MCEM-8080
MICROCOMPUTER
SYSTEM
HAL MCEM-8080 MICROCOMPUTER SYSTEM

TECHNICAL MANUAL

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

WARRANTY

The HAL Communications Corp. MCEM-8080 Microcomputer System is fully guaranteed against defects in materials and workmanship for a period of one year. Should repair or replacement parts be required, notify HAL Communications Corp. promptly. Please do not return your unit to the factory for repair or adjustment until you have received a written return authorization.

HAL Communications assumes no responsibility for the repair or replacement of parts or units which have been damaged, abused, improperly installed, or modified and reserves the right to change the design of this equipment without incurring obligation to incorporate such changes into existing units. Operation of this equipment with improper power supply voltages (as described in this manual) will invalidate the warranty.

************************************************************
CONTENTS:

1. SYSTEM COMPONENTS ........................................... 1-1
   1.1 8080A Microprocessor .................................... 1-1
   1.2 Processor Control Circuitry ................................ 1-1
   1.3 Random Access Memory ...................................... 1-1
   1.4 Read Only Memory ........................................... 1-2
   1.5 Serial Input/Output (I/O) .................................. 1-2
   1.6 Parallel Data Input/Output .................................. 1-3
   1.7 Bus Indicators and Control ................................ 1-3
   1.8 Connectors used on the MCEM-8080 ......................... 1-5

2. INSTALLATION OF THE MCEM-8080 ............................... 2-1
   2.1 Initial Installation ........................................ 2-1
   2.2 Connection of Serial Input/Output Devices ............... 2-1
   2.3 Connection of Power Supplies to the MCEM .................. 2-9
   2.4 Connection of Parallel I/O Devices ....................... 2-9
   2.5 I/O Connector Pin Assignments .............................. 2-10
   2.6 Universal Processor Bus Connector .......................... 2-12

3. OPERATION OF THE MCEM-8080 ................................... 3-1
   3.1 Software Monitor ........................................... 3-1
   3.2 Monitor Subroutines ........................................ 3-5

4. SYSTEM ADDRESS ASSIGNMENTS .................................. 4-1
   4.1 Random Access Memory (RAM) ................................ 4-1
   4.2 Read Only Memory (ROM) ..................................... 4-1
   4.3 Input/Output (I/O) Assignments ............................. 4-2

5. OPERATING HINTS ............................................... 5-1
   5.1 Power-on Start Up .......................................... 5-1
   5.2 Changing the Monitor Mode .................................. 5-2
   5.3 Manually Writing a Memory Location ......................... 5-3
   5.4 Manually Jumping to a Program Address ..................... 5-3
   5.5 Manually Writing to an Output Port ......................... 5-4
   5.6 Using the Break-point Register for Debug ................... 5-4
   5.7 Using the E and R Commands for Debugging ................... 5-4

6. PROGRAM EXAMPLES .............................................. 6-1

7. SOFTWARE MONITOR LISTINGS .................................... 7-1

8. CIRCUIT BOARD LAYOUT AND SCHEMATIC DIAGRAMS .................. 8-1

9. APPENDICES .................................................... 9-1
   A. PROM, EPROM, ROM Connections and Markings ................. 9-1
   B. Input/Output Jumper Connections ............................. 9-6
   C. Addition of INTREQ, RESIN, and BUSEN to the UPB .......... 9-10
   D. 8080A Assembly Language Coding Forms ...................... 9-12
TABLES:

Table 2.1  ASCII Character Code .......................... 2-2
Table 2.2  Baudot Character Code ......................... 2-3
Table 2.3  Oscillator Frequency and Period ............... 2-5
Table 2.4  Power Requirements and Connections ........... 2-9
Table 2.5  I/O Connections ............................... 2-10
Table 2.6  Universal Processor Bus Connections .......... 2-12

ILLUSTRATIONS:

