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
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.
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, 110, 300, 600, 1200, 2400, 4800

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 format)
Width: 12.0 in.
Height: 6.75 in.
Depth: 0.50 in.
Weight: approx. 12 oz.

Electrical Characteristics (DC Power)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>5V ±5%</td>
</tr>
<tr>
<td>V&lt;sub&gt;DD&lt;/sub&gt;</td>
<td>12V ±5%</td>
</tr>
<tr>
<td>V&lt;sub&gt;BB&lt;/sub&gt;</td>
<td>-10V ±5%</td>
</tr>
<tr>
<td>or</td>
<td>-12V ±5%</td>
</tr>
</tbody>
</table>

1.3 Amps
.35 Amps
.20 Amps

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: 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.
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

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
- Miscellaneous non-Intel component pack

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

- 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>
<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>Part No.</th>
<th>Description</th>
<th>Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54-028 IC, 74161</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>54-092 IC, 7406</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>54-135 IC, 79M05AUC</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>54-136 IC, 93S16</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>68-009 Socket, 40-pin</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>68-007 Socket, 24-pin</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>68-177 Socket, 28-pin</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>68-102 Connect female, right angle, or 25-pin</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>68-077 Connect male, 25-pin</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>68-006 Socket, 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-AAA</td>
</tr>
<tr>
<td>68-077</td>
<td>Male 25 pin connector (solder) Low profile 18 pin DIP socket</td>
<td>Cinch Jones</td>
<td>DB-25P</td>
</tr>
<tr>
<td>68-007</td>
<td>Low profile 24 pin DIP socket</td>
<td>TI</td>
<td>C93-24-02</td>
</tr>
<tr>
<td>68-009</td>
<td>Low profile 40 pin DIP socket</td>
<td>TI</td>
<td>C93-40-02</td>
</tr>
<tr>
<td></td>
<td>Small 30/60 PCB connector Small 30/60 PCB connector (wire-wrap)</td>
<td>CDC</td>
<td>97167901 (w-w)</td>
</tr>
<tr>
<td></td>
<td>Large 43/86 PCB connector</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 silkscreen. 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 2-3. Parts List, Small Component Bag

<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 (brown-black-orange)</td>
<td>1</td>
</tr>
<tr>
<td>56-024</td>
<td>Resistor, 1 KΩ, 1/4W (brown-black-red)</td>
<td>16</td>
</tr>
<tr>
<td>56-033</td>
<td>Resistor, 2.7 KΩ, 1/4W (red-violet-red)</td>
<td>1</td>
</tr>
<tr>
<td>56-006</td>
<td>Resistor, 47 Ω, 1/4W (yellow-violet-black)</td>
<td>2</td>
</tr>
<tr>
<td>56-017</td>
<td>Resistor, 390 Ω, 1/4W (orange-white-brown)</td>
<td>1</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 (yellow-violet-red)</td>
<td>1</td>
</tr>
<tr>
<td>56-209</td>
<td>Resistor, 560 KΩ, 1/4W (green-blue-yellow)</td>
<td>1</td>
</tr>
<tr>
<td>56-039</td>
<td>Resistor, 5.1 KΩ, 1/4W (green-brown-yellow)</td>
<td>2</td>
</tr>
<tr>
<td>56-112</td>
<td>Resistor, 150 Ω, 1/4W (brown-green-brown)</td>
<td>1</td>
</tr>
<tr>
<td>56-020</td>
<td>Resistor, 510 Ω, 1/4W (green-brown-brown)</td>
<td>1</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 (brown-orange-brown)</td>
<td>1</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 pF, 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>84-010</td>
<td>Screw, 4-40 x 3/8&quot; pan</td>
<td>1</td>
</tr>
<tr>
<td>84-073</td>
<td>Screw, 2-56 x 3/8&quot; pan</td>
<td>4</td>
</tr>
<tr>
<td>84-016</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>84-027</td>
<td>Washer, #6 nylon</td>
<td>5</td>
</tr>
<tr>
<td>84-070</td>
<td>Nut, 6-32</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, silk screened 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.
MCS-80™ SYSTEM DESIGN KIT

Figure 2-5. Construction Figure #3.

MCS-80™ SYSTEM DESIGN KIT

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.

Table 2-5. Construction Table #2.

<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>

Note 1. (From Table 2-5)
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).

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.

**IMPORTANT**

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

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.

**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).
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 2-12. 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 BUSSES**

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.

$\omega_1$ and $\omega_2$ Clocks

$\omega_1$ and $\omega_2$ 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 issues status 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, $\omega_1$ and $\omega_2$, 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 jumpered 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.

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.

**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 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.
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 Jumpers</th>
<th>TTY Configuration Jumpers</th>
<th>TTL Configuration Jumpers</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></td>
</tr>
<tr>
<td>27 to 28</td>
<td>15 to 16</td>
<td></td>
</tr>
<tr>
<td>21 to 22</td>
<td>21 to 22</td>
<td></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 alphabetically-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>CRT REC.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CRT XMIT</td>
<td>DATA</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>+12 VDC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SIGNAL GND</td>
<td></td>
<td>SIGNAL GND</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>+12 VDC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>TTY REC</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>TTY XMIT</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>+12 VDC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>TTY REC</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>TTY XMIT</td>
<td>RETURN</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>Ø2 (TTL)</td>
<td>2.048 MHz Clock</td>
</tr>
<tr>
<td>D</td>
<td>DSR</td>
<td>Data Set Ready</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>OSC</td>
<td>18.432 MHz Oscillator</td>
</tr>
<tr>
<td>L</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.
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).

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>
<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>
<tr>
<td></td>
<td>-12V ±5%</td>
<td></td>
<td></td>
<td></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.
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 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.

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 in and console character out, 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
0009 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 "$") terminates the digit string. If an odd number of hex digits have been entered, a 0 will be appended to the string.

Example

11410
112233445566778899$  
This command puts the following pattern into RAM:
1410 11 22 33 44 55 66 77 88 99
11440
1234567899$

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 to 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 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

01400, 145G*

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

ISIS-II 8080/8085 MACRO ASSEMBLER, X108           SDK80  PAGE  1

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>;***********************************************************************</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>; ABSTRACT</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>; =========</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>; THIS PROGRAM RUNS ON THE 8080A BOARD AND IS DESIGNED TO PROVIDE</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>; THE USER WITH A MINIMAL MONITOR. BY USING THIS PROGRAM,</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>; THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>; A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>; ALREADY IN MEMORY. THE MONITOR ALSO PROVIDES THE USER WITH</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>; ROUTINES FOR PERFORMING CONSOLE I/O.</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>; PROGRAM ORGANIZATION</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>; =========</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>; THE LISTING IS ORGANIZED IN THE FOLLOWING WAY. FIRST THE COMMAND</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>; RECOGNIZER, WHICH IS THE HIGHEST LEVEL ROUTINE IN THE PROGRAM.</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>; NEXT, ARE THE ROUTINES TO IMPLEMENT THE VARIOUS COMMANDS, FINALLY</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td>; THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK. WITHIN</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td>; EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td>; ORDER, BY ENTRY POINT OF THE ROUTINE.</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td>; THIS PROGRAM EXPECTS TO RUN IN THE FIRST 1K OF ADDRESS SPACE.</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td>; IF, FOR SOME REASON, THE PROGRAM IS RE-ORG’ED, CARE SHOULD</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td>; BE TAKEN TO MAKE SURE THAT THE TRANSFER INSTRUCTIONS FOR RST 1</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td>; AND RST 7 ARE ADJUSTED APPROPRIATELY.</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td>; THE PROGRAM ALSO EXPECTS THAT RAM LOCATIONS 5K-1 TO 5K-256,</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td>; INCLUSIVE, ARE RESERVED FOR THE PROGRAM’S OWN USE. THESE</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td>; LOCATIONS MAY BE ALTERED, HOWEVER, BY CHANGING THE EQU’ED</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td>; SYMBOL “DATA” AS DESIRED.</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td>; LIST OF FUNCTIONS</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td>; ===========</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td>; GETCM</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td>; DCMD</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td>; GCMD</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
<td>LINE</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>52</td>
<td>;</td>
<td>52</td>
<td>ICMD</td>
</tr>
<tr>
<td>53</td>
<td>;</td>
<td>53</td>
<td>MCMD</td>
</tr>
<tr>
<td>54</td>
<td>;</td>
<td>54</td>
<td>SCMD</td>
</tr>
<tr>
<td>55</td>
<td>;</td>
<td>55</td>
<td>XCMD</td>
</tr>
<tr>
<td>56</td>
<td>;</td>
<td>56</td>
<td>----</td>
</tr>
<tr>
<td>57</td>
<td>;</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>;</td>
<td>58</td>
<td>BREAK</td>
</tr>
<tr>
<td>59</td>
<td>;</td>
<td>59</td>
<td>CI</td>
</tr>
<tr>
<td>60</td>
<td>;</td>
<td>60</td>
<td>CNVBN</td>
</tr>
<tr>
<td>61</td>
<td>;</td>
<td>61</td>
<td>CO</td>
</tr>
<tr>
<td>62</td>
<td>;</td>
<td>62</td>
<td>CROUT</td>
</tr>
<tr>
<td>63</td>
<td>;</td>
<td>63</td>
<td>ECHO</td>
</tr>
<tr>
<td>64</td>
<td>;</td>
<td>64</td>
<td>ERROR</td>
</tr>
<tr>
<td>65</td>
<td>;</td>
<td>65</td>
<td>FRET</td>
</tr>
<tr>
<td>66</td>
<td>;</td>
<td>66</td>
<td>GETCH</td>
</tr>
<tr>
<td>67</td>
<td>;</td>
<td>67</td>
<td>GETMX</td>
</tr>
<tr>
<td>68</td>
<td>;</td>
<td>68</td>
<td>GETNM</td>
</tr>
<tr>
<td>69</td>
<td>;</td>
<td>69</td>
<td>HILO</td>
</tr>
<tr>
<td>70</td>
<td>;</td>
<td>70</td>
<td>NMOUT</td>
</tr>
<tr>
<td>71</td>
<td>;</td>
<td>71</td>
<td>PRVAL</td>
</tr>
<tr>
<td>72</td>
<td>;</td>
<td>72</td>
<td>REGDS</td>
</tr>
<tr>
<td>73</td>
<td>;</td>
<td>73</td>
<td>RGADR</td>
</tr>
<tr>
<td>74</td>
<td>;</td>
<td>74</td>
<td>RSTTF</td>
</tr>
<tr>
<td>75</td>
<td>;</td>
<td>75</td>
<td>SRET</td>
</tr>
<tr>
<td>76</td>
<td>;</td>
<td>76</td>
<td>STHF0</td>
</tr>
<tr>
<td>77</td>
<td>;</td>
<td>77</td>
<td>STHLF</td>
</tr>
<tr>
<td>78</td>
<td>;</td>
<td>78</td>
<td>VALDG</td>
</tr>
<tr>
<td>79</td>
<td>;</td>
<td>79</td>
<td>VALDL</td>
</tr>
<tr>
<td>80</td>
<td>;</td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>;</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td></td>
<td>82</td>
<td>ORG 0H</td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td>83</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
<td>;</td>
<td>84</td>
<td><strong>MONITOR EQUATES</strong></td>
</tr>
<tr>
<td>85</td>
<td>;</td>
<td>85</td>
<td><strong>MONITOR EQUATES</strong></td>
</tr>
<tr>
<td>86</td>
<td>;</td>
<td>86</td>
<td><strong>MONITOR EQUATES</strong></td>
</tr>
<tr>
<td>87</td>
<td>;</td>
<td>87</td>
<td><strong>MONITOR EQUATES</strong></td>
</tr>
<tr>
<td>88</td>
<td>;</td>
<td>88</td>
<td><strong>MONITOR EQUATES</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>89</td>
<td>;</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>;</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>001B</td>
<td></td>
<td>91</td>
<td>BRCHR EQU 1BH</td>
</tr>
<tr>
<td>13FD</td>
<td></td>
<td>92</td>
<td>BSLCE EQU 13FDH</td>
</tr>
<tr>
<td>03FA</td>
<td></td>
<td>93</td>
<td>BRTAB EQU 3FAH</td>
</tr>
<tr>
<td>0027</td>
<td></td>
<td>94</td>
<td>CMD EQU 027H</td>
</tr>
<tr>
<td>00FB</td>
<td></td>
<td>95</td>
<td>CNCTL EQU 0FBB</td>
</tr>
<tr>
<td>00FA</td>
<td></td>
<td>96</td>
<td>CNIN EQU 0FAH</td>
</tr>
<tr>
<td>00FA</td>
<td></td>
<td>97</td>
<td>CNOUT EQU 0FAH</td>
</tr>
<tr>
<td>00FB</td>
<td></td>
<td>98</td>
<td>CONST EQU 0FBH</td>
</tr>
<tr>
<td>000D</td>
<td></td>
<td>99</td>
<td>CR EQU 0DH</td>
</tr>
<tr>
<td>1300</td>
<td></td>
<td>100</td>
<td>DATA EQU 5*1024-256</td>
</tr>
<tr>
<td>001B</td>
<td></td>
<td>101</td>
<td>ESC EQU 1BH</td>
</tr>
<tr>
<td>000F</td>
<td></td>
<td>102</td>
<td>HCHAR EQU 0FH</td>
</tr>
<tr>
<td>00FF</td>
<td></td>
<td>103</td>
<td>INVRT EQU 0FFH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOC  OBJ  LINE  SOURCE STATEMENT

