MCS-80 System Design Kit

- CPU
- System Controller
- System Bus
- Peripheral Interface
- PPM
- RAM
- UART
- Bootstrap
- Clock Generator

Intel Corporation 1978
## CONTENTS

Data Sheet .................................................. i
Specifications ............................................. ii

### CHAPTER 1
**INTRODUCTION**

- General .................................................. 1
- Suggestions .............................................. 1
- Functional Definition ................................ 1
- Registration Card Information ..................... 2

### CHAPTER 2
**HOW TO ASSEMBLE THE KIT**

- General .................................................. 3
- Tools Required ........................................... 3
- Optional Rotary Switch ................................. 3
- Parts List ............................................... 3
- Assembly Procedure ................................... 6
- Assembly of Area One .................................. 7
- Assembly of Area Two .................................. 10
- Assembly of Area Three ................................ 12

### CHAPTER 3
**THEORY OF OPERATION**

- General .................................................. 15
- System Busses .......................................... 16
- Reset Switch ............................................ 16
- Clock Generator and Clock Crystal ................. 16
- 8080A CPU ............................................... 16
- System Controller ....................................... 16
- Address Buffers (Optional) ........................... 17
- SDK-80 Memory .......................................... 17
- Baud Rate Generator .................................... 18
- I/O Communication Interface .......................... 18
- Peripheral Interfaces ................................. 18

### CHAPTER 4
**FINAL ASSEMBLY AND CHECKOUT**

- General .................................................. 19
- Jumper-Wiring the Board ............................... 19
- Installing Integrated Circuits ....................... 21
- Start-Up Procedure ..................................... 23
- Troubleshooting Hints .................................. 23

### CHAPTER 5
**SDK-80 MONITOR**

- Introduction ........................................... 24
- Monitor Operations ..................................... 24

### APPENDIX A

- Monitor Listing ......................................... 29

### APPENDIX B

- SDK-80 Schematics .................................... 57

### APPENDIX C

- Board Layout with Component Values ............... 61
8080 SYSTEM DESIGN KIT (SDK-80)

- Complete Single Board Microcomputer System including CPU, Memory and I/O
- Easy to Assemble Kit-Form
- High-Performance (2µs Instruction Cycle)
- Interfaces Directly with most Terminals (75-4800 Baud)
- Large Wire-Wrap area for Custom Interfaces
- Extensive System Monitor Software in ROM
- PC Board Format and Power, Compatible with INTELLEC® MDS
- Complete MCS-80™ User’s Library

The 8080 System Design Kit (SDK-80) is a complete, single board, microcomputer system in kit form. It contains all necessary components, including resistors, caps, crystal and miscellaneous hardware to complete construction. Included is a pre-programmed ROM that contains the system monitor for general software utilities and system diagnostics.

All that is required for operation are power supplies and a suitable terminal; TTY, CRT, etc., (level conversions and baud rate generation included on board).

The SDK-80 is an inexpensive, high-performance prototype system that has designed-in flexibility for simple interface to the users application.

Inte1 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.
SDK-80 SPECIFICATIONS

Central Processor
CPU: 8080A
Instruction Cycle: 1.95 microsecond
Tcy: 488 ns

Memory
ROM: 2K bytes (expandable to 4K bytes) 8708/8308
RAM: 256 bytes (expandable to 1K bytes) 8111
Addressing:
  ROM 0000-0FFF
  RAM 1000-13FF

Input/Output
Parallel: One 8255 for 24 lines (expandable to 48 lines).
Serial: One 8251 USART.
  On-board baud rate generator (jumper selectable).
  Baud Rates: 75 1200 110 2400 300 4800 600

Interfaces
Bus: All signals TTL compatible.
Parallel I/O: All signals TTL compatible.
Serial I/O: RS232C/EIA
  20 mil A. Current loop TTY
  TTL (one TTL load)

Interrupts
Single level: Generates RST7 vector TTL compatible input.

DMA
Hold Request: Jumper selectable.

Software
System Monitor: Pre-programmed 8708 or 8308 ROM Address; 0000-03FF.

Features:
  Display Memory Contents (D)
  Move blocks of memory (M)
  Substitute memory locations (S)
  Insert hex code (I)
  Examine Registers (X)
  Program Control (G)
  Power-up start or system reset start.

I/O: Console Device (serial I/O)

Literature
Design Library:
  8080 Users Manual
  8080 Assembly Language Manual
  PL/M Programming Manual
  MDS Brochure
  Reference Card (Programmers)
  SDK-80 User's Guide

Connectors
I/O: 25 pin female (RS232C)
PCB: MDS format

Physical Characteristics (MDS Mechanical formal)
Width: 12.0 in.
Height: 6.75 in.
Depth: 0.50 in.
Weight: approx. 12 oz.

Electrical Characteristics (DC Power)
  $V_{CC}$ 5V ±5% 1.3 Amps
  $V_{DD}$ 12V ±5% .35 Amps
  $V_{BB}$ -10V ±5% .20 Amps
  or -12V ±5%

Environmental
Operating Temperature: 0-70°C
CHAPTER 1
INTRODUCTION

GENERAL
The 8080 System Design Kit (SDK-80) is a complete microcomputer system in kit form. It is simple to assemble (construction time is 6 hours) and provides an excellent training/prototype vehicle for evaluation of the 8080 microcomputer system (MCS-80™).

The SDK-80 is an extremely flexible design and allows easy interface to an existing application or custom interface development.

An extensive system monitor is included in a pre-programmed ROM for general software utilities and system diagnostics.

The System Design Kit User’s Guide will instruct the user how to assemble his kit and configure it to match the selected terminal and peripheral devices. It is suggested that the User’s Guide be followed in the exact sequence that it is written to assure successful completion of the system.

SUGGESTIONS
The 8080 Microcomputer Systems User’s Manual is included with the SDK-80 and it would be extremely beneficial to the user that he read and understand the operation of the 8080A and associated peripheral components prior to beginning the assembly of the SDK-80.

Every effort has been made to allow the SDK-80 to interface directly with most common terminals but with the wide array of display terminals available it is not possible to perfectly interface each one with the SDK-80 hardware and software. The user should carefully examine the requirements of his particular terminal interface and adapt the SDK-80 accordingly.

FUNCTIONAL DEFINITION
The SDK-80 is shipped with a complement of parts that allows the user to construct an operating small system with the following features:

CPU: 8080A (see 8080 User's Manual for details)
1.95 µs Instruction Cycle

RAM: 8111 (static 256 x 4) 2 included for 256 byte storage

ROM: 8708/8308 (1K x 8)
1 Pre-programmed system monitor
1 User-programmed (erasable 8708)

I/O: 8251 (Programmable Communication Interface)
1 Serial communication with terminal
8255 (Programmable Peripheral Interface)
1 General user I/O, 24 lines

Serial
Interface: TTL
20mA current loop (TTY)
RS-232 (EIA)

Baud
User-selected by jumper or switch
Rate: 75, 110, 300
600, 1200, 2400, 4800

Interrupt: Single level, vectored (RST-7)

The SDK-80 has many designed-in features for expandability without the necessity of cutting PC runs or adding extra logic. The maximum configuration of the SDK-80 is as follows:

RAM: 8111 (static 256 x 4)
Up to 8 for 1K x 8 storage

ROM: 8708/8308 (1K x 8)
Up to 4 for 4K x 8 storage

I/O: 8255 (Programmable Peripheral Interface)
2 General user I/O, 48 lines

Expanding the SDK-80 to the maximum configuration is a simple matter of purchasing the extra memory and I/O components and installing them on the board.
REGISTRATION CARD

On the back cover of the User’s Guide is the Registration card for the SDK-80. Please fill it out completely and return it to INTEL upon completion of the kit. The Registration Card assures you of being updated with the latest information on the 8080 microcomputer system and any additional updates on your 8080 System Design Kit.

SDK-80 REGISTRATION CARD

[Registration card fields: Name, Title, Company, Address, City, State, Mail Stop, Zip Code, Distributor, Location]

GENERAL INQUIRY

How long did it take to assemble? _______ Hours
Did it work the first time? Yes [ ] No [ ]
If No, please comment: __________________________

General Comments: __________________________

What make/model terminal did you use?

Brand: __________________________

Assembled Board.
CHAPTER 2
HOW TO ASSEMBLE THE KIT

GENERAL
The MCS-80™ System Design Kit is shipped in a single cardboard carton. This chapter will take you from the point of receiving this carton to the point where you are ready to insert the IC (Integrated Circuit) chips.

Follow the instructions carefully and make a check mark in the boxes provided after you have completed each step.

Your work area should be an uncluttered, well-lit table or desk with access to an AC wall socket or extension cord.

TOOLS REQUIRED
Before starting the project, you should make sure the proper tools are at hand and are in good operating condition. These tools will be required to assemble the MCS-80 System Design Kit:

- A pair of needle-nose pliers
- A small Phillips head screwdriver
- A 1/4” standard flat head screwdriver
- A pair of small diagonal cutters
- A 25-watt pencil-type soldering iron
- A spool of rosin core solder with 60:40 tin-lead content.

IMPORTANT
Use only rosin core solder for all electrical soldering!

- A VOM (Volt-Ohm-Milliammeter) test meter
If available, a dual-probe oscilloscope would also be helpful.

OPTIONAL ROTARY SWITCH
The Design Kit is complete for most applications; however, applications requiring multiple baud rates will need an additional rotary switch. We have allocated a position on the circuit board for this switch. One possible switch is Spectrol 87-12-19, available from Spectrol Electronics Corp., 17070 East Gale Avenue, City of Industry, CA 91744. Phone: (213) 964-6565.

PARTS LIST
With the proper tools at hand, you are now ready to take inventory of the carton.

The contents of the carton are divided into two compartments. One compartment contains the kit’s documentation, the other contains the SDK-80. In addition to the User’s Guide that you are now reading, the following documents are included:

- 8080 Assembly Language Programming Manual
- 8080 Assembly Language Reference Card
- 8008/8080 PL/M™ Programming Manual
- MDS Brochure
- 8080 Microcomputer Systems User’s Manual

The components of the MCS-80 System Design Kit come in four packages:

- Printed wiring (PW) board, PN 1000609-01
- Miscellaneous small component bag
- SDK-80 Intel component pack

CAUTION
Do not handle the IC’s until instructed to do so.

- Miscellaneous non-Intel component pack
If any of the above component packages or documents are missing, call your distributor immediately. If not, lay each of the component packages on your work table and proceed reading.
SDK-80 Intel® Component Pack

The Intel component pack contains the Intel IC's needed in the Kit. The numbers indicated in Figure 2-1 correspond to the “Item Number” in Table 2-1.

☐ Verify that all items in Table 2-1 are included. Do not remove components from backing.

Table 2-1. Parts List, Intel Component Pack.

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Part No.</th>
<th>Description</th>
<th>Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52-016</td>
<td>IC, Intel® 8205</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>52-035</td>
<td>IC, Intel® 8251</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>52-045</td>
<td>IC, Intel® 8224</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>52-046</td>
<td>IC, Intel® 8228</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>52-047</td>
<td>IC, Intel® 8255</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>52-059</td>
<td>IC, Intel® 8708</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>52-058</td>
<td>IC, Intel® 8080A</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>52-062</td>
<td>IC, Intel® 8111</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>52-605</td>
<td>IC, Intel® SDK-80 Monitor ROM</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 2-1. Intel® Component Pack.
Non-Intel® Component Pack

The non-Intel component pack contains miscellaneous components produced by other manufacturers. Figure 2-2 shows the arrangement of these components. The circled numbers are keyed to Table 2-2.

- Verify that all items in Table 2-2 are included. Do not remove components from backing.

Table 2-2. Parts List, Non-Intel Component Pack.

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Part No.</th>
<th>Description</th>
<th>Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54-028</td>
<td>IC, 74161</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>54-092</td>
<td>IC, 7406</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>54-135</td>
<td>IC, 79M05AUC</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>54-136</td>
<td>IC, 93S16</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>68-009</td>
<td>Socket, 40-pin</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>68-007</td>
<td>Socket, 24-pin</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>68-177</td>
<td>Socket, 28-pin</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>68-102</td>
<td>Connect female, right angle, or 25-pin</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>68-077</td>
<td>Connect male, 25-pin</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>68-006</td>
<td>Socket, or 16-pin</td>
<td>1</td>
</tr>
</tbody>
</table>

Auxiliary Components For SDK-80 Expansion (Not Supplied With Kit)

<table>
<thead>
<tr>
<th>Part No.</th>
<th>Description</th>
<th>Manufacturer</th>
<th>Manufacturer Part No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>66-032</td>
<td>Right angle push button switch SPDT</td>
<td>C &amp; K</td>
<td>8125R2</td>
</tr>
<tr>
<td>68-102</td>
<td>Right angle female 25 pin connector</td>
<td>ITT Cannon</td>
<td>DBC-255-AA</td>
</tr>
<tr>
<td>68-077</td>
<td>Male 25 pin connector (solder)</td>
<td>Cinch Jones</td>
<td>DB-25P</td>
</tr>
<tr>
<td>66-007</td>
<td>Low profile 18 pin DIP socket</td>
<td>TI</td>
<td>C93-18-02</td>
</tr>
<tr>
<td>66-008</td>
<td>Low profile 24 pin DIP socket</td>
<td>TI</td>
<td>C93-24-02</td>
</tr>
<tr>
<td>66-009</td>
<td>Low profile 40 pin DIP socket</td>
<td>TI</td>
<td>C93-40-02</td>
</tr>
<tr>
<td>66-007</td>
<td>Small 30/60 PCB connector</td>
<td>CDC</td>
<td>97167901 (w-w)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>97169001 (solder)</td>
</tr>
<tr>
<td>66-009</td>
<td>Large 43/86 PCB connector (wire-wrap)</td>
<td>Viking</td>
<td>VPB01EA3A00A-1</td>
</tr>
</tbody>
</table>

Figure 2-2. Non-Intel® Component Pack.
Small Component Bag

The small component bag contains the miscellaneous resistors, capacitors, screws, etc. needed to support the integrated circuits. Proceed as follows:

☐ Open the bag and spread the components in front of you.

☐ Separate the components into groups; i.e., resistors in one group, capacitors in another, screws and nuts and washers in another, etc.

☐ Verify that all parts listed in Table 2-3 are included.

ASSEMBLY PROCEDURE

Now that you have verified that all of the parts are included, you are ready to begin assembling the board.

This section of the text will be supplemented with drawings that illustrate the area of the board under construction. Components to be installed later will be shown as unshaded. Components being installed will be shown as shaded. Components that were installed earlier in the procedure will be shown blackened.

We shall assemble the board components by area, where the board may be thought of as being divided into three major areas. **Area one** will encompass that portion of the board to the left of J1 Pin 1 on the silk screen. **Area two** will encompass that portion of the board that lies below J1 and J2. **Area three** will encompass everything to the right of area two.

