SDK-85
System Design Kit
User's Manual

Manual Order Number 9800451B
SDK-85
System Design Kit
User's Manual

Manual Order Number 9800451B

Copyright © 1978 Intel Corporation

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051
Additional copies of this manual or other Intel literature may be obtained from:

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

The information in this document 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 document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9(a)(9).

No part of this document 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:

i
ICE
ICS
Intel
Intelevision
Intelic

iSBC
Library Manager
MCS
Megachassis
Micromap
Multibus

Multimode
PROMPT
Promware
RMX
UPI
μScope

and the combination of ICE, iCS, iSBC, MCS, or RMX and a numerical suffix.
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 ............. 2-6
SOLDERING TECHNIQUES
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 8085A cpu (1.3 μs instruction cycle)
- Popular 8080A Instruction Set
- Direct Teletypewriter Interface
- Interactive LED Display
- Large Wire-Wrap Area for Custom-Designed Circuit
- System Monitor Software in ROM

SDK-85 SPECIFICATIONS

Central Processor

CPU: 8085A
Instruction Cycle: 1.3 microsecond
$T_{cy} = 330$ ns

Memory

ROM: 2K bytes (expandable to 4K bytes)
8355 or 8755
RAM: 256 bytes (expandable to 512 bytes) 8155
Addressing: ROM 0000-07FF (expandable to 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.

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.
Interrupts

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

DMA

Hold Request: Jumper selectable. TTL compatible input.

Software

System Monitor: Preprogrammed 8755 or 8355 ROM
Addresses: 0000-07FF
I/O: Keyboard/Display or TTY (serial I/O)

Literature

Design Library (Provided with kit):
- SDK-85 User’s Manual
- Microcomputer Systems Databook
- MCS-85 User’s Manual
- 8080/8085 Assembly Language Programming Manual

Physical Characteristics

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

Electrical Characteristics (DC Power Required)

$V_{CC} : +5V \pm 5\%$  
$V_{TTY} : -10V \pm 10\%$  
($V_{TTY}$ required only if teletypewriter is to be connected to the kit)

Environmental

Operating Temperature: 0-55°C

Figure 1-2. Finished Computer
CHAPTER 2
HOW TO ASSEMBLE THE KIT

2-1 GENERAL

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

**CAUTION**

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

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

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

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

2-2 GETTING ORGANIZED

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

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

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

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

CAUTION

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

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

- Red plastic window (covered with protective paper)
- Two strips of double-coated adhesive tape
Next, open the bag of electrical parts and sort them out by type and value. Give yourself plenty of unobstructed work space and try not to let tiny parts skitter away from you. The bag should yield the following:

**Resistors, 1/4 Watt**

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

**Resistor, 1/2 Watt**

- 1 100 Ohm (brown-black-brown)  
  R1

**Resistors, 1 Watt**

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

**Capacitor, tantalum**

- 1 22 μf, 15V  
  C1

**Capacitor, mono**

- 2 1 μf, 25V  
  C5, 20

---

**Resistor Color Code**

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

The first three bands employ the color code below:

<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>
</tbody>
</table>

The fourth band indicates percentage tolerance of the resistor value.

- First significant digit
- Second significant digit
- Number of following zeroes
- Gold = 5%; silver = 10% tolerance
Capacitor, ceramic

7 0.1 μf  C11-16, 18

Transistor

16 2N2907 transistors  Q1-16

Crystal, clock

1 6.144 MHz  Y1

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

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

6 alphanumeric LED (light-emitting diode) displays

DS1-6

24 pushbutton switches, with keycaps labeled

S1-24

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

Large, 40-pin ICs (integrated circuits)

1 8085A microprocessor (cpu)  A11

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

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

1 8279 keyboard/display interface  A13
Small, 16-pin ICs

- 1 8205 address decoder
- 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.

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.
Place capacitor C1 near the top edge of the board.
Place C1 in place. Clip excess lead ends.

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

**Assembly of TTY Interface Area—**

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

- Install a 430 Ohm resistor (yellow-orange-brown) at R35.
- Solder these two resistors in place, then clip their excess lead ends.
- Install transistors Q1 and Q2, and solder and clip them.
Assembly of Processing Area

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

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

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

- A11, for the 8085 cpu.
- A14, for the PROM (ROM)-I/O device, an 8755 or 8355.

- A16, for the RAM-I/O device, an 8155.
- Solder the three sockets in, and check carefully for solder bridges.
- Install a 3k Ohm resistor (orange-black-red) at R7.
- Install a 3.9k Ohm resistor (orange-white-red) at R8.
- Solder these two resistors and clip off their lead ends.

Install three 0.1 uf ceramic capacitors at:
- C11
- C12
- C13
- Solder them and clip off excess lead length.
- Install a 1 uf capacitor at C20.
- Install a 1k resistor (brown-black-red) at R31.
- Install a 51k resistor (green-brown-orange) at R32.
- Install a 200 Ohm resistor (red-black-brown) at R33.
- Solder these four components in place and trim their leads.

Install 0.1 uf ceramic capacitors at:
- C16
- C18
- Now solder the capacitors you have installed, and clip off their excess lead ends.
Assembly of Keyboard and Display Area

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

- R9
- R12
- R15
- R18
- R21
- R24
- R28 (Careful—the location pattern changes here!)
- R29

Now solder all eight resistors in place and clip their excess lead ends.
Install six 270 Ohm resistors (red-violet-brown) at:

- R10
- R13
- R16
- R19
- R22
- R25

Solder these six resistors and clip their excess lead ends.
Install eight 24 Ohm resistors (red-yellow-black) at:

- R11
- R14
- R17
- R20
- R23
- R26
- R27 (Again, note the change in location pattern.)
- R30

Solder these eight resistors and clip their excess lead ends.
Install fourteen 2N2907 transistors in two rows. Position the seven transistors in the top row so that their indexing tabs point upward and to the left, at:

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

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

- Q10
- Q11
- Q12
- Q13
- Q14
- Q15
- Q16

Press all of the transistors down to about 1/8 inch from the surface of the board. Let them stand approximately straight up. Then, turn the board over and solder all of their leads in place and trim the lead ends.
- Install one of the 40-pin DIP sockets, for the 8279 Keyboard-Display Controller, at A13, and solder it in.
- Install the 74LS156 scan decoder at A12, and solder it.

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

- DS1
- DS2
- DS3
- DS4
- DS5
- DS6

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

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

Install two 0.1 uf ceramic capacitors at:

- C14
- C15
- Solder the leads and clip them off close to the board.
The easiest method of doing this is to insert each button in its turn, bend its leads over on the back of the board to hold it in place, and go on until all buttons are in place, then solder all of them in one pass, with the board lying flat on the work surface and weighted down to make sure the switches are uniformly held firmly against the front surface of the board.

- Reset
- Vect Intr
- C
- D
- E
- F
- Single Step
- Go
- H
- 9
- A
- B
- Subst Mem
- Exam Reg
- 4
- 5
- 6
- 7
- Pch
- Pcl
- Next
- Exec
- 0
- 1
- 2
- 3
- 1

- All soldered in place

□ 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.
CHAPTER 3
FINAL ASSEMBLY AND CHECKOUT

3-1 GENERAL

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

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

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

3-2 STRAPPING INSTRUCTIONS

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

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

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

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

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

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

3-3 POWER SUPPLY WIRING

(See Figure 3-6.)

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

(Text continues on page 3-4.)
TABLE 3-1
ROM/PROM STRAPPING

<table>
<thead>
<tr>
<th>Device Location</th>
<th>8355</th>
<th>8755</th>
<th>8755A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A14 (The SDK-85 Monitor ROM)</td>
<td>No Straps Required</td>
<td>Strap 28-29</td>
<td>Strap 29-30</td>
</tr>
<tr>
<td>A15</td>
<td>Strap 31-32</td>
<td>Strap 32-33</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 3-2
TELETYPEWRITER-KEYBOARD STRAPPING

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

*Note: No devices may be installed in positions A13, A14, A15, A16, and A17 if this option is strapped.
3-5 STARTING THE FIRST TIME

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

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

Energize the +5 Volt power supply.

Press the RUN button on the keyboard. The display should respond by reading out “– 80 85.”

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

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

CAUTION

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

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

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

If there is no response to the `reset` command,

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

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

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

The numbers are:

(800)  538-8014 when calling from out-
(800)  538-8015 side California
(800)  672-3507 California only

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

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

### Table 3-5

**Interface Connector J7**

**PIN ASSIGNMENTS**

<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>2</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>Open (Ground)</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>12</td>
<td>RX-</td>
<td>Receive Return (−)</td>
</tr>
<tr>
<td>13</td>
<td>RX+</td>
<td>Receive (+)</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>17</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>19</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>24</td>
<td>TX-</td>
<td>Transmit Return (−)</td>
</tr>
<tr>
<td>25</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>
<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>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>27</td>
<td>28</td>
<td>—</td>
<td>Buffered D6</td>
<td>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>29</td>
<td>30</td>
<td>—</td>
<td>Buffered D5</td>
<td>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>31</td>
<td>32</td>
<td>—</td>
<td>Buffered D4</td>
<td>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>33</td>
<td>34</td>
<td>—</td>
<td>Buffered D3</td>
<td>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>35</td>
<td>36</td>
<td>—</td>
<td>Buffered D2</td>
<td>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>37</td>
<td>38</td>
<td>—</td>
<td>Buffered D1</td>
<td>1/O</td>
</tr>
<tr>
<td>GND</td>
<td>39</td>
<td>40</td>
<td>D0</td>
<td>Buffered D0</td>
<td>1/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</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>PORT 0</td>
<td>P0-5</td>
</tr>
<tr>
<td>P0-2</td>
<td>13</td>
<td>14</td>
<td></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>
<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>P2BH-5</td>
</tr>
<tr>
<td>P2BH-2</td>
<td>3</td>
<td>4</td>
<td>PORT 2BH</td>
<td>P2BH-3</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 OUT</td>
<td>23</td>
<td>24</td>
<td>TIMER OUT/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>Spectra Strip</td>
<td>800-576</td>
</tr>
<tr>
<td>J2</td>
<td>Bus Expansion</td>
<td>40</td>
<td>Spectra Strip</td>
<td>800-576</td>
</tr>
<tr>
<td>J3</td>
<td>I/O Ports</td>
<td>34</td>
<td>Spectra Strip</td>
<td>800-579</td>
</tr>
<tr>
<td>J4</td>
<td>I/O Ports</td>
<td>26</td>
<td>Spectra Strip</td>
<td>800-583</td>
</tr>
<tr>
<td>J5</td>
<td>I/O Ports and Timer</td>
<td>26</td>
<td>Spectra Strip</td>
<td>800-583</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></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>Molex</td>
<td>Model No. 1261</td>
</tr>
<tr>
<td></td>
<td>Recepticle</td>
<td></td>
<td></td>
<td>03-09-1064</td>
</tr>
<tr>
<td></td>
<td>Plug</td>
<td></td>
<td></td>
<td>03-09-2062</td>
</tr>
</tbody>
</table>
CHAPTER 4
OPERATING INSTRUCTIONS

4-1  WHAT IT DOES

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

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

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

Two of the keyboard buttons continue to function in teletypewriter mode, as well as in keyboard/display mode. These are the [SET] and the [RST] 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 [SET] 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>
<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 NEXT 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> NEXT <data> NEXT <data> ... EXEC`

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

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

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

Pressing NEXT while the address FFFF is being displayed will cause address 0000 to be displayed.
Whenever the command changes the contents of a memory location, it also verifies that the change has occurred correctly. If the contents of the location do not agree with what the new value should be (i.e., if the memory location is in ROM or is nonexistent), an error message is generated.

**SUBSTITUTE MEMORY EXAMPLE 1**

Using $\text{SUIT MEM}$ to list the first few Monitor locations:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{SUIT MEM}$</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>0</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>

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

After loading the above program, use $\text{SUIT MEM}$ again to go back and check locations 2000-2002 to see that they contain:

<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.

**SUBSTITUTE MEMORY EXAMPLE 2**

Using $\text{SUIT MEM}$ to enter a small program:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{SUIT MEM}$</td>
<td>.</td>
<td></td>
</tr>
<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>6</td>
<td>2000</td>
<td>3E.</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>4</td>
<td>2001</td>
<td>47.</td>
</tr>
<tr>
<td>NEXT</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>
<tr>
<td>EXEC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

\[
\text{EXAM} \quad \text{<reg> EX} \quad \text{<data>} \quad \text{EX} \quad \text{<data>} \quad \ldots \quad \text{EX}
\]

The examine command allows you to display and modify the contents of the 8085 CPU registers. Pressing the \text{EXAM} 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 \text{EX} or \text{EX} key is pressed.

Pressing \text{EX} 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 \text{EX} will place the contents displayed in the data field in the register named in the address field, and will also terminate the command.

Pressing \text{EX} while register PCL is being displayed has the same effect as pressing \text{EX}.

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:

\[
\begin{array}{cccccc}
7 & 6 & 5 & 4 & 3 & 2 \\
0 & 0 & 0 & 0 & E & I \\
\end{array}
\]

\text{interrupt mask}

\text{interrupt enable flag}

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & E & I \\
7.5 & 6.5 & 5.5 \\
\end{array}
\]

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

The format for the flag byte is:

\[
\begin{array}{cccccccc}
S & Z & X & A & C & X & P & X \\
\end{array}
\]

\text{carry}

\text{parity}

\text{aux carry}

\text{zero}

\text{sign}

\text{x = undefined}

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

TABLE 4-2

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

The format for the flag byte is:

\[
\begin{array}{cccccccc}
S & Z & X & A & C & X & P & X \\
\end{array}
\]

\text{carry}

\text{parity}

\text{aux carry}

\text{zero}

\text{sign}

\text{x = undefined}

4-4
### EXAMINE REGISTER EXAMPLE 1

Using **RAM** to initialize the 8085's stack pointer to 20C2:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 SPH</td>
<td></td>
<td>**.</td>
</tr>
<tr>
<td>2 SPH</td>
<td></td>
<td>02.</td>
</tr>
<tr>
<td>0 SPH</td>
<td></td>
<td>20.</td>
</tr>
<tr>
<td><strong>NEXT</strong></td>
<td></td>
<td>**.</td>
</tr>
<tr>
<td>C SPL</td>
<td></td>
<td>0C.</td>
</tr>
<tr>
<td>2 SPL</td>
<td></td>
<td>C2.</td>
</tr>
</tbody>
</table>

### EXAMINE REGISTER EXAMPLE 2

Using **EXE** 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><strong>EXE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></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>SPH</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>
</tbody>
</table>

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

`@ (<address>) [EXEC]`

Pressing the `@` 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 `[RUN]` or after execution of an RST 0, RST 1, or JMP 0 instruction in program.

**IMPORTANT:**

Note that because of the way the GO and SINGLE STEP commands are implemented in the Monitor, `[@]` 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 C2 using `[SPL]` , then try it again. (Locations 20C2 to 20FF are reserved for the monitor program, therefore the stack pointer must be set to 20C2 or lower so as not to interfere with the monitor.)
GO COMMAND EXAMPLE

Now you can execute the program you entered in Example 2 of the \texttt{LOAD} command. First, check to make sure the 3-location program is in memory, then the program will be executed.

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBR</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0002.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0020.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0200.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2000.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEXT</td>
<td>2000</td>
<td>3E.</td>
<td>MVI A, 47</td>
</tr>
<tr>
<td>NEXT</td>
<td>2001</td>
<td>47.</td>
<td></td>
</tr>
<tr>
<td>NEXT</td>
<td>2002</td>
<td>CF.</td>
<td>RST 1</td>
</tr>
<tr>
<td>EXEC</td>
<td>–</td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>****.</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0002.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0020.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0200.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2000.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXEC</td>
<td>– 80 85</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\textbf{NOTE:} **** denotes "don't care" values

Recall that this small program loads the A register with the number 47 and restarts the monitor. To verify that the A register now holds 47 and to get more practice using \texttt{LOAD} and \texttt{EXEC}, try the following sequence:

<table>
<thead>
<tr>
<th>KEY</th>
<th>ADDR</th>
<th>DATA</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAM</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>47.</td>
<td>A reg now holds 47.</td>
</tr>
<tr>
<td>0</td>
<td>A</td>
<td>00.</td>
<td></td>
</tr>
<tr>
<td>EXEC</td>
<td>–</td>
<td></td>
<td>Now A holds 0</td>
</tr>
<tr>
<td>0</td>
<td>0002.</td>
<td></td>
<td>Run the small Program again</td>
</tr>
<tr>
<td>0</td>
<td>0020.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2000.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXEC</td>
<td>– 80 85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>47</td>
<td>Now A holds 47 again</td>
</tr>
</tbody>
</table>

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

\texttt{SINGLE STEP} \ (<\text{address}>\>) \ \texttt{NEXT} \ \texttt{NEXT} \ldots \ \texttt{EXEC}

Pressing the \texttt{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 \texttt{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 \texttt{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 \texttt{SINGLE STEP} key takes you back to the single step mode, and subsequent pressing of the \texttt{NEXT} key allows you to continue, instruction by instruction, through your program.

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

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\textbf{KEY} & \textbf{ADDR} & \textbf{DATA} \\
\hline
\texttt{SINGLE STEP} & ****. & ** \\
0 & 0008. & \\
\texttt{NEXT} & 000b. & E1 \\
\texttt{NEXT} & 000C. & 22 \\
\texttt{NEXT} & 000F. & F5 \\
\texttt{NEXT} & 0010. & E1 \\
\hline
\end{tabular}
\end{center}

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

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\texttt{EXEC} & \_ & \\
05 & 0010. & E1 \\
\texttt{EXEC} & \_ 80 & 85 \\
\hline
\end{tabular}
\end{center}
Vector Interrupt:

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

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

\textbf{IMPORTANT:} Two conditions must be satisfied for the Vector Interrupt feature to be enabled:

1. Interrupts must be enabled (by executing an \textit{EI} instruction).
2. RST 7.5 must be unmasked (mask reset by the \textit{SIM} instruction or by modifying the I-Register).

Program Debugging — The Use of Breakpoints

Along with the “cold start” reset caused when the \textit{Next} 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 \textit{Next} (‘‘--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 \textit{Next}. The PC value of the next instruction appears in the address field of the display. Then press \textit{Exec} to continue execution.

\textbf{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 \textit{Next} or \textit{Exec} by pressing any illegal key instead.

\textbf{Memory Substitution Errors}

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

\textbf{4-3 TELETYPETRITER OPERATION}

\textbf{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 \textit{Next} 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.

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

**Display Memory Command, D:**

\[ D \langle\text{low address}\rangle, \langle\text{high address}\rangle \]

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 \langle\text{low address}\rangle and \langle\text{high address}\rangle, inclusive, on the console. Each line of the listing begins with the address of the first memory location displayed on that line, represented as 4 hexadecimal digits, followed by up to 16 memory locations, each one represented by 2 hexadecimal digits.

**Program Execute Command, G:**

\[ G \langle\text{entry point}\rangle \]

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.
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.

