SDK-85
System Design Kit
User’s Manual

Manual Order Number 9800451A

Copyright © 1977 Intel Corporation

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051
The information in this manual is subject to change without notice. Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this manual. Intel Corporation makes no commitment to update nor to keep current the information contained in this manual.

No part of this manual may be copied or reproduced in any form or by any means without the prior written consent of Intel Corporation.

The following are trademarks of Intel Corporation and may be used only to describe Intel Products.

ICE-30  MCS
ICE-80  MEGACHASSIS
INSITE  MICROMAP
INTEL  MULTIBUS
INTELLEC  PROMPT
LIBRARY MANAGER  UPI
CONTENTS

CHAPTER 1
DESCRIPTION ........................................ 1-1

CHAPTER 2
HOW TO ASSEMBLE THE KIT
GENERAL ........................................ 2-1
GETTING ORGANIZED ................................ 2-1
SELECTING TOOLS AND MATERIALS ............. 2-2
UNPACKING AND SORTING PARTS ................. 2-3
A REVIEW OF BASIC ASSEMBLY AND
SOLDERING TECHNIQUES .................................. 2-6
ASSEMBLY PROCEDURE ................................ 2-7

CHAPTER 3
FINAL ASSEMBLY AND CHECKOUT
GENERAL ........................................ 3-1
STRAPPING OPTIONS ................................ 3-1
POWER SUPPLY WIRING ................................ 3-1
INSTALLING LARGE IC DEVICES .................... 3-4
STARTING THE FIRST TIME ......................... 3-6
WHAT IF IT DOESN'T? ................................ 3-7
CONNECTING A TELETYPewriter ................... 3-8

CHAPTER 4
OPERATING INSTRUCTIONS
WHAT IT DOES ........................................ 4-1
THE BUTTONS AND DISPLAYS ....................... 4-1
Reset ........................................ 4-2
Substitute Memory ................................ 4-2
Examine Registers ................................ 4-4
Go ........................................ 4-6
Single Step ........................................ 4-8
Vector Interrupt .................................... 4-9
Program Debugging —
  The Use of Breakpoints ................................ 4-9
Error Conditions — Illegal Key ................. 4-9
Memory Substitution Errors .................... 4-9
TELETYPewriter OPERATION ....................... 4-9
Console Commands ................................ 4-9
Use of the Monitor for Programming
  and Checkout ................................ 4-10
Command Structure ................................ 4-10
Display Memory Command, D .................... 4-10
Program Execute Command, G ................... 4-10
Insert Instructions into RAM, I ................ 4-11
Move Memory Command, M ......................... 4-11
Substitute Memory Command, S ................ 4-12
Examine/Modify CPU Registers
  Command, X .................................... 4-12
Programming Debugging —
  Breakpoint Facility ................................ 4-13
Error Conditions — Invalid Characters .......... 4-13
Address Value Errors ................................ 4-13

CHAPTER 5
THE HARDWARE
OVERVIEW ........................................ 5-1
SYSTEM COMPONENTS ................................ 5-1
  The 8085 CPU & The System Buses .......... 5-1
  The 8155 ................................ 5-1
  The 8355 & 8755 ................................ 5-2
  The 8279 ................................ 5-2
  The 8205 ................................ 5-2
SDK-85 MEMORY ADDRESSING .................... 5-3
INPUT/OUTPUT PORT AND
  PERIPHERAL DEVICE ADDRESSING ............ 5-5
Accessing the 8279 Keyboard/Display
  Controller .................................... 5-5
PROCESSOR INTERRUPT ALLOCATION ............. 5-7
THE SERIAL DATA INTERFACE ...................... 5-7
CONVERTER CIRCUIT FOR
  RS232C SERIAL PORT .......................... 5-8
ADDITIONAL INTERFACES ......................... 5-8

CHAPTER 6
THE SOFTWARE
THE SDK-85 MONITOR ................................ 6-1
PROGRAMMING HINTS ................................ 6-1
  Stack Pointer ................................ 6-1
  RAM-I/O Command Status Register (CSR) .... 6-1
  Access to Monitor Routines .................... 6-1
PROGRAMMING EXAMPLES ......................... 6-1

APPENDIX A
MONITOR LISTING

APPENDIX B
DIAGRAMS
Figure 1-1. SDK-85 System Design Kit
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 8085 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: 8085
Instruction Cycle: 1.3 microsecond
T<sub>cy</sub>: 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 OFFF 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).
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
- MCS-85 User’s Manual
- 8080/8085 Assembly Language Programming Manual
- Intellec® MDS Brochure
- ICE-85 Data Sheet
- PL/M-80 Data Sheet
- 8085/8080 Assembly Language Reference Card

Physical Characteristics

Width: 12.0 in.
Height: 10 in.
Depth 0.50 in.
Weight: approx. 12 oz.

Electrical Characteristics (DC Power Required)

$V_{CC}: +5V \pm 5\%$ 1.3A
$V_{TTY}: -10V \pm 10\%$ 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
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.
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.
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)
- **1** 47 Ohm (yellow-violet-black)
- **1** 200 Ohm (red-black-brown)
- **6** 270 Ohm (red-violet-brown)
- **2** 1k (1,000) Ohm (brown-black-red)
- **1** 1.6k Ohm (brown-blue-red)
- **1** 2.7k Ohm (red-violet-red)
- **9** 3k Ohm (orange-black-red)
- **1** 3.9k Ohm (orange-white-red)
- **1** 4.7k Ohm (yellow-violet-red)
- **1** 51k Ohm (green-brown-orange)

**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:

<table>
<thead>
<tr>
<th>Color</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>0</td>
</tr>
<tr>
<td>Brown</td>
<td>1</td>
</tr>
<tr>
<td>Red</td>
<td>2</td>
</tr>
<tr>
<td>Orange</td>
<td>3</td>
</tr>
<tr>
<td>Yellow</td>
<td>4</td>
</tr>
<tr>
<td>Green</td>
<td>5</td>
</tr>
<tr>
<td>Blue</td>
<td>6</td>
</tr>
<tr>
<td>Violet</td>
<td>7</td>
</tr>
<tr>
<td>Gray</td>
<td>8</td>
</tr>
<tr>
<td>White</td>
<td>9</td>
</tr>
</tbody>
</table>

The fourth band indicates percentage tolerance of the resistor value.

- **Gold** = 5%; **Silver** = 10% tolerance

**Resistors, 1 Watt**

- **1** 200 Ohm (red-black-brown)
- **1** 430 Ohm (yellow-orange-brown)

**Capacitor, tantalum**

- **1** 22 μf, 15V

**Capacitor, mono**

- **2** 1 μf, 25V
Capacitor, ceramic

- 7 0.1 \( \mu \text{f} \) 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

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 8085 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
- 1 74LS156 scan decoder

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.
- 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.

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 OPTIONS

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. 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.

Table 3-4 lists the strapping connections that may be used when the optional bus expansion driver function is implemented. 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>
<thead>
<tr>
<th>Device Location</th>
<th>8355</th>
<th>8755</th>
<th>8755A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Figure 3-1</td>
<td>Figure 3-2a</td>
<td>Figure 3-2b</td>
</tr>
<tr>
<td>A14</td>
<td>No Straps</td>
<td>Strap</td>
<td>Strap</td>
</tr>
<tr>
<td></td>
<td>Required</td>
<td>28-29</td>
<td>29-30</td>
</tr>
<tr>
<td>A15</td>
<td>Strap</td>
<td>Strap</td>
<td>Strap</td>
</tr>
<tr>
<td></td>
<td>31-32</td>
<td>32-33</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 3-2
TELETYPewriter-KEYBOARD STRAPPING

<table>
<thead>
<tr>
<th>TELETYPewriter</th>
<th>KEYBOARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3-3</td>
<td>Figure 3-4</td>
</tr>
<tr>
<td>Strap 22-23</td>
<td>Strap 23-24</td>
</tr>
</tbody>
</table>

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 3-3 Teletypewriter Strapping Option

Figure 3-4 Keyboard-Display Strapping Option

Figure 3-5 Disabling Unused Keyboard Controller Functions

Figure 3-6 Power Supply Connections
hole pattern on the board; if this connector is used, parallel two pins on the +5V bus and three pins on the GND bus. If you are going to use a teletype-writer, 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>
<thead>
<tr>
<th>TABLE 3-4</th>
<th>OPTIONAL BUS EXPANSION STRAPPING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FUNCTION</strong></td>
<td><strong>BASIC KIT WITHOUT EXPANSION MEMORY</strong></td>
</tr>
<tr>
<td></td>
<td>(Figure 3-7)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RST 6.5</td>
<td>Strap 3-5</td>
</tr>
<tr>
<td>HOLD</td>
<td>Strap 6-8</td>
</tr>
<tr>
<td>INTR</td>
<td>Strap 20-21</td>
</tr>
<tr>
<td>Memory Address Locations</td>
<td>Leave 25-26-27 unstrapped.</td>
</tr>
</tbody>
</table>

*Note: No devices may be installed in positions A13, A14, A15, A16, and A17 if this option is strapped.
Figure 3-7 Strapping Options 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
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 2-5 and 2-6.)
3-6 WHAT IF IT DOESN'T?

If there is no response to the 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) - 538-9311 when calling from outside 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 5 PM, Pacific time.

### TABLE 3-5
INTERFACE CONNECTOR J7
PIN ASSIGNMENTS

<table>
<thead>
<tr>
<th>PIN</th>
<th>MARKING</th>
<th>ASSIGNMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Ground</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Open</td>
</tr>
<tr>
<td>24</td>
<td>RX-</td>
<td>Receive Return (−)</td>
</tr>
<tr>
<td>12</td>
<td>RX+</td>
<td>Receive (+)</td>
</tr>
<tr>
<td>25</td>
<td>TX−</td>
<td>Transmit Return (−)</td>
</tr>
<tr>
<td>13</td>
<td>TX+</td>
<td>Transmit (+)</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>ASSIGNMENT</th>
<th>PIN</th>
<th>PIN</th>
<th>MARKING</th>
<th>ASSIGNMENT</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>1</td>
<td>2</td>
<td>—</td>
<td>OPEN</td>
<td>−</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>CLK</td>
<td>Buffered CLK</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>5</td>
<td>6</td>
<td>S1</td>
<td>Buffered S1</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>7</td>
<td>8</td>
<td>S0</td>
<td>Buffered S0</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>9</td>
<td>10</td>
<td>ALE</td>
<td>Buffered ALE</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>11</td>
<td>12</td>
<td>HLDA</td>
<td>Buffered HLDA</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>13</td>
<td>14</td>
<td>HOLD</td>
<td>Buffered HOLD</td>
<td>1</td>
</tr>
<tr>
<td>GND</td>
<td>15</td>
<td>16</td>
<td>INTA/</td>
<td>Buffered INTA</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>17</td>
<td>18</td>
<td>INTR</td>
<td>INTR</td>
<td>1</td>
</tr>
<tr>
<td>GND</td>
<td>19</td>
<td>20</td>
<td>RST 6.5</td>
<td>Buffered RST 6.5</td>
<td>1</td>
</tr>
<tr>
<td>GND</td>
<td>21</td>
<td>22</td>
<td>RST</td>
<td>Buffered RESET OUT</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>23</td>
<td>24</td>
<td>RST IN/</td>
<td>RESET INPUT</td>
<td>1</td>
</tr>
<tr>
<td>GND</td>
<td>25</td>
<td>26</td>
<td>D7</td>
<td>Buffered D7</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>27</td>
<td>28</td>
<td>—</td>
<td>Buffered D6</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>29</td>
<td>30</td>
<td>—</td>
<td>Buffered D5</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>31</td>
<td>32</td>
<td>—</td>
<td>Buffered D4</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>33</td>
<td>34</td>
<td>—</td>
<td>Buffered D3</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>35</td>
<td>36</td>
<td>—</td>
<td>Buffered D2</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>37</td>
<td>38</td>
<td>—</td>
<td>Buffered D1</td>
<td>I/O</td>
</tr>
<tr>
<td>GND</td>
<td>39</td>
<td>40</td>
<td>D0</td>
<td>Buffered D0</td>
<td>I/O</td>
</tr>
<tr>
<td>ASSIGNMENT</td>
<td>PIN</td>
<td>PIN</td>
<td>MARKING</td>
<td>ASSIGNMENT</td>
<td>I/O</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>------------</td>
<td>-----</td>
</tr>
<tr>
<td>GND</td>
<td>1</td>
<td>2</td>
<td>RDY</td>
<td>READY</td>
<td>1</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>WR/</td>
<td>Buffered WR</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>5</td>
<td>6</td>
<td>RD/</td>
<td>Buffered RD</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>7</td>
<td>8</td>
<td>IO/M</td>
<td>Buffered IO/M</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>9</td>
<td>10</td>
<td>A15</td>
<td>Buffered A15</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>11</td>
<td>12</td>
<td></td>
<td>Buffered A14</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>13</td>
<td>14</td>
<td></td>
<td>Buffered A13</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>15</td>
<td>16</td>
<td></td>
<td>Buffered A12</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>17</td>
<td>18</td>
<td></td>
<td>Buffered A11</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>19</td>
<td>20</td>
<td></td>
<td>Buffered A10</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>21</td>
<td>22</td>
<td>ADDRESS BUS</td>
<td>Buffered A9</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>23</td>
<td>24</td>
<td></td>
<td>Buffered A8</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>25</td>
<td>26</td>
<td></td>
<td>Buffered A7</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>27</td>
<td>28</td>
<td></td>
<td>Buffered A6</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>29</td>
<td>30</td>
<td></td>
<td>Buffered A5</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>31</td>
<td>32</td>
<td></td>
<td>Buffered A4</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>33</td>
<td>34</td>
<td></td>
<td>Buffered A3</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>35</td>
<td>36</td>
<td></td>
<td>Buffered A2</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>37</td>
<td>38</td>
<td></td>
<td>Buffered A1</td>
<td>0</td>
</tr>
<tr>
<td>GND</td>
<td>39</td>
<td>40</td>
<td>A0</td>
<td>Buffered A0</td>
<td>0</td>
</tr>
</tbody>
</table>
### TABLE 3-8
I/O PORT CONNECTOR J3 PIN ASSIGNMENTS

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

*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

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

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

<table>
<thead>
<tr>
<th>ASSIGNMENT</th>
<th>PIN</th>
<th>PIN</th>
<th>MARKING</th>
<th>ASSIGNMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2BH-4</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2BH-2</td>
<td>3</td>
<td>4</td>
<td>PORT 2BH</td>
<td></td>
</tr>
<tr>
<td>P2BH-0</td>
<td>5</td>
<td>6</td>
<td></td>
<td>P2BH-1</td>
</tr>
<tr>
<td>P2AH-6</td>
<td>7</td>
<td>8</td>
<td></td>
<td>P2AH-7</td>
</tr>
<tr>
<td>P2AH-4</td>
<td>9</td>
<td>10</td>
<td></td>
<td>P2AH-5</td>
</tr>
<tr>
<td>P2AH-2</td>
<td>11</td>
<td>12</td>
<td>PORT 2AH</td>
<td>P2AH-3</td>
</tr>
<tr>
<td>P2AH-0</td>
<td>13</td>
<td>14</td>
<td></td>
<td>P2AH-1</td>
</tr>
<tr>
<td>P29H-6</td>
<td>15</td>
<td>16</td>
<td></td>
<td>P29H-7</td>
</tr>
<tr>
<td>P29H-4</td>
<td>17</td>
<td>18</td>
<td></td>
<td>P29H-5</td>
</tr>
<tr>
<td>P29H-2</td>
<td>19</td>
<td>20</td>
<td>PORT 29H</td>
<td>P29H-3</td>
</tr>
<tr>
<td>P29H-0</td>
<td>21</td>
<td>22</td>
<td></td>
<td>P29H-1</td>
</tr>
<tr>
<td>Timer ÛOUT</td>
<td>23</td>
<td>24</td>
<td>TIMER ÛOUT/IN</td>
<td>Timer In</td>
</tr>
<tr>
<td>GROUND</td>
<td>25</td>
<td>26</td>
<td></td>
<td>GROUND</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>REFERENCE DESIGNATION</th>
<th>FUNCTION</th>
<th>NO. OF PINS</th>
<th>MFR.</th>
<th>MFR'S. PART NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Bus Expansion</td>
<td>40</td>
<td>3M</td>
<td>3432-4005</td>
</tr>
<tr>
<td>J2</td>
<td>Bus Expansion</td>
<td>40</td>
<td>3M</td>
<td>3432-4005</td>
</tr>
<tr>
<td>J3</td>
<td>I/O Ports</td>
<td>34</td>
<td>3M</td>
<td>3431-4005</td>
</tr>
<tr>
<td>J4</td>
<td>I/O Ports</td>
<td>26</td>
<td>3M</td>
<td>3429-4005</td>
</tr>
<tr>
<td>J5</td>
<td>I/O Ports and Timer</td>
<td>26</td>
<td>3M</td>
<td>3429-4005</td>
</tr>
<tr>
<td>J6</td>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J7</td>
<td>TTY Interface</td>
<td>25</td>
<td>AMP</td>
<td>206584</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td></td>
<td>AMP</td>
<td>206604</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td></td>
<td>Molex</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Power Supply</td>
<td>6</td>
<td></td>
<td>Molex</td>
</tr>
</tbody>
</table>

**TABLE 3-11**

**SUGGESTED CONNECTOR TYPES**
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 \[\text{8}\] and the \[\text{@}\] 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 \[\text{8}\] 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

<table>
<thead>
<tr>
<th>HEX</th>
<th>DECIMAL</th>
<th>BINARY</th>
<th>LED DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>1011</td>
<td>b</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>1100</td>
<td>c</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td>1101</td>
<td>d</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td>1111</td>
<td>F</td>
</tr>
</tbody>
</table>
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 [M] 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:

```
<address> [M] (<data>) [M] (<data>) . . . [EXC]
```

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 [M] 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 [EXC] or [M] key is pressed.

Pressing [M] 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 [EXC] 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 [M] 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 SUBS_MEM to list the first few Monitor locations:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000.</td>
<td></td>
</tr>
<tr>
<td>NEXT</td>
<td>0000</td>
<td>3E.</td>
</tr>
<tr>
<td>NEXT</td>
<td>0001</td>
<td>00.</td>
</tr>
<tr>
<td>NEXT</td>
<td>0002</td>
<td>32.</td>
</tr>
<tr>
<td>NEXT</td>
<td>0003</td>
<td>00.</td>
</tr>
<tr>
<td>EXEC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SUBSTITUTE MEMORY EXAMPLE 2

Using SUBS_MEM to enter a small program:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0002.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0020.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0200.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2000.</td>
<td></td>
</tr>
<tr>
<td>NEXT</td>
<td>2000</td>
<td>**.</td>
</tr>
<tr>
<td>3</td>
<td>2000</td>
<td>03.</td>
</tr>
<tr>
<td>NEXT</td>
<td>2001</td>
<td>**.</td>
</tr>
<tr>
<td>4</td>
<td>2001</td>
<td>04.</td>
</tr>
<tr>
<td>NEXT</td>
<td>2001</td>
<td>47.</td>
</tr>
<tr>
<td>0</td>
<td>2002</td>
<td>**.</td>
</tr>
<tr>
<td>C</td>
<td>2002</td>
<td>0C.</td>
</tr>
<tr>
<td>F</td>
<td>2002</td>
<td>CF.</td>
</tr>
</tbody>
</table>

**NOTE:** ** represents unpredictable values.

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

### CORRESPONDING 8085 ASSEMBLY LANGUAGE INSTRUCTIONS

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DATA</th>
<th>INSTRUCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>3E</td>
<td>MVI A, 47H</td>
</tr>
<tr>
<td>2001</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>CF</td>
<td>RST 1</td>
</tr>
</tbody>
</table>

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

The examine command allows you to display and modify the contents of the 8085 CPU registers. Pressing the 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 or key is pressed.

Pressing 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 will place the contents displayed in the data field in the register named in the address field, and will also terminate the command.

Pressing while register PCL is being displayed has the same effect as pressing.

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:

\[
\begin{array}{c|c|c|c|c|c|c}
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & I & M & M & M \\
\end{array}
\]

\text{INTERRUPT MASK}

\text{INTERRUPT ENABLE FLAG}

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

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{KEY/DISPLAY} & \textbf{REGISTER} \\
\hline
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 \\
\hline
\end{tabular}
\caption{REGISTER DISPLAY SEQUENCE}
\end{table}
EXAMINE REGISTER EXAMPLE 1

Using ~ to initialize the 8085's stack pointer to 20C8:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAM REG</td>
<td>SPH</td>
<td>**.</td>
</tr>
<tr>
<td>4</td>
<td>SPH</td>
<td>02.</td>
</tr>
<tr>
<td>2</td>
<td>SPH</td>
<td>20.</td>
</tr>
<tr>
<td>NEXT</td>
<td>SPL</td>
<td>**.</td>
</tr>
<tr>
<td>C</td>
<td>SPL</td>
<td>0C.</td>
</tr>
<tr>
<td>0</td>
<td>SPL</td>
<td>C8.</td>
</tr>
<tr>
<td>EXEC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

EXAMINE REGISTER EXAMPLE 2

Using ~ to examine the contents of the 8085's Registers:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAM REG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>b</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>C</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>d</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>E</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>F</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>I</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>H</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>L</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>SPL</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>PCH</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT</td>
<td>PCL</td>
<td>**.</td>
</tr>
<tr>
<td>NEXT or EXEC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

`00 (address)` \[EXEC\]

Pressing the `00` 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 `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 `EXEC` generates an error message.

The monitor regains control of the CPU only after a `MENT` or after execution of an RST 0, RST 1, or JMP 0 instruction in program.

Note that because of the way the GO and SINGLE STEP commands are implemented in the Monitor, `00` and `SING` 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 C8 using `EXEC`, then try it again. (Locations 20C8 to 20FF are reserved for the monitor program, therefore the stack pointer must be set to 20C8 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 \( \text{GO} \) command. First, check to make sure the 3-location program is in memory, then the program will be executed.

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 \( \text{GO} \), try the following sequence:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0002.</td>
<td>A</td>
<td>A 47. A reg now holds 47.</td>
</tr>
<tr>
<td>0</td>
<td>0020.</td>
<td>0</td>
<td>A 00.</td>
</tr>
<tr>
<td>0</td>
<td>0200.</td>
<td>0</td>
<td>A 00.</td>
</tr>
<tr>
<td>0</td>
<td>2000.</td>
<td>EXEC</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>3E</td>
<td>MVI A, 47</td>
</tr>
<tr>
<td>NEXT</td>
<td>2001</td>
<td>47</td>
<td>GO ****. **</td>
</tr>
<tr>
<td>NEXT</td>
<td>2002</td>
<td>CF</td>
<td>RST 1 2 0002.</td>
</tr>
<tr>
<td>EXEC</td>
<td>-</td>
<td>0</td>
<td>0020. Now A holds 0</td>
</tr>
<tr>
<td>EXEC</td>
<td>****.</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0002.</td>
<td>0</td>
<td>0200.</td>
</tr>
<tr>
<td>0</td>
<td>0020.</td>
<td>0</td>
<td>2000.</td>
</tr>
<tr>
<td>0</td>
<td>0200.</td>
<td>EXEC</td>
<td>- 80 85</td>
</tr>
<tr>
<td>0</td>
<td>2000.</td>
<td>EXEC</td>
<td>- 80 85</td>
</tr>
<tr>
<td>EXEC</td>
<td>2</td>
<td>A</td>
<td>A 47 Now A holds 47 again</td>
</tr>
</tbody>
</table>

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

Now try placing other values in location 2001 using \( \text{MEM} \) and use \( \text{GO} \) to execute the program again, seeing how those values are loaded into the A register after execution.
Single Step:

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 SOK-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 SOK-85 Monitor. This is what you should see on the display:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE STEP</td>
<td>****</td>
<td>**</td>
</tr>
<tr>
<td>0</td>
<td>0008</td>
<td></td>
</tr>
<tr>
<td>NEXT</td>
<td>000b</td>
<td>E1</td>
</tr>
<tr>
<td>NEXT</td>
<td>000C</td>
<td>22</td>
</tr>
<tr>
<td>NEXT</td>
<td>000E</td>
<td>F5</td>
</tr>
<tr>
<td>NEXT</td>
<td>0010</td>
<td>E1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td>0010</td>
<td>E1</td>
</tr>
<tr>
<td>EXEC</td>
<td>- 80</td>
<td>85</td>
</tr>
</tbody>
</table>
Vector Interrupt:

The key is similar to the 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 20D4 in user RAM. You may place any instruction you wish in Locations 20D4 thru 20D6 (e.g., a branch to a keyboard interrupt routine). The monitor does not regain control without specific action (a RST or 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 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 ('-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 . The PC value of the next instruction appears in the address field of the display. Then press 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 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 TELETEYPEWRITER 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 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 [Esc] 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 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

<table>
<thead>
<tr>
<th>D9, 26</th>
</tr>
</thead>
<tbody>
<tr>
<td>0009   EF 20 E1 22 F2 20 F5</td>
</tr>
<tr>
<td>0010   E1 22 ED 20 21 00 00 39 22 F4 20 21 ED 20 F9 C5</td>
</tr>
<tr>
<td>0020   D5 C3 3F 00 C3 57 01</td>
</tr>
</tbody>
</table>
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

12010
1122345566778899$

This command puts the following pattern into RAM:

2010 11 22 33 44 55 66 77 88 99

I COMMAND EXAMPLE 2

12040
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>
<thead>
<tr>
<th>IDENTIFIER CODE</th>
<th>REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Register A</td>
</tr>
<tr>
<td>B</td>
<td>Register B</td>
</tr>
<tr>
<td>C</td>
<td>Register C</td>
</tr>
<tr>
<td>D</td>
<td>Register D</td>
</tr>
<tr>
<td>E</td>
<td>Register E</td>
</tr>
<tr>
<td>F</td>
<td>Flags byte</td>
</tr>
<tr>
<td>I</td>
<td>Interrupt Mask</td>
</tr>
<tr>
<td>H</td>
<td>Register H</td>
</tr>
<tr>
<td>L</td>
<td>Register L</td>
</tr>
<tr>
<td>M</td>
<td>Registers H and L combined</td>
</tr>
<tr>
<td>S</td>
<td>Stack Pointer</td>
</tr>
<tr>
<td>P</td>
<td>Program Counter</td>
</tr>
</tbody>
</table>

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:

```
7 6 5 4 3 2 1 0
S Z X A C X P X C
```

- **S**: SIGN
- **Z**: ZERO
- **X**: UNDEFINED
- **A**: AUX CARRY
- **C**: CARRY
- **P**: PARITY
- **C**: ZERO
- **M**: SIGN

The format for the I register:

```
7 6 5 4 3 2 1 0
0 0 0 1 E M M M
```

- **E**: INTERRUPT MASK
- **M**: INTERRUPT ENABLE FLAG

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.
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.