Figure 1.1  MCEM-8080 Microcomputer System ............. iv
Figure 2.1  EIA - RS-232C Serial I/O (standard jumpers)... 2-6
Figure 2.2  Dual current loop serial I/O .................. 2-7
Figure 2.3  Single current loop serial I/O ................. 2-8
Figure 2.4  Bit assignment and timing for parallel output.. 2-11
Figure 2.5  Bit assignment and timing for parallel input ... 2-11
Figure 8.1  MCEM-8080 circuit board layout ............... 8-2
Figure 8.2  Schematic drawing conventions ................ 8-3
Figure 8.3  MCEM-8080 CPU Interface / Control .......... 8-4
Figure 8.4  MCEM-8080 Processor Memory .................. 8-5
Figure 8.5  MCEM-8080 Input/Output Interface ............. 8-6
Figure A.1  Memory jumpers for 3624-4 PROMs .............. 9-2
Figure A.2  Memory jumpers for 8704 EPROM ................. 9-3
Figure A.3  Memory jumpers for 8708 or 8308 ROMs ........ 9-4
Figure A.4  PROM color coding ............................ 9-5
Figure B.1  EIA - RS-232C Serial I/O Jumper Locations .. 9-7
Figure B.2  Dual current loop serial I/O Jumper Locations . 9-8
Figure B.3  Single current loop serial I/O Jumper Locations. 9-9
Figure C.1  Addition of RESIN, BUSEN, and INTREQ to UPB . 9-11
INTRODUCTION

The HAL MCEM-8080 Microcomputer System is a single printed-circuit board computer that can be used for program development or for specific control applications. The MCEM-8080 is designed around the Intel 8080A single chip, 8-bit, N-channel microprocessor integrated circuit. The MCEM-8080 printed circuit board contains the microprocessor IC, its timing and control circuitry, both Read Only Memory (ROM) and Random Access Memory (RAM) integrated circuitry, and timing and control for Input / Output (I/O) interfacing. Other accessories such as additional RAM, Keyboard/Video Display unit, tape cassette memory, and power supplies can be used with the basic MCEM circuit board. This manual discusses ONLY the basic MCEM-8080 board - the operation of the accessories is discussed in separate manuals furnished with each unit.

1. System Components

The HAL MCEM-8080 Microcomputer System contains the following basic and optional components:

1.1 8080A Microprocessor

The 8080A is an eight bit microprocessor integrated circuit with an instruction repertoire of 73 instructions. The execution time of these instructions varies from 2.0 μsec. to 9.0 μsec. The 8080A integrated circuit itself contains all of the circuitry required to address the memory, address Input / Output (I/O) devices, and manipulate data. A more detailed discussion of the 8080A will be found in pages 2-1 to 2-20 of the Intel manual (Intel; pp 2-1 to 2-20).

1.2 Processor Control Circuitry

Two additional integrated circuits are used in conjunction with the 8080A to provide all of the timing and control signals for the microprocessor system. These are the 8228 Bus Controller IC and the 8224 Clock Generator IC.

1.2.1 8228 Bus Controller (Intel; pp 5-7 to 5-12)

A type 8228 integrated circuit is used to decode signals from the 8080A and generate the required bus control signals. This device also buffers the 8080A data bus signals and will support a single vector interrupt (RST 7).

1.2.2 8224 Clock Generator (Intel; pp 5-1 to 5-6)

A type 8224 integrated circuit generates all system timing signals. An 18 MHz crystal is used with the device to generate the 2.0 MHz processor timing signals, power-on reset signal and ready line synchronization pulses.

1.3 Random Access Memory

The standard MCEM circuit board is provided with 1024 bytes of Random Access Memory (RAM). This memory can be used by the user's programs, but the lower 64 bytes are required for the software monitor program. Additional circuit board space is provided so that an additional 1024 bytes ("1K") of RAM can be installed on the MCEM board (factory installation is recommended). All RAM integrated circuits should be type 8102A-4, a device featuring an access time of 450 nsec. or less. Slower RAM devices should NOT be used as they may cause improper operation of the system. Further information on the 8102A-4 is found on pages 5-79 through 5-82 of the Intel manual (Intel; pp 5-79 to 5-82).