<table>
<thead>
<tr>
<th>Part No.</th>
<th>Description</th>
<th>Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>62-008</td>
<td>Crystal, 18.432 MHz</td>
<td>1</td>
</tr>
<tr>
<td>56-044</td>
<td>Resistor, 10 KΩ, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(brown-black-orange)</td>
<td></td>
</tr>
<tr>
<td>56-024</td>
<td>Resistor, 1 KΩ, 1/4W</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>(brown-black-red)</td>
<td></td>
</tr>
<tr>
<td>56-033</td>
<td>Resistor, 2.7 KΩ, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(red-violet-red)</td>
<td></td>
</tr>
<tr>
<td>56-006</td>
<td>Resistor, 47 Ω, 1/4W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(yellow-violet-black)</td>
<td></td>
</tr>
<tr>
<td>56-017</td>
<td>Resistor, 390 Ω, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(orange-white-brown)</td>
<td></td>
</tr>
<tr>
<td>56-106</td>
<td>Resistor, 430 Ω, 1W</td>
<td>1</td>
</tr>
<tr>
<td>56-038</td>
<td>Resistor, 4.7 KΩ, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(yellow-violet-red)</td>
<td></td>
</tr>
<tr>
<td>56-209</td>
<td>Resistor, 560 KΩ, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(green-blue-yellow)</td>
<td></td>
</tr>
<tr>
<td>56-039</td>
<td>Resistor, 5.1 KΩ, 1/4W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(green-brown)</td>
<td></td>
</tr>
<tr>
<td>56-112</td>
<td>Resistor, 150 Ω, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(brown-green-brown)</td>
<td></td>
</tr>
<tr>
<td>56-020</td>
<td>Resistor, 510 Ω, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(green-brown-brown)</td>
<td></td>
</tr>
<tr>
<td>56-210</td>
<td>Resistor, 430 Ω, 1.5W</td>
<td>1</td>
</tr>
<tr>
<td>56-213</td>
<td>Resistor, 130 Ω, 1/4W</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(brown-orange-brown)</td>
<td></td>
</tr>
<tr>
<td>60-003</td>
<td>Diode, 1N914</td>
<td>2</td>
</tr>
<tr>
<td>58-006</td>
<td>Transistor, 2N2222</td>
<td>2</td>
</tr>
<tr>
<td>58-003</td>
<td>Transistor, 2N2907</td>
<td>1</td>
</tr>
<tr>
<td>66-032</td>
<td>Switch, right angle, SPDT</td>
<td>1</td>
</tr>
<tr>
<td>64-042</td>
<td>Capacitor, 1 uf, 50V tant</td>
<td>2</td>
</tr>
<tr>
<td>64-012</td>
<td>Capacitor, 22 uf, 15V tant</td>
<td>3</td>
</tr>
<tr>
<td>64-052</td>
<td>Capacitor, 10 μF, mica</td>
<td>1</td>
</tr>
<tr>
<td>64-050</td>
<td>Capacitor, .1 uf, 50V monolithic (bright-colored)</td>
<td>13</td>
</tr>
<tr>
<td>64-022</td>
<td>Capacitor, .01 uf, ceramic disc (orange)</td>
<td>16</td>
</tr>
<tr>
<td>82-015</td>
<td>Terminal lugs, 2010B</td>
<td>6</td>
</tr>
<tr>
<td>82-072</td>
<td>Spacer, nylon 1/4&quot; x 7/16&quot; o.d.</td>
<td>5</td>
</tr>
<tr>
<td>82-069</td>
<td>Rubber feet</td>
<td>5</td>
</tr>
<tr>
<td>64-010</td>
<td>Screw, 4-40 x 3/8&quot; pan</td>
<td>1</td>
</tr>
<tr>
<td>64-073</td>
<td>Screw, 2-56 x 3/8&quot; pan</td>
<td>4</td>
</tr>
<tr>
<td>64-016</td>
<td>Screw, 6-32 x 3/4&quot; pan</td>
<td>5</td>
</tr>
<tr>
<td>84-069</td>
<td>Nut, 4-40 plain hex</td>
<td>1</td>
</tr>
<tr>
<td>84-042</td>
<td>Nut, 2-56</td>
<td>4</td>
</tr>
<tr>
<td>84-068</td>
<td>Washer, #2 nylon</td>
<td>4</td>
</tr>
<tr>
<td>64-027</td>
<td>Washer, #6 nylon</td>
<td>5</td>
</tr>
<tr>
<td>84-070</td>
<td>Nut, 5-32</td>
<td>5</td>
</tr>
<tr>
<td>84-059</td>
<td>Washer, #4 nylon</td>
<td>2</td>
</tr>
</tbody>
</table>
Assembly of Area One

Proceed as follows:

☐ Lay the PW board on your work area, silkscreened side up, so that the edge connectors are directly in front of you.

☐ Your first step will be to install the five rubber feet. There will be one foot in each corner and one in the middle of the board. At each of these locations, place a nylon spacer and rubber foot underneath the board and insert a screw (5-32x3/4) through them from the bottom. Then attach the screw at the top using a #6 nylon washer and a 6-32 hex nut.

☐ Referring to Table 2-4, solder resistors R1-R4 and R21-R24 in place. Figure 2-3 shows this area of the board.

**HINT**

Save all scrap resistor leads for later use as jumper wires.

### Table 2-4. Construction Table #1.

<table>
<thead>
<tr>
<th>Resistor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>130, 1/4W (brown-orange-brown)</td>
</tr>
<tr>
<td>R2</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R3</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R4</td>
<td>560K, 1/4W (green-blue-yellow)</td>
</tr>
<tr>
<td>R21</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R22</td>
<td>5.1K, 1/4W (green-brown-red)</td>
</tr>
<tr>
<td>R23</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R24</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
</tbody>
</table>

Figure 2-3. Construction Figure #1.
Solder capacitors C1-C3 and C10-C16 in place (Figure 2-4). C1 is a 1 uf, 50V tant. C2, C3, and C11-C14 0.01 uf ceramic. C10 is a 10 pF mica. C15 and C16 are 22 uf, 15V tant.

**IMPORTANT**

Be sure that electrolytic capacitors C1, C15 and C16 are installed so that their “+” ends are positioned as shown on the board.

- Solder diode CR1 in place. Make sure the arrow is pointing as shown on the silk-screen.
- Solder Reset Switch SW1 in place.
- Solder terminal lugs E1-E6 in place. Figure 2-5 shows the installation completed in the last three steps.
- Solder the Spectrol rotary baud rate switch in place, if applicable.
- Set the clock crystal on the board at location Y1 and use a pencil to mark the bend points of each of the leads.
- Use needle-nose pliers to bend each of the leads at a right angle at the points you marked with pencil.
- Insert the leads through the board and solder them on the bottom.
- Strap a piece of scrap resistor wire over the crystal, pushing each end through one of the drilled holes. Solder from the bottom.
- Solder the 16-pin socket into location A8. You have now completed area one. Compare your board with Figure 2-6.

Figure 2-4. Construction Figure #2.
Figure 2-5. Construction Figure #3.

Figure 2-6. Construction Figure #4.
Assembly of Area Two

Board area two will contain the 8080A and its related logic. The assembly procedure is as follows:

- Referring to Table 2-5, solder resistors R5-R17, R19, R20, R25, and R26 in place. Figure 2-7 shows this area of the board.

Note 1. (From Table 2-6)
The resistor to be installed in R13 depends upon what negative voltage level your power supply delivers. For -10V, use a 390Ω, 1/4W resistor (orange-white-brown). For -12V or -15V, use a 510Ω 1/4W resistor (green-brown-brown).

<table>
<thead>
<tr>
<th>Resistor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5</td>
<td>5.1K, 1/4W (green-brown-red)</td>
</tr>
<tr>
<td>R6</td>
<td>4.7K, 1/4W (yellow-violet-red)</td>
</tr>
<tr>
<td>R7</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R8</td>
<td>10K, 1/4W (brown-black-orange)</td>
</tr>
<tr>
<td>R9</td>
<td>2.7K, 1/4W (red-violet-red)</td>
</tr>
<tr>
<td>R10</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R11</td>
<td>47, 1/4W (yellow-violet-black)</td>
</tr>
<tr>
<td>R12</td>
<td>150, 1/4W (brown-green-brown)</td>
</tr>
<tr>
<td>R13</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>R14</td>
<td>47, 1/4W (yellow-violet-black)</td>
</tr>
<tr>
<td>R15</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R16</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R17</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R19</td>
<td>430, 1W (RS-1A)</td>
</tr>
<tr>
<td>R20</td>
<td>430, 1.5W (G-2)</td>
</tr>
<tr>
<td>R25</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
<tr>
<td>R26</td>
<td>1K, 1/4W (brown-black-red)</td>
</tr>
</tbody>
</table>

Figure 2-7. Construction Figure #5.
Solder capacitors C4-C9 and C17-C22 into place. C4, C6-C9, and C17-C20 are 0.01 uf ceramic. C5 is 0.1 uf, 50V mono. C21 is 1 uf, 50V tant. C22 is 22 uf, 15V tant.

**IMPORTANT**
Be sure that electrolytic capacitors C21 and C22 are positioned so that their "+" ends are positioned as shown on the board.

- Solder transistors Q1, Q2, and Q3 into place. Q1 and Q2 are 2N2222. Q3 is a 2N2907.
- Solder diode CR2 into place. Make sure the arrow is as shown on the board. Figure 2-8 shows the progress made in the last three steps.

**IMPORTANT**
The metal tabs on the transistors must be positioned as shown on the board.

- Solder 40-pin sockets into locations A3 and A10.
- Solder 28-pin sockets into locations A7 and A9.
- Insert a 25-pin female connector into location J1. At each of two locations, place a 2-56x3/8 screw through the connector from the top and secure it at the bottom with a #2 nylon washer and a 2-56 nut.
- Solder the 25 connector pins onto the board from the bottom.

**NOTE**
The 25-pin male connector provided can be used to interface your hardware to connector J1.

Area two should now be complete. Compare your board with Figure 2-9 (next page).

Figure 2-8. Construction Figure #6.
Assembly of Area Three

Board **area three** contains the RAM and ROM memory logic. Proceed as follows:

- Solder resistors R27-R30 in place. All are 1K, 1/4W (brown-black-red).
- Solder capacitor C23 in place. It is a 0.01 uf ceramic.
- Solder capacitors C24-C35 in place. All of these capacitors are 0.1 uf, 50V mono. Figure 2-10 shows the progress made in the last three steps.
- Solder 24-pin socket into location A14.

- Set the voltage regulator (79M05AUC) on the board and use a pencil to mark the bend point on each of the three leads.
- Use needle-nose pliers to bend each of the three leads at a right angle at the points you marked with pencil.
- Referring to Figure 2-11, fasten the regulator to the board with a 4-40 x 3/8 screw, 2 #4 nylon washers, and a 4-40 nut.
- Solder the three leads in place on the bottom of the board and clip off any excess lead lengths.
Insert a 25-pin female connector into location J3. At each of two locations, place a 2-56x3/8 screw through the connector from the top and secure it at the bottom with a #2 nylon washer and a 2-56 nut.

Solder the 25 connector pins onto the board from the bottom.

NOTE
The 25-pin male connector provided can be used to interface your hardware to connector J3.

You have now completed area three. Compare your board with Figure 2-12 (next page).

Figure 2-11. Voltage Regulator (VR1) Installation.

Figure 2-10. Construction Figure #8.
Figure 2-12. Construction Figure #9.
CHAPTER 3
THEORY OF OPERATION

GENERAL
Now that you have assembled the structure of the SDK-80 it is time to discuss the internal composition of the design. We will do this by presenting the functional organization of the SDK-80 logic and, in the process, bring in the decisions that you, as the user, must make before completing the kit.
Figure 3-1 is a functional block diagram of the SDK-80. It has been purposely drawn as simple as possible in order to give a basis for discussion. You will note that this figure shows only the major signals in the unit. For this reason, some occasional reference to the SDK-80 schematics (Appendix B) will be in order.
The text to follow describes each of the elements in the block diagram.

Figure 3-1. SDK-80 Functional Block Diagram.
SYSTEM BUSES
The SDK-80 logic is built around three system busses: the data bus, the address bus and the control bus. All of the MCS-80* components communicate via these three busses.

The system busses can be selectively enabled/disabled from the user system if the board is jumpered for that capability. Bus enable jumpering is described in the System Bus Enable section of Chapter 4.

Each bus is more fully described in the 8080 Microcomputer Systems User's Manual.

RESET SWITCH
The Reset Switch gives you the capability of forcing a reset to the SDK-80 logic at any time. When the switch is pressed, the Clock Generator will send a RESET signal throughout the system. The Reset Switch should be pressed each time you power-up the system.

CLOCK GENERATOR AND CLOCK CRYSTAL
The 8224 Clock Generator provides the primary timing to the system. It generates the high-level clocks necessary to drive the 8080A CPU, synchronizes the READY signal into the CPU, and transmits the power-up (and Reset Switch) reset signal.

Ø1 and Ø2 Clocks
Ø1 and Ø2 are 2.048 MHz clocks for the CPU. They are derived from OSC using an internal divide-by-nine function.

RESET Signal
RESET is the primary reset signal to the system logic. It is asserted both at power-up and when the Reset Switch is pressed. RESET clears the CPU, disables the RAM Decoder, and resets the USART. RESET is available to the user system at pad V.

READY Signal
READY can provide a synchronized READY to the CPU, derived from an external asynchronous RDYIN signal (pad P).

OSC Signal
OSC provides an 18.432 MHz input to the Baud Rate Generator. This 18.432 MHz rate was chosen for two reasons. First, it permits the 8080A CPU to run at very close to its maximum speed. Second, it is a convenient rate to use in designing a simple, but highly stable, Baud Rate Generator.

STSTB (Status Strobe) Signal
At the beginning of each machine cycle, the CPU issues status information on its data bus. STSTB causes the 8228 System Controller to store this information into its status latch. STSTB is available to the user system as STATUS STROBE at pad J.

8080A CPU
The 8080A CPU is thoroughly described in the Intel® 8080 Microcomputer Systems User's Manual and need not be repeated here.

The CPU clocks, Ø1 and Ø2, will be supplied (at 2.048 MHz) by the Clock Generator.

The data bus will interface directly to the System Controller and the address bus will enter the system through the Address Buffers, if applicable.

There are two separate jumper-wire options with the CPU. The first option allows an external HOLD signal to be presented to the CPU via pad R. The second option allows an external READY signal to force a Wait state in the CPU. It should be pointed out, however, that the 8080A and SDK-80 memory chips have been designed to operate without Wait states. The option permits you to force a Wait if desired, though. Both of these jumper options are described in the Hold And Wait Options section of Chapter 4.

SYSTEM CONTROLLER
The 8228 System Controller generates the control bus signals that provide read and write functions for I/O and memory.

