMDS  1428   2442#
NEWLN  1336#  1490
NMOUT  1473   1475   1480   1634   1698   1705   2158#  2230   2236
NMTBL   962   1205#
NODOT   125#   244    249    365    368    479    533    550    592    610    614    966
NUMC    273   1108#
NUMRG   807   1241#
NXTRG   334    805#
OBTIM  1351#  1823
OBUFF   716    734   2556#
OUT05   833    838#
OUT10   837    842#
OUT15   844#   864
OUT20   853    855    858#
OUTPT   247    252    370    531    535    612    616    675    831#   967   1071   1092
PCHSV  1239   2545#
PCLSV  1240   2544#
PERIO   128#   330    352    360    385    399    505    682
PRMPT   129#   271
PRTY0  1337#  1961
PRVAL  2167   2172   2190#
PSAV    177    363    398    406    439    586   1017   1513   2498   2543#
RAMST    93#   124    137    942   1690   2227
RDK10   887    890#
RDKBD   272    351    384    652    882#   889   1037
READ    130#   755
REG05  2214#  2241
REG10  2218   2221#
REG15  2233   2237#
REGDS  1664   2212#
RES10   188    223#
RETF    686    808    906#  1039   1044
RETT    811    922#  1051
RGA05  2264#  2271
RGA10  2269   2272#
RGADR  1668   2261#
RGLOC   318    327    936#
RGNAM   317    957#
RGPTB  1047   1180#
RGPTR   806    810    937    958   1050   2559#
RGTBL   939   1227#  1241
RMUSE    98#   124    137
RSET5   197   2518#
RSET6   207   2519#
RSET7   217   2521#
RSR05  1001   1004#
RSR10  1003   1006   1009#
RST65   212   2520#
RSTOR   371    427    993#  1523
RTAB   2213   2262   2462#  2466
RTABS  1723   2263   2466#