Within the processor memory space, the standard "1K" bytes of RAM occupy locations between 0 and 1023 (0 - 3FF - Hex). The second (optional) "1K" bytes of RAM occupy locations between 1024 and 2047 (400 H - 7FF H). The software monitor uses RAM locations between 0 and 63 (0 - 3F H).
1.4 Read Only Memory

The MCEM system is provided with sufficient circuit board space for 4096 bytes of EPROM (Erasable Programmable Read Only Memory) or 2048 bytes of bi-polar PROM (Programmable Read Only Memory - NOT erasable). The device selection is made by selection of the proper circuit board jumpers. Four socket locations are provided for the ROM - all four must be of the same type (EPROM or PROM). The ROM occupies consecutive memory locations, starting at 32,768 (8000H).

1.4.1 EPROM

Either a type 8708 or 8704 EPROM integrated circuit (Intel; pp 5-45 to 5-50) can be used on the MCEM board. The 8708 is a 1024 x 8 device and the 8704 is a 512 x 8 device. Refer to Appendix A for proper jumper placement.

1.4.2 PROM

Type 3624 PROM integrated circuits can be used on the MCEM. This IC is the standard device furnished with the MCEM. The 3624 is a bi-polar PROM with a 512 x 8 organization. Up to four 3624's can be used on the MCEM-8080 circuit board. NOTE: Production MCEM-8080 circuit boards are jumpered for use of this device on the circuit board. If it is desired to use other devices, refer to Appendix A for details.

1.4.3 ROM

A type 8308 ROM integrated circuit (Intel; pp 5-59 & 5-60) can also be used in the HAL MCEM-8080. This is a mask-programmed version of the 8708. Refer to Appendix A for jumper details.

1.4.4 Monitor Software ROM

The HAL software monitor can be resident in either 2-3624, 1-8708, or 1-8308 ROM integrated circuits. Either 2-3624 or 1-8308 ROM is standard with the MCEM. The monitor software is 1024 bytes in length and begins at location 32,768 (8000H).

1.5 Serial Input / Output (I/O)

The standard MCEM-8080 provides for either synchronous or asynchronous serial data interface. The software monitor supports asynchronous serial I/O in either Baudot (5-unit) or ASCII (8-unit) codes.

1.5.1 8251 USART

A type 8251 integrated circuit (Intel; pp 5-135 to 5-146) Universal Synchronous/Asynchronous Receiver/Transmitter (USART) is used to input and output serial data. This device is fully programmable and is controlled by the processor. Parallel-to-serial and serial-to-parallel conversions as well as word length selection and parity are controlled by the 8251.
1.5.2 Serial Timing Oscillator

A type 555 integrated circuit timer is used to generate the serial data baud rate. The data rate is screw-driver adjustable on the circuit board. The actual 555 clock frequency is 4 times the baud rate in ASCII mode and 16 times the baud rate in Baudot mode.

1.5.3 EIA - RS-232C Data Interface

Two operational amplifiers (both halves of a type 1458 IC) are used as RS-232 drivers and receivers. The serial output of the 8251 USART is directly converted to a ± 5 volt signal, with -5 volts representing the "mark" signal condition and +5 volts as "space". The output impedance of the circuit is approximately 400 ohms. For input data, an operational amplifier is used as a sense amplifier and level converter. Input voltages greater than +1.0 volts are interpreted to be in the "space" condition and those less than +1.0 volts as "mark". The input impedance is approximately 2700 ohms. This input will properly sense TTL-level signals, as well as EIA - RS-232C signals.

1.5.4 Current Loop Interface

Current loop signals with either 20 or 60 ma mark currents can also be connected to the MCEM-8080. Two optical isolator integrated circuits are used to convert between the floating current loop circuit and the RS-232 levels. These sensors are separated so that one can be used for data input and the other for output (separate current loops - "full-duplex" operation). The two circuits can also be series connected to provide both data input and output on a single current loop circuit ("half-duplex" operation).

1.6 Parallel Data Input / Output

A type 8255 integrated circuit (Intel; pp 5-113 to 5-133) is provided to allow parallel data interfacing. This device, called the "Programmable Peripheral Interface", consists of three buffered 8-bit parallel data ports. The software monitor utilizes the 8255 for parallel I/O operations.

1.7 Bus Indicators and Control

A number of indicators (small LEDs - Light Emitting Diodes) and switches are installed along the front edge of the MCEM-8080 circuit board to permit evaluation and control of the processor operation.