![SDK-85 Functional Block Diagram](image-url)

Figure 5-1 SDK-85 Functional Block Diagram
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 20C8 through 20D6 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 5-1
8205 CHIP ENABLES

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

AXX = IC# on schematic diagram in Appendix
N/C = not connected — available for user expansion
TABLE 5-2
MONITOR-RESERVED RAM LOCATIONS

<table>
<thead>
<tr>
<th>LOC.</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20C8</td>
<td>User may place a JMP instr. to a RST 5 routine in locs 20C8-20CA.</td>
</tr>
<tr>
<td>20C8</td>
<td>JMP to RST 6 routine</td>
</tr>
<tr>
<td>20CE</td>
<td>JMP to RST 6.5 routine (hardwired user interrupt)</td>
</tr>
<tr>
<td>20D1</td>
<td>JMP to RST 7 routine</td>
</tr>
<tr>
<td>20D4</td>
<td>JMP to “VECT INTR” key routine</td>
</tr>
<tr>
<td>20D7-20E8</td>
<td>Monitor Stack (temporary storage used by monitor)</td>
</tr>
<tr>
<td>20E9</td>
<td>E Register</td>
</tr>
<tr>
<td>20EA</td>
<td>D Register</td>
</tr>
<tr>
<td>20EB</td>
<td>C Register</td>
</tr>
<tr>
<td>20EC</td>
<td>B Register</td>
</tr>
<tr>
<td>20ED</td>
<td>Flags</td>
</tr>
<tr>
<td>20EE</td>
<td>A Register</td>
</tr>
<tr>
<td>20EF</td>
<td>L Register</td>
</tr>
<tr>
<td>20F0</td>
<td>H Register</td>
</tr>
<tr>
<td>20F1</td>
<td>Interrupt Mask</td>
</tr>
<tr>
<td>20F2</td>
<td>Prog. Cntr. — Low byte</td>
</tr>
<tr>
<td>20F3</td>
<td>Prog. Cntr. — HI byte</td>
</tr>
<tr>
<td>20F4</td>
<td>Stack Ptr. — Low byte</td>
</tr>
<tr>
<td>20F5</td>
<td>Stack Ptr. — HI byte</td>
</tr>
<tr>
<td>20F6</td>
<td>Current Address</td>
</tr>
<tr>
<td>20F8</td>
<td>Current Data</td>
</tr>
<tr>
<td>20F9-20FC</td>
<td>Output buffer &amp; Temp Locs.</td>
</tr>
<tr>
<td>20FD</td>
<td>Register Pointer</td>
</tr>
<tr>
<td>20FE</td>
<td>Input Buffer</td>
</tr>
<tr>
<td>20FF</td>
<td>8155 Command/Status REGISTER image (loaded by user)</td>
</tr>
</tbody>
</table>
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.

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.

### 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

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

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.

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.

**TABLE 5-4**  
ACCESSING THE 8279 KEYBOARD DISPLAY CONTROLLER

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>READ/WRITE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1800</td>
<td>Read</td>
<td>Read Keyboard FIFO</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>Write Data to Display</td>
</tr>
<tr>
<td>1900</td>
<td>Read</td>
<td>Read Status Word</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>Write Command Word</td>
</tr>
</tbody>
</table>
The display digits are stored within the 8279 display RAM in the locations listed in Table 5-5.

### TABLE 5-5

<table>
<thead>
<tr>
<th>8279 DISPLAY RAM LOCATION</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address digit 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Data Digit 1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>UNUSED</td>
</tr>
<tr>
<td>7</td>
<td>UNUSED</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>INPUT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST 5.5</td>
<td>Dedicated to 8279</td>
</tr>
<tr>
<td>RST 6.5</td>
<td>Available User Interrupt</td>
</tr>
<tr>
<td>RST 7.5</td>
<td>[button interrupt]</td>
</tr>
<tr>
<td>TRAP</td>
<td>8155 Timer Interrupt</td>
</tr>
<tr>
<td>INTR</td>
<td>Available User Interrupt</td>
</tr>
</tbody>
</table>

### 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](image-url)
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 20C8 through 20D6 in RAM are reserved by the monitor for jump instructions when all interrupts are used. Thus, you should set the stack pointer initially at 20C8 (by the use of the program instruction LXI SP, 20C8H (31 C8 20), the keyboard command \( \text{LXI SP, 20C8H} \), 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 20C8 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 \( \text{SOFT} \) 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>
<thead>
<tr>
<th>Calling Address</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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            | NMOOUT   | Hex Number Printer  
NMOOUT 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. |
| 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. |
| 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.  
NOTE: For RDKBD to work correctly, you must first:  
1. Unmask RST 5.5 using the SIM instruction.  
2. Enable interrupts using the EI 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. |
### TABLE 6-1
MONITOR ROUTINE CALLING ADDRESSES (CONT’D)

<table>
<thead>
<tr>
<th>Calling Address</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 be sent.  |

<table>
<thead>
<tr>
<th>Character Displayed</th>
<th>Hexadecimal memory content pointed to by the HL register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>a</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>d</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>10</td>
</tr>
<tr>
<td>P</td>
<td>11</td>
</tr>
<tr>
<td>I</td>
<td>12</td>
</tr>
<tr>
<td>r</td>
<td>13</td>
</tr>
<tr>
<td>Blank</td>
<td>15</td>
</tr>
</tbody>
</table>
PROGRAM EXAMPLE — RDKBD

After executing [02] 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 [] to see that the key value is now in the A register.

<table>
<thead>
<tr>
<th>LOC</th>
<th>CONTENTS</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31</td>
<td>LXI SP, 20C8H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2001</td>
<td>C8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>3E</td>
<td>MVI A, 08H</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>30</td>
<td>SIM</td>
<td>; unmask interrupt</td>
</tr>
<tr>
<td>2006</td>
<td>FB</td>
<td>EI</td>
<td>; enable interrupt</td>
</tr>
<tr>
<td>2007</td>
<td>CD</td>
<td>CALL RDKBD</td>
<td>; read keyboard value</td>
</tr>
<tr>
<td>2008</td>
<td>E7</td>
<td></td>
<td>; into Reg A</td>
</tr>
<tr>
<td>2009</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200A</td>
<td>CF</td>
<td>RST 1</td>
<td>; break point, go back to monitor</td>
</tr>
</tbody>
</table>

PROGRAM EXAMPLE — UPDDT

Display FF in data field of display.

<table>
<thead>
<tr>
<th>LOC</th>
<th>CONTENTS</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31</td>
<td>LXI SP, 20C8H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2001</td>
<td>C8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>3E</td>
<td>MVI A, FFH</td>
<td>; load FF into Reg A</td>
</tr>
<tr>
<td>2004</td>
<td>FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>CD</td>
<td>CALL UPDDT</td>
<td>; output Reg A to data field</td>
</tr>
<tr>
<td>2006</td>
<td>6E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>76</td>
<td>HLT</td>
<td>; HALT</td>
</tr>
</tbody>
</table>

To change the display value use [ ] 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.

<table>
<thead>
<tr>
<th>LOC</th>
<th>CONTENTS</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31C820</td>
<td>LXI SP, 20C8H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2003</td>
<td>3E08</td>
<td>MVI A, 08H</td>
<td>; unmask interrupt</td>
</tr>
<tr>
<td>2005</td>
<td>30</td>
<td>SIM</td>
<td>; enable interrupt</td>
</tr>
<tr>
<td>2006</td>
<td>FB</td>
<td>LOOP: EI</td>
<td>; read keyboard value into Reg A</td>
</tr>
<tr>
<td>2007</td>
<td>CDE702</td>
<td>CALL RDKBD</td>
<td>; output Reg A to data field</td>
</tr>
<tr>
<td>200A</td>
<td>CD6E03</td>
<td>CALL UPDDT</td>
<td></td>
</tr>
<tr>
<td>200D</td>
<td>C30620</td>
<td>JMP LOOP</td>
<td>; keep looping</td>
</tr>
</tbody>
</table>
PROGRAM EXAMPLE — COUNTDOWN

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

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CONTENTS</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31</td>
<td>LXI SP, 2080H</td>
<td>INITIALIZE STACK POINTER.</td>
</tr>
<tr>
<td>2001</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>3E</td>
<td>MVI A, 08</td>
<td>USE THE 8085\text{'S SIM INSTR TO</td>
</tr>
<tr>
<td>2004</td>
<td>08</td>
<td></td>
<td>ENABLE THE VECT INTR BUTTON.</td>
</tr>
<tr>
<td>2005</td>
<td>30</td>
<td>SIM</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>FB</td>
<td>LOOP: EI</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>78</td>
<td>MOV A, B</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>3C</td>
<td>INR A</td>
<td>INCREMENT AND ADJUST THE COUNT</td>
</tr>
<tr>
<td>2009</td>
<td>27</td>
<td>DAA</td>
<td>FOR DECIMAL COUNTING.</td>
</tr>
<tr>
<td>200A</td>
<td>47</td>
<td>MOV B, A</td>
<td></td>
</tr>
<tr>
<td>200B</td>
<td>C5</td>
<td>PUSH B</td>
<td></td>
</tr>
<tr>
<td>200C</td>
<td>CD</td>
<td>CALL UPDDT</td>
<td>DISPLAY COUNT IN DATA FIELD OF</td>
</tr>
<tr>
<td>200D</td>
<td>6E</td>
<td></td>
<td>DISPLAY.</td>
</tr>
<tr>
<td>200E</td>
<td>03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200F</td>
<td>16</td>
<td>MVI D, 18H</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>CD</td>
<td>CALL DELAY</td>
<td>WAIT OUT A PROGRAMMABLE DELAY</td>
</tr>
<tr>
<td>2012</td>
<td>F1</td>
<td></td>
<td>PERIOD BEFORE CONTINUING.</td>
</tr>
<tr>
<td>2013</td>
<td>05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>C1</td>
<td>POP B</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>C3</td>
<td>JMP LOOP</td>
<td>GO BACK TO THE BEGINNING.</td>
</tr>
<tr>
<td>2016</td>
<td>06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20D4</td>
<td>FB</td>
<td>EI</td>
<td>CONTROL BRANCHES TO LOCATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20D4 WHEN VECT INTR IS PRESSED.</td>
</tr>
<tr>
<td>20D5</td>
<td>76</td>
<td>HLT</td>
<td>WAIT HERE FOR KEY DEPRESSION.</td>
</tr>
<tr>
<td>20D6</td>
<td>C9</td>
<td>RET</td>
<td>RESUME THE COUNT.</td>
</tr>
</tbody>
</table>

To execute the program, type in 2000 \( \text{INR} \).

Try to stop the count right at 00 using the \( \text{VECT} \) key.

Change the speed of the count by using \( \text{SHORT} \) to vary the contents of location 2010.
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 (G2010~).

The display will flash “HELP”.

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DATA</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>31C820</td>
<td>LXI SP, 20C8H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2013</td>
<td>3E01</td>
<td>MVI A, 1</td>
<td>; use data field</td>
</tr>
<tr>
<td>2015</td>
<td>0600</td>
<td>MVI B, 0</td>
<td>; no decimal indicator</td>
</tr>
<tr>
<td>2017</td>
<td>210420</td>
<td>LXI H, 2004H</td>
<td>; use characters starting at Location 2004</td>
</tr>
<tr>
<td>201A</td>
<td>CDB702</td>
<td>CALL OUTPT</td>
<td>; output the two characters to data field</td>
</tr>
<tr>
<td>201D</td>
<td>3E00</td>
<td>MVI A, 0</td>
<td>; use address field</td>
</tr>
<tr>
<td>201F</td>
<td>0600</td>
<td>MVI B, 0</td>
<td>; no decimal indicator</td>
</tr>
<tr>
<td>2021</td>
<td>210020</td>
<td>LXI H, 2000H</td>
<td>; use characters starting at Location 2000</td>
</tr>
<tr>
<td>2024</td>
<td>CDB702</td>
<td>CALL OUTPT</td>
<td>; output the four characters to address field</td>
</tr>
<tr>
<td>2027</td>
<td>11FFFF</td>
<td>LXI D, 0FFFFH*</td>
<td>; set up delay value</td>
</tr>
<tr>
<td>202A</td>
<td>CDF105</td>
<td>CALL DELAY</td>
<td>; time delay</td>
</tr>
<tr>
<td>202D</td>
<td>3E00</td>
<td>MOV A, 0</td>
<td>;</td>
</tr>
<tr>
<td>202F</td>
<td>0600</td>
<td>MOV B, 0</td>
<td>;</td>
</tr>
<tr>
<td>2031</td>
<td>210420</td>
<td>LXI H, 2004H</td>
<td>; output BLANKS to</td>
</tr>
<tr>
<td>2034</td>
<td>CDB702</td>
<td>CALL OUTPT</td>
<td>; Display</td>
</tr>
<tr>
<td>2037</td>
<td>11FFFF</td>
<td>LXI D, 0FFFFH</td>
<td>;</td>
</tr>
<tr>
<td>203A</td>
<td>CDF105</td>
<td>CALL DELAY</td>
<td>; time Delay</td>
</tr>
<tr>
<td>203D</td>
<td>C31D20</td>
<td>JMP DPY</td>
<td>; REPEAT</td>
</tr>
</tbody>
</table>

†Delay time proportional to value. Any number from 1 through FFFF may be chosen.
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. TELETYPewriter) 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 IS REACHED, THE "ARG - " IS REMOVED.
LOC DEJ      SEQ                      SOURCE STATEMENT

53 ; CLDST
54 ; DISPC
55 ; ERR
56 ; GTXEX
57 ; HRDSP
58 ; IIMINT
59 ; INSDG
60 ; NXTRG
61 ; OUTPT
62 ; RKED
63 ; RETF
64 ; RETT
65 ; RLOC
66 ; RSTOP
67 ; SETRG
68 ; UPDAD
69 ; UPDDT
70 ;
71 ; NAME SDK85
72 ;*******************************************************************************
73 ; SET CONDITIONAL ASSEMBLY FLAG
74 ;*******************************************************************************
75 ;
76;
77;
78;
79;
80 WAITS SET 0 ; 0=NO WAIT STATES
81 ; 1=A WAIT STATE IS GENERATED FOR EVERY M CYCLE
82 ; THE APPROPRIATE DELAY TIME MUST BE USED FOR
83 ; TTY DELAY OR SET UP SINGLE
84 ; STEP TIMER FOR EACH CASE
85 ;
86 ;
87 ;*******************************************************************************
88 ; MONITOR EQUATES
89 ;*******************************************************************************
90 ;
91 ;
92 ;
93 RAMST EQU 2000H ; START ADDRESS OF RAM - THIS PROGRAM ASSUMES
94 ; THAT 256 BYTES OF RANDOM ACCESS MEMORY BEGIN AT THIS ADDRESS.
95 ; THE PROGRAM USES STORAGE AT THE END OF THIS SPACE FOR VARIABLES.
96 ; SAVING REGISTERS AND THE PROGRAM STACK
97 ;
98 RHUSE EQU 23 ; RAM USAGE - CURRENTLY, 23 BYTES ARE USED FOR
99 ; /SAVING REGISTERS AND VARIABLES
100 ;
101 SKLN EQU 18 ; MONITOR STACK USAGE - MAX OF 9 LEVELS
102 ;
103 UBLN EQU 15 ; 5 USER BRANCHES - 3 BYTES EACH
104 ;
105 ADFLD EQU 0 ; INDICATES USE OF ADDRESS FIELD OF DISPLAY
106 ADISP EQU 90H ; CONTROL CHARACTER TO INDICATE OUTPUT TO
107 ; /ADDRESS FIELD OF DISPLAY
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>108</td>
<td>CNTRL</td>
<td>EQU 1900H ; ADDRESS FOR SENDING CONTROL CHARACTERS TO</td>
</tr>
<tr>
<td>109</td>
<td></td>
<td></td>
<td>; /DISPLAY CHIP</td>
</tr>
<tr>
<td>0001</td>
<td>110</td>
<td>COMMA</td>
<td>EQU 11H ; COMMA FROM KEYBOARD</td>
</tr>
<tr>
<td>0000</td>
<td>111</td>
<td>CSNIT</td>
<td>EQU 0 ; INITIAL VALUE FOR COMMAND STATUS REGISTER</td>
</tr>
<tr>
<td>0020</td>
<td>112</td>
<td>CSR</td>
<td>EQU 20H ; OUTPUT PORT FOR COMMAND STATUS REGISTER</td>
</tr>
<tr>
<td>0094</td>
<td>113</td>
<td>DDISP</td>
<td>EQU 94H ; CONTROL CHARACTER TO INDICATE OUTPUT TO</td>
</tr>
<tr>
<td>114</td>
<td></td>
<td></td>
<td>; /DATA FIELD OF DISPLAY</td>
</tr>
<tr>
<td>0001</td>
<td>115</td>
<td>DOT</td>
<td>EQU 1 ; INDICATOR FOR DOT IN DISPLAY</td>
</tr>
<tr>
<td>1800</td>
<td>116</td>
<td>DFLD</td>
<td>EQU 1800H ; ADDRESS FOR SENDING CHARACTERS TO DISPLAY</td>
</tr>
<tr>
<td>0001</td>
<td>117</td>
<td>DFLD</td>
<td>EQU 1 ; INDICATES USE OF DATA FIELD OF DISPLAY</td>
</tr>
<tr>
<td>0008</td>
<td>118</td>
<td>DTMSK</td>
<td>EQU 08H ; MASK FOR TURNING ON DOT IN DISPLAY</td>
</tr>
<tr>
<td>0080</td>
<td>119</td>
<td>EMTPY</td>
<td>EQU 80H ; HIGH ORDER 1 INDICATES EMPTY INPUT BUFFER</td>
</tr>
<tr>
<td>00CC</td>
<td>120</td>
<td>KENIT</td>
<td>EQU 0CCH ; CONTROL CHARACTER TO SET DISPLAY OUTPUT TO</td>
</tr>
<tr>
<td>121</td>
<td></td>
<td></td>
<td>; /ALL ONES DURING BLANKING PERIOD</td>
</tr>
<tr>
<td>0000</td>
<td>122</td>
<td>KMODE</td>
<td>EQU 0 ; CONTROL CHAR. TO SET KEYBOARD/DISPLAY MODE</td>
</tr>
<tr>
<td>123</td>
<td></td>
<td></td>
<td>; (2 KEY ROLLOVER, 8 CHARACTER LEFT ENTRY)</td>
</tr>
<tr>
<td>20E9</td>
<td>124</td>
<td>MSTK</td>
<td>EQU RAMST + 256 - RMUSE ; START OF MONITOR STACK</td>
</tr>
<tr>
<td>0000</td>
<td>125</td>
<td>NODOT</td>
<td>EQU 0 ; INDICATOR FOR NO DOT IN DISPLAY</td>
</tr>
<tr>
<td>126</td>
<td></td>
<td>NUMC</td>
<td>- DEFINED LATER ; NUMBER OF COMMANDS</td>
</tr>
<tr>
<td>127</td>
<td></td>
<td>NUMRG</td>
<td>- DEFINED LATER ; NUMBER OF REGISTER SAVE LOCATIONS</td>
</tr>
<tr>
<td>0010</td>
<td>128</td>
<td>PERIO</td>
<td>EQU 10H ; PERIOD FROM KEYBOARD</td>
</tr>
<tr>
<td>00FB</td>
<td>129</td>
<td>PRMPT</td>
<td>EQU OFBH ; PROMPT CHARACTER FOR DISPLAY (DASH)</td>
</tr>
<tr>
<td>0040</td>
<td>130</td>
<td>READ</td>
<td>EQU 40H ; CONTROL CHARACTER TO INDICATE INPUT FROM</td>
</tr>
<tr>
<td>131</td>
<td></td>
<td></td>
<td>; /KEYBOARD</td>
</tr>
<tr>
<td>0025</td>
<td>132</td>
<td>TIMHI</td>
<td>EQU 25H ; OUTPUT PORT FOR HIGH ORDER BYTE OF TIMER VALUE</td>
</tr>
<tr>
<td>0024</td>
<td>133</td>
<td>TIMLO</td>
<td>EQU 24H ; OUTPUT PORT FOR LOW ORDER BYTE OF TIMER VALUE</td>
</tr>
<tr>
<td>0040</td>
<td>134</td>
<td>TMODE</td>
<td>EQU 40H ; TIMER MODE - SQUARE WAVE, AUTO RELOAD</td>
</tr>
<tr>
<td>00CO</td>
<td>135</td>
<td>TSTRT</td>
<td>EQU 0COH ; START TIMER</td>
</tr>
<tr>
<td>00CE</td>
<td>136</td>
<td>UNMSK</td>
<td>EQU 0EH ; UNMASK INPUT INTERRUPT</td>
</tr>
<tr>
<td>20C8</td>
<td>137</td>
<td>USRBR</td>
<td>EQU RAMST + 256 - (RMUSE + SKLN + UBRLN) ; START OF USER</td>
</tr>
<tr>
<td>138</td>
<td></td>
<td></td>
<td>; /BRANCH LOCATIONS</td>
</tr>
<tr>
<td>139</td>
<td></td>
<td>IF</td>
<td>1-WAITS ; TIMER VALUE FOR SINGLE STEP IF NO WAIT STATE</td>
</tr>
<tr>
<td>140</td>
<td></td>
<td>TIMER</td>
<td>EQU 197</td>
</tr>
<tr>
<td>141</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>142</td>
<td></td>
<td>IF</td>
<td>WAITS ; TIMER VALUE FOR SINGLE STEP IF ONE WAIT STATE INSERTE</td>
</tr>
<tr>
<td>143</td>
<td></td>
<td>TIMER</td>
<td>EQU 237</td>
</tr>
<tr>
<td>144</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>145</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>146</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>147</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>148</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>149</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>151</td>
<td></td>
<td></td>
<td>; ENDF</td>
</tr>
<tr>
<td>152</td>
<td></td>
<td>TRUE</td>
<td>MACRO WHERE ; BRANCH IF FUNCTION RETURNS TRUE</td>
</tr>
<tr>
<td>153</td>
<td></td>
<td>JC</td>
<td>WHERE</td>
</tr>
<tr>
<td>154</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
<tr>
<td>155</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
<tr>
<td>156</td>
<td></td>
<td>FALSE</td>
<td>MACRO WHERE ; BRANCH IF FUNCTION RETURNS FALSE</td>
</tr>
<tr>
<td>157</td>
<td></td>
<td>JNC</td>
<td>WHERE</td>
</tr>
<tr>
<td>158</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
<tr>
<td>159</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
<tr>
<td>160</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
<tr>
<td>161</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
<tr>
<td>162</td>
<td></td>
<td></td>
<td>; ENDM</td>
</tr>
</tbody>
</table>
LOC OBJ SEQ SOURCE STATEMENT

163 ; "RESET" KEY ENTRY POINT - COLD START
164 ; RST 0 ENTRY POINT
165 ;

0000 3E00 166 MVI A,KMODE ; GET CONTROL CHARACTER
0002 3200 167 STA CNTRL ; SET KEYBOARD/DISPLAY MODE
0005 C3F1 168 JMP CLDBK ; GO FINISH COLD START
169 CLDBK:
170 ;
171 ; RST 1 ENTRY POINT - WARM START
172 ;

0008 173 ORG 8
174 ; SAVE REGISTERS
0008 22EF 175 SHLD LSAV ; SAVE H & L REGISTERS
000E E1 176 POP H ; GET USER PROGRAM COUNTER FROM TOP OF STACK
000C 22F2 177 SHLD PSAV ; /AND SAVE IT
000F F5 178 PUSH PSW
0010 E1 179 POP H
0011 22ED 180 SHLD PSAV ; SAVE FLIP/FLOPS & REGISTER A
0014 210000 181 LXI H,0 ; CLEAR H & L
0017 39 182 DAD SP ; GET USER STACK POINTER
0018 22F4 183 SHLD SSAV ; /AND SAVE IT
001B 21ED 184 LXI H,BSAV+1 ; SET STACK POINTER FOR SAVING
001F C5 185 PUSH B ; SAVE B & C
0020 D5 187 PUSH D ; SAVE D & E
0021 C3F0 188 JMP RES10 ; LEAVE ROOM FOR VECTORED INTERRUPTS

189 ;
190 ; TIMER INTERRUPT (TRAP) ENTRY POINT (RST 4.5)
0024 191 ORG 24H
0024 C357 192 JMP STP25 ; BACK TO SINGLE STEP ROUTINE
193 ;
194 ; RST 5 ENTRY POINT
195 ;
0028 196 ORG 28H
0028 C3C8 197 JMP RSET5 ; BRANCH TO RST 5 LOCATION IN RAM
196 ;

199 ; INPUT INTERRUPT ENTRY POINT (RST 5.5)
200 ;
002C 201 ORG 2CH
002C C3E0 202 JMP ININT ; BRANCH TO INPUT INTERRUPT ROUTINE
203 ;
204 ; RST 6 ENTRY POINT
205 ;
0030 206 ORG 30H
0030 C3C8 207 JMP RSET6 ; BRANCH TO RST 6 LOCATION IN RAM
208 ;
209 ; HARD WIRED USER INTERRUPT ENTRY POINT (RST 6.5)
210 ;
0034 211 ORG 34H
0034 C3CE 212 JMP RST65 ; BRANCH TO RST 6.5 LOCATION IN RAM
213 ;
214 ; RST 7 ENTRY POINT
215 ;
0038 216 ORG 38H
0038 C3D1 217 JMP RSET7 ; BRANCH TO RST 7 LOCATION IN RAM
ORIGINAL TEXT:

DOC DIRECTIVES:

% 8080/8085 MACRO ASSEMBLER X108 SDK85

LOC OBJ SEQ SOURCE STATEMENT

218 ;
219 ; ***** "VINCORED INTERRUPT" KEY ENTRY POINT (RST 7.5)
003C 220 ORG 3CH
003C C3D420 221 JMP USINT ; BRANCH TO USER INTERRUPT LOCATION IN RAM
222 ;
223 RES10: ; CONTINUE SAVING USER STATUS
003F 224 D1M ; GET USER INTERRUPT STATUS AND INTERRUPT MASK
0040 E60F 225 ANI OFH ; KEEP STATUS & MASK BITS
0042 32F120 226 STA ISA ; SAVE INTERRUPT STATUS & MASK
0045 3EOE 227 MVI A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE
0047 30 228 SIM
0048 F3 229 DI ; INTERRUPTS DISABLED WHILE MONITOR IS RUNNING
0049 230 (EXCEPT WHEN WAITING FOR INPUT)
0049 231 RIM ; TTY OR KEYBOARD MONITOR?
004A 07 232 RLC ; IS TTY CONNECTED?
004B DAEA03 233 JC GO ; YES - BRANCH TO TTY MONITOR
234 ; NO - ENTER KEYBOARD MONITOR
235 ;
236 ;
237 ;
238 ;
239 ;
240 ;
241 ;
242 ;
243 XRA A ; ARG - USE ADDRESS FIELD OF DISPLAY
244 MVI B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0051 21A603 245 LXI H,SGNAD ; ARG - GET ADDRESS OF ADDRESS FIELD PORTION OF
246 ; /SIGN-ON MESSAGE
0054 CDB702 247 CALL OUTPT ; OUTPUT SIGN-ON MESSAGE TO ADDRESS FIELD
0057 3E80 248 MVI A,DTFLD ; ARG - USE DATA FIELD OF DISPLAY
0059 0600 249 MVI B,NODOT ; ARG - NO DOT IN DATA FIELD
005B 21A003 250 LXI H,SGNAD ; ARG - GET ADDRESS OF DATA FIELD PORTION OF
251 ; /SIGN-ON MESSAGE
005E CDB702 252 CALL OUTPT ; OUTPUT SIGN-ON MESSAGE TO DATA FIELD
0061 3E80 253 MVI A,EMPTY
254 STA IBUFF ; SET INPUT BUFFER EMPTY FLAG
255 ;
256 ;
257 ;
258 ; FUNCTION: CMMND - COMMAND RECOGNIZER
259 ; INPUTS: NONE
260 ; OUTPUTS: NONE
261 ; CALLS: RDKBD,ERR,SUBST,EXAM,GOCMD,SSTEP
262 ; DESEOYS: A,B,C,E,H,L,F/F'S
263 ;
264 CMMND:
0066 21E920 265 LXI H,MNSTK ; INITIALIZE MONITOR STACK POINTER
0069 F9 266 SPHL
267 ; OUTPUT PROMPT CHARACTER TO DISPLAY
006A 210019 268 LXI H,CNTRL ; GET ADDRESS FOR CONTROL CHARACTER
006D 3690 269 MVI A,ADISP ; OUTPUT CONTROL CHARACTER TO USE ADDRESS FIELD
006F 25 270 DCR H ; ADDRESS FOR OUTPUT CHARACTER
0070 3E80 271 MVI A,PRMPT ; OUTPUT PROMPT CHARACTER
0072 CDE702 272 CALL RDKBD ; READ KEYSBOARD
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0075</td>
<td>010400</td>
<td>273</td>
<td>LXI B, NUMC ; COUNTER FOR NUMBER OF COMMANDS IN C</td>
</tr>
<tr>
<td>0078</td>
<td>217803</td>
<td>274</td>
<td>LXI H, CMDTAB ; GET ADDRESS OF COMMAND TABLE</td>
</tr>
<tr>
<td>007B</td>
<td>BE</td>
<td>275</td>
<td>CMD10:</td>
</tr>
<tr>
<td>007C</td>
<td>CA8700</td>
<td>276</td>
<td>CIN M ; RECOGNIZE THE COMMAND?</td>
</tr>
<tr>
<td>007F</td>
<td>23</td>
<td>277</td>
<td>JZ CMD15 ; YES - GO PROCESS IT</td>
</tr>
<tr>
<td>0080</td>
<td>OD</td>
<td>278</td>
<td>INX H ; NO - NEXT COMMAND TABLE ENTRY</td>
</tr>
<tr>
<td>0081</td>
<td>C2B00</td>
<td>279</td>
<td>DCR C ; END OF TABLE?</td>
</tr>
<tr>
<td>0084</td>
<td>C31502</td>
<td>280</td>
<td>JNZ CMD10 ; NO - GO CHECK NEXT ENTRY</td>
</tr>
<tr>
<td>0087</td>
<td>217C03</td>
<td>281</td>
<td>; YES - COMMAND UNKNOWN</td>
</tr>
<tr>
<td>008A</td>
<td>OD</td>
<td>282</td>
<td>JMP ERR ; DISPLAY ERROR MESSAGE AND GET ANOTHER COMMAND</td>
</tr>
<tr>
<td>008B</td>
<td>09</td>
<td>283</td>
<td>CMD15:</td>
</tr>
<tr>
<td>008C</td>
<td>09</td>
<td>284</td>
<td>LXI H, CMDAD ; GET ADDRESS OF COMMAND ADDRESS TABLE</td>
</tr>
<tr>
<td>008D</td>
<td>7E</td>
<td>285</td>
<td>DCR C ; ADJUST COMMAND COUNTER</td>
</tr>
<tr>
<td>008E</td>
<td>23</td>
<td>286</td>
<td>; COUNTER ACTS AS POINTER TO COMMAND ADDRESS TABLE</td>
</tr>
<tr>
<td>008F</td>
<td>66</td>
<td>287</td>
<td>DAD B ; ADD POINTER TO TABLE ADDRESS TWICE BECAUSE</td>
</tr>
<tr>
<td>0090</td>
<td>6F</td>
<td>288</td>
<td>DAD B ; TABLE HAS 2 BYTE ENTRIES</td>
</tr>
<tr>
<td>0091</td>
<td>E9</td>
<td>289</td>
<td>MOV A, M ; GET LOW ORDER BYTE OF COMMAND ADDRESS</td>
</tr>
<tr>
<td>0092</td>
<td>0601</td>
<td>290</td>
<td>INX H</td>
</tr>
<tr>
<td>0094</td>
<td>CD701</td>
<td>291</td>
<td>MOV H, M ; GET HIGH ORDER BYTE OF COMMAND ADDRESS IN H</td>
</tr>
<tr>
<td>0097</td>
<td>CD403</td>
<td>292</td>
<td>MOV L, A ; PUT LOW ORDER BYTE IN L</td>
</tr>
<tr>
<td>0099</td>
<td>E9</td>
<td>293</td>
<td>; COMMAND ROUTINE ADDRESS IS NOW IN H &amp; L</td>
</tr>
<tr>
<td>009A</td>
<td>D21502</td>
<td>294</td>
<td>PCHL ; BRANCH TO ADDRESS IN H &amp; L</td>
</tr>
</tbody>
</table>

**COMMAND ROUTINES**

**FUNCTION: EXAM - EXAMINE AND MODIFY REGISTERS**

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0092</td>
<td>0601</td>
<td>309</td>
<td>MVI B, DOT ; ARG - DOT IN ADDRESS FIELD OF DISPLAY</td>
</tr>
<tr>
<td>0094</td>
<td>CD701</td>
<td>310</td>
<td>CALL CLEAR ; CLEAR DISPLAY</td>
</tr>
<tr>
<td>0097</td>
<td>CD403</td>
<td>311</td>
<td>CALL SETRG ; GET REGISTER DESIGNATOR FROM KEYBOARD AND</td>
</tr>
<tr>
<td>0099</td>
<td>E9</td>
<td>312</td>
<td>; SET REGISTER POINTER ACCORDINGLY</td>
</tr>
<tr>
<td>0102</td>
<td>CD093</td>
<td>313</td>
<td>; WAS CHARACTER A REGISTER DESIGNATOR?</td>
</tr>
<tr>
<td>00A0</td>
<td>CDFC02</td>
<td>314</td>
<td>FALSE ERR ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND</td>
</tr>
<tr>
<td>00A3</td>
<td>7E</td>
<td>315+</td>
<td>JNC ERR</td>
</tr>
<tr>
<td>00A4</td>
<td>32F820</td>
<td>316</td>
<td>EXM05:</td>
</tr>
<tr>
<td>00A5</td>
<td>0601</td>
<td>317</td>
<td>CALL RNAM ; OUTPUT REGISTER NAME TO ADDRESS FIELD</td>
</tr>
<tr>
<td>00A7</td>
<td>CD603</td>
<td>318</td>
<td>CALL RGLOC ; GET REGISTER SAVE LOCATION IN H &amp; L</td>
</tr>
<tr>
<td>00AC</td>
<td>0601</td>
<td>319</td>
<td>MOV A, M ; GET REGISTER CONTENTS</td>
</tr>
<tr>
<td>00AE</td>
<td>CD202</td>
<td>320</td>
<td>STA CURDT ; STORE REGISTER CONTENTS AT CURRENT DATA</td>
</tr>
<tr>
<td>00B1</td>
<td>D2B800</td>
<td>321</td>
<td>MVI P, DOT ; ARG - DOT IN DATA FIELD</td>
</tr>
<tr>
<td>00B4</td>
<td>CDFC02</td>
<td>322</td>
<td>CALL UPDDT ; UPDATE DATA FIELD OF DISPLAY</td>
</tr>
<tr>
<td>00B5</td>
<td>CD603</td>
<td>323</td>
<td>MVI B, DTFLD ; ARG - USE DATA FIELD OF DISPLAY</td>
</tr>
<tr>
<td>00BF</td>
<td>CD202</td>
<td>324</td>
<td>CALL GTHEX ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?</td>
</tr>
<tr>
<td>00C1</td>
<td>D2800</td>
<td>325</td>
<td>FALSE EXM10 ; NO - DO NOT UPDATE REGISTER CONTENTS</td>
</tr>
<tr>
<td>00C4</td>
<td>CDFC02</td>
<td>326+</td>
<td>JNC EXM10</td>
</tr>
<tr>
<td>00C7</td>
<td>CD403</td>
<td>327</td>
<td>CALL RGLOC ; YES - GET REGISTER SAVE LOCATION IN H &amp; L</td>
</tr>
</tbody>
</table>
LOC OBJ  SEQ  SOURCE STATEMENT

00B7 73 328  MOV  M,E ; UPDATE REGISTER CONTENTS
00B8 FE10 329  EXM10:
00B9 CAE901 330  CPI  PERIO ; WAS LAST CHARACTER A PERIOD ?
00BD FE11 331  JZ  CLDIS ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
00BF C21502 332  CPI  COMMA ; WAS LAST CHARACTER ',' ?
00C2 CDA802 333  JNZ  ERR ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
00C5 DA9D00 334  CALL  NXRG ; YES - ADVANCE REGISTER POINTER TO
00C8 3E901 335  /NEXT REGISTER
336  ; ANY MORE REGISTERS ?
337  TRUE  EXM05 ; YES - CONTINUE PROCESSING WITH NEXT REGISTER
00CE D49D00 338  JC  EXM05
00CF 3E901 339  JMP  CLDIS ; NO - CLEAR DISPLAY AND TERMINATE COMMAND

340 ;
341 ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; }
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00FD</td>
<td>CD0002</td>
<td>383</td>
<td>CALL DISPC; DISPLAY USER PROGRAM COUNTER</td>
</tr>
<tr>
<td>0100</td>
<td>CD8702</td>
<td>384</td>
<td>CALL RDKBD; READ FROM KEYBOARD</td>
</tr>
<tr>
<td>0103</td>
<td>FE10</td>
<td>385</td>
<td>CPI PERIO; WAS CHARACTER A PERIOD?</td>
</tr>
<tr>
<td>0105</td>
<td>CAE901</td>
<td>386</td>
<td>JZ CLDIS; YES - CLEAR DISPLAY AND TERMINATE COMMAND</td>
</tr>
<tr>
<td>0108</td>
<td>FE11</td>
<td>387</td>
<td>CPI COMMA; WAS LAST CHARACTER ',', '?'</td>
</tr>
<tr>
<td>010A</td>
<td>CA2601</td>
<td>388</td>
<td>JZ STP20; YES - GO SET TIMER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>389</td>
<td>; NO - CHARACTER FROM KEYBOARD WAS NEITHER PERIOD NOR COMMA</td>
</tr>
<tr>
<td>010D</td>
<td>32FE20</td>
<td>390</td>
<td>STA IBUFF; REPLACE THE CHARACTER IN THE INPUT BUFFER</td>
</tr>
<tr>
<td>0110</td>
<td>0601</td>
<td>391</td>
<td>MVI B,DOT; ARG - DOT IN ADDRESS FIELD</td>
</tr>
<tr>
<td>0112</td>
<td>CDD701</td>
<td>392</td>
<td>CALL CLEAR; CLEAR DISPLAY</td>
</tr>
<tr>
<td>0115</td>
<td>0600</td>
<td>393</td>
<td>MVI B,ADFLD; ARG - USE ADDRESS FIELD OF DISPLAY</td>
</tr>
<tr>
<td>0117</td>
<td>CD2B02</td>
<td>394</td>
<td>CALL GTHEX; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>395</td>
<td>FALSE ERR; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND</td>
</tr>
<tr>
<td>011A</td>
<td>D21502</td>
<td>396</td>
<td>JNC ERR</td>
</tr>
<tr>
<td>011D</td>
<td>EB</td>
<td>397</td>
<td>XCHG</td>
</tr>
<tr>
<td>011E</td>
<td>22F220</td>
<td>398</td>
<td>SHLD PSAV; HEX VALUE IS NEW USER PC</td>
</tr>
<tr>
<td>0121</td>
<td>FE10</td>
<td>399</td>
<td>CPI PERIO; WAS LAST CHARACTER FROM GTHEX A PERIOD?</td>
</tr>
<tr>
<td>0123</td>
<td>CAE901</td>
<td>400</td>
<td>JZ CLDIS; YES - CLEAR DISPLAY AND TERMINATE COMMAND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>401</td>
<td>NO - MUST HAVE BEEN A COMMA</td>
</tr>
<tr>
<td>0126</td>
<td>3AF120</td>
<td>402</td>
<td>STP20:</td>
</tr>
<tr>
<td>0129</td>
<td>E608</td>
<td>403</td>
<td>LDA ISAV; GET USER INTERRUPT MASK</td>
</tr>
<tr>
<td>012B</td>
<td>32FD20</td>
<td>404</td>
<td>ANI 08H; KEEP INTERRUPT STATUS</td>
</tr>
<tr>
<td>012B</td>
<td>32FD20</td>
<td>405</td>
<td>STA TEMP; SAVE USER INTERRUPT STATUS</td>
</tr>
<tr>
<td>012E</td>
<td>2AF220</td>
<td>406</td>
<td>LLHL PSAV; GET USER PC</td>
</tr>
<tr>
<td>0131</td>
<td>7E</td>
<td>407</td>
<td>MOV A,M; GET USER INSTRUCTION</td>
</tr>
<tr>
<td>0132</td>
<td>3EF3</td>
<td>408</td>
<td>CPI (DI); DI INSTRUCTION</td>
</tr>
<tr>
<td>0134</td>
<td>C23B01</td>
<td>409</td>
<td>JNZ STP21; NO</td>
</tr>
<tr>
<td>0137</td>
<td>AF</td>
<td>410</td>
<td>XRA A; YES - RESET USER INTERRUPT STATUS</td>
</tr>
<tr>
<td>013B</td>
<td>C34201</td>
<td>411</td>
<td>JMP STP22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>412</td>
<td>STP21:</td>
</tr>
<tr>
<td>013B</td>
<td>FEB</td>
<td>413</td>
<td>CPI (EI); EI INSTRUCTION ?</td>
</tr>
<tr>
<td>013D</td>
<td>C2501</td>
<td>414</td>
<td>JNZ STP23; NO</td>
</tr>
<tr>
<td>0140</td>
<td>3E08</td>
<td>415</td>
<td>MVI A,08H; YES - SET USER INTERRUPT STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>416</td>
<td>STP22:</td>
</tr>
<tr>
<td>0142</td>
<td>32FD20</td>
<td>417</td>
<td>STA TEMP; SAVE NEW USER INTERRUPT STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>418</td>
<td>STP23:</td>
</tr>
<tr>
<td>0145</td>
<td>3E40</td>
<td>419</td>
<td>MVI A, (TIMER SHR 8) OR TMODE; HIGH ORDER BITS OF TIMER VALUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>420</td>
<td>; /OR'ED WITH TIMER MODE</td>
</tr>
<tr>
<td>0147</td>
<td>D325</td>
<td>421</td>
<td>OUT TIMHI</td>
</tr>
<tr>
<td>0149</td>
<td>3EC5</td>
<td>422</td>
<td>MVI A,TIMER AND OFFH; LOW ORDER BITS OF TIMER VALUE</td>
</tr>
<tr>
<td>014B</td>
<td>D324</td>
<td>423</td>
<td>OUT TIMLO</td>
</tr>
<tr>
<td>014D</td>
<td>3AFF20</td>
<td>424</td>
<td>LDA USCSR; GET USER IMAGE OF WHAT'S IN CSR</td>
</tr>
<tr>
<td>0150</td>
<td>F6C0</td>
<td>425</td>
<td>ORI TSTRT; SET TIMER COMMAND BITS TO START TIMER</td>
</tr>
<tr>
<td>0152</td>
<td>D320</td>
<td>426</td>
<td>OUT CSR; START TIMER</td>
</tr>
<tr>
<td>0154</td>
<td>C31B03</td>
<td>427</td>
<td>JMP RSTOR; RESTORE USER REGISTERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>428</td>
<td>; STP25:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>429</td>
<td>; BRANCH HERE WHEN TIMER INTERRUPTS AFTER</td>
</tr>
<tr>
<td>0157</td>
<td>F5</td>
<td>430</td>
<td>; ONE USER INSTRUCTION</td>
</tr>
<tr>
<td>0158</td>
<td>3AFF20</td>
<td>431</td>
<td>PUSH PSW</td>
</tr>
<tr>
<td>015B</td>
<td>E63F</td>
<td>432</td>
<td>LDA USCSR; GET USER IMAGE OF WHAT'S IN CSR</td>
</tr>
<tr>
<td>015D</td>
<td>F640</td>
<td>433</td>
<td>ANI 3FH; CLEAR 2 HIGH ORDER BITS</td>
</tr>
<tr>
<td>015F</td>
<td>D320</td>
<td>434</td>
<td>ORI 40H; SET TIMER STOP BIT</td>
</tr>
<tr>
<td>0161</td>
<td>F1</td>
<td>435</td>
<td>OUT CSR; STOP TIMER</td>
</tr>
<tr>
<td>0162</td>
<td>22F220</td>
<td>436</td>
<td>FCP PSW; RETRIEVE PSW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>437</td>
<td>SHLD LSAV; SAVE H &amp; L</td>
</tr>
</tbody>
</table>
LOC OBJ SEQ SOURCE STATEMENT
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 PSAV ; 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 SSAY ; SAVE USER STACK POINTER
0175 21ED20 446 LXI H,SSAV+1 ; SET MONITOR STACK POINTER FOR
0178 F9 447  SPLH ;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 0TH ; 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 3EOE 455  MVI A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE
0187 30 456  SIM
0188 C3FD00 457  JMP SSTEP ; GO GET READY FOR ANOTHER INSTRUCTION

FUNCTION: SUBST - SUBSTITUTE MEMORY
INPUTS: NONE
OUTPUTS: NONE
CALLS: CLEAR,GTHEX,UPDAD,UPDDT,ERR
DESTROYS: A,B,C,D,E,H,L,F/F'S

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?
0197 22FD20 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
019C FB11 477  CPI COMMA ; WAS ',' THE LAST Character FROM KEYBOARD?
019E C2C001 478  JNZ SUB10 ; NO - GO TERMINATE THE COMMAND
01A1 0600 479  MVI B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
01A3 CD5F03 480  CALL UPAD ; 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 32F620 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)
01B8 FALSE SUB10 ; NO - LEAVE DATA UNCHANGED AT CURRENT ADDRESS
01BB D2C401 490+  JNC SUB10
01BE 2AF620 491  LHLD CURAD ; YES - GET CURRENT ADDRESS IN H & L
01BF 73 492  MOV M,E ; STORE NEW DATA AT CURRENT ADDRESS
LOC OBJ  SEQ  SOURCE STATEMENT

493  ; MAKE SURE DATA WAS ACTUALLY STORED IN CASE
494  ; CURRENT ADDRESS IS IN ROM OR IS NON-EXISTANT
01EF 7B  495  MOV  A,E  ; DATA TO A FOR COMPARISON
01C0 BE  496  CMP  M  ; WAS DATA STORED CORRECTLY?
01C1 C21502  497  JNZ  ERR  ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
498  SUB10:
01C4 2AF620  499  LHLD  CURAD  ; INCREMENT CURRENT ADDRESS
01C7 23  500  INX  H
01C8 22F620  501  SHLD  CURAD
01CB F1  502  POP  PSW  ; RETRIEVE LAST CHARACTER
01CC C39C01  503  JMP  SUB05
504  SUB15:
01CF FE10  505  CPI  PERIO  ; WAS LAST CHARACTER ',' ?
01D1 C21502  506  JNZ  ERR  ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
01D4 C38901  507  JMP  CLDIS  ; YES - CLEAR DISPLAY AND TERMINATE COMMAND
508  ;
509  ;
510  ;
511  :
512  ;
513  ;
514  ;
515  ;
516  ; FUNCTION: CLEAR - CLEAR THE DISPLAY
517  ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT IN ADDRESS FIELD OF DISPLAY
518  ;         - 0 MEANS NO DOT
519  ; OUTPUTS: NONE
520  ; CALLS: OUTPT
521  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
522  ; DESCRIPTION: CLEAR SENDS BLANK CHARACTERS TO BOTH THE ADDRESS FIELD
523  ; AND THE DATA FIELD OF THE DISPLAY. IF THE DOT FLAG IS
524  ; SET THEN A DOT WILL APPEAR AT THE RIGHT EDGE OF THE
525  ; ADDRESS FIELD.
526  ;
527  CLEAR:
01D7 AF  528  XRA  A  ; ARG - USE ADDRESS FIELD OF DISPLAY
529  ;
01D8 219A03  530  LXI  H,BLANKS  ; ARG - ADDRESS OF BLANKS FOR DISPLAY
01DB CDB702  531  CALL  OUTPT  ; OUTPUT BLANKS TO ADDRESS FIELD
01DE 3EO1  532  MVI  A,DTFLD  ; ARG - USE DATA FIELD OF DISPLAY
01E0 0600  533  MVI  B,NODOT  ; ARG - NO DOT IN DATA FIELD
01E2 219A03  534  LXI  H,BLANKS  ; ARG - ADDRESS OF BLANKS FOR DISPLAY
01E5 CDB702  535  CALL  OUTPT  ; OUTPUT BLANKS TO DATA FIELD
01E6 C9  536  RET  ; RETURN
537  ;
538  ; FUNCTION: CLDIS - CLEAR DISPLAY AND TERMINATE COMMAND
539  ; INPUTS: NONE
540  ; OUTPUTS: NONE
541  ; CALLS: CLEAR
542  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
543  ; DESCRIPTION: CLDIS IS JUMPED TO BY COMMAND ROUTINES WISHING TO
544  ; TERMINATE NORMALLY. CLDIS CLEARS THE DISPLAY AND
545  ; BRANCHES TO THE COMMAND RECOGNIZER.
### ISIS-II 8080/8085 MACRO ASSEMBLER, x108

#### LOC OBJ  SEQ  SOURCE STATEMENT

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>548</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>549</td>
<td>CLDIS:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01E9 0600</td>
<td>550</td>
<td>MVI</td>
<td>B,NODOT ; ARG - NO DOT IN ADDRESS FIELD</td>
</tr>
<tr>
<td>01EB CDD701</td>
<td>551</td>
<td>CALL</td>
<td>CLEAR ; CLEAR THE DISPLAY</td>
</tr>
<tr>
<td>01EE C36600</td>
<td>552</td>
<td>JMP</td>
<td>CMMND ; GO GET ANOTHER COMMAND</td>
</tr>
<tr>
<td>553</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>554</td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>555</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>556</td>
<td>FUNCTION: CLDST - COLD START</td>
<td></td>
<td></td>
</tr>
<tr>
<td>557</td>
<td>INPUTS: NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>558</td>
<td>OUTPUTS: NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>559</td>
<td>CALLS: NOTHING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>560</td>
<td>DESTROYS: A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>561</td>
<td>DESCRIPTION: CLDST IS JUMPED TO BY THE MAIN COLD START PROCEDURE, COMPLETES COLD START INITIALIZATION, AND JUMPS BACK TO THE MAIN COLD START PROCEDURE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>562</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>563</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>565</td>
<td>CLDST:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01F1 3ECC</td>
<td>566</td>
<td>MVI</td>
<td>A,KENIT ; GET CONTROL CHARACTER</td>
</tr>
<tr>
<td>01F3 320019</td>
<td>567</td>
<td>STA</td>
<td>CNTRL ; INITIALIZE KEYBOARD/DISPLAY BLANKING</td>
</tr>
<tr>
<td>01F6 3EO0</td>
<td>568</td>
<td>MVI</td>
<td>A,CSNIT ; INITIAL VALUE OF COMMAND STATUS REGISTER</td>
</tr>
<tr>
<td>01FA D320</td>
<td>569</td>
<td>OUT</td>
<td>CSR ; INITIALIZE CSR</td>
</tr>
<tr>
<td>01FB 32FF20</td>
<td>570</td>
<td>STA</td>
<td>USCSR ; INITIALIZE USER CSR VALUE</td>
</tr>
<tr>
<td>01FD C30800</td>
<td>571</td>
<td>JMP</td>
<td>CLDBK ; BACK TO MAIN PROCEDURE</td>
</tr>
<tr>
<td>572</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>573</td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>574</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>575</td>
<td>FUNCTION: DISPC - DISPLAY PROGRAM COUNTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>576</td>
<td>INPUTS: NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>577</td>
<td>OUTPUTS: NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>578</td>
<td>CALLS: UPDAD,UPDDT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>579</td>
<td>DESTROYS: A,B,C,D,E,H,L,F/F'S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>581</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>582</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>583</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>584</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>585</td>
<td>DISPC:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0200 2AF220</td>
<td>586</td>
<td>LHLD</td>
<td>PSAV ; GET USER PROGRAM COUNTER</td>
</tr>
<tr>
<td>0203 22F620</td>
<td>587</td>
<td>SHLD</td>
<td>CURAD ; MAKE IT THE CURRENT ADDRESS</td>
</tr>
<tr>
<td>0206 7E</td>
<td>588</td>
<td>MOV</td>
<td>A,M ; GET THE INSTRUCTION AT THAT ADDRESS</td>
</tr>
<tr>
<td>0207 32F820</td>
<td>589</td>
<td>STA</td>
<td>CURDT ; MAKE IT THE CURRENT DATA</td>
</tr>
<tr>
<td>020A 0601</td>
<td>590</td>
<td>MVI</td>
<td>B,DOT ; ARG - DOT IN ADDRESS FIELD</td>
</tr>
<tr>
<td>020C CD5F03</td>
<td>591</td>
<td>CALL</td>
<td>UPDAD ; UPDATE ADDRESS FIELD OF DISPLAY</td>
</tr>
<tr>
<td>020F 0600</td>
<td>592</td>
<td>MVI</td>
<td>B,NODOT ; ARG - NO DOT IN DATA FIELD</td>
</tr>
<tr>
<td>0211 CD6B03</td>
<td>593</td>
<td>CALL</td>
<td>UPDDT ; UPDATE DATA FIELD OF DISPLAY</td>
</tr>
<tr>
<td>0214 C9</td>
<td>594</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>595</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>596</td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>597</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>598</td>
<td>FUNCTION: ERR - DISPLAY ERROR MESSAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599</td>
<td>INPUTS: NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>OUTPUTS: NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>601</td>
<td>CALLS: OUTPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>602</td>
<td>DESTROYS: A,B,C,D,E,H,L,F/F'S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOC   OBJ   SEQ   SOURCE STATEMENT

603 ; DESCRIPTION: ERR IS JUMPED TO BY COMMAND ROUTINES WISHING TO
604 ; TERMINATE BECAUSE OF AN ERROR.
605 ; ERR OUTPUTS AN ERROR MESSAGE TO THE DISPLAY AND
606 ; BRANCHES TO THE COMMAND RECOGNIZER.
607 ;
608 ERR:

0215 AF   609   XRA   A       ; ARG - USE ADDRESS FIELD
0216 0600  610   MVI   B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0218 219803 611   LXI   H,ERMSG ; ARG - ADDRESS OF ERROR MESSAGE
021B CDB702 612   CALL   OUTPT ; OUTPUT ERROR MESSAGE TO ADDRESS FIELD
021E 3E01   613   MVI   A,DTPLD ; ARG - USE DATA FIELD
0220 0600   614   MVI   B,NODOT ; ARG - NO DOT IN DATA FIELD
0222 219A03 615   LXI   H,BLNKS ; ARG - ADDRESS OF BLANKS FOR DISPLAY
0225 CDB702 616   CALL   OUTPT ; OUTPUT BLANKS TO DATA FIELD
0228 C36600 617   JMP   CMMND ; GO GET A NEW COMMAND

618 ;
619 ;
620 ;
621 ; FUNCTION: GTHEX - GET HEX DIGITS
622 ; INPUTS: B - DISPLAY FLAG - 0 MEANS USE ADDRESS FIELD OF DISPLAY
623 ;         - 1 MEANS USE DATA FIELD OF DISPLAY
624 ; OUTPUTS: A - LAST CHARACTER READ FROM KEYBOARD
625 ; DE - HEX DIGITS FROM KEYBOARD EVALUATED MODULO 2**16
626 ; CARRY - SET IF AT LEAST ONE VALID HEX DIGIT WAS READ
627 ;            - RESET OTHERWISE
628 ; CALLS: RDKBD, INSDG, HXDSP, OUTPT
629 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
630 ; DESCRIPTION: GTHEX ACCEPTS A STRING OF HEX DIGITS FROM THE KEYBOARD,
631 ; DISPLAYS THEM AS THEY ARE RECEIVED, AND RETURNS THEIR
632 ; VALUE AS A 16 BIT INTEGER. IF MORE THAN 4 HEX DIGITS
633 ; ARE RECEIVED, ONLY THE LAST 4 ARE USED. IF THE DISPLAY
634 ; IS SET, THE LAST 2 HEX DIGITS ARE DISPLAYED IN THE
635 ; DATA FIELD OF THE DISPLAY, OTHERWISE, THE LAST 4 HEX
636 ; DIGITS ARE DISPLAYED IN THE ADDRESS FIELD OF THE
637 ; DISPLAY. IN EITHER CASE, A DOT WILL BE DISPLAYED AT THE
638 ; RIGHTMOST EDGE OF THE FIELD. A CHARACTER WHICH IS NOT
639 ; A HEX DIGIT TERMINATES THE STRING AND IS RETURNED AS
640 ; AN OUTPUT OF THE FUNCTION. IF THE TERMINATOR IS NOT
641 ; A PERIOD OR A COMMA THEN ANY HEX DIGITS WHICH MAY HAVE
642 ; BEEN RECEIVED ARE CONSIDERED TO BE INVALID. THE
643 ; FUNCTION RETURNS A FLAG INDICATING WHETHER OR NOT ANY
644 ; VALID HEX DIGITS WERE RECEIVED.
645 ;
646 GTHEX:

0226 0E00   647   MVI   C,0 ; RESET HEX DIGIT FLAG
022D C5     648   PUSH  B ; SAVE DISPLAY AND HEX DIGIT FLAGS
022E 110000 649   LXI   D,0 ; SET HEX VALUE TO ZERO
0231 D5     650   PUSH  D ; SAVE HEX VALUE

0232 CDE702 651   CALL   RDKBD ; READ KEYBOARD
0235 FE10   652   CPI   10H ; IS CHARACTER A HEX DIGIT?
0237 D25502 653   JNC   GTH05 ; NO - GO CHECK FOR TERMINATOR
0238 D1     654   POP   D ; ARG - RETRIEVE HEX VALUE
023B CD9F02 655   CALL   INCDG ; INSERT NEW DIGIT IN HEX VALUE
GTH10:
GTH20:
GTH25:

SOURCE STATEMENT

LOC OBJ SEQ SOURCE STATEMENT
023E C1 658 POP B ; RETRIEVE DISPLAY FLAG
023F 0E01 659 MVI C,1 ; SET HEX DIGIT FLAG
0240 0E01 660 ; (I.E. A HEX DIGIT HAS BEEN READ)
0241 C5 661 PUSH B ; SAVE DISPLAY AND HEX DIGIT FLAGS
0242 D5 662 PUSH D ; SAVE HEX VALUE
0243 78 663 MOV A,B ; TEST DISPLAY FLAG
0244 0F 664 RRC ; SHOULD ADDRESS FIELD OF DISPLAY BE USED ?
0245 D24902 665 JNC GTH10 ; YES - USE HEX VALUE AS IS
0246 0E01 666 ; NO - ONLY LOW ORDER BYTE OF HEX VALUE SHOULD
0247 0F 667 ; BE USED FOR DATA FIELD OF DISPLAY
0248 53 668 MOV D,E ; PUT LOW ORDER BYTE OF HEX VALUE IN D
0249 GTH10:
024A CD6C02 669 MOV D,E ; ARG - HEX VALUE TO BE EXPANDED IS IN D & E
024B 0E01 670 CALL HXDSP ; EXPAND HEX VALUE FOR DISPLAY
024C 78 671 ; ARG - ADDRESS OF EXPANDED HEX VALUE IN H & L
024D 0E01 672 MOV A,B ; ARG - PUT DISPLAY FLAG IN A
024E 0E01 673 MOV B,DOT ; ARG - DOT IN APPROPRIATE FIELD
024F DB7F02 674 CALL OUTPUT ; OUTPUT HEX VALUE TO DISPLAY
0250 C33202 675 JMP GTH25 ; GO GET NEXT CHARACTER
0251 0E01 676 ; LAST CHARACTER WAS NOT A HEX DIGIT
0252 D1 677 POP D ; RETRIEVE HEX VALUE
0253 C1 678 POP B ; RETRIEVE HEX DIGIT FLAG IN C
0254 FE11 679 MOV A,B ; ARG - ADDRESS OF EXPANDED HEX VALUE IN H & L
0255 CA6702 680 MOV A,B ; ARG - PUT DISPLAY FLAG IN A
0256 FE10 681 JMP GTH25 ; YES - READY TO RETURN
0257 FE10 682 CPI PERIO ; NO - WAS LAST CHARACTER ',' ?
0258 CA6702 683 JMP GTH25 ; YES - READY TO RETURN
0259 0E01 684 ; NO - INVALID TERMINATOR - IGNORE ANY HEX DIGITS READ
025A C3F702 685 LXI D,0 ; SET HEX VALUE TO ZERO
025B C3F702 686 JMP RETF ; RETURN FALSE
025C 0F 687 GTH25:
025D 47 688 MOV A,B,C ; SAVE LAST CHARACTER
025E 79 689 MOV A,C ; SHIFT HEX DIGIT FLAG TO
025F 0F 690 RRC ;/CARRY BIT
0260 78 691 MOV A,B ; RESTORE LAST CHARACTER
0261 C9 692 RET ; RETURN
0262 7A 693 ;
0263 0F 694 ;******************************************************************************
0264 0F 695 ;
0265 0F 696 ; FUNCTION: HXDSP - EXPAND HEX DIGITS FOR DISPLAY
0266 0F 697 ; INPUTS: DE - 4 HEX DIGITS
0267 0F 698 ; OUTPUTS: HL - ADDRESS OF OUTPUT BUFFER
0268 0F 699 ; CALLS: NOTHING
0269 0F 700 ; DESTROYS: A,H,L,F/F'S
026A 0F 701 ; DESCRIPTION: HXDSP EXPANDS EACH INPUT BYTE TO 2 BYTES IN A FORM
026B 0F 702 ; SUITABLE FOR DISPLAY BY THE OUTPUT ROUTINES. EACH INPUT
026C 0F 703 ; BYTE IS DIVIDED INTO 2 HEX DIGITS. EACH HEX DIGIT IS
026D 0F 704 ; PLACED IN THE LOW ORDER 4 BITS OF A BYTE WHOSE HIGH
026E 0F 705 ; ORDER 4 BITS ARE SET TO ZERO. THE RESULTING BYTE IS
026F 0F 706 ; STORED IN THE OUTPUT BUFFER. THE FUNCTION RETURNS THE
0270 0F 707 ; ADDRESS OF THE OUTPUT BUFFER.
0271 0F 708 ;
0272 0F 709 HXDSP:
0273 7A 710 MOV A,D ; GET FIRST DATA BYTE
0274 0F 711 RRC ; CONVERT 4 HIGH ORDER BITS
0275 0F 712 RRC ; /TO A SINGLE CHARACTER
**Source Statement**

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>026F</td>
<td>0F</td>
<td>713</td>
<td>RRC</td>
</tr>
<tr>
<td>0270</td>
<td>0F</td>
<td>714</td>
<td>RRC</td>
</tr>
<tr>
<td>0271</td>
<td>E60F</td>
<td>715</td>
<td>ANI OFH ; GET ADDRESS OF OUTPUT BUFFER</td>
</tr>
<tr>
<td>0273</td>
<td>21F920</td>
<td>716</td>
<td>LXI H, OBUF ; STORE CHARACTER IN OUTPUT BUFFER</td>
</tr>
<tr>
<td>0276</td>
<td>77</td>
<td>717</td>
<td>MOV M, A ; GET FIRST DATA BYTE AND CONVERT 4 LOW ORDER</td>
</tr>
<tr>
<td>0277</td>
<td>7A</td>
<td>718</td>
<td>MOV A, D ; NEXT BUFFER POSITION</td>
</tr>
<tr>
<td>0278</td>
<td>E60F</td>
<td>719</td>
<td>ANI OFH ; /BITS TO A SINGLE CHARACTER</td>
</tr>
<tr>
<td>027A</td>
<td>23</td>
<td>720</td>
<td>INX H ; NEXT BUFFER POSITION</td>
</tr>
<tr>
<td>027B</td>
<td>77</td>
<td>721</td>
<td>MOV M, A ; STORE CHARACTER IN BUFFER</td>
</tr>
<tr>
<td>027C</td>
<td>7B</td>
<td>722</td>
<td>MOV A, E ; GET SECOND DATA BYTE</td>
</tr>
<tr>
<td>027D</td>
<td>OF</td>
<td>723</td>
<td>RRC ; CONVERT 4 HIGH ORDER BITS</td>
</tr>
<tr>
<td>027E</td>
<td>OF</td>
<td>724</td>
<td>RRC ; TO A SINGLE CHARACTER</td>
</tr>
<tr>
<td>027F</td>
<td>OF</td>
<td>725</td>
<td>RRC</td>
</tr>
<tr>
<td>0280</td>
<td>OF</td>
<td>726</td>
<td>RRC</td>
</tr>
<tr>
<td>0281</td>
<td>E60F</td>
<td>727</td>
<td>ANI OFH ; NEXT BUFFER POSITION</td>
</tr>
<tr>
<td>0283</td>
<td>23</td>
<td>728</td>
<td>INX H ; STORE CHARACTER IN BUFFER</td>
</tr>
<tr>
<td>0284</td>
<td>77</td>
<td>729</td>
<td>MOV M, A ; STORE CHARACTER IN BUFFER</td>
</tr>
<tr>
<td>0285</td>
<td>7B</td>
<td>730</td>
<td>MOV A, E ; GET SECOND DATA BYTE</td>
</tr>
<tr>
<td>0286</td>
<td>E60F</td>
<td>731</td>
<td>ANI OFH ; /BITS TO A SINGLE CHARACTER</td>
</tr>
<tr>
<td>0288</td>
<td>23</td>
<td>732</td>
<td>INX H ; NEXT BUFFER POSITION</td>
</tr>
<tr>
<td>0289</td>
<td>77</td>
<td>733</td>
<td>MOV M, A ; STORE CHARACTER IN BUFFER</td>
</tr>
<tr>
<td>028A</td>
<td>21F920</td>
<td>734</td>
<td>LXI H, OBUF ; RETURN ADDRESS OF OUTPUT BUFFER IN H &amp; L</td>
</tr>
</tbody>
</table>

**Function: ININ1 - 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.

**Function: INSDG - INSERT HEX DIGIT**
IISIS-II 8080/8085 MACRO ASSEMBLER, X108

LOC OBJ

SEQ

SOURCE STATEMENT

768 ; INPUTS: A - HEX DIGIT TO BE INSERTED
769 ; DE - HEX VALUE
770 ; OUTPUTS: DE - HEX VALUE WITH DIGIT INSERTED
771 ; CALLS: NOTHING
772 ; DESTROYS: A,F/F'S
773 ; DESCRIPTION: INSDG SHIFTS THE CONTENTS OF D & E LEFT 4 BITS
774 ; (1 HEX DIGIT) AND INSERTS THE HEX DIGIT IN A IN THE LOW
775 ; ORDER DIGIT POSITION OF THE RESULT. A IS ASSUMED TO
776 ; CONTAIN A SINGLE HEX DIGIT IN THE LOW ORDER 4 BITS AND
777 ; ZEROS IN THE HIGH ORDER 4 BITS.
778 ;
779 INSDG:

029F EB
02A0 29
02A1 29
02A2 29
02A3 29
02A4 6F
02A5 6F
02A6 EB
02A7 C9

029F EB
02A0 29
02A1 29
02A2 29
02A3 29
02A4 65
02A5 6F
02A6 EB
02A7 C9

780 XCHG ; PUT D & E IN H & L
781 DAD H ; SHIFT H & L LEFT 4 BITS
782 DAD H
783 DAD H
784 DAD H
785 ADD L ; INSERT LOW ORDER DIGIT
786 MOV L,A
787 XCHG ; PUT H & L BACK IN D & E
788 RET
789 ;**********************************************************************
790 ;FUNCTION: NXTRG - ADVANCE REGISTER POINTER TO NEXT REGISTER
791 ; INPUTS: NONE
792 ; OUTPUTS: CARRY - 1 IF POINTER IS ADVANCED SUCCESSFULLY
793 ; 0 OTHERWISE
794 ; CALLS: NOTHING
795 ; DESTROYS: A,F/F'S
796 ; DESCRIPTION: IF THE REGISTER POINTER POINTS TO THE LAST REGISTER IN
797 ; THE EXAMINE REGISTER SEQUENCE, THE POINTER IS NOT
798 ; CHANGED AND THE FUNCTION RETURNS FALSE. IF THE REGISTER
799 ; POINTER DOES NOT POINT TO THE LAST REGISTER THEN THE
800 ; POINTER IS ADVANCED TO THE NEXT REGISTER IN THE SEQUENCE
801 ; AND THE FUNCTION RETURNS TRUE.
802 ;
803 ;NXTRG:
804 ;**********************************************************************
805 NXTRG:

02A8 3AFD20
02AB FE0C
02AD D2F702
02B0 3C
02B1 32FD20
02B4 C3FA02

02A8 3AFD20
02AB FE0C
02AD D2F702
02B0 3C
02B1 32FD20
02B4 C3FA02

806 LDA RGPT ; GET REGISTER POINTER
807 CPI NUMRG-1 ; DOES POINTER POINT TO LAST REGISTER?
808 JNC RETF ; YES - UNABLE TO ADVANCE REGISTER - RETURN FALSE
809 INR A ; NO - ADVANCE REGISTER POINTER
810 STA RGPT ; SAVE REGISTER POINTER
811 JMP RETT ; RETURN TRUE
812 ;**********************************************************************
813 ;FUNCTION: OUTPT - OUTPUT CHARACTERS TO DISPLAY
814 ; INPUTS: A - DISPLAY FLAG - 0 = USE ADDRESS FIELD
815 ; 1 = USE DATA FIELD
816 ; B - DOT FLAG - 1 = OUTPUT DOT AT RIGHT EDGE OF FIELD
817 ; 0 = NO DOT
818 ; HL - ADDRESS OF CHARACTERS TO BE OUTPUT
819 ; CALLS: NOTHING
820 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
821 ;
LOC   OBJ   SEQ   SOURCE STATEMENT

823 ; DESCRIPTION: OUTPT SENDS CHARACTERS TO THE DISPLAY. THE ADDRESS
824 ; OF THE CHARACTERS IS RECEIVED AS AN ARGUMENT. EITHER
825 ; 2 CHARACTERS ARE SENT TO THE DATA FIELD, OR 4 CHARACTERS
826 ; ARE SENT TO THE ADDRESS FIELD, DEPENDING ON THE
827 ; DISPLAY FLAG ARGUMENT. THE DOT FLAG ARGUMENT DETERMINES
828 ; WHETHER OR NOT A DOT (DECIMAL POINT) WILL BE SENT
829 ; ALONG WITH THE LAST OUTPUT CHARACTER.
830 ;
831 ;
832 ;
833 ;
834 ;
835 ;
836 ;
837 ;
838 ;
839 ;
840 ;
841 ;
842 ;
843 ;
844 ;
845 ;
846 ;
847 ;
848 ;
849 ;
850 ;
851 ;
852 ;
853 ;
854 ;
855 ;
856 ;
857 ;
858 ;
859 ;
860 ;
861 ;
862 ;
863 ;
864 ;
865 ;
866 ;
867 ;
868 ;
869 ;
870 ;
871 ;
872 ;
873 ;
874 ;
875 ;
876 ;
877 ;

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
### LOC OBJ SEQ SOURCE STATEMENT

<table>
<thead>
<tr>
<th>02E7 21FE20</th>
<th>882</th>
<th>RDKBD:</th>
</tr>
</thead>
<tbody>
<tr>
<td>883</td>
<td>883</td>
<td>LIH,IBUFF</td>
</tr>
<tr>
<td>884</td>
<td>884</td>
<td>MOV A,M</td>
</tr>
<tr>
<td>885</td>
<td>885</td>
<td>JPE</td>
</tr>
<tr>
<td>886</td>
<td>886</td>
<td>ORA A</td>
</tr>
<tr>
<td>887</td>
<td>887</td>
<td>JP RDK10</td>
</tr>
<tr>
<td>888</td>
<td>888</td>
<td>EI</td>
</tr>
<tr>
<td>889</td>
<td>889</td>
<td>JMP RDKBD</td>
</tr>
<tr>
<td>890</td>
<td>RDK10:</td>
<td></td>
</tr>
<tr>
<td>891</td>
<td>891</td>
<td>MVI M,EMPTY</td>
</tr>
<tr>
<td>892</td>
<td>892</td>
<td>DI</td>
</tr>
<tr>
<td>893</td>
<td>893</td>
<td>RET</td>
</tr>
<tr>
<td>894</td>
<td>894</td>
<td></td>
</tr>
<tr>
<td>895</td>
<td>895</td>
<td>;******************************************************</td>
</tr>
<tr>
<td>896</td>
<td>896</td>
<td>;</td>
</tr>
<tr>
<td>897</td>
<td>897</td>
<td>; FUNCTION: RETF - RETURN FALSE</td>
</tr>
<tr>
<td>898</td>
<td>898</td>
<td>; INPUTS: NONE</td>
</tr>
<tr>
<td>899</td>
<td>899</td>
<td>; OUTPUTS: CARRY = 0 (FALSE)</td>
</tr>
<tr>
<td>900</td>
<td>900</td>
<td>; CALLS: NOTHING</td>
</tr>
<tr>
<td>901</td>
<td>901</td>
<td>; DESTROYS: CARRY</td>
</tr>
<tr>
<td>902</td>
<td>902</td>
<td>; DESCRIPTION: RETF IS JUMPED TO BY FUNCTIONS WISHING TO RETURN FALSE.</td>
</tr>
<tr>
<td>903</td>
<td>903</td>
<td>; RETF SETS CARRY TO 0 AND RETURNS TO THE CALLER OF</td>
</tr>
<tr>
<td>904</td>
<td>904</td>
<td>; THE ROUTINE INVOKING RETF.</td>
</tr>
<tr>
<td>905</td>
<td>905</td>
<td>;</td>
</tr>
<tr>
<td>906</td>
<td>906</td>
<td>; RETF:</td>
</tr>
<tr>
<td>907</td>
<td>907</td>
<td>STC</td>
</tr>
<tr>
<td>908</td>
<td>908</td>
<td>CMC</td>
</tr>
<tr>
<td>909</td>
<td>909</td>
<td>RET</td>
</tr>
<tr>
<td>910</td>
<td>910</td>
<td>;******************************************************</td>
</tr>
<tr>
<td>911</td>
<td>911</td>
<td>;</td>
</tr>
<tr>
<td>912</td>
<td>912</td>
<td>; FUNCTION: RETT - RETURN TRUE</td>
</tr>
<tr>
<td>913</td>
<td>913</td>
<td>; INPUTS: NONE</td>
</tr>
<tr>
<td>914</td>
<td>914</td>
<td>; OUTPUTS: CARRY = 1 (TRUE)</td>
</tr>
<tr>
<td>915</td>
<td>915</td>
<td>; CALLS: NOTHING</td>
</tr>
<tr>
<td>916</td>
<td>916</td>
<td>; DESTROYS: CARRY</td>
</tr>
<tr>
<td>917</td>
<td>917</td>
<td>; DESCRIPTION: RETT IS JUMPED TO BY ROUTINES WISHING TO RETURN TRUE.</td>
</tr>
<tr>
<td>918</td>
<td>918</td>
<td>; RETT SETS CARRY TO 1 AND RETURNS TO THE CALLER OF</td>
</tr>
<tr>
<td>919</td>
<td>919</td>
<td>; THE ROUTINE INVOKING RETT.</td>
</tr>
<tr>
<td>920</td>
<td>920</td>
<td>;</td>
</tr>
<tr>
<td>921</td>
<td>921</td>
<td>; RETT:</td>
</tr>
<tr>
<td>922</td>
<td>922</td>
<td>STC</td>
</tr>
<tr>
<td>923</td>
<td>923</td>
<td>RET</td>
</tr>
<tr>
<td>924</td>
<td>924</td>
<td>;******************************************************</td>
</tr>
<tr>
<td>925</td>
<td>925</td>
<td>;</td>
</tr>
<tr>
<td>926</td>
<td>926</td>
<td>; FUNCTION: RGLOC - GET REGISTER SAVE LOCATION</td>
</tr>
<tr>
<td>927</td>
<td>927</td>
<td>; INPUTS: NONE</td>
</tr>
<tr>
<td>928</td>
<td>928</td>
<td>; OUTPUTS: HL - REGISTER SAVE LOCATION</td>
</tr>
<tr>
<td>929</td>
<td>929</td>
<td>; CALLS: NOTHING</td>
</tr>
<tr>
<td>930</td>
<td>930</td>
<td>; DESTROYS: B,C,H,L,F/F'S</td>
</tr>
<tr>
<td>931</td>
<td>931</td>
<td>;</td>
</tr>
<tr>
<td>932</td>
<td>932</td>
<td>;</td>
</tr>
</tbody>
</table>
LOC  OBJ  SEQ  SOURCE STATEMENT

933 ; DESCRIPTION: RGLOC RETURNS THE SAVE LOCATION OF THE REGISTER
934 ; INDICATED BY THE CURRENT REGISTER POINTER VALUE.
935 ;
936 RGLOC:
02FC 2AFD20 937 LHLD RGPTR ; GET REGISTER POINTER
02FF 2600 938 MVI H,0 ; /IN H & L
0301 01ED03 939 LXI B,RGTBL ; GET REGISTER SAVE LOCATION TABLE ADDRESS
0304 09 940 DAD B ; POINTER INDEXES TABLE
0305 6E 941 MOV L,M ; GET LOW ORDER BYTE OF REGISTER SAVE LOC.
0306 2620 942 MVI H,(RAMST SHR 8) ; GET HIGH ORDER BYTE OF
943 ; /REGISTER SAVE LOCATION
0308 C9 944 RET

945 ;*******************************************************************************
946;
947 ; FUNCTION: RGNAME - DISPLAY REGISTER NAME
948 ; INPUTS: NONE
949 ; OUTPUTS: NONE
950 ; CALLS: OUTPT
951 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
952 ; DESCRIPTION: RGNAME DISPLAYS, IN THE ADDRESS FIELD OF THE DISPLAY,
953 ; THE REGISTER NAME CORRESPONDING TO THE CURRENT
954 ; REGISTER POINTER VALUE.
955 ;
956 RGNAME:
0309 2AFD20 957 LHLD RGPTR ; GET REGISTER POINTER
030C 2600 958 MVI H,0
030E 29 959 DAD H ; MULTIPLY POINTER VALUE BY 4
030F 29 960 DAD H ; /(REGISTER NAME TABLE HAS 4 BYTE ENTRIES)
0310 01B903 961 LXI B,NMTBL ; GET ADDRESS OF START OF REGISTER NAME TABLE
0313 09 962 DAD B ; ARG - ADD TABLE ADDRESS TO POINTER - RESULT IS
963 ; ADDRESS OF APPROPRIATE REGISTER NAME IN H & L
0314 AF 964 XRA A ; ARG - USE ADDRESS FIELD OF DISPLAY
0315 0600 965 MVI B,NODOT ; ARG - NO DOT IN ADDRESS FIELD
0317 CDB702 966 CALL OUTPT ; OUTPUT REGISTER NAME TO ADDRESS FIELD
031A C9 967 RET

968 ;*******************************************************************************
970;
971 ; FUNCTION: RSTOR - RESTOR USER REGISTERS
972 ; INPUTS: NONE
973 ; OUTPUTS: NONE
974 ; CALLS: NOTHING
975 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
976 ; DESCRIPTION: RSTOR RESTORES ALL CPU REGISTERS, FLIP/FLOPS,
977 ; INTERRUPT STATUS, INTERRUPT MASK, STACK POINTER
978 ; AND PROGRAM COUNTER FROM THEIR RESPECTIVE
979 ; SAVE LOCATIONS IN MEMORY. BY RESTORING THE PROGRAM
980 ; COUNTER, THE ROUTINE EFFECTIVELY TRANSFERS CONTROL TO
981 ; THE ADDRESS IN THE PROGRAM COUNTER SAVE LOCATION.
982 ;
983 ; THE TIMING OF THIS ROUTINE IS CRITICAL TO THE
984 ; CORRECT OPERATION OF THE SINGLE STEP ROUTINE.
985 ; IF ANY MODIFICATION CHANGES THE NUMBER OF CPU
986 ; STATED NECESSARY TO EXECUTE THIS ROUTINE THEN THE
SOURCE STATEMENT

TIMER VALUE MUST BE ADJUSTED BY THE SAME NUMBER.

THIS IS ALSO THE ENTRY POINT FOR THE TTY MONITOR

TO RESTORE REGISTERS.

RSTOR:

LDA ISAV ; GET USER INTERRUPT MASK
ORI 18H ; ENABLE SETTING OF INTERRUPT MASK AND
; /RESET IV3C FLIP FLOP
SIM ; RESTORE USER INTERRUPT MASK
RESTORE USER INTERRUPT STATUS
LDA ISAV ; GET USER INTERRUPT MASK
ANI 08H ; SHOULD USER INTERRUPTS BE ENABLED?
JZ RSR05 ; NO - LEAVE INTERRUPTS DISABLED
E1 ; YES - ENABLE INTERRUPTS FOR USER PROGRAM
JMP RSR10

STC ; DUMMY INSTRUCTIONS - WHEN SINGLE STEP ROUTINE
JNC RSR10 ; /IS BEING USED, THE TIMER IS RUNNING AND
; /EXECUTE TIME FOR THIS ROUTINE MUST NOT
; VARY.