They are available to the user system as shown below:

- I/O W is at pad E
- I/O R is at pad L
- MEMW is at pad U
- MEMR is at pad T

The System Controller also buffers the data bus.

Interrupt
A single-level interrupt structure is provided such that whenever pad H (INT REQ) is
grounded, the System Controller causes a Restart instruction (RST 7) to be inserted into the CPU. This feature provides a single interrupt vector without using additional components, such as an interrupt instruction port. Multiple level interrupts will require additional chips to be installed into the wire-wrap area.

ADDRESS BUFFERS (OPTIONAL)
The 8212 Address Buffers are not included in the System Design Kit, but must be added if more than a nominal amount of memory (more than 1024 bytes of RAM and more than 4K bytes of ROM) is used. The Address Buffers are tri-state TTL buffers that provide 15mA drive.

The address bus level can be forced to the high-impedance state by inputting a high level on pad S (SYSTEM BUS ENABLE), if the board is jumped for this capability.

SDK-80 MEMORY
The SDK-80 has two types of memory. Its ROM Memory can accommodate from 1K to 4K bytes, where the lower 1K bytes are dedicated to the system monitor. Its RAM Memory can accommodate from 256 to 1K bytes, in which all but the uppermost 30 bytes (addresses 13E2-13FF) are useable by your system. Figure 3-2 is a map of SDK-80 memory.

![SDK-80 Memory Map](image)

The ROM that installs into board location A14 has been pre-programmed with the SDK-80 system monitor.

The 8708/8308 that installs into board locations A15, A16, and A17 can be used to hold a program that you have developed and checked out in RAM.

The 8205 ROM Decoder selects the ROM chip being addressed. Figure 3-3 shows the ROM address format.

![ROM Address Format](image)

RAM Decoder and RAM Memory
In the standard configuration, the SDK-80 can accommodate up to eight 256x4 Static MOS Random Access Memory (RAM) chips. Two of these chips are supplied in the System Design Kit, so users requiring only 256 bytes of memory need not install additional RAM chips.

The 8205 RAM Decoder selects the RAM chip pair being addressed. Figure 3-4 shows the RAM address format.

![RAM Address Format](image)

RAM access is disabled whenever the RESET signal from the Clock Generator is asserted.
**BAUD RATE GENERATOR**

The Baud Rate Generator circuit supplies the transmitter and receiver clocks to the I/O Communication Interface. This circuit is made up of three IC chips: one 93S16 and two 74161s. The Baud Rate Generator takes the 18.432 MHz OSC signal from the Clock Generator and, by internal division, generates a series of signals which represent baud rates between 75 and 4800. The baud rate that will be presented to the I/O Communication Interface is determined by jumper-wiring or a rotary switch. This selection will be discussed in the Baud Rate Selection section of Chapter 4.

**I/O COMMUNICATION INTERFACE**

The 8251 I/O Communication Interface is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) chip that accommodates any data communications required by the SDK-80 system. The I/O Communication Interface can accept parallel data from the data bus and send it serially to an external device. It can also accept serial data from an external device and put it onto the data bus in parallel form when eight bits have been collected. Figure 3-5 shows the address format for communications.

The baud rate at which the I/O Communication Interface will transmit and receive data is governed by the Baud Rate Generator.

The I/O Communication Interface circuit on the board also includes some jumpers that select the communication input/output level. Any of three levels may be selected:

- RS-232 level, which is typically used for CRT applications
- Current-loop level, for TTY applications
- TTL level.

The input/output level jumpering is discussed in the Communication Level Selection section of Chapter 4.

**PERIPHERAL INTERFACES**

The 8255 Programmable Peripheral Interfaces provide the user's primary access point to the SDK-80 data bus. One 8255 chip is supplied in the System Design Kit.

![Figure 3-5. I/O Communication Interface Address Format.](image)

Each Peripheral Interface chip provides three 8-bit parallel I/O ports, each of which is independently addressable. Figure 3-6 shows the address format for I/O port selection.

![Figure 3-6. Peripheral Interface Address Format.](image)

The output pins of the Peripheral Interfaces are totally uncommitted and may be jumper-wired to best suit your particular application. For example, they might be wired directly to the interface plugs or, alternately, they might be wired to standard TTL buffers in the wire-wrap area before coming back to the plugs. This wiring is further discussed in the Output Wiring section of Chapter 4.
CHAPTER 4
FINAL ASSEMBLY AND CHECKOUT

GENERAL
At this point in the manual you should have completed the preliminary assembly and read the theory of operation. You can now finish the board assembly and begin a checkout sequence.

JUMPER-WIRING THE BOARD
The SDK-80 is designed to be used in virtually any evaluation application and can be jumpered to suit your particular requirements. These questions will help you decide what jumpers are needed:

1. Will you ever want the CPU to enter a Hold or Wait state?
2. Will you ever want to disable the system busses?
3. What type of input device will you use to communicate with the SDK-80 (e.g., CRT, Teletype)?
4. What is its baud rate?
5. Will you be using 8212 Address Buffer chips?
6. What kind of information will be transferred to/from the SDK-80?

If you have a fairly good idea of the answers to all of these questions, you are ready to start jumper-wiring the board. The scrap leads that have been cut from previously-installed resistors are a good source of jumper wire. However, use 22-gauge insulated wire in situations where any jumpers may make contact with each other.

Hold and Wait Options
The SDK-80 is designed to run without Hold or Wait states. However, a jumper-wire option is available to give either capability.

☐ To disable the Hold state, wire J5-2 to J5-3.
☐ To enable the Hold state, wire J5-1 to J5-2.
☐ If READY is to force an 8080 Wait state, wire J5-8 to J5-9. If not, wire J5-8 to J5-7.

System Bus Enable
One jumper is available to make it possible to selectively disable the SDK-80 system bus.

☐ If the bus will be selectively disabled, wire J5-5 to J5-6.
☐ If the bus should remain enabled at all times, wire J5-4 to J5-5.

Baud Rate Selection
The communications baud rate can be selected in two ways, depending on the application. If only one baud rate will be employed, the rate can be selected by installing a single jumper wire. If two or more baud rates will be employed in the application, however, the Spectrol rotary switch installed in Chapter 1 will be used for this purpose.

☐ To select a fixed baud rate, jumper pad 29 to one of the pads 31-37 per Table 4-1.

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Wire Pad 29 To</th>
</tr>
</thead>
<tbody>
<tr>
<td>4800</td>
<td>31</td>
</tr>
<tr>
<td>2400</td>
<td>32</td>
</tr>
<tr>
<td>1200</td>
<td>33</td>
</tr>
<tr>
<td>600</td>
<td>34</td>
</tr>
<tr>
<td>300</td>
<td>35</td>
</tr>
<tr>
<td>150</td>
<td>36</td>
</tr>
<tr>
<td>75 or 110</td>
<td>37</td>
</tr>
</tbody>
</table>

☐ For 110 baud, the standard Teletype rate, wire pad 4 to pad 5.

Communication Level Selection
Any of three communication levels can be selected: CRT, Teletype, or TTL. All serial data is passed through connector J3.
**Table 4-2. Communication Level Jumper Table.**

<table>
<thead>
<tr>
<th>CRT Configuration Jumper Position</th>
<th>TTY Configuration Jumper Position</th>
<th>TTL Configuration Jumper Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>23 to 24</td>
<td>23 to 26</td>
<td>23 to 25</td>
</tr>
<tr>
<td>17 to 18</td>
<td>18 to 19</td>
<td>17 to 18</td>
</tr>
<tr>
<td>9 to 10</td>
<td>10 to 11</td>
<td>12 to 13</td>
</tr>
<tr>
<td>13 to 14</td>
<td>13 to 14</td>
<td>2 to 3</td>
</tr>
<tr>
<td>2 to 3</td>
<td>1 to 2</td>
<td>20 to 21</td>
</tr>
<tr>
<td>6 to 8</td>
<td>7 to 8</td>
<td>8</td>
</tr>
<tr>
<td>27 to 28</td>
<td>15 to 16</td>
<td>21 to 22</td>
</tr>
</tbody>
</table>

- Jumper wire pads 1 through 28 per Table 4-2.
- If your system does not contain a modem, jumper pad A to pad B.

**Address Bus Jumpers**

If you do not use 8212 Address Buffer chips on your SDK-80, the address bus must be jumpered across locations A11 and A12. In this situation, connect the following jumpers AT BOTH LOCATION A11 AND LOCATION A12. All jumpers should be installed from the circuit side of the board i.e., NOT the silk-screen side.

- Jumper pad 3 to pad 4.
- Jumper pad 5 to pad 6.
- Jumper pad 7 to pad 8.
- Jumper pad 9 to pad 10.
- Jumper pad 15 to pad 16.
- Jumper pad 17 to pad 18.
- Jumper pad 19 to pad 20.
- Jumper pad 21 to pad 22.

**Output Wiring**

Connector J3 is dedicated as a communications interface (see Table 4-3) and is, in fact, the only committed interface in the SDK-80. All other interfacing is at the discretion of the user.

For example, the 8255 Peripheral Interface might be jumpered directly to connector J1 or, alternately, might be jumpered to TTL buffers in the wire-wrap area before being passed to J1. Conversely, you might wish to add a switch array to the 8255 area in order to send data to the CPU.

Your System Design Kit includes male connectors that mate with the female connectors installed at J1 and J3.

A group of control signals are available at the alphabetic-labeled pads in area two of the board. Table 4-4 identifies these pads.

**Table 4-3. Pin Assignments for Communications Interface (J3).**

<table>
<thead>
<tr>
<th>J3 Pin</th>
<th>CRT Configuration</th>
<th>TTY Configuration</th>
<th>TTL Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>TTL REC.</td>
</tr>
<tr>
<td>2</td>
<td>CRT REC. DATA</td>
<td></td>
<td>TTL REC. DATA</td>
</tr>
<tr>
<td>3</td>
<td>CRT XMIT DATA</td>
<td></td>
<td>TTL XMIT DATA</td>
</tr>
<tr>
<td>4</td>
<td>+12 VDC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SIGNAL GND</td>
<td></td>
<td>SIGNAL GND</td>
</tr>
<tr>
<td>6</td>
<td>+12 VDC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>TTY REC (w)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>TTY XMIT (w)</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4-4. SDK-80 Control Bus Pads.**

<table>
<thead>
<tr>
<th>Pad</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>CTS</td>
<td>Clear To Send</td>
</tr>
<tr>
<td>B</td>
<td>RTS</td>
<td>Request To Send</td>
</tr>
<tr>
<td>C</td>
<td>02 (TTL)</td>
<td>2.048 MHz Clock</td>
</tr>
<tr>
<td>D</td>
<td>OSC</td>
<td>18.432 MHz Oscillator</td>
</tr>
<tr>
<td>E</td>
<td>I/O W</td>
<td>I/O Write</td>
</tr>
<tr>
<td>F</td>
<td>DTR</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>H</td>
<td>INT REQ</td>
<td>Interrupt Request</td>
</tr>
<tr>
<td>J</td>
<td>STATUS STROBE</td>
<td>Status is on Data Bus</td>
</tr>
<tr>
<td>K</td>
<td>I/O R</td>
<td>I/O Read</td>
</tr>
<tr>
<td>M</td>
<td>HLDA</td>
<td>Hold Acknowledge</td>
</tr>
<tr>
<td>N</td>
<td>INTA</td>
<td>Interrupt Acknowledge</td>
</tr>
<tr>
<td>P</td>
<td>READY</td>
<td>Ready</td>
</tr>
<tr>
<td>R</td>
<td>HOLD</td>
<td>Hold</td>
</tr>
<tr>
<td>S</td>
<td>SYSTEM BUS ENABLE</td>
<td>Enables Data Bus and Address Bus</td>
</tr>
<tr>
<td>T</td>
<td>MEMR</td>
<td>Memory Read</td>
</tr>
<tr>
<td>U</td>
<td>MEMW</td>
<td>Memory Write</td>
</tr>
<tr>
<td>V</td>
<td>RESET</td>
<td>Reset</td>
</tr>
</tbody>
</table>
INSTALLING INTEGRATED CIRCUITS

You have now reached the point where you will start installing IC's in the board, but a few words are in order before you begin.

Special Precautions For Handling MOS IC's

The Kit's MOS IC's (8080, 8111, 8251, 8255, and 8708) are particularly susceptible to static electricity. They can be easily damaged if proper care is not taken in handling them. For this reason, the following steps should be adhered to as closely as possible:

1. All equipment (soldering iron, tools, solder, etc.) should be at the same potential as the PW board, the assembler, the work surface and the IC itself along with its container. This can be accomplished by continuous physical contact with the work surface, the components, and everything else involved in the operation.

2. When handling the IC, develop the habit of first touching the conductive container in which it is stored before touching the IC itself.

3. Always touch the SDK-80's PW board before touching the IC to the board. Try to maintain this contact as much as possible while installing the IC.

4. Handle the IC by the edges. Avoid touching the pins as much as possible.

5. In general, never touch anything to the IC that you have not touched first while touching both it and the IC itself.

Aligning the IC Pins

The connector pins of Integrated Circuit chips are very fragile and can be easily pushed out of line. In fact, sometimes IC's will arrive with one or more pins out of line. Trying to install a misaligned IC is a hapless task and, worse, might cause permanent damage to the chip.

Aligning the pins of an IC is an easy job. Simply lay the IC on its side on your work surface, hold the chip by its body and exert enough pressure so that all pins are perpendicular to the body.

Chip Orientation

The IC's must be correctly oriented on the board or they will not operate properly. One end of the chip will carry some sort of identifying mark, typically a notch or a dot or a +

sign. The chip must be installed so that this identifier corresponds to the silkscreened "1" on the board.

Installing IC Chips

After orienting the IC, follow these steps to install it in the board:

1. Start the pins on one side of the IC into their respective holes on the silk-screened side of the PW board. DO NOT PUSH THE PINS IN ALL THE WAY. If you have difficulty getting the pins into the holes, use the tip of a small screwdriver to guide them.

2. Start the pins on the other side of the IC into their holes in the same manner. When all of the pins have been started, set the IC in place by gently rocking it back and forth until it rests as close as possible to the board or socket.

3. If the IC is not installed in a socket, turn the board over and solder each pin to the foil pattern on the back side of the board. Be sure to solder each pin and be careful not to leave any solder bridges.

Removing IC Chips

If required, an IC chip can be removed from a socket by gently rocking it back and forth to start its release. When a gap exists between the chip and socket, pry it gently at alternate ends until the pins start to come loose. A popsicle stick or small screwdriver works well here. Then hold the chip by the ends and pull it free. Try to keep the chip fairly parallel to the socket throughout this operation.

Clock Generator

Besides the 8080, the most critical chip in the SDK-80 circuit is the 8224 Clock Generator.

- Insert the 8224 Clock Chip into the socket at location A8.