SCM05  1626#  1643
SCM10  1629   1632#
SCM15  1639   1641#
SCMD   1622#  2429
SDK85    71
SETRG   311   1036#
SGNAD   245   1175#
SGNDT   250   1176#
SGNON  1386   2422#  2424
SKLN    101#   137
SPHSV  1237   2548#
SPLSV  1238   2547#
SRET   2007   2141   2291#  2380   2381   2386   2405   2407   2409
SSAV    183    445   1015   2495   2546#
SSTEP   382#   457   1115
SSTRT  1341#  1828
STH05  2342   2348#
STHF0  1568   1571   2309#
STHLF  1558   2314   2334#
STOPB  1342#  1833
STP20   388    402#
```

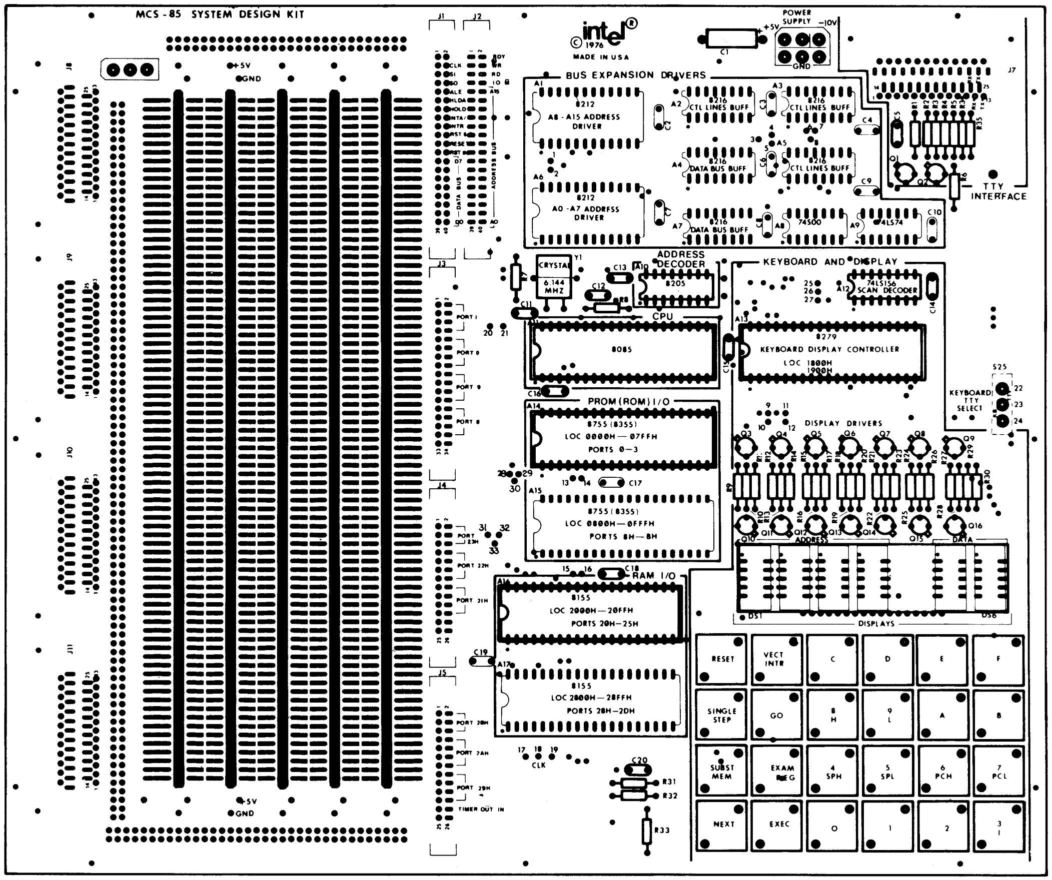
```
STP21   409    412#
STP22   411    416#
STP23   414    418#
STP25   192    429#
STRT    1343#  1819
SUB05   476#   503
SUB10   490    498#
SUB15   478    504#
SUBST   467#   1117
TEMP    405    417    452    1541   1559   1565   1674   1676   1713   1729   2310   2340   2557#
TERM    1344#  1548
TIM4    1352#  1835
TIMER   140#   419    422
TIMHI   132#   421
TIMLO   133#   423
TMODE   134#   419
TRUE    152#   337    1551
TSTRT   135#   425
UBRLN   103#   137
UNMSK   136#   227    455
UPDAD   480    591    1064#
UPDDT   322    485    593    1085#
UPPER   1345#  1540
USCSR   424    432    570    2561#
USINT   221    2522#
USRBR   137#   2516
VALDG   1553   2010   2375#
VALDL   1550   1998   2402#
WAIT    1353#  1761
WAITS   80#    139    1349
XCM05   1663   1666#
XCM10   1675#  1726
XCM15   1678   1681#
XCM18   1684   1687#
XCM20   1702   1706#
XCM25   1717   1720#
XCM27   1722#  1732
XCM30   1711   1727#
XCMD    1657#  2428