Rsr05:

STC ; DUMMY INSTRUCTIONS - WHEN SINGLE STEP ROUTINE
JNC RSR10 ; /IS BEING USED, THE TIMER IS RUNNING AND
; /EXECUTE TIME FOR THIS ROUTINE MUST NOT
; VARY.

H,MNSTK SET MONITOR STACK POINTER TO START OF STACK
/WHICH IS ALSO END OF REGISTER SAVE AREA
POP D ; RESTORE REGISTERS
POP B
POP PSW
LHLD Ssav ; RESTORE USER STACK POINTER
LHLD PSAV
LHLD LSav ; RESTORE H & L REGISTERS
LHLD LSav ; RESTORE H & L REGISTERS
RET ; JUMP TO USER PROGRAM COUNTER

FUNCTION: SETRG - SET REGISTER POINTER
PUTS: 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'.

CALL RDKBD ; READ FROM KEYBOARD
IS CHARACTER A DIGIT?
; CHARACTER IS NOT A
FUNCTION RETURNS 'FALSE'. OTHERWISE, THE
; REGISTER DESIGNATOR
; TRY TO CONVERT REGISTER DESIGNATOR TO
; INDEX INTO REGISTER POINTER TABLE
LOC OBJ SEQ SOURCE STATEMENT

034E DAF702 1043 ; WAS CONVERSION SUCCESSFUL?
0351 4F 1044 JC RETF ; NO - RETURN FALSE
0352 0600 1045 MOV C,A ; INDEX TO B & C
0354 21AC03 1046 MVI B,0 ;
0357 09 1047 LXI H,RGTPB ; GET ADDRESS OF REGISTER POINTER TABLE
0358 7E 1048 DAD B ; INDEX POINTS INTO TABLE
0359 32FD20 1049 MOV A,M ; GET REGISTER POINTER FROM TABLE
035C C3FA02 1050 STA RGPTR ; SAVE REGISTER POINTER
035D 0357 1051 JMP RETT ; RETURN TRUE

052 ;
053 ;
054 ;
055 ; FUNCTION: UPDAD - UPDATE ADDRESS FIELD OF DISPLAY
056 ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT AT RIGHT EDGE OF FIELD
057 ; 0 MEANS NO DOT
058 ; OUTPUTS: NONE
059 ; CALLS: HXDSP,OUTPT
060 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
061 ; DESCRIPTION: UPDAD UPDATES THE ADDRESS FIELD OF THE DISPLAY USING
062 ; THE CURRENT ADDRESS.
063 ;
064 UPDAD:
065 035F 2AF620 1065 LHLD CURAD ; GET CURRENT ADDRESS
066 0362 EB 1066 XCHG ; ARG - PUT CURRENT ADDRESS IN D & E
067 0363 CD6C02 1067 CALL HXDSP ; EXPAND CURRENT ADDRESS FOR DISPLAY
068 0366 AF 1068 ; ARG - ADDRESS OF EXPANDED ADDRESS IS IN H & L
069 0367 CDE7C2 1069 XRA A ; ARG - USE ADDRESS FIELD OF DISPLAY
070 036A C9 1070 ; ARG - DOT FLAG IS IN B
071 0371 1071 CALL OUTPT ; OUTPUT CURRENT ADDRESS TO ADDRESS FIELD
072 0372 1072 RET

073 ;
074 ; FUNCTION: UPDDT - UPDATE DATA FIELD OF DISPLAY
075 ; INPUTS: B - DOT FLAG - 1 MEANS PUT DOT AT RIGHT EDGE OF FIELD
076 ; 0 MEANS NO DOT
077 ; OUTPUTS: NONE
078 ; CALLS: HXDSP,OUTDT
079 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
080 ; DESCRIPTION: UPDDT UPDATES THE DATA FIELD OF THE DISPLAY USING
081 ; THE CURRENT DATA BYTE.
082 ;
083 UPDDT:
084 036B 3AF820 1086 LDA CURDT ; GET CURRENT DATA
085 036F 57 1087 MOV D,A ; ARG - PUT CURRENT DATA IN D
086 036F CD6C02 1088 CALL HXDSP ; EXPAND CURRENT DATA FOR DISPLAY
087 0372 3E01 1089 ; ARG - ADDRESS OF EXPANDED DATA IS IN H & L
088 0374 CDE7C2 1090 MVI A,DTFLD ; ARG - USE DATA FIELD OF DISPLAY
089 0377 C9 1091 ; ARG - DOT FLAG IS IN B
090 0394 1092 CALL OUTPT ; OUTPUT CURRENT DATA TO DATA FIELD
091 0395 1093 RET

096 ;
097 ; MONITOR TABLES
LOC OBJ SEQ SOURCE STATEMENT

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 FDO0 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 OF3H ; 0
0385 60 1130 DB 60H ; 1
0386 B5 1131 DB 0B5H ; 2
0387 F4 1132 DB OF4H ; 3
0388 66 1133 DB 66H ; 4
0005 1134 FIVE EQU $ - DSPTB
0005 1135 LETRS EQU $ - DSPTB
0389 D6 1136 DB OD6H ; 5 AND S
038A D7 1137 DB OD7H ; 6
038B 70 1138 DB 70H ; 7
0008 1139 EIGHT EQU $ - DSPTB
038C F7 1140 DB OF7H ; 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 LETRA EQU $ - DSPTB
0392 97 1151 DB 97H ; E
000F 1152 LETRF EQU $ - DSPTB
SOURCE STATEMENT

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0393 17</td>
<td>1153</td>
<td>DB 17H ; F</td>
</tr>
<tr>
<td>0010</td>
<td>1154</td>
<td>LETRH EQU $ - DSPTB</td>
</tr>
<tr>
<td>0394 67</td>
<td>1155</td>
<td>DB 67H ; H</td>
</tr>
<tr>
<td>0011</td>
<td>1156</td>
<td>LETRL EQU $ - DSPTB</td>
</tr>
<tr>
<td>0395 83</td>
<td>1157</td>
<td>DB 83H ; L</td>
</tr>
<tr>
<td>0012</td>
<td>1158</td>
<td>LETRP EQU $ - DSPTB</td>
</tr>
<tr>
<td>0396 37</td>
<td>1159</td>
<td>DB 37H ; P</td>
</tr>
<tr>
<td>0013</td>
<td>1160</td>
<td>LETRI EQU $ - DSPTB</td>
</tr>
<tr>
<td>0397 60</td>
<td>1161</td>
<td>DB 60H ; I</td>
</tr>
<tr>
<td>0014</td>
<td>1162</td>
<td>LETRR EQU $ - DSPTB</td>
</tr>
<tr>
<td>0398 05</td>
<td>1163</td>
<td>DB 05H ; R (LOWER CASE)</td>
</tr>
<tr>
<td>0015</td>
<td>1164</td>
<td>BLANK EQU $ - DSPTB</td>
</tr>
<tr>
<td>0399 00</td>
<td>1165</td>
<td>DB 00H ; BLANK</td>
</tr>
<tr>
<td></td>
<td>1166</td>
<td>;</td>
</tr>
</tbody>
</table>

MESSAGES FOR OUTPUT TO DISPLAY

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>039A 15</td>
<td>1171</td>
<td>BLNKS: DB BLANK,BLANK,BLANK,BLANK ; FOR ADDRESS OR DATA FIELD</td>
</tr>
<tr>
<td>039B 15</td>
<td>1172</td>
<td>ERMSG: DB BLANK,LETRE,LETTR,LETTR ; ERROR MESSAGE FOR ADDR. FIELD</td>
</tr>
<tr>
<td>039C 15</td>
<td>1173</td>
<td>EXMSG: DB LETRE,BLANK,BLANK,BLANK ; EXECUTION MESSAGE</td>
</tr>
<tr>
<td>039D 15</td>
<td>1174</td>
<td>; /FOR ADDRESS FIELD</td>
</tr>
<tr>
<td>039E 15</td>
<td>1175</td>
<td>SGNAD: DB BLANK,BLANK,EIGHT,ZERO ; SIGN ON MESSAGE (ADDR. FIELD)</td>
</tr>
<tr>
<td>03A0 14</td>
<td>1176</td>
<td>SGNDT: DB EIGHT,FIVE ; SIGN ON MESSAGE (DATA FIELD)</td>
</tr>
<tr>
<td>03A2 0E</td>
<td>1177</td>
<td>;</td>
</tr>
<tr>
<td>03A3 15</td>
<td>1178</td>
<td>;</td>
</tr>
</tbody>
</table>

REGISTER POINTER TABLE

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>03A4 15</td>
<td>1180</td>
<td>RGPTB: ; REGISTER POINTER TABLE</td>
</tr>
<tr>
<td></td>
<td>1181</td>
<td>; THE ENTRIES IN THIS TABLE ARE IN THE SAME ORDER</td>
</tr>
<tr>
<td></td>
<td>1182</td>
<td>; AS THE REGISTER DESIGNATOR KEYS ON THE KEYBOARD.</td>
</tr>
<tr>
<td></td>
<td>1183</td>
<td>; EACH ENTRY CONTAINS THE REGISTER POINTER VALUE WHICH</td>
</tr>
<tr>
<td></td>
<td>1184</td>
<td>; CORRESPONDS TO THE REGISTER DESIGNATOR. REGISTER</td>
</tr>
<tr>
<td></td>
<td>1185</td>
<td>; POINTER VALUES ARE USED TO POINT INTO THE REGISTER</td>
</tr>
<tr>
<td></td>
<td>1186</td>
<td>; NAME TABLE (NMTBL) AND REGISTER SAVE LOCATION</td>
</tr>
<tr>
<td></td>
<td>1187</td>
<td>; TABLE (RGTBL).</td>
</tr>
<tr>
<td></td>
<td>1188</td>
<td>;</td>
</tr>
<tr>
<td>03A5 15</td>
<td>1189</td>
<td>DB 6 ; INTERRUPT MASK</td>
</tr>
<tr>
<td>03A6 15</td>
<td>1190</td>
<td>DB 9 ; SPH</td>
</tr>
<tr>
<td>03A7 15</td>
<td>1191</td>
<td>DB 10 ; SPL</td>
</tr>
<tr>
<td>03A8 0A</td>
<td>1192</td>
<td>DB 11 ; PCH</td>
</tr>
<tr>
<td>03A9 00</td>
<td>1193</td>
<td>DB 12 ; PCL</td>
</tr>
<tr>
<td>03AA 08</td>
<td>1194</td>
<td>DD 7 ; H</td>
</tr>
</tbody>
</table>
**NMTBL**: REGISTER NAME TABLE

**03B9 15**
**03BA 15**
**03BB 15**
**03BC 0A**
**03BD 15**
**03BE 15**
**03BF 15**
**03C0 0B**
**03C1 15**
**03C2 15**
**03C3 15**
**03C4 0C**
**03C5 15**
**03C6 15**
**03C7 15**
**03C8 0D**
**03C9 15**
**03CA 15**
**03CB 15**
**03CC 0B**
**03CD 15**
**03CE 15**
**03CF 15**
**03D0 0F**
**03D1 15**
**03D2 15**
**03D3 15**
**03D4 13**
**03D5 15**
**03D6 15**
**03D7 15**
**03D8 10**
**03D9 15**
**03DA 15**
**03DB 15**
**03DC 11**
**03DD 15**
**03DE 05**
**03DF 12**
**03E0 10**
**03E1 15**
**03E2 05**
**03E3 12**
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>03E4</td>
<td>11</td>
<td>1218</td>
<td>BLANK,LETRP,LETRC,LETRH ; PROGRAM COUNTER HIGH BYTE</td>
</tr>
<tr>
<td>03E5</td>
<td>15</td>
<td>1219</td>
<td>BLANK,LETRP,LETRC,LETRL ; PROGRAM COUNTER LOW BYTE</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>03ED</td>
<td>EE</td>
<td>1228</td>
<td>ASAV AND OFFH ; A REGISTER</td>
</tr>
<tr>
<td>03EE</td>
<td>EC</td>
<td>1229</td>
<td>BSAV AND OFFH ; B REGISTER</td>
</tr>
<tr>
<td>03EF</td>
<td>EB</td>
<td>1230</td>
<td>CSAV AND OFFH ; C REGISTER</td>
</tr>
<tr>
<td>03F0</td>
<td>EA</td>
<td>1231</td>
<td>DSAV AND OFFH ; D REGISTER</td>
</tr>
<tr>
<td>03F1</td>
<td>E9</td>
<td>1232</td>
<td>ESAV AND OFFH ; E REGISTER</td>
</tr>
<tr>
<td>03F2</td>
<td>ED</td>
<td>1233</td>
<td>FSAV AND OFFH ; FLAGS</td>
</tr>
<tr>
<td>03F3</td>
<td>F1</td>
<td>1234</td>
<td>ISAV AND OFFH ; INTERRUPT MASK</td>
</tr>
<tr>
<td>03F4</td>
<td>F0</td>
<td>1235</td>
<td>HSAV AND OFFH ; H REGISTER</td>
</tr>
<tr>
<td>03F5</td>
<td>EF</td>
<td>1236</td>
<td>LSAV AND OFFH ; L REGISTER</td>
</tr>
<tr>
<td>03F6</td>
<td>F5</td>
<td>1237</td>
<td>SPSAV AND OFFH ; STACK POINTER HIGH ORDER BYTE</td>
</tr>
<tr>
<td>03F7</td>
<td>F4</td>
<td>1238</td>
<td>SPLSV AND OFFH ; STACK POINTER LOW ORDER BYTE</td>
</tr>
<tr>
<td>03F8</td>
<td>F3</td>
<td>1239</td>
<td>PCHSV AND OFFH ; PROGRAM COUNTER HIGH ORDER BYTE</td>
</tr>
<tr>
<td>03F9</td>
<td>F2</td>
<td>1240</td>
<td>PCLSV AND OFFH ; PROGRAM COUNTER LOW ORDER BYTE</td>
</tr>
<tr>
<td>000D</td>
<td></td>
<td>1241 NUMRG EQU ($ - RGTBL) ; NUMBER OF ENTRIES IN</td>
<td></td>
</tr>
</tbody>
</table>

1242 ; /REGISTER SAVE LOCATION TABLE

1243 ;
1244 ;**********************************************************
1245 ;
1246 ;
1247 ;
1248 ;
1249 ;**********************************************************
1250 ;
1251 ;
1252 ; ABSTRACT
1253 ;
1254 ;
1255 ;
1256 ; THIS PROGRAM WAS ADAPTED, WITH FEW CHANGES, FROM THE SDK-80 MONITOR.
1257 ; THIS PROGRAM RUNS ON THE 8085 BOARD AND IS DESIGNED TO PROVIDE
1258 ; THE USER WITH A MINIMAL MONITOR. BY USING THIS PROGRAM,
1259 ; THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD
1260 ; A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS
1261 ; ALREADY IN MEMORY. THE MONITOR ALSO PROVIDES THE USER WITH
1262 ; ROUTINES FOR PERFORMING CONSOLE I/O.
1263 ;
1264 ; PROGRAM ORGANIZATION

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
STHPO
STHLF
VALDG
VALDL

MONITOR EQUATES
LOC OBJ SEQ SOURCE STATEMENT

1321 ;******************************************************************************
1322 ;
1323 ;
001B 1324 BCHR EQU 1BH ; CODE FOR BREAK CHARACTER (ESCAPE)
07FA 1325 BRTAB EQU 07FAH ; LOCATION OF START OF BRANCH TABLE IN ROM
000D 1326 CR EQU 0DH ; CODE FOR CARRIAGE RETURN
001B 1327 ESC EQU 1BH ; CODE FOR ESCAPE CHARACTER
000F 1328 HCHAR EQU OPH ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
00FF 1329 INVRT EQU 0FFH ; MASK TO INVERT HALF BYTE FLAG
000A 1330 LF EQU 0AH ; CODE FOR LINE FEED
0000 1331 LOWER EQU 0 ; DENOTES LOWER HALF OF BYTE IN ICMD
1332 ; LSIGNON EQU --- ; LENGTH OF SIGNON MESSAGE - DEFINED LATER
1333 ; MNSTK EQU --- ; START OF MONITOR STACK - DEFINED IN
1334 ;
1335 ; NCMD S EQU --- ; NUMBER OF VALID COMMANDS - DEFINED LATER
1336 ; NLWLN EQU OFH ; MASK FOR CHECKING MEMORY ADDR DISPLAY
000T 1337 PRTO EQU 07FH ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
1338 ; RAMST EQU --- ; START ADDRESS OF RAM - DEFINED IN
1339 ;
1340 ; RTABS EQU --- ; SIZE OF ENTRY IN RTAE TABLE
1341 ; SSTR T EQU 80H ; SHIF TED START BIT
1342 ; STOPB EQU 40H ; STOP BIT
1343 ; STRT EQU 0COH ; UNSHIF TED START BIT
001B 1344 TERM EQU 1BH ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
00FF 1345 UPPER EQU 0FFH ; DENOTES UPPER HALF OF BYTE IN ICMD
1346 ;
1347 ; DELAY VALUES IF NO WAIT STATE
1348 ;
1349 ; IF 1-WAITS
0480 1350 IBTIM EQU 1152 ; INTER-BIT TIME DELAY
0480 1351 OBTIM EQU 1152 ; OUTPUT INTER-BIT TIME DELAY
0900 1352 TIM2 EQU 2304 ; 2 BIT TIME DELAY
0240 1353 WAIT EQU 576 ; DELAY UNTIL READY TO SAMPLE BITS
1354 ; ENDIF
1355 ;
1356 ; DELAY VALUES IF ONE WAIT STATE
1357 ;
1358 ; IF WAITS
1359 IBTIM EQU 928 ; INTER-BIT DELAY
1360 OBTIM EQU 928 ; OUTPUT INTER-BIT TIME DELAY
1361 TIM2 EQU 1859 ; 2 BIT TIME DELAY
1362 WAIT EQU 464 ; DELAY UNTIL READY TO SAMPLE BITS
1363 ENDIF
1364 ;
1365 ;
1366 ;******************************************************************************
1367 ;
1368 ;
1369 ;
1370 ;
1371 ;
1372 ;
1373 ;
1374 ;
1375 ;

RESTART ENTRY POINT
LOC OBJ                  SEQ                      SOURCE STATEMENT

1376 ;                     *******************************************
1377 ;                     PRINT SIGNON MESSAGE
1378 ;                     *******************************************
1379 ;                     GO:
1380 ;                     MSGL:
1381 ;                     LXI H,SGNON ; GET ADDRESS OF SIGNON MESSAGE
1382 ;                     MVI B,LSGNON ; COUNTER FOR CHARACTERS IN MESSAGE
1383 ;                     03FF 7E
1384 ;                     MOV C,M ; FETCH NEXT CHAR TO C REG
1385 ;                     03F0 614
1386 ;                     CALL GO ; SEND IT TO THE CONSOLE
1387 ;                     0400 C3405
1388 ;                     CALL INX ; POINT TO NEXT CHARACTER
1389 ;                     0404 05
1390 ;                     DCR B ; DECREMENT BYTE COUNTER
1391 ;                     0405 C20F03
1392 ;                     JNZ MSGL ; RETURN FOR NEXT CHARACTER
1393 ;                     03FA 218C07
1394 ;                     ************************************************************
1395 ;                     COMMAND RECOGNIZING ROUTINE
1396 ;                     ************************************************************
1397 ;                     GETCM:
1398 ;                     GTC03:
1399 ;                     LXI SPHL
1400 ;                     MVI CALL
1401 ;                     CALL JfJlP
1402 ;                     CALL CALL
1403 ;                     MOV LXI
1404 ;                     LXI AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
1405 ;                     H,MNSTK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
1406 ;                     CALL ECHO,ERROR
1407 ;                     CALL GETCM,ECHO,ERROR ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
1408 ;                     CALL A,B,C,H,L,F/F'S
1409 ;                     CALL CALL ; PROMPT CHARACTER TO C
1410 ;                     CALL CALL ; SEND PROMPT CHARACTER TO USER TERMINAL
1411 ;                     CALL CALL ; WANT TO LEAVE ROOM FOR RST BRANCH
1412 ;                     CALL CALL ; GET COMMAND CHARACTER TO A
1413 ;                     CALL CALL ; ECHO CHARACTER TO USER
1414 ;                     CALL MOV ; PUT COMMAND CHARACTER INTO ACCUMULATOR
1415 ;                     CALL LXI ; C CONTAINS LOOP AND INDEX COUNT
1416 ;                     CALL LXI ; HL POINTS INTO COMMAND TABLE
1417 ;                     CALL LXI ; HL POINTS INTO COMMAND TABLE
1418 ;                     CALL LXI ; HL POINTS INTO COMMAND TABLE
1419 ;                     LXI H,MNSTK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
1420 ;                     SPHL ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
1421 ;                     MVI C,'.' ; PROMPT CHARACTER TO C
1422 ;                     CALL ECHO ; SEND PROMPT CHARACTER TO USER TERMINAL
1423 ;                     JMP GTC03 ; WANT TO LEAVE ROOM FOR RST BRANCH
1424 ;                     CALL GETCM ; GET COMMAND CHARACTER TO A
1425 ;                     CALL CALL ; ECHO CHARACTER TO USER
1426 ;                     CALL MOV ; PUT COMMAND CHARACTER INTO ACCUMULATOR
1427 ;                     CALL LXI ; C CONTAINS LOOP AND INDEX COUNT
1428 ;                     CALL LXI ; HL POINTS INTO COMMAND TABLE
1429 ;                     CALL LXI ; HL POINTS INTO COMMAND TABLE
1430 ;                     GTC05:
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0421 BE</td>
<td>1431</td>
<td>CMP</td>
<td>M</td>
</tr>
<tr>
<td>0422 CA2D04</td>
<td>1432</td>
<td>JZ</td>
<td>GTC10</td>
</tr>
<tr>
<td>0425 23</td>
<td>1433</td>
<td>INX</td>
<td>H</td>
</tr>
<tr>
<td>0426 0D</td>
<td>1434</td>
<td>DCR</td>
<td>C</td>
</tr>
<tr>
<td>0427 C22104</td>
<td>1435</td>
<td>JNZ</td>
<td>GTC05</td>
</tr>
<tr>
<td>042A C31106</td>
<td>1436</td>
<td>JMP</td>
<td>ERROR</td>
</tr>
<tr>
<td>042D 21A007</td>
<td>1437</td>
<td>LXI</td>
<td>H,CADR</td>
</tr>
<tr>
<td>0430 09</td>
<td>1440</td>
<td>DAD</td>
<td>B</td>
</tr>
<tr>
<td>0431 09</td>
<td>1441</td>
<td>DAD</td>
<td>B</td>
</tr>
<tr>
<td>0432 7E</td>
<td>1442</td>
<td>MOV</td>
<td>A,M</td>
</tr>
<tr>
<td>0433 23</td>
<td>1443</td>
<td>INX</td>
<td>H</td>
</tr>
<tr>
<td>0434 66</td>
<td>1444</td>
<td>MOV</td>
<td>H,M</td>
</tr>
<tr>
<td>0435 EF</td>
<td>1445</td>
<td>MOV</td>
<td>L,A</td>
</tr>
<tr>
<td>0436 89</td>
<td>1446</td>
<td>PCHL</td>
<td></td>
</tr>
</tbody>
</table>

**SEQ SOURCE STATEMENT**

<table>
<thead>
<tr>
<th>LOCS</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0437 0E02</td>
<td>1466</td>
<td>MVI</td>
<td>C,2</td>
</tr>
<tr>
<td>0439 CD5B06</td>
<td>1467</td>
<td>CALL</td>
<td>GETNM</td>
</tr>
<tr>
<td>043C D1</td>
<td>1468</td>
<td>POP</td>
<td>D</td>
</tr>
<tr>
<td>043D E1</td>
<td>1469</td>
<td>POP</td>
<td>H</td>
</tr>
<tr>
<td>043E CDEB05</td>
<td>1470</td>
<td>DCM5:</td>
<td>CALL</td>
</tr>
<tr>
<td>0441 7C</td>
<td>1471</td>
<td>CALL</td>
<td>LCRSTH</td>
</tr>
<tr>
<td>0442 CDC706</td>
<td>1472</td>
<td>MOV</td>
<td>A,H</td>
</tr>
<tr>
<td>0444 7D</td>
<td>1473</td>
<td>CALL</td>
<td>NMOUT</td>
</tr>
<tr>
<td>0446 CDC706</td>
<td>1474</td>
<td>MOV</td>
<td>A,L</td>
</tr>
<tr>
<td>0449 0E20</td>
<td>1475</td>
<td>CALL</td>
<td>NMOUT</td>
</tr>
<tr>
<td>044B CDF805</td>
<td>1476</td>
<td>DCM10:</td>
<td>MVI</td>
</tr>
<tr>
<td>044E 7E</td>
<td>1477</td>
<td>CALL</td>
<td>ECHO</td>
</tr>
<tr>
<td>044F CDC706</td>
<td>1478</td>
<td>CALL</td>
<td>ECHO</td>
</tr>
<tr>
<td>0452 CDA006</td>
<td>1479</td>
<td>MOV</td>
<td>A,H</td>
</tr>
<tr>
<td>0455 D25E04</td>
<td>1480</td>
<td>CALL</td>
<td>NMOUT</td>
</tr>
<tr>
<td>0456 CDEB05</td>
<td>1481</td>
<td>CALL</td>
<td>NMOUT</td>
</tr>
</tbody>
</table>

**FUNCTION: DCMD**

- **INPUTS:** None
- **OUTPUTS:** None
- **CALLS:** ECHO, NMOUT, HILO, GETCM, CROUT, GETNM
- **DESTROYS:** A, B, C, D, E, H, L, F/P'S
- **DESCRIPTION:** DCMD IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND

**COMMAND IMPLEMENTING ROUTINES**

- **DCMD:**
  - MVI C,2
  - CALL GETNM
  - POP D
  - POP H
  - CALL CROUT
  - MOV A,H
  - CALL NMOUT
  - MOV A,L
  - CALL NMOUT
  - MVI C,' '
  - CALL ECHO
  - MOV A,H
  - CALL NMOUT
  - CALL HILO
  - CALL NMOUT
  - CALL CROUT

- **DCM10:**
  - MVI C,' '
DCM15:
JMP
INX
MOV
ANI
JNZ
JMP
GETCM
H
A,L
NEWLN
DCM10
DCM05
ALL
DONE
IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
GET
LOW ORDER BITS OF NEW ADDRESS
SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
START OF NEW LINE
NO - NOT AT END OF LINE
YES - START NEW LINE WITH ADDRESS

GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.

FUNCTION: GCMD
INPUTS: NONE
OUTPUTS: NONE
CALLS: ERROR,GETHX,RSTTF
DESTROYS: A,B,C,D,E,H,L,F/F'S
DESCRIPTION: GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.