Power, Clock and Reset Verification

With this single chip installed, we can check the power and clock inputs and the operation of the Reset Switch. The procedure is as follows:

- Connect your power supply to terminal lugs E1-E6 on the SDK-80 board.
NOTE
The SDK-80 edge connector is power-compatible with Intel's MDS (Microcomputer Development System). If you have an MDS, the SDK-80 can derive its power through installation in the MDS chassis.

- Turn power on.
- Using a voltmeter, verify +5 VDC at the pad provided.
- Verify +12 VDC at the "+12" pad.
- Verify your supply's negative voltage at the "-10" pad.
- Verify -5 VDC at the "-5" pad, near location A17.
- Press the Reset Switch a few times and check for +4 VDC at A8, pin 1 (RESET).

NOTE
Develop the habit of pressing the Reset Switch each time you power-up the system.

- If you have an oscilloscope, verify that A8 pins 10 and 11 each show 2.048 MHz clocks (02 and 01, respectively).
- Using an oscilloscope, verify that A8 pin 12 shows an 18.432 MHz clock (OSC).
- Turn the power off.

Remainder of SDK-80 ICs
After having verified that the SDK-80 logic is correctly receiving power, the system clocks and the RESET signal, you can finish installing the chip complement. Some of the IC's will plug into sockets, others will have to be soldered onto the board.

The procedure is as follows:
- Solder the 93S16 chip into location A1.
- Solder a 74161 chip into locations A2 and A5.
- Solder the 7406 chip into location A6.
- If applicable, solder 8212 chips into locations A11 and A12.
- Solder 8205 chips into locations A13 and A18.
- Solder 8111 chips into locations A25 and A26.

Table 4-5. Power Requirements.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Voltage</th>
<th>Minimum System</th>
<th>Maximum System</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_CC</td>
<td>+5V ±5%</td>
<td>1.3</td>
<td>2.1</td>
<td>Amps</td>
</tr>
<tr>
<td>V_DD</td>
<td>+12V ±5%</td>
<td>.35</td>
<td>.45</td>
<td>Amps</td>
</tr>
<tr>
<td>V_BB</td>
<td>-10V ±5%</td>
<td>.20</td>
<td>.30</td>
<td>Amps</td>
</tr>
</tbody>
</table>
START-UP PROCEDURE
You have now completed the SDK-80 assembly and are ready to start up the system. The start-up procedure is as follows:

- Plug your system communication monitor (CRT, Teletype, etc.) into the SDK-80 connector J3.
- Turn power on at both the SDK-80 power supply and your communication monitor.
- Press the Reset Switch.

At this point, your monitor will display the following message:

MCS-80™ KIT

Congratulations! You are now ready to start using the system.

TROUBLESHOOTING HINTS
If the SDK-80 system does not work properly, turn the power off and investigate these areas:

1. Verify that all resistors have been properly installed and are correctly color-coded. Appendix C summarizes the component values.
2. Verify that all capacitors have been properly installed and that all electrolytic capacitors are installed with proper polarity.
3. Verify that both diodes (CR1 and CR2) have been installed with proper polarity.
4. Verify that the metal tabs of all three transistors are properly positioned.
5. Verify that all IC's are installed with their "1"-end identifiers correctly oriented.
6. Verify that all jumpers have been properly installed.
7. If the above hints do not fix the problem contact the distributor where the SDK-80 was purchased.

Assembled Board (Without Jumpers).
CHAPTER 5  SDK-80 MONITOR

INTRODUCTION
The SDK-80 Monitor is an Intel® 8080 program provided in a pre-programmed ROM. The Monitor accepts and acts upon user commands to operate the SDK-80. It also provides input and output facilities in the form of I/O drivers for user console devices. The Monitor provides the following facilities:

- Displaying selected areas of memory.
- Initiating execution of user programs.
- Setting “BREAK POINTS” in user programs.
- Modifying contents of memory and processor registers.
- Inputting hexadecimal data from the console device to memory.

The Monitor communicates with the user through an interactive console device, normally a Teletype or CRT Terminal. The dialogue between the operator and Monitor consists of user-originated commands in the Monitor’s command language, and Monitor responses, either in the form of a printed message or an action being performed. After the cold start procedure (described under the heading, “Cold Start Procedures” in Section III), the Monitor begins the dialogue by typing the sign-on message on the console and then requests a command by presenting 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. The valid range of numbers is from 1 to 4 hex digits. Longer numbers may be entered, but such numbers will be evaluated modulo $2^{16}$ so that they will fall into the range specified above.

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.

MONITOR OPERATIONS
The SDK-80 Monitor is a command controlled operations supervisor for the 8080 Microcomputer System Design Kit. Control commands are discussed in Section II, “Command Structure”.

I. FUNCTIONAL SPECIFICATION

A. General Characteristics and Scope of Product
The monitor is a program written in Intel® 8080 macro assembly language. The monitor resides in 1K ($K = 1024$ bytes) of programmed ROM and is located in the address space of the 8080 microcomputer between 0 and 1K. The non-volatile nature of the program’s storage medium means that the monitor is available for use immediately after power-on or reset.

B. Description of All Major Functions Performed

1. CONSOLE COMMANDS
The monitor communicates with the operator via an interactive console, normally a teletypewriter. The dialogue between the operator and the monitor consists of commands in the monitor’s command language and the monitor’s responses. After the cold start procedure, the monitor begins the dialogue by typing a sign-on message on the console and then requests a command by presenting 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. The valid range of numbers is from 1 to 4 hex digits. Longer numbers may be entered, but such numbers will be evaluated modulo $2^{16}$ so that they will fall into the range specified above.

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.

2. USE OF THE MONITOR FOR PROGRAMMING AND CHECKOUT
The monitor allows the user to enter, check out, and execute small demonstration programs. The monitor contains facilities for memory modification, 8080 CPU register display and modification, program loading from the console device, program initiation, and the recognition of an “RST 7” instruction as an unconditional branch to RAM address 13FDH. By inserting RST 7 instructions in a program under test, or by using the hardware generated RST 7 instruction (if available), the user can cause execution of a program to transfer to a dedicated location, for whatever purposes he desires.

When the user wishes to re-enter the
monitor, he should use an RST 1 instruction, either generated by hardware or coded into his program. When entered in this manner, the monitor will automatically save the state of the 8080: specifically, it will save all registers (A, B, C, D, E, H, L), the CPU flags (F), the user's Program Counter (PC), and the user's Stack Pointer (SP). These may be examined with the X command. When the operator enters a G command, these values will be restored.

3. I/O SYSTEM

The I/O system provides two routines, console character input and console character output, which the user may call upon to read and write, respectively, characters from and to the console device.

C. Applicable Standards

Throughout this specification, the numbering convention for bits in a word is that bit 0 is the least significant, or rightmost bit.

The internal code set used by the monitor is 7 bit (no parity) ASCII.

II. INTERFACE SPECIFICATIONS

A. 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 in Section IV.C.

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, the user can abort any command, before he enters 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, will be interpreted as a null parameter. Null parameters are illegal in all commands except the "X" command (see below).

Items enclosed in square brackets "[" and "]" are optional. The consequences of including or omitting them are discussed in the text.

1. DISPLAY MEMORY COMMAND, D

D <low address>, <high address>

Selected areas of addressable memory may be accessed and displayed by the D command. The D command produces a formatted listing of the memory area between <low address> and <high address>, inclusive, on the console device. 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.

The D command may be aborted during execution by typing an Escape (ESC) on the console. The command will be terminated immediately, and a new prompt issued.

Example

D9,2A
0000 00 11 22 33 44 55 66
0010 77 88 99 AA BB CC DD EE FF 10 20 30 40 50 60 70
0020 80 90 A0 B0 C0 D0 E0 F0 01 02 03

2. PROGRAM EXECUTE COMMAND, G

G[<entry point>]

Control of the CPU is transferred from the monitor to the user program by means of the program execute command, G. The <entry point> should be an address in RAM which contains an instruction in the user's 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.

Example

G1400

Control is passed to location 1400H.

3. INSERT INSTRUCTIONS INTO RAM, I

I <address>

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 section IV.C.1). The character ESC or ALTMODE (which is echoed to the console as "S") terminates the digit string. If an odd number of hex digits have been entered, a 0 will be appended to the string.

Example

II410
112233445566778899$

This command puts the following pattern into RAM:

1410 11 22 33 44 55 66 77 88 99
II440
123456789$

This command puts the following pattern into RAM:

1440 12 34 56 78 90

Note that, since an odd number of hexadecimal digits were entered initially, a 0 was appended to the digit string.

4. MOVE MEMORY COMMAND, M

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

The M command moves the contents of memory <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 1410 contains 1AH, the command M1410, 141F, 1411 will result in locations 1410 to 1420 containing "1A1A1A..."

The monitor will continue to move data until the source field is exhausted, or until it reaches address 0FFFFH. If the monitor reaches address 0FFFFH without exhausting the source field, it will move data into this location, then stop.

Example

M1410, 150F, 1510

256 bytes of memory are moved from 1410-150F to 1510-160F by this command.

5. SUBSTITUTE MEMORY COMMAND, S

S <address>

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

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

ii. The contents of the location is displayed, followed by a dash (-).

iii. 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.

iv. If a space or comma was typed in step (iii), the next memory location will be displayed as in step (ii). If a carriage return was typed, the S command will be terminated.

Example

S1450 AA BB CC 01-13 23-24

Location 1450, which contains AA is unchanged, but location 1451 (which used to contain BB) now contains CC, 1452 (which used to contain 01) now contains 13, and 1453 (which used to contain 23) now contains 24.

6. EXAMINE AND MODIFY CPU REGISTERS COMMAND, X

X [<register identifier>]

Display and modification of the CPU registers is accomplished via the X command. The X command uses <register identifier> to select the particular register to be displayed. A register identifier is a single alphabetic character denoting a register, defined as follows:
A — 8080 CPU register A
B — 8080 CPU register B
C — 8080 CPU register C
D — 8080 CPU register D
E — 8080 CPU register E
F — 8080 CPU flags byte, displayed in the form as it is stored by the “PUSH PSW” (hex code F5) instruction
H — 8080 CPU register H
L — 8080 CPU register L
M — 8080 CPU registers H and L combined
P — 8080 Program Counter
S — 8080 Stack Pointer

The command operates as follows:

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

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

iii. 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.

iv. If a space or comma was typed in step (iii), the next register in sequence (alphabetical order) will be displayed as in step ii (unless S was just displayed in which case the command is terminated). If a carriage return was entered in step iii, the X command is terminated.

v. If a carriage return was typed in step (i) above, an annotated list of all registers and their contents are displayed.

Example
XA AA BB CC DD EE FF 12 34 1234 0000  
XA AA 23 CC 01 EE FF 12 34 1234 1010  
X
A AA B 23 C CC D 01 E EE F FF H 12 L 34 M 1234 P 01CF S 03CD

B. Console Device Drivers

The monitor interfaces to the console device via a universal synchronous/asynchronous receiver/transmitter (USART). The monitor drivers interface with the USART according to the USART specifications. At the time of the assembly of the kit, the USART may be configured for a particular type of console interface. The actual console device must conform to this interface.

C. Using the I/O System

The user may access the two monitor I/O system routines from his program by calling the routine desired. The following paragraphs describe the routines available and their respective functions.

CI — Console Input
This routine returns a character received from the console device to the caller in the A-register. The A register and the CPU condition codes are affected by this operation. The entry point of this routine is 3FDH.

Example

    CI EQU 3FDH
    ...
    CALL CI
    STA DATA
    ...

CO — Console Output
This routine transmits a character, passed from the caller in the C-register, to the console device. The A and C registers, and the CPU condition codes, are affected by this operation. The entry point of this routine is 3FAH.

Example

    CO EQU 3FAH
    ...
    MVI C, ""
    CALL CO

III. OPERATING SPECIFICATIONS

A. Product Activation Instructions

1. COLD START PROCEDURE

   After a power-on or reset, the monitor will begin execution at location 0 in ROM. The monitor will perform an initialization sequence, and then display a sign-on message on the console. When the monitor is ready for a command, it will prompt with a period, ".".

2. USE OF RAM STORAGE IN THE MONITOR

   The monitor dynamically assigns its RAM stack near the top of the first 1K bytes of RAM (address space from 4K to 5K). The top 3 bytes in this block of RAM are reserved for a transfer address, supplied
by the user, which is used as a destination location for RST 7 instructions (or the optional hardwired instruction). Several additional bytes are used, below the stack, for temporary storage. Except for RAM addresses 5K-1 to 5K-256, all other RAM is available for the user.

3. BREAK POINT FACILITY
The monitor treats the RST 1 instruction (CF hex) 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-80" "KIT" on the console device. The user can at that time display the contents of the CPU status by initiating an "X" command. After examining the machine status and making changes if necessary the user can resume execution of his program by simply inputting "G" and Carriage Return on the console device. By using the RST 1 break point facilities of the monitor the user can step through large portions of his program by inserting RST 1 instructions at key locations. This technique can significantly reduce the amount of time it takes to debug software.

4. INTERRUPT PROCESSING
The SDK-80 hardware is designed so that an external device can interrupt the CPU and execute an automatic RST 7 instruction. The monitor, upon execution of a RST 7 instruction, automatically executes an unconditional JUMP to RAM location (13FDH). This facility allows the user to initiate his program upon command of a peripheral device, such as a switch closure, without the activation of the monitor's program control command "G". At any time during the execution of the interrupt-invoked program, the user may re-enter the monitor by executing a RST 1 instruction. The sign-on message "MCS-80 KIT" will be displayed on the console device and all monitor commands are available to the user. To resume the user program, simply input "G" and Carriage Return on the console device.

B. Error Conditions

1. INVALID CHARACTERS
The monitor checks the validity of each character as it is entered from the console. As soon as the monitor determines that the last character entered is illegal in its context, the monitor aborts the command and issues an "*" to indicate the error.

Example
DI400, 1456*

The character G was encountered in a parameter list where only hexadecimal digits and delimiters are valid.
Y*

Y is not a valid command.

2. 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 the operator specifies a part of memory in a command which does not exist in his particular configuration. 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.
APPENDIX A. MONITOR LISTING

8080 MACRO ASSEMBLER, VER 2.3 ERRORS = 0 PAGE 1

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

PROGRAM: 8080A BOARD MONITOR

COPYRIGHT (C) 1975
INTEL CORPORATION
3865 BOWERS AVENUE
SANTA CLARA, CALIFORNIA 95051

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

ABSTRACT

********

THIS PROGRAM RUNS ON THE 8080A BOARD AND IS DESIGNED TO PROVIDE
THE USER WITH A MINIMAL MONITOR. BY USING THIS PROGRAM,
THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD
A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS
ALREADY IN MEMORY. THE MONITOR ALSO PROVIDES THE USER WITH
ROUTINES FOR PERFORMING CONSOLE I/O.