<table>
<thead>
<tr>
<th>I COMMAND EXAMPLE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2010</td>
</tr>
<tr>
<td>1122344566778899$</td>
</tr>
<tr>
<td>This command puts the following pattern into RAM:</td>
</tr>
<tr>
<td>2010 11 22 33 44 55 66 77 88 99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I COMMAND EXAMPLE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2040</td>
</tr>
<tr>
<td>123456789$</td>
</tr>
<tr>
<td>This command puts the following pattern into RAM:</td>
</tr>
<tr>
<td>2040 12 34 56 78 90</td>
</tr>
<tr>
<td>Note that since an odd number of hexadecimal digits was entered initially, a zero was appended to the digit string.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>M COMMAND EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2010, 204F, 2050</td>
</tr>
<tr>
<td>64 bytes of memory are moved from 2010-204F to 2050-208F by this command.</td>
</tr>
</tbody>
</table>
Substitute Memory Command, S:

\[ S \text{ <address> (<data>+)} \]

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:

\[ X \text{ (<register identifier>+)} \]

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

### TABLE 4-3

<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  X  P  X  C
```

- CARRY
- PARITY
- AUX CARRY
- ZERO
- SIGN
- X = UNDEFINED

The format for the I register:

```
 7 6 5 4 3 2 1 0
0 0 0 0 1 I M 7.5 M 6.5 M 5.5
```

- INTERRUPT MASK
- 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.
CHAPTER 5
THE HARDWARE

5-1 OVERVIEW

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

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

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

5-2 SYSTEM COMPONENTS

The 8085 CPU & The System Buses

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

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

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

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

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

The 8155

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

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

The 8155’s timer is used by the SDK-85 monitor’s Single Step routine to interrupt the processor following the execution of each instruction.
The 8355 & 8755

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

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

The 8279

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

The 8205

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

---

Figure 5-1 SDK-85 Functional Block Diagram
### TABLE 5-1
8205 CHIP ENABLES

<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>1800-1FFF</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>2800-2FFF</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>3800-3FFF</td>
<td>N/C</td>
</tr>
</tbody>
</table>

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

### 5-3 SDK-85 MEMORY ADDRESSING

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>LOC.</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20C2</td>
<td>User may place a JMP instr. to a RST 5 routine in locs 20C2 — 20C4.</td>
</tr>
<tr>
<td>20C5</td>
<td>JMP to RST 6 routine</td>
</tr>
<tr>
<td>20C8</td>
<td>JMP to RST 6.5 routine (hardwired user interrupt)</td>
</tr>
<tr>
<td>20C8</td>
<td>JMP to RST 7 routine</td>
</tr>
<tr>
<td>20CE</td>
<td>JMP to “VECT INTR” key routine</td>
</tr>
<tr>
<td>20D1-20E8</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</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.

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

## 5-4 INPUT/OUTPUT PORT AND PERIPHERAL DEVICE ADDRESSING

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

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

**Hardware Note:** The timer/clock 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.

Figure 5-3 Data Format

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

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

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

Figure 5-4 Display Configuration

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>
<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</td>
</tr>
<tr>
<td>5</td>
<td>1</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>
<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.
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.

ADDITIONAL INTERFACES

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

Figure 5-5 Modification for RS-232c Operation
CHAPTER 6
THE SOFTWARE

6-1 THE SDK-85 MONITOR

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

6-2 PROGRAMMING HINTS

Stack Pointer

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

It is important always to define the stack pointer at the beginning of your program to avoid storing data in the wrong place. Locations 20C2 through 20D0 in RAM are reserved by the monitor for jump instructions when all interrupts are used. Thus, you should set the stack pointer initially at 20C2 (by the use of the program instruction LXI SP, 20C2H (31 C2 20), the keyboard command \( \text{ENTER} \) \( \frac{4}{20} \) \( \text{C} \) \( \frac{C}{2} \) \( \text{ENTER} \), or the teletypewriter "XS" command) in order to keep your own stack clear of data and programs you want to protect. If less than the full complement of interrupts is utilized, some or all of the unused space above 20C2 can be allocated to stack as described above. Remember that the stack must still occupy an unbroken string of contiguous memory locations.

RAM-I/O Command Status Register (CSR)

The basic 8155 command status register (port 20) is used to set up the on-chip I/O ports and timer. It can only be written to; it cannot be read. You can write to this register in your programs, but there is a precaution you should take: at any time when you write to the CSR in the basic RAM, you should also write the same pattern to RAM location 20FF. The reason is this: The \( \text{ESC} \) 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            | NMOUT    | Hex Number Printer  
NMOUT converts the 8-bit unsigned integer in the A register into 2 ASCII characters representing the 2 hex digits and prints the two digits on the teletypewriter. The contents of the A, B and C registers and the condition code flags are affected. |
| 0363            | UPDAD    | Update Address  
Update address field of the display. The contents of the D-E register pair are displayed in the address field of the display. The contents of all the CPU registers and flags are affected. |
| 036E            | UPDDT    | Update Data  
Update data field of the display. The contents of the A register are displayed in hex notation in the data field of the display. The contents of all of the CPU registers and flags are affected. |
| 02E7            | RDKBD    | Read Keyboard  
This routine waits until a character is entered on the hex keypad and upon return places the value of the character in the A register. The A, H, and L registers and the flag flip flops are affected.  
NOTE: For RDKBD to work correctly, you must first:  
1. Unmask RST 5.5 using the SIM instruction. |
| 05F1            | DELAY    | Time Delay  
This routine takes the 16-bit contents of register pair DE and counts down to zero, then returns to the calling program. The A, D, and E registers and the flags are affected. |
### TABLE 6-1
**MONITOR ROUTINE CALLING ADDRESSES (CONT’D)**

<table>
<thead>
<tr>
<th>Calling Address</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>02B7</td>
<td>OUTPT</td>
<td><strong>Output Characters to Display</strong>&lt;br&gt;The routine sends characters to the display with the parameters set up by registers A, B, H and L.&lt;br&gt;Reg A = 0 = use address field&lt;br&gt; = 1 = use data field&lt;br&gt;Reg B = 0 = decimal point off&lt;br&gt; = 1 = decimal point at right edge of field&lt;br&gt;Reg HL = starting address of characters to to sent.</td>
</tr>
</tbody>
</table>

<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>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
</tr>
<tr>
<td>A</td>
<td>0A</td>
</tr>
<tr>
<td>b</td>
<td>0B</td>
</tr>
<tr>
<td>C</td>
<td>0C</td>
</tr>
<tr>
<td>d</td>
<td>0D</td>
</tr>
<tr>
<td>E</td>
<td>0E</td>
</tr>
<tr>
<td>F</td>
<td>0F</td>
</tr>
<tr>
<td>H</td>
<td>10</td>
</tr>
<tr>
<td>L</td>
<td>11</td>
</tr>
<tr>
<td>P</td>
<td>12</td>
</tr>
<tr>
<td>r</td>
<td>13</td>
</tr>
<tr>
<td>S</td>
<td>14</td>
</tr>
<tr>
<td>Blank</td>
<td>05</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>
PROGRAM EXAMPLE – RDKBD

After executing 0000, 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 [HLM] to see that the key value is now in the A register.

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DATA</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31</td>
<td>LXI SP, 20C2H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2001</td>
<td>C2</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>CD</td>
<td>CALL RDKBD</td>
<td>; read keyboard value</td>
</tr>
<tr>
<td>2007</td>
<td>E7</td>
<td></td>
<td>; into Reg A</td>
</tr>
<tr>
<td>2008</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</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>ADDRESS</th>
<th>DATA</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31</td>
<td>LXI SP, 20C2H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2001</td>
<td>C2</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 [HLM] 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>ADDRESS</th>
<th>DATA</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31C220</td>
<td>LXI SP, 20C2H</td>
<td>; define stack pointer</td>
</tr>
<tr>
<td>2003</td>
<td>3E08</td>
<td>MVI A, 08H</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>30</td>
<td>SIM</td>
<td>; unmask interrupt</td>
</tr>
<tr>
<td>2006</td>
<td>CDE702</td>
<td>LOOP: CALL RDKBD</td>
<td>; read keyboard value into Reg A</td>
</tr>
<tr>
<td>2009</td>
<td>CD6E03</td>
<td>CALL UPDDT</td>
<td>; output Reg A to data field</td>
</tr>
<tr>
<td>200C</td>
<td>C30620</td>
<td>JMP LOOP</td>
<td>; keep looping</td>
</tr>
</tbody>
</table>
### PROGRAM EXAMPLE – ADD TWO NUMBERS IN HEX NOTATION

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

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

Note: for decimal (base 10) addition of digits 0-9, insert the DAA instruction (opcode 27) between ADD B and CALL UPDDT in the above program.

Additional Suggestion: Try modifying this program to perform 2-digit decimal number addition. (Hint: use the 8085's RLC instruction.)

### PROGRAM EXAMPLE – 4-DIGIT HEX COUNTER

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

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DATA</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>31C220</td>
<td>LXI SP 20C2</td>
<td>; initialize stack pointer</td>
</tr>
<tr>
<td>2003</td>
<td>13</td>
<td>LOOP: INX D</td>
<td>; add 1 to the 16-bit count</td>
</tr>
<tr>
<td>2004</td>
<td>D5</td>
<td>PUSH D</td>
<td>; save the count in the stack</td>
</tr>
<tr>
<td>2005</td>
<td>CD6303</td>
<td>CALL UPDAD</td>
<td>; display the count</td>
</tr>
<tr>
<td>2008</td>
<td>110018</td>
<td>LXID, 1800</td>
<td>; set delay count</td>
</tr>
<tr>
<td>200B</td>
<td>CD1F05</td>
<td>CALL DELAY</td>
<td>; wait out the delay</td>
</tr>
<tr>
<td>200E</td>
<td>D1</td>
<td>POP D</td>
<td>; restore the count to D &amp; E regs</td>
</tr>
<tr>
<td>200F</td>
<td>C30320</td>
<td>JMP LOOP</td>
<td>; keep counting</td>
</tr>
</tbody>
</table>
PROGRAM EXAMPLE – DECIMAL COUNTER

The following program displays a count in the data field of the display. The count may be stopped by pressing the [WB] button. The count resumes when any other key (except [MTR]) 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>DATA</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’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>20CE</td>
<td>FB</td>
<td>EI</td>
<td>CONTROL BRANCHES TO LOCATION</td>
</tr>
<tr>
<td>20CF</td>
<td>76</td>
<td>HLT</td>
<td>20CE WHEN VECT INTR IS Pressed.</td>
</tr>
<tr>
<td>20DO</td>
<td>C9</td>
<td>RET</td>
<td>WAIT HERE FOR KEY DEPRESSION.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESUME THE COUNT.</td>
</tr>
</tbody>
</table>

To execute the program, type in [20] 2000 [EXE].

Try to stop the count right at 00 using the [SLC] key.

Change the speed of the count by using [MTR] to vary the contents of location 2010.

Additional Suggestions:

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

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DATA</th>
<th>SYMBOLIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>200A</td>
<td>FE60</td>
<td>CPI A, 60</td>
<td>check to see if count = 60</td>
</tr>
<tr>
<td>200C</td>
<td>C21020</td>
<td>JNZ 2010</td>
<td>continue if count ≠ 60.</td>
</tr>
<tr>
<td>200D</td>
<td>AF</td>
<td>XRA A</td>
<td>if count = 60 then set the count = 0.</td>
</tr>
</tbody>
</table>

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

Additional Programming Idea: Expand on the digital stopwatch program by displaying hours and minutes in the address field of the LED display.
## PROGRAM EXAMPLE — FLASH HELP

Load into Locations 2000 through 2007 (use the Substitute Memory command) the following data: 10, 0E, 11, 12, 15, 15, 15, 15. Then load and execute the following program (\( w_{2010} \text{H} \)). 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>31C220</td>
<td>LXI SP, 20C2H</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>210620</td>
<td>LXI H, 2006H</td>
<td>; use characters starting</td>
</tr>
<tr>
<td>201A</td>
<td>CDB702</td>
<td>CALL OUTPT</td>
<td>; at Location 2006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>; output the two characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>; 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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>; at Location 2000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>; output the four characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>; to address field</td>
</tr>
<tr>
<td>2024</td>
<td>CDB702</td>
<td>CALL OUTPT</td>
<td>; set up delay value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>; time delay</td>
</tr>
<tr>
<td>2027</td>
<td>1FFFFF</td>
<td>LXI D, 0FFFFH*</td>
<td>; output BLANKS to</td>
</tr>
<tr>
<td>202A</td>
<td>CDF105</td>
<td>CALL DELAY</td>
<td>; Display</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>; display</td>
</tr>
<tr>
<td>2034</td>
<td>CDB702</td>
<td>CALL OUTPT</td>
<td>; time Delay</td>
</tr>
<tr>
<td>2037</td>
<td>1FFFFF</td>
<td>LXI D, 0FFFFH</td>
<td>;</td>
</tr>
<tr>
<td>203A</td>
<td>CDF105</td>
<td>CALL DELAY</td>
<td>;</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.

### Additional Suggestions:

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

In addition, you may place the hex codes from p. 6-3 for a 2-letter word (like "HI") in memory locations 2004 and 2005, and the 2-letter word will flash in between the flashes of the 4-letter word.
### PROGRAM EXAMPLE – USING THE 8155 AND 8355 DEVICE OUTPUT PORTS

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

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

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