ZERO    1128#  1175

CROSS REFERENCE COMPLETE
```

```
RGNAM    317    957#
RGPTB   1047   1180#
RGPTR    806    810     937     958    1050    2559#
RGTBL    939   1227#   1241
RMUSE     98#   124     137
RSET5    197   2518#
RSET6    207   2519#
RSET7    217   2521#
RSR05   1001   1004#
RSR10   1003   1006    1009#
RST65    212   2520#
RSTOR    371    427     993#   1523
RTAB    2213   2262    2462#   2466
RTABS   1723   2263    2466#
SCM05   1626#  1643
SCM10   1629   1632#
SCM15   1639   1641#
SCMD    1622#  2429
SDK85     71
SETRG    311   1036#
SGNAD    245   1175#
SGNDT    250   1176#
SGNON   1386   2422#   2424
SKLN     101#   137
SPHSV   1237   2548#
SPLSV   1238   2547#
SRET    2007   2141    2291#   2380    2381    2386    2405    2407    2409
SSAV     183    445    1015    2495    2546#
SSTEP    382#   457    1115
SSTRT   1341#  1828
STH05   2342   2348#
STHFO   1568   1571    2309#
STHLF   1558   2314    2334#
STOPB   1342#  1833
STP20    388    402#
STP21    409    412#
STP22    411    416#
STP23    414    418#
STP25    192    429#
STRT    1343#  1819
SUB05    476#   503
SUB10    490    498#
SUB15    478    504#
SUBST    467#  1117
TEMP     405    417     452    1541    1559    1565    1674    1676    1713    1729    2310    2340    2557#
TERM    1344#  1548
TIM2    1352#  1835
TIMER    140#   419     422
TIMHI    132#   421
TIMLO    133#   423
TMODE    134#   419
TRUE     152#   337    1551
TSTRT    135#   425
UBRLN    103#   137
UNMSK    136#   227     455