000A  104 LF   EQU  0AH    ; CODE FOR LINE FEED
0000  105 LOWER EQU  0       ; DENOTES LOWER HALF OF BYTE IN ICMD
0006  106 LSGNON EQU ---    ; LENGTH OF SIGNON MESSAGE - DEFINED LATER
00CF  107 MODE EQU 0CFH    ; MODE SET FOR USART INITIALIZATION
0108  108 ;MSTAK EQU ---    ; START OF MONITOR STACK - DEFINED LATER
0109  109 ;NCMDS EQU ---    ; NUMBER OF VALID COMMANDS
000F  110 NEWLN EQU 0FH    ; MASK FOR CHECKING MEMORY ADDR DISPLAY
007F  111 PRTY0 EQU 07FH    ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
13ED  112 REGS EQU DATA+255-18  ; START OF REGISTER SAVE AREA
0002  113 RBR EQU 2       ; MASK TO TEST RECEIVER STATUS
0038  114 RSTU EQU 38H    ; TRANSFER LOCATION FOR RST7 INSTRUCTION
0115  115 ;RTABS EQU ---    ; SIZE OF ENTRY IN RTAB TABLE
001B  116 TERM EQU 1BH    ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
0001  117 TRDY EQU 1       ; MASK TO TEST TRANSMITTER STATUS
00FF  118 UPPER EQU 0FFH   ; DENOTES UPPER HALF OF BYTE IN ICMD

;*****************************************************************
;                       USART INITIALIZATION CODE
;*****************************************************************

THE USART IS ASSUMED TO COME UP IN THE RESET POSITION (THIS
FUNCTION IS TAKEN CARE OF BY THE HARDWARE). THE USART WILL
BE INITIALIZED IN THE SAME WAY FOR EITHER A TTY OR CRT INTERFACE. THE FOLLOWING PARAMETERS ARE USED:

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

;*****************************************************************
;                       USART INITIALIZATION CODE
;*****************************************************************

;*****************************************************************
;                       MONITOR MACROS
;*****************************************************************

MODE INSTRUCTION
       ===- ===- ===-
LOC  OBJ  LINE  SOURCE STATEMENT

156 ;  COMMAND INSTRUCTION
157 ;  =*=*=*=*=*=*=*=*=*=*=*=*=*
158 ;
159 ;  NO HUNT MODE
160 ;  NOT(RTS) FORCED TO 0
161 ;  RECEIVE ENABLED
162 ;  TRANSMIT ENABLED
163 ;
0000 3ECF 164  MVI   A,MODE
0002 D3FB 165  OUT   CNCTL ; OUTPUT MODE SET TO USART
0004 3E27 166  MVI   A,CMD ;
0006 D3FB 167  OUT   CNCTL ; OUTPUT COMMAND WORD TO USART
168 ;
169 ;*****************************************************************
170 ;
171 ;                       RESTART ENTRY POINT
172 ;
173 ;*****************************************************************

0008 22F313 177  SHLD  LSAVE ; SAVE HL REGISTERS
000B E1 178  POP   H ; GET TOP OF STACK ENTRY
000C 22F513 179  SHLD  PSAVE ; ASSUME THIS IS LAST P COUNTER
000F 210000 180  LXI   H,0 ; CLEAR HL
0012 39 181  DAD   SP ; GET STACK POINTER VALUE
0013 22F713 182  SHLD  SSAVE ; SAVE USER`S STACK POINTER
0016 21F313 183  LXI   H,ASAVE+1 ; NEW VALUE FOR STACK POINTER
0019 F9 184  SPLH  ; SET MONITOR STACK POINTER FOR REG SAVE
001A F5 185  PUSH  PSW ; SAVE A AND FLAGS
001B C5 186  PUSH  B ; SAVE B AND C
001C D5 187  PUSH  D ; SAVE D AND E
188 ;
189 ;*****************************************************************
190 ;
191 ;                       PRINT SIGNON MESSAGE
192 ;
193 ;*****************************************************************

001D 219D03 197  LXI   H,SGNON ; GET ADDRESS OF SIGNON MESSAGE
0020 060E 198  MVI   B,LSGNON ; COUNTER FOR CHARACTERS IN MESSAGE
199 MSGL:
0022 4E 200  MOV   C,M ; FETCH NEXT CHAR TO C REG
0023 CDE301 201  CALL  CO ; SEND IT TO THE CONSOLE
0026 23 202  INX   H ; POINT TO NEXT CHARACTER
0027 05 203  DCR   B ; DECREMENT BYTE COUNTER
0028 C22200 204  JNE   MSGL ; RETURN FOR NEXT CHARACTER

205 ;
206 ;
207 ;*****************************************************************
FUNCTION: GETCM
INPUTS: NONE
OUTPUTS: NONE
CALLS: GETCH, ECHO, ERROR
DESTROYS: A, B, C, H, L, F/F'S
DESCRIPTION: GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
CHARACTER TABLE. IF SUCCESSFUL, THE ROUTINE
CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
IS TRANSFERRED TO THIS ROUTINE. IF THE CHARACTER
DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
THE ERROR HANDLER.