PROGRAM ORGANIZATION

********

THE LISTING IS ORGANIZED IN THE FOLLOWING WAY. FIRST THE COMMAND
RECOGNIZER, WHICH IS THE HIGHEST LEVEL ROUTINE IN THE PROGRAM.
NEXT THE ROUTINES TO IMPLEMENT THE VARIOUS COMMANDS. FINALLY,
THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK. WITHIN
EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL
ORDER, BY ENTRY POINT OF THE ROUTINE.

THIS PROGRAM EXPECTS TO RUN IN THE FIRST 1K OF ADDRESS SPACE.
IF, FOR SOME REASON, THE PROGRAM IS 80-ORG'BD, CARE SHOULD
BE TAKEN TO MAKE SURE THAT THE TRANSFER INSTRUCTIONS FOR RST 1
AND RST 7 ARE ADJUSTED APPROPRIATELY.

THE PROGRAM ALSO EXPECTS THAT RAM LOCATIONS 5K-1 TO 5K-256,
INCLUSIVE, ARE RESERVED FOR THE PROGRAM'S OWN USE. THESE
LOCATIONS MAY BE ALTERED, HOWEVER, BY CHANGING THE EQU'ED
SYMBOL "DATA" AS DESIRED.

LIST OF FUNCTIONS

****

GETCM

DCMD

GCMD

********

29
**8080 MACRO ASSEMBLER, VER 2.3 ERRORS = 0 PAGE 2**

```assembly
; ICMD
; MCMD
; SCMD
; XCMD
;
; BREAK
; CI
; CNVBN
; CO
; CROUT
; ECHO
; ERROR
; FRET
; GETCH
; GETCHX
; GETNM
; HILO
; NMOUL
; PRVAL
; REGDS
; RGDAD
; RSTTP
; SRET
; STHF
; STILF
; VALDG
; VALDL
;
```

```assembly
ORG 0H

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

MONITOR EQUATES

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