GCMD:
GCH05:
GCM10:
CALL FALSE JNC MOV CPI JNZ LXI MOV INX MOV JMP MOV CPI JNZ JMP GETHX GCM05 GCM05
GET ADDRESS (IF PRESENT) FROM INPUT STREAM BRANCH IF NO NUMBER PRESENT A,D ELSE, GET TERMINATOR CR SEE IF CARRIAGE RETURN ERROR ERROR IF NOT PROPERLY TERMINATED H,PSAV; WANT NUMBER TO REPLACE SAVE PGM COUNTER M,C H,B A,D CR ERROR RSTOR IF NO STARTING ADDRESS, MAKE SURE THAT ICARRIAGE TERMINATED COMMAND ERROR IF NOT RESTORE REGISTERS AND BEGIN EXECUTION (RSTOR IS IN KEYBOARD MONITOR)

FUNCTION: ICMD
INPUTS: NONE
OUTPUTS: NONE
CALLS: ERROR,ECHO,GETCH,VALDL,VALDG,CNVBN,STHLF,GETNM,CROUT
DESTROYS: A,B,C,D,E,H,L,F/F'S
DESCRIPTION: ICMD IMPLEMENTS THE INSERT CODE INTO MEMORY (I) COMMAND.

ICMD:
MVI C,1
CALL GETNM ; GET SINGLE NUMBER FROM INPUT STREAM
LOC  OBJ  SEQ  SOURCE STATEMENT

048D 32FD20  1541  STA  TEMP  ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
0490 D1  1542  POP  D  ; ADDRESS OF START TO DE
0493  CMC05:
0491 CD1FO6  1544  CALL  GETCH  ; GET A CHARACTER FROM INPUT STREAM
0494 4F  1545  MOV  C,A
0495 CD8F05  1546  CALL  ECHO  ; ECHO IT
0498 79  1547  MOV  A,C  ; PUT CHARACTER BACK INTO A
0499 FE1B  1548  CPI  TBM  ; SEE IF CHARACTER IS A TERMINATING CHARACTER
049B CAC704  1549  JZ  ICM25  ; IF SO, ALL DONE ENTERING CHARACTERS
049C CD9F07  1550  CALL  VALDL  ; ELSE, SEE IF VALID DELIMITER
049D 1551  TRUE  ICM05  ; IF SO SIMPLY IGNORE THIS CHARACTER
04A1 DA9104  1552+  JC  ICM05
04A4 CD5F07  1553  CALL  VALDG  ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
04A7 D2C104  1554  FALSE  ICM20  ; IF NOT, BRANCH TO HANDLE ERROR CONDITION
04AE CD8B05  1555  CALL  CNVBN  ; CONVERT DIGIT TO BINARY
04AD 4F  1556  MOV  C,A  ; MOVE RESULT TO C
04AE CD3F07  1557  CALL  STLHF  ; STORE IN APPROPRIATE HALF WORD
04B1 3AFD20  1558  LDA  TEMP  ; GET HALF BYTE FLAG
04B4 BT  1559  ORA  A  ; SET F/F'S
04B5 C2B904  1560  JNZ  ICM10  ; BRANCH IF FLAG SET FOR UPPER
04B8 13  1561  INX  D  ; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
04B9 EEFF  1562  XRI  INVRT  ; TOGGLE STATE OF FLAG
04BB 32FD20  1563  STA  TEMP  ; PUT NEW VALUE OF FLAG BACK
04BE C39104  1564  JMP  ICM05  ; PROCESS NEXT DIGIT
04C1 CD3407  1565  CALL  STHFO  ; ILLEGAL CHARACTER
04C4 C31106  1566  JMP  ICM25  ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
04C7 CD3407  1567  CALL  STHFO  ; ADD CARRIAGE RETURN
04CA CDE805  1568  CALL  CROUT
04CD C30804  1569  JMP  GETCM
04D0 OE03
04D2 CD5B06  1570  CALL  GETNM  ; GET 3 NUMBERS FROM INPUT STREAM
04D5 C1  1571  POP  B  ; DESTINATION ADDRESS TO BC
04D6 E1  1572  POP  H  ; ENDING ADDRESS TO HL
04D7 D1  1573  POP  D  ; STARTING ADDRESS TO DE
04D8 B5  1574  CALL  HILO
04DA 62  1575  MOV  H,D
04DB 6B  1576  MOV  L,E  ; SOURCE ADDRESS TO HL

FUNCTION: MCMD
INPUTS: NONE
OUTPUTS: NONE
CALLS: GETCH,HIL0,GETNM
DESTROYS: A,B,C,D,E,H,L,F/F'S
DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.
**LOC** | **OBJ** | **SEQ** | **SOURCE STATEMENT**
--- | --- | --- | ---
04DB 7E | 1596 | MOV A,M | ; GET SOURCE BYTE
04DC 6C | 1597 | MOV H,B | ; DESTINATION ADDRESS TO HL
04DD 69 | 1598 | MOV L,C | ; MOVE BYTE TO DESTINATION
04DE 77 | 1599 | MOV M,A | ; INCREMENT DESTINATION ADDRESS
04DF 03 | 1600 | INX B | ; INCREMENT SOURCE ADDRESS
04EO 78 | 1601 | MOV A,B | ; TEST FOR DESTINATION ADDRESS OVERFLOW
04F1 B1 | 1602 | ORA C | ; CALL HILO - SEE IF ENDING ADDR>=SOURCE ADDR
04F2 CA0804 | 1603 | JZ GETCM | ; IF SO, CAN TERMINATE COMMAND
04F3 13 | 1604 | INX D | ; ELSE, GET BACK ENDING ADDRESS
04F4 E1 | 1605 | POP H | ; GETCM | ; IF NOT, COMMAND IS DONE
04F6 CDA006 | 1606 | CALL HILO | ; SEE IF ENDING ADDR>=SOURCE ADDR
04F7 D0804 | 1607 | JNC GETCM | ; IF NOT, COMMAND IS DONE
04F8 C3D804 | 1608 | JMP MCM05 | ; MOVE ANOTHER BYTE
04F9 | 1610 | ;
04F9 | 1611 | ;
04FA CD2606 | 1623 | CALL GETHX | ; GET A NUMBER, IF PRESENT, FROM INPUT
04FB C5 | 1624 | PUSH B | ; GET NUMBER TO HL - DENOTES MEMORY LOCATION
04FC E1 | 1625 | POP H | ; GET TERMINATOR
04FD CD2606 | 1626 | SCM05: | ; SEE IF SPACE
04FE 7A | 1627 | MOV A,D | ; YES - CONTINUE PROCESSING
04FF CA0005 | 1628 | CPI ' ' | ; ELSE, SEE IF COMMA
0500 FE2C | 1629 | JZ SCM10 | ; TERMINATE COMMAND
0501 C20804 | 1630 | CPI ',' | ; GET CONTENTS OF SPECIFIED LOCATION TO A
0502 7E | 1631 | JNZ GETCM | ; DISPLAY CONTENTS ON CONSOLE
0503 CDC706 | 1632 | SCM10: | ; USE DASH FOR SEPARATOR
0504 0E2D | 1633 | CALL GETHX | ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
0505 CDF805 | 1634 | CALL ECHO | ; IF NO VALUE PRESENT, BRANCH
0506 CD2606 | 1635 | CALL GETHX | ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
0507 2D1005 | 1636 | JNC SCM15 | ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
0508 71 | 1637 | MOV M,C | ; DISPLAY CONTENTS ON CONSOLE
0509 23 | 1638 | INX H | ; TERMINATE COMMAND
050A C3F504 | 1639 | JMP SCM05 | ; GET A NUMBER, IF PRESENT, FROM INPUT
050B | 1640 | ;
050C | 1641 | ;
050D 23 | 1642 | INX H | ; TERMINATE COMMAND
050E C3F504 | 1643 | JMP SCM05 | ; TERMINATE COMMAND
LOC    OBJ    SEQ    SOURCE STATEMENT

1651 ; OUTPUTS: NONE

1652 ; CALLS: GETCH,ECHO,REGDS,GETCM,ERROR,RGADR,NMOUT,CROUT,GETHX

1653 ; DESTRUCTS: A,B,C,D,E,H,L,F/F'S

1654 ; DESCRIPTION: XCND IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)

1655 ; COMMAND.

1656 ;

1657 XCMD:

051A  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

0527  4F    1666   MOV A,C

0528  CD1B07  1667   CALL RGADR ; CONVERT IDENTIFIER INTO RTAE TABLE ADDR

052B  C5    1668   PUSH B

052C  E1    1669   POP H ; PUT POINTER TO REGISTER ENTRY INTO HL

052D  0E20  1670   MOV C,"","";

052F  CDF805  1671   CALL ECHO ; ECHO SPACE TO USER

0532  79    1672   MOV A,C

0533  32FD20  1673   CALL XCM10:

0536  3AFD20  1674   STA TEMP ; PUT SPACE INTO TEMP AS DELIMITER

0539  FE20   1675   LDA TEMP ; GET TERMINATOR

053B  C4A305  1676   CPI "" ; SEE IF A BLANK

053F  CA4305  1677   JZ XCM15 ; YES - GO CHECK POINTER INTO TABLE

0543  3E17  1678   CPI "","" ; NO - SEE IF COMMA

0546  C20804  1679   JNZ GETCM ; NO - MUST BE CARRIAGE RETURN TO END COMMAND

0549  7E    1680   MOV A,M

054B  47    1681   CPI "" ; SET F/F'S

054C  C24E05  1682   JNZ XCM15 ; BRANCH IF NOT AT END OF TABLE

054F  CDE805  1683   CALL CRUT ; ELSE, OUTPUT CARRIAGE RETURN LINE FEED

0552  C30804  1684   JMP GETCM ; AND EXIT

0555  3F20  1685   JMP XCM18;

0558  E1    1686   FLG

055B  5E    1687   MOV B,M

055C  1620  1688   MOV D,RAMST SHR 8 ; ADDRESS OF SAVE LOCATION FROM TABLE

055D  23    1689   INX H

055F  46    1690   MOV B,M ; FETCH LENGTH FLAG FROM TABLE

0562  D5    1691   PUSH D ; SAVE ADDRESS OF SAVE LOCATION

0565  D5    1692   PUSH D

0566  E1    1693   MOV H ; MOVE ADDRESS TO HL

0567  C5    1694   PUSH B ; SAVE LENGTH FLAG

056A  7E    1695   MOV A,M ; GET 8 BITS OF REGISTER FROM SAVE LOCATION

056B  CDC706  1696   CALL NMOUT ; DISPLAY IT

056C  F1    1697   POP PSW ; GET BACK LENGTH FLAG

056D  5F    1698   PUSH PSW ; SAVE IT AGAIN

056E  77    1699   MOV A,M

056F  CA6705  1700   JMP XCM20 ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY

0572  2B    1701   JZ XCM20

0575  7E    1702   DCX H ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS

0578  CDC706  1703   CALL NMOUT ; DISPLAY THEM

057B  FE0D   1704   CPI CR

057D  FE804  1705   CALL XCM18;

057F  FE0D   1706   CPI CR
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0567</td>
<td>0E2D</td>
<td>1706</td>
<td>XCM20:</td>
</tr>
<tr>
<td>0569</td>
<td>CDF805</td>
<td>1707</td>
<td>MVI C,'-'</td>
</tr>
<tr>
<td>056C</td>
<td>CD2606</td>
<td>1708</td>
<td>CALL ECHO</td>
</tr>
<tr>
<td>056E</td>
<td>D28705</td>
<td>1709</td>
<td>CALL GETHX</td>
</tr>
<tr>
<td>0572</td>
<td>7A</td>
<td>1710</td>
<td>FALSE XCM30</td>
</tr>
<tr>
<td>0573</td>
<td>32FD20</td>
<td>1711+</td>
<td>JNC XCM30</td>
</tr>
<tr>
<td>0576</td>
<td>F1</td>
<td>1712</td>
<td>MOV A,D</td>
</tr>
<tr>
<td>0577</td>
<td>B1</td>
<td>1713</td>
<td>STA TEMP</td>
</tr>
<tr>
<td>0578</td>
<td>B7</td>
<td>1714</td>
<td>POP PSW</td>
</tr>
<tr>
<td>0579</td>
<td>CA7E05</td>
<td>1715</td>
<td>POP H</td>
</tr>
<tr>
<td>057C</td>
<td>70</td>
<td>1716</td>
<td>ORA A</td>
</tr>
<tr>
<td>057D</td>
<td>2B</td>
<td>1717</td>
<td>SET P/F'S</td>
</tr>
<tr>
<td>057E</td>
<td>71</td>
<td>1718</td>
<td>JZ XCM25</td>
</tr>
<tr>
<td>057F</td>
<td>110300</td>
<td>1719</td>
<td>MOV M,B</td>
</tr>
<tr>
<td>0582</td>
<td>B1</td>
<td>1720</td>
<td>DCX H</td>
</tr>
<tr>
<td>0583</td>
<td>19</td>
<td>1721</td>
<td>POINT TO SAVE LOCATION FOR LOWER 8 BITS</td>
</tr>
<tr>
<td>0584</td>
<td>C33605</td>
<td>1722</td>
<td>MOV M,C</td>
</tr>
<tr>
<td>0587</td>
<td>7A</td>
<td>1723</td>
<td>STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG</td>
</tr>
<tr>
<td>0588</td>
<td>32FD20</td>
<td>1724</td>
<td>XCM27:</td>
</tr>
<tr>
<td>0588</td>
<td>D1</td>
<td>1725</td>
<td>LXI D,RTABS</td>
</tr>
<tr>
<td>058C</td>
<td>D1</td>
<td>1726</td>
<td>POP H</td>
</tr>
<tr>
<td>058D</td>
<td>C37F05</td>
<td>1727</td>
<td>POINTER INTO REGISTER TABLE RTAB</td>
</tr>
<tr>
<td>0590</td>
<td>F3</td>
<td>1728</td>
<td>MOV A,D</td>
</tr>
<tr>
<td>0591</td>
<td>D5</td>
<td>1729</td>
<td>GET TERMINATOR</td>
</tr>
<tr>
<td>0592</td>
<td>20</td>
<td>1730</td>
<td>STA TEMP</td>
</tr>
<tr>
<td>0593</td>
<td>17</td>
<td>1731</td>
<td>SAVE IN MEMORY</td>
</tr>
<tr>
<td>0594</td>
<td>DA9205</td>
<td>1732</td>
<td>JMP XCM27</td>
</tr>
</tbody>
</table>

**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.

**SOURCE STATEMENT**

```
0567 0E2D 1706 XCM20:
0569 CDF805 1707 MVI C,'-' ; USE DASH AS SEPARATOR
056C CD2606 1708 CALL ECHO ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
056E D28705 1709 CALL GETHX ; NO - GO CHECK FOR NEXT REGISTER
0572 7A 1710 FALSE XCM30
0573 32FD20 1711+ JNC XCM30
0576 F1 1712 MOV A,D
0577 B1 1713 STA TEMP ; ELSE, SAVE THE TERMINATOR FOR NOW
0578 B7 1714 POP PSW ; GET BACK LENGTH FLAG
0579 CA7E05 1715 POP H ; PUT ADDRESS OF SAVE LOCATION INTO HL
057C 70 1716 ORA A ; SET P/F'S
057D 2B 1717 JZ XCM25 ; IF 8 BIT REGISTER, BRANCH
057E 71 1718 MOV M,B ; SAVE UPPER 8 BITS
057F 110300 1719 DCX H ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
0582 B1 1720 MOV M,C ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
0583 19 1721 XCM27: ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
0584 C33605 1722 LXI D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
0588 32FD20 1723 POP H ; POINTER INTO REGISTER TABLE RTAB
0588 D1 1724 STA TEMP ; SAVE IN MEMORY
058C D1 1725 POP D ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
058D C37F05 1726 JMP XCM10 ; DO NEXT REGISTER
0590 F3 1727 XCM30: ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
0591 D5 1728 MOV A,D ; GET TERMINATOR
0592 20 1729 STA TEMP ; SAVE IN MEMORY
0593 17 1730 POP D ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
0594 DA9205 1731 POP D ; OF SAVE LOCATION
0595 CI:
0595 CI05: |
0595 CI05: |
```

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0597</td>
<td>111002</td>
<td>1761</td>
<td>LXI D, WAIT ; WAIT UNTIL MIDDLE OF BIT</td>
</tr>
<tr>
<td>059A</td>
<td>CDF105</td>
<td>1762</td>
<td>CALL DELAY</td>
</tr>
<tr>
<td>059D</td>
<td>C5</td>
<td>1763</td>
<td>PUSH B ; SAVE BC</td>
</tr>
<tr>
<td>059E</td>
<td>010800</td>
<td>1764</td>
<td>LXI B, 8 ; B&lt;-0, C&lt;-# BITS TO RECEIVE</td>
</tr>
<tr>
<td>05AE</td>
<td>CI10:</td>
<td>1765</td>
<td></td>
</tr>
<tr>
<td>05A1</td>
<td>118004</td>
<td>1766</td>
<td>LXI D, IBTIM</td>
</tr>
<tr>
<td>05A4</td>
<td>CDF105</td>
<td>1767</td>
<td>CALL DELAY ; WAIT UNTIL MIDDLE OF NEXT BIT</td>
</tr>
<tr>
<td>05A7</td>
<td>20</td>
<td>1768</td>
<td>RIM ; GET THE BIT</td>
</tr>
<tr>
<td>05A8</td>
<td>17</td>
<td>1769</td>
<td>RAL ; INTO CARRY</td>
</tr>
<tr>
<td>05A9</td>
<td>76</td>
<td>1770</td>
<td>MOV A,B ; GET PARTIAL RESULT</td>
</tr>
<tr>
<td>05AA</td>
<td>1F</td>
<td>1771</td>
<td>RAR ; SHIFT IN NEXT DATA BIT</td>
</tr>
<tr>
<td>05AB</td>
<td>47</td>
<td>1772</td>
<td>MOV B,A ; REPLACE RESULT</td>
</tr>
<tr>
<td>05AC</td>
<td>0D</td>
<td>1773</td>
<td>DCR C ; DEC COUNT OF BITS TO GO</td>
</tr>
<tr>
<td>05AD</td>
<td>C2A105</td>
<td>1774</td>
<td>JNZ CI10 ; BRANCH IF MORE LEFT</td>
</tr>
<tr>
<td>05B0</td>
<td>118004</td>
<td>1775</td>
<td>LXI D, IBTIM ; ELSE, WANT TO WAIT OUT STOP BIT</td>
</tr>
<tr>
<td>05B3</td>
<td>CDF105</td>
<td>1776</td>
<td>CALL DELAY</td>
</tr>
<tr>
<td>05BD</td>
<td>70</td>
<td>1777</td>
<td>MOV A,B ; GET RESULT</td>
</tr>
<tr>
<td>05BE</td>
<td>C1</td>
<td>1778</td>
<td>POP B</td>
</tr>
<tr>
<td>05B8</td>
<td>D1</td>
<td>1779</td>
<td>POP D ; RESTORE SAVED REGISTERS</td>
</tr>
<tr>
<td>05B9</td>
<td>F6</td>
<td>1780</td>
<td>EI</td>
</tr>
<tr>
<td>05BA</td>
<td>C9</td>
<td>1781</td>
<td>RET ; THAT'S IT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>05BB</td>
<td>79</td>
<td>1782 ;</td>
<td></td>
</tr>
<tr>
<td>05BC</td>
<td>D630</td>
<td>1783 ;</td>
<td></td>
</tr>
<tr>
<td>05BE</td>
<td>FD3A</td>
<td>1784 ; ***********************************************</td>
<td></td>
</tr>
<tr>
<td>05C0</td>
<td>F8</td>
<td>1785 ;</td>
<td></td>
</tr>
<tr>
<td>05C1</td>
<td>D607</td>
<td>1786 ;</td>
<td></td>
</tr>
<tr>
<td>05C3</td>
<td>C9</td>
<td>1787 ;</td>
<td></td>
</tr>
<tr>
<td>05CC</td>
<td>70</td>
<td>1788 ;</td>
<td></td>
</tr>
<tr>
<td>05CE</td>
<td>1F</td>
<td>1789 ;</td>
<td></td>
</tr>
<tr>
<td>05CF</td>
<td>47</td>
<td>1790 ;</td>
<td></td>
</tr>
<tr>
<td>05D0</td>
<td>0D</td>
<td>1791 ;</td>
<td></td>
</tr>
<tr>
<td>05D1</td>
<td>C2A105</td>
<td>1792 ;</td>
<td></td>
</tr>
<tr>
<td>05D2</td>
<td>70</td>
<td>1793 ;</td>
<td></td>
</tr>
<tr>
<td>05D3</td>
<td>C1</td>
<td>1794 ;</td>
<td></td>
</tr>
<tr>
<td>05D4</td>
<td>78</td>
<td>1795 ;</td>
<td></td>
</tr>
<tr>
<td>05D5</td>
<td>DB</td>
<td>1796 ;</td>
<td></td>
</tr>
<tr>
<td>05D6</td>
<td>1F</td>
<td>1797 ;</td>
<td></td>
</tr>
<tr>
<td>05D7</td>
<td>47</td>
<td>1798 ;</td>
<td></td>
</tr>
<tr>
<td>05D8</td>
<td>0D</td>
<td>1799 ;</td>
<td></td>
</tr>
<tr>
<td>05D9</td>
<td>70</td>
<td>1800 ;</td>
<td></td>
</tr>
<tr>
<td>05DA</td>
<td>C1</td>
<td>1801 ;</td>
<td></td>
</tr>
<tr>
<td>05DB</td>
<td>78</td>
<td>1802 ;</td>
<td></td>
</tr>
<tr>
<td>05DC</td>
<td>DB</td>
<td>1803 ;</td>
<td></td>
</tr>
<tr>
<td>05DD</td>
<td>1F</td>
<td>1804 ;</td>
<td></td>
</tr>
<tr>
<td>05DE</td>
<td>47</td>
<td>1805 ; ***********************************************</td>
<td></td>
</tr>
<tr>
<td>05DF</td>
<td>0D</td>
<td>1806 ;</td>
<td></td>
</tr>
<tr>
<td>05E0</td>
<td>70</td>
<td>1807 ;</td>
<td></td>
</tr>
<tr>
<td>05E1</td>
<td>1F</td>
<td>1808 ;</td>
<td></td>
</tr>
<tr>
<td>05E2</td>
<td>47</td>
<td>1809 ;</td>
<td></td>
</tr>
<tr>
<td>05E3</td>
<td>0D</td>
<td>1810 ;</td>
<td></td>
</tr>
<tr>
<td>05E4</td>
<td>70</td>
<td>1811 ;</td>
<td></td>
</tr>
<tr>
<td>05E5</td>
<td>C1</td>
<td>1812 ;</td>
<td></td>
</tr>
<tr>
<td>05E6</td>
<td>78</td>
<td>1813 ;</td>
<td></td>
</tr>
<tr>
<td>05E7</td>
<td>DB</td>
<td>1814 ;</td>
<td></td>
</tr>
<tr>
<td>05E8</td>
<td>1F</td>
<td>1815 CO:</td>
<td></td>
</tr>
</tbody>
</table>

**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.

05BB | 79 | MOV A,C ; SUBTRACT CODE FOR '0' FROM ARGUMENT
05BC | D630 | SUI '0' ; WANT TO TEST FOR RESULT OF 0 TO 9
05BE | FD3A | CPI 10 ; IF SO, THEN ALL DONE
05C0 | F8 | 1800 | RM ; IF SO, THEN ALL DONE
05C1 | D607 | 1801 | SUI 7 ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
05C3 | C9 | 1802 | 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**

**DESCRIPTION:** CO SENDS ITS INPUT ARGUMENT TO THE TTY.
LOC OBJ SEQ SOURCE STATEMENT

05C4 F3 1816 DI
05C5 C5 1817 PUSH B ; SAVE BC
05C6 D5 1818 PUSH D ; SAVE DE
05C7 3EC0 1819 MVI A,STRT ; START BIT MASK
05C9 0607 1820 MVI B,7 ; B WILL COUNT BITS TO SEND

05CB 30 1822 SIM ; SEND A BIT
05CC 118004 1823 LXI D,OBTIM ; WAIT FOR TTY TO HANDLE IT
05CF CDF105 1824 CALL DELAY
05D2 79 1825 MOV A,C ; PICK UP BITS LEFT TO SEND
05D3 1F 1826 RAR ; LOW ORDER BIT TO CARRY
05D4 4F 1827 MOV C,A ; PUT REST BACK
05D5 3EB0 1828 MVI A,SBSTRT ; SHIFTED ENABLE BIT
05D7 1F 1829 RAR ; SHIFT IN DATA BIT
05D8 EE80 1830 XRI 80H ; COMPLEMENT DATA BIT
05DA 05 1831 DCR B ; DEC COUNT
05DB F2CB05 1832 JP CO05 ; SEND IF MORE BITS NEED TO BE SENT
05DE 3E40 1833 MVI A,STOPB ; ELSE, SEND STOP BIT
05E0 30 1834 SIM
05E1 110009 1835 LXI D,TIM2 ; WAIT OUT PARITY BIT
05E4 CDF105 1836 CALL DELAY
05E7 D1 1837 POP D
05E8 C1 1838 POP B ; RESTORE SAVED REGISTERS
05E9 FB 1839 EI
05EA C9 1840 RET ; ALL DONE

;******************************************************************************

1841 ;
1842 ;
1843 ;
1844 ;
1845 ;
1846 ; FUNCTION CROUT
1847 ; INPUTS: NONE
1848 ; OUTPUTS: NONE
1849 ; CALLS: ECHO
1850 ; DESTROYS: A,B,C,F/F'S
1851 ; DESCRIPTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE
1852 ; FEED) TO THE CONSOLE.
1853 ;
1854 CROUT:
05EB 080D 1855 MVI C,CR
05ED CDF805 1856 CALL ECHO
05F0 C9 1857 RET

;******************************************************************************

1858 ;
1859 ;
1860 ;
1861 ;
1862 ;
1863 ; FUNCTION: DELAY
1864 ; INPUTS: DE - 16 BIT INTEGER DENOTING NUMBER OF TIMES TO LOOP
1865 ; OUTPUTS: NONE
1866 ; CALLS: NOTHING
1867 ; DESTROYS: A,D,E,F/F'S
1868 ; DESCRIPTION: DELAY DOES NOT RETURN TO CALLER UNTIL INPUT ARGUMENT
1869 ; IS COUNTED DOWN TO 0.
1870 ;
LOC OBJ SEQ SOURCE STATEMENT

1871 DELAY:
05F1 1B 1872 DCX D ; DECREMENT INPUT ARGUMENT
05F2 7A 1873 MOV A.D
05F3 B3 1874 ORA E
05F4 C2105 1875 JNZ DELAY ; IF ARGUMENT NOT 0, KEEP GOING
05F7 C9 1876 RET