1.7.1 Address Indicators

The entire 16 bits of the 8080 address bus are displayed on 16 LEDs. The lamps are grouped in four-lamp clusters, four clusters total. Each group of four lamps represents a single hexadecimal (HEX) character, 0 through F. An illuminated lamp indicates a logic "1" condition. Within a four-lamp cluster, the least significant bit (LSB) is represented by the right-hand lamp. Similarly, the right-hand cluster of four lamps represents the least significant hexadecimal character.
1.7.2 Data Indication

Eight lamps (in two four-lamp clusters) are used to indicate the state of the processor data bus. These lamps are immediately to the left of the address lamps. As before, the right-hand lamp represents the LSB and an illuminated lamp represents a logical "1" for that bit.

1.7.3 Bus Control Indication

The four lamps on the extreme left end of the circuit board indicate the state of the I/O Read, I/O Write, Memory Read, and Memory Write (left-to-right order) signals from the processor. An illuminated lamp indicates which of these operations is active. A complete description of the function of these signals is found in the Intel manual (Intel; pp 5-7 to 5-12).

1.7.4 Manual Data Switches

Immediately in front of the eight data lamps are located two, four-section miniature switches. The switches provide manual control of the contents of the data bus. These switches can be used to enter data only when the Data Bus Override (DBO) switch (to the right of the data switches) is in the ON position. The data switch settings at any other time does not affect the processor. The switches are arranged in the same manner as the lamps, LSB to the right.

1.7.5 Run / Stop Switch

A miniature toggle switch on the right-hand section of the board (labeled RUN - STOP) allows manual control of the 8080A Ready line. When this switch is set to the RUN position, the processor will continue to operate (unless halted by the program or some other control). When in the STOP position, the processor is halted and only the manual STEP and RESET switches will cause processor activity.

1.7.6 Reset Switch

The far right-hand push-button switch (labeled RESET) is a momentary contact type that can be used to manually reset the 8080A. A reset operation causes the program counter to set to zero and the interrupt flip-flop to be cleared. Processor execution commences at location 0000 when the reset switch is released. Application of DC power supplies automatically issues a reset function.

1.7.7 Single Step Switch

The STEP switch (located between the RUN - STOP and RESET switches) allows manual stepping of the computer, one MEMORY cycle at a time. This switch only functions when the processor has been halted by either the RUN - STOP switch or the break point register. It is important to remember that some instructions require more than one memory cycle and therefore more than one operation of step switch to complete.
1.7.8 Break Point Register Switches

In the middle of the control area of the circuit board are located four, four-section miniature switches. These 16 switches form a "break point register". Circuitry is provided to compare the value of this switch register with the address bus and cause the 8080A to stop operation if the two are equal. This function is similar to a programmable stop. Once the 8080A is halted due to a break point "match", it can only be caused to continue running by either manual stepping with the STEP switch or by resetting the break point switches to a new value.

1.7.9 Memory Write and Output Write Switches

Two momentary switches are located on the far left-hand side of the circuit board. These switches allow manual operation of memory or output functions. The MEMORY WRITE switch will cause a manual memory write function when depressed, overriding the normal bus control from the 8228 integrated circuit. Similarly, depression of the OUTPUT WRITE switch will cause an output write function, again overriding the normal control from the 8228.

1.8 Connectors used on the MCEM-8080

There are three connectors used on the basic MCEM-8080 circuit board. These connectors are used for I/O Interface, Power Input, and connection to the Universal Processor Bus. Mating connectors for each are furnished with the MCEM.

1.8.1 I/O Interface Connector

Input / Output (I/O) connections to the MCEM are made through a 36 pin circuit board edge connector (0.156" finger spacing, 18 pin double readout) located on the left edge of the board. All three parallel I/O ports of the 8255 are available on this connector as well as connections for serial data. The form of serial data to be used is selected with circuit board jumpers.

1.8.2 Power Connector

Power connections to the MCEM are made through the 12 pin edge connector (0.156" finger spacing 6 pin double readout) located in the upper right-hand corner of the circuit board. The MCEM requires ±12 volt and +5 volt power supplies.