```

<table>
<thead>
<tr>
<th>DEC</th>
<th>HEX</th>
<th>ASCII</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>001B</td>
<td>1BH</td>
<td></td>
<td>CODE FOR BREAK CHARACTER (ESCAPE)</td>
</tr>
<tr>
<td>13FD</td>
<td>13FH</td>
<td></td>
<td>LOCATION OF USER BRANCH INSTRUCTION IN RAM</td>
</tr>
<tr>
<td>03FA</td>
<td>3FAH</td>
<td></td>
<td>LOCATION OF START OF BRANCH TABLE IN ROM</td>
</tr>
<tr>
<td>0027</td>
<td>027H</td>
<td></td>
<td>COMMAND INSTRUCTION FOR USART INITIALIZATION</td>
</tr>
<tr>
<td>00FB</td>
<td>0FBH</td>
<td></td>
<td>CONSOLE (USART) CONTROL PORT</td>
</tr>
<tr>
<td>00FA</td>
<td>0FAH</td>
<td></td>
<td>CONSOLE INPUT PORT</td>
</tr>
<tr>
<td>002A</td>
<td>02AH</td>
<td></td>
<td>CONSOLE OUTPUT PORT</td>
</tr>
<tr>
<td>00FB</td>
<td>0FBH</td>
<td></td>
<td>CONSOLE STATUS INPUT PORT</td>
</tr>
<tr>
<td>008D</td>
<td>0DH</td>
<td></td>
<td>CODE FOR CARRIAGE RETURN</td>
</tr>
<tr>
<td>1300</td>
<td>5*1024 - 256</td>
<td></td>
<td>START OF MONITOR RAM USAGE</td>
</tr>
</tbody>
</table>

```
8080 MACRO ASSEMBLER, VER 2.3 ERRORS = 0 PAGE 3

001B ESC EQU 1BH ; CODE FOR ESCAPE CHARACTER
000F HCHAR EQU 0FH ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
00FF INVRT EQU 0FFH ; MASK TO INVERT HALF BYTE FLAG
000A LF EQU 0AH ; CODE FOR LINE FEED
0009 LOWER EQU 0 ; DENOTES LOWER HALF OF BYTE IN ICMD
00CF LSIGN EQU ----- ; LENGTH OF SIGNON MESSAGE - DEFINED LATER
000F MODE EQU 0CFH ; MODE SET FOR USHRT INITIALIZATION
; MTHAK EQU ----- ; START OF MONITOR STACK - DEFINED LATER
; NCMDS EQU ----- ; NUMBER OF VALID COMMANDS
000F NEWLN EQU 0FH ; MASK FOR CHECKING MEMORY ADDR DISPLAY
13ED PRT70 EQU 07FH ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
0002 REGS EQU DATA+255-10 ; START OF REGISTER SAVE AREA
0038 RSTU EQU 38H ; TRANSFER LOCATION FOR RST 7 INSTRUCTION
; RTABS EQU ----- ; SIZE OF ENTRY IN RTAB TABLE
001B TERM EQU 1BH ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
0001 TRDY EQU 1 ; MASK TO TEST TRANSMITTER STATUS
00FF UPPER EQU 0FFH ; DENOTES UPPER HALF OF BYTE IN ICMD

; -------------------------------
; MONITOR MACROS
; -------------------------------
; TRUE MACRO WHERE ; BRANCH IF FUNCTION RETURNS TRUE (SUCCESS)
; JC WHERE ENDM
; FALSE MACRO WHERE ; BRANCH IF FUNCTION RETURNS FALSE (FAILURE)
; JNC WHERE ENDM
; -------------------------------
; USHRT INITIALIZATION CODE
; -------------------------------

THE USHRT IS ASSUMED TO COME UP IN THE RESET POSITION (THIS
FUNCTION IS TAKEN CARE OF BY THE HARDWARE). THE USHRT WILL
BE INITIALIZED IN THE SAME WAY FOR EITHER A TTY OR CRT
INTERFACE. THE FOLLOWING PARAMETERS ARE USED:
MODE INSTRUCTION
====
2 STOP BITS
PARITY DISABLED
8 BIT CHARACTERS
BAUD RATE FACTOR OF 64

COMMAND INSTRUCTION
====
NO HUNT MODE
NOT (RTS) FORCED TO 0
RECEIVE ENABLED
DATA TERMINAL READY
TRANSMIT ENABLED

MVI A,MODE
MVI A,CMO
OUT CNCTL ; OUTPUT MODE SET TO USART
OUT CNCTL ; OUTPUT COMMAND WORD TO USART

RESTART ENTRY POINT

SHLD LSAVE ; SAVE HL REGISTERS
POP H ; GET TOP OF STACK ENTRY
SHLD PSAVE ; ASSUME THIS IS LAST P COUNTER
LI H,0 ; CLEAR HL
DAD SP ; GET STACK POINTER VALUE
SHLD SS Save ; SAVE USER'S STACK POINTER
LXI H,ASAVE+1 ; NEW VALUE FOR STACK POINTER
SPHL ; SET MONITOR STACK POINTER FOR REG SAVE
PUSH PSW ; SAVE A AND FLAGS
PUSH B ; SAVE B AND C
PUSH D ; SAVE D AND E

PRINT SIGNON MESSAGE
8080 MACRO ASSEMBLER, VER 2.3 ERRORS = 0 PAGE 5

;*****************************************************************************;
; 001D 219D03  LXI   H,SGN0N ; GET ADDRESS OF SIGNON MESSAGE
; 0020 060E  MVI   B,LSGN0N ; COUNTER FOR CHARACTERS IN MESSAGE
; 0022  4E    MOV   C,N ; FETCH NEXT CHAR TO C REG
; 0023  CDB301 CALL  CO ; SEND IT TO THE CONSOLE
; 0026  23    INX   H ; POINT TO NEXT CHARACTER
; 0027  05    DCR   B ; DECREMENT BYTES COUNTER
; 0028  C2200  JNZ   MSGL ; RETURN FOR NEXT CHARACTER

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

COMMAND RECOGNIZING ROUTINE

;*****************************************************************************;
; FUNCTION: GETCM
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: GETCH,ECHO,ERROR
; DESTROYS: A,B,C,H,L,F/E'S
; DESCRIPTION: GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
; AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
; CHARACTER TABLE. IF SUCCESSFUL, THE ROUTINE
; CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
; A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
; IS TRANSFERRED TO THIS ROUTINE. IF THE CHARACTER
; DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
; THE ERROR HANDLER.

; GETCM:
; 0028 21ED13  LXI   H,MSTAK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
; 002E  F9    SPHL  ; STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
; 002F  0E2E  MVI   C,'.' ; PROMPT CHARACTER TO C
; 0031  CDF401 CALL  ECHO ; SEND PROMPT CHARACTER TO USER TERMINAL
; 0034  C3B00  JMP   GTCM3 ; WANT TO LEAVE ROOM FOR RST BRANCH
; 0038  C3FD13 ORG   RSTU ; ORG TO RST TRANSFER LOCATION
; 0038  C3FD13 JMP   USRBR ; JUMP TO USER BRANCH LOCATION
; 0038  GTCM3: CALL  GETCH ; GET COMMAND CHARACTER TO A
; 003E  CDF401 CALL  ECHO ; ECHO CHARACTER TO USER
; 0041  79    MOV   A,C ; PUT COMMAND CHARACTER INTO ACCUMULATOR
; 0042  010600 LXI   B,NCMDS ; C CONTAINS LOOP AND INDEX COUNT
; 0045  219603 LXI   H,CTAB ; HL POINTS INTO COMMAND TABLE
0048  GTC05:
0048  BE  CMP  M  ; COMPARE TABLE ENTRY AND CHARACTER
0049  CA5400  JZ  GTC10  ; BRANCH IF EQUAL - COMMAND RECOGNIZED
004C  23  INX  H  ; ELSE, INCREMENT TABLE POINTER
004D  8D  DCR  C  ; DECREMENT LOOP COUNT
004E  C24800  JNZ  GTC05  ; BRANCH IF NOT AT TABLE END
0051  C3D02  JMP  ERROR  ; ELSE, COMMAND CHARACTER IS ILLEGAL
0054  GTC10:
0054  21AB03  LXI  H,CADR  ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
0057  09  DAD  B  ; /OP COMMAND ROUTINE ADDRESSES
0058  09  DAD  B  ; ADD WHAT IS LEFT OF LOOP COUNT
0059  7E  MOV  A,M  ; ADD AGAIN - EACH ENTRY IN CADR IS 2 BYTES LONG
005A  23  INX  H  ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
005B  66  MOV  H,M  ; POINT TO NEXT BYTE IN TABLE
005C  6F  MOV  L,A  ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
005D  E9  PCHL  ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE

COMMAND IMPLEMENTING ROUTINES

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

005E  0E02  MVI  C,2  ; GET 2 NUMBERS FROM INPUT STREAM
0060  CD5702  CALL  GETNM
0063  D1  POP  D  ; ENDING ADDRESS TO DE
0064  E1  POP  H  ; STARTING ADDRESS TO HL
0065  DCM05:
0065  CDEE01  CALL  CROUT  ; ECHO CARRIAGE RETURN/LINE FEED
0068  7C  MOV  A,H  ; DISPLAY ADDRESS OF FIRST LOCATION IN LINE
0069  CDC302  CALL  NMOUT
006C  7D  MOV  A,L  ; ADDRESS IS 2 BYTES LONG
006D  CDC302  CALL  NMOUT
0070  DCM10:
0070  0E20  MVI  C,,'  ; USE BLANK AS SEPARATOR
0072  CDF401  CALL  ECHO
0075  7E  MOV  A,M  ; GET CONTENTS OF NEXT MEMORY LOCATION
0076  CDC302  CALL  NMOUT  ; DISPLAY CONTENTS
0800 MACRO ASSEMBLER, VER 2.3  ERRORS = 0  PAGE 7

0079  CDB01  CALL  BREAK  ; SEE IF USER WANTS OUT
007C  1  +  TRUE  DCM12  ; IF SO, BRANCH
007F  CD9C02  CALL  HILO  ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
0082  1  +  FALSE  DCM15  ; /GREATER THAN OR EQUAL TO ENDING ADDRESS
0085  DCM12:
008B  DCM15:
008F  CDEE81  CALL  CROUT  ; CARRIAGE RETURN/LINE FEED TO END LINE
0092  C36080  JMP  GETCM  ; ALL DONE
0095  C32000  JMP  GETCM  ; /START OF NEW LINE
0098  7D  MOV  A,L  ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
009B  E60F  ANI  NEWLN  ; /START OF NEW LINE
009E  C27000  JNZ  DCM10  ; NO - NOT AT END OF LINE
00A1  C36030  JMP  DCM10  ; YES - START NEW LINE WITH ADDRESS

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

0095  GCMD:
0095  CD2202  CALL  GETHX  ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
0098  1  +  FALSE  GCM05  ; BRANCH IF NO NUMBER PRESENT
009B  D2A00  +  JNC  GCM05
009C  7A  MOV  A,D  ; ELSE, GET TERMINATOR
009E  FE0D  CPI  CR  ; SEE IF CARRIAGE RETURN
00A1  C20D02  JNZ  ERROR  ; ERROR IF NOT PROPERLY TERMINATED
00A4  21F513  LXI  H,PSAVE  ; WANT NUMBER TO REPLACE SAVE PGM COUNTER
00A5  71  MOV  M,C
00A6  23  INX  H
00A7  70  MOV  M,B  ; IF NO STARTING ADDRESS, MAKE SURE THAT
00AB  C30000  JMP  GCM05:
00AD  7A  MOV  A,D  ; /CARRIAGE RETURN TERMINATED COMMAND
00B0  FE0D  CPI  CR  ; ERROR IF NOT
00B3  C20D02  JNZ  ERROR
00B6  GCM10:
00B8  C32033  JMP  RSTTF  ; RESTORE REGISTERS AND BEGIN EXECUTION

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

ICMD:
00B3 0E01 MVI C,1; GET SINGLE NUMBER FROM INPUT STREAM
00B5 CD5702 CALL GETNM
00B8 3EFF MVI A,UPPER
00BA 32913 STA TEMP; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
00BD D1 POP D; ADDRESS OF START TO DE
00BE ICM05: ICMB5:
00BE CD1B02 CALL GETCH; GET A CHARACTER FROM INPUT STREAM
00C1 4F MOV C,A; ECHO IT
00C2 CDF401 CALL ECHO
00C5 79 MOV A,C; PUT CHARACTER BACK INTO A
00C6 PE1B CPI TERM; SEE IF CHARACTER IS A TERMINATING CHARACTER
00C8 CAF400 JZ ICM25; IF SO, ALL DONE ENTERING CHARACTERS
00CB CD0A03 CALL VALDL; ELSE, SEE IF VALID DELIMITER
1 1 + TRUE ICM05; IF SO SIMPLY IGNORE THIS CHARACTER
00CE 1 DABEB0 + JC ICM05; ELSE, CHECK TO SEE IF VALID HEX DIGIT
00DF 1 CD6F03 CALL VALDG; IF NOT, BRANCH TO HANDLE ERROR CONDITION
00E0 1 D2E00 + JNC ICM28; IF NOT, BRANCH TO HANDLE ERROR CONDITION
00E3 CDDA01 CALL CVBN; CONVERT DIGIT TO BINARY
00E6 4F MOV C,A; MOVE RESULT TO C
00E9 CD5003 CALL STLFP; STORE IN APPROPRIATE HALF WORD
00EB 3AF913 LDA TEMP; GET HALF BYTE FLAG
00F1 B7 ORA A; SET F/F'S
00F2 C2E600 JNZ ICM10; BRANCH IF FLAG SET FOR UPPER
00F5 13 INX D; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
00F8 ICM10:
00F9 BEEF XRI INVRT; TOGGLE STATE OF FLAG
00FA 32F913 STA TEMP; PUT NEW VALUE OF FLAG BACK
00FB C3E00 JMP ICMB5; PROCESS NEXT DIGIT
00FC ICM20:
00FD CD4503 CALL STLFP; ILLEGAL CHARACTER
00FE C3D00 JMP ERROR; MAKE SURE ENTIRE BYTE FILLED THEN ERROR
00FF C4503 CALL STLFP; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
00F7 C5E00 CALL CROUT; ADD CARRIAGE RETURN
00F9 C32B00 JMP GETCH;
; FUNCTION: MCMD
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: GETCM, HILO, GETNM
; DESTROYS: A, B, C, D, E, H, L, F/F's
; DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.

; MCMD:
00FD  E003    MVI    C, 3
00FF  CD5702  CALL   GETNM ; GET 3 NUMBERS FROM INPUT STREAM
0102  C1      POP     B ; DESTINATION ADDRESS TO BC
0103  E1      POP     H ; ENDING ADDRESS TO HL
0104  D1      POP     D ; STARTING ADDRESS TO DE
0105  E5      PUSH    H ; SAVE ENDING ADDRESS
0106  62      MOV     H, D ; SOURCE ADDRESS TO HL
0107  68      MOV     L, E ; SOURCE ADDRESS TO HL
0108  7E      MOV     A, M ; GET SOURCE BYTE
0109  60      MOV     H, B
010A  69      MOV     L, C ; DESTINATION ADDRESS TO HL
010B  77      MOV     M, A ; MOVE BYTE TO DESTINATION
010C  03      INX     B ; INCREMENT DESTINATION ADDRESS
010D  78      MOV     A, B
010E  E1      ORA     C ; TEST FOR DESTINATION ADDRESS OVERFLOW
010F  CA2B00  CALL   GETCM ; IF SO, CAN TERMINATE COMMAND
0112  13      INX     D ; INCREMENT SOURCE ADDRESS
0113  E1      POP     H ; ELSE, GET BACK ENDING ADDRESS
0114  CD9C02  CALL   HILO ; SEE IF ENDING ADDR>=SOURCE ADDR
  1      JMP     false  GETCM ; IF NOT, COMMAND IS DONE
0117  D22B00  CALL   SCM05 ; MOVE ANOTHER BYTE

;**********************************************************
;
; FUNCTION: SCMD
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: GETMX, GETCM, NMOUT, ECHO
; DESTROYS: A, B, C, D, E, H, L, F/F's
; DESCRIPTION: SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.

; SCM05:
011D  CD2202  CALL   GETMX ; GET A NUMBER, IF PRESENT, FROM INPUT
0120  C5      PUSH    B ; GET NUMBER TO HL - DENOTES MEMORY LOCATION
0121  E1      POP     H
0122  7A      MOV     A, D ; GET TERMINATOR
0123  FE20    CPI     ; SEE IF SPACE
0125  CA2D01  JZ     SCM10 ; YES - CONTINUE PROCESSING
0120 FE2C CPI ',' ; ELSE, SEE IF COMMA
012A C22B00 JNZ GETCM ; NO - TERMINATE COMMAND
012D SCM10:
012D 7E MOV A,M ; GET CONTENTS OF SPECIFIED LOCATION TO A
012E CDC302 CALL NMOOUT ; DISPLAY CONTENTS ON CONSOLE
0131 0E2D MVI C, '-'
0133 CDF401 CALL ECHO ; USE DASH FOR SEPARATOR
0136 CD2202 CALL GETHX ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
0139 1 FALE SCN15 ; IF NO VALUE PRESENT, BRANCH
013D 1 D23D01 + JNC SCN15
013C 71 MOV M,C ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
013D 23 INX H ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
013E C32201 JMP SCN05

;*****************************************************************************
;
; FUNCTION: XCMD
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: GETCH, ECHO, REG0S, GETCM, ERROR, RGADR, NMOOUT, CROUT, GETHX
; DESTROYS: A,B,C,D,E,H,L,F/P'S
; DESCRIPTION: XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE {X} COMMAND.
;
; XCMD:
0141 CD1B02 CALL GETCH ; GET REGISTER IDENTIFIER
0144 4F MOV C,A
0145 CDF401 CALL ECHO ; ECHO IT
0148 79 MOV A,C
0149 FE0D CPI CR
014B C25401 JNZ XCM05 ; BRANCH IF NOT CARRIAGE RETURN
014E CDE602 CALL REG0S ; ELSE, DISPLAY REGISTER CONTENTS
0151 C32B00 JMP GETCM ; THEN TERMINATE COMMAND
0154 XCM05:
0154 4F MOV C,A ; GET REGISTER IDENTIFIER TO C
0155 CD1703 CALL RGADR ; CONVERT IDENTIFIER INTO RTAB TABLE ADDR
0158 C5 PUSH B
0159 E1 POP H ; PUT POINTER TO REGISTER ENTRY INTO HL
015A 0E20 MVI C,'.'
015C CDF401 CALL ECHO ; ECHO SPACE TO USER
015F 79 MOV A,C
0160 32F913 STA TEMP ; PUT SPACE INTO TEMP AS DELIMITER