UPDAD    480    591    1064#
UPDDT    322    485     593    1085#
UPPER   1345#  1540
USCSR    424    432     570    2561#
USINT    221   2522#
USRBR    137#  2516
VALDG   1553   2010    2375#
VALDL   1550   1998    2402#
WAIT    1353#  1761
WAITS     80#   139    1349
XCM05   1663   1666#
XCM10   1675#  1726
XCM15   1678   1681#
XCM18   1684   1687#
XCM20   1702   1706#
XCM25   1717   1720#
XCM27   1722#  1732
XCM30   1711   1727#
XCMD    1657#  2428
ZERO    1128#  1175

CROSS REFERENCE COMPLETE
```

APPENDIX B
DIAGRAMS



REVISIONS				
LTR	DESCRIPTION	SIGNATURE AND DATE DFT	CHK	ENGR
A	PROD REL	5/77	5/23/77	5/31/77

ITEM NO.	PART NUMBER	DESCRIPTION
PARTS LIST		

QUANTITY PER DASH NO.

NEXT ASSY	USED ON
4001117	SDK 85

SCALE: NONE				
TOL. ±	2 PLC —	3 PLC —	ANGLE —	
MATERIAL: —				
FINISH: —				

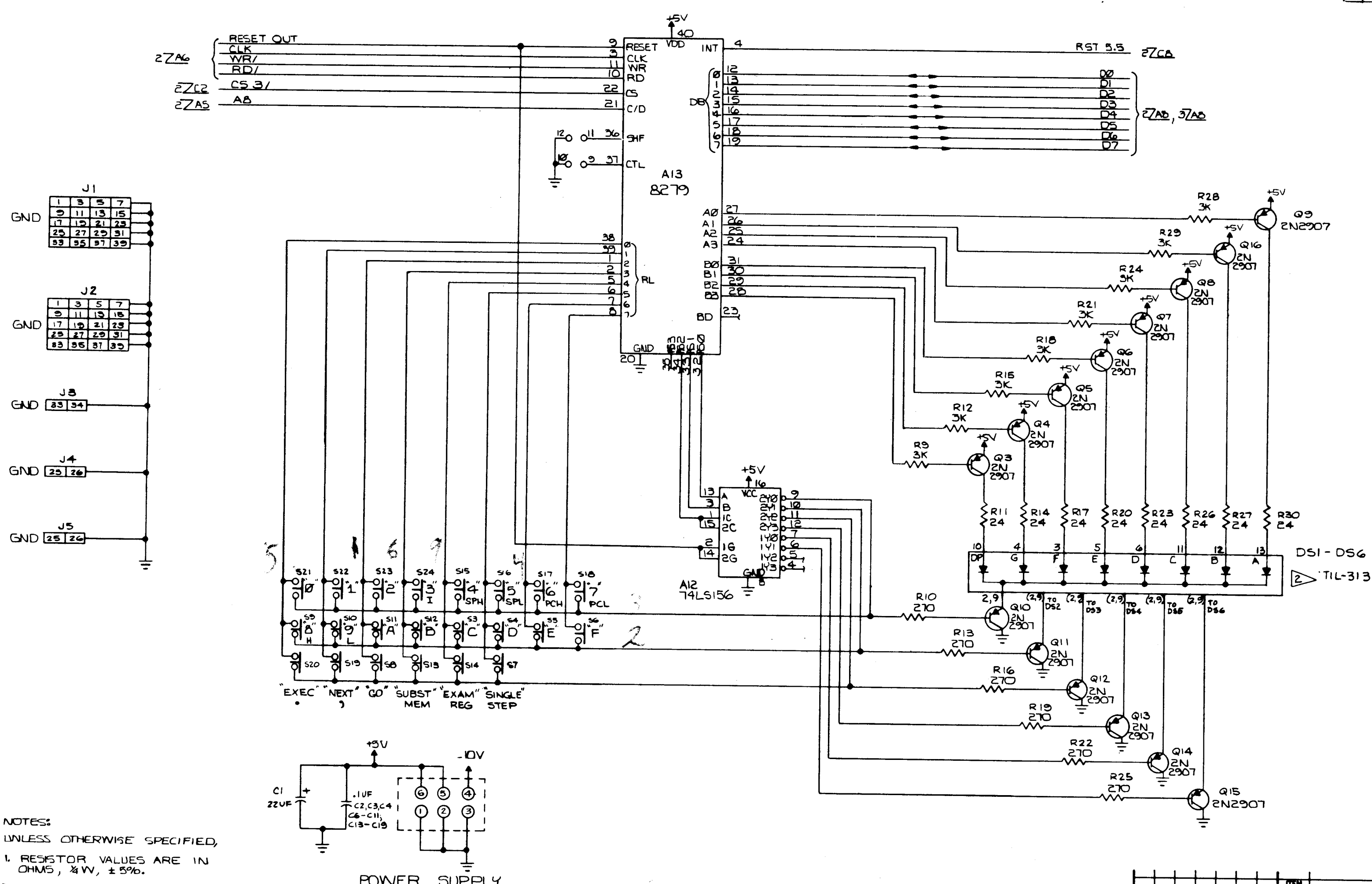
SIGNATURE	DATE
DRN BY S. McCulloy	5/77
CHK BY J. B. CHATWIN	5/23/77
DFT APPD	
ENGR APPD A. Cheung	5/31/77
AUTH BY J. Wiener	6/1/77
CODE:	SHEET 1 OF 1

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE PWA SYSTEM DESIGN KIT

SIZE	DEPT	DRAWING NO.	REV
D	416	1001119	A

LTR	DESCRIPTION	SIGNATURE AND DATE		
		DFT	CHK	ENGR
A	PROD REL	5/77	5/2/77	



NOTES:

UNLESS OTHERWISE SPECIFIED,

1. RESISTOR VALUES ARE IN OHMS, 1/4W, ±5%.

2. THIS PIN OUT ARRANGEMENT REPRESENTS 6 INDIVIDUAL 7 SEGMENT LED DISPLAYS (TIL 313). ALL ANODE CONNECTIONS OF THE CORRESPONDING SEGMENTS ARE WIRED TOGETHER. CATHODES ARE WIRED SEPARATELY TO EACH TRANSISTOR (Q10-Q15)