1.8.3 Universal Processor Bus Connector

Direct connection to the computer address, data, and control lines can be made through the 40 pin Universal Processor Bus (UPB) connector located in the lower right-hand corner of the board. A mating connector and attached ribbon cable are supplied for use of this feature. Connection of options such as additional memory and the Keyboard/Video Display unit is made through the UPB connector.
2. INSTALLATION OF THE MCEM-8080

2.1 Initial Inspection

Upon receipt of the MCEM-8080, unpack the circuit board and accessories and inspect them for evidence of shipping damage. If evidence of shipping damage is found, contact the carrier immediately. Before discarding the packing material, check that all parts and accessories are accounted for. If any are missing, please notify the factory or distributor in writing. The following parts and accessories are furnished with the MCEM-8080:

Accessories and Parts:

1 - 40 pin Universal Processor Bus (UPB) connector with 2 ft. of ribbon cable attached.
1 - 36 pin edge connector
1 - 12 pin edge connector
1 - MCEM Operating Manual
1 - Intel 8080 Microcomputer System User's Manual
1 - Intel 8080 Assembly Language Reference Card

2.2 Connection of Serial Input / Output Devices

The MCEM-8080 standard circuitry and software will support serial I/O (Input/Output) operations in either the 7-unit ASCII code OR the 5-unit Baudot code at a variety of baud rates. The code to be used is selected with circuit board jumpers. The MCEM-8080 is usually factory connected for the ASCII code.

2.2.1 ASCII Serial I/O Operation

The ASCII mode is selected by strapping pin 22 (DSR) of the 8251 (circuit number 15, left edge of board) to ground (see Appendix B). In ASCII mode, all serial communications is performed with a 7-bit ASCII format. This format is:

1 - start bit (space)
7 - data bits
1 - parity bit (set to space)
2 - stop bits (mark)
11 - bits per character

The serial baud rate timing is screw driver adjustable from 100 to 600 baud. The unit is factory adjusted for 300 baud (30 characters per second). As noted in section 1.5.2, the 555 timer is set to 16 times the output baud rate (e.g., 16 x 300 = 4800 Hz for 300 baud). Table 2.1 contains a list of the ASCII character set used and their corresponding hexadecimal values. Common ASCII baud rates and the corresponding oscillator frequencies and periods are listed in Table 2.3.
Table 2.1  ASCII Character Code