GETCM:

ORG RSTU ; ORG TO RST TRANSFER LOCATION

GTC03:

GTC10:

GTC05:
FUNCTION: DCMD

DESCRIPTION: DCMD IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND

DCMD:

005E 0E02  MVI     C,2     ; GET TWO NUMBERS FROM INPUT STREAM
0060 CD5702  CALL    GETNM
0063 D1  POP     D       ; ENDING ADDRESS TO DE
0064 E1  POP     H       ; STARTING ADDRESS TO HL

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   ; ADDRESS IS 2 BYTES LONG
0070 0E20  MVI     C,‘ ’
0072 CDF401  CALL    ECHO    ; USE BLANK AS SEPARATOR
0075 7E  MOV     A,M     ; GET CONTENTS OF NEXT MEMORY LOCATION
0076 CDC302  CALL    NMOUT   ; DISPLAY CONTENTS
0079 CD8601  CALL    BREAK   ; SEE IF USER WANTS OUT
007A 76  TRUE    DCM12    ; IF SO, BRANCH
007B CDC500  JNZ     DCM10    ; NO - NOT AT END OF LINE
007C CD9C02  CALL    HILO    ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
007F CDC302  CALL    NMOUT   ; /GREATER THAN OR EQUAL TO ENDING ADDRESS

DCM10:

0082 D28B00  JNC     DCM15    ; IF NOT, MORE TO DISPLAY
0085 CDDE01  CALL    CROUT   ; CARRIAGE RETURN/LINE FEED TO END LINE
0088 C32B00  JMP     GETCM   ; ALL DONE

DCM15:

008B 23  INX     H       ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
008C 7D  MOV     A,L     ; GET LOW ORDER BITS OF NEW ADDRESS
008D E60F  ANI     NEWLN   ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
0092 C27000  JMP     DCM05    ; YES - START NEW LINE WITH ADDRESS

311 ;*****************************************************************
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>312</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>313</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>314</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; FUNCTION: GCMD</td>
</tr>
<tr>
<td>315</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; INPUTS: NONE</td>
</tr>
<tr>
<td>316</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; OUTPUTS: NONE</td>
</tr>
<tr>
<td>317</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; CALLS: ERROR, GETHX, RSTTF</td>
</tr>
<tr>
<td>318</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; DESTROYS: A, B, C, D, E, H, L, F/F'S</td>
</tr>
<tr>
<td>319</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; DESCRIPTION: GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.</td>
</tr>
<tr>
<td>320</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>321</td>
<td>;</td>
<td>;</td>
<td>;</td>
<td>; GCMD:</td>
</tr>
</tbody>
</table>

0095 CD2202 322 CALL GETHX ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM |
| 323 | ;   | ;    | ;      | ; FALSE GCM05 ; BRANCH IF NO NUMBER PRESENT |
| 324+| ;   | ;    | ;      | JNC GCM05 |
| 0098 D2AA00 324 MOV A, D ; ELSE, GET TERMINATOR |
| 009B 7A 325 MOV A, D ; ELSE, GET TERMINATOR |
| 009C FE0D 326 CPI CR ; SEE IF CARRIAGE RETURN |
| 009E C20002 327 JNZ ERROR ; ERROR IF NOT PROPERLY TERMINATED |
| 00A1 21F513 328 LXI H, PSAVE ; WANT NUMBER TO REPLACE SAVE PGM COUNTER |
| 00A4 71 329 MOV M, C |
| 00A5 23 330 INX H |
| 00A6 70 331 MOV M, B |
| 00A7 C3B000 332 JMP GCM10 |

00AA 7A 333 MOV A, D ; IF NO STARTING ADDRESS, MAKE SURE THAT |
| 00AB FE0D 334 MOV A, D ; IF NO STARTING ADDRESS, MAKE SURE THAT |
| 00AD C20002 335 CPI CR ; /CARRIAGE RETURN TERMINATED COMMAND |
| 00AE 71 336 MOV M, C |
| 00AF 23 337 INX H |
| 00A0 70 338 MOV M, B |
| 00A1 C3B000 339 JMP GCM10 |

00A2 7A 340 GCM05: |
| 00A3 7A 341 MOV A, D ; IF NO STARTING ADDRESS, MAKE SURE THAT |
| 00A4 FE0D 342 MOV A, D ; IF NO STARTING ADDRESS, MAKE SURE THAT |
| 00A5 C20002 343 CPI CR ; /CARRIAGE RETURN TERMINATED COMMAND |
| 00A6 71 344 MOV M, C |
| 00A7 23 345 INX H |
| 00A8 70 346 MOV M, B |
| 00A9 C3B000 347 JMP GCM10 |

00A2 7A 348 ;***************************************************************** |
| 00A3 7A 349 ; FUNCTION: ICM |
| 00A4 7A 350 ; INPUTS: NONE |
| 00A5 FE0D 351 ; OUTPUTS: NONE |
| 00A6 C20002 352 ; CALLS: ERROR, ECHO, GETCH, VALDL, VALDG, CVNVN, STHL, GETNM, CROUTE |
| 00A7 7A 353 ; DESTROYS: A, B, C, D, E, H, L, F/F'S |
| 00A8 7A 354 ; DESCRIPTION: ICM IMPLEMENTS THE INSERT CODE INTO MEMORY (I) COMMAND. |
| 00A9 7A 355 ;         |
| 00AA 7A 356 ;         |
| 00AB 7A 357 ;         |
| 00AC 7A 358 ;         |
| 00AD 7A 359 ;         |
| 00AE 7A 360 ;         |
| 00AF 7A 361 ;         |
| 00A0 7A 362 ;         |
| 00A1 7A 363 ;         |

00B3 0E01 353 MVI C, 1 |
| 00B4 CD5702 354 CALL GETNM ; GET SINGLE NUMBER FROM INPUT STREAM |
| 00B5 3EFF 355 MVI A, UPPER |
| 00B6 32F913 356 STA TEMP ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG |
| 00B7 D1 357 POP D ; ADDRESS OF START TO DE |

00B9 C3B000 358 ICM05: |
| 00BA CD1B02 359 CALL GETCH ; GET A CHARACTER FROM INPUT STREAM |
| 00BB 4F 360 MOV C, A |
| 00BC CD401 361 CALL ECHO ; ECHO IT |
| 00BD 79 362 MOV A, C ; PUT CHARACTER BACK INTO A |
| 00BE FE1B 363 CPI TERM ; SEE IF CHARACTER IS A TERMINATING CHARACTER |
LOC OBJ LINE SOURCE STATEMENT

00C8 CAF400 364 JZ ICM25 ; IF SO, ALL DONE ENTERING CHARACTERS
00CB CD8A03 365 CALL VALDL ; ELSE, SEE IF VALID DELIMITER
00CE DABE00 366 TRUE ICM05 ; IF SO SIMPLY IGNORE THIS CHARACTER
00D1 CD6F03 367+ JC ICM05
00D4 D2E00 368 CALL VALDG ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
00D7 CD4503 369 FALSE ICM20 ; IF NOT, BRANCH TO HANDLE ERROR CONDITION

00DA 4F 370+ MOV C,A ; MOVE RESULT TO C
00DB CD5003 371 CALL CNVBN ; CONVERT DIGIT TO BINARY
00DE 3AF913 372 LDA TEMP ; GET HALF BYTE FLAG
00E1 B7 373 ORA A ; SET F/F'S
00E4 D2EE00 374 JNC ICM10 ; BRANCH IF FLAG SET FOR UPPER
00E7 CDEE00 375 XRI INVRT ; TOGGLE STATE OF FLAG
00E9 C32B00 376 JMP ICM05 ; PROCESS NEXT DIGIT

00EE CD4503 377 CALL STHF0 ; ILLEGAL CHARACTER
00F1 C30D02 378 JMP ICM10 ; BRANCH IF FLAG SET FOR UPPER
00F4 CD4503 379 CALL STHF0 ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
00F7 C3EB00 380 CALL CROUT ; ADD CARRIAGE RETURN
00FA C32B00 381 JMP GETCM ;

392 ;
393 ;*****************************************************************
394 ; FUNCTION: MCMD
395 ; INPUTS: NONE
396 ; OUTPUTS: NONE
397 ; CALLS: GETCM,HILO,GETNM
398 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
399 ; DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.

400 ;
401 MCMD:
00FD 0E03 402 MVI C,3 ;
00FF CD5702 403 CALL GETNM ; GET 3 NUMBERS FROM INPUT STREAM
0102 C1 404 POP B ; DESTINATION ADDRESS TO BC
0103 E1 405 POP H ; ENDING ADDRESS TO HL
0104 D1 406 POP D ; STARTING ADDRESS TO DE
0105 E5 408 PUSH H ; SAVE ENDING ADDRESS
0106 E6 409 MOV H,D
0107 6B 410 MOV L,E ; SOURCE ADDRESS TO HL
0108 7E 411 MOV A,M ; GET SOURCE BYTE
0109 60 412 MOV H,B
010A 69 413 MOV L,C ; DESTINATION ADDRESS TO HL
010B 77 414 MOV M,A ; MOVE BYTE TO DESTINATION
010C 03 415 INX B ; INCREMENT DESTINATION ADDRESS
LOC OBJ LINE SOURCE STATEMENT

010D 78  416  MOV  A,B      ; TEST FOR DESTINATION ADDRESS OVERFLOW
010E B1  417  ORA  C       ; IF SO, CAN TERMINATE COMMAND
0110 CA2B00  418  JZ   GETCM ; IF SO, CAN TERMINATE COMMAND
0112 13  419  INX  D      ; INCREMENT SOURCE ADDRESS
0113 E1  420  POP  H      ; ELSE, GET BACK ENDING ADDRESS
0114 CD9C02  421  CALL  HILO  ; SEE IF ENDING ADDR>=SOURCE ADDR
0117 D22B00  422  FALSE  GETCM ; IF NOT, COMMAND IS DONE
011A C30501  423+ JNC  GETCM

37

**FUNCTION: SCMD**

**INPUTS:** NONE

**OUTPUTS:** NONE

**CALLS:** GETHX, GETCM, NMOUT, ECHO

**DESTROYS:** A, B, C, D, E, H, L, F/F'

**DESCRIPTION:** SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.

SCMD:

011D CD2202  438  CALL  GETHX  ; GET A NUMBER, IF PRESENT, FROM INPUT
0120 C5  439  PUSH  B
0121 E1  440  POP  H      ; GET NUMBER TO HL - DENOTES MEMORY LOCATION

SCM05:

0122 7A  442  MOV  A,D  ; GET TERMINATOR
0123 FE20  443  CPI  ''   ; SEE IF SPACE
0125 CA2D01  444  JZ   SCM10 ; YES - CONTINUE PROCESSING
0126 FE2C  445  CPI  ',' ; ELSE, SEE IF COMMA
0128 C22B00  446  JNZ  GETCM ; IF NO VALUE PRESENT, BRANCH

SCM15:

0129 D23D01  454  CALL  GETHX ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
0130 C71  455  MOV  M,C  ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED

SCM05:

0132 23  457  INX  H      ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
0135 C32201  458  JMP  SCM05