0163 XCM10:
0163 3AF913 LDA TEMP ; GET TERMINATOR
0166 FE20 CPI ; SEE IF A BLANK
0169 CA7001 JZ XCM15 ; YES - GO CHECK POINTER INTO TABLE
016B FE2C CPI ; NO - SEE IF COMMA
016D C22B00 JNZ GETCM ; NO - MUST BE CARRIAGE RETURN TO END COMMAND
0170  XCM15:
0170  7E    MOV    A,M
0171  B7    ORA    A      ; SET P/F's
0172  C27B01 JNZ XCM18 ; BRANCH IF NOT AT END OF TABLE
0175  CD6E01 CALL    CROUT ; ELSE, OUTPUT CARRIAGE RETURN LINE FEED
0178  C32B00 JMP GETCM ; AND EXIT
017B  XCM18:
017B  E5    PUSH   H      ; PUT POINTER ON STACK
017C  5E    MOV    E,M
017D  1613  MOV    D,DATA SHR 8 ; FETCH ADDRESS OF SAVE LOCATION FROM TABLE
017F  23    INX    H
0180  46    MOV    B,M ; FETCH LENGTH FLAG FROM TABLE
0181  D5    PUSH   D ; SAVE ADDRESS OF SAVE LOCATION
0182  D5    PUSH   D
0183  E1    POP    H ; MOVE ADDRESS TO HL
0184  C5    PUSH   B ; SAVE LENGTH FLAG
0185  7E    MOV    A,M ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
0186  CD3C02 CALL    NMOUT ; DISPLAY IT
0189  F1    POP    PSW ; GET BACK LENGTH FLAG
018A  F5    PUSH   PSW ; SAVE IT AGAIN
018B  B7    ORA    A ; SET P/F's
018C  CA9401 JZ    XCM20 ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
018F  2B    DCX    H ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
0190  7E    MOV    A,M
0191  CDC382 CALL    NMOUT ; DISPLAY THEM
0194  XCM20:
0194  092D  MOV    C,'-'; USE DASH AS SEPARATOR
0196  CDF401 CALL    ECHO
0199  CD2202 CALL    GETBH ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
019C  D2B401 JNC    XCM30 ; NO - GO CHECK FOR NEXT REGISTER
019F  7A    MOV    A,D
01A0  32F913 STA    TEMP ; ELSE, SAVE THE TERMINATOR FOR NOW
01A3  F1    POP    PSW ; GET BACK LENGTH FLAG
01A4  E1    POP    H ; PUT ADDRESS OF SAVE LOCATION INTO HL
01A5  B7    ORA    A ; SET P/F's
01A6  CAAB01 JZ    XCM25 ; IF 8 BIT REGISTER, BRANCH
01A9  70    MOV    M,B ; SAVE UPPER 8 BITS
01AA  2B    DCX    H ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
01AB  XCM25:
01AB  71    MOV    N,C ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
01AC  XCM27:
01AC  110300 LXI    D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
01AF  51    POP    H ; POINTER INTO REGISTER TABLE RTAB
01B0  19    DAD    D ; ADD ENTRY SIZE TO POINTER
01B1  C36301 JMP    XCM10 ; DO NEXT REGISTER
01B4  XCM30
01B4  7A    MOV    A,D ; GET TERMINATOR
01B5  32F913 STA    TEMP ; SAVE IN MEMORY
01B6  D1    POP    D ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
01B9  D1    POP    D ; /OF SAVE LOCATION
UTILITY ROUTINES

FUNCTION: BREAK
INPUTS: NONE
OUTPUTS: CARRY - 1 IF ESCAPE CHARA
         - 0 IF ANY OTHER CHARACTER OR NO CHARACTER PENDING
CALLS: NOTHING
DESTROYS: A,F,P'S
DESCRIPTION: BREAK IS USED TO SENSE AN ESCAPE CHARACTER FROM
            THE USER. IF NO CHARACTER IS PENDING, OR IF THE
            PENDING CHARACTER IS NOT THE ESCAPE, THEN A FAILURE
            RETURN (CARRY=0) IS TAKEN. IN THIS CASE, THE
            PENDING CHARACTER (IF ANY) IS LOST. IF THE PENDING
            CHARACTER IS AN ESCAPE CHARACTER, BREAK TAKES A SUCCESS
            RETURN (CARRY=1).

BREAK:
01BD DBPB IN CONST ; GET CONSOLE STATUS
01BF E602 ANI HBR ; SEE IF CHARACTER PENDING
01C1 CA1802 JZ FRET ; NO - TAKE FAILURE RETURN
01C4 DBPA IN CNTN ; YES - PICK UP CHARACTER
01C6 E67F ANI PRTY0 ; STRIP OFF PARITY BIT
01C8 FE18 CPI BRCHR ; SEE IF BREAK CHARACTER
01CA CA4303 JZ SRET ; YES - SUCCESS RETURN
01CD C31802 JMP FRET ; NO - FAILURE RETURN - CHARACTER LOST

FUNCTION: CI
INPUTS: NONE
OUTPUTS: A - CHARACTER FROM CONSOLE
CALLS: NOTHING
DESTROYS: A,F,P'S
DESCRIPTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
CONSOLE AND THEN RETURNS THE CHARACTER, VIA THE A
REGISTER, TO THE CALLING ROUTINE. THIS ROUTINE
IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.

CI:
8080 MACRO ASSEMBLER, VER 2.3 ERRORS = 0 PAGE 13

01D0 DBFB IN CONST ; GET STATUS OF CONSOLE
01D2 E802 ANI RBR ; CHECK FOR RECEIVER BUFFER READY
01D4 CAD001 JZ CI ; NOT YET - WAIT
01D7 DBFA IN CNIN ; READY SO GET CHARACTER
01D9 C9 RET

;*****************************************************************************
;
; FUNCTION: CNVBN
; INPUTS: C - ASCII CHARACTER '0'-'9' OR 'A'-'F'
; OUTPUTS: A - 0 TO F HEX
; CALLS: NOTHING
; DESTROYS: A,P/F/S
; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
; CNVBN INTO ITS CORRESPONDING BINARY VALUE. CNVBN
; DOES NOT CHECK THE VALIDITY OF ITS INPUT.
;
01DA 79 MOV A,C ; SUBTRACT CODE FOR '0' FROM ARGUMENT
01DB D630 SUI '0' ; WANT TO TEST FOR RESULT OF 0 TO 9
01DD FE0A CPI 10 ; IF SO, THEN ALL DONE
01DF F9 RM ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
01E0 D607 SUI 7 ; SO RETURN AFTER SUBTRACTING BIAS OF 7
01E2 C9 RET

;*****************************************************************************
;
; FUNCTION: CO
; INPUTS: C - CHARACTER TO OUTPUT TO CONSOLE
; OUTPUTS: C - CHARACTER OUTPUT TO CONSOLE
; CALLS: NOTHING
; DESTROYS: A,P/F/S
; DESCRIPTION: CO WAITS UNTIL THE CONSOLE IS READY TO ACCEPT A CHARACTER
; AND THEN SENDS THE INPUT ARGUMENT TO THE CONSOLE.
;
01E3 DBFB IN CONST ; GET STATUS OF CONSOLE
01E5 E601 ANI TRDY ; SEE IF TRANSMITTER READY
01E7 CAE301 JZ CO ; NO - WAIT
01E9 79 MOV A,C ; ELSE, MOVE CHARACTER TO A REGISTER FOR OUTPUT
01EA D3FA OUT COUT ; SEND TO CONSOLE
01ED C9 RET

;*****************************************************************************
FUNCTION CROUT
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: ECHO
; DESTROYS: A,B,C,F/F'S
; DESCRIPTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE
; FEED) TO THE CONSOLE.

CROUT:
01EE 0E0D  MVI C,CR
01F0 CDF401  CALL ECHO
01F3  C9   RET

FUNCTION: ECHO
; INPUTS: C - CHARACTER TO ECHO TO TERMINAL
; OUTPUTS: C - CHARACTER ECHOED TO TERMINAL
; CALLS: CO
; DESTROYS: A,B,F/F'S
; DESCRIPTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA
; THE MONITOR, SENDS THAT CHARACTER TO THE USER
; TERMINAL. A CARRIAGE RETURN IS ECHOED AS A CARRIAGE
; RETURN LINE FEED, AND AN ESCAPE CHARACTER IS ECHOED AS $.

ECHO:
01F4 41   MOV B,C ; SAVE ARGUMENT
01F5 3E1B  MVI A,ESC
01F7 88   CMP B ; SEE IF ECHOING AN ESCAPE CHARACTER
01F8 C2FD01 JNZ ECH05 ; NO - BRANCH
01FB 0E24  MVI C,'$' ; YES - ECHO AS $
01FD ECH05  CALL CO ; DO OUTPUT THROUGH MONITOR
0200 3E0D  MVI A,CR
0202 88   CMP B ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RETURN
0203 C20A02 JNZ ECH10 ; NO - NO NEED TO TAKE SPECIAL ACTION
0206 0E0A  MVI C,LF ; YES - WANT TO ECHO LINE FEED, TOO
0208 CDE301 CALL CO

ECH10:
020B 48   MOV C,B ; RESTORE ARGUMENT
020C  C9   RET

FUNCTION: ERROR
; INPUTS: NONE
; OUTPUTS: NONE
; CALLS: ECHO, CROUT, GETCM
; DESTROYS: A, B, C, F/F's
; DESCRIPTION: ERROR PRINTS THE ERROR CHARACTER (CURRENTLY AN ASTERISK)
; ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
; AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.

020D     ERROR:
020D     0E2A   MVI   C, 7
020F     CDP401  CALL  ECHO  ; SEND * TO CONSOLE
0212     CDEEB1  CALL  CROUT  ; SKIP TO BEGINNING OF NEXT LINE
0215     C32B00  JMP  GETCM  ; TRY AGAIN FOR ANOTHER COMMAND

;******************************************************************************
;
; FUNCTION: FRET
; INPUTS: NONE
; OUTPUTS: CARRY - ALWAYS 0
; CALLS: NOTHING
; DESTROYS: CARRY
; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
; INDICATE FAILURE ON RETURN. FRET SETS THE CARRY
; FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
; CALLER OF THE ROUTINE INVOKING FRET.

0218     FRET:
0218     37   STC  ; FIRST SET CARRY TRUE
0219     3F   CMC  ; THEN COMPLEMENT IT TO MAKE IT FALSE
021A     C9   RST  ; RETURN APPROPRIATELY

;******************************************************************************
;
; FUNCTION: GETCH
; INPUTS: NONE
; OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
; CALLS: CI
; DESTROYS: A, C, F/F's
; DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
; TO THE CALLING PROGRAM.

021B     GETCH:
021B     CDD001  CALL  CI  ; GET CHARACTER FROM TERMINAL
021E     E67F   ANL  PR40  ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
0220     4F     MOV  C,A  ; PUT VALUE IN C REGISTER FOR RETURN
0221     C9     RET

;******************************************************************************
FUNCTION: GETHX
INPUTS: NONE
OUTPUTS: BC - 16 BIT INTEGER
D - CHARACTER WHICH TERMINATED THE INTEGER
CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
0 IF FIRST CHARACTER IS DELIMITER
CALLS: GETHX, ECHO, VALDL, VALDG, CWBN, ERROR
DESTROYS: A, B, C, D, E, F, P, F'S
DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
INTEGER. IF MORE THAN 4 HEX DIGITS ARE ENTERED,
ONLY THE LAST 4 ARE USED. THE NUMBER TERMINATES WHEN
A VALID DELIMITER IS ENCOUNTERED. THE DELIMITER IS
ALSO RETURNED AS AN OUTPUT OF THE FUNCTION. ILLEGAL
CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
ERROR INDICATION. IF THE FIRST (VALID) CHARACTER
ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
OF BC ARE UNDEFINED.

GETHX:
0222 E5 PUSH H ; SAVE HL
0223 210000 LXI H, 0 ; INITIALIZE RESULT
0226 1E00 MVI E, 0 ; INITIALIZE DIGIT FLAG TO FALSE
0228 GHX05:
0228 CD1B02 CALL GETHX ; GET A CHARACTER
022B 4F MOV C, A
022C CDF401 CALL ECHO ; ECHO THE CHARACTER
022F CD8A83 CALL VALDL ; SEE IF DELIMITER
1 FALSE GHX10 ; NO - BRANCH
0232 1 D24102 + JNC GHX10
0235 51 MOV D, C ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
0236 E5 PUSH H
0237 C1 POP B ; MOVE RESULT TO BC
0238 E1 POP H ; RESTORE HL
0239 7B MOV A, E ; GET FLAG
023A 87 ORA A ; SET P/F'S
023B C24303 JNZ SRET ; IF FLAG NON-0, A NUMBER HAS BEEN FOUND
023E CA1802 JS RET ; ELSE, DELIMITER WAS FIRST CHARACTER
0241 GHX10:
0241 CD6F03 CALL VALDG ; IF NOT DELIMITER, SEE IF Digit
1 FALSE ERROR ; ERROR IF NOT A VALID DIGIT, EITHER
0244 1 D20D02 + JNC ERROR
0247 CD6A01 CALL CWBN ; CONVERT DIGIT TO ITS BINARY VALUE
024A 1EFF MVI E, OF0H ; SET DIGIT FLAG NON-0
024C 29 DAD H ; *2
024D 29 DAD H ; *4
024E 29 DAD H ; *8
FUNCTION: GETNM

INPUTS: C - COUNT OF NUMBERS TO FIND IN INPUT STREAM
OUTPUTS: TOP OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON TOP
OF STACK)
CALLS: GETHX, HILO, ERROR
DESTROYS: A, B, C, D, E, H, L, F/P/S
DESCRIPTION: GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1
AND 3, INCLUSIVE, IN THE INPUT
STREAM AND RETURNS THEIR VALUES ON THE STACK. IF 2
OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
SECOND NUMBERS WILL BE SET EQUAL. THE LAST NUMBER
REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
OR AN ERROR INDICATION WILL RESULT.

0257 GETNM:
0257 2803 MVI L,3 ; PUT MAXIMUM ARGUMENT COUNT INTO L
0259 79 MOV A,C ; GET THE ACTUAL ARGUMENT COUNT
025A E603 ANI 3 ; FORCE TO MAXIMUM OF 3
025C C8 RZ ; IF 0, DON'T BOTHER TO DO ANYTHING
025D 67 MOV H,A ; ELSE, PUT ACTUAL COUNT INTO H
025E GNM05:
025E CD202 CALL GETHX ; GET A NUMBER FROM INPUT STREAM
1 + 1 D20D02 JNC ERROR ; ERROR IF NOT THERE - TOO FEW NUMBERS
0261 25 PUSH B ; ELSE, SAVE NUMBER ON STACK
0264 C5 DCR L ; DECREMENT MAXIMUM ARGUMENT COUNT
0265 2D DCR H ; DECREMENT ACTUAL ARGUMENT COUNT
0267 CA7S02 JZ GNM10 ; BRANCH IF NO MORE NUMBERS WANTED
026A 7A MOV A,D ; ELSE, GET NUMBER TERMINATOR TO A
026B FE0D CPI CR ; SEE IF CARRIAGE RETURN
026D CA0D02 JZ ERROR ; ERROR IF SO - TOO FEW NUMBERS
0270 C35E02 JMP GNM05 ; ELSE, PROCESS NEXT NUMBER
0273 GNM10:
0273 7A MOV A,D ; WHEN COUNT 0, CHECK LAST TERMINATOR
0274 FE0D CPI CR
0276 C20D02 JNZ ERROR ; ERROR IF NOT CARRIAGE RETURN
0279 01FFFF LXI B,0FFFFH ; HL GETS LARGEST NUMBER
027C 7D MOV A,L ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT
027D B7 ORA A ; CHECK FOR 0
027E CA5602 JZ GNM20 ; IF YES, 3 NUMBERS WERE INPUT
0281 C5 PUS H B ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFH
0282 2D DCR L
0283 C28102 JNZ GNM15
0286 GNM20:
0286 C1 POP B ; GET THE 3 ARGUMENTS OUT
0287 D1 POP D
0288 E1 POP H
0289 C9C02 CALL HILO ; SEE IF FIRST >= SECOND
+ 1 1 C09C02 JNC GNM25 ; NO - BRANCH
028C 1 D29102 + JNC GNM25
028F 5A MOV D,H
0290 5D MOV E,L ; YES - MAKE SECOND EQUAL TO THE FIRST
0291 GNM25:
0291 E3 XTHL ; PUT FIRST ON STACK - GET RETURN ADDR
0292 D5 PUSH D ; PUT SECOND ON STACK
0293 C5 PUSH B ; PUT THIRD ON STACK
0294 E5 PUSH H ; PUT RETURN ADDRESS ON STACK
0295 3D DCR A ; DECREMENT RESIDUAL COUNT
0296 F8 ADD ; IF NEGATIVE, PROPER RESULTS ON STACK
0297 E1 POP H ; ELSE, GET RETURN ADDR
0298 E3 XTHL ; REPLACE TOP RESULT WITH RETURN ADDR
0299 C39502 JMP GNM30 ; TRY AGAIN

***************************************************************
FUNCTION: HILO
INPUTS: DE - 16 BIT INTEGER
HL - 16 BIT INTEGER
OUTPUTS: CARRY - 0 IF HL<DE
- 1 IF HL>=DE
CALLS: NOTHING
DESTROYS: E/F'S
DESCRIPTION: HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE. THE
INTEGERS ARE TREATED AS UNSIGNED NUMBERS. THE CARRY
BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.

HILO:
029C C5 PUS H B ; SAVE BC
029D 47 MOV B,A ; SAVE A IN B REGISTER
029E E5 PUSH H ; SAVE HL PAIR
029F 7A MOV A,D ; CHECK FOR DE = 0000H
02A0 B3 ORA E
02A1 CADD02 JZ HIL05 ; WE'RE AUTOMATICALLY DONE IF IT IS
02A4 23 INX H ; INCREMENT HL BY 1
02A5 7C MOV A,H ; WANT TO TEST FOR 0 RESULT AFTER
02A6 B5 ORA L ; /INCREMENTING
02A7 CADD02 JZ HIL05 ; IF SO, HL MUST HAVE CONTAINED 0FFFH

46
02AA E1 POP H ; IF NOT, RESTORE ORIGINAL HL
02AB D5 PUSH D ; SAVE DE
02AC 3EFF MVI A,0FFH ; WANT TO TAKE 2'S COMPLEMENT OF DE CONTENTS
02AE AA XRA D
02AF 57 MOV D,A
02B0 3EFF MVI A,0FFH
02B1 AB XRA E
02B2 5F MOV E,A
02B4 13 INX D ; 2'S COMPLEMENT OF DE TO DE