<table>
<thead>
<tr>
<th>3 Most Significant Bits</th>
<th>4 Least Significant Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 NUL</td>
<td>7 RUB</td>
</tr>
<tr>
<td>1 SOH DC1 !</td>
<td>8 BS CAN (</td>
</tr>
<tr>
<td>2 STX DC2 &quot;</td>
<td>9 HT EM )</td>
</tr>
<tr>
<td>3 ETX DC3 #</td>
<td>A LF SUB *</td>
</tr>
<tr>
<td>4 EOT DC4 $</td>
<td>B VT ESC +</td>
</tr>
<tr>
<td>5 ENQ NAK %</td>
<td>C FF FS ,</td>
</tr>
<tr>
<td>6 ACK SYN &amp;</td>
<td>D CR GS - = M ] m</td>
</tr>
<tr>
<td>7 BEL ETB '</td>
<td>E SO RS . = N ^ n</td>
</tr>
<tr>
<td>8 BS CAN (</td>
<td>F SI US / = O o</td>
</tr>
<tr>
<td>9 HT EM )</td>
<td>ACK = acknowledge</td>
</tr>
<tr>
<td>A LF SUB *</td>
<td>BEL = bell</td>
</tr>
<tr>
<td>B VT ESC +</td>
<td>BS = backspace</td>
</tr>
<tr>
<td>C FF FS ,</td>
<td>CAN = cancel</td>
</tr>
<tr>
<td>D CR GS - = M ] m</td>
<td>CR = carriage return</td>
</tr>
<tr>
<td>E SO RS . = N ^ n</td>
<td>DC1 = device control 1</td>
</tr>
<tr>
<td>F SI US / = O o</td>
<td>DC2 = device control 2</td>
</tr>
<tr>
<td>ACK = acknowledge</td>
<td>DC3 = device control 3</td>
</tr>
<tr>
<td>BEL = bell</td>
<td>DC4 = device control 4</td>
</tr>
<tr>
<td>BS = backspace</td>
<td>DLE = data link escape</td>
</tr>
<tr>
<td>CAN = cancel</td>
<td>EM = end of medium</td>
</tr>
<tr>
<td>CR = carriage return</td>
<td>ENQ = WRU = enquiry</td>
</tr>
<tr>
<td>DC1 = device control 1</td>
<td>EOT = end of transmission</td>
</tr>
<tr>
<td>DC2 = device control 2</td>
<td>ESC = escape</td>
</tr>
<tr>
<td>DC3 = device control 3</td>
<td>ETB = end of transmission block</td>
</tr>
<tr>
<td>DC4 = device control 4</td>
<td>ETX = end of text</td>
</tr>
<tr>
<td>DLE = data link escape</td>
<td>FF = form feed</td>
</tr>
<tr>
<td>EM = end of medium</td>
<td>FS = file separator</td>
</tr>
<tr>
<td>ENQ = WRU = enquiry</td>
<td>GS = group separator</td>
</tr>
<tr>
<td>EOT = end of transmission</td>
<td>HT = horizontal tabulation</td>
</tr>
<tr>
<td>ESC = escape</td>
<td>LF = line feed</td>
</tr>
<tr>
<td>ETB = end of transmission block</td>
<td>NAK = negative acknowledge</td>
</tr>
<tr>
<td>ETX = end of text</td>
<td>NUL = null</td>
</tr>
<tr>
<td>FF = form feed</td>
<td>OUT = delete (= DEL)</td>
</tr>
<tr>
<td>H X h x</td>
<td>RUB = delete (= DEL)</td>
</tr>
<tr>
<td>I Y i y</td>
<td>OUT = delete (= DEL)</td>
</tr>
<tr>
<td>J Z j z</td>
<td>SI = shift in</td>
</tr>
<tr>
<td>K [ k {</td>
<td>SO = shift out</td>
</tr>
<tr>
<td>L \ l ]</td>
<td>SOH = start of heading</td>
</tr>
<tr>
<td>M ] m</td>
<td>STX = start of text</td>
</tr>
<tr>
<td>N ^ n</td>
<td>SUB = substitute</td>
</tr>
<tr>
<td>O o RUB</td>
<td>SYN = synchronous idle</td>
</tr>
<tr>
<td>OUT</td>
<td>US = unit separator</td>
</tr>
</tbody>
</table>

Mark = logical 1
Data is transmitted LSB first.
Table 2.2  Baudot Character Code

<table>
<thead>
<tr>
<th>Most Significant Bit (1)</th>
<th>Letters</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø</td>
<td>1</td>
<td>Ø</td>
</tr>
<tr>
<td>Ø</td>
<td>BLANK</td>
<td>T</td>
</tr>
<tr>
<td>Ø</td>
<td>E</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>LF</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>W</td>
</tr>
<tr>
<td>3</td>
<td>SPACE</td>
<td>H</td>
</tr>
<tr>
<td>4</td>
<td>SPACE</td>
<td>#</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>I</td>
<td>P</td>
</tr>
<tr>
<td>7</td>
<td>U</td>
<td>Q</td>
</tr>
<tr>
<td>8</td>
<td>CR</td>
<td>O</td>
</tr>
<tr>
<td>9</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>B</td>
<td>J</td>
<td>FIG</td>
</tr>
<tr>
<td>C</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>X</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>V</td>
</tr>
<tr>
<td>F</td>
<td>K</td>
<td>LTR</td>
</tr>
</tbody>
</table>

BEL = bell (or *), BLANK = blank (non print or space), CR = carriage return, FIG = figures case, LTR = letters case, LF = line feed, MARK = logical 1, Data is transmitted, LSB first.
circuit board. THE MCEM-8080 CAN BE DAMAGED IF THE I/O CONNECTOR IS REVERSED (particularly if connected to high-voltage current loop circuits).

2.6 Universal Processor Bus Connector  

See Addendum 2.