```
1: ;******************************************************************************
2: 3: PROGRAM: SDK-85 MONITOR VER 2.1
4: 5: COPYRIGHT (C) 1977
6: 7: INTEL CORPORATION
8: 8: 3865 BOWERS AVENUE
9: 9: SANTA CLARA, CALIFORNIA 95051
10: ;******************************************************************************
11: 12: ABSTRACT
13: 14: -------
15: 16: THIS PROGRAM IS A SMALL MONITOR FOR THE INTEL 8865 KIT AND
17: PROVIDES A MINIMUM LEVEL OF UTILITY FUNCTIONS FOR THE USER EMPLOYING
18: EITHER AN INTERACTIVE CONSOLE (I.E. TELETYP) ON THE KIT'S
19: STANDARD/LED DISPLAY. THE KEYBOARD MONITOR ALLOWS THE USER TO PERFORM
20: SUCH FUNCTIONS AS MEMORY AND REGISTER MANIPULATION, PROGRAM LOADING,
21: PROGRAM EXECUTION, INTERRUPT OF AN EXECUTING PROGRAM, AND
22: SYSTEM RESET.
23: 24: PROGRAM ORGANIZATION
25: 26: -------------
27: 28: THE PROGRAM IS ORGANIZED AS FOLLOWS:
29: 30: 1) COLD START ROUTINE (RESET)
31: 2) WARM START - REGISTER SAVE ROUTINE
32: 3) INTERRUPT VECTORS
33: 4) KEYBOARD MONITOR
34: 5) TTY MONITOR
35: 6) LAYOUT OF RAM USAGE
36: 37: THE KEYBOARD MONITOR BEGINS WITH THE COMMAND RECOGNIZER, FOLLOWED BY
38: THE COMMAND ROUTINE SECTION, UTILITY ROUTINE SECTION AND MONITOR
39: TABLES. THE COMMAND AND UTILITY ROUTINES ARE IN ALPHABETICAL ORDER
40: WITHIN THEIR RESPECTIVE SECTIONS.
41: THROUGHOUT THE KEYBOARD MONITOR, A COMMENT FIELD BEGINNING
42: WITH "ARG - " INDICATES A STATEMENT WHICH LOADS A VALUE INTO
43: A REGISTER AS AN ARGUMENT FOR A FUNCTION. WHEN THE DESIRED VALUE
44: LIST OF KEYBOARD MONITOR ROUTINES
45: ***** **** ******** ********
46: 47: CMNNRD
48: 49: ------
50: 51: EXAM
52: GOCMD
53: SSTEP
54: SUBST
55: -----
56: CLEAR
57: CLDIS
58: CLODS
59: DISPC
60: ERR
61: GTEHX
62: HXDSP
63: INTN
64: INSDC
65: NAXTR
66: OUTPT
67: KRDIO
68: RETF
69: RETT
70: ROLOC
71: RSTOR
72: SPRINTF
73: UPDAD
74: UPDST
75: NAME SDK85
76: *****************
77: SET CONDITIONAL ASSEMBLY FLAG
78: 79: 80: WAITS SET 0 ;0=NO WAIT STATES
81: 1=M 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: MONITOR EQUATES
88: 89:
90: 91: RAMST EQU 20000 ; START ADDRESS OF RAM - THIS PROGRAM ASSUMES
92: THAT 256 BYTES OF RANDOM ACCESS MEMORY BEGIN AT THIS ADDRESS.
93: THE PROGRAM USES STORAGE AT THE END OF THIS SPACE FOR VARIABLES,
94: SAVING REGISTERS AND THE PROGRAM STACK
95:
96:
97:
```
LOC  OBJ  SEQ  SOURCE STATEMENT
#17  98  RMUSE  EQU  23  ; RAM USAGE - CURRENTLY, 23 BYTES ARE USED FOR
99  ; /SAVING REGISTERS AND VARIABLES
100  ;
101  SKLN  EQU  24  ; MONITOR STACK USAGE - MAX OF 12 LEVELS
102  ;
103  UBRLN  EQU  15  ; 5 USER BRANCHES - 3 BYTES EACH
104  ;
105  ADLFL  EQU  0  ; INDICATES USE OF ADDRESS FIELD OF DISPLAY
106  ; CONTROL CHARACTER TO INDICATE OUTPUT TO
107  ; ADDRESS FIELD OF DISPLAY
108  ;
109  CNTRL  EQU  1988  ; ADDRESS FOR SENDING CONTROL CHARACTERS TO
110  ; /DISPLAY CHIP
111  SLN  EQU  11H  ; SLN FROM KEYBOARD
112  CNTNI  EQU  1H  ; INITIAL VALUE FOR COMMAND STATUS REGISTER
113  SN  EQU  70H  ; OUTPUT PORT FOR COMMAND STATUS REGISTER
114  ;
115  CMAS  EQU  94H  ; CONTROL CHARACTER TO INDICATE OUTPUT TO
116  ; DATA FIELD OF DISPLAY
117  DTP  EQU  08H  ; CONTROL CHARACTER TO SET KEYBOARD/DISPLAY MODE
118  ; (2 KEY ROLLOVER, 8 CHARACTER LEFT ENTRY)
119  ;
120  KMSE  EQU  0  ; INDICATOR FOR NO DOT IN DISPLAY
121  ;
122  RMAS  EQU  256 - RMUSE ; START OF MONITOR STACK
123  ;
124  MNOD  EQU  0  ; INDICATOR FOR NO DOT IN DISPLAY
125  ;
126  PERIOD  EQU  100H  ; PERIOD FROM KEYBOARD
127  ;
128  RET  EQU  0FFH  ; PROMPT CHARACTER FOR DISPLAY (DASH)
129  ;
130  KEYBOARD  EQU  40H  ; CONTROL CHARACTER TO INDICATE INPUT FROM
131  ; KEYBOARD
132  TIMHI  EQU  25H  ; OUTPUT PORT FOR HIGH ORDER BYTE OF TIMER VALUE
133  ;
134  TIMLO  EQU  24H  ; OUTPUT PORT FOR LOW ORDER BYTE OF TIMER VALUE
135  ;
136  MTCM  EQU  40H  ; TIMER MODE - SQUARE WAVE, AUTO RELOAD
137  ;
138  STT  EQU  0CH  ; START TIMER
139  ;
140  UNMAS  EQU  0EH  ; UNMASK INPUT INTERRUPT
141  ;
142  USBR  EQU  255 - (RMUSE + SKLN + UBRLN) ; START OF USER
143  ; /BRANCH LOCATION
144  ;
145  IF    1-Waits ; TIMER VALUE FOR SINGLE STEP IF NO WAIT STATE
146  IF    1-Waits ; TIMER VALUE FOR SINGLE STEP IF NO WAIT STATE INSERTED
147  ;
148  IF    1-Waits ; TIMER VALUE FOR SINGLE STEP IF NO WAIT STATE INSERTED
149  ;
150  ;
151  ;
152  ;
153  ;
154  ;
155  ;
156  ;
157  ;
158  ;
159  ;
160  ;
161  ;
162  ;
163  ;
164  ;
165  ;
166  ;
167  ;
168  ;
169  ;
170  ;
171  ;
172  ;
173  ;
174  ;
175  ;
176  ;
177  ;
178  ;
179  ;
180  ;
181  ;
182  ;
183  ;
184  ;
185  ;
186  ;
187  ;
188  ;
189  ;
190  ;
191  ;
192  ;
193  ;
194  ;
195  ;
196  ;
197  ;
198  ;
199  ;
200  ;
201  ;
202  ;
0026 C3C2E0 197 JMP RSET6 ; BRANCH TO RST 5 LOCATION IN RAM
0026 C3C2E2 199 ; ***** INPUT INTERRUPT ENTRY POINT (RST 5.5)

002C 19A JMP ININT ; BRANCH TO INPUT INTERRUPT ROUTINE

00C8 C3C5B0 19C JMP RSET6 ; BRANCH TO RST 6 LOCATION IN RAM

0033 C3C520 19E ; ***** HARD WIRE USER INTERRUPT ENTRY POINT (RST 6.5)

003B 19F JMP RSET6 ; BRANCH TO RST 6 LOCATION IN RAM

003C C3C820 1A1 JMP RSET6 ; BRANCH TO RST 6.5 LOCATION IN RAM

003F 1A3 ; ***** RST 7 ENTRY POINT

0040 C3C8B0 1A5 JMP RSET7 ; BRANCH TO RST 7 LOCATION IN RAM

0043 C3C8E0 1A7 ; "VECTORED INTERRUPT" KEY ENTRY POINT (RST 7.5)

004C 1A9 JMP USINT ; BRANCH TO USER INTERRUPT LOCATION IN RAM

005F 1AA ; CONTINUE SAVING USER STATUS

0060 1AB RIM ; GET USER INTERRUPT STATUS AND INTERRUPT MASK

0064 32F100 1AC ANI 0FH ; KEEP STATUS & MASK BITS

0065 32F1EE 1A9 STA ISAV ; SAVE INTERRUPT STATUS & MASK

0069 32F130 1BB MVI A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE

006B 32F160 1BF SIM ; INTERRUPTS DISABLED WHILE MONITOR IS RUNNING

006D 32F190 1C3 DI ; (EXCEPT WHEN WAITING FOR INPUT)

006F 32F1C0 1C5 RIM ; TTY OR KEYBOARD MONITOR ?

0071 32F1E0 1C7 RLC ; IS TTY CONNECTED ?

0073 32F210 1C9 JC GO ; YES - BRANCH TO TTY MONITOR

0074 32F230 1CD NO - ENTER KEYBOARD MONITOR

0075 32F250 1DE ;***********************************************************

0076 32F270 1F0 BEGINNING OF KEYBOARD MONITOR CODE

0077 32F290 1F2 ;***********************************************************

0078 32F2B0 1F4 OUTPUT SIGN-ON MESSAGES

0079 32F2D0 1F6 XRA A ; ARG - USE ADDRESS FIELD OF DISPLAY

007A 32F300 1F8 MVI B,400H ; ARG - NO DOT IN ADDRESS FIELD

007B 32F330 1FA 21A683 MVI H,80H ; ARG - GET ADDRESS OF ADDRESS FIELD PORTION OF

007C 32F360 1FB 20C072 CALL OUTPT ; OUTPUT SIGN-ON MESSAGE TO ADDRESS FIELD

007D 32F390 1FD 20C7E1 MVI A,DPRD ; ARG - USE DATA FIELD OF DISPLAY

007E 32F3C0 1FE 20C980 MVI B,400H ; ARG - NO DOT IN DATA FIELD

007F 32F3F0 1FF 20C9F0 MVI H,80H ; ARG - GET ADDRESS OF DATA FIELD PORTION OF

0080 32F420 20C9F3 CALL OUTPT ; OUTPUT SIGN-ON MESSAGE TO DATA FIELD

0081 32F450 20CFB MVI A,430H ; ARG - EMPTY

0082 32F480 20D61 STA IBUFF ; SET INPUT BUFFER EMPTY FLAG

0083 32F4A0 20D74 ;***********************************************************

0084 32F4C0 20D76 ;***********************************************************

0085 32F4E0 20D78 ; FUNCTION: CMNND - COMMAND RECOGNIZER

0086 32F500 20D7A ; INPUTS: NONE

0087 32F520 20D7C ; OUTPUTS: NONE

0088 32F540 20D80 ; CALLS: RORRD,ERR,SUBST,EXAM,GOCMD,SSTEP

0089 32F560 20D82 ; DESTRYS: A,B,C,D,E,H,W,L,P,F/S

008A 32F580 20D85 ;***********************************************************

008B 32F5A0 20D87 CMNND:

008C 32F5C0 20D89 LXI H,MSDK ; INITIALIZE MONITOR STACK POINTER

008D 32F5E0 20D8B SPHL

008E 32F600 20D8D ; OUTPUT PROMPT CHARACTER TO DISPLAY

008F 32F620 20D8F LXI H,CHNL ; GET ADDRESS FOR CONTROL CHARACTER

0090 32F640 20D91 MVI A,DISP ; OUTPUT CONTROL CHARACTER TO USE ADDRESS FIELD

0091 32F670 20D93 DCR H ; ADDRESS FOR OUTPUT CHARACTER

0092 32F690 20D95 MVI H,PRMT ; OUTPUT PROMPT CHARACTER

0093 32F6B0 20D97 CALL ROKBD ; READ KEYBOARD

0094 32F6E0 20D99 LXI B,NUMC ; COUNTER FOR NUMBER OF COMMANDS IN C

0095 32F700 20D9B LXI H,CMDTB ; GET ADDRESS OF COMMAND TABLE

0096 32F720 20D9D 27C MD18:

0097 32F740 20D9F 276 CMP M ; RECOGNIZE THE COMMAND ?

0098 32F760 20DA1 277 JZ CMD15 ; YES - GO PROCESS IT

0099 32F780 20DA3 279 INX H ; NO - NEXT COMMAND TABLE ENTRY

009A 32F7A0 20DA5 27B DCR C ; END OF TABLE ?

009B 32F7C0 20DA7 27D JNZ CMD18 ; NO - GO CHECK NEXT ENTRY

009C 32F7E0 20DAD 281 JC GO ; YES - COMMAND UNKNOWN

009D 32F800 20DAB 283 JMP ERR ; DISPLAY ERROR MESSAGE AND GET ANOTHER COMMAND

009E 32F820 20DAD 285 LXI H,CMDAD ; GET ADDRESS OF COMMAND ADDRESS TABLE

009F 32F840 20DAB 287 DAC C ; ADJUST COMMAND COUNTER

00A0 32F860 20DAD 289 ; COUNTER ACTS AS POINTER TO COMMAND ADDRESS TABLE

00A1 32F880 20DAA 28B JZ ADD1 ; ADD POINTERS TO ADDRESS ADDRESS TWICE

00A2 32F8A0 20DAD 28D ADD1 ; TABLE HAS 12 BYTE ENTRIES

00A3 32F8C0 20DAB 28F MOV A,M ; GET LOW ORDER BYTE OF COMMAND ADDRESS

00A4 32F8E0 20DAD 291 MOV H,M ; GET HIGH ORDER BYTE OF COMMAND ADDRESS IN H

00A5 32F900 20DAD 293 MOV L,A ; PUT LOW ORDER BYTE IN L

00A6 32F920 20DAB 295 ; COMMAND ROUTINE ADDRESS IS NOW IN H & L

00A7 32F940 20DAD 297 PCHL ; BRANCH TO ADDRESS IN H & L

00A8 32F960 20DAD 299 ;***********************************************************
**LOC** 0013 **SEQ** 1

**SOURCE STATEMENT**


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

**INPUTS: NONE**

**OUTPUTS: NONE**

**CALLS: CLEAR,SETING,ERR,RGAM,RGLOC,UPDTP,GTENK,MXTNG**

**DESTROYS: A,B,C,D,E,H,L,P/F’S**

---

**EXAM: 329**

**MV1 B.DOT ; ARG - DOT IN ADDRESS FIELD OF DISPLAY**

**CALL CLEAR ; CLEAR DISPLAY**

**CALL SETRNG ; GET REGISTER DESIGNATOR FROM KEYBOARD AND**

**/SET REGISTER POINTER ACCORDINGLY**

**MV1 B.DOT ; WAS CHARACTER A REGISTER DESIGNATOR?**

**FALSE ERR ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND**

**JNC ERR**

**EXM05: 316**

**CALL RGAM ; OUTPUT REGISTER NAME TO ADDRESS FIELD**

**CALL RGLOC ; GET REGISTER SAVE LOCATION IN H & L**

**MOV A,M ; GET REGISTER CONTENTS**

**STA CURDOT ; STORE REGISTER CONTENTS AT CURRENT DATA**

**MV1 B.DOT ; ARG - DOT IN DATA FIELD**

**CALL UPDTP ; UPDATE DATA FIELD OF DISPLAY**

**MV1 B.DOTP ; ARG - USE DATA FIELD OF DISPLAY**

**CALL GTENK ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?**

**FALSE EXM10 ; NO - DO NOT UPDATE REGISTER CONTENTS**

**JNC EXM10**

**CALL RGLOC ; YES - GET REGISTER SAVE LOCATION IN H & L**

**MOV M,E ; UPDATE REGISTER CONTENTS**

**EXM10: 320**

**CPI PERIO ; WAS LAST CHARACTER A PERIOD ?**

**JS CLDIS ; YES - CLEAR DISPLAY AND TERMINATE COMMAND**

**CPI COMMA ; WAS LAST CHARACTER ',' ?**

**JNC EXM10 ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND**

**CALL MXTNG ; YES - ADVANCE REGISTER POINTER TO**

**/NEXT REGISTER**

**ANY MORE REGISTERS ?**

**TRUE EXM85 ; YES - CONTINUE PROCESSING WITH NEXT REGISTER**

**JMP CLDIS ; NO - CLEAR DISPLAY AND TERMINATE COMMAND**

---

**GCMD: 348**

**CALL DISPC ; DISPLAY USER PROGRAM COUNTER**

**CALL RRDDB ; READ FROM KEYBOARD**

**CPI PERIO ; IS CHARACTER A PERIOD ?**

**JS G18 ; YES - GO EXECUTE THE COMMAND**

**NO - ARG - CHARACTER IS STILL IN A**

**STA IBUFF ; REPLACE CHARACTER IN INPUT BUFFER**

**MV1 B.DOT ; ARG - DOT IN ADDRESS FIELD**

**CALL CLEAR ; CLEAR DISPLAY**

**MV1 B.ADFLD ; ARG - USE ADDRESS FIELD**

**CALL GTENK ; GET HEX DIGITS**

**CPI PERIO ; WAS LAST CHARACTER A PERIOD ?**

**JNC ERR ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND**

**MV1 G18 ; PUT HEX VALUE FROM GTENK TO H & L**

**SHLD PSAV ; HEX VALUE IS NEW USER PC**

**DISPC ; DISPLAY USER PROGRAM COUNTER**

**CPI PERIO ; WAS CHARACTER A PERIOD ?**

**JS CLOIS ; YES - CLEAR DISPLAY AND TERMINATE COMMAND**

**CPI COMMA ; WAS LAST CHARACTER ',' ?**

**JS STP28 ; YES - GO SET TIMER**

**NO - CHARACTER FROM KEYBOARD WAS NEITHER PERIOD NOR COMMA**

**STA IBUFF ; REPLACE THE CHARACTER IN THE INPUT BUFFER**

**MV1 B.DOT ; ARG - DOT IN ADDRESS FIELD**

**CALL CLEAR ; CLEAR DISPLAY**

**MV1 B.ADFLD ; ARG - USE ADDRESS FIELD OF DISPLAY**

**CALL GTENK ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED ?**

**FALSE ERR ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND**
ISIS-II 8088/8085 MACRO ASSEMBLER, X188  SDX85  PAGE 5

LOC OBJ SEQ SOURCE STATEMENT
0110 D21582 396+ JNC ERR ; HEX VALUE FROM GTHEX TO H & L
0110 DE 397 XCHG ; HEX VALUE IS NEW USER PC
0110 22F20 398 SHLD PSAV ; HEX VALUE IS NEW USER PC
0112 F218 399 CPI PERIO ; WAS LAST CHARACTER FROM GTHEX A PERIOD?
0123 C55E01 400 JI CDIS ; YES = CLEAR DISPLAY AND TERMINATE COMMAND
0123 401 NO - MUST HAVE BEEN A COMMA
0126 402 STP20:
0126 403 LDA ISAV ; GET USER INTERRUPT MASK
0129 404 AHI 388 ; KEEP INTERRUPT STATUS
0126 405 STA TEM ; SAVE USER INTERRUPT STATUS
0126 406 LLDH PSAV ; GET USER PC
0131 407 MOV A,M ; GET USER INSTRUCTION
0132 408 FFI3 ; CPI (DI) ; DI INSTRUCTION ?
0134 409 C30801 ; JMP STP21 ; NO
0137 410 XRA A ; YES = RESET USER INTERRUPT STATUS
0138 411 C43201 ; JMP STP22
0138 412 STP21:
0138 413 FEF8 ; CPI (EI) ; EI INSTRUCTION ?
0138 414 C24301 ; JMP STP23 ; NO
0145 415 3060 ; MVI A,88H ; YES = SET USER INTERRUPT STATUS
0145 416 3062 ; STA TEM ; SAVE NEW USER INTERRUPT STATUS
0145 417 3075 ; STP23:
0145 419 MVI A,(TIMER SHR 8) OR TMODE ; HIGH ORDER BITS OF TIMER VALUE
0145 420 ; OR ED WITH TMODE
0147 421 D35 ; OUT TEMI
0149 422 3E55 ; MVI A,TIMER AND $FFH ; LOW ORDER BITS OF TIMER VALUE
014B 423 D34 ; OUT TLMO
014D 424 3AFF ; LDA USCR ; GET USER IMAGE OF WHAT'S IN CSR
014F 425 3E55 ; ORI TPSRT ; SET TIMER COMMAND BITS TO START TIMER
0152 426 D38 ; OUT CSR ; START TIMER
0154 427 C31803 ; JMP ASTO ; RESTORE USER REGISTERS
0154 428 D20 ;
0154 429 STP25: ; BRANCH HERE WHEN TIMER INTERRUPTS AFTER
0154 430 D10 ; FIVE USER INSTRUCTION
0155 430 F5 ; PUSH PSW ; SAVE PSW
015B 431 3AFF ; LDA USCR ; GET USER IMAGE OF WHAT'S IN CSR
015B 432 3E55 ; MVI A,TIMER AND $FFH ; LOW ORDER BITS OF TIMER VALUE
015D 433 F64 ; OUT CSR ; START TIMER
015F 435 D32 ; OUT TLMO
0161 436 3E55 ; MVI A,TIMER AND $FFH ; LOW ORDER BITS OF TIMER VALUE
0162 437 24EF0 ; SHLD LSAX ; SAVE H & L
0163 438 3E55 ; MVI A,0H ; GET USER PROGRAM COUNTER FROM TOP OF STACK
0165 439 22F20 ; SHLD PSAV ; SAVE USER PC
0165 43B 440 F5 ; PUSH PSW
0165 441 441 ; POP H
0165 442 22ED00 ; SHLD PSAX ; SAVE FLIP/FLOPS AND A REGISTER
0166 443 240000 ; LDA B,0 ; CLEAR H & L
0171 444 3F ; GET USER STACK POINTER
0172 445 22F20 ; SHLD SBAX ; SAVE USER STACK POINTER
0175 446 21ED00 ; LDI H,BSAX+1 ; GET MONITOR STACK POINTER FOR
0176 447 F99 ; SPH / SAVING REMAINING USER REGISTERS
0179 448 C5 ; PUSH B ; SAVE B & C
0179 449 D4 ; PUSH D ; SAVE D & E
017B 450 28 ; OUT RMS ; GET USER INTERRUPT MASK
017C 451 8E07 ; ANI 07H ; KEEP MASK BITS
0178 452 21F20 ; LDI H,TEMP ; GET USER INTERRUPT STATUS
0181 453 B6 ; ORA M ; OR IT INTO MASK
0182 454 24EF0 ; STA ISAV ; SAVE INTERRUPT STATUS & MASK
0185 455 3060 ; MVI A,UNMSK ; UNMASK INTERRUPTS FOR MONITOR USE
0187 456 30 ; SIM
0187 456 C30000 ; JMP GSTEP ; GO GET READY FOR ANOTHER INSTRUCTION
0187 457 458 ;
0187 459 ;*****************************************************************************
0187 459 ;
0187 459 ; FUNCTION: SUBST - SUBSTITUTE MEMORY
0187 459 ; INPUTS: NONE
0187 459 ; OUTPUTS: NONE
0187 459 ; CALLS: CLEAR,GTHEX,UPDAD,UPDPT,ERR
0187 459 ; DESTRUCTS: A,B,C,D,E,A.L,F/F'S
0187 459 ;
0187 466 ; SUBST:
018B 466 0601 ; MVI B,DOT ; ARG - DOT IN ADDRESS FIELD
018B 469 0071 ; CALL CLEAR ; CLEAR THE DISPLAY
018B 469 0071 ; CALL CLEAR ; CLEAR THE DISPLAY
018B 468 04EF ; MVI B,ADFDL ; ARG = USE ADDRESS FIELD OF DISPLAY
0192 46B 22B002 ; CALL GTHEX ; GET HEX DIGITS - WERE ANY DIGITS RECEIVED?
019F 472 7650 ; CALL FALSE ERR ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND
0200 473 0078 ; JNC ERR
0199 474 0601 ; XCHG ; ASSIGN HEX VALUE RETURNED BY GTHEX TO
0199 475 22F60 ; SHLD CURAD ; / CURRENT ADDRESS
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
019E 476 2F50 ; SUBST:
LOC OAJ   SEQ     SOURCE STATEMENT

$01CE BE  496  CMP   M       ; WAS DATA STORED CORRECTLY?
$01C1 C25B02 497  JNZ   ERR       ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND

$01C4 2AF620 498  SUB18:  LHLD  CURAD       ; INCREMENT CURRENT ADDRESS

$01C7 25F620 500  INX   H       ;
$01CB F1 581  SHLD  CURAD       ;
$01CC C39C81 582  POP   PSW       ; RETRIEVE LAST CHARACTER

$01CF 890 583  JMP   SUB85       ;

$01D0 2159B2 584  SUB15:  CPI   PERIO     ; WAS LAST CHARACTER '.'?

$01D4 215901 585  JNZ   ERR       ; NO - DISPLAY ERROR MSG. AND TERMINATE COMMAND

$01D7 CD5D 586  JMP   CLDIS       ; YES - CLEAR DISPLAY AND TERMINATE COMMAND

$01D8 219A03 587  OUTPT  ; OUTPUTS: NONE

$01DB CD7B02 588  CALL  OUTPT       ; CALLS: OUTPT

$01DE 8680 589  MVIC  A,DFLDF       ; DECRYPTS A,B,C,D,E,H,L,F,F'S

$01D5 219A01 590  CALL  OUTPT       ; CALLS: CLEAR

$01E0 8680 591  MVIC  B,NODOT       ; ADDRESS OF BLANKS FOR DISPLAY

$01E5 CD7B02 592  CALL  OUTPT       ; ADDRESS FIELD.

$01E8 89 593  RET       ;

$01E9 8680 594  MVIC  B,NODOT       ; ARG - NO DOT IN ADDRESS FIELD

$01EE C36600 595  CALL  CLEAR       ; CLEAR THE DISPLAY

$01F1 8680 596  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$01F3 86819 597  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$01F6 C800 598  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$01F8 3230 599  OUT   CSR       ; INITIALIZE CSR

$01FA 32FF20 600  STA   USCSR       ; INITIALIZE USER CSR VALUE

$01FD C30000 601  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$0200 2AF20 602  MVIC  A,KNIT       ; GET CONTROL CHARACTER

$0203 2D0A03 603  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$0206 7C 604  JMP   CMND       ; GET ANOTHER COMMAND

$020B 8680 605  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$020E 86819 606  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$0211 8680 607  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$0214 3230 608  OUT   CSR       ; INITIALIZE CSR

$0216 32FF20 609  STA   USCSR       ; INITIALIZE USER CSR VALUE

$0219 C30000 610  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$021C 8F6 611  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$021E 8680 612  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$0221 86819 613  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$0224 8680 614  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$0227 3230 615  OUT   CSR       ; INITIALIZE CSR

$0229 32FF20 616  STA   USCSR       ; INITIALIZE USER CSR VALUE

$022C C30000 617  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$022F 8F6 618  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$0231 8680 619  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$0234 86819 620  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$0237 8680 621  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$023A 3230 622  OUT   CSR       ; INITIALIZE CSR

$023C 32FF20 623  STA   USCSR       ; INITIALIZE USER CSR VALUE

$023F C30000 624  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$0242 8F6 625  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$0245 8680 626  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$0248 86819 627  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$024B 8680 628  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$024E 3230 629  OUT   CSR       ; INITIALIZE CSR

$0251 32FF20 630  STA   USCSR       ; INITIALIZE USER CSR VALUE

$0254 C30000 631  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$0257 8F6 632  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$025A 8680 633  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$025D 86819 634  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$0260 8680 635  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$0263 3230 636  OUT   CSR       ; INITIALIZE CSR

$0266 32FF20 637  STA   USCSR       ; INITIALIZE USER CSR VALUE

$0269 C30000 638  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$026C 8F6 639  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$026F 8680 640  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$0272 86819 641  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$0275 8680 642  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$0278 3230 643  OUT   CSR       ; INITIALIZE CSR

$027B 32FF20 644  STA   USCSR       ; INITIALIZE USER CSR VALUE

$027E C30000 645  JMP   CDBRK       ; BACK TO MAIN PROCEDURE

$0281 8F6 646  CALL  CLDST       ; CLEARS THE DISPLAY AND BRANCH TO THE COMMAND RECOGNIZER.

$0284 8680 647  MVIC  A,KBNIT       ; GET CONTROL CHARACTER

$0287 86819 648  STA   CWHL       ; INITIALIZE KEYBOARD/DISPLAY BLANKING

$028A 8680 649  MVIC  A,CBNIT       ; INITIAL VALUE OF COMMAND STATUS REGISTER

$028D 3230 650  OUT   CSR       ; INITIALIZE CSR

$028F 32FF20 651  STA   USCSR       ; INITIALIZE USER CSR VALUE

$0292 C30000 652  JMP   CDBRK       ; BACK TO MAIN PROCEDURE
LOC OBJ SEQ SOURCE STATEMENT

596  ************************************************************
597  
598  ; FUNCTION: ERR - DISPLAY ERROR MESSAGE
599  
600  ; INPUTS: NONE
601  ; CALLS: OUTPT
602  ; DESTROYS: A,B,C,D,E,H,L,F,F'S
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 RECOGNISER.
607  
608  0215 AF 689  XRA A  ; ARG - USE ADDRESS FIELD
609  0216 8000 688  MVI B,MOOUT  ; ARG - NO DOT IN ADDRESS FIELD
610  0218 219E83 681  LXI H,ERMSG  ; ARG - ADDRESS OF ERROR MESSAGE
611  021B C07802 612  CALL OUTPT  ; OUTPUT ERROR MESSAGE TO ADDRESS FIELD
612  021E 36E1 613  MVI A,DTFIELD  ; ARG - USE DATA FIELD
613  0220 8000 614  MVI B,MOOUT  ; ARG - NO DOT IN DATA FIELD
614  0222 219A83 615  LXI H,BLANKS  ; ARG - ADDRESS OF BLANKS FOR DISPLAY
615  0225 C07802 616  CALL OUTPT  ; OUTPUT BLANKS TO DATA FIELD
616  0228 C3600 617  JMP COMMAD  ; GO GET A NEW COMMAND
617  
618  0219 AF 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  ; D - HEX DIGITS FROM KEYBOARD EVALUATED MODULO 2**16
626  ; CARRY - IF AT LEAST ONE VALID HEX DIGIT WAS READ
627  ; ELSE - RESET OTHERWISE
628  ; CALLS: RDKBD,INSDG,HDSKP,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  ; FLAG 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  0220 8000 647  MVI C,0  ; RESERVE HEX DIGIT FLAG
648  0220 C5 648  PUSH B  ; SAVE DISPLAY AND HEX DIGIT FLAGS
649  0222 110000 649  LXI D,0  ; SET HEX VALUE TO ZERO
650  0231 D5 650  PUSH D  ; SAVE HEX VALUE
651 GTHX:
652 0232 C07802 652  CALL RDKBD  ; READ KEYBOARD
653 0235 FE18 653  CPI 18H  ; IS CHARACTER A HEX DIGIT?
654 0237 D25002 654  JNC GTH28  ; NO - GO CHECK FOR TERMINATOR
655 023A D1 655  POP D  ; ARG - RETRIEVE HEX VALUE
656 023B C09802 657  CALL INSDG  ; INSERT HEX DIGIT IN HEX VALUE
658 023E C01 658  POP B  ; RETRIEVE DISPLAY FLAG
659 023F 8000 659  MVI C,1  ; SET HEX DIGIT FLAG
660 0241 C5 660  PUSH B  ; SAVE DISPLAY AND HEX DIGIT FLAGS
661 0242 D5 662  PUSH D  ; SAVE HEX VALUE
663 0245 78 663  MOV A,C  ; TEST DISPLAY FLAG
664 0246 88 664  RRC  ; SHOULD ADDRESS FIELD OF DISPLAY BE USED?
665 0245 D24902 666  JNC GTH18  ; YES - USE HEX VALUE AS IS
667 0248 53 668  MOV D,E  ; PUT LOW ORDER BYTE OF HEX VALUE IN D
669 GTH18:
669 0249 C06C02 669  CALL HDSKP  ; EXPAND HEX VALUE FOR DISPLAY
670 024A 78 670  MOV A,B  ; ARG - ADDRESS OF EXPANDED HEX VALUE IN H & L
671 024B 8601 671  CALL C07802  ; CALL OUTPT - OUTPUT HEX VALUE TO DISPLAY
672 024F C07802 672  CALL OUTPT  ; OUTPUT HEX VALUE TO DISPLAY
673 0252 C33202 673  JMP GTH85  ; GO GET NEXT CHARACTER
674 0255 D1 674  POP D  ; RETRIEVE HEX VALUE
675 0256 C1 675  POP B  ; RETRIEVE HEX DIGIT FLAG IN C
676 0257 FE11 676  CPI COMMA  ; WAS LAST CHARACTER ',' ?
677 0258 CA6702 677  CALLS J2  ; GTH25 - YES - READY TO RETURN
678 025E FE10 678  CPI PERIOD  ; NO - WAS LAST CHARACTER '.' ?
679 025A CA6702 679  CALLS J2  ; GTH25 - YES - READY TO RETURN
680 0261 110000 681  LXI D,0  ; SET HEX VALUE TO ZERO
682 0264 C3F700 682  JMP RETF  ; RETURN FALSE
683 GTH25:
683 0267 47 683  MOV B,A  ; SAVE LAST CHARACTER
684 0268 79 684  MOV A,C  ; SHIF Tet HEX DIGIT FLAG TO
685 0269 0F 685  RRC  ;/CARRY BIT
686 026A 78 686  MOV A,B  ; RESTORE LAST CHARACTER
687 026B 99 687  RET  ; RETURN
688 026C 93 689  
689 
690 
691 
692 
693 
694 
695 

*****************************************************************************
LOC OBJ  SEQ  SOURCE STATEMENT

696 ; FUNCTION: NKDSP - EXPAND HEX DIGITS FOR DISPLAY
697 ; INPUTS: DE - 4 HEX DIGITS
698 ; OUTPUTS: HL - ADDRESS OF OUTPUT BUFFER
699 ; CALLS: NOTHING
700 ; DESTROYS: A,H,L,F/F'S
701 ; DESCRIPTION: NKDSP EXPANDS EACH INPUT BYTE TO 2 BYTES IN A FORM
702 ; SUITABLE FOR DISPLAY BY THE OUTPUT ROUTINES. EACH INPUT
703 ; BYTE IS DIVIDED INTO 2 HEX DIGITS. EACH HEX DIGIT IS
704 ; PLACED IN THE LOW ORDER 4 BITS OF A BYTE WHOSE HIGH
705 ; ORDER 4 BITS ARE SET TO ZERO. THE RESULTING BYTE IS
706 ; STORED IN THE OUTPUT BUFFER. THE FUNCTION RETURNS THE
707 ; ADDRESS OF THE OUTPUT BUFFER.
708 ;
709 NKDSP:

\(026C\) 7A  
\(026D\) 8F  
\(026E\) 8F  
\(026F\) 8F  
\(0270\) 8F  
\(0271\) E6F  
\(0273\) 21F920  
\(0274\) 77  
\(0275\) 77  
\(0276\) 77  
\(0277\) 7A  
\(0278\) E6F  
\(0279\) 7A  
\(027A\) 23  
\(027B\) 77  
\(027C\) 7A  
\(027D\) 8F  
\(027E\) 8F  
\(027F\) 8F  
\(0280\) 8F  
\(0281\) E6F  
\(0282\) 7A  
\(0284\) 77  
\(0285\) 7A  
\(0286\) E6F  
\(0287\) 7A  
\(0288\) 23  
\(0289\) 77  
\(028A\) 21F920  
\(028D\) C9  

710 MOV A,D ; GET FIRST DATA BYTE
711 RC
712 RC
713 RC
714 RC
715 ANI 0FH
716 LXI H,OBUFF ; GET ADDRESS OF OUTPUT BUFFER
717 MOV M,A ; STORE CHARACTER IN OUTPUT BUFFER
718 MOV A,D ; GET FIRST DATA BYTE AND CONVERT 4 LOW ORDER
719 ANI 0FH ; /BITS TO A SINGLE CHARACTER
720 INX H ; NEXT BUFFER POSITION
721 MOV M,A ; STORE CHARACTER IN BUFFER
722 MOV A,E ; GET SECOND DATA BYTE
723 RC ; CONVERT 4 HIGH ORDER BITS
724 RC
725 RC
726 RC
727 ANI 0FH
728 INX H ; NEXT BUFFER POSITION
729 MOV M,A ; STORE CHARACTER IN BUFFER
730 MOV M,A ; STORE CHARACTER IN BUFFER
731 LXI H,OBUFF ; RETURN ADDRESS OF OUTPUT BUFFER IN H & L
732 RET

733 ;*******************************************************************************
734 ; FUNCTION: INIT - INPUT INTERRUPT PROCESSING
735 ; INPUTS: NONE
736 ; OUTPUTS: NONE
737 ; CALLS: NOTHING
738 ; DESTROYS: NOTHING
739 ; DESCRIPTION: INIT IS ENTERED BY MEANS OF AN INTERRUPT VECTOR (IV2C)
740 ; WHEN THE READ KEYBOARD ROUTINE IS WAITING FOR A
741 ; CHARACTER AND THE USER HAS PRESSED A KEY ON THE
742 ; KEYBOARD (EXCEPT "RESET" OR "VECTORED INTERRUPT")
743 ; STORES THE INPUT CHARACTER IN THE INPUT BUFFER AND
744 ; RETURNS CONTROL TO THE READ KEYBOARD ROUTINE.
745 ;*******************************************************************************
746 INIT:

\(028E\) E5  
\(028F\) F5  
\(0290\) 21E019  
\(0291\) 3640  
\(0292\) 25  
\(0296\) 7E  
\(0297\) E6F  
\(0299\) 32FE20  
\(029C\) F1  
\(029D\) E1  
\(029E\) C9  

752 PUSH H ; SAVE H & L
753 PUSH PSW ; SAVE F/F'S & REGISTER A
754 LXI H,CTRL ; ADDRESS FOR CONTROL CHARACTER OUTPUT
755 MVI M,READ ; OUTPUT CONTROL CHARACTER FOR READING
756 ; FROM KEYBOARD
757 MOV A,M ; READ A CHARACTER
758 ANI 3FH ; SEND 2 HIGH ORDER BITS
759 STA IBUFF ; STORE CHARACTER IN INPUT BUFFER
760 POP PSW ; RESTORE F/F'S & REGISTER A
761 POP H ; RESTORE H & L
762 RET

763 ;*******************************************************************************
764 ; FUNCTION: INSDG - INSERT HEX DIGIT
765 ; INPUTS: A - HEX DIGIT TO BE INSERTED
766 ; DE - HEX VALUE WITH DIGIT INSERTED
767 ; CALLS: NOTHING
768 ; DESTROYS: A,F/F'S
769 ; DESCRIPTION: INSDG SHIFTS THE CONTENTS OF D & E LEFT 4 BITS
770 ; (1 HEX DIGIT) AND INSERTS THE HEX DIGIT IN A IN THE LOW
771 ; ORDER DIGIT POSITION OF THE RESULT. A IS ASSUMED TO
772 ; CONTAIN A SINGLE HEX DIGIT IN THE LOW ORDER 4 BITS AND
773 ; ZEROS IN THE HIGH ORDER 4 BITS.
774 ;*******************************************************************************
775 INSDG:

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

780 XCNC ; 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 XCNC ; 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

LOC OBJ    SEQ SOURCE STATEMENT

795   ; - 0 OTHERWISE
796   ; CALLS: NOTHING
797   ; DESTROYS: A,F,P,F'S
798   ; DESCRIPTION:
799   ; IF THE REGISTER POINTER POINTS TO THE LAST REGISTER IN
800   ; THE EXAMINE REGISTER SEQUENCE, THE POINTER IS NOT
801   ; CHANGED AND THE FUNCTION RETURNS FALSE. IF THE REGISTER
802   ; POINTER DOES NOT POINT TO THE LAST REGISTER THEN THE
803   ; POINTER IS ADVANCED TO THE NEXT REGISTER IN THE SEQUENCE
804   ; AND THE FUNCTION RETURNS TRUE.
805   ;
806   ;
807   ;
808   ;
809   ;
810   ;
811   ;
812   ;
813   ;
814   ;
815   ; FUNCTION: OUTP - OUTPUT CHARACTERS TO DISPLAY
816   ; INPUTS: A - DISPLAY FLAG - 0 = USE DATA FIELD
817   ; B - DOT FLAG - 1 = OUTPUT DOT AT RIGHT EDGE OF FIELD
818   ; 0 = NO DOT
819   ; HL - ADDRESS OF CHARACTERS TO BE OUTPUT
820   ; CALLS: NOTHING
821   ; DESTROYS: A,B,C,D,E,H,L,F,P,F'S
822   ; DESCRIPTION:
823   ; OUTP 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   ; OUTP:
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   ; OUT2B:
859   ;
860   ;
861   ;
862   ;
863   ;
864   ;
865   ;
866   ;
867   ;
868   ;
869   ; FUNCTION: RDKB - READ KEYBOARD
870   ; INPUTS: NONE
871   ; CALLS: NOTHING
872   ; DESTROYS: A,H,L,F,P,F'S
873   ; DESCRIPTION:
874   ; RDKB DETERMINES WHETHER OR NOT THERE IS A CHARACTER IN
875   ; THE INPUT BUFFER. IF NOT, THE FUNCTION ENABLES
876   ; INTERRUPTS AND LOOPS UNTIL THE INPUT INTERRUPT
877   ; ROUTINE STORES A CHARACTER IN THE BUFFER. WHEN
878   ; THE BUFFER CONTAINS A CHARACTER, THE FUNCTION FLAGS
879   ; THE BUFFER AS EMPTY AND RETURNS THE CHARACTER
880   ; AS OUTPUT.
881   ;
882   ; RDKB:
883   ;
884   ;
885   ;
886   ;
887   ;
888   ;
889   ;
890   ;
891   ;
892   ;
893   ;
894   ;
895   ;
896   ;
897   ;
898   ;
899   ;
900   ;
901   ;
902   ;
903   ;
904   ;
905   ;
906   ;
907   ;
908   ;
909   ;
910   ;
911   ;
912   ;
913   ;
914   ;
915   ;
916   ;
917   ;
918   ;
919   ;
920   ;
921   ;
922   ;
923   ;
924   ;
925   ;
926   ;
927   ;
928   ;
929   ;
930   ;
931   ;
932   ;
933   ;
934   ;
935   ;
936   ;
937   ;
938   ;
939   ;
940   ;
941   ;
942   ;
943   ;
944   ;
945   ;
946   ;
947   ;
948   ;
949   ;
950   ;
951   ;
952   ;
953   ;
954   ;
955   ;
956   ;
957   ;
958   ;
959   ;
960   ;
961   ;
962   ;
963   ;
964   ;
965   ;
966   ;
967   ;
968   ;
969   ;
970   ;
971   ;
972   ;
973   ;
974   ;
975   ;
976   ;
977   ;
978   ;
979   ;
980   ;
981   ;
982   ;
983   ;
984   ;
985   ;
986   ;
987   ;
988   ;
989   ;
990   ;
991   ;
992   ;
993   ;
**Source Statement**