3. JUMPER 5 TO 3 AND 6 TO 8 WHEN BUS EXPANSION CIRCUITS ARE NOT INSTALLED.

SPARE GATE COUNT		
TYPE	REF DES	QTY

REF DESIGNATIONS	
LAST USED	NOT USED
C20	
A17	
R35	
Q16	
DS6	
33	
S25	

NEXT ASSY	USED ON
4001117	SDK 85

SCALE: NONE

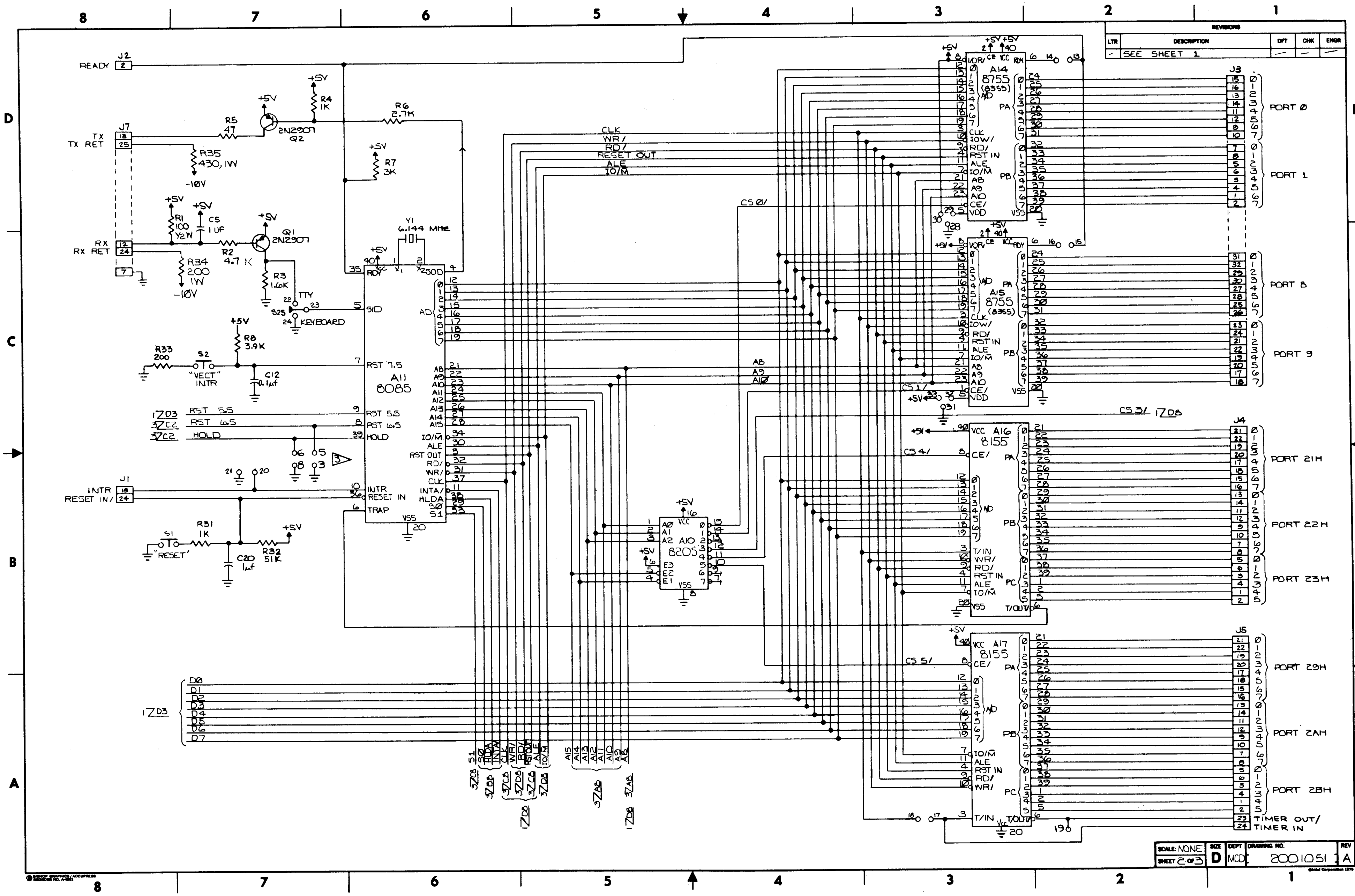
DRN BY S.McAuley 8/76

TITLE: SCHEMATIC SYSTEM DESIGN KIT

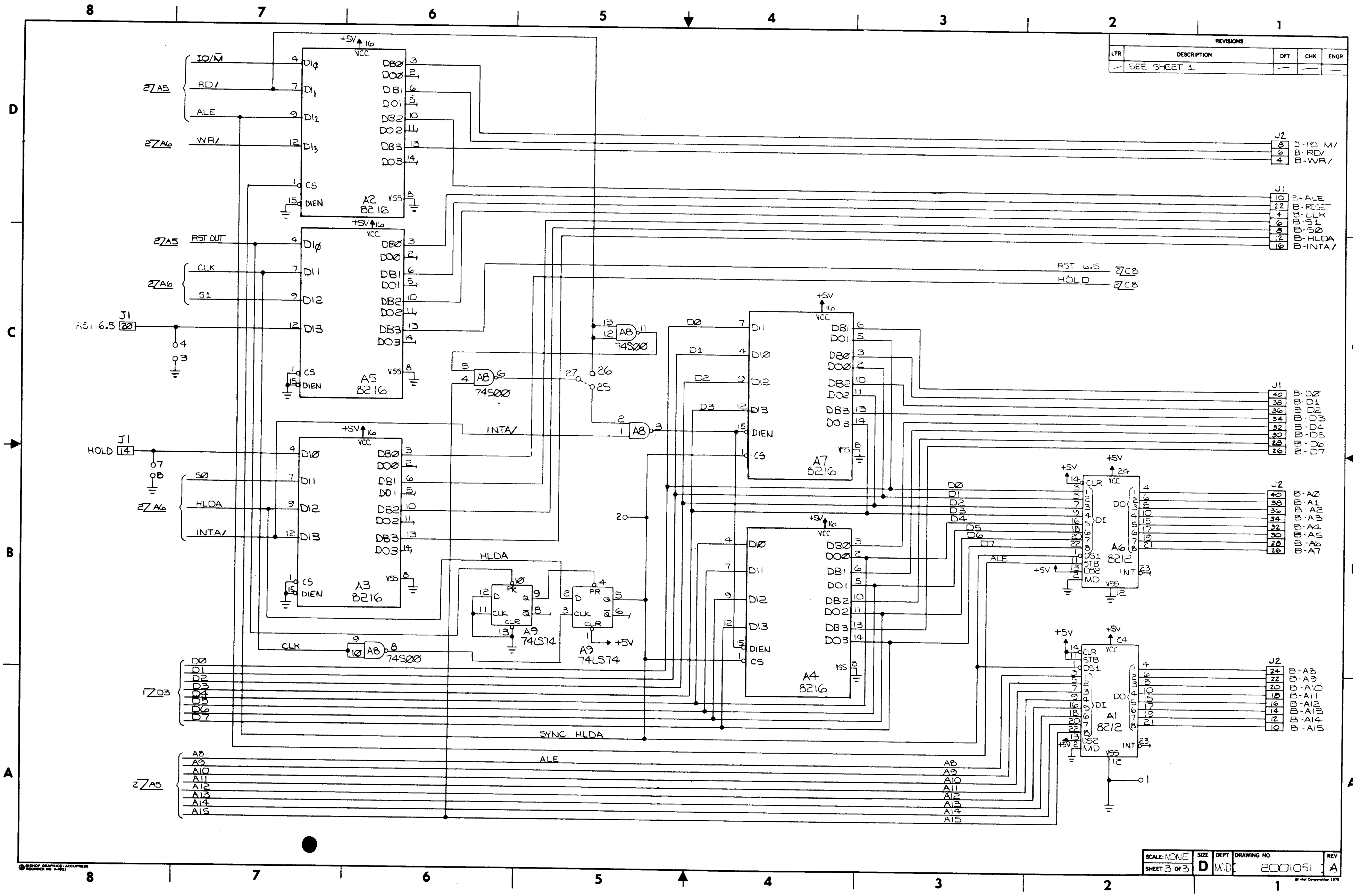
intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

SIZE D | DEPT MCD | DRAWING NO. 2001051 | REV A

SHEET 1 OF 3



REVISIONS
LTR DESCRIPTION DFT CHK ENGR
SEE SHEET 1
READY
J2
TX
TX RET
J7
RX
RX RET
R5 47
2N2907 Q2
R4 1K
R6 2.7K
R7 3K
R35 430, 1W
-10V
R1 100 1/2W
C5 1 UF
R2 4.7 K
Q1 2N2907
R34 200 1W
R3 1.6K
TTY
KEYBOARD
Y1 6.144 MHz
R8 3.9K
R33 200
S2 "VECT" INTR
C12 0.1µf
A11 8085
RDY
SID
RST 7.5
RST 5.5
RST 6.5
HOLD
INTR
RESET IN
TRAP
SOD
AD
IO/M
ALE
RST OUT
RD/
WR/
CLK
INTA/
HLDA
S0
S1
VSS
1ZD3 RST 5.5
3ZC2 RST 6.5
3ZC2 HOLD
J1
INTR
RESET IN/
S1 "RESET"
R31 1K
C20 1µf