**FUNCTION: XCMD**

**INPUTS:** NONE

**OUTPUTS:** NONE

**CALLS:** GETCH, ECHO, REGDS, GETCM, ERROR, RGADR, NMOUT, CROUT, GETHX
LOC  OBJ  LINE  SOURCE STATEMENT

468 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
469 ; DESCRIPTION: XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X) 470 ; COMMAND.
471 ;
472 XCMD:

0141 CD1B02 473 CALL GETCH ; GET REGISTER IDENTIFIER
0144 4F 474 MOV C,A ;
0145 CDF401 475 CALL ECHO ; ECHO IT
0148 79 476 MOV A,C
0149 FE0D 477 CPI CR
014B C25401 478 JNZ XCM05 ; BRANCH IF NOT CARRIAGE RETURN
014E CDE602 479 CALL REGDS ; ELSE, DISPLAY REGISTER CONTENTS
0151 C32B00 480 JMP GETCM ; THEN TERMINATE COMMAND

481 XCM05:

0154 4F 482 MOV C,A ; GET REGISTER IDENTIFIER TO C
0155 CD1703 483 CALL RGADR ; CONVERT IDENTIFIER INTO RTAB TABLE ADDR
0158 C5 484 PUSH B
0159 E1 485 MOV A,C
015C CDF401 486 CALL ECHO ; ECHO SPACE TO USER
015F 79 488 MOV A,C
0160 32F913 489 STA TEMP ; PUT SPACE INTO TEMP AS DELIMITER

490 XCM10:

0163 3AF913 491 LDA TEMP ; GET TERMINATOR
0166 FE20 492 CPI ' ' ; SEE IF A BLANK
0168 CA7001 493 JZ XCM15 ; YES - GO CHECK POINTER INTO TABLE
016B FE2C 494 CPI ',' ; NO - SEE IF COMMA
016D C22B00 495 JMP GETCM ; NO - MUST BE CARRIAGE RETURN TO END COMMAND
016F 7E 496 MOV A,M
0171 B7 497 ORA A ; SET F/F'S
0172 C27B01 498 JNZ XCM18 ; BRANCH IF NOT AT END OF TABLE
0175 CDE001 499 CALL CROUT ; ELSE, OUTPUT CARRIAGE RETURN LINE FEED
0178 C32B00 500 JMP GETCM ; AND EXIT
017B E5 501 PUSH H ; PUT POINTER ON STACK
017C 5E 502 MOV E,M
017D 1613 503 MVI D,DATA SHR 8 ; FETCH ADDRESS OF SAVE LOCATION FROM TABLE

017F 23 506 INX H ;
0180 46 507 MOV B,M ; FETCH LENGTH FLAG FROM TABLE
0181 D5 508 PUSH D ; SAVE ADDRESS OF SAVE LOCATION
0182 D5 509 PUSH D
0183 E1 510 POP H ; MOVE ADDRESS TO HL
0184 C5 511 PUSH B ; SAVE LENGTH FLAG
0185 7E 512 MOV A,M ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
0186 CDC302 513 CALL NMOUT ; DISPLAY IT
0189 F1 514 POP PSW ; GET BACK LENGTH FLAG
018A F5 515 PUSH PSW ; SAVE IT AGAIN
018B B7 516 ORA A ; SET F/F'S
018C CA9401 517 JZ XCM20 ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
018F 2B 518 DCX H ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
0190 7E 519 MOV A,M
**ISIS-II 8080/8085 MACRO ASSEMBLER, X108**

**SDK80 PAGE 11**

**LOC OBJ LINE SOURCE STATEMENT**

0191 CDC302 520 CALL NMOUT ; DISPLAY THEM
0192 0E2D 521 XCM20:
0193 0E2D 522 MVI C,',''
0194 CDF401 523 CALL ECHO ; USE DASH AS SEPARATOR
0195 CD2202 524 CALL GETHX ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
0196 FALSE XCM30 ; NO - GO CHECK FOR NEXT REGISTER
019C D2B401 526+ JNC XCM30
019F 7A 527 MOV A,D ;
01A0 32F913 528 STA TEMP ; ELSE, SAVE THE TERMINATOR FOR NOW
01A3 E1 529 POP PSW ; GET BACK LENGTH FLAG
01A4 41 530 POP H ; PUT ADDRESS OF SAVE LOCATION INTO HL
01A5 B7 531 ORA A ; SET F/F'S
01A6 CAAB01 532 JNZ XCM25 ; IF 8 BIT REGISTER, BRANCH
01AA 70 533 MOV M,B ; SAVE UPPER 8 BITS
01A9 2B 534 DCX H ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
01AB 71 535 XCM25:
01AC 110300 536 MOV M,C ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
01AD 010300 537 XCM27:
01AC 110300 538 LXI D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
01AF E1 539 POP H ; POINTER INTO REGISTER TABLE RTAB
01B0 19 540 DAD D ; ADD ENTRY SIZE TO POINTER
01B1 C36301 541 JMP XCM10 ; DO NEXT REGISTER
01B2 7A 542 XCM30:
01B4 7A 543 MOV A,D ; GET TERMINATOR
01B5 32F913 544 STA TEMP ; SAVE IN MEMORY
01B9 D1 545 POP D ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
01BA D1 546 POP D ; /OF SAVE LOCATION
01BA C3AC01 547 JMP XCM27 ; GO INCREMENT REGISTER TABLE POINTER

548 ;
549 ;*******************************************
550 ;********************
551 ; UTILITY ROUTINES
552 ;*****************************************************************
553 ; FUNCTION: BREAK
554 ; INPUTS: NONE
556 ; DESTROYS: A,F/F'S
558 ; OUTPUTS: CARRY - 1 IF ESCAPE CHARACTER INPUT
560 ; - 0 IF ANY OTHER CHARACTER OR NO CHARACTER PENDING
562 ; CALLS: NOTHING
564 ; DESCRIPTION: BREAK IS USED TO SENSE AN ESCAPE CHARACTER FROM
566 ; THE USER. IF NO CHARACTER IS PENDING, OR IF THE
568 ; PENDING CHARACTER IS NOT THE ESCAPE, THEN A FAILURE
570 ; RETURN (CARRY=0) IS TAKEN. IN THIS CASE, THE
572 ; PENDING CHARACTER (IF ANY) IS LOST. IF THE PENDING
574 ; CHARACTER IS AN ESCAPE CHARACTER, BREAK TAKES A SUCCESS
576 ; RETURN (CARRY=1).
578 ;
579 ; BREAK:
LOC OBJ LINE SOURCE STATEMENT

01BD DBFB 572 IN CONST ; GET CONSOLE STATUS
01BF E602 573 ANI RBR ; SEE IF CHARACTER PENDING
01C1 CA1802 574 JZ FRET ; NO - TAKE FAILURE RETURN
01C4 DBFA 575 IN CNIN ; YES - PICK UP CHARACTER
01C6 E67F 576 ANI PRTY0 ; STRIP OFF PARITY BIT
01C8 FE1B 577 CPI BRCHR ; SEE IF BREAK CHARACTER
01CA CA4303 578 JZ SRET ; YES - SUCCESS RETURN
01CD C31802 579 JMP FRET ; NO - FAILURE RETURN - CHARACTER LOST

580 ;
581 ;
582 ;*****************************************************************
583 ; FUNCTION: CI
584 ; INPUTS: NONE
585 ; OUTPUTS: A - CHARACTER FROM CONSOLE
586 ; CALLS: NOTHING
587 ; DESTROYS: A,F/F'S
588 ; DESCRIPTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
589 ; CONSOLE AND THEN RETURNS THE CHARACTER, VIA THE A
590 ; REGISTER, TO THE CALLING ROUTINE. THIS ROUTINE
591 ; IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.
592 ;
593 ; CI:
594 ;
595 CI:
596 IN CONST ; GET STATUS OF CONSOLE
01D0 DBFB 597 IN CONST ; GET STATUS OF CONSOLE
01D2 E602 598 ANI RBR ; CHECK FOR RECEIVER BUFFER READY
01D4 CAD001 599 JZ CI ; NOT YET - WAIT
01D7 DBFA 600 IN CNIN ; READY SO GET CHARACTER
01DA 79 601 RET

602 ;
603 ;*****************************************************************
604 ;
605 ; FUNCTION: CNVBN
606 ; INPUTS: C - ASCII CHARACTER '0'-'9' OR 'A'-'F'
607 ; OUTPUTS: A - 0 TO F HEX
608 ; CALLS: NOTHING
609 ; DESTROYS: A,F/F'S
610 ; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
611 ; CHARACTER INTO ITS CORRESPONDING BINARY VALUE. CNVBN
612 ; DOES NOT CHECK THE VALIDITY OF ITS INPUT.
613 ;
614 ;
615 CNVBN:
616 MOV A,C
01DA 79 617 SUI '0' ; SUBTRACT CODE FOR '0' FROM ARGUMENT
01DB E630 618 CPI 10 ; WANT TO TEST FOR RESULT OF 0 TO 9
01DD FE0A 619 RM ; IF SO, THEN ALL DONE
01DF F8 620 SUI 7 ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
01E0 D607 621 RET ; SO RETURN AFTER SUBTRACTING BIAS OF 7
01E2 C9 622 ;
623 ;
FUNCTION: CO
INPUTS: C - CHARACTER TO OUTPUT TO CONSOLE
OUTPUTS: C - CHARACTER OUTPUT TO CONSOLE
CALLS: NOTHING
DESTROYS: A,F/F'S
DESCRIPTION: CO WAITS UNTIL THE CONSOLE IS READY TO ACCEPT A CHARACTER
AND THEN SENDS THE INPUT ARGUMENT TO THE CONSOLE.