```assembly
894  895  896  897  898  899  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990  991  992  993

**LOC OBJ SEQ**

FUNCTION: RETF - RETURN FALSE
INPUTS: NONE
OUTPUTS: CARRY = 0 (FALSE)
CALLS: NOTHING
DESTRUCTS: CARRY
DESCRIPTION: RETF IS JUMPED TO BY FUNCTIONS WISHING TO RETURN FALSE.
RETRET SETS CARRY TO 0 AND RETURNS TO THE CALLER OF
THE ROUTINE INVOKING RETF.

#2F7 37
#2F8 38
#2F9 C9

FUNCTION: RETT - RETURN TRUE
INPUTS: NONE
OUTPUTS: CARRY = 1 (TRUE)
CALLS: NOTHING
DESTRUCTS: CARRY
DESCRIPTION: RETT JUMPED TO BY ROUTINES WISHING TO RETURN TRUE.
RETRET SETS CARRY TO 1 AND RETURNS TO THE CALLER OF
THE ROUTINE INVOKING RETT.

#2FA 37
#2FB C9

FUNCTION: HLOC - GET REGISTER SAVE LOCATION
INPUTS: NONE
OUTPUTS: HL - REGISTER SAVE LOCATION
CALLS: NOTHING
DESTRUCTS: B,C,H,L,F'/S
DESCRIPTION: HLOC RETURNS THE SAVE LOCATION OF THE REGISTER
INDICATED BY THE CURRENT REGISTER POINTER VALUE.

#2FC 2AFD28
#2FD AF
#305 66
#306 2628
#308 C9

FUNCTION: RGNAM - DISPLAY REGISTER NAME
INPUTS: NONE
OUTPUTS: NONE
CALLS: OUTPT
DESTRUCTS: A,B,C,D,E,H,L,F'/S
DESCRIPTION: RGNAM DISPLAYS, IN THE ADDRESS FIELD OF THE DISPLAY,
THE REGISTER NAME CORRESPONDING TO THE CURRENT
REGISTER POINTER VALUE.

#31A 2AFD28
#33C 2668
#33F 29
#310 81B803
#313 89

FUNCTION: RSTOR - RESTOR USER REGISTERS
INPUTS: NONE
OUTPUTS: NONE
CALLS: NOTHING
DESTRUCTS: A,B,C,D,E,H,L,F'/S
DESCRIPTION: RSTOR RESTORES ALL CPU REGISTERS, FLIP/FLOPS,
INTERRUPT STATUS, INTERRUPT MASK, STACK POINTER
AND PROGRAM COUNTER FROM THEIR RESPECTIVE
SAVE LOCATIONS IN MEMORY. BY RESTORING THE PROGRAM
COUNTER, THE ROUTINE EFFECTIVELY TRANSFERS CONTROL TO
THE ADDRESS IN THE PROGRAM COUNTER SAVE LOCATION.
THE TIMING OF THIS ROUTINE IS CRITICAL TO THE
CORRECT OPERATION OF THE SINGLE STEP ROUTINE.
IF ANY MODIFICATION CHANGES THE NUMBER OF CPU
STATES NEEDED TO EXECUTE THIS ROUTINE THEN THE
TIMER VALUE MUST BE ADJUSTED BY THE SAME NUMBER.

***** THIS IS ALSO THE ENTRY POINT FOR THE TTY MONITOR
TO RESTORE REGISTERS.

RSTOR:
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>335</td>
<td>0A</td>
<td>994</td>
<td>LDA ISAV : GET USER INTERRUPT MASK</td>
</tr>
<tr>
<td>335</td>
<td>04</td>
<td>995</td>
<td>ORI 18H : ENABLE SETTING OF INTERRUPT MASK AND</td>
</tr>
<tr>
<td>335</td>
<td>08</td>
<td>996</td>
<td>/RESET REG.5 FLIP FLOP</td>
</tr>
<tr>
<td>335</td>
<td>18</td>
<td>997</td>
<td>SIM : RESTORE USER INTERRUPT MASK</td>
</tr>
<tr>
<td>335</td>
<td>1C</td>
<td>998</td>
<td>LDA ISAV : GET USER INTERRUPT MASK</td>
</tr>
<tr>
<td>335</td>
<td>20</td>
<td>999</td>
<td>ANI 00H : SHOULD USER INTERRUPTS BE ENABLED ?</td>
</tr>
<tr>
<td>335</td>
<td>24</td>
<td>1000</td>
<td>JZ RSRR5 : NO - LEAVE INTERRUPTS DISABLED</td>
</tr>
<tr>
<td>335</td>
<td>28</td>
<td>1001</td>
<td>EI : YES - ENABLE INTERRUPTS FOR USER PROGRAM</td>
</tr>
<tr>
<td>335</td>
<td>2C</td>
<td>1002</td>
<td>JMP RSRR6 :</td>
</tr>
<tr>
<td>335</td>
<td>30</td>
<td>1003</td>
<td>STC : DUMMY INSTRUCTIONS - WHEN SINGLE STEP ROUTINE</td>
</tr>
<tr>
<td>335</td>
<td>34</td>
<td>1004</td>
<td>JNC RSRR6 : /IS BEING USED, THE TIMER IS RUNNING AND</td>
</tr>
<tr>
<td>335</td>
<td>38</td>
<td>1005</td>
<td>/EXECUTE TIME FOR THIS ROUTINE MUST NOT</td>
</tr>
<tr>
<td>335</td>
<td>3C</td>
<td>1006</td>
<td>/VARY.</td>
</tr>
<tr>
<td>335</td>
<td>40</td>
<td>1007</td>
<td>RSRR6:</td>
</tr>
<tr>
<td>335</td>
<td>44</td>
<td>1008</td>
<td>LXI @H, MASTE : SET MONITOR STACK POINTER TO START OF STACK</td>
</tr>
<tr>
<td>335</td>
<td>48</td>
<td>1009</td>
<td>SPHL : /WHICH IS ALSO END OF REGISTER SAVE AREA</td>
</tr>
<tr>
<td>335</td>
<td>4C</td>
<td>1010</td>
<td>POP D : RESTORE REGISTERS</td>
</tr>
<tr>
<td>335</td>
<td>50</td>
<td>1011</td>
<td>POP B :</td>
</tr>
<tr>
<td>335</td>
<td>54</td>
<td>1012</td>
<td>POP PSW :</td>
</tr>
<tr>
<td>335</td>
<td>58</td>
<td>1013</td>
<td>LLOD SAV : RESTORE USER STACK POINTER</td>
</tr>
<tr>
<td>335</td>
<td>5C</td>
<td>1014</td>
<td>SPHL :</td>
</tr>
<tr>
<td>335</td>
<td>60</td>
<td>1015</td>
<td>LLOD PSAV :</td>
</tr>
<tr>
<td>335</td>
<td>64</td>
<td>1016</td>
<td>PUSH H : PUT USER PROGRAM COUNTER ON STACK</td>
</tr>
<tr>
<td>335</td>
<td>68</td>
<td>1017</td>
<td>LLOD LSIV : RESTORE H &amp; L REGISTERS</td>
</tr>
<tr>
<td>335</td>
<td>6C</td>
<td>1018</td>
<td>RET : JUMP TO USER PROGRAM COUNTER</td>
</tr>
<tr>
<td>335</td>
<td>70</td>
<td>1019</td>
<td></td>
</tr>
<tr>
<td>335</td>
<td>74</td>
<td>1020</td>
<td></td>
</tr>
</tbody>
</table>
LOC OBJ SEQ SOURCE STATEMENT
0377 C9 1093 RET
1094  : -------------------------------------------------------------
1095  : MONITOR TABLES
1096  :
1097  :
1098  :
1099  :
1100  :
1101  ; COMMAND TABLE
1102  ; COMMAND CHARACTERS AS RECEIVED FROM KEYBOARD
1103  CMODT:
0376 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
0084 1108 NUMC EQU \$-CMODT ; NUMBER OF COMMANDS
1109  :
1110  :
1111  ; COMMAND ROUTINE ADDRESS TABLE
1112  ; (MUST BE IN REVERSE ORDER OF COMMAND TABLE)
1113  CMODA:
037C FOO0 1114 DW SSTEP ; ADDRESS OF SINGLE STEP ROUTINE
037E 9200 1115 DW EXAM ; ADDRESS OF EXAMINE REGISTERS ROUTINE
0380 8001 1116 DW SUBST ; ADDRESS OF SUBSTITUTE MEMORY COMMAND
0382 8000 1117 DW GOCMD ; ADDRESS OF GO ROUTINE
1118  :
1119  :
1120  :
1121  :
1122 DSPTB: ; TABLE FOR TRANSLATING CHARACTERS FOR OUTPUT
1123  :
1124  :
1125  DISPLAY
1126  :
1127  :
0000 1128 ZERO EQU \$-DSPTB
1129 0304 1129 DB 03H ; 0
1130 0305 60 1130 DB 60H ; 1
1131 0306 50 1131 DB 50H ; 2
1132 0307 40 1132 DB 40H ; 3
1133 0308 66 1133 DB 66H ; 4
1134 0005 1134 FIVE EQU \$-DSPTB
1135 0135 LETES EQU \$-DSPTB
1136 0309 D6 1136 DB 06H ; S AND S
1137 030A D7 1137 DB 07H ; T
1138 030B 70 1138 DB 70H ; 7
1139 0400 1139 EIGHT EQU \$-DSPTB
1140 03BC F7 1140 DB 07H ; 8
1141 03BD 76 1141 DB 76H ; 9
1142 008A 1142 LETRA EQU \$-DSPTB
1143 03BE 77 1143 DB 77H ; A
1144 008B 1144 LETRB EQU \$-DSPTB
1145 03BF C7 1145 DB 0CH ; B (LOWER CASE)
1146 008C 1146 LETRC EQU \$-DSPTB
1147 0398 93 1147 DB 93H ; C
1148 008D 1148 LETRD EQU \$-DSPTB
1149 0399 E5 1149 DB 0EH ; D (LOWER CASE)
1150 008E 1150 LETRE EQU \$-DSPTB
1151 039A 97 1151 DB 97H ; E
1152 008F 1152 LETRF EQU \$-DSPTB
1153 039B 17 1153 DB 17H ; F
0010 1154 LETRH EQU \$-DSPTB
1155 0394 67 1155 DB 67H ; G
0011 1156 LETRL EQU \$-DSPTB
1157 0395 63 1157 DB 63H ; H
0012 1158 LETRP EQU \$-DSPTB
1159 0396 37 1159 DB 37H ; I
0013 1160 LETRI EQU \$-DSPTB
1161 0397 68 1161 DB 68H ; J
0014 1162 LETRR EQU \$-DSPTB
1163 0398 05 1163 DB 05H ; K (LOWER CASE)
0015 1164 BLANK EQU \$-DSPTB
1165 0399 80 1165 DB 08H ; BLANK
1166  :
1167  ;************************************************************************
1168  :
1169  ; MESSAGES FOR OUTPUT TO DISPLAY
1170  :
039A 15 1171 BLANKS: D3 BLANK,BLANK,BLANK,BLANK ; FOR ADDRESS OR DATA FIELD
039B 15 1172 ERMESG: DB BLANK,LETTER,LETTER,LETTER ; ERROR MESSAGE FOR ADDR. FIELD
039C 15 1173 ERMESG: DB LETTER,BLANK,BLANK,BLANK ; EXECUTION MESSAGE
039D 15 1174 ;FOR ADDRESS FIELD
039E 15 1175 SGNAD: DB BLANK,BLANK,EIGHT,ZERO ; SIGN ON MESSAGE (ADDR. FIELD)
039F 8E 1176 SGNAD: DB EIGHT,FIVE ; SIGN ON MESSAGE (DATA FIELD)
LOC OBJ SEQ SOURCE STATEMENT