The processor bus of the 8080A can be extended with a 40 conductor ribbon cable attached to the Universal Processor Bus (UPB) connector. The total length of this cable should not exceed 24 inches. The total external loads should not exceed three standard TTL loads on the address and control lines and 5, LOW CURRENT, bus receiver loads on the data lines. The connections to the UPB connector are shown in Table 2.6.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
<th>Pin</th>
<th>Function</th>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A12</td>
<td>15</td>
<td>A1</td>
<td>29</td>
<td>A14</td>
</tr>
<tr>
<td>2</td>
<td>+12</td>
<td>16</td>
<td>MEMR</td>
<td>30</td>
<td>(NC)</td>
</tr>
<tr>
<td>3</td>
<td>A1Ø</td>
<td>17</td>
<td>A3</td>
<td>31</td>
<td>A15</td>
</tr>
<tr>
<td>4</td>
<td>+5</td>
<td>18</td>
<td>I/O R</td>
<td>32</td>
<td>Locating Key</td>
</tr>
<tr>
<td>5</td>
<td>A8</td>
<td>19</td>
<td>A5</td>
<td>33</td>
<td>DBØ</td>
</tr>
<tr>
<td>6</td>
<td>Ground</td>
<td>20</td>
<td>T70 W</td>
<td>34</td>
<td>DB4</td>
</tr>
<tr>
<td>7</td>
<td>A6</td>
<td>21</td>
<td>A7</td>
<td>35</td>
<td>DB1</td>
</tr>
<tr>
<td>8</td>
<td>Ground</td>
<td>22</td>
<td>RESET</td>
<td>36</td>
<td>DB5</td>
</tr>
<tr>
<td>9</td>
<td>AØ</td>
<td>23</td>
<td>A9</td>
<td>37</td>
<td>DB2</td>
</tr>
<tr>
<td>10</td>
<td>Ø2 (TTL)</td>
<td>24</td>
<td>RDY</td>
<td>38</td>
<td>DB6</td>
</tr>
<tr>
<td>11</td>
<td>A2</td>
<td>25</td>
<td>A11</td>
<td>39</td>
<td>DB3</td>
</tr>
<tr>
<td>12</td>
<td>(NC)</td>
<td>26</td>
<td>(NC)</td>
<td>4Ø</td>
<td>DB7</td>
</tr>
<tr>
<td>13</td>
<td>A4</td>
<td>27</td>
<td>A13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>MEMW</td>
<td>28</td>
<td>(NC)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Connections with the (NC) designator may have a function assigned but not connected on the factory standard units.
3. **Operation of the MCEM-8080**

3.1 **Software Monitor**

The software monitor supplied with the MCEM-8080 properly interfaces the serial I/O port, the parallel I/O ports, the keyboard display option, or other user-defined I/O devices. The monitor allows the user to perform the following operations. These commands are entered from the console.

3.1.1 **Load hex (hexadecimal) files.**

Large files can be loaded into the MCEM-8080 RAM from the reader device by using the following format:

```
: 10 0030 00 3E57......23 8E
```

**COLON:** All records must start with a COLON character. Any characters preceding the COLON are ignored.

**RECORD LENGTH:** The number of load bytes in the data field is specified as a number between 00 and FF (0 to 255). This is a hexadecimal number and is either two characters long or a single character followed by a comma (i.e., 07 = 7,). If a zero length record is entered, the load is terminated and control is restored to the monitor.

**LOAD ADDRESS:** The memory location into which the first byte of the data field will be written is specified here. Successive bytes in the data field will be written into successively higher memory locations. This number is either four characters long or less if terminated with a comma (i.e., 032E = 32E,).

**RECORD TYPE:** The record type is specified here. With the present version of the monitor (Version 1.1), all records are of type zero (enter 00).

**DATA FIELD:** The actual data to be written into memory is specified here. These are two character hex bytes and each pair of characters is converted to eight bits to be loaded into memory.
SUMCHECK: This hex byte represents the negative sum of all bytes (the load address is two bytes) in the record. The SUMCHECK value is such that, when modulo 256 is added to all of the other bytes of the record, the total will equal zero. This is a validity check on the record. If the SUMCHECK fails, an "X" will be printed on the serial output device. However, the data will still be loaded if the SUMCHECK fails.