CO:

01E3 DBFB
   IN      CONST   ; GET STATUS OF CONSOLE
01E5 E601
   ANI     TRDY    ; SEE IF TRANSMITTER READY
01E7 CAE301
   JZ      CO      ; NO WAIT
01EA 79           639         MOV     A,C     ; ELSE, MOVE CHARACTER TO A REGISTER FOR OUTPUT
01EB D3FA         640         OUT     CNOUT   ; SEND TO CONSOLE
01ED C9           641         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    ; OUTPUT CARRIAGE RETURN TO USER TERMINAL
01F3 C9           658         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
LOC  OBJ  LINE  SOURCE STATEMENT

01F5 3E1B  676  MVI  A,ESC
01F7 B8  677  CMP  B   ; SEE IF ECHOING AN ESCAPE CHARACTER
01FB 0E24  679  MVI  C,‘$’  ; YES - ECHO AS $

680 ECH05:
01FD CDE301  681  CALL  CO  ; DO OUTPUT THROUGH MONITOR
0200 3EO0  682  MVI  A,CR
0202 B8  683  CMP  B   ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RETURN
0203 C20B02  684  JNZ  ECH10  ; NO - NO NEED TO TAKE SPECIAL ACTION
0206 0E2A  685  MVI  C,LF   ; YES - WANT TO ECHO LINE FEED, TOO
0208 CDE301  686  CALL  CO
687 ECH10:
020B 48  688  MOV  C,B  ; RESTORE ARGUMENT
020C C9  689  RET

690 ;
691 ;
692 ;******************************************************************************
693 ; FUNCTION: ERROR
694 ; INPUTS: NONE
695 ; OUTPUTS: NONE
696 ; CALLS: ECHO,CROUT,GETCM
697 ; DESTROYS: A,B,C,F/F’S
698 ; DESCRIPTION: ERROR Printer THE ERROR CHARACTER (CURRENTLY A NUMBER SIGN)
699 ; ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
700 ; AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.
701 ;
702 ;
703 ;
704 ERROR:
020D 0E2A  705  MVI  C,‘*’
020F CDF401  706  CALL  ECHO  ; SEND # TO CONSOLE
707 EXIT:
0212 CDEE01  708  CALL  CROUT  ; SKIP TO BEGINNING OF NEXT LINE
0215 C32B00  709  JMP  GETCM  ; TRY AGAIN FOR ANOTHER COMMAND
710 ;
711 ;
712 ;******************************************************************************
713 ;
714 ;
715 ; FUNCTION: FRET
716 ; INPUTS: NONE
717 ; OUTPUTS: CARRY - ALWAYS 0
718 ; CALLS: NOTHING
719 ; DESTROYS: CARRY
720 ; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
721 ; INDICATE FAILURE ON RETURN. FRET SETS THE CARRY
722 ; FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
723 ; CALLER OF THE ROUTINE INVOKING FRET.
724 ;
725 FRET:
0218 37  726  STC  ; FIRST SET CARRY TRUE
0219 3F  727  CMC  ; THEN COMPLEMENT IT TO MAKE IT FALSE
LOC OBJ LINE SOURCE STATEMENT

021A C9 728    RET ; RETURN APPROPRIATELY
729 ;
730 ;
731 ;**********************************************************************
732 ;
733 ; FUNCTION: GETCH
734 ; INPUTS: NONE
735 ; OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
736 ; CALLS: CI
737 ; DESTROYS: A,C,F/F'S
738 ; DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
739 ; TO THE CALLING PROGRAM.
740 ;
741 ; GETCH:
021B CDD001 742 CALL CI ; GET CHARACTER FROM TERMINAL
021E E67F 743 ANI PRTY0 ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
0220 4F 744 MOV C,A ; PUT VALUE IN C REGISTER FOR RETURN
0221 C9 745 RET
746 ;
747 ;
748 ;
749 ;**********************************************************************
750 ; FUNCTION: GETHX
751 ; INPUTS: NONE
752 ; OUTPUTS: BC - 16 BIT INTEGER
753 ; D - CHARACTER WHICH TERMINATED THE INTEGER
754 ; CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
755 ; 0 IF FIRST CHARACTER IS DELIMITER
756 ; CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR
757 ; DESTROYS: A,B,C,D,E,F/F'S
758 ; DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
759 ; STREAM AND RETURNS THEIR VALUE AS A 16 BIT INTEGER.
760 ; IF MORE THAN 4 HEX DIGITS ARE ENTERED,
761 ; ONLY THE LAST 4 ARE USED. THE NUMBER TERMINATES WHEN
762 ; A VALID DELIMITER IS ENCOUNTERED. THE DELIMITER IS
763 ; ALSO RETURNED AS AN OUTPUT OF THE FUNCTION. ILLEGAL
764 ; CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
765 ; ERROR INDICATION. IF THE FIRST (VALID) CHARACTER
766 ; ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
767 ; GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
768 ; OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
769 ; OF BC ARE UNDEFINED.
770 ;
771 ; GETHX:
0222 E5 772 PUSH H ; SAVE HL
0223 210000 773 LXI H,0 ; INITIALIZE RESULT
0226 1B00 774 MVI E,0 ; INITIALIZE DIGIT FLAG TO FALSE
0227 777 GHX05:
0228 CD1B02 778 CALL GETCH ; GET A CHARACTER
022B 4F 779 MOV C,A ;
**FUNCTION: GETNM**

**INPUTS:** C - COUNT OF NUMBERS TO FIND IN INPUT STREAM

**OUTPUTS:** TOP OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON TOP OF STACK)

**CALLS:** GETHX, HILO, ERROR

**DESTROYS:** A, B, C, D, E, H, L, F/F'S

**DESCRIPTION:** GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1 AND 3, INCLUSIVE, IN THE INPUT STREAM AND RETURNS THEIR VALUES ON THE STACK. IF 2 OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND SECOND NUMBERS WILL BE SET EQUAL. THE LAST NUMBER REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN, OR AN ERROR INDICATION WILL RESULT.

**GETNM:**

0257 2E03 827  MVI  L,3  ; PUT MAXIMUM ARGUMENT COUNT INTO L

0259 79   828  MOV  A,C  ; GET THE ACTUAL ARGUMENT COUNT

025A E603 829  ANI  3   ; FORCE TO MAXIMUM OF 3

025C C8 830  RZ   ; IF 0, DON'T BOTHER TO DO ANYTHING

025D 67 831  MOV  H,A  ; ELSE, PUT ACTUAL COUNT INTO H
LOC   OBJ      LINE    SOURCE STATEMENT

025E CD2202  832      GNM05:
0261 D20D02  833      CALL GETHX ; GET A NUMBER FROM INPUT STREAM
0264 C5     834      FALSE ERROR ; ERROR IF NOT THERE - TOO FEW NUMBERS
0265 2D     835+     JNC ERROR
0269 CA7302  836      CALL ERROR ; ERROR IF TOO FEW NUMBERS
0270 7A     837      MOV A,D ; ELSE, SAVE NUMBER ON STACK
0273 2D     838      DCR L ; DECREMENT MAXIMUM ARGUMENT COUNT
0276 CA8602  839      JZ GNM10  ; BRANCH IF NO MORE NUMBERS WANTED
0279 7A     840      MOV A,D ; ELSE, GET NUMBER TERMINATOR TO A
027C 2D     841      DCR H ; DECREMENT ACTUAL ARGUMENT COUNT
027E C5     842      JNZ GNM20 ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFFH
0281 C5     843      JMP GNM05 ; ELSE, PROCESS NEXT NUMBER
0284 2D     844      GNM10:
0287 C1     845      MOV A,D ; WHEN COUNT 0, CHECK LAST TERMINATOR
028A C5     846      CPI CR
028D C9C02  847      JNZ ERROR ; ERROR IF NOT CARRIAGE RETURN
028F C35E02 848      JNC GNM25 ; ELSE, MAKE SECOND EQUAL TO THE FIRST
0292 C35E02 849      MOV D,H
0295 3D     850      XTHL ; PUT FIRST ON STACK - GET RETURN ADDR
0298 C39502  851      JMP GNM30 ; TRY AGAIN
029B 3D     852      DCR A ; DECREMENT RESIDUAL COUNT
029E E3     853      XTHL ; REPLACE TOP RESULT WITH RETURN ADDR
02A1 3D     854      RM  ; IF NEGATIVE, PROPER RESULTS ON STACK
02A4 E3     855      POP H ; ELSE, GET RETURN ADDR
02A7 E3     856      XTHL ; REPLACE TOP RESULT WITH RETURN ADDR
02A9 C39502  857      JMP GNM30 ; TRY AGAIN
02A9 D29102 858      JNC GNM25 ; NO - BRANCH
02AC C28102  859      POP H ; ELSE, GET RETURN ADDR
02AF C28102 860      XTHL ; REPLACE TOP RESULT WITH RETURN ADDR
02B2 C35E02  861      STACK ;....FUNCTION: HILO
02B5 E3     862+     JMP GNM30 ; TRY AGAIN
02B8 C28102  863      POP D,H
02B9 2D     864      MOV E,L ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFFH
02B2 C29102  865      GNM20:
02B5 C29102  866      POP B ; GET THE 3 ARGUMENTS OUT
02BF C29102  867      POP D
02C4 CD9C02  868      POP H
02C9 C39502  869      POP B ; GET THE 3 ARGUMENTS OUT
02CB C39502  870      POP H ; GET THE 3 ARGUMENTS OUT
02CE C39502  871      POP A ; GET THE 3 ARGUMENTS OUT
02D0 C39502  872      POP D ; GET THE 3 ARGUMENTS OUT
02D5 C39502  873      POP H ; GET THE 3 ARGUMENTS OUT
02DA C39502  874      POP A ; GET THE 3 ARGUMENTS OUT
02DF C39502  875      POP D,H
02E4 C39502  876 ;
02E9 E3     877 ;
02EB C39502  878 ;*****************************************************************
02EC C39502  879 ;FUNCTION: HILO
02FC C39502  880 ;INPUTS: DE - 16 BIT INTEGER
02FF C39502  881 ;HL - 16 BIT INTEGER