1170 ;
1176 RGPB: ; REGISTER POINTER TABLE
1181 ; THE ENTRIES IN THIS TABLE ARE IN THE SAME ORDER
1182 ; AS THE REGISTER DESIGNATOR KEYS ON THE KEYBOARD.
1183 ; EACH ENTRY CONTAINS THE REGISTER POINTER VALUE WHICH
1184 ; CORRESPONDS TO THE REGISTER DESIGNATOR. REGISTER
1185 ; POINTER VALUES ARE USED TO POINT INTO THE REGISTER
1186 ; NAME TABLE (NMTBL) AND REGISTER SAVE LOCATION
1187 ; TABLE (RGTBL).
1188 ;
1189 #3AC 86 DB 6 ; INTERRUPT MASK
1190 #3AD 89 DB 9 ; SPH
1191 #3AE 8A DB 10 ; SPL
1192 #3AF 8B DB 11 ; PCH
1193 #3B0 8C DB 12 ; PCL
1194 #3B1 87 DB 7 ; H
1195 #3B2 88 DB 8 ; L
1196 #3B3 89 DB 0 ; A
1197 #3B4 8A DB 1 ; B
1198 #3B5 8B DB 2 ; C
1199 #3B6 8C DB 3 ; D
1200 #3B7 8D DB 4 ; E
1201 #3B8 8E DB 5 ; FLAGS
1202 ;
1203 ;*************************************************************
1204 ;
1205 NMTBL: ; REGISTER NAME TABLE
1206 ; NAMES OF REGISTERS IN DISPLAY FORMAT
1207 DB BLANK,BLANK,BLANK,LETRA ; A REGISTER
1208 DB BLANK,BLANK,BLANK,LETRE ; B REGISTER
1209 DB BLANK,BLANK,BLANK,LETEC ; C REGISTER
1210 DB BLANK,BLANK,BLANK,LETIB ; D REGISTER
1211 DB BLANK,BLANK,BLANK,LETRE ; E REGISTER
1212 DB BLANK,BLANK,BLANK,LETRE ; F REGISTER
1213 DB BLANK,BLANK,BLANK,LETRI ; INTERRUPT MASK
1214 DB BLANK,BLANK,BLANK,LETRE ; H REGISTER
1215 DB BLANK,BLANK,BLANK,LETIB ; I REGISTER
1216 DB BLANK,LETER,LETRE,LETIB ; STACK POINTER HIGH ORDER BYTE
1217 DB BLANK,LETIB,LETIP,LETIB ; STACK POINTER LOW ORDER BYTE
1218 DB BLANK,LETIP,LETIC,LETIR ; PROGRAM COUNTER HIGH BYTE
1219 DB BLANK,LETIP,LETIR,LETIB ; PROGRAM COUNTER LOW BYTE
1220 ;
1221 ;*************************************************************
1222 ;
1223 RGTBL:
1224 DB ASAV AND 0FHF ; A REGISTER
1225 DB BSAV AND 0FHF ; B REGISTER
1226 DB CSAV AND 0FHF ; C REGISTER
1227 DB DS1V AND 0FHF ; D REGISTER
1228 DB ES1V AND 0FHF ; E REGISTER
1229 DB FS1V AND 0FHF ; F REGISTER
1230 DB ISAV AND 0FHF ; INTERRUPT MASK
1231 DB BSAV AND 0FHF ; R REGISTER
1232 DB LS1V AND 0FHF ; L REGISTER
1233 DB BS1V AND 0FHF ; STACK POINTER HIGH ORDER BYTE
1234 DB SPSLV AND 0FHF ; STACK POINTER LOW ORDER BYTE
1235 DB PCSV AND 0FHF ; PROGRAM COUNTER HIGH ORDER BYTE
### 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>83F9</td>
<td>F2</td>
<td>1240</td>
<td>DB PCELF AND $0FFH ; PROGRAM COUNTER LOW ORDER BYTE</td>
</tr>
<tr>
<td>83F9</td>
<td>F2</td>
<td>1241</td>
<td>NUMRG EQU ($ - RGTBL) ; NUMBER OF ENTRIES IN</td>
</tr>
<tr>
<td>1242</td>
<td></td>
<td></td>
<td>/REGISTER SAVE LOCATION TABLE</td>
</tr>
<tr>
<td>1243</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1244</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1245</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1246</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1247</td>
<td></td>
<td></td>
<td>; SDK-85 TTY MONITOR</td>
</tr>
<tr>
<td>1248</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1249</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1250</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1251</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1252</td>
<td></td>
<td></td>
<td>; ABSTRACT</td>
</tr>
<tr>
<td>1253</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1254</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1255</td>
<td></td>
<td></td>
<td>; THIS PROGRAM WAS ADAPT Pes FROM THE SDK-88 MONITOR.</td>
</tr>
<tr>
<td>1256</td>
<td></td>
<td></td>
<td>; THIS PROGRAM RUNS ON THE 8855 BOARD AND IS DESIGNED TO PROVIDE</td>
</tr>
<tr>
<td>1257</td>
<td></td>
<td></td>
<td>; THE USER WITH A MINIMAL MONITOR. BY USING THIS PROGRAM,</td>
</tr>
<tr>
<td>1258</td>
<td></td>
<td></td>
<td>; THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD</td>
</tr>
<tr>
<td>1259</td>
<td></td>
<td></td>
<td>; A PROGRAM (IN ABSOLUTIY) INTO RAM, AND EXECUTE INSTRUCTIONS</td>
</tr>
<tr>
<td>1260</td>
<td></td>
<td></td>
<td>; BEFORE IN MEMORY. THE MONITOR ALSO PROVIDES THE USER WITH</td>
</tr>
<tr>
<td>1261</td>
<td></td>
<td></td>
<td>; ROUTINES FOR PERFORMING CONE1/0.</td>
</tr>
<tr>
<td>1262</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1263</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1264</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1265</td>
<td></td>
<td></td>
<td>; PROGRAM ORGANIZATION</td>
</tr>
<tr>
<td>1266</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1267</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1268</td>
<td></td>
<td></td>
<td>; THE LISTING IS ORGANIZED IN THE FOLLOWING WAY. FIRST THE COMMAND</td>
</tr>
<tr>
<td>1269</td>
<td></td>
<td></td>
<td>; RECOGNIZE, WHICH IS THE HIGHEST LEVEL ROUTINE IN THE PROGRAM.</td>
</tr>
<tr>
<td>1270</td>
<td></td>
<td></td>
<td>; NEXT THE ROUTINES TO IMPLEMENT THE VARIOUS COMMANDS.</td>
</tr>
<tr>
<td>1271</td>
<td></td>
<td></td>
<td>; THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK. WITHIN</td>
</tr>
<tr>
<td>1272</td>
<td></td>
<td></td>
<td>; EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL ORDER, BY ENTRY POINT</td>
</tr>
<tr>
<td>1273</td>
<td></td>
<td></td>
<td>; OF THE ROUTINE.</td>
</tr>
<tr>
<td>1274</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1275</td>
<td></td>
<td></td>
<td>; MACROS USED IN THE TTY MONITOR ARE DEFINED IN THE KEYBOARD MONITOR.</td>
</tr>
<tr>
<td>1276</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1277</td>
<td></td>
<td></td>
<td>; LIST OF FUNCTIONS</td>
</tr>
<tr>
<td>1278</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1279</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1280</td>
<td></td>
<td></td>
<td>; GETCM</td>
</tr>
<tr>
<td>1281</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1282</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1283</td>
<td></td>
<td></td>
<td>; DCMDO</td>
</tr>
<tr>
<td>1284</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1285</td>
<td></td>
<td></td>
<td>; ICMDO</td>
</tr>
<tr>
<td>1286</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1287</td>
<td></td>
<td></td>
<td>; SCMD</td>
</tr>
<tr>
<td>1288</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1289</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1290</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1291</td>
<td></td>
<td></td>
<td>; CI</td>
</tr>
<tr>
<td>1292</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1293</td>
<td></td>
<td></td>
<td>; CVNBN</td>
</tr>
<tr>
<td>1294</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1295</td>
<td></td>
<td></td>
<td>; CRUT</td>
</tr>
<tr>
<td>1296</td>
<td></td>
<td></td>
<td>; DELAY</td>
</tr>
<tr>
<td>1297</td>
<td></td>
<td></td>
<td>; ECHO</td>
</tr>
<tr>
<td>1298</td>
<td></td>
<td></td>
<td>; ERROR</td>
</tr>
<tr>
<td>1299</td>
<td></td>
<td></td>
<td>; FRET</td>
</tr>
<tr>
<td>1300</td>
<td></td>
<td></td>
<td>; GETCH</td>
</tr>
<tr>
<td>1301</td>
<td></td>
<td></td>
<td>; GETHX</td>
</tr>
<tr>
<td>1302</td>
<td></td>
<td></td>
<td>; GETNM</td>
</tr>
<tr>
<td>1303</td>
<td></td>
<td></td>
<td>; HILO</td>
</tr>
<tr>
<td>1304</td>
<td></td>
<td></td>
<td>; NMOJ</td>
</tr>
<tr>
<td>1305</td>
<td></td>
<td></td>
<td>; PVAL</td>
</tr>
<tr>
<td>1306</td>
<td></td>
<td></td>
<td>; REGDS</td>
</tr>
<tr>
<td>1307</td>
<td></td>
<td></td>
<td>; RGADR</td>
</tr>
<tr>
<td>1308</td>
<td></td>
<td></td>
<td>; SRET</td>
</tr>
<tr>
<td>1309</td>
<td></td>
<td></td>
<td>; STIF0</td>
</tr>
<tr>
<td>1310</td>
<td></td>
<td></td>
<td>; STILP</td>
</tr>
<tr>
<td>1311</td>
<td></td>
<td></td>
<td>; VALDG</td>
</tr>
<tr>
<td>1312</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1313</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1314</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1315</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1316</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1317</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1318</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1319</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1320</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1321</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1322</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1323</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>1324</td>
<td></td>
<td></td>
<td>; BRCHR EQU 1BH ; CODE FOR BREAK CHARACTER (ESCAPE)</td>
</tr>
<tr>
<td>1325</td>
<td></td>
<td></td>
<td>; BRTP5 EQU $0F8H ; LOCATION OF START OF BRANCH TABLE IN ROM</td>
</tr>
<tr>
<td>1326</td>
<td></td>
<td></td>
<td>; CR EQU $00H ; CODE FOR CARRIAGE RETURN</td>
</tr>
<tr>
<td>1327</td>
<td></td>
<td></td>
<td>; ESC EQU 1BH ; CODE FOR ESCAPE CHARACTER</td>
</tr>
<tr>
<td>1328</td>
<td></td>
<td></td>
<td>; HCHAR EQU $0FH ; MASK TO SELECT LOWER HEX CHAR FROM BYTE</td>
</tr>
<tr>
<td>1329</td>
<td></td>
<td></td>
<td>; INVR EQU $0FFH ; MASK TO INVERT BYTE FLAG</td>
</tr>
<tr>
<td>1330</td>
<td></td>
<td></td>
<td>; LF EQU $0AH ; CODE FOR LINE FEED</td>
</tr>
<tr>
<td>1331</td>
<td></td>
<td></td>
<td>; LOWER EQU 0 ; DENOTES LOWER HALF OF BYTE IN ICM</td>
</tr>
<tr>
<td>1332</td>
<td></td>
<td></td>
<td>; LSGNON EQU --- ; LENGTH OF SIGNON MESSAGE - DEFINED LATER</td>
</tr>
<tr>
<td>1333</td>
<td></td>
<td></td>
<td>; MARST EQU --- ; START OF MONITOR STACK - DEFINED IN</td>
</tr>
<tr>
<td>1334</td>
<td></td>
<td></td>
<td>; /KEYBOARD MONITOR</td>
</tr>
<tr>
<td>1335</td>
<td></td>
<td></td>
<td>; NCOMP EQU --- ; NUMBER OF VALID COMMANDS - DEFINED LATER</td>
</tr>
<tr>
<td>1336</td>
<td></td>
<td></td>
<td>; NEWCH EQU $0FH ; MASK FOR CHECKING MEMORY ADDR DISPLAY</td>
</tr>
<tr>
<td>1337</td>
<td></td>
<td></td>
<td>; PRTVAL EQU $0F8H ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR</td>
</tr>
<tr>
<td>1338</td>
<td></td>
<td></td>
<td>; RAMST EQU --- ; START ADDRESS OF RAM - DEFINED IN</td>
</tr>
<tr>
<td>1339</td>
<td></td>
<td></td>
<td>; /KEYBOARD MONITOR</td>
</tr>
</tbody>
</table>

### MONITOR EQUIATES

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>001B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007F</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOC OBJ SEQ SOURCE STATEMENT

1340  RTABS EQU --- SIZE OF ENTRY IN RTAB TABLE
1341  SSTRT EQU 40H SHIFTED START BIT
1342  STOPB EQU 48H STOP BIT
1343  STRT EQU 0CH UNSHIFTED START BIT
1344  TERM EQU 0BH CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
1345  UPER EQU 0FFH DENOTES UPPER HALF OF BYTE IN ICMD

1346  
1347  ; DELAY VALUES IF NO WAIT STATE

1348  ;
1349  048C IF 1-WAITS
1350  048C IBTIM EQU 1164 INTER-BIT TIME DELAY
1351  048C OBTIM EQU 1164 OUTPUT INTER-BIT TIME DELAY
1352  0246 TIM4 EQU 4656 4 BIT TIME DELAY
1353  0246 WAIT EQU 582 DELAY UNTIL READY TO SAMPLE BITS
1354  ENDIF

1355  ;
1356  03FA B3C87 ; DELAY VALUES IF ONE WAIT STATE

1357  ;
1358  03FF 4E IF WAITS
1359  03FF IBTIM EQU 938 INTER-BIT DELAY
1360  03FF OBTIM EQU 938 OUTPUT INTER-BIT TIME DELAY
1361  03FF TIM4 EQU 3720 4 BIT TIME DELAY
1362  03FF WAIT EQU 465 DELAY UNTIL READY TO SAMPLE BITS

1363  ;

1364  ;
1365  ;*********************************************************************************************
1366  ; RESTART ENTRY POINT
1367  ;
1368  ;

1369  ;*********************************************************************************************
1370  ; PRINT SIGNON MESSAGE

1371  ;
1372  ;

1373  ;
1374  ;
1375  ;
1376  ;

1377  ;
1378  ;
1379  ;

1380  ;
1381  ;
1382  ;*********************************************************************************************
1383  ; COMMAND RECOGNIZING ROUTINE

1384  ;
1385  ;
1386  ;

1387  ;

1388  ;

1389  ;

1390  ;

1391  ;

1392  ;

1393  ;

1394  ;

1395  ;

1396  ;

1397  ;

1398  ;

1399  ;

1400  ;

1401  ;

1402  ;*********************************************************************************************
1403  ; FUNCTION: GETCM
1404  ; INPUTS: NONE
1405  ; OUTPUTS: NONE
1406  ; CALLS: GETCH,ECHO,ERROR
1407  ; DESTROYS: A,B,C,H,L,F,P,S
1408  ; DESCRIPTION: GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
1409  ; AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
1410  ; TABLE. IF SUCCESSFUL, THE ROUTINE
1411  ; CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
1412  ; A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
1413  ; IS TRANSFERRED TO THIS ROUTINE. IF THE CHARACTER
1414  ; DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
1415  ; THE ERROR HANDLER.

1416  ;

1417  ;

1418  GETCM:

1419  LXI H, MSKTB 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 GTCB3 WANT TO LEAVE ROOM FOR RST BRANCH

1424  GTCB3:

1425  CALL GETCH ; GET COMMAND CHARACTER TO A

1426  CALL ECHO ; ECHO COMMAND TO USER

1427  MOV A,C ; PUT COMMAND CHARACTER INTO ACCUMULATOR

1428  LXI B, MVCMD ; C CONTAINS LOOP AND INDEX COUNT

1429  LXI H, CTAB ; HL POINTS INTO COMMAND TABLE

1430  GTCB5:

1431  LXI H, CDR COMPARE TABLE ENTRY AND CHARACTER

1432  JZ GTCB10 BRANCH IF CDR = GTCB10 = COMMAND RECOGNIZED

1433  INX H INCREASE TREE COMMAND POINTER

1434  DCR C DECREMENT LOOP COUNT

1435  JNZ GTCB5 BRANCH IF NOT AT TABLE END

1436  JMP ERROR ELSE, COMMAND CHARACTER IS ILLEGAL

1437  GTCB10:

1438  LXI H, CDR IF GOOD COMMAND, LOAD ADDRESS OF TABLE

1439  ;
LOC OBJ SEQ SOURCE STATEMENT

**COMMAND IMPLEMENTING Routines**

#437 88B2 MOV C,2 ; GET 2 NUMBERS FROM INPUT STREAM
#439 C5B9 86 CALL GETNM
#43C D1 POP D ; ENDING ADDRESS TO DE
#43D E1 POP H ; STARTING ADDRESS TO HL
#443 C5B8 85 :
#444 7C CALL CROUT ; ECHO CARRIAGE RETURN/LINE FEED
#444 CDC76 MOV A,H ; DISPLAY ADDRESS OF FIRST LOCATION IN LINE
#445 7D CALL NNOUT
#446 CDC76 MOV A,L ; ADDRESS IS 2 BYTES LONG
#447 DCMBD: CALL NNOUT
#449 E820 MOV C,0
#44B CD98 85 CALL ECHO ; USE BLANK AS SEPARATOR
#44D 7E MOV A,M ; GET CONTENTS OF NEXT MEMORY LOCATION
#450 CDC70 MOV NNOUT ; DISPLAY CONTENTS
#453 0A86 CALL HIL0 ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
#454 Calculates ; GREATEST THAN OR EQUAL TO ENDING ADDRESS
#455 D258 84 CALL CROUT ; CARRIAGE RETURN/LINE FEED TO END LINE
#45B C88B 84 CALL CROUT
#45E 23 JMP GETCH ; ALL DONE
#45F 7D MOV A,L ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
#46B 58BF CALL NEWLN ; IF LAST HEX DIGIT OF ADDRESS DENOTES
#46D C498 84 JNC DCMB1 ; START OF NEW LINE
#46F 028B JMP DCMB5 ; YES - START NEW LINE WITH ADDRESS

**FUNCTION: DCMD**

#468 C026 86 CALL GETHX ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
#46B D7D0 84 FALSE GMC85 ; BRANCH IF NO NUMBER PRESENT
#46E 7A MOV A,D ; ELSE, GET TERMINATOR
#46F FE01 CPI CR ; SEE IF CARRIAGE RETURN
#471 C118 C6 JNZ ERROR ; ERROR IF NOT PROPERLY TERMINATED
#474 21F220 LXI H,PSAV ; WANT NUMBER TO REPLACE SAVE FOM COUNTER
#477 71 MOV M,C
#478 23 INC
#479 88 MOV M,B
#47A C380 84 JMP GCM18
#47B 7A MOV A,D ; IF NO STARTING ADDRESS, MAKE SURE THAT
#47E FE06 CPI CR ; /CARRIAGE RETURN TERMINATED COMMAND
#480 C71 8C JNZ ERROR ; ERROR IF NOT
#483 C180 83 JMP RSTOR ; RESTORE REGISTES AND BEGIN EXECUTION
#483 C180 83 (RSTOR IS IN KEYBOARD MONITOR)
#485 25 JMP ;
#486 8881 MOV C,1