02B5 7D MOV A,L
02B6 83 ADD E ; ADD HL AND DE
02B7 7C MOV A,H
02B8 8A ADC D ; THIS OPERATION SETS CARRY PROPERLY
02B9 D1 POP D ; RESTORE ORIGINAL DE CONTENTS
02BA 78 MOV A,B ; RESTORE ORIGINAL CONTENTS OF A
02BB C1 POP B ; RESTORE ORIGINAL CONTENTS OF BC
02BC C9 RET ; RETURN WITH CARRY SET AS REQUIRED
02BD E1 POP H ; IF HL CONTAINS 0FFFFFF, THEN CARRY CAN
02BE 78 MOV A,B ; ONLY BE SET TO 1
02BF C1 POP B ; RESTORE ORIGINAL CONTENTS OF REGISTERS
02C0 C34303 JMP SRET ; SET CARRY AND RETURN

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

FUNCTION: NMOUT

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

NMOUT:
02C3 E5 PUSH H ; SAVE HL - DESTROYED BY PRVAL
02C4 F5 PUSH PSW ; SAVE ARGUMENT
02C5 0F RRC
02C6 0F RRC
02C7 0F RRC
02C8 0F RRC ; GET UPPER 4 BITS TO LOW 4 BIT POSITIONS
02C9 E60F ANI HCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
02CB 4F MOV C,A
02CC CDE02 CALL PRVAL ; CONVERT LOWER 4 BITS TO ASCII
02CP CDF401 CALL ECHO ; SEND TO TERMINAL
02D2 F1 POP PSW ; GET BACK ARGUMENT
02D3 E60F ANI HCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
; 8080 MACRO ASSEMBLER, VER 2.3  ERRORS = 0  PAGE 28

02D5  4F    MOV    C,A 
02D6  CDDE02  CALL  PRVAL 
02D9  CDF401  CALL  ECHO 
02DC  E1    POP    H ; RESTORE SAVED VALUE OF HL 
02DD  C9    RET 

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

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

02DE  21BF03  LXI    H,DIGTB ; ADDRESS OF TABLE 
02E1  0600  MVI    B,0 ; CLEAR HIGH ORDER BITS OF BC 
02E3  09    DAD    B ; ADD DIGIT VALUE TO HL ADDRESS 
02E4  4E    MOV    C,M ; FETCH CHARACTER FROM MEMORY 
02E5  C9    RET 

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

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

02E6  21CF03  LXI    H,RTAB ; LOAD HL WITH ADDRESS OF START OF TABLE 
02E9  REG05: 
02E9  4E    MOV    C,M ; GET PRINT SYMBOL OF REGISTER 
02EA  79    MOV    A,C 
02EB  B7    ORA    A ; TEST FOR 0 - END OF TABLE 
02EC  C2F302  JNZ    REG0 ; IF NOT END, BRANCH 
02EE  CD0E01  CALL  COUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END 
02F2  C9    RET 
02F3  REG10:

48
02F3  CDF401  CALL  ECHO  ; ECHO CHARACTER
02F6  0E3D  MVI  C, '='
02F8  CDF401  CALL  ECHO  ; OUTPUT EQUALS SIGN, I.E. A=
02FB  23  INX  H  ; POINT TO START OF SAVE LOCATION ADDRESS
02FC  5E  MOV  E, M  ; GET LSP OF SAVE LOCATION ADDRESS TO E
02FD  1513  MVI  D, DATA SHR 0  ; PUT MSP OF SAVE LOC ADDRESS INTO D
02FF  23  INX  H  ; POINT TO LENGTH FLAG
0300  1A  LDAX  D  ; GET CONTENTS OF SAVE ADDRESS
0301  CDC302  CALL  NMOUT  ; DISPLAY ON CONSOLE
0304  7E  MOV  A, M  ; GET LENGTH FLAG
0305  B7  ORA  A  ; SET SIGN F/F
0306  CA0203  JR  REG15  ; IF 0, REGISTER IS 8 BITS
0309  1B  CDX  D  ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
030A  1A  LDAX  D  ; GET LOWER 8 BITS
030B  CDC302  CALL  NMOUT  ; DISPLAY THEM
030C  REG15:
030E  0E20  MVI  C, ','
0310  CDF401  CALL  ECHO
0313  23  INX  H  ; POINT TO START OF NEXT TABLE ENTRY
0314  C3E902  JMP  REG05  ; DO NEXT REGISTER

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

FUNCTION: RGADR

INPUTS: C - CHARACTER DENOTING REGISTER
OUTPUTS: BC - ADDRESS OF ENTRY IN RTAB CORRESPONDING TO REGISTER
CALLS: ERROR
DESTROYS: A, B, C, D, E, H, L, F/F'S
DESCRIPTION: RGADR TAKES A SINGLE CHARACTER AS INPUT. THIS CHARACTER
DENOTES A REGISTER. RGADR SEARCHES THE TABLE RTAB
FOR A MATCH ON THE INPUT ARGUMENT. IF ONE OCCURS,
RGADR RETURNS THE ADDRESS OF THE ADDRESS OF THE
SAVE LOCATION CORRESPONDING TO THE REGISTER. THIS
ADDRESS POINTS INTO RTAB. IF NO MATCH OCCURS, THEN
THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS
PASSED TO THE ERROR ROUTINE.

0317  RGADR:
0317  21CF03  LXI  H, RTAB  ; HL GETS ADDRESS OF TABLE START
031A  110300  LDI  D, RTAB$  ; DE GET SIZE OF A TABLE ENTRY
031D  RGA05:
0320  7E  MOV  A, M  ; GET REGISTER IDENTIFIER
0321  B7  ORA  A  ; CHECK FOR TABLE END (IDENTIFIER IS 0)
0322  CA0D02  JZ  ERROR  ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL
0323  B9  CMP  C  ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT
0324  CA2A03  JZ  RGA10  ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR
0326  19  DAD  D  ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY
0327  C31D03  JMP  RGA05  ; TRY AGAIN
032A  RGA10:
032A  23    INX    H      ; IF A MATCH, INCREMENT TABLE POINTER TO 
032B  44    MOV    B,H    ; SAVE LOCATION ADDRESS 
032C  4D    MOV    C,L    ; RETURN THIS VALUE 
032D  C9    RET 

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

; FUNCTION: RSTTF 
; INPUTS: NONE 
; OUTPUTS: NONE 
; CALLS: NOTHING 
; DESTROYS: A,B,C,D,E,H,L,SP,PSW 
; DESCRIPTION: RSTTF RESTORES ALL CPU REGISTER, PLL/FLOPS, STACK 
; POINTER AND PROGRAM COUNTER FROM THEIR RESPECTIVE 
; SAVE LOCATIONS IN MEMORY. THE ROUTINE THEN TRANSFERS 
; CONTROL TO THE LOCATION SPECIFIED BY THE PROGRAM 
; COUNTER (I.E. THE RESTORED VALUE). THE ROUTINE 
; EXITS WITH THE INTERRUPTS ENABLED. 

032E  F3    DI      ; DISABLE INTERRUPTS WHILE RESTORING THINGS 
032F  21ED13  LDI    H,NSTAK ; SET MONITOR STACK POINTER TO START OF STACK 
0330  F9    SPHL   
0331  D1    POP    D      ; START ALSO END OF REGISTER SAVE AREA 
0332  C1    POP    B 
0333  F1    POP    PSW   
0334  2AF713  LHL    SSAVE ; RESTORE USER STACK POINTER 
0335  F9    SPHL   
0336  2AF513  LHL    PSAVE ; PUT USER RETURN ADDRESS ON USER STACK 
0337  E5    PUSH   H      
0338  2AF313  LHL    LSAVE ; RESTORE HL REGISTERS 
0339  PB    EI     ; ENABLE INTERRUPTS NOW 
033A  C9    RET     ; JUMP TO RESTORED PC LOCATION 

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

; FUNCTION: SRET 
; INPUTS: NONE 
; OUTPUTS: CARRY = 1 
; CALLS: NOTHING 
; DESTROYS: CARRY 
; DESCRIPTION: SRET IS JUMPED TO BY Routines WISHING TO RETURN SUCCESS. 
; SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE 
; CALLER OF THE ROUTINE INVOKING SRET. 

033B  37    STC    ; SET CARRY TRUE
0344 C9    RET    ; RETURN APPROPRIATELY

FUNCTION: STHF0
INPUTS: DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
OUTPUTS: NONE
CALLS: STHLF
DESTROYS: A,B,C,H,L,F/F’S
DESCRIPTION: STHF0 CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF
IT IS SET TO LOWER. IF SO, STHF0 STORES A 0 TO
PAD OUT THE LOWER HALF OF THE ADDRESSED BYTE;
OTHERWISE, THE ROUTINE TAKES NO ACTION.

0345 3AP913 LDA TEMP   ; GET HALF BYTE FLAG
0348 B7   ORA A       ; SET F/F’S
0349 C0   RNZ         ; IF SET TO UPPER, DON’T DO ANYTHING
034A 0E00 MVI C,0     ; ELSE, WANT TO STORE THE VALUE 0
034C CD5003 CALL STHLF ; DO IT
034F C9   RET

FUNCTION: STHLF
INPUTS: C - 4 BIT VALUE TO BE STORED IN HALF BYTE
DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
OUTPUTS: NONE
CALLS: NOTHING
DESTROYS: A,B,C,H,L,F/F’S
DESCRIPTION: STHLF TAKES THE 4 BIT VALUE IN C AND STORES IT IN
HALF OF THE BYTE ADDRESSED BY REGISTERS DE. THE
HALF BYTE USED (EITHER UPPER OR LOWER) IS DENOTED
BY THE VALUE OF THE FLAG IN TEMP. STHLF ASSUMES
THAT THIS FLAG HAS BEEN PREVIOUSLY SET
(NOMINALLY BY ICMD).

STHLF:
0350 D5   PUSH D      ; MOVE ADDRESS OF BYTE INTO HL
0351 E1   POP H       ; MOVE ADDRESS OF BYTE INTO HL
0352 79   MOV A,C     ; GET VALUE
0353 E50F  ANI 0FH    ; FORCE TO 4 BIT LENGTH
0355 4F   MOV C,A     ; PUT VALUE BACK
0356 3AP913 LDA TEMP  ; GET HALF BYTE FLAG
0359 B7   ORA A       ; CHECK FOR LOWER HALF
035A C26303 JNZ STH05 ; BRANCH IF NOT
035D 7E   MOV A,M     ; ELSE, GET BYTE

51
8080 MACRO ASSEMBLER, VER 2.3  ERRORS = 0  PAGE 24

0360  E600  ANI  0F00H  ; CLEAR LOWER 4 BITS
0361  B1  ORA  C  ; OR IN VALUE
0362  77  MOV  M,A  ; PUT BYTE BACK
0362  C9  RET

0363  STH05:
0363  7E  MOV  A,M  ; IF UPPER HALF, GET BYTE
0364  60F  ANI  0FH  ; CLEAR UPPER 4 BITS
0366  47  MOV  B,A  ; SAVE BYTE IN B
0367  79  MOV  A,C  ; GET VALUE
0368  0F  RRC
0369  0F  RRC
036A  0F  RRC  ; ALIGN TO UPPER 4 BITS
036B  0F  RRC
036C  00  ORA  B  ; OR IN ORIGINAL LOWER 4 BITS
036D  77  MOV  M,A  ; PUT NEW CONFIGURATION BACK
036E  C9  RET

******************************************************************************
;
FUNCTION: VALDG
; INPUTS: C - ASCII CHARACTER
; OUTPUTS: CARRY - 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
; CALLS: NOTHING
; DESTROYS: A,F/F'S
; DESCRIPTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
; AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
; (0-F), AND FAILURE OTHERWISE.

VALDG:
036F  79  MOV  A,C
0370  FE30  CPI  '0'  ; TEST CHARACTER AGAINST '0'
0372  FA1002  JM  FRET  ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
0375  FE30  CPI  '9'  ; ELSE, SEE IF IN RANGE '0'-'9'
0377  FA4303  JM  SRET  ; CODE BETWEEN '0' AND '9'
037A  CA4303  JZ  SRET  ; CODE EQUAL '9'
037D  FE41  CPI  'A'  ; NOT A DIGIT - TRY FOR A LETTER
037F  FA1002  JM  FRET  ; NO - CODE BETWEEN '9' AND 'A'
0382  FE41  CPI  'G'
0384  F21002  JP  FRET  ; NO - CODE GREATER THAN 'F'
0387  C34303  JMP  SRET  ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE

******************************************************************************
;
FUNCTION: VALDL
; INPUTS: C - CHARACTER
; OUTPUTS: CARRY - 1 IF INPUT ARGUMENT VALID DELIMITER
CALLS: NOTHING
DESTROYS: A,F/F'S
DESCRIPTION: VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID
DELIMITER CHARACTER (SPACE, COMMA, CARRIAGE RETURN) AND
FAILURE OTHERWISE.

VALDL:

038A 79 MOV A,C ; CHECK FOR COMMA
038B FE2C CPI ;
038D CA4303 JZ SRET ; CHECK FOR SPACE
0390 FE8D CPI CR ; CHECK FOR CARRIAGE RETURN
0392 CA4303 JZ SRET
0395 FE20 CPI ;
0397 CA4303 JZ SRET
039A C31802 JMP PRET ; ERROR IF NONE OF THE ABOVE

MONITOR TABLES

SIGNON: ; SIGNON MESSAGE
039D 0D0A4D43 DB CR,LF,'MCS-80 KIT',CR,LF
03A1 532D3830
03A5 204B4954
03A9 0D0A
000E LSIGNON EQU $SIGNON ; LENGTH OF SIGNON MESSAGE

CASDR: ; TABLE OF ADDRESSES OF COMMAND ROUTINES
03AB 0000 DW $ ; DUMMY
03AD 4101 DW XCMD
03AF 1D01 DW SCMD
03B1 FD00 DW MCMD
03B3 B300 DW ICMD
03B5 9500 DW GCMD
03B7 5E00 DW DCMD

CTAB: ; TABLE OF VALID COMMAND CHARACTERS

44 DB 'D'
03BB 47 DB 'C'
03BC 49 DB 'I'
03BD 4D DB 'M'
03BE 53 DB 'S'
03BF 56 DB 'X'
0006 NCMDSC EQU $CTAB ; NUMBER OF VALID COMMANDS
; DIGTB: ; TABLE OF PRINT VALUES OF HEX DIGITS

038F 30 DB '0'
038F 31 DB '1'
038F 32 DB '2'
038F 33 DB '3'
038F 34 DB '4'
038F 35 DB '5'
038F 36 DB '6'
038F 37 DB '7'
038F 38 DB '8'
038F 39 DB '9'
038F 3A DB 'A'
038F 3B DB 'B'
038F 3C DB 'C'
038F 3D DB 'D'
038F 3E DB 'E'
038F 3F DB 'F'

; RTAB: ; TABLE OF REGISTER INFORMATION

03C0 41 DB 'A'; REGISTER IDENTIFIER
03D0 F2 DB ASAVE AND 0FFH ; ADDRESS OF REGISTER SAVE LOCATION
03D1 00 DB 0 ; LENGTH FLAG - 0=9 BITS, 1=16 BITS
0383 RTAB S=RTAB ; SIZE OF AN ENTRY IN THIS TABLE
03D2 42 DB 'B'
03D3 F0 DB BSsave AND 0FFH
03D4 00 DB 0
03D5 43 DB 'C'
03D6 EF DB CSave AND 0FFH
03D7 00 DB 0
03D8 44 DB 'D'
03D9 EE DB DSsave AND 0FFH
03DA 00 DB 0
03DB 45 DB 'E'
03DC ED DB ESsave AND 0FFH
03DD 00 DB 0
03DE 46 DB 'F'
03DF F1 DB FSsave AND 0FFH
03E0 00 DB 0
03E1 40 DB 'G'
03E2 F4 DB Hsave AND 0FFH
03E3 00 DB 0
03E4 4C DB 'L'
03E5 F3 DB Lsave AND 0FFH
03E6 00 DB 0
03E7 40 DB 'M'
03E8 F4 DB Hsave AND 0FFH
03E9 01 DB 1
03EA 50 DB 'P'
03EB F6 DB Psave+1 AND 0FFH
03EC 01 DB 1
03ED 53   DB  'S'
03EE F8   DB  SSAVE+1 AND 0FFE
03EF 01   DB  1
03F0 00   DB  0 ; END OF TABLE MARKERS
03F1 00   DB  0
03FA   ORG BRTAB
03FB C3E301 JMP CO ; BRANCH TABLE FOR USER ACCESSIBLE ROUTINES
03FC C3D001 JMP CI

;*********************************************************************
;*********************************************************************
; 1300 ORG DATA ; ORG TO REGISTER SAVE - STACK GOES IN HERE
; 1300 ORG REGS
; 1300 MSTAK EQU $ ; START OF MONITOR STACK
1301 BSAVE: DB  0 ; E REGISTER SAVE LOCATION
1302 DSAVE: DB  0 ; D REGISTER SAVE LOCATION
1303 CSAVE: DB  0 ; C REGISTER SAVE LOCATION
1304 BSAVE: DB  0 ; D REGISTER SAVE LOCATION
1305 FSAVE: DB  0 ; FLAGS SAVE LOCATION
1306 ASAVE: DB  0 ; A REGISTER SAVE LOCATION
1307 LSAVE: DB  0 ; L REGISTER SAVE LOCATION
1308 HSAVE: DB  0 ; H REGISTER SAVE LOCATION
1309 PSAVE: DW  0 ; PGM COUNTER SAVE LOCATION
130A SSAVE: DW  0 ; USER STACK POINTER SAVE LOCATION
130B TEMP: DB  0 ; TEMPORARY MONITOR CELL
130C BRC# BRLOC ; ORG TO USER BRANCH LOCATION
0003 USRB#$ DS  3 ; BRANCH GOES IN HERE
; END

NO PROGRAM ERRORS
<table>
<thead>
<tr>
<th>Symbol Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0007</td>
</tr>
<tr>
<td>BREAK 01BD</td>
</tr>
<tr>
<td>C 0001</td>
</tr>
<tr>
<td>CNTRL 00FB</td>
</tr>
<tr>
<td>CO 01E3</td>
</tr>
<tr>
<td>CSAVE 13EF</td>
</tr>
<tr>
<td>DCM05 0065</td>
</tr>
<tr>
<td>DCM1 005E</td>
</tr>
<tr>
<td>ECH05 01FD</td>
</tr>
<tr>
<td>ESAVE 13ED</td>
</tr>
<tr>
<td>FSAYE 13F1</td>
</tr>
<tr>
<td>GETCH 021B</td>
</tr>
<tr>
<td>GML05 0228</td>
</tr>
<tr>
<td>GNML5 0281</td>
</tr>
<tr>
<td>GO 0008 *</td>
</tr>
<tr>
<td>H 0004</td>
</tr>
<tr>
<td>HSAVE 13F4</td>
</tr>
<tr>
<td>ICM25 00F4</td>
</tr>
<tr>
<td>IF 000A</td>
</tr>
<tr>
<td>M 0006</td>
</tr>
<tr>
<td>MSGL 002E</td>
</tr>
<tr>
<td>MNOUT 02C3</td>
</tr>
<tr>
<td>PSW 0006</td>
</tr>
<tr>
<td>REG15 030E</td>
</tr>
<tr>
<td>RGA10 032A</td>
</tr>
<tr>
<td>RTAB 03CF</td>
</tr>
<tr>
<td>SCM15 013D</td>
</tr>
<tr>
<td>SRET 0343</td>
</tr>
<tr>
<td>STNLP 0350</td>
</tr>
<tr>
<td>TRUE 0F9P</td>
</tr>
<tr>
<td>VALDL 038A</td>
</tr>
<tr>
<td>XCM18 017B</td>
</tr>
<tr>
<td>XCM30 01B4</td>
</tr>
</tbody>
</table>

* 02
* 03
* 04
* 05
* 06
* 07
* 08
* 09
* 10
* 11
* 12
* 13
..