45
LOC OBJ LINE SOURCE STATEMENT

884 ; OUTPUTS: CARRY - 0 IF HL<DE
885 ; - 1 IF HL>=DE
886 ; CALLS: NOTHING
887 ; DESTROYS: A,F/F'S
888 ; DESCRIPTION: HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE. THE
889 ; INTEGERS ARE TREATED AS UNSIGNED NUMBERS. THE CARRY
890 ; BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.
891 ;
892 HILO:
893 029C C5 893 PUSH B ; SAVE BC
894 029D 47 894 MOV B,A ; SAVE A REGISTER
895 029E 23 895 INX H ; INCREMENT HL BY 1
896 029F 7C 896 MOV A,H ; WANT TO TEST FOR 0 RESULT AFTER
897 02A0 B5 897 ORA L ; /INCREMENTING
898 02A1 CABS02 898 JZ HIL05 ; WE`RE AUTOMATICALLY DONE IF IT IS
899 02A2 23 899 INX H ; INCREMENT HL BY 1
900 02A5 7C 900 MOV A,H ; WANT TO TEST FOR 0 RESULT AFTER
901 02A6 B5 901 ORA L ; /INCREMENTING
902 02A7 CABS02 902 JZ HIL05 ; IF SO, HL MUST HAVE CONTAINED 0FFFFH
903 02A8 E1 903 POP H ; IF NOT, RESTORE ORIGINAL HL
904 02A9 D5 904 PUSH D ; SAVE DE
905 02AA 3EFF 905 MVI A,0FFH ; Want TO TAKE 2`S COMPLEMENT OF DE CONTENTS
906 02AB AA 906 XRA D ;
907 02AF 57 907 MOV D,A ;
908 02B0 3EFF 908 MVI A,0FFH ;
909 02B2 AB 909 XRA E ;
910 02B3 5F 910 MOV E,A ;
911 02B4 13 911 INX D ; 2`S COMPLEMENT ODE TO DE
912 02B5 7D 912 MOV A,L ;
913 02B6 83 913 ADD E ; ADD HL AND DE
914 02B7 7C 914 MOV A,H ;
915 02B8 8A 915 ADC D ; THIS OPERATION SETS CARRY PROPERLY
916 02B9 D1 916 POP D ; RESTORE ORIGINAL DE CONTENTS
917 02BA 78 917 MOV A,B ; RESTORE ORIGINAL CONTENTS OF A
918 02BB C1 918 POP B ; RESTORE ORIGINAL CONTENTS OF BC
919 02BC C9 919 RET ; RETURN WITH CARRY SET AS REQUIRED
920 HIL05:
921 02BD E1 921 POP H ; IF HL CONTAINS 0FFFFH, THEN CARRY CAN
922 02BE 78 922 MOV A,B ; /Only BE WSET TO 1
923 02BF C1 923 POP B ; RESTORE ORIGINAL CONTENTS OF REGISTERS
924 02C0 C34303 924 JMP SRET ; SET CARRY AND RETURN
925 ;
926 ;**********************************************************
928 ;
929 ;
930 ; FUNCTION: NMOUT
931 ; INPUTS: A - 8 BIT INTEGER
932 ; OUTPUTS: NONE
933 ; CALLS: ECHO,PRVAL
934 ; DESTROYS: A,B,C,F/F'S
935 ; 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.

Push H; save HL - destroyed by PRVAL

Push PSW; save argument

RRC

RRC

RRC

RRC

RRC

RRC

RRC

RRC

RRC

RRC

Push PSW; get back argument

Pop PSW; mask out upper 4 bits - want 1 hex char

Pop H; restore saved value of HL

Ret

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

Function: PRVAL

Inputs: A - integer, range 0 to F

Outputs: A - 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-A-F. PRVAL does not check the validity of its input argument.

LXI H,DIGTB; address of table

MVI B,0; clear high order bits of BC

Add digit value to HL address

MOV C,M; fetch character from memory

Ret

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

Function: REGDS

Inputs: None

Outputs: None

Calls: ECHO,NMOUT,ERROR,CROUT
**REGDS**:

```
02E6 21CF03 02E9 4E 02EA 79 02EB B7 02EC C2F302 02EF CDEE01 02F2 C9
02F3 CDF401 02F6 0E3D 02F8 CDF401 02FB 23 0300 CDC402
```

1004 REG10:
```
0300 1A 0300 CDC302 0304 7E
```

1015 REG15:
```
0304 7E 030A 1B
```

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

**FUNCTION: RGADR**

```
030E 0E20 0310 CDF401 0313 23 0314 C3E902
```

1027 ;***************************************************************************

**DESCRIPTION: RGADR TAKES A SINGLE CHARACTER AS INPUT. THIS CHARACTER

1028 ; DENOTES A REGISTER. RGADR SEARCHES THE TABLE RTAB

1029 ; FOR A MATCH ON THE INPUT ARGUMENT. IF ONE OCCURS,

1030 ; RGADR RETURNS THE ADDRESS OF THE ADDRESS OF THE

1031 ; SAVE LOCATION CORRESPONDING TO THE REGISTER. THIS
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1040</td>
<td>;</td>
<td>1045</td>
<td>ADDRESS POINTS INTO RTAB. IF NO MATCH OCCURS, THEN</td>
</tr>
<tr>
<td>1041</td>
<td>;</td>
<td>1046</td>
<td>THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS</td>
</tr>
<tr>
<td>1042</td>
<td>;</td>
<td>1047</td>
<td>PASSED TO THE ERROR ROUTINE.</td>
</tr>
<tr>
<td>1043</td>
<td>;</td>
<td>1048</td>
<td>RGADR:</td>
</tr>
<tr>
<td>0317</td>
<td>21CF03</td>
<td>1049</td>
<td>LXI H,RTAB ; HL GETS ADDRESS OF TABLE START</td>
</tr>
<tr>
<td>031A</td>
<td>110300</td>
<td>1050</td>
<td>LXI D,RTABS ; DE GET SIZE OF A TABLE ENTRY</td>
</tr>
<tr>
<td>031D</td>
<td>7E</td>
<td>1051</td>
<td>MOV A,M ; GET REGISTER IDENTIFIER</td>
</tr>
<tr>
<td>031E</td>
<td>B7</td>
<td>1052</td>
<td>ORA A ; CHECK FOR TABLE END (IDENTIFIER IS 0)</td>
</tr>
<tr>
<td>031F</td>
<td>CA0002</td>
<td>1053</td>
<td>JZ ERROR ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL</td>
</tr>
<tr>
<td>0322</td>
<td>B9</td>
<td>1054</td>
<td>CMP C ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT</td>
</tr>
<tr>
<td>0323</td>
<td>CA2A03</td>
<td>1055</td>
<td>JZ RGA10 ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR</td>
</tr>
<tr>
<td>0326</td>
<td>19</td>
<td></td>
<td>DAD D ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY</td>
</tr>
<tr>
<td>0327</td>
<td>C31D03</td>
<td>1057</td>
<td>JMP RGA05 ; TRY AGAIN</td>
</tr>
<tr>
<td>032A</td>
<td>23</td>
<td>1058</td>
<td>INX H ; IF A MATCH, INCREMENT TABLE POINTER TO</td>
</tr>
<tr>
<td>032B</td>
<td>44</td>
<td>1059</td>
<td>MOV C,L ; RETURN THIS VALUE</td>
</tr>
<tr>
<td>032C</td>
<td>C9</td>
<td>1060</td>
<td>RET</td>
</tr>
</tbody>
</table>

**RSTTF:***

**FUNCTION:** RSTTF  
**INPUTS:** NONE  
**OUTPUTS:** NONE  
**CALLS:** NOTHING  
**DESTROYS:** A,B,C,D,E,H,L,F/F'S  
**DESCRIPTION:** RSTTF RESTORES ALL CPU REGISTER, FLIP/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.  