**FUNCTION: ICMD**

#487 C026 86 CALL GETHX ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
#48B D7D0 84 FALSE GMC85 ; BRANCH IF NO NUMBER PRESENT
#48E 7A MOV A,D ; ELSE, GET TERMINATOR
#48F FE01 CPI CR ; SEE IF CARRIAGE RETURN
LOC OBJ SOURCES STATEMENT
0488 CDS86 1539 CAL'. GETMN ; GET SINGLE NUMBER FROM INPUT STREAM
0490 3FP 1540 MVI A,UPPER
0490 D2PD8 1541 STA TEMP ; TEMP WILL HOLD THE UPPER/LOWER BYTE FLAG
0490 D1 1542 POP D ; ADDRESS OF START TO DE
0490 D1 1543 ICM85:
0491 CDPF6 1544 CALL GETCH ; GET A CHARACTER FROM INPUT STREAM
0494 4F 1545 MOV C,A
0495 CDP85 1546 CALL ECRO ; ECHO IT
0498 79 1547 MOV A,C ; PUT CHARACTER BACK INTO A
0499 FF18 1548 CVI TERM ; SEE IF CHARACTER IS A TERMINATING CHARACTER
049B CAC74 1549 JS IC25 ; IF SO, ALL DONE ENTERING CHARACTERS
049E CD797 1550 CALL VALDL ; ELSE, SEE IF VALID DELIMITER
04A1 D5B14 1552+ TRUE IC85 ; IF SO SIMPLY IGNORE THIS CHARACTER
04A4 C5857 1553 CALL VALD5 ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
04A7 D2C14 1554 FALSE IC28 ; IF NOT, BRANCH TO HANDLE ERROR CONDITION
04AD B089 1555+ CALL CHVB8 ; CONVERT DIGIT TO BINARY
045C 4F 1556 MOV C,A
045F CDP85 1557 CALL ECRO ; MOVE RESULT TO C
0463 CDP85 1558 CALL STRLP ; STORE IN APPROPRIATE HALF WORD
0461 3D28 1559 LDA TEMP ; GET HALF BYTE FLAG
0465 85 155D OA A ; SET P/PS
046B C8894 1561 JSI ICM18 ; BRANCH IF FLAG SET FOR UPPER
0468 13 1562 INX D ; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
0469 E8F 1563 ICM18:
046B 32PD8 1564 XRI INVR ; TOGGLE STATE OF FLAG
046B STA TEMP ; PUT NEW VALUE OF FLAG BACK
046E C9184 1565 JMP ICM85 ; PROCESS NEXT DIGIT
046F C3487 1567 ICM82:
0473 C31186 1568 CALL STMPS ; ILLEGAL CHARACTER
0474 C31186 1569 JMP ERROR ; MAKE SURE ENTIRE BYTE FILLED THEN ERROR
0475 ICM85:
0477 C3487 1571 CALL STMPS ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
0478 CD685 1572 CALL CHKOUT ; ADD CARRIAGE RETURN
047C C8894 1573 JMP GETCM
0480 E03 1574 ;
0482 CDS86 1575 ;
0484 E03 1576 ;**************************************************************************************
0484 E03 1577 ;
0484 E03 1578 ;
0484 E03 1579 ; FUNCTION: MCMD
0484 E03 1580 ; INPUTS: NONE
0484 E03 1581 ; OUTPUTS: NONE
0484 E03 1582 ; CALLS: GETCM,HILO,GETMN
0484 E03 1583 ; DESTROYS: A,B,C,D,E,H,L,F/P/S
0484 E03 1584 ; DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.
0484 E03 1585 ;
0484 E03 1586 ; MCMD:
0484 E03 1587 ;
0484 E03 1588 ;
0484 E03 1589 ;
0484 E03 1590 ;
0484 E03 1591 ;
0484 E03 1592 ;
0484 E03 1593 ;
0484 E03 1594 ;
0484 E03 1595 ;
0484 E03 1596 ;
0484 E03 1597 ;
0484 E03 1598 ;
0484 E03 1599 ;
0484 E03 1600 ;
0484 E03 1601 ;
0484 E03 1602 ;
0484 E03 1603 ;
0484 E03 1604 ;
0484 E03 1605 ;
0484 E03 1606 ;
0484 E03 1607 ;
0484 E03 1608+ ;
0484 E03 1609 ;
0484 E03 1610+ ;
0484 E03 1611+ ;**************************************************************************************
0484 E03 1612+ ;
0484 E03 1613+ ;
0484 E03 1614 ;
0484 E03 1615 ; FUNCTION: SCMD
0484 E03 1616 ; INPUTS: NONE
0484 E03 1617 ; OUTPUTS: NONE
0484 E03 1618 ; CALLS: GETMN,GETCM,NNOUT,ECHO
0484 E03 1619 ; DESTROYS: A,B,C,D,E,H,L,F/P/S
0484 E03 1620 ; DESCRIPTION: SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.
0484 E03 1621 ;
0484 E03 1622 ; SCMD:
0484 E03 1623 ;
0484 E03 1624 ;
0484 E03 1625 ;
0484 E03 1626 ;
0484 E03 1627 ;
0484 E03 1628 ;
0484 E03 1629 ;
0484 E03 1630 ;
0484 E03 1631 ;
0484 E03 1632 ;
0484 E03 1633+ ;
0484 E03 1634 ;
0484 E03 1635 ;
0484 E03 1636 ;
0484 E03 1637 ;
LOC   OBJ   SEQ  SOURCE STATEMENT

1638  FALSE   XCM15  ; IF NO VALUE PRESENT, BRANCH
1639  JNC   XCM15
1640  MOV   M,C  ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
1641  INX   H  ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
1642  JMP   XCM85
1643  ;
1644  ;
1645  ;
1646  ;
1647  ;
1648  ;
1649  ; FUNCTION: XCMD
1650  ; INPUTS: NONE
1651  ; OUTPUTS: NONE
1652  ; CALLS: GETCH, ECHO, REGDS, GETCM, ERROR, RGA3R, NMOUT, CROUT, GETEX
1653  ; DESTROYS: A, B, C, D, E, H, L, F, F'S
1654  ; DESCRIPTION: XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X) COMMAND.
1655  ;
1656  ; XCMD:
1657  CALL   GETCM  ; GET REGISTER IDENTIFIER
1658  MOV   C,A
1659  CALL   ECHO   ; ECHO IT
1660  MOV   A,C
1661  CPI   CR
1662  JNZ   XCM85  ; BRANCH IF NOT CARRIAGE RETURN
1663  CALL   REGDS  ; ELSE, DISPLAY REGISTER CONTENTS
1664  JMP   XCM85  ; THEN TERMINATE COMMAND
1665  ; XCM85:
1666  CALL   ECHO   ; ECHO SPACE TO USER
1667  STA   TEMP  ; PUT SPACE INTO TEMP AS DELIMITER
1668  LDA   TEMP   ; GET TERMINATOR
1669  CPI   ' '  ; SEE IF A BLANK
1670  JSR   XCM15  ; YES - GO CHECK POINTER INTO TABLE
1671  CPI   ' '  ; NO - SEE IF COMMA
1672  JSR   XCM15  ; NO - MUST CARRIAGE RETURN TO END COMMAND
1673  MOV   A,M
1674  ORA   A  ; SET F/F'S
1675  JSR   XCM18  ; BRANCH IF NOT AT END OF TABLE
1676  CALL   CROUT  ; ELSE, OUTPUT CARRIAGE RETURN LINE FEED
1677  JMP   XCM84  ; AND EXIT
1678  ; XCM18:
1679  PUSH   H  ; PUT POINTER ON STACK
1680  MOV   E,M
1681  MOV   B,M  ; ADDRESS OF SAVE LOCATION FROM TABLE
1682  MOV   D,RAMST SHR B
1683  INX   H
1684  MOV   B,M  ; FETCH LENGTH FLAG FROM TABLE
1685  PUSH   D  ; SAVE ADDRESS OF SAVE LOCATION
1686  POP   H  ; MOVE ADDRESS TO HL
1687  PUSH   B  ; SAVE LENGTH FLAG
1688  MOV   A,M  ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
1689  CALL   NMOUT  ; DISPLAY IT
1690  POP   PSW  ; GET BACK LENGTH FLAG
1691  POP   PSW  ; SAVE IT AGAIN
1692  ORA   A  ; SET F/F'S
1693  JSR   XCM20  ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
1694  DCX   H  ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
1695  CALL   NMOUT  ; DISPLAY THEM
1696  ; XCM20:
1697  MOV   C, ' '  ; USE DASH AS SEPARATOR
1698  CALL   GETEX  ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
1699  JNC   XCM38  ; NO - GO CHECK FOR NEXT REGISTER
1700  ; XCM38:
1701  JNC   XCM38
1702  MOV   A,D
1703  STA   TEMP  ; ELSE, SAVE THE TERMINATOR FOR NOW
1704  POP   PSW  ; GET BACK LENGTH FLAG
1705  POP   H  ; PUT ADDRESS OF SAVE LOCATION INTO HL
1706  ORA   A  ; SET F/F'S
1707  JSR   XCM25  ; IF 8 BIT REGISTER, BRANCH
1708  MOV   M,B  ; SAVE UPPER 8 BITS
1709  DCX   H  ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
1710  JSR   XCM25  ; DO NEXT REGISTER
1711  MOV   M,C  ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
1712  CALL   GETEX  ; SIZE OF ENTRY IN RTAB TABLE
1713  JSR   XCM30  ; POINTER INTO REGISTER TABLE RTAB
1714  ADD   ENTRY SIZE TO POINTER
1715  JMP   XCM10  ; DO NEXT REGISTER
1716  ; XCM27:
1717  ; XCM30:
1718  MOV   A,D  ; GET TERMINATOR
1719  STA   TEMP  ; SAVE IN MEMORY
1720  POP   D  ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
1721  POP   D  ; OF SAVE LOCATION
1722  JMP   XCM27  ; GO INCREMENT REGISTER TABLE POINTER
1723  ;
LOC OBJ SEQ SOURCE STATEMENT

1735 ;***************************************
1736 ; UTILITY Routines
1737 ;
1738 ;
1739 ;
1740 ;
1741 ;
1742 ;
1743 ;
1744 ; FUNCTION: CI
1745 ; INPUTS: NONE
1746 ; OUTPUTS: A = CHARACTER FROM TTY
1747 ; CALLS: DELAY
1748 ; DESCRIBES: A,F/F's
1749 ; DESCRIPTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
1750 ; TTY AND THEN RETURNS THE CHARACTER, VIA THE A
1751 ; REGISTER, TO THE CALLING ROUTINE. THIS ROUTINE
1752 ; IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.
1753 ;
1754 CI: 1755
1756 PUSH D ; SAVE DE
1757 C105:
1758 RIM ; GET INPUT BIT
1759 JC CI05 ; INTO CARRY WITH IT
1760 LXI D,WAIT ; WAIT FOR MIDDLE OF BIT
1761 CALL DELAY
1762 LXI B,0800 ; SAVE BC
1763 LXI B,0 ; B<--0, C<--1 BITS TO RECEIVE
1764 C110:
1765 LXI D,IBTIM
1766 CALL DELAY ; WAIT UNTIL MIDDLE OF NEXT BIT
1767 CALL DELAY
1768 RIM ; GET THE BIT
1769 RAL ; INTO CARRY
1770 MOV A,B ; GET PARTIAL RESULT
1771 MOV C,A ; SHIFT IN NEXT DATA BIT
1772 MOV B,A ; REPLACE RESULT
1773 DCR C ; DEC COUNT OF BITS TO GO
1774 JNZ C110 ; BRANCH IF MORE LEFT
1775 LXI D,IBTIM ; ELSE, WANT TO WAIT OUT STOP BIT
1776 CALL DELAY
1777 MOV A,B ; GET RESULT
1778 POP B ; RESTORE SAVED REGISTERS
1779 POP A
1780 EI
1781 RET ; THAT'S IT
1782 ;
1783 ;
1784 ;*************************************************************
1785 ;
1786 ; FUNCTION: CNVBN
1787 ; INPUTS: C = ASCII CHARACTER '0'-'9' OR 'A'- 'F'
1788 ; OUTPUTS: A = # TO F HEX
1789 ; CALLS: NOTHING
1790 ; DESTROYS: A,F/F's
1791 ; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
1792 ; CNVBN INTO ITS CORRESPONDING BINARY VALUE. CNVBN
1793 ; DOES NOT CHECK THE VALIDITY OF ITS INPUT.
1794 ;
1795 ; CNVBN:
1796 ;
1797 MOV A,C
1798 SUI '9' ; SUBTRACT CODE FOR '9' FROM ARGUMENT
1799 CPI 10 ; WANT TO TEST FOR RESULT OF 0 TO 9
1800 BM ; IF SO, THEN ALL DONE
1801 SUI 7 ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
1802 RET ; SO RETURN AFTER SUBTRACTING BIAS OF 7
1803 ;
1804 ;
1805 ;*************************************************************
1806 ;
1807 ; FUNCTION: CO
1808 ; INPUTS: C = CHARACTER TO OUTPUT TO TTY
1809 ; OUTPUTS: C = CHARACTER OUTPUT TO TTY
1810 ; CALLS: DELAY
1811 ; DESTROYS: A,F/F's
1812 ; DESCRIPTION: CO SENDS ITS INPUT ARGUMENT TO THE TTY.
1813 ;
1814 CO:
1815
1816 PUSH B ; SAVE BC
1817 PUSH D ; SAVE DE
1818 MVI A,STRT ; START BIT MASK
1819 MVI B,7 ; B WILL COUNT BITS TO SEND
1820 COB5:
1821 SIM ; SEND A BIT
1822 LXI D,IBTIM ; WAIT FOR TTY TO HANDLE IT
1823 CALL DELAY
1824 MOV A,C ; PICK UP BITS LEFT TO SEND
1825 MOV C,A ; LOW ORDER BIT TO CARRY
1826 MOV C,A ; PUT REST BACK
1827 MVI A,STRT ; SHIFTED ENABLE BIT
1828 MVI C,1 ; SHIFT IN DATA BIT
1829 XRI 88H ; COMPLEMENT DATA BIT
1830 DCR B ; DEC COUNT
1831 MOV A,COB5 ; SEND IF MORE BITS NEED TO BE SENT
1832 MVI A,STOPB ; ELSE, SEND STOP BITS
1833 SIM
1834
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>05E1</td>
<td>113012</td>
<td>1835</td>
<td>LUI D,TIM4 ; WAIT 4 BIT TIME (FAKE PARITY + 3 STOP BITS)</td>
</tr>
<tr>
<td>05E4</td>
<td>CDF#05</td>
<td>1836</td>
<td>CALL DELAY</td>
</tr>
<tr>
<td>05E7</td>
<td>D1</td>
<td>1837</td>
<td>POP D</td>
</tr>
<tr>
<td>05E8</td>
<td>C1</td>
<td>1838</td>
<td>POP B ; RESTORE SAVED REGISTERS</td>
</tr>
<tr>
<td>05E9</td>
<td>FB</td>
<td>1839</td>
<td>EI</td>
</tr>
<tr>
<td>05EA</td>
<td>C9</td>
<td>1840</td>
<td>RET ; ALL DONE</td>
</tr>
<tr>
<td>1841</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1842</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1843</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1844</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1845</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1846</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1847</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1848</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1849</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1850</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1851</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1852</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1853</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1854</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#5EB</td>
<td>#EBD</td>
<td>1855</td>
<td>MVI C,CR</td>
</tr>
<tr>
<td>#5ED</td>
<td>CDF#05</td>
<td>1856</td>
<td>CALL ECHO</td>
</tr>
<tr>
<td>#5F0</td>
<td>C9</td>
<td>1857</td>
<td>RET</td>
</tr>
<tr>
<td>1858</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1859</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1860</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1861</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1862</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1863</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1864</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1865</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1866</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1867</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1868</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1869</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1870</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1871</td>
<td>DELAY:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1872</td>
<td>dca D</td>
<td>1872</td>
<td>DECREMENT INPUT ARGUMENT</td>
</tr>
<tr>
<td>1873</td>
<td>mov A,D</td>
<td>1873</td>
<td></td>
</tr>
<tr>
<td>1874</td>
<td>ora E</td>
<td>1874</td>
<td></td>
</tr>
<tr>
<td>1875</td>
<td>jne DELAY ; IF ARGUMENT NOT 0, KEEP GOING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1876</td>
<td>ret</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1877</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1878</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1879</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1880</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1881</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1882</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1883</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1884</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1885</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1886</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1887</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1888</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1889</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1890</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1891</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1892</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1893</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1894</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1895</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1896</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1897</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1898</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1899</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1900</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1901</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1902</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1903</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1904</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1905</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1906</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1907</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1908</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1909</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1910</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1911</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1912</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1913</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1914</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1915</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1916</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1917</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1918</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1919</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1920</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1921</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1922</td>
<td>ERROR:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1923</td>
<td>mvi C,'*'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1924</td>
<td>call ECHO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1925</td>
<td>call CROUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1926</td>
<td>jmp GETCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1927</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1928</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1929</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1930</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1931</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1932</td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1933</td>
<td>;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOC OBJ  SEQ  SOURCE STATEMENT

1934 : OUTPUTS: CARRY - AL'WAYS 0
1935 : CALLS: NOTHING
1936 : DESTRUCTS: 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 : FUNCTION: GETCH
1948 : INPUTS: NONE
1949 : OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
1950 : CALLS: C'I
1951 : DESTRUCTS: A,C,F/F'S
1952 : DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
1953 : TO THE CALLING PROGRAM.
1954 :
1955 GETCH:
061F CD90 C5  1956  CALL  CI         ; GET CHARACTER FROM TERMINAL
0622 E7F  1957  ANI  PRT@#         ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
0624 4F  1958  MOV  C,A         ; PUT VALUE IN C REGISTER FOR RETURN
0625 C9  1959  RET         ;
1960 :
1961 : FUNCTION: GETHX
1962 : INPUTS: NONE
1963 : OUTPUTS: BC - 16 BIT INTEGER
1964 : D - CHARACTER WHICH TERMINATED THE INTEGER
1965 : CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
1966 : 0 IF FIRST CHARACTER IS DELIMITER
1967 : CALLS: GETCH,ECHO,VALLO,VALDG,CNVBN,ERROR
1968 : DESTRUCTS: A,B,C,D,E,F/F'S
1969 : DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
1970 : STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
1971 : INTEGER. IF MORE THAN 4 HEX DIGITS ARE ENTERED,
1972 : ONLY THE LAST 4 ARE USED. THE NUMBER TERMINATES WHEN
1973 : A VALID DELIMITER IS ENCOUNTERED. THE DELIMITER IS
1974 : ALSO RETURNED AS AN OUTPUT OF THE FUNCTION. ILLEGAL
1975 : CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
1976 : ERROR INDICATION. IF THE FIRST (VALID) CHARACTER
1977 : ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
1978 : GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
1979 : OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
1980 : OF BC ARE UNEDEFINED.
1981 :
1982 : GETHX:
0626 E5  1983  PUSH  H         ; SAVE HL
0627 F140 C0  1984  LXI  H,0         ; INITIALIZE RESULT
0628 1E09  1985  MVI  E,0         ; INITIALIZE DIGIT FLAG TO FALSE
1986 :
062C D1F0 C5  1987  CALL  GETCH         ; GET A CHARACTER
062D 4F  1988  MOV  C,A         ; ECHO THE CHARACTER
062E CD90 C5  1989  CALL  ECHO         ; ECHO THE CHARACTER
062F C079 C5  1990  CALL  VALLO         ; SEE IF DELIMITER
0630 D240 C5  1991  FALSE  GHX10         ; NO - BRANCH
0631 51  1992  MOV  D,C         ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
0632 A5  1993  PUSH  H         ;
0633 250C  1994  POP  B         ; MOVE RESULT TO BC
0634 C1  1995  MOV  A,E         ; RESTORE HL
0635 7B  1996  MOV  A,L         ; GET FLAG
0636 B7  1997  MOV  A,#F/P'S         ; SET F/P'S
0637 C230 C7  1998  ORA  A         ;
0638 2307 C5  1999  JNZ  SRET         ; IF FLAG NON-B, A NUMBER HAS BEEN FOUND
0639 8E C5  2000  JZ  FRET         ; ELSE, DELIMITER WAS FIRST CHARACTER
063A 5D58 C7  2001  CALL  VALDG         ; IF NOT DELIMITER, SEE IF DIGIT
063B D211 C5  2002  FALSE  ERROR         ; ERROR IF NOT A VALID DIGIT, EITHER
063C C060 C5  2003  CALL  CNVBN         ; CONVERT DIGIT TO ITS BINARY VALUE
063D 4EFF  2004  MVI  E,#FFH         ; SET DIGIT FLAG NON-B
063E 58 C9  2005  DAD  H         ; *2
063F 52 C9  2006  DAD  H         ; *4
0640 52 C9  2007  DAD  H         ; *8
0641 52 C9  2008  DAD  H         ; *16
0642 54 8680  2009  MVI  B,#B         ; CLEAR UPPER 8 BITS OF BC PAIR
0643 56 4F  2010  MOV  C,A         ; BINARY VALUE OF CHARACTER INTO C
0644 56 89  2011  DAD  B         ; ADD THIS VALUE TO PARTIAL RESULT
0645 C32C C6  2012  JMP  GHX85         ; GET NEXT CHARACTER
2013 :
2014 :
2015 :
2016 :
2017 :
2018 :
2019 :
2020 :
2021 :
2022 :
2023 :
2024 :
2025 :
2026 :
2027 :
2028 :
2029 :
2030 :
2031 :
2032 :
2033 : FUNCTION: GETHM
2034 : INPUTS: C - COUNT OF NUMBERS TO FIND IN INPUT STREAM
2035 : OUTPUTS: TOPE OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON TOP
2036 : OF STACK)
2037 : CALLS: GETHM,HIL0,ERROR
2038 : DESTRUCTS: A,B,C,D,E,F,L,F/P'S
LOC OBJ
SEQ SOURCE STATEMENT
GTHMN: GETMN FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1
AND 3, INCLUSIVE, IN THE INPUT
STREAM AND RETURNS THEIR VALUES ON THE STACK. IF 2
OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
SECOND NUMBERS WILL BE SET EQUAL. THE LAST NUMBER
REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
OR AN ERROR INDICATION WILL RESULT.