1877 ;
1878 ;
1879 ;************************************************************************************
1880 ;
1881 ;
1882 ; FUNCTION: ECHO
1883 ; INPUTS: C - CHARACTER TO ECHO TO TERMINAL
1884 ; OUTPUTS: C - CHARACTER ECHOED TO TERMINAL
1885 ; CALLS: CO
1886 ; DESTROYS: A,B,F/F'S
1887 ; DESCRIPTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA
1888 ; THE MONITOR, SENDS THAT CHARACTER TO THE USER
1889 ; TERMINAL. A CARRIAGE RETURN IS ECHOED AS A CARRIAGE
1890 ; RETURN LINE FEED, AND AN ESCAPE CHARACTER IS ECHOED AS $.
1891 ;
1892 ECHO:
05F8 41 1893 MOV B.C ; SAVE ARGUMENT
05F9 3E1B 1894 MVI A,ESC
05FB BB 1895 CMP B ; SEE IF ECHOING AN ESCAPE CHARACTER
05FC C20106 1896 JNZ ECHO5 ; NO - BRANCH
05FF 0E24 1897 MVI C,'$' ; YES - ECHO AS $

1898 ECHO5:
0601 CDC405 1899 CALL CO ; DO OUTPUT THROUGH MONITOR
0604 3E0D 1900 MVI A,CR
0606 BB 1901 CMP B ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RETURN
0607 C20F06 1902 JNZ ECH10 ; NO - NO NEED TO TAKE SPECIAL ACTION
060A 0E0A 1903 MVI C,LF ; YES - WANT TO ECHO LINE FEED, TOO
060C CDC405 1904 CALL CO

1905 ECH10:
060F 48 1906 MOV C,B ; RESTORE ARGUMENT
0610 C9 1907 RET

1908 ;
1909 ;
1910 ;
1911 ;
1912 ;
1913 ; FUNCTION: ERROR
1914 ; INPUTS: NONE
1915 ; OUTPUTS: NONE
1916 ; CALLS: ECHO,CROUT,OFTCM
1917 ; DESTROYS: A,B,C,F/F'S
1918 ; DESCRIPTION: ERROR PRINTS THE ERROR CHARACTER (CURRENTLY AN ASTERISK)
1919 ; ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
1920 ; AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.
1921 ;
1922 ERROR:
0611 0E2A 1923 MVI C,' *'
0613 CDF805 1924 CALL ECHO ; SEND * TO CONSOLE
0616 CDEB05 1925 CALL CROUT ; SKIP TO BEGINNING OF NEXT LINE
LOC OBJ SEQ SOURCE STATEMENT

0619 C30804 1926 JMP GETCM ; TRY AGAIN FOR ANOTHER COMMAND
1927 ;
1928 ;
1929 ;**********************************************************************
1930 ;
1931 ;
1932 ; FUNCTION: FRET
1933 ; INPUTS: NONE
1934 ; OUTPUTS: CARRY - ALWAYS 0
1935 ; CALLS: NOTHING
1936 ; DESTROYS: CARRY
1937 ; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
1938 ; INDICATE FAILURE ON RETURN. FRET SETS THE CARRY
1939 ; FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
1940 ; CALLER OF THE ROUTINE INVOKING FRET.
1941 ;
1942 FRET:
061C 37 1943 STC ; FIRST SET CARRY TRUE
061D 3F 1944 CMC ; THEN COMPLEMENT IT TO MAKE IT FALSE
061E C9 1945 RET ; RETURN APPROPRIATELY
1946 ;
1947 ;
1948 ;**********************************************************************
1949 ;
1950 ;
1951 ; FUNCTION: GETCH
1952 ; INPUTS: NONE
1953 ; OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
1954 ; CALLS: CI
1955 ; DESTROYS: A,C,F/F'S
1956 ; DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
1957 ; TO THE CALLING PROGRAM.
1958 ;
1959 GETCH:
061F CD9005 1960 CALL CI ; GET CHARACTER FROM TERMINAL
0622 E67F 1961 ANI PRTYO ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
0624 4F 1962 MOV C,A ; PUT VALUE IN C REGISTER FOR RETURN
0625 C9 1963 RET
1964 ;
1965 ;
1966 ;**********************************************************************
1967 ;
1968 ;
1969 ; FUNCTION: GETHX
1970 ; INPUTS: NONE
1971 ; OUTPUTS: BC - 16 BIT INTEGER
1972 ; D - CHARACTER WHICH TERMINATED THE INTEGER
1973 ; CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
1974 ; 0 IF FIRST CHARACTER IS DELIMITER
1975 ; CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR
1976 ; DESTROYS: A,B,C,D,E,F/F'S
1977 ; DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
1978 ; STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
1979 ; INTEGER. IF MORE THAN 4 HEX DIGITS ARE ENTERED,
1980 ; ONLY THE LAST 4 ARE USED. THE NUMBER TERMINATES WHEN
LaC OBJ 0626 E5 0627 210000 062A 1EOO 062C CD1F06 062F CDF805 0633 CD7907 0636 D24506 0639 51 063A E5 063B C1 063C E1 063D 7B 063F C23207 0642 CA1C06 0645 CD5E07 0648 D21106 064B CDBB05 064E 1EFF 0650 29 0651 29 0652 29 0653 29 0654 0600 0656 4F 0657 09 0658 C32C06

LOC OBJ SEQ SOURCE STATEMENT

1081 ; A VALID DELIMITER IS ENCOUNTERED. THE DELIMITER IS
1082 ; ALSO RETURNED AS AN OUTPUT OF THE FUNCTION. ILLEGAL
1083 ; CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
1084 ; ERROR INDICATION. IF THE FIRST (VALID) CHARACTER
1085 ; ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
1086 ; GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
1087 ; OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
1088 ; OF BC ARE UNDEFINED.
1089 ;
1090 GETHX:
0626 E5 1991 PUSH H ; SAVE HL
0627 210000 1992 LXI H,0 ; INITIALIZE RESULT
062A 1E00 1993 MVI E,0 ; INITIALIZE DIGIT FLAG TO FALSE
062C CD1F06 1994 GETCH ; GET A CHARACTER
062F 4F 1996 MOV C,A
0630 CDF805 1997 CALL ECHO ; ECHO THE CHARACTER
0633 CD7907 1998 CALL VALDL ; SEE IF DELIMITER
0636 D24506 1999 FALSE GHX10 ; NO - BRANCH
0639 51 2000+ JNC GHX10 ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
063A E5 2001 MOV D,C
063B C1 2002 PUSH H ; MOVE RESULT TO BC
063C E1 2003 POP H ; RESTORE HL
063D 7B 2004 MOV A,E ; GET FLAG
063E B7 2005 ORA A ; SET F/F'S
063F C23207 2006 JNZ SRET ; IF FLAG NON-0, A NUMBER HAS BEEN FOUND
0642 CA1C06 2007 JZ FRET ; ELSE, DELIMITER WAS FIRST CHARACTER
0645 CD5E07 2009 GHX10: 2010 CALL VALDG ; IF NOT DELIMITER, SEE IF DIGIT
0648 D21106 2011 FALSE ERROR ; ERROR IF NOT A VALID DIGIT, EITHER
064B CDBB05 2012+ JNC ERROR
064E 1EFF 2013 CALL CNVBN ; CONVERT DIGIT TO ITS BINARY VALUE
0650 29 2014 MVI E,OFFH ; SET DIGIT FLAG NON-0
0651 29 2015 DAD H ; *2
0652 29 2016 DAD H ; *4
0653 29 2017 DAD H ; *8
0654 0600 2018 MVI B,O ; CLEAR UPPER 8 BITS OF BC PAIR
0656 4F 2019 MOV C,A ; BINARY VALUE OF CHARACTER INTO C
0657 09 2020 DAD B ; ADD THIS VALUE TO PARTIAL RESULT
0658 C32C06 2021 JMP GHX05 ; GET NEXT CHARACTER

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.


FUNCTION: GETHX

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
DESCRIPTION: GETHX FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1 AND 3, INCLUSIVE, IN THE INPUT
LOC OBJ SEQ SOURCE STATEMENT

2036 ; STREAM AND RETURNS THEIR VALUES ON THE STACK. IF 2
2037 ; OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
2038 ; LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
2039 ; SECOND NUMBERS WILL BE SET EQUAL. THE LAST NUMBER
2040 ; REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
2041 ; OR AN ERROR INDICATION WILL RESULT.
2042 ;
2043 GETNM:

0605 B 2E03 2044 MVI L,3 ; PUT MAXIMUM ARGUMENT COUNT INTO L
0605 D 79 2045 MOV A,C ; GET THE ACTUAL ARGUMENT COUNT
0605 E B603 2046 ANI 3 ; FORCE TO MAXIMUM OF 3
0606 0C 2047 RZ ; IF 0, DON'T BOTHER TO DO ANYTHING
0606 17 2048 MOV H,A ; ELSE, PUT ACTUAL COUNT INTO H
2049 GNM05:

0606 2C 2D60 2050 CALL GETHX ; GET A NUMBER FROM INPUT STREAM
0606 37 2051 FALSE ; ERROR IF NOT THERE - TOO FEW NUMBERS
0606 4C 2110 2052 JNC ERROR
0606 56 C5 2053 PUSH B ; ELSE, SAVE NUMBER ON STACK
0606 60 2D 2054 DCR L ; DECREMENT MAXIMUM ARGUMENT COUNT
0606 65 25 2055 DCR H ; DECREMENT ACTUAL ARGUMENT COUNT
0606 6E B 2706 2056 JZ GNM10 ; BRANCH IF NO MORE NUMBERS WANTED
0606 77 7A 2057 MOV A,D ; ELSE, GET NUMBER TERMINATOR TO A
0606 7D 5E0D 2058 CPI CR ; SEE IF CARRIAGE RETURN
0606 80 A 2106 2059 JZ ERROR ; ERROR IF SO - TOO FEW NUMBERS
0606 84 C36206 2060 JMP GNM05 ; ELSE, PROCESS NEXT NUMBER
2061 GNM10:

0606 8F 7A 2062 MOV A,D ; WHEN COUNT 0, CHECK LAST TERMINATOR
0606 97 FE0D 2063 CPI CR
0606 A3 C21106 2064 JNZ ERROR ; ERROR IF NOT CARRIAGE RETURN
0606 AD 01FFFF 2065 LXI B,0FFFFH ; HL GETS LARGEST NUMBER
0606 B4 7D 2066 MOV A,L ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT
0606 B7 81 2067 ORA A ; CHECK FOR 0
0606 B8 CA8A06 2068 JZ GNM20 ; IF YES, 3 NUMBERS WERE INPUT
2069 GNM15:

0606 B9 C5 2070 PUSH B ; IF NOT, FILL REMAINING ARGUMENTS WITH OFFFFH
0606 BC 2D 2071 DCR L
0606 C7 C28506 2072 JNZ GNM15
2073 GNM20:

0606 CD A 2074 POP B ; GET THE 3 ARGUMENTS OUT
0606 D3 B1 2075 POP D
0606 D8 E1 2076 POP H
0606 E1 CDA006 2077 CALL HILO ; SEE IF FIRST >= SECOND
0606 F1 2078 FALSE GNM25 ; NO - BRANCH
0606 F6 D29506 2079 JNC GNM25
0606 F7 5A 2080 MOV D,H
0606 FA 5D 2081 MOV E,L ; YES - MAKE SECOND EQUAL TO THE FIRST
2082 GNM25:

0606 FE E3 2083 XTHL ; PUT FIRST ON STACK - GET RETURN ADDR
0606 F6 D5 2084 PUSH D ; PUT SECOND ON STACK
0606 F7 C5 2085 PUSH B ; PUT THIRD ON STACK
0606 FA 8E 2086 PUSH H ; PUT RETURN ADDRESS ON STACK
2087 GNM30:

0606 F9 3D 2088 DCR A ; DECREMENT RESIDUAL COUNT
0606 FA F8 2089 RM ; IF NEGATIVE, PROPER RESULTS ON STACK
0607 E1 2090 POP H ; ELSE, GET RETURN ADDR
FUNCTION: HILO
INPUTS:
- DE - 16 BIT INTEGER
- HL - 16 BIT INTEGER
OUTPUTS:
- CARRY - 1 IF HL<DE
- - 1 IF HL>DE
CALLS: NOTHING
DESCRIPTION: HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE. THE
INTERGERS ARE TREATED AS UNSIGNED NUMBERS. THE CARRY
BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.

HIL05:
06A0 C5 2110 PUSH B ; SAVE BC
06A1 47 2111 MOV B,A ; SAVE A IN B REGISTER
06A2 E5 2112 PUSH H ; SAVE HL PAIR
06A3 7A 2113 MOV A,D ; CHECK FOR DE = 0000H
06A4 B3 2114 ORA E
06A5 CAC106 2115 JZ HIL05 ; WE'RE AUTOMATICALLY DONE IF IT IS
06A8 23 2116 INX H ; INCREMENT HL BY 1
06A9 7C 2117 MOV A,H ; WANT TO TEST FOR 0 RESULT AFTER
06AA BE 2118 ORA L ; /INCREMENTING
06AB CAC106 2119 JZ HIL05 ; IF SO, HL MUST HAVE CONTAINED OFFFFH
06AE E1 2120 POP H ; IF NOT, RESTORE ORIGINAL HL
06AF D5 2121 PUSH D ; SAVE DE
06B0 3EFF 2122 MVI A,OFFH ; WANT TO TAKE 2'S COMPLEMENT OF DE CONTENTS
06B2 AA 2123 XRA D
06B3 57 2124 MOV D,A
06B4 3EFF 2125 MVI A,OFFH
06B6 AB 2126 XRA E
06B7 5F 2127 MOV E,A
06B8 13 2128 INX D ; 2'S COMPLEMENT OF DE TO DE
06B9 7D 2129 MOV A,L
06BA 83 2130 ADD E ; ADD HL AND DE
06BB 7C 2131 MOV A,H
06BC 8A 2132 ADC D ; THIS OPERATION SETS CARRY PROPERLY
06BD D1 2133 POP D ; RESTORE ORIGINAL DE CONTENTS
06BE 78 2134 MOV A,B ; RESTORE ORIGINAL CONTENTS OF A
06BF C1 2135 POP B ; RESTORE ORIGINAL CONTENTS OF BC
06C0 C9 2136 RET ; RETURN WITH CARRY SET AS REQUIRED
HIL05:
06C1 E1 2137 POP H ; IF HL CONTAINS OFFFFH, THEN CARRY CAN
06C2 78 2138 MOV A,B ; /ONLY BE SET TO 1
06C3 C1 2140 POP B ; RESTORE ORIGINAL CONTENTS OF REGISTERS
06C4 C33207 2141 JMP SRET ; SET CARRY AND RETURN
06C4 2142 ;
06C4 2143 ;
06C4 2144 ;
LOC OBJ SEQ SOURCE STATEMENT

2146 ;
2147 ; FUNCTION: NMOUT
2148 ; INPUTS: A - 8 BIT INTEGER
2149 ; OUTPUTS: NONE
2150 ; CALLS: ECHO,PRVAL
2151 ; DESTROYS: A,B,C,F/F'S
2152 ; DESCRIPTION: NMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER IN THE
2153 ; A REGISTER INTO 2 ASCII CHARACTERS. THE ASCII CHARACTERS
2154 ; ARE THE ONES REPRESENTING THE 8 BITS. THESE TWO
2155 ; CHARACTERS ARE SENT TO THE CONSOLE AT THE CURRENT PRINT
2156 ; POSITION OF THE CONSOLE.
2157 ;
2158 NMOUT:

06C7 E5 2159 PUSH H ; SAVE HL - DESTROYED BY PRVAL
06C6 F5 2160 PUSH PSW ; SAVE ARGUMENT
06C9 0F 2161 RRC
06CA 0F 2162 RRC
06CB 0F 2163 RRC
06CC 0F 2164 RRC ; GET UPPER 4 BITS TO LOW 4 BIT POSITIONS
06CD E60F 2165 ANI HCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
06CF 4F 2166 MOV C,A
06D0 CDE206 2167 CALL PRVAL ; CONVERT LOWER 4 BITS TO ASCII
06D3 CDF805 2168 CALL ECHO ; SEND TO TERMINAL
06D6 F1 2169 POP PSW ; GET BACK ARGUMENT
06D7 E60F 2170 ANI HCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
06D9 4F 2171 MOV C,A
06DA CDE206 2172 CALL PRVAL
06DD CDF805 2173 CALL ECHO
06E0 E1 2174 POP H ; RESTORE SAVED VALUE OF HL
06E1 C9 2175 RET
2176 ;
2177 ;******************************************************************************
2178 ;
2179 ;******************************************************************************
2180 ;
2181 ; FUNCTION: PRVAL
2182 ; INPUTS: C - INTEGER, RANGE 0 TO F
2183 ; OUTPUTS: C - ASCII CHARACTER
2184 ; CALLS: NOTHING
2185 ; DESTROYS: B,C,H,L,F/F'S
2186 ; DESCRIPTION: PRVAL CONVERTS A NUMBER IN THE RANGE 0 TO F HEX TO
2187 ; THE CORRESPONDING ASCII CHARACTER, 0-9,A-F. PRVAL
2188 ; DOES NOT CHECK THE VALIDITY OF ITS INPUT ARGUMENT.
2189 ;
2190 PRVAL:

06E2 215407 2191 LXI H,DIGTB ; ADDRESS OF TABLE
06E5 0600 2192 MVI E,0 ; CLEAR HIGH ORDER BITS OF BC
06E7 09 2193 DAD B ; ADD DIGIT VALUE TO HL ADDRESS
06E8 4E 2194 MOV C,M ; FETCH CHARACTER FROM MEMORY
06E9 C9 2195 RET
2196 ;
2197 ;******************************************************************************
2198 ;******************************************************************************
2199 ;
2200 ;
LOC OBJ SEQ SOURCE STATEMENT

FUNCTION: REGDS
INPUTS: NONE
OUTPUTS: NONE
CALLS: ECHO, NMOUT, ERROR, CROUT
DESCRIPTION: REGDS DISPLAYS THE CONTENTS OF THE REGISTER SAVE LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE. THE DISPLAY IS DRIVEN FROM A TABLE, RTAE, WHICH CONTAINS THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS, AND LENGTH (8 OR 16 BITS).

REGDS:
06EA 21C407
06ED 4E
06EE 79
06EF B7
06F0 06F3 D2F706
06F3 CDEB05
06F6 C9
06F7 06FA OE3D
06FC CDFB05
0700 23
0701 AE
0702 CDC706
0703 23
0704 1A
0705 23
0706 1B
0707 1A
070F CDC706
0712 0E20
0714 CDF805
0717 23
0718 C3ED06

06ED 4E
06EE 79
06EF B7
06F0 C2F706
06F3 CDEB05
06F6 C9
0700 23
0701 AE
0702 CDC706
0703 23
0704 1A
0705 23
0706 1B
0707 1A
070F CDC706
0712 0E20
0714 CDF805
0717 23
0718 C3ED06

SOURCE STATEMENT
FUNCTION: REGDS
INPUTS: NONE
OUTPUTS: NONE
CALLS: ECHO, NMOUT, ERROR, CROUT
DESCRIPTION: REGDS DISPLAYS THE CONTENTS OF THE REGISTER SAVE LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE. THE DISPLAY IS DRIVEN FROM A TABLE, RTAE, WHICH CONTAINS THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS, AND LENGTH (8 OR 16 BITS).

REGDS:
LXI H, RTAB ; LOAD HL WITH ADDRESS OF START OF TABLE

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG10:
CALL ECHO ; ECHO CHARACTER
MOV C, 'a'
CALL ECHO ; OUTPUT EQUALS SIGN, I.E. A=
LXI H, RTAB ; LOAD HL WITH ADDRESS OF START OF TABLE

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG10:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

REG15:
MOV A, M ; GET LENGTH FLAG
JZ REG15 ; IF 0, REGISTER IS 8 BITS
DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
LDAX D ; GET LOWER 8 BITS
CALL NMOUT ; DISPLAY THEM

REG05:
MOV C, M ; GET PRINT SYMBOL OF REGISTER
MOV A, C
ORA A ; TEST FOR 0 - END OF TABLE
JNZ REG10 ; IF NOT END, BRANCH
CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
/DISPLAY

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
LOC OBJ   SEQ    SOURCE STATEMENT

2256 :   SAVE LOCATION CORRESPONDING TO THE REGISTER. THIS
2257 :   ADDRESS POINTS INTO RTAB. IF NO MATCH OCCURS, THEN
2258 :   THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS
2259 :   PASSED TO THE ERROR ROUTINE.
2260 :
2261 RGADR:
071B 21C407 2262 LXI H,RTAB ; HL GETS ADDRESS OF TABLE START
071E 110300 2263 LXI D,RTABS ; DE GET SIZE OF A TABLE ENTRY
2264 RGA05:
0721 7E 2265 MOV A,M ; GET REGISTER IDENTIFIER
0722 B7 2266 ORA A ; CHECK FOR TABLE END (IDENTIFIER IS 0)
0723 CA1106 2267 JZ ERROR ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL
0726 B9 2268 CMP C ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT
0727 CA2E07 2269 JZ RGA10 ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR
072A 19 2270 DAD D ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY
072B C32107 2271 JMP RGA05 ; TRY AGAIN
2272 RGA10:
072E 23 2273 INX H ; IF A MATCH, INCREMENT TABLE POINTER TO
072F 44 2274 MOV B,H ; SAVE LOCATION ADDRESS
0730 4D 2275 MOV C,L ; RETURN THIS VALUE
0731 C9 2276 RET
2277 :
2278 :
2279 ;******************************************************************************
2280 :
2281 :
2282 ; FUNCTION: SRET
2283 ; INPUTS: NONE
2284 ; OUTPUTS: CARRY = 1
2285 ; CALLS: NOTHING
2286 ; DESTROYS: CARRY
2287 ; DESCRIPTION: SRET IS JUMPED TO BY ROUTINES WISHING TO RETURN SUCCESS.
2288 ; SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE
2289 ; CALLER OF THE ROUTINE INVOKING SRET.
2290 :
2291 SRET:
0732 37 2292 STC ; SET CARRY TRUE
0733 C9 2293 RET ; RETURN APPROPRIATELY
2294 :
2295 :
2296 :
2297 :
2298 :
2299 ; FUNCTION: STHFO
2300 ; INPUTS: DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
2301 ; OUTPUTS: NONE
2302 ; CALLS: STHLF
2303 ; DESTROYS: A,B,C,H,L,F/F'S
2304 ; DESCRIPTION: STHFO CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF
2305 ; IT IS SET TO LOWER. IF SO, STHFO STORES A 0 TO
2306 ; PAD OUT THE LOWER HALF OF THE ADDRESSED BYTE;
2307 ; OTHERWISE, THE ROUTINE TAKES NO ACTION.
2308 :
2309 STHFO:
0734 3AFD20 2310 LDA TEMP ; GET HALF BYTE FLAG
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0737</td>
<td>B7</td>
<td>2311</td>
<td>ORA A</td>
</tr>
<tr>
<td>0738</td>
<td>C0</td>
<td>2312</td>
<td>RNZ</td>
</tr>
<tr>
<td>0739</td>
<td>0E00</td>
<td>2313</td>
<td>MVI C,O</td>
</tr>
<tr>
<td>073B</td>
<td>CD3F07</td>
<td>2314</td>
<td>CALL STHLF</td>
</tr>
<tr>
<td>073E</td>
<td>C9</td>
<td>2315</td>
<td>RET</td>
</tr>
</tbody>
</table>

;*********************************************************************************************

;FUNCTION: STHLF
2321 ;INPUTS: C - 4 BIT VALUE TO BE STORED IN HALF BYTE
2322 ;DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
2323 ;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).