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>032E</td>
<td>F3</td>
<td>1078</td>
<td>DI ; DISABLE INTERRUPTS WHILE RESTORING THINGS</td>
</tr>
<tr>
<td>032F</td>
<td>21ED13</td>
<td>1079</td>
<td>LXI H,MSTAK ; SET MONITOR STACK POINTER TO START OF STACK</td>
</tr>
<tr>
<td>0332</td>
<td>F9</td>
<td>1080</td>
<td>SPHL ;</td>
</tr>
<tr>
<td>0333</td>
<td>D1</td>
<td>1081</td>
<td>POP D ; START ALSO END OF REGISTER SAVE AREA</td>
</tr>
<tr>
<td>0334</td>
<td>C1</td>
<td>1082</td>
<td>POP B ;</td>
</tr>
<tr>
<td>0335</td>
<td>F1</td>
<td>1083</td>
<td>POP PSW ;</td>
</tr>
<tr>
<td>0336</td>
<td>2AF713</td>
<td>1084</td>
<td>LHLD SSAVE ; RESTORE USER STACK POINTER</td>
</tr>
<tr>
<td>0339</td>
<td>F9</td>
<td>1085</td>
<td>SPHL ;</td>
</tr>
<tr>
<td>033A</td>
<td>2AF513</td>
<td>1086</td>
<td>LHLD PSAVE ;</td>
</tr>
<tr>
<td>033D</td>
<td>E5</td>
<td>1087</td>
<td>PUSH H ; PUT USER RETURN ADDRESS ON USER STACK</td>
</tr>
<tr>
<td>033E</td>
<td>2AF313</td>
<td>1088</td>
<td>LHLD LSAVE ; RESTORE HL REGISTERS</td>
</tr>
<tr>
<td>0341</td>
<td>FB</td>
<td>1089</td>
<td>EI ; ENABLE INTERRUPTS NOW</td>
</tr>
<tr>
<td>0342</td>
<td>C9</td>
<td>1090</td>
<td>RET ; JUMP TO RESTORED PC LOCATION</td>
</tr>
<tr>
<td>1091</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
<td>LINE</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>1092</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1093</td>
<td></td>
<td></td>
<td>;*******************************************************************</td>
</tr>
<tr>
<td>1094</td>
<td></td>
<td></td>
<td>1095 ;</td>
</tr>
<tr>
<td>1096</td>
<td></td>
<td></td>
<td>; FUNCTION: SRET</td>
</tr>
<tr>
<td>1097</td>
<td></td>
<td></td>
<td>; INPUTS: NONE</td>
</tr>
<tr>
<td>1098</td>
<td></td>
<td></td>
<td>; OUTPUTS: CARRY = 1</td>
</tr>
<tr>
<td>1099</td>
<td></td>
<td></td>
<td>; CALLS: NOTHING</td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td>; DESTROYS: CARRY</td>
</tr>
<tr>
<td>1101</td>
<td></td>
<td></td>
<td>; DESCRIPTION: SRET IS JUMPED TO BY ROUTINES WISHING TO RETURN SUCCESS.</td>
</tr>
<tr>
<td>1102</td>
<td></td>
<td></td>
<td>1103 ;</td>
</tr>
<tr>
<td>1104</td>
<td></td>
<td></td>
<td>1105 ;</td>
</tr>
<tr>
<td>1106</td>
<td>0343</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>1107</td>
<td>0344</td>
<td>C9</td>
<td></td>
</tr>
<tr>
<td>1108</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1109</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td>;*******************************************************************</td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1112</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1113</td>
<td></td>
<td></td>
<td>; FUNCTION: STHF0</td>
</tr>
<tr>
<td>1114</td>
<td></td>
<td></td>
<td>; INPUTS: DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO</td>
</tr>
<tr>
<td>1115</td>
<td></td>
<td></td>
<td>; OUTPUTS: NONE</td>
</tr>
<tr>
<td>1116</td>
<td></td>
<td></td>
<td>; CALLS: NOTHING</td>
</tr>
<tr>
<td>1117</td>
<td></td>
<td></td>
<td>; DESTROYS: A,B,C,H,L,F/F'S</td>
</tr>
<tr>
<td>1118</td>
<td></td>
<td></td>
<td>; DESCRIPTION: STHF0 CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF</td>
</tr>
<tr>
<td>1119</td>
<td></td>
<td></td>
<td>1120 ;</td>
</tr>
<tr>
<td>1121</td>
<td></td>
<td></td>
<td>1122 ;</td>
</tr>
<tr>
<td>1123</td>
<td></td>
<td></td>
<td>1124 ;</td>
</tr>
<tr>
<td>1124</td>
<td>0345</td>
<td>3AF9</td>
<td></td>
</tr>
<tr>
<td>1125</td>
<td>0348</td>
<td>B7</td>
<td></td>
</tr>
<tr>
<td>1126</td>
<td>0349</td>
<td>C0</td>
<td></td>
</tr>
<tr>
<td>1127</td>
<td>034A</td>
<td>0E00</td>
<td></td>
</tr>
<tr>
<td>1128</td>
<td>034C</td>
<td>CD50</td>
<td></td>
</tr>
<tr>
<td>1129</td>
<td>034F</td>
<td>C9</td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1131</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1132</td>
<td></td>
<td></td>
<td>;*******************************************************************</td>
</tr>
<tr>
<td>1133</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1134</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1135</td>
<td></td>
<td></td>
<td>; FUNCTION: STHLF</td>
</tr>
<tr>
<td>1136</td>
<td></td>
<td></td>
<td>; INPUTS: C - 4 BIT VALUE TO BE STORED IN HALF BYTE</td>
</tr>
<tr>
<td>1137</td>
<td></td>
<td></td>
<td>; DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO</td>
</tr>
<tr>
<td>1138</td>
<td></td>
<td></td>
<td>; OUTPUTS: NONE</td>
</tr>
<tr>
<td>1139</td>
<td></td>
<td></td>
<td>; CALLS: NOTHING</td>
</tr>
<tr>
<td>1140</td>
<td></td>
<td></td>
<td>; DESTROYS: A,B,C,H,L,F/F'S</td>
</tr>
<tr>
<td>1141</td>
<td></td>
<td></td>
<td>; DESCRIPTION: STHLF TAKES THE 4 BIT VALUE IN C AND STORES IT IN</td>
</tr>
<tr>
<td>1142</td>
<td></td>
<td></td>
<td>1143 ;</td>
</tr>
<tr>
<td>1143</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOC OBJ LINE SOURCE STATEMENT

1144 ; BY THE VALUE OF THE FLAG IN TEMP. STHLF ASSUMES
1145 ; THAT THIS FLAG HAS BEEN PREVIOUSLY SET
1146 ; (NOMINALLY BY ICMD).
1147 ;
1148 STHLF:
0350 D5 1149 PUSH D
0351 E1 1150 POP H ; MOVE ADDRESS OF BYTE INTO HL
0352 79 1151 MOV A,C ; GET VALUE
0353 E60F 1152 ANI 0FH ; FORCE TO 4 BIT LENGTH
0355 4F 1153 MOV C,A ; PUT VALUE BACK
0356 3AF913 1154 LDA TEMP ; GET HALF BYTE FLAG
0359 B7 1155 ORA A ; CHECK FOR LOWER HALF
035A C26303 1156 JNZ STH05 ; BRANCH IF NOT
035D 7E 1157 MOV A,M ; ELSE, GET BYTE
035E E6F0 1158 ANI 0F0H ; CLEAR LOWER 4 BITS
0360 B1 1159 ORA C ; OR IN VALUE
0361 77 1160 MOV M,A ; PUT BYTE BACK
0362 C9 1161 RET

1162 STH05:
0363 7E 1163 MOV A,M ; IF UPPER HALF, GET BYTE
0364 E60F 1164 ANI 0FH ; CLEAR UPPER 4 BITS
0365 47 1165 MOV B,A ; SAVE BYTE IN B
0366 79 1166 MOV A,C ; GET VALUE
0368 0F 1167 RRC
0369 0F 1168 RRC
036A 0F 1169 RRC
036B 0F 1170 RRC ; ALIGN TO UPPER 4 BITS
036C B0 1171 ORA B ; OR IN ORIGINAL LOWER 4 BITS
036D 77 1172 MOV M,A ; PUT NEW CONFIGURATION BACK
036E C9 1173 RET

1174 ;
1175 ;
1176 ;*****************************************************************
1177 ; FUNCTION: VALDG
1178 ; INPUTS: C - ASCII CHARACTER
1179 ; OUTPUTS: CARRY - 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
1180 ; - 0 OTHERWISE
1181 ; CALLS: NOTHING
1182 ; DESTROYS: A,F/F'S
1183 ; DESCRIPTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
1184 ; AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
1185 ; (0-9,A-F), AND FAILURE OTHERWISE.
1186 ;
1187 ; VALDG:
036F 79 1188 MOV A,C
0370 FE30 1189 CPI '0' ; TEST CHARACTER AGAINST '0'
0372 FA1802 1190 JM FRET ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
0375 FE39 1191 CPI '9' ; ELSE, SEE IF IN RANGE '0'-'9'
0377 FA4303 1192 JM SRET ; CODE BETWEEN '0' AND '9'
037A CA4303 1193 JZ SRET ; CODE EQUAL '9'

51
LOC OBJ LINE SOURCE STATEMENT

037D FE41  1196  CPI  'A' ; NOT A DIGIT - TRY FOR A LETTER
037F FA1802  1197  JM  FRET ; NO - CODE BETWEEN '9' AND 'A'
0382 FE47  1198  CPI  'G'
0384 F21802  1199  JP  FRET ; NO - CODE GREATER THAN 'F'
0387 C34303  1200  JMP  SRET ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE
038A 79  1201 ;
038B FE2C  1202         CPI  ',' ; CHECK FOR COMMA
038D CA4303  1203  JZ  SRET
0390 FE0D  1204  CPI  CR ; CHECK FOR CARRIAGE RETURN
0392 CA4303  1205  JZ  SRET
0395 FE20  1206  CPI  ' ' ; CHECK FOR SPACE
0397 CA4303  1207  JZ  SRET
039A C31802  1208  JMP  FRET ; ERROR IF NONE OF THE ABOVE

039D 0D  1209 ;
039E 0A  1210 ;
039F 4D43532D  1211 ;
03A3 3830204B  1212 ;
03A7 4954  1213 ;
03A9 0D  1214 ;
03AA 0A  1215 ;
000E  1216 ;
1236 LSGNON EQU $-SGNON ; LENGTH OF SIGNON MESSAGE
1239 ;
1238 CADR: ; TABLE OF ADDRESSES OF COMMAND ROUTINES
03AB 0000  1239  DW  0 ; DUMMY
03AD 4101  1240  DW  XCMD
03AF 1D01  1241  DW  SCMD

FUNCTION: VALDL
INPUTS: C - CHARACTER
OUTPUTS: CARRY - 1 IF INPUT ARGUMENT VALID DELIMITER
         0 OTHERWISE
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  1217  MOV  A,C
038B FE2C  1218  CPI  ',' ; CHECK FOR COMMA
038D CA4303  1219  JZ  SRET
0390 FE0D  1220  CPI  CR ; CHECK FOR CARRIAGE RETURN
0392 CA4303  1221  JZ  SRET
0395 FE20  1222  CPI  ' ' ; CHECK FOR SPACE
0397 CA4303  1223  JZ  SRET
039A C31802  1224  JMP  FRET ; ERROR IF NONE OF THE ABOVE