GTHMN:
MVI L,3 ; PUT MAXIMUM ARGUMENT COUNT INTO L
MOV A,C ; GET THE ACTUAL ARGUMENT COUNT
ANI J ; FORCE TO MAXIMUM OF 3
MOV Z,A ; IF 0, DON'T BOTHER TO DO ANYTHING
MOV H,A ; ELSE, PUT ACTUAL COUNT INTO H

GTHMN: CALL GTHMX ; GET A NUMBER FROM INPUT STREAM
JNC ERROR ; ERROR IF NOT THERE - TOO FEW NUMBERS

GTHMN: PUSH B ; ELSE, SAVE NUMBER ON STACK
DEC R ; DECREMENT MAXIMUM ARGUMENT COUNT

GTHMN: JS GTHMN ; BRANCH IF NO MORE NUMBERS WANTED
MOV A,D ; ELSE, GET NUMBER TERMINATOR TO A

GTHMN: CPI CR ; SEE IF CARRY RETURN
JS ERROR ; ERROR IF GO - TOO FEW NUMBERS

GTHMN: JMP GTHMN ; ELSE, PROCESS NEXT NUMBER

GTHMN: MOV A,D ; WHEN COUNT 0, CHECK LAST TERMINATOR
CPI CR ; IF NOT, FILL REMAINING ARGUMENTS WITH $FFFFH

GTHMN: CALL MILO ; SEE IF FIRST >= SECOND

GTHMN: MOV A,L ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT

GTHMN: JS GTHMN ; IF YES, 3 NUMBERS WERE INPUT

GTHMN: PUSH B ; IF NOT, FILL REMAINING ARGUMENTS WITH $FFFFH

GTHMN: POP B ; GET THE 3 ARGUMENTS OUT
POP D ; POP H
POP H ; POP H

GTHMN: CALL MILO ; CALL MILO

GTHMN: MOV A,L ; IF FIRST >= SECOND

GTHMN: MOV A,L ; NO - BRANCH

GTHMN: MOV A,L ; INCREMENT COUNT

GTHMN: MOV A,L ; CHECK FOR DE = $FFFF

GTHMN: MOV A,L ; YES - MAKE SECOND EQUAL TO THE FIRST

GTHMN: MOV A,L ; PUT FIRST ON STACK - GET RETURN ADDR

GTHMN: MOV A,L ; IF NOT, FOCUS ORIGINAL DE

GTHMN: MOV A,L ; PUT ADDRESS ADDRESS ON STACK

GTHMN: MOV A,L ; REPLACE TOP RESULT WITH RETURN ADDR

GTHMN: JS GTHMN ; TRY AGAIN

FUNCTION: MILO
INPUTS: DE - 16 BIT INTEGER
HL - 16 BIT INTEGER
OUTPUTS: CARRY = 1 IF HL<DE
1 IF HL>>DE

CALLS: NOTHING

DESCRIPTION: MILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE. THE
INTEGERS ARE TREATED AS UNSIGNED NUMBERS. THE CARRY
BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.

MILO:
MOV B,A ; SAVE RC
MOV B,A ; SAVE A IN B REGISTER
MOV H,A ; SAVE HL PAIR
MOV A,D ; CHECK FOR DE = $FFFF
ORA D ; HIL05 ; WE'RE AUTOMATICALLY DONE IF IT IS
MOV A,H ; INCREMENT HL BY 1
INC H ; WANT TO TEST FOR 0 RESULT AFTER
MOV A,H ; INCREMENTING
INC H ; IF SO, HL MUST HAVE CONTAINED $FFFF
MOV H,A ; IF NOT, RESTORE ORIGINAL HL
INC D ; SAVE DE
MOV A,H ; WANT TO TAKE 2'S COMPLEMENT OF DE CONTENTS
MOV D,A ; 2'S COMPLEMENT OF DE TO DE
MOV A,$FFFF ; THIS OPERATION SETS CARRY PROPERLY
MOV A,D ; RESTORE ORIGINAL DE CONTENTS
MOV A,D ; THIS OPERATION SETS CARRY PROPERLY
MOV D,A ; RESTORE ORIGINAL DE CONTENTS
LOC     OBJ   SEQ     SOURCE STATEMENT
#66E 178 2134  MOV  A,B ; RESTORE ORIGINAL CONTENTS OF A
#66F 173 2135  MOV  B   ; RESTORE ORIGINAL CONTENTS OF BC
#66C 176 2136  RET   ; RETURN WITH CARRY SET AS REQUIRED
#667 177 2137  HLT$   ;
#66C 178 2138  POP  H ; IF HL CONTAINS $FFFFFF, THEN CARRY CAN
#66F 179 2139  MOV  A,B ; ONLY BE SET TO 1
#663 174 2140  POP  B ; RESTORE ORIGINAL CONTENTS OF REGISTERS
#664 175 2141  JMP  SRET ; SET CARRY AND RETURN

;-----------------------------

; FUNCTION: NMOUT
; INPUTS: A - 8 BIT INTEGER
; OUTPUTS: NONE
; CALLS: ECHO,PRVAL
; DESTROYS: A,B,C,D,E,H,L,F/F'S
; DESCRIPTION: NMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER IN THE
; REGISTER INTO 2 ASCII CHARACTERS. THE ASCII CHARACTERS
; ARE THE ONES REPRESENTING THE 8 BITS. THESE TWO
; CHARACTERS ARE SENT TO THE CONSOLE AT THE CURRENT PRINT
; POSITION OF THE CONSOLE.

#667 175 2159  PUSH  H ; SAVE HL - DESTROYED BY PRVAL
#66F 176 2160  PUSH  PSW ; SAVE ARGUMENT
#66C 177 2161  RRC
#667 178 2162  RRC
#66C 179 2163  RRC
#66F 174 2164  RRC
#66D 176 2165  ANL  HCCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
#660 177 2166  MOV  C,A
#66D 178 2167  CALL  PRVAL ; CONVERT LOWER 4 BITS TO ASCII
#66D 179 2168  CALL  ECHO ; SEND TO TERMINAL
#66F 174 2169  POP  PSW ; GET BACK ARGUMENT
#667 175 2170  ANL  HCCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
#660 176 2171  MOV  C,A
#66D 177 2172  CALL  PRVAL
#66D 178 2173  CALL  ECHO
#66F 174 2174  POP  H ; RESTORE SAVED VALUE OF HL
#66E 175 2175  RET

;-----------------------------

; FUNCTION: PRVAL
; INPUTS: C - INTEGER, RANGE 0 TO 255
; OUTPUTS: C - ASCII CHARACTER
; CALLS: NOTHING
; DESTROYS: B,C,H,L,F/F'S
; DESCRIPTION: PRVAL CONVERTS A NUMBER IN THE RANGE 0 TO 255 TO AN ASCII CHARACTER, 0-9,A-F. PRVAL
; DOES NOT CHECK THE VALIDITY OF ITS INPUT ARGUMENT.

#66E 21848 2191  LXI  B, DIGITS ; ADDRESS OF TABLE
#66F 21849 2192  MVI B,0 ; CLEAR HIGH ORDER BITS OF BC
#667 2184A 2193  DAD B ; ADD DIGIT VALUE TO HL ADDRESS
#66F 2184B 2194  MOV C,M ; FETCH CHARACTER FROM MEMORY
#66E 2184C 2195  RET

;-----------------------------

; FUNCTION: REGS
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: ECHO,NMOUT,ERROR,CROUT
; DESTROYS: A,B,C,E,H,L,F/F'S
; DESCRIPTION: REGS DISPLAYS THE CONTENTS OF THE REGISTER SAVE
; LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE. THE
; DISPLAY IS DRIVEN FROM A TABLE, RTAB, WHICH CONTAINS
; THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS,
; AND LENGTH (8 OR 16 BITS).

#66A 21C40 2211  REGS:
#66A 21C41 2212  LXI  H,RTAB ; LOAD HL WITH ADDRESS OF START OF TABLE
#66D 21C42 2213  MOV  C,M ; GET PRINT SYMBOL OF REGISTER
#66D 21C43 2214  MOV  A,C
#66F 21C44 2215  CALL  ECHO ; PRINT SYMBOL OF REGISTER
#66F 21C45 2216  MOV  A,C
#66F 21C46 2217  GRA A ; TEST FOR 0 - END OF TABLE
#66F 21C47 2218  JMI  REG$8 ; IF NOT END, BRANCH
#66F 21C48 2219  JMI  REG$8 ; IF NOT END, BRANCH
#66F 21C49 2220  CALL  CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
#66F 21C4A 2221  RET ; /DISPLAY

;-----------------------------

; CALL ECOC ; ECHO CHARACTER
#66F 21C4B 2222  CALL  ECHO
#66F 21C4C 2223  MVI C, A ; ENTRY POINT OF REGS
#66F 21C4D 2224  CALL  ECHO ; OUTPUT EQUALS SIGN, I.E. A=
#66F 21C4E 2225  INX H ; POINT TO START OF SAVE LOCATION ADDRESS
#66F 21C4F 2226  MOV E,M ; GET L&P OF SAVE LOCATION ADDRESS TO E
#66F 21C50 2227  MOV D,RAMSH SR#8 ; PUT M&P OF SAVE LOC ADDRESS INTO D
#66F 21C51 2228  INX H ; POINT TO LENGTH FLAG
#66F 21C52 2229  LOAX D ; GET CONTENTS OF SAVE ADDRESS
#66F 21C53 2230  CALL  NMOUT ; DISPLAY ON CONSOLE
#66F 21C54 2231  MOV A,M ; GET LENGTH FLAG
#66F 21C55 2232  GRA A ; SET SIGN F/F
#66F 21C56 2233  JZ  REG5 ; IF 0, REGISTER IS 8 BITS

;-----------------------------
LOC OBJ  SEQ  SOURCE STATEMENT

2334 3THLF:
073D D5  2335  PUSH D
0748 E1  2336  POP H ; MOVE ADDRESS OF BYTE INTO HL
0741 79  2337  MOV A,C ; GET VALUE
0742 668F 2338  ANI $0FH ; FORCE TO 4 BITLENGTH
0744 4F  2339  MOV C,A ; PUT VALUE BACK
0745 3FDF8 2340  LDA TEMP ; GET HALF BYTE FLAG
0748 B7  2341  ORA A ; CHECK FOR LOWER HALF
0749 C2S87 2342  JNS STHB5 ; BRANCH IF NOT
074C TE  2343  MOV A,M ; ELSE, GET BYTE
074D E0F0 2344  ANI $0FH ; CLEAR LOWER 4 BITS
074F 61  2345  ORA C ; OR IN VALUE
0750 77  2346  MOV M,A ; PUT BYTE BACK
0751 C9  2347  RET
2348  STHB5:
0752 TE  2349  MOV A,M ; IF UPPER HALF, GET BYTE
0753 668F 2350  ANI $0FH ; CLEAR UPPER 4 BITS
0755 47  2351  MOV B,A ; SAVE BYTE IN B
0756 79  2352  MOV A,C ; GET VALUE
0757 8F  2353  RRC
0758 8F  2354  RRC
0759 8F  2355  RRC
075A 8F  2356  RRC ; ALIGN TO UPPER 4 BITS
075B 8F  2357  ORA B ; OR IN ORIGINAL LOWER 4 BITS
075C 77  2358  MOV M,A ; PUT NEW CONFIGURATION BACK
2359  RET
2360 ;
2361 ;*****************************************************************
2362 ; FUNCTION: VALDG
2363 ; INPUTS: C - ASCII CHARACTER
2364 ; OUTPUTS: CARRY = 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
2365 ; # OTHERWISE
2366 ; CALLS: NOTHING
2367 ; DESTROYS: A,F,P'S
2368 ; DESCRIPTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
2369 ; AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
2370 ; (0-9,A-F) AND FAILURE OTHERWISE.
2371 ;*************************************************************************
2372 ; VALDG:
2373 75E 79
2374 75F PE38
2375 761 FAEC86
2376 764 FE39
2377 766 FA2S7
2378 769 CA3287
2379 76C FC84
2380 773 FC2C86
2381 776 C32S7
2382 777 PE2C
2383 77C CA3287
2384 77F FE8D
2385 780 CA3287
2386 784 FC84
2387 786 CA3287
2388 789 C32C86
2389 ;*****************************************************************
2390 ; FUNCTION: VALDL
2391 ; INPUTS: C - CHARACTER
2392 ; OUTPUTS: CARRY = 1 IF INPUT ARGUMENT VALID DELIMITER
2393 ; # OTHERWISE
2394 ; CALLS: NOTHING
2395 ; DESTROYS: A,F,P'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 ; VALDL:
2401 0779 79
2402 077A PE2C
2403 077C CA3287
2404 077F FE8D
2405 0780 CA3287
2406 0784 FC84
2407 0786 CA3287
2408 0789 C32C86
2409 ;*****************************************************************
2410 ; MONITOR TABLES
2411 ;*************************************************************************
2412 ; SIGNON: ; SIGNON MESSAGE
2413 078C 80
2414 078D 0A
2415 078E 53444020
2416 0792 39352020
2417 0796 28564552
2418 0799 283223E1
2419 079E 9D
2420 079F 8A
2421 0814 2422 SIGMON EQU $-SIGNON ; LENGTH OF SIGNON MESSAGE
2423 2424 CADR: ; TABLE OF ADDRESSES OF COMMAND ROUTINES
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>#7A0</td>
<td>#00#</td>
<td>2427</td>
<td>DW 0 ; DUMMY</td>
</tr>
<tr>
<td>#7A2</td>
<td>#485</td>
<td>2428</td>
<td>DW XCMD</td>
</tr>
<tr>
<td>#7A4</td>
<td>#004</td>
<td>2429</td>
<td>DW SCMD</td>
</tr>
<tr>
<td>#7A6</td>
<td>#004</td>
<td>2430</td>
<td>DW MCMD</td>
</tr>
<tr>
<td>#7A8</td>
<td>#004</td>
<td>2431</td>
<td>DW ICMD</td>
</tr>
<tr>
<td>#7AA</td>
<td>#004</td>
<td>2432</td>
<td>DW GCMD</td>
</tr>
<tr>
<td>#7AC</td>
<td>#7E4</td>
<td>2433</td>
<td>DW DCMC</td>
</tr>
<tr>
<td>#7A4</td>
<td>#44</td>
<td>2434</td>
<td></td>
</tr>
<tr>
<td>#7AF</td>
<td>#47</td>
<td>2435</td>
<td>CTAB: ; TABLE OF VALID COMMAND CHARACTERS</td>
</tr>
<tr>
<td>#7B0</td>
<td>#49</td>
<td>2436</td>
<td>DB 'G'</td>
</tr>
<tr>
<td>#7B1</td>
<td>#40</td>
<td>2437</td>
<td>DB 'I'</td>
</tr>
<tr>
<td>#7B2</td>
<td>#35</td>
<td>2438</td>
<td>DB 'K'</td>
</tr>
<tr>
<td>#7B3</td>
<td>#58</td>
<td>2439</td>
<td>DB 'S'</td>
</tr>
<tr>
<td>#7B6</td>
<td>#0</td>
<td>2440</td>
<td>DB 'X'</td>
</tr>
<tr>
<td>#7B8</td>
<td>#0</td>
<td>2441</td>
<td>NCMDS EQU 5-CTAB ; NUMBER OF VALID COMMANDS</td>
</tr>
<tr>
<td>#7B4</td>
<td>#38</td>
<td>2442</td>
<td></td>
</tr>
<tr>
<td>#7B5</td>
<td>#31</td>
<td>2443</td>
<td>DIGITS:</td>
</tr>
<tr>
<td>#7B6</td>
<td>#32</td>
<td>2444</td>
<td>DB '0'</td>
</tr>
<tr>
<td>#7B7</td>
<td>#33</td>
<td>2445</td>
<td>DB '1'</td>
</tr>
<tr>
<td>#7B8</td>
<td>#34</td>
<td>2446</td>
<td>DB '2'</td>
</tr>
<tr>
<td>#7B9</td>
<td>#35</td>
<td>2447</td>
<td>DB '3'</td>
</tr>
<tr>
<td>#7BA</td>
<td>#36</td>
<td>2448</td>
<td>DB '4'</td>
</tr>
<tr>
<td>#7BB</td>
<td>#37</td>
<td>2449</td>
<td>DB '5'</td>
</tr>
<tr>
<td>#7BC</td>
<td>#38</td>
<td>2450</td>
<td>DB '6'</td>
</tr>
<tr>
<td>#7BD</td>
<td>#39</td>
<td>2451</td>
<td>DB '7'</td>
</tr>
<tr>
<td>#7BE</td>
<td>#40</td>
<td>2452</td>
<td>DB '8'</td>
</tr>
<tr>
<td>#7BF</td>
<td>#41</td>
<td>2453</td>
<td>DB '9'</td>
</tr>
<tr>
<td>#7C0</td>
<td>#42</td>
<td>2454</td>
<td>DB 'A'</td>
</tr>
<tr>
<td>#7C1</td>
<td>#43</td>
<td>2455</td>
<td>DB 'B'</td>
</tr>
<tr>
<td>#7C2</td>
<td>#44</td>
<td>2456</td>
<td>DB 'C'</td>
</tr>
<tr>
<td>#7C3</td>
<td>#45</td>
<td>2457</td>
<td>DB 'D'</td>
</tr>
<tr>
<td>#7C4</td>
<td>#46</td>
<td>2458</td>
<td>DB 'E'</td>
</tr>
<tr>
<td>#7C5</td>
<td>#47</td>
<td>2459</td>
<td>DB 'F'</td>
</tr>
<tr>
<td>#7C6</td>
<td>#48</td>
<td>2460</td>
<td></td>
</tr>
<tr>
<td>#7C7</td>
<td>#49</td>
<td>2461</td>
<td>; TABLE OF REGISTER INFORMATION</td>
</tr>
<tr>
<td>#7C8</td>
<td>#0</td>
<td>2462</td>
<td>RTAB:</td>
</tr>
<tr>
<td>#7C9</td>
<td>#0</td>
<td>2463</td>
<td>DB 'A' ; REGISTER IDENTIFIER</td>
</tr>
<tr>
<td>#7CA</td>
<td>#0</td>
<td>2464</td>
<td>DB ASAV AND #FF8 ; ADDRESS OF REGISTER SAVE LOCATION</td>
</tr>
<tr>
<td>#7CB</td>
<td>#0</td>
<td>2465</td>
<td>DB 0 ; LENGTH FLAG - #8 BITS, I=16 BITS</td>
</tr>
<tr>
<td>#7CC</td>
<td>#0</td>
<td>2466</td>
<td>ASAVS EQU $-RTAB ; SIZE OF AN ENTRY IN THIS TABLE</td>
</tr>
<tr>
<td>#7CD</td>
<td>#0</td>
<td>2471</td>
<td>DB 'B'</td>
</tr>
<tr>
<td>#7CE</td>
<td>#0</td>
<td>2472</td>
<td>DB BSAV AND #FF8</td>
</tr>
<tr>
<td>#7CF</td>
<td>#0</td>
<td>2473</td>
<td>DB 'C'</td>
</tr>
<tr>
<td>#7D0</td>
<td>#0</td>
<td>2474</td>
<td>DB CSAV AND #FF8</td>
</tr>
<tr>
<td>#7D1</td>
<td>#0</td>
<td>2475</td>
<td>DB 'D'</td>
</tr>
<tr>
<td>#7D2</td>
<td>#0</td>
<td>2476</td>
<td>DB DSAV AND #FF8</td>
</tr>
<tr>
<td>#7D3</td>
<td>#0</td>
<td>2477</td>
<td>DB 'E'</td>
</tr>
<tr>
<td>#7D4</td>
<td>#0</td>
<td>2478</td>
<td>DB ESAVE AND #FF8</td>
</tr>
<tr>
<td>#7D5</td>
<td>#0</td>
<td>2479</td>
<td>DB 'F'</td>
</tr>
<tr>
<td>#7D6</td>
<td>#0</td>
<td>2480</td>
<td>DB FSAVE AND #FF8</td>
</tr>
<tr>
<td>#7D7</td>
<td>#0</td>
<td>2481</td>
<td>DB 0</td>
</tr>
<tr>
<td>#7D8</td>
<td>#0</td>
<td>2482</td>
<td>DB 0</td>
</tr>
<tr>
<td>#7D9</td>
<td>#0</td>
<td>2483</td>
<td>DB ISAV AND #FF8</td>
</tr>
<tr>
<td>#7DA</td>
<td>#0</td>
<td>2484</td>
<td>DB 'I'</td>
</tr>
<tr>
<td>#7DB</td>
<td>#0</td>
<td>2485</td>
<td>DB 'H'</td>
</tr>
<tr>
<td>#7DC</td>
<td>#0</td>
<td>2486</td>
<td>DB HSAY AND #FF8</td>
</tr>
<tr>
<td>#7DD</td>
<td>#0</td>
<td>2487</td>
<td>DB 0</td>
</tr>
<tr>
<td>#7DE</td>
<td>#0</td>
<td>2488</td>
<td>DB 'L'</td>
</tr>
<tr>
<td>#7DF</td>
<td>#0</td>
<td>2489</td>
<td>DB LSAV AND #FF8</td>
</tr>
<tr>
<td>#7E0</td>
<td>#0</td>
<td>2490</td>
<td>DB 0</td>
</tr>
<tr>
<td>#7E1</td>
<td>#0</td>
<td>2491</td>
<td>DB 'M'</td>
</tr>
<tr>
<td>#7E2</td>
<td>#0</td>
<td>2492</td>
<td>DB HSAY AND #FF8</td>
</tr>
<tr>
<td>#7E3</td>
<td>#0</td>
<td>2493</td>
<td>DB 1</td>
</tr>
<tr>
<td>#7E4</td>
<td>#0</td>
<td>2494</td>
<td>DB 'N'</td>
</tr>
<tr>
<td>#7E5</td>
<td>#0</td>
<td>2495</td>
<td>DB SSAV AND #FF8</td>
</tr>
<tr>
<td>#7E6</td>
<td>#0</td>
<td>2496</td>
<td>DB 1</td>
</tr>
<tr>
<td>#7E7</td>
<td>#0</td>
<td>2497</td>
<td>DB 'P'</td>
</tr>
<tr>
<td>#7E8</td>
<td>#0</td>
<td>2498</td>
<td>DB PSAV AND #FF8</td>
</tr>
<tr>
<td>#7E9</td>
<td>#0</td>
<td>2499</td>
<td>DB 1</td>
</tr>
<tr>
<td>#7EA</td>
<td>#0</td>
<td>2500</td>
<td>DB 0 ; END OF TABLE MARKERS</td>
</tr>
<tr>
<td>#7EC</td>
<td>#0</td>
<td>2501</td>
<td></td>
</tr>
<tr>
<td>#7EF</td>
<td>#0</td>
<td>2502</td>
<td></td>
</tr>
<tr>
<td>#7F0</td>
<td>#0</td>
<td>2503</td>
<td>ORG BRTAB ; BRANCH TABLE FOR USER ACCESSIBLE ROUTINES</td>
</tr>
<tr>
<td>#7F1</td>
<td>#0</td>
<td>2504</td>
<td></td>
</tr>
<tr>
<td>#7F2</td>
<td>#0</td>
<td>2505</td>
<td>JMP CO ; TTY CONSOLE OUTPUT</td>
</tr>
<tr>
<td>#7F3</td>
<td>#0</td>
<td>2506</td>
<td>JMP CI ; TTY CONSOLE INPUT</td>
</tr>
<tr>
<td>#7F4</td>
<td>#0</td>
<td>2507</td>
<td></td>
</tr>
</tbody>
</table>