2334 STHLF:

2335 PUSH D
2336 POP H            ; MOVE ADDRESS OF BYTE INTO HL
2337 MOV A,C          ; GET VALUE
2338 ANI OFH         ; FORCE TO 4 BIT LENGTH
2339 MOV C,A        ; PUT VALUE BACK
2340 LDA TEMP       ; GET HALF BYTE FLAG
2341 ORA A          ; CHECK FOR LOWER HALF
2342 JNZ STH05      ; BRANCH IF NOT
2343 MOV A,M        ; ELSE, GET BYTE
2344 ANI OFOH      ; CLEAR LOWER 4 BITS
2345 ORA C          ; OR IN VALUE
2346 MOV M,A        ; PUT BYTE BACK
2347 RET

2348 STH05:

2349 MOV A,M        ; IF UPPER HALF, GET BYTE
2350 ANI OFH       ; CLEAR UPPER 4 BITS
2351 MOV B,A       ; SAVE BYTE IN B
2352 MOV A,C       ; GET VALUE
2353 RRC
2354 RRC
2355 RRC
2356 RRC
2357 ORA B        ; OR IN ORIGINAL LOWER 4 BITS
2358 MOV M,A      ; PUT NEW CONFIGURATION BACK
2359 RET

2360 ;
2361 ;
2362 ;*********************************************************************************************

;FUNCTION: VALDC
LOC  OBJ  SEQ  SOURCE STATEMENT

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   ; TEST CHARACTER AGAINST '0'
075F FE30 2377 CPI '0'   ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
0761 FA1C06 2378 JM  FRET  ; CODE BETWEEN '0' AND '9'
0764 FE39 2379 CPI '9'   ; ELSE, SEE IF IN RANGE '0'- '9'
0766 FA3207 2380 JM  SRRET ; CODE EQUAL '9'
0769 CA3207 2381 JZ  SRRET ; NOT A DIGIT - TRY FOR A LETTER
076C FE41 2382 CPI 'A'   ; CODE BETWEEN '0' AND '9'
076E FA1C06 2383 JM  FRET  ; NO - CODE BETWEEN '9' AND 'A'
0771 FE47 2384 CPI 'G'   ; CODE GREATER THAN 'F'
0773 FA2C06 2385 JP  FRET  ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE
0776 C33207 2386 JMP  SRRET ;

2387 ;
2388 ;
2389 ;**********************************************************************
2390 ; FUNCTION: VALDL
2391 ; INPUTS:  C - CHARACTER
2392 ; OUTPUTS:  CARRY - 1 IF INPUT ARGUMENT VALID DELIMITER
2393 ; - 0 OTHERWISE
2394 ; CALLS: NOTHING
2395 ; DESTROYS: A,F/F'S
2396 ; DESCRIPTION: VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID
2397 ; DELIMITER CHARACTER (SPACE, COMMA, CARRIAGE RETURN) AND
2398 ; FAILURE OTHERWISE.
2399 ;
2400 ;
2401 ;
2402 VALDL:

0779 79  2403 MOV  A,C   ; CHECK FOR COMMA
077A FE2C 2404 CPI ','   ; CHECK FOR CARRIAGE RETURN
077C CA3207 2405 JZ  SRRET ;
077F FE0D 2406 CPI CR   ; CHECK FOR SPACE
0781 CA3207 2407 JZ  SRRET ;
0784 FE20 2408 CPI ''   ; ERROR IF NONE OF THE ABOVE
0786 CA3207 2409 JZ  SRRET ;
0789 C31C06 2410 JMP  FRET  ;

2411 ;
2412 ;
2413 ;
2414 ;
2415 ;
2416 ;
2417 ;
2418 ;
2419 ;
2420 ;

MONITOR TABLES
LOC OBJ SEQ SOURCE STATEMENT

2421 ;
2422 SGNON: CR,LF,'SDK-85 VER 1.2',CR,LF
078C OD 2423 DB
078E 53444B2D 078D 0A
0792 38352020 0796 20564552
079A 20312E32 079E OD
079F 0A
0014 2424 LSGNON EQU $-SGNON ; LENGTH OF SIGNON MESSAGE
2425 ;
2426 CADR: CR,LF,'SDK-85 VER 1.2',CR,LF
07AO 0000 2427 DW 0
07A2 1405 2428 DW XCMD
07A4 F004 2429 DW SCMD
07A6 D004 2430 DW MCMD
07A8 8004 2431 DW ICMD
07AA 6004 2432 DW GCMD
07AC 3704 2433 DW DCMD
07AE 44 2434 ;
07AS 4D 2435 CTAB: 'D'
07B0 49 2436 DB '0'
07B1 4D 2437 DB 'I'
07B2 53 2438 DB 'M'
07B3 58 2439 DB 'S'
0006 2440 DB 'X'
2441 DB
2442 NCMDS EQU $-CTAB ; NUMBER OF VALID COMMANDS
2443 ;
2444 DIGTB: CR,LF,'SDK-85 VER 1.2',CR,LF
07B4 30 2445 DE '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'
07CA 43 2457 DB 'C'
07CB 44 2458 DB 'D'
07CC 45 2459 DB 'E'
07CD 46 2460 DB 'F'
2461 ;
2462 RTAB: CR,LF,'SDK-85 VER 1.2',CR,LF
07CE 41 2463 DB 'A'
07CF 4E 2464 DB ASAV AND OFFH ; 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
07CE 42 2467 DB 'B'
07C8 EC 2468 DB BSAT AND OFFH
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>07C9</td>
<td>00</td>
<td>2469</td>
<td>DB 0</td>
</tr>
<tr>
<td>07CA</td>
<td>43</td>
<td>2470</td>
<td>DB 'C'</td>
</tr>
<tr>
<td>07CE</td>
<td>63</td>
<td>2471</td>
<td>DB CSAV AND OFFH</td>
</tr>
<tr>
<td>07CC</td>
<td>00</td>
<td>2472</td>
<td>DB 0</td>
</tr>
<tr>
<td>07CD</td>
<td>44</td>
<td>2473</td>
<td>DB 'D'</td>
</tr>
<tr>
<td>07CE</td>
<td>A3</td>
<td>2474</td>
<td>DB CSAV AND OFFH</td>
</tr>
<tr>
<td>07CF</td>
<td>00</td>
<td>2475</td>
<td>DB 0</td>
</tr>
<tr>
<td>07D0</td>
<td>45</td>
<td>2476</td>
<td>DB 'E'</td>
</tr>
<tr>
<td>07D1</td>
<td>E9</td>
<td>2477</td>
<td>DB CSAV AND OFFH</td>
</tr>
<tr>
<td>07D2</td>
<td>00</td>
<td>2478</td>
<td>DB 0</td>
</tr>
<tr>
<td>07D3</td>
<td>46</td>
<td>2479</td>
<td>DB 'F'</td>
</tr>
<tr>
<td>07D4</td>
<td>ED</td>
<td>2480</td>
<td>DB PSAV AND OFFH</td>
</tr>
<tr>
<td>07D5</td>
<td>00</td>
<td>2481</td>
<td>DB 0</td>
</tr>
<tr>
<td>07D6</td>
<td>49</td>
<td>2482</td>
<td>DB 'I'</td>
</tr>
<tr>
<td>07D7</td>
<td>F1</td>
<td>2483</td>
<td>DB ISA V AND OFFH</td>
</tr>
<tr>
<td>07D8</td>
<td>00</td>
<td>2484</td>
<td>DB 0</td>
</tr>
<tr>
<td>07D9</td>
<td>4E</td>
<td>2485</td>
<td>DB 'H'</td>
</tr>
<tr>
<td>07DA</td>
<td>50</td>
<td>2486</td>
<td>DB HSAV AND OFFH</td>
</tr>
<tr>
<td>07DE</td>
<td>00</td>
<td>2487</td>
<td>DB 0</td>
</tr>
<tr>
<td>07DC</td>
<td>4C</td>
<td>2488</td>
<td>DB 'L'</td>
</tr>
<tr>
<td>07DD</td>
<td>EF</td>
<td>2489</td>
<td>DB LSAV AND OFFH</td>
</tr>
<tr>
<td>07DE</td>
<td>00</td>
<td>2490</td>
<td>DB 0</td>
</tr>
<tr>
<td>07DF</td>
<td>4D</td>
<td>2491</td>
<td>DB 'M'</td>
</tr>
<tr>
<td>07E0</td>
<td>F0</td>
<td>2492</td>
<td>DB HSAV AND OFFH</td>
</tr>
<tr>
<td>07E1</td>
<td>01</td>
<td>2493</td>
<td>DB 1</td>
</tr>
<tr>
<td>07E2</td>
<td>53</td>
<td>2494</td>
<td>DB 'N'</td>
</tr>
<tr>
<td>07E3</td>
<td>F5</td>
<td>2495</td>
<td>DB SSAV+1 AND OFFH</td>
</tr>
<tr>
<td>07E4</td>
<td>01</td>
<td>2496</td>
<td>DB 1</td>
</tr>
<tr>
<td>07E5</td>
<td>50</td>
<td>2497</td>
<td>DB 'P'</td>
</tr>
<tr>
<td>07E6</td>
<td>F3</td>
<td>2498</td>
<td>DB PSAV+1 AND OFFH</td>
</tr>
<tr>
<td>07E7</td>
<td>01</td>
<td>2499</td>
<td>DB 1</td>
</tr>
<tr>
<td>07E8</td>
<td>00</td>
<td>2500</td>
<td>DB 0</td>
</tr>
<tr>
<td>07E9</td>
<td>00</td>
<td>2501</td>
<td>DB 0</td>
</tr>
<tr>
<td>07FA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2502</td>
<td>2503</td>
<td>ORG BTAB</td>
<td>BRANCH TABLE FOR USER ACCESSIBLE ROUTINES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORG BTAB</td>
<td></td>
</tr>
<tr>
<td>07FA</td>
<td>C3C405</td>
<td>JMP CO</td>
<td>TTY CONSOLE OUTPUT</td>
</tr>
<tr>
<td>07FD</td>
<td>C39005</td>
<td>JMP CI</td>
<td>TTY CONSOLE INPUT</td>
</tr>
</tbody>
</table>

*************************************************************

2509 ; IN THE FOLLOWING LOCATIONS, THE USER MAY PLACE JUMP INSTRUCTIONS TO 2510 ; ROUTINES FOR HANDLING THE FOLLOWING:- 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 ; 2516 ; ORG USERS   ; START OF USER BRANCH LOCATIONS 2517 ; 2518 RSET5: DB 0,0,0 ; JUMP TO RST 5 ROUTINE 2519 ; 251A 00 251B 00 251C 00 ; JUMP TO RST 6 ROUTINE 251D 00
## PUBLIC SYMBOLS

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>20C5 00</td>
<td>2520</td>
<td>RST75:</td>
<td>DB 0,0,0 ; JUMP TO RST 6.5 (HARDWIRED USER INTERRUPT)</td>
</tr>
<tr>
<td>20C0 00</td>
<td>2521</td>
<td>RSET7:</td>
<td>DB 0,0,0 ; JUMP TO PST 7 ROUTINE</td>
</tr>
<tr>
<td>20D4 00</td>
<td>2522</td>
<td>USINT:</td>
<td>DB 0,0,0 ; JUMP TO &quot;VECTORED INTERRUPT&quot; KEY ROUTINE</td>
</tr>
<tr>
<td>20D5 00</td>
<td>2523</td>
<td>;</td>
<td>;</td>
</tr>
</tbody>
</table>

## EXTERNAL SYMBOLS

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>20E9 00</td>
<td>2534</td>
<td>ESAVE:</td>
<td>DB 0 ; E REGISTER</td>
</tr>
<tr>
<td>20E8 00</td>
<td>2535</td>
<td>DSAVE:</td>
<td>DB 0 ; D REGISTER</td>
</tr>
<tr>
<td>20EB 00</td>
<td>2536</td>
<td>CSAVE:</td>
<td>DB 0 ; C REGISTER</td>
</tr>
<tr>
<td>20ED 00</td>
<td>2537</td>
<td>BSAVE:</td>
<td>DB 0 ; B REGISTER</td>
</tr>
<tr>
<td>20EE 00</td>
<td>2538</td>
<td>FSAVE:</td>
<td>DB 0 ; FLAGS</td>
</tr>
<tr>
<td>20F0 00</td>
<td>2539</td>
<td>ASAVE:</td>
<td>DB 0 ; A REGISTER</td>
</tr>
<tr>
<td>20F1 00</td>
<td>2540</td>
<td>LSAVE:</td>
<td>DB 0 ; L REGISTER</td>
</tr>
<tr>
<td>20F6 00</td>
<td>2541</td>
<td>HSAVE:</td>
<td>DE 0 ; H REGISTER</td>
</tr>
<tr>
<td>20F1 00</td>
<td>2542</td>
<td>ISAVE:</td>
<td>DB 0 ; INTERRUPT MASK</td>
</tr>
<tr>
<td>20F2 00</td>
<td>2543</td>
<td>PSAV:</td>
<td>; PROGRAM COUNTER</td>
</tr>
<tr>
<td>20F3 00</td>
<td>2544</td>
<td>PCLSV:</td>
<td>DB 0 ; LOW ORDER BYTE</td>
</tr>
<tr>
<td>20F4 00</td>
<td>2545</td>
<td>PCHSV:</td>
<td>DB 0 ; HIGH ORDER BYTE</td>
</tr>
<tr>
<td>20F5 00</td>
<td>2546</td>
<td>SSAVE:</td>
<td>; STACK POINTER</td>
</tr>
<tr>
<td>20F5 00</td>
<td>2547</td>
<td>TPSLV:</td>
<td>DE 0 ; LOW ORDER BYTE</td>
</tr>
<tr>
<td>20F5 00</td>
<td>2548</td>
<td>TPSHV:</td>
<td>DB 0 ; HIGH ORDER BYTE</td>
</tr>
<tr>
<td>20F5 00</td>
<td>2549</td>
<td>;</td>
<td>;</td>
</tr>
</tbody>
</table>

**MONITOR STORAGE LOCATIONS**

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>20F6 0000</td>
<td>2554</td>
<td>CURAD:</td>
<td>DW 0 ; CURRENT ADDRESS</td>
</tr>
<tr>
<td>20F8 004</td>
<td>2555</td>
<td>CURDT:</td>
<td>DB 0 ; CURRENT DATA</td>
</tr>
<tr>
<td>0004</td>
<td>2556</td>
<td>OBUFF:</td>
<td>DS 4 ; OUTPUT BUFFER</td>
</tr>
<tr>
<td>20FD 00</td>
<td>2557</td>
<td>TTEM:</td>
<td>; TEMPORARY LOCATION FOR TTY MONITOR</td>
</tr>
<tr>
<td>20F0 00</td>
<td>2556</td>
<td>TTEMP:</td>
<td>; TEMPORARY LOCATION FOR SINGLE STEP ROUTINE</td>
</tr>
<tr>
<td>20F0 0C</td>
<td>2559</td>
<td>RPRTR:</td>
<td>DB 0 ; REGISTER POINTER</td>
</tr>
<tr>
<td>20FF 00</td>
<td>2560</td>
<td>JBUFF:</td>
<td>DB 6 ; INBUF BUFFER</td>
</tr>
<tr>
<td>20FF 00</td>
<td>2561</td>
<td>UCSER:</td>
<td>DB 0 ; USER SHOULD STORE IMAGE OF CSR HERE EACH TIME</td>
</tr>
<tr>
<td>2562</td>
<td>2563</td>
<td>;</td>
<td>/ Routine will destroy CSR contents.</td>
</tr>
</tbody>
</table>

**END**
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERMSG</td>
<td>611 1172#</td>
</tr>
<tr>
<td>ERR</td>
<td>282 315 333 361 396 473 497 506 608#</td>
</tr>
<tr>
<td>ERROR</td>
<td>1436 1512 1521 1569 1922# 2012 2052 2059 2064 2067</td>
</tr>
<tr>
<td>ESAVE</td>
<td>1832 2477 253#</td>
</tr>
<tr>
<td>ESC</td>
<td>1327# 1894</td>
</tr>
<tr>
<td>EXAM</td>
<td>308# 1116</td>
</tr>
<tr>
<td>EXM05</td>
<td>316# 338</td>
</tr>
<tr>
<td>EXM10</td>
<td>326 329#</td>
</tr>
<tr>
<td>EXMSG</td>
<td>359 1173#</td>
</tr>
<tr>
<td>FALS</td>
<td>156# 314 325 395 472 489 1483 1508 1554 1607 1638 1710 1999 2011 2051 2078</td>
</tr>
<tr>
<td>FIVE</td>
<td>1334# 1172#</td>
</tr>
<tr>
<td>FRET</td>
<td>1942# 2008 2378 2383 2385 2410</td>
</tr>
<tr>
<td>FSAVE</td>
<td>180 442 1233 2480 253#</td>
</tr>
<tr>
<td>GI0</td>
<td>363 364#</td>
</tr>
<tr>
<td>GCMD05</td>
<td>1209 1510#</td>
</tr>
<tr>
<td>GCMD10</td>
<td>1517 1523#</td>
</tr>
<tr>
<td>GCMD15</td>
<td>1506# 2432</td>
</tr>
<tr>
<td>GETCH</td>
<td>1425 1544 1559# 1995</td>
</tr>
<tr>
<td>GETCM</td>
<td>1418# 1486 1573 1603 1608 1631 1665 1690 1696 1926</td>
</tr>
<tr>
<td>GETUX</td>
<td>1507 1623 1637 1709 1990# 2050</td>
</tr>
<tr>
<td>GETUXM</td>
<td>1437 1539 1588 2043#</td>
</tr>
<tr>
<td>GHX05</td>
<td>1994# 2022</td>
</tr>
<tr>
<td>GHX10</td>
<td>2000 2009#</td>
</tr>
<tr>
<td>GMM05</td>
<td>2049# 2060</td>
</tr>
<tr>
<td>GMM10</td>
<td>2056 2061#</td>
</tr>
<tr>
<td>GMM15</td>
<td>2069# 2072</td>
</tr>
<tr>
<td>GMM20</td>
<td>2068 2073#</td>
</tr>
<tr>
<td>GMM25</td>
<td>2079 2082#</td>
</tr>
<tr>
<td>GMM30</td>
<td>2087# 2092</td>
</tr>
<tr>
<td>GO</td>
<td>233 1395$</td>
</tr>
<tr>
<td>GOCMD</td>
<td>349# 1116</td>
</tr>
<tr>
<td>GT03</td>
<td>1423 1424#</td>
</tr>
<tr>
<td>GT05</td>
<td>1430# 1435</td>
</tr>
<tr>
<td>GT10</td>
<td>1432 1437#</td>
</tr>
<tr>
<td>GT05</td>
<td>651# 676</td>
</tr>
<tr>
<td>GTH10</td>
<td>665 669#</td>
</tr>
<tr>
<td>GTH20</td>
<td>654 677#</td>
</tr>
<tr>
<td>GTH25</td>
<td>681 683 687#</td>
</tr>
<tr>
<td>GETNX</td>
<td>324 359 394 471 487 646#</td>
</tr>
<tr>
<td>WCCHAR</td>
<td>1328# 2155 2170</td>
</tr>
<tr>
<td>HIL05</td>
<td>2115 2119 2137#</td>
</tr>
<tr>
<td>HIL0</td>
<td>1481 1606 2077 2109#</td>
</tr>
<tr>
<td>HSVAT</td>
<td>1235 2486 2492 2541#</td>
</tr>
<tr>
<td>HDSF</td>
<td>671 709# 1067 1096</td>
</tr>
<tr>
<td>IBTIM</td>
<td>1350# 1766 1775</td>
</tr>
<tr>
<td>IBUFF</td>
<td>254 355 390 760 883 2560#</td>
</tr>
<tr>
<td>IC05</td>
<td>1543# 1552 1566</td>
</tr>
<tr>
<td>ICM10</td>
<td>1561 1563#</td>
</tr>
<tr>
<td>ICM20</td>
<td>1555 1567#</td>
</tr>
<tr>
<td>ICM25</td>
<td>1549 1570#</td>
</tr>
<tr>
<td>ICM30</td>
<td>1537# 2431</td>
</tr>
<tr>
<td>ININT</td>
<td>202 751#</td>
</tr>
<tr>
<td>INSDG</td>
<td>657 779#</td>
</tr>
<tr>
<td>INVT</td>
<td>1329# 1564</td>
</tr>
<tr>
<td>ISAY</td>
<td>258 403 454 994 999 1234 2483 2542#</td>
</tr>
<tr>
<td>KBINIT</td>
<td>120# 566</td>
</tr>
<tr>
<td>KMODE</td>
<td>122# 166</td>
</tr>
<tr>
<td>LETRA</td>
<td>1142 # 1207</td>
</tr>
<tr>
<td>LETRB</td>
<td>1144 # 1208</td>
</tr>
<tr>
<td>LETRC</td>
<td>1146 # 1209</td>
</tr>
<tr>
<td>LETRD</td>
<td>1148 # 1210</td>
</tr>
<tr>
<td>LETRE</td>
<td>1150 # 1211</td>
</tr>
<tr>
<td>LETRF</td>
<td>1152 # 1212</td>
</tr>
<tr>
<td>LETRG</td>
<td>1154 # 1214</td>
</tr>
<tr>
<td>LETRH</td>
<td>1156 # 1213</td>
</tr>
<tr>
<td>LETRL</td>
<td>1158 # 1215</td>
</tr>
<tr>
<td>LETRM</td>
<td>1156 # 1216</td>
</tr>
<tr>
<td>LETRN</td>
<td>1158 # 1217</td>
</tr>
<tr>
<td>LETRP</td>
<td>1150 # 1219</td>
</tr>
<tr>
<td>LETRR</td>
<td>1152 # 1219</td>
</tr>
<tr>
<td>LETRS</td>
<td>1154 # 1219</td>
</tr>
<tr>
<td>LEVR</td>
<td>1156 # 1219</td>
</tr>
<tr>
<td>LSAV</td>
<td>175 437 1019 1236 2489 2540</td>
</tr>
<tr>
<td>LSGNOK</td>
<td>1337 2424</td>
</tr>
<tr>
<td>MCMD</td>
<td>1592 1609</td>
</tr>
<tr>
<td>MCMD</td>
<td>1586 # 2430</td>
</tr>
<tr>
<td>MSDK</td>
<td>124 # 265 1010 1419 2530</td>
</tr>
<tr>
<td>MSGL</td>
<td>1388 1393</td>
</tr>
<tr>
<td>MCMDS</td>
<td>1428 2442</td>
</tr>
<tr>
<td>NEWLN</td>
<td>1336 # 1490</td>
</tr>
<tr>
<td>NOUT</td>
<td>1473 1475 1480 1634 1698 1705 2156 # 2230 2236</td>
</tr>
<tr>
<td>NMTBL</td>
<td>962 1205</td>
</tr>
<tr>
<td>NODOT</td>
<td>125 # 244 249 365 368 479 533 550 592 610 614 966</td>
</tr>
<tr>
<td>NUMC</td>
<td>273 1108</td>
</tr>
<tr>
<td>NUMRG</td>
<td>807 1241</td>
</tr>
<tr>
<td>NXTRG</td>
<td>334 865</td>
</tr>
<tr>
<td>OPTIM</td>
<td>1351 # 1823</td>
</tr>
<tr>
<td>OBUFF</td>
<td>716 734 2556</td>
</tr>
<tr>
<td>OUTS</td>
<td>833 838</td>
</tr>
<tr>
<td>OUT10</td>
<td>837 842</td>
</tr>
<tr>
<td>OUT15</td>
<td>844 # 864</td>
</tr>
<tr>
<td>OUT20</td>
<td>853 855 858</td>
</tr>
<tr>
<td>OUTPT</td>
<td>247 252 370 531 535 612 616 675 831 # 967 1071 1092</td>
</tr>
<tr>
<td>PCRV</td>
<td>1239 2545</td>
</tr>
<tr>
<td>PCLV</td>
<td>1240 2547</td>
</tr>
<tr>
<td>PERIO</td>
<td>128 # 330 352 360 385 399 505 582</td>
</tr>
<tr>
<td>PFRMT</td>
<td>129 # 271</td>
</tr>
<tr>
<td>PRT10</td>
<td>1337 # 1961</td>
</tr>
<tr>
<td>FVAL</td>
<td>2167 2172 2190</td>
</tr>
<tr>
<td>PSAV</td>
<td>177 363 398 406 439 586 1017 1513 2498 2543</td>
</tr>
<tr>
<td>RAMST</td>
<td>93 # 12</td>
</tr>
<tr>
<td>RDK10</td>
<td>887 890</td>
</tr>
<tr>
<td>RDKBD</td>
<td>272 351 384 652 882 # 889 1037</td>
</tr>
<tr>
<td>READ</td>
<td>130 # 755</td>
</tr>
<tr>
<td>REGS</td>
<td>2214 # 2231</td>
</tr>
<tr>
<td>REG10</td>
<td>2218 2221</td>
</tr>
<tr>
<td>REG15</td>
<td>2233 2237</td>
</tr>
<tr>
<td>REGDS</td>
<td>1664 2212</td>
</tr>
<tr>
<td>RES10</td>
<td>185 223</td>
</tr>
<tr>
<td>RETF</td>
<td>686 808 906 # 1039 1044</td>
</tr>
<tr>
<td>RETT</td>
<td>811 922 # 1051</td>
</tr>
<tr>
<td>RGA05</td>
<td>2234 # 2271</td>
</tr>
<tr>
<td>RGA10</td>
<td>2269 2272</td>
</tr>
<tr>
<td>RGDAR</td>
<td>1668 2281</td>
</tr>
<tr>
<td>ROLCC</td>
<td>318 327 936</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>BGNAM</td>
<td>317 957#</td>
</tr>
<tr>
<td>RGPTB</td>
<td>1047 1180#</td>
</tr>
<tr>
<td>RGPTR</td>
<td>806 810 937 958 1050 2559#</td>
</tr>
<tr>
<td>RGTRL</td>
<td>939 1227# 1241</td>
</tr>
<tr>
<td>RMUSE</td>
<td>58# 124 137</td>
</tr>
<tr>
<td>RSET5</td>
<td>197 2518#</td>
</tr>
<tr>
<td>RSET6</td>
<td>207 2519#</td>
</tr>
<tr>
<td>RSET7</td>
<td>217 2521#</td>
</tr>
<tr>
<td>RSR05</td>
<td>1001 1004#</td>
</tr>
<tr>
<td>RSRT0</td>
<td>1003 1006 1009#</td>
</tr>
<tr>
<td>RSRT1</td>
<td>212 2520#</td>
</tr>
<tr>
<td>RSRTB</td>
<td>371 427 933# 1523</td>
</tr>
<tr>
<td>RTAB</td>
<td>2213 2262 2462# 2466</td>
</tr>
<tr>
<td>RTABT</td>
<td>1723 2263 2466#</td>
</tr>
<tr>
<td>SCM05</td>
<td>1626# 1643</td>
</tr>
<tr>
<td>SCM10</td>
<td>1629 1632#</td>
</tr>
<tr>
<td>SCM15</td>
<td>1639 1641#</td>
</tr>
<tr>
<td>SCMD</td>
<td>1622# 2429</td>
</tr>
<tr>
<td>SDK5</td>
<td>71</td>
</tr>
<tr>
<td>SETRG</td>
<td>311 1036#</td>
</tr>
<tr>
<td>SGNAD</td>
<td>245 1175#</td>
</tr>
<tr>
<td>SGNVT</td>
<td>250 1176#</td>
</tr>
<tr>
<td>SGWON</td>
<td>1386 2422# 2424</td>
</tr>
<tr>
<td>SKLN</td>
<td>101# 137</td>
</tr>
<tr>
<td>SFHST</td>
<td>1237 2548#</td>
</tr>
<tr>
<td>SPLST</td>
<td>1238 2547#</td>
</tr>
<tr>
<td>SRET</td>
<td>2007 2141 2291# 2380 2381 2386 2405 2407 2409</td>
</tr>
<tr>
<td>SSTEP</td>
<td>382# 457 1115</td>
</tr>
<tr>
<td>SSTRT</td>
<td>1341# 1828</td>
</tr>
<tr>
<td>STH05</td>
<td>2342 2348#</td>
</tr>
<tr>
<td>STHFO</td>
<td>1568 1571 2309#</td>
</tr>
<tr>
<td>STHFL</td>
<td>1558 2314 2334#</td>
</tr>
<tr>
<td>STOPB</td>
<td>1342# 153</td>
</tr>
<tr>
<td>STP20</td>
<td>388 402#</td>
</tr>
<tr>
<td>STP21</td>
<td>409 412#</td>
</tr>
<tr>
<td>STP22</td>
<td>411 416#</td>
</tr>
<tr>
<td>STP23</td>
<td>414 418#</td>
</tr>
<tr>
<td>STP25</td>
<td>192 429#</td>
</tr>
<tr>
<td>STAT</td>
<td>134# 1819</td>
</tr>
<tr>
<td>SUB05</td>
<td>476# 503</td>
</tr>
<tr>
<td>SUB10</td>
<td>490 498#</td>
</tr>
<tr>
<td>SUB15</td>
<td>478 504#</td>
</tr>
<tr>
<td>SUBST</td>
<td>467# 1117</td>
</tr>
<tr>
<td>TEMP</td>
<td>405 417 452 1541 1559 1565 1674 1675 1713 1729 2310 2340 2557#</td>
</tr>
<tr>
<td>TERM</td>
<td>1344# 1548</td>
</tr>
<tr>
<td>TIM2</td>
<td>1352# 1835</td>
</tr>
<tr>
<td>TIMER</td>
<td>1400# 419 422</td>
</tr>
<tr>
<td>TMIH1</td>
<td>132# 421</td>
</tr>
<tr>
<td>TMLC</td>
<td>133# 423</td>
</tr>
<tr>
<td>TMODE</td>
<td>134# 424</td>
</tr>
<tr>
<td>TSTTR</td>
<td>152# 337 1551</td>
</tr>
<tr>
<td>TSTRT</td>
<td>135# 425</td>
</tr>
<tr>
<td>UBLN</td>
<td>103# 137</td>
</tr>
<tr>
<td>UMMSK</td>
<td>136# 227 455</td>
</tr>
<tr>
<td>UPDAD</td>
<td>460 591 1084#</td>
</tr>
<tr>
<td>UPDDT</td>
<td>322 485 593 1085#</td>
</tr>
</tbody>
</table>
ISIS-II ASSEMBLER SYMOL CROSS REFERENCE, X108

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
WATTS 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#
XCM35 1557# 2428
ZERO 1128# 1175

CROSS REFERENCE COMPLETE