1225 ;
1226 ;
1227 ;*****************************************************************
1228 ;                       MONITOR TABLES
1229 ;*****************************************************************
1230 ;
1231 ;*****************************************************************
1232 ;
1233 ;
1234 SGNON:                  ; SIGNON MESSAGE
1235    DB  CR,LF,'MCS-80 KIT',CR,LF
039D 0D  1236  DB  CR,LF,'MCS-80 KIT',CR,LF
039E 0A  1237 ;
039F 4D43532D  1238 ;
03A3 3830204B  1239 ;
03A7 4954  1240 ;
03A9 0D  1241 ;
03AA 0A  1242 ;
000E  1243 ;
1236 LSGNON EQU $-SGNON ; LENGTH OF SIGNON MESSAGE
1239 ;
1238 CADR: ; TABLE OF ADDRESSES OF COMMAND ROUTINES
03AB 0000  1239  DW  0 ; DUMMY
03AD 4101  1240  DW  XCMD
03AF 1D01  1241  DW  SCMD
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>03B1</td>
<td>FD00</td>
<td>1242</td>
<td>DW</td>
<td>MCMD</td>
</tr>
<tr>
<td>03B3</td>
<td>B300</td>
<td>1243</td>
<td>DW</td>
<td>ICMD</td>
</tr>
<tr>
<td>03B5</td>
<td>9500</td>
<td>1244</td>
<td>DW</td>
<td>GCMD</td>
</tr>
<tr>
<td>03B7</td>
<td>5E00</td>
<td>1245</td>
<td>DW</td>
<td>DCMD</td>
</tr>
<tr>
<td>1246</td>
<td>;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1247</td>
<td>CTAB:</td>
<td>;</td>
<td></td>
<td>TABLE OF VALID COMMAND CHARACTERS</td>
</tr>
<tr>
<td>03B9</td>
<td>44</td>
<td>1248</td>
<td>DB</td>
<td>'D'</td>
</tr>
<tr>
<td>03BA</td>
<td>47</td>
<td>1249</td>
<td>DB</td>
<td>'G'</td>
</tr>
<tr>
<td>03BB</td>
<td>49</td>
<td>1250</td>
<td>DB</td>
<td>'I'</td>
</tr>
<tr>
<td>03BC</td>
<td>4D</td>
<td>1251</td>
<td>DB</td>
<td>'M'</td>
</tr>
<tr>
<td>03BD</td>
<td>53</td>
<td>1252</td>
<td>DB</td>
<td>'S'</td>
</tr>
<tr>
<td>03BE</td>
<td>58</td>
<td>1253</td>
<td>DB</td>
<td>'X'</td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1254</td>
<td>NCMD</td>
<td>EQU</td>
<td>$-CTAB</td>
<td>NUMBER OF VALID COMMANDS</td>
</tr>
<tr>
<td>1255</td>
<td>;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1256</td>
<td>DIGTB:</td>
<td>;</td>
<td></td>
<td>TABLE OF REGISTER INFORMATION</td>
</tr>
<tr>
<td>03BF</td>
<td>30</td>
<td>1257</td>
<td>DB</td>
<td>'0'</td>
</tr>
<tr>
<td>03C0</td>
<td>31</td>
<td>1258</td>
<td>DB</td>
<td>'1'</td>
</tr>
<tr>
<td>03C1</td>
<td>32</td>
<td>1259</td>
<td>DB</td>
<td>'2'</td>
</tr>
<tr>
<td>03C2</td>
<td>33</td>
<td>1260</td>
<td>DB</td>
<td>'3'</td>
</tr>
<tr>
<td>03C3</td>
<td>34</td>
<td>1261</td>
<td>DB</td>
<td>'4'</td>
</tr>
<tr>
<td>03C4</td>
<td>35</td>
<td>1262</td>
<td>DB</td>
<td>'5'</td>
</tr>
<tr>
<td>03C5</td>
<td>36</td>
<td>1263</td>
<td>DB</td>
<td>'6'</td>
</tr>
<tr>
<td>03C6</td>
<td>37</td>
<td>1264</td>
<td>DB</td>
<td>'7'</td>
</tr>
<tr>
<td>03C7</td>
<td>38</td>
<td>1265</td>
<td>DB</td>
<td>'8'</td>
</tr>
<tr>
<td>03C8</td>
<td>39</td>
<td>1266</td>
<td>DB</td>
<td>'9'</td>
</tr>
<tr>
<td>03C9</td>
<td>41</td>
<td>1267</td>
<td>DB</td>
<td>'A'</td>
</tr>
<tr>
<td>03CA</td>
<td>42</td>
<td>1268</td>
<td>DB</td>
<td>'B'</td>
</tr>
<tr>
<td>03CB</td>
<td>43</td>
<td>1269</td>
<td>DB</td>
<td>'C'</td>
</tr>
<tr>
<td>03CC</td>
<td>44</td>
<td>1270</td>
<td>DB</td>
<td>'D'</td>
</tr>
<tr>
<td>03CD</td>
<td>45</td>
<td>1271</td>
<td>DB</td>
<td>'E'</td>
</tr>
<tr>
<td>03CE</td>
<td>46</td>
<td>1272</td>
<td>DB</td>
<td>'F'</td>
</tr>
<tr>
<td>1273</td>
<td>;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1274</td>
<td>RTAB:</td>
<td>;</td>
<td></td>
<td>TABLE OF REGISTER INFORMATION</td>
</tr>
<tr>
<td>03CF</td>
<td>41</td>
<td>1275</td>
<td>DB</td>
<td>'A'</td>
</tr>
<tr>
<td>03D0</td>
<td>F2</td>
<td>1276</td>
<td>DB</td>
<td>ASAVE AND OFFH ; ADDRESS OF REGISTER SAVE LOCATION</td>
</tr>
<tr>
<td>03D1</td>
<td>00</td>
<td>1277</td>
<td>DB</td>
<td>0 ; LENGTH FLAG - 0=8 BITS, 1=16 BITS</td>
</tr>
<tr>
<td>0003</td>
<td>1278</td>
<td>RTABS</td>
<td>EQU</td>
<td>$-RTAB ; SIZE OF AN ENTRY IN THIS TABLE</td>
</tr>
<tr>
<td>03D2</td>
<td>42</td>
<td>1279</td>
<td>DB</td>
<td>'B'</td>
</tr>
<tr>
<td>03D3</td>
<td>F0</td>
<td>1280</td>
<td>DB</td>
<td>BSAVE AND OFFH</td>
</tr>
<tr>
<td>03D4</td>
<td>00</td>
<td>1281</td>
<td>DB</td>
<td>0</td>
</tr>
<tr>
<td>03D5</td>
<td>43</td>
<td>1282</td>
<td>DB</td>
<td>'C'</td>
</tr>
<tr>
<td>03D6</td>
<td>EF</td>
<td>1283</td>
<td>DB</td>
<td>CSAVE AND OFFH</td>
</tr>
<tr>
<td>03D7</td>
<td>00</td>
<td>1284</td>
<td>DB</td>
<td>0</td>
</tr>
<tr>
<td>03D8</td>
<td>44</td>
<td>1285</td>
<td>DB</td>
<td>'D'</td>
</tr>
<tr>
<td>03D9</td>
<td>EE</td>
<td>1286</td>
<td>DB</td>
<td>DSAVE AND OFFH</td>
</tr>
<tr>
<td>03DA</td>
<td>00</td>
<td>1287</td>
<td>DB</td>
<td>0</td>
</tr>
<tr>
<td>03DB</td>
<td>45</td>
<td>1288</td>
<td>DB</td>
<td>'E'</td>
</tr>
<tr>
<td>03DC</td>
<td>ED</td>
<td>1289</td>
<td>DB</td>
<td>ESAVE AND OFFH</td>
</tr>
<tr>
<td>03DD</td>
<td>00</td>
<td>1290</td>
<td>DB</td>
<td>0</td>
</tr>
<tr>
<td>03DE</td>
<td>46</td>
<td>1291</td>
<td>DB</td>
<td>'F'</td>
</tr>
<tr>
<td>03DF</td>
<td>F1</td>
<td>1292</td>
<td>DB</td>
<td>FSAVE AND OFFH</td>
</tr>
<tr>
<td>03E0</td>
<td>00</td>
<td>1293</td>
<td>DB</td>
<td>0</td>
</tr>
</tbody>
</table>
LOC  OBJ  LINE  SOURCE  STATEMENT

03E1  48   DB      'H'
03E2  F4   DB      HSAVE AND OFFH
03E3  00   DB      0
03E4  4C   DB      'L'
03E5  F3   DB      LSAVE AND OFFH
03E6  00   DB      0
03E7  4D   DB      'M'
03E8  F4   DB      HSAVE AND OFFH
03E9  01   DB      1
03EA  50   DB      'P'
03EB  F6   DB      PSAVE+1 AND OFFH
03EC  01   DB      1
03ED  53   DB      'S'
03EE  F8   DB      SSAVE+1 AND OFFH
03EF  01   DB      1
03F0  00   DB      0 ; END OF TABLE MARKERS
03F1  00   DB      0

03FA  00   ORG     DATA
03FB  00   ORG     REGS   ; ORG TO REGISTER SAVE - STACK GOES IN HERE
03FC  00   MSTAK   EQU     $       ; START OF MONITOR STACK
03FD  00   DSAVE:  DB      0       ; D REGISTER SAVE LOCATION
03FE  00   CSAVE:  DB      0       ; C REGISTER SAVE LOCATION
03FF  00   BSAVE:  DB      0       ; B REGISTER SAVE LOCATION
0330  00   PSAVE:  DB      0       ; FLAGS SAVE LOCATION
0331  00   ASAVE:  DB      0       ; A REGISTER SAVE LOCATION
0332  00   LSAVE:  DB      0       ; L REGISTER SAVE LOCATION
0333  00   HSAVE:  DB      0       ; H REGISTER SAVE LOCATION
0334  0000  PSAVE:  DW      0       ; PGM COUNTER SAVE LOCATION
0335  0000  SSAVE:  DW      0       ; USER STACK POINTER SAVE LOCATION
0336  00   TEMF:   DB      0       ; TEMPORARY MONITOR CELL

PUBLIC SYMBOLS
EXTERNAL SYMBOLS

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL

USER SYMBOLS

EXTERNAL
NOTES:
- ALL RESISTORS ARE IN OHMS
- ALL CAPACITORS ARE IN uF UNLESS OTHERWISE NOTED