**Jump Instructions:**

- #7F0

**In the Following Locations, the User May Place Jump Instructions To:**

- #28C

- #28C2

- #28C3

- #28C4

- #28C5

- #28C6

- #28C7

- #28C8

- #28C9

- #28CA

- #28D

- #28D1

- #28D2

- #28D3

- #28D4

- #28D5

- #28D6

- #28D7

- #28D8

- #28D9

- #28DA

- #28DB

- #28DC

- #28DD

- #28DE

- #28DF

- #28E0

- #28E1

- #28E2

- #28E3

- #28E4

- #28E5

- #28E6

- #28E7

- #28E8

- #28E9

- #28EA

- #28EB

- #28EC

- #28ED

- #28EE

- #28EF

- #28F0

- #28F1

- #28F2

- #28F3

- #28F4

- #28F5

- #28F6

- #28F7

- #28F8

- #28F9

- #28FA

- #28FB

- #28FC

- #28FD

- #28FE

- #28FF
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICM05</td>
<td>1543#</td>
<td>1552 1566</td>
</tr>
<tr>
<td>ICM10</td>
<td>1561#</td>
<td>1563#</td>
</tr>
<tr>
<td>ICM20</td>
<td>1555#</td>
<td>1567#</td>
</tr>
<tr>
<td>ICM25</td>
<td>1549#</td>
<td>1550#</td>
</tr>
<tr>
<td>ICMQ</td>
<td>1537#</td>
<td>2431</td>
</tr>
<tr>
<td>ININT</td>
<td>292#</td>
<td>751#</td>
</tr>
<tr>
<td>INSO</td>
<td>657#</td>
<td>779#</td>
</tr>
<tr>
<td>INVNT</td>
<td>1329#</td>
<td>1564</td>
</tr>
<tr>
<td>ISAV</td>
<td>226#</td>
<td>403#</td>
</tr>
<tr>
<td>KBKIT</td>
<td>1205#</td>
<td>563</td>
</tr>
<tr>
<td>KMODE</td>
<td>1229#</td>
<td>166</td>
</tr>
<tr>
<td>LETHA</td>
<td>1142#</td>
<td>1287</td>
</tr>
<tr>
<td>LETIB</td>
<td>1144#</td>
<td>1206</td>
</tr>
<tr>
<td>LETRC</td>
<td>1146#</td>
<td>1289 1210</td>
</tr>
<tr>
<td>LETRD</td>
<td>1148#</td>
<td>1210</td>
</tr>
<tr>
<td>LETRE</td>
<td>1150#</td>
<td>1172 1173</td>
</tr>
<tr>
<td>LETER</td>
<td>1152#</td>
<td>1212</td>
</tr>
<tr>
<td>LETRH</td>
<td>1154#</td>
<td>1214 1216</td>
</tr>
<tr>
<td>LETRI</td>
<td>1156#</td>
<td>1213</td>
</tr>
<tr>
<td>LETER</td>
<td>1158#</td>
<td>1212 1217</td>
</tr>
<tr>
<td>LETER</td>
<td>1164#</td>
<td>1172</td>
</tr>
<tr>
<td>LETRS</td>
<td>1135#</td>
<td>1216 1217</td>
</tr>
<tr>
<td>LF</td>
<td>1330#</td>
<td>1903</td>
</tr>
<tr>
<td>LOWER</td>
<td>1331#</td>
<td></td>
</tr>
<tr>
<td>LSAB</td>
<td>175#</td>
<td>437 1019</td>
</tr>
<tr>
<td>LSNODM</td>
<td>1387#</td>
<td>2424#</td>
</tr>
<tr>
<td>MCMQ5</td>
<td>1592#</td>
<td>1689</td>
</tr>
<tr>
<td>MCMQ6</td>
<td>1596#</td>
<td>2430</td>
</tr>
<tr>
<td>MNSTK</td>
<td>124#</td>
<td>265 1018</td>
</tr>
<tr>
<td>MSGQ1</td>
<td>1384#</td>
<td>1393</td>
</tr>
<tr>
<td>MCMQ7</td>
<td>1426#</td>
<td>2442#</td>
</tr>
<tr>
<td>NEWLN</td>
<td>1336#</td>
<td>1490</td>
</tr>
<tr>
<td>NMOV</td>
<td>1473#</td>
<td>1475 1488</td>
</tr>
<tr>
<td>NTBL</td>
<td>962#</td>
<td>1205#</td>
</tr>
<tr>
<td>NODQ</td>
<td>125#</td>
<td>244 249</td>
</tr>
<tr>
<td>NUMQ5</td>
<td>273#</td>
<td>1100#</td>
</tr>
<tr>
<td>NUMRQ</td>
<td>807#</td>
<td>1241#</td>
</tr>
<tr>
<td>NXRQ</td>
<td>334#</td>
<td>889#</td>
</tr>
<tr>
<td>OBTQM</td>
<td>1351#</td>
<td>1823</td>
</tr>
<tr>
<td>ORBUFF</td>
<td>716#</td>
<td>734 2556#</td>
</tr>
<tr>
<td>OUTQ1</td>
<td>833#</td>
<td>310#</td>
</tr>
<tr>
<td>OUTQ2</td>
<td>837#</td>
<td>842#</td>
</tr>
<tr>
<td>OUTQ5</td>
<td>844#</td>
<td>864</td>
</tr>
<tr>
<td>OUTQ7</td>
<td>852#</td>
<td>585 585#</td>
</tr>
<tr>
<td>OUTQ7</td>
<td>247#</td>
<td>252 370</td>
</tr>
<tr>
<td>PCHIQ</td>
<td>1234#</td>
<td>254##</td>
</tr>
<tr>
<td>PCLIQ</td>
<td>1248#</td>
<td>2544#</td>
</tr>
<tr>
<td>PERIO</td>
<td>129#</td>
<td>330 352</td>
</tr>
<tr>
<td>PRTQ1</td>
<td>129#</td>
<td>271</td>
</tr>
<tr>
<td>PRVQ</td>
<td>1337#</td>
<td>1961</td>
</tr>
<tr>
<td>PVAV</td>
<td>177#</td>
<td>363 398</td>
</tr>
<tr>
<td>RAMST</td>
<td>93#</td>
<td>124 137</td>
</tr>
<tr>
<td>RDLU</td>
<td>887#</td>
<td>890#</td>
</tr>
<tr>
<td>RDMRE</td>
<td>272#</td>
<td>351 304</td>
</tr>
<tr>
<td>READ</td>
<td>130#</td>
<td>755</td>
</tr>
<tr>
<td>REGB9</td>
<td>22144#</td>
<td>2241</td>
</tr>
<tr>
<td>REGB10</td>
<td>2216#</td>
<td>2211#</td>
</tr>
<tr>
<td>REGL15</td>
<td>2233#</td>
<td>2337#</td>
</tr>
<tr>
<td>REGIS</td>
<td>1664#</td>
<td>2212#</td>
</tr>
<tr>
<td>RES10</td>
<td>188#</td>
<td>223#</td>
</tr>
<tr>
<td>RETP</td>
<td>666#</td>
<td>806# 906#</td>
</tr>
<tr>
<td>RETQ1</td>
<td>811#</td>
<td>924# 1051</td>
</tr>
<tr>
<td>RGAB5</td>
<td>22649#</td>
<td>2271</td>
</tr>
<tr>
<td>RGAR10</td>
<td>2269#</td>
<td>2372#</td>
</tr>
<tr>
<td>RGAR5</td>
<td>1668#</td>
<td>2261#</td>
</tr>
<tr>
<td>RGLOC</td>
<td>318#</td>
<td>327 936#</td>
</tr>
<tr>
<td>RGAM5</td>
<td>317#</td>
<td>957#</td>
</tr>
<tr>
<td>RGFB2</td>
<td>1807#</td>
<td>1100#</td>
</tr>
<tr>
<td>RGFFR</td>
<td>86#</td>
<td>810 937 958</td>
</tr>
<tr>
<td>RDTIQ</td>
<td>939#</td>
<td>12278 1241</td>
</tr>
<tr>
<td>RNUSE</td>
<td>98#</td>
<td>124 137</td>
</tr>
<tr>
<td>RSET15</td>
<td>197#</td>
<td>2510#</td>
</tr>
<tr>
<td>RSET6</td>
<td>287#</td>
<td>2519#</td>
</tr>
<tr>
<td>RSET7</td>
<td>217#</td>
<td>2521#</td>
</tr>
<tr>
<td>RSRB5</td>
<td>1881#</td>
<td>1804#</td>
</tr>
<tr>
<td>RSRQ10</td>
<td>1808#</td>
<td>1806 1809#</td>
</tr>
<tr>
<td>RSTQ5</td>
<td>212#</td>
<td>2520#</td>
</tr>
<tr>
<td>RSTD1</td>
<td>371#</td>
<td>427 993#</td>
</tr>
<tr>
<td>RTAB</td>
<td>2213#</td>
<td>2262# 2462#</td>
</tr>
<tr>
<td>RTAB5</td>
<td>1723#</td>
<td>2263# 2466#</td>
</tr>
<tr>
<td>SCNB5</td>
<td>16264#</td>
<td>1643</td>
</tr>
<tr>
<td>SCMQ10</td>
<td>1629#</td>
<td>1632#</td>
</tr>
<tr>
<td>SCMQ15</td>
<td>1639#</td>
<td>1641#</td>
</tr>
<tr>
<td>SCMQ20</td>
<td>1622#</td>
<td>2429</td>
</tr>
<tr>
<td>SDRB5</td>
<td>71#</td>
<td></td>
</tr>
<tr>
<td>SDRQG</td>
<td>311#</td>
<td>1036#</td>
</tr>
<tr>
<td>SGAM6</td>
<td>245#</td>
<td>11757</td>
</tr>
<tr>
<td>SGMOT</td>
<td>258#</td>
<td>1176#</td>
</tr>
<tr>
<td>SGNODM</td>
<td>1386#</td>
<td>2422# 2424</td>
</tr>
<tr>
<td>SKLQ1</td>
<td>181#</td>
<td>137</td>
</tr>
<tr>
<td>SPMRV</td>
<td>1237#</td>
<td>2545#</td>
</tr>
<tr>
<td>SPLQ2</td>
<td>1238#</td>
<td>2547#</td>
</tr>
<tr>
<td>SRET</td>
<td>280#</td>
<td>214 2291#</td>
</tr>
<tr>
<td>SSAY</td>
<td>183#</td>
<td>445 1015 2495</td>
</tr>
<tr>
<td>SSTD7</td>
<td>382#</td>
<td>457 1115</td>
</tr>
<tr>
<td>SSTRH</td>
<td>1348#</td>
<td>1828</td>
</tr>
<tr>
<td>SSTRQ5</td>
<td>2342#</td>
<td>2340#</td>
</tr>
<tr>
<td>STFPQ</td>
<td>1568#</td>
<td>1571 2389#</td>
</tr>
<tr>
<td>STHLF</td>
<td>1556#</td>
<td>2314 2339#</td>
</tr>
<tr>
<td>STOPB</td>
<td>1342#</td>
<td>1833</td>
</tr>
<tr>
<td>STOP1</td>
<td>388#</td>
<td>402#</td>
</tr>
</tbody>
</table>
STP21  409  412#
STP22  411  416#
STP23  414  418#
STP25  192  429#
START  1343#  1819
SUB85  476#  583
SUB1#  490  498#
SUB15  478  584#
SUBST  467#  1117
TEMP  405  417  452  1541  1559  1565  1674  1676  1713  1729  2318  2340  2557#
TERM  1344#  1540
TIM4  1352#  1835
TIMER  140#  419  422
TIMHI  132#  421
TIMLO  133#  423
TMODE  134#  419
TRUE  152#  337  1551
TSTRT  135#  425
UBRIN  103#  137
UNMSK  136#  227  455
UPDAD  409  591  18644#
UPDST  322  485  593  1805#
UPPER  1345#  1540
UBCSR  424  432  570  2561#
USINT  221  2522#
USRR  137#  2516
VALDG  1553#  2810  2375#
VALDL  1559#  1998  2402#
WAIT  1353#  1761
WAITM  88#  139  1349
XCM85  1663  1666#
XCM16  1675#  1726
XCM16  1678  1681#
XCM16  1684  1687#
XCM48  1702  1706#
XCM25  1717  1720#
XCM22  1722#  1732
XCM38  1711  1727#
XCMO  1657#  2426
ZERO  1320#  1175

CROSS REFERENCE COMPLETE
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGNAM</td>
<td>317</td>
</tr>
<tr>
<td>RGPTB</td>
<td>1047</td>
</tr>
<tr>
<td>RGPTR</td>
<td>806</td>
</tr>
<tr>
<td>RTCBL</td>
<td>939</td>
</tr>
<tr>
<td>RMUSE</td>
<td>98</td>
</tr>
<tr>
<td>RSET5</td>
<td>197</td>
</tr>
<tr>
<td>RSET6</td>
<td>207</td>
</tr>
<tr>
<td>RSET7</td>
<td>217</td>
</tr>
<tr>
<td>RSR05</td>
<td>1001</td>
</tr>
<tr>
<td>RSR10</td>
<td>1003</td>
</tr>
<tr>
<td>RST05</td>
<td>212</td>
</tr>
<tr>
<td>RSTOR</td>
<td>371</td>
</tr>
<tr>
<td>RTAB</td>
<td>2213</td>
</tr>
<tr>
<td>RTAB5</td>
<td>1723</td>
</tr>
<tr>
<td>SCH05</td>
<td>1626</td>
</tr>
<tr>
<td>SCH10</td>
<td>1629</td>
</tr>
<tr>
<td>SCH15</td>
<td>1639</td>
</tr>
<tr>
<td>SCHD</td>
<td>1622</td>
</tr>
<tr>
<td>SDBK5</td>
<td>71</td>
</tr>
<tr>
<td>SETRG</td>
<td>311</td>
</tr>
<tr>
<td>SGNAD</td>
<td>245</td>
</tr>
<tr>
<td>SGNDT</td>
<td>250</td>
</tr>
<tr>
<td>SGNON</td>
<td>1386</td>
</tr>
<tr>
<td>SKLN</td>
<td>101</td>
</tr>
<tr>
<td>SPF5V</td>
<td>1237</td>
</tr>
<tr>
<td>SPLSV</td>
<td>1338</td>
</tr>
<tr>
<td>SRET</td>
<td>2007</td>
</tr>
<tr>
<td>SSAV</td>
<td>183</td>
</tr>
<tr>
<td>STREP</td>
<td>382</td>
</tr>
<tr>
<td>STRT</td>
<td>134</td>
</tr>
<tr>
<td>STH05</td>
<td>2342</td>
</tr>
<tr>
<td>STHP0</td>
<td>1568</td>
</tr>
<tr>
<td>STULP</td>
<td>1558</td>
</tr>
<tr>
<td>STOPB</td>
<td>1342</td>
</tr>
<tr>
<td>STP20</td>
<td>388</td>
</tr>
<tr>
<td>STP21</td>
<td>409</td>
</tr>
<tr>
<td>STP22</td>
<td>411</td>
</tr>
<tr>
<td>STP23</td>
<td>414</td>
</tr>
<tr>
<td>STP25</td>
<td>414</td>
</tr>
<tr>
<td>STRT</td>
<td>1343</td>
</tr>
<tr>
<td>SUB05</td>
<td>476</td>
</tr>
<tr>
<td>SUB10</td>
<td>490</td>
</tr>
<tr>
<td>SUB15</td>
<td>478</td>
</tr>
<tr>
<td>SUBSET</td>
<td>475</td>
</tr>
<tr>
<td>TEMP</td>
<td>405</td>
</tr>
<tr>
<td>TERM</td>
<td>1344</td>
</tr>
<tr>
<td>TIM2</td>
<td>1352</td>
</tr>
<tr>
<td>TIMER</td>
<td>140</td>
</tr>
<tr>
<td>TIMHI</td>
<td>132</td>
</tr>
<tr>
<td>TIMLO</td>
<td>130</td>
</tr>
<tr>
<td>TMODE</td>
<td>134</td>
</tr>
<tr>
<td>TRUE</td>
<td>152</td>
</tr>
<tr>
<td>TSSTRT</td>
<td>135</td>
</tr>
<tr>
<td>UBLNL</td>
<td>103</td>
</tr>
<tr>
<td>UNNGX</td>
<td>136</td>
</tr>
<tr>
<td>UPDAD</td>
<td>480</td>
</tr>
<tr>
<td>UPDST</td>
<td>322</td>
</tr>
<tr>
<td>UPPER</td>
<td>1345</td>
</tr>
<tr>
<td>USCR</td>
<td>424</td>
</tr>
<tr>
<td>USINT</td>
<td>221</td>
</tr>
<tr>
<td>USRBR</td>
<td>137</td>
</tr>
<tr>
<td>VALDG</td>
<td>1553</td>
</tr>
<tr>
<td>VALDL</td>
<td>1550</td>
</tr>
<tr>
<td>WAIT</td>
<td>1353</td>
</tr>
<tr>
<td>WAITS</td>
<td>80</td>
</tr>
<tr>
<td>XCM05</td>
<td>1663</td>
</tr>
<tr>
<td>XCM10</td>
<td>1675</td>
</tr>
<tr>
<td>XCM15</td>
<td>1676</td>
</tr>
<tr>
<td>XCM18</td>
<td>1684</td>
</tr>
<tr>
<td>XCM20</td>
<td>1702</td>
</tr>
<tr>
<td>XCM25</td>
<td>1717</td>
</tr>
<tr>
<td>XCM27</td>
<td>1722</td>
</tr>
<tr>
<td>XCM30</td>
<td>1711</td>
</tr>
<tr>
<td>XCMD</td>
<td>1657</td>
</tr>
<tr>
<td>ZER0</td>
<td>128</td>
</tr>
</tbody>
</table>

**CROSS REFERENCE COMPLETE**
REQUEST FOR READER'S COMMENTS

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

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

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


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


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


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


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

NAME ___________________________________ DATE ______________________
TITLE ___________________________________
COMPANY NAME/DEPARTMENT ________________________________
ADDRESS ________________________________________________
CITY _______________________________ STATE ____________ ZIP CODE __________________

Please check here if you require a written reply. □
WE'D LIKE YOUR COMMENTS ... 

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

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA
POSTAGE WILL BE PAID BY ADDRESSEE

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