R32 51K
+5V
CLK
WR/
RD/
RESET OUT
ALE
IO/M
CS0/
A8
A9
A10
A10 8205
A0 A1 A2 E3 E2 E1
VCC VSS
A14 8755 (8355)
A15 8755 (8355)
IOR/
CE
VCC
RDY
PA
PB
IOW/
RST IN
CE/
VDD
VSS
CS1/
CS3/ 1ZD8
CS4/
CS5/
A16 8155
A17 8155
PC
T/IN
T/OUT
J3
PORT 0
PORT 1
PORT 8
PORT 9
J4
PORT 21H
PORT 22H
PORT 23H
J5
PORT 29H
PORT 2AH
PORT 2BH
TIMER OUT/
TIMER IN
D0 D1 D2 D3 D4 D5 D6 D7
1ZD3
S1 S0 INTA/ CLK WR/ RD/ RST OUT ALE IO/M
A15 A14 A13 A12 A11 A10 A9 A8
3ZC8 3ZD8 3ZC8 3ZD8 3ZC8 3ZD8
1ZD8
3ZA8
1ZD8
SCALE: NONE
SIZE D
DEPT MCD
DRAWING NO. 2001051
REV A
SHEET 2 OF 3


REVISIONS
LTR DESCRIPTION DFT CHK ENGR
SEE SHEET 1
J2 B-IO/M B-RD/ B-WR/
J1 B-ALE B-RESET B-CLK B-S1 B-SØ B-HLDA B-INTA/
RST 6.5 ZC8
HOLD ZC8
J1 B-DØ B-D1 B-D2 B-D3 B-D4 B-D5 B-D6 B-D7
J2 B-AØ B-A1 B-A2 B-A3 B-A4 B-A5 B-A6 B-A7
J2 B-A8 B-A9 B-A10 B-A11 B-A12 B-A13 B-A14 B-A15
A2 8216
A5 8216
A3 8216
A7 8216
A4 8216
A6 8212
A1 8212
A8 74S00
A9 74LS74
IO/M RD/ ALE WR/ RST OUT CLK S1 SØ HLDA INTA/
RST 6.5
HOLD
SYNC HLDA
SCALE: NONE
SHEET 3 OF 3
SIZE D
DEPT MCD
DRAWING NO. 2001051
REV A
©Intel Corporation 1975

intel®

Manual Order Number 9800451B

REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

____

____

____

____

____

____

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

____

____

____

____

____

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

____

____

____

____

____

____

4. Did you have any difficulty understanding descriptions or wording? Where?

____

____

____

____

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. ____________

NAME ______________________________ DATE ______________

TITLE ____

COMPANY NAME/DEPARTMENT ______________________________

ADDRESS ____

CITY ____________________ STATE ____________ ZIP CODE ____________

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S 6-2000
3065 Bowers Avenue
Santa Clara, CA 95051

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A.