The format used to specify a load file is:

```
  .L 0 }
```

  Carriage Return (CR)

  Load offset (added to the load address portion of the hex records to load memory other than that specified, up to four hex characters, 0000 - FFFF)

  L indicates load

  Prompting period issued by the monitor

After receiving this command, the monitor will begin searching for the first colon.

3.1.2 Dump or Display

The contents of memory can be dumped (or displayed) by specifying the range to be dumped. The output generated is compatible with the load command so that memory areas can first be dumped and then loaded. The format of the dump is in a number of hex records (of maximum length = 10 H) until the entire range is depleted. For clarity, spaces are inserted between the various bytes but the monitor ignores spaces on input so that the dumped file is compatible with the load file routine. The dumped file is sent to the punch device. The command format is:

```
  .D 300 , 400 }
```

  Carriage Return (CR)

  First undumped byte

  Comma separator

  First dumped byte

  D (Dump) command

  Monitor prompting period

This example command will cause display of all memory contents between locations 300 H to 400 H - 1 as 16, 16 byte records. A zero length record is always added at the end.
3.1.3 Insert Memory Data

Individual locations in memory can be modified by using this command. The command format is:

. I 82E }
   Carriage Return (CR)
   Starting memory address
   I (Insert) command
   Monitor prompting period

The output generated is of the following format:

82E = 27
   Present memory contents
   Equals sign
   Space character
   Address

After this has been output, a comma is typed followed by a new byte and when done, written into memory. If it is desired to leave the memory location unchanged, any non-comma character can be typed. After the new data has been entered, the address is incremented and displayed again. For example, consider:

. I 82E  
   (Insert memory command, generated by user)

82E = 27 , 2E
   New data (entered by user)
   Comma (entered by user)
   Response by computer

82F = 87  
   (Computer response indicating contents of next location)

If any character between "G" and "Z" is typed instead of a hex character, control returns to the monitor.

3.1.4 JUMP Command

Program control can be transferred to a specific location through the JUMP command. This command can be used to "jump" to a user program or subroutine. The format for this command:
3.1.5 RETURN command

Program control can be transferred to a specific location and the CPU registers restored to a predetermined value by executing a RETURN command. The format of this command is:

\[ R \ 283 \]

Carriage Return (CR)
Destination address
RETURN command
Prompting period issued by monitor

Twelve register values are restored by this command including:

<table>
<thead>
<tr>
<th>Register</th>
<th>Stored at Memory Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>37</td>
</tr>
<tr>
<td>C</td>
<td>36</td>
</tr>
<tr>
<td>D</td>
<td>35</td>
</tr>
<tr>
<td>E</td>
<td>34</td>
</tr>
<tr>
<td>A (Accumulator)</td>
<td>33</td>
</tr>
<tr>
<td>PSW (Processor Status Word)</td>
<td>32</td>
</tr>
<tr>
<td>H</td>
<td>31</td>
</tr>
<tr>
<td>L</td>
<td>30</td>
</tr>
<tr>
<td>PC (high order program counter)</td>
<td>2F</td>
</tr>
<tr>
<td>PCL (low order program counter)</td>
<td>2E</td>
</tr>
<tr>
<td>SP (high order stack pointer)</td>
<td>2D</td>
</tr>
<tr>
<td>SPL (low order stack pointer)</td>
<td>2C</td>
</tr>
</tbody>
</table>

The initial value (to be restored) of these registers can be set by using an I (Insert) command to the memory location used for storage. These locations are shown in the above list. Note that, during the process of restoring the registers, the stack area indicated by SP (SPH & SPL) is used as temporary storage and therefore SP should contain a valid RAM address. If the destination address specified in the command is zero, the destination is taken from the storage area.

3.1.6 STOP command

A STOP command can be initiated at any time at which the monitor is expecting a control character by typing an "S" (or any other letter between
"G" and "Z"). As explained in section 3.1.4, this will cause the command to be aborted and control is returned to the monitor. The monitor will then issue a new prompting period.

3.1.7 EXIT command

An exit from a program to the monitor can be executed by entering a RST 7 instruction or a CALL 38 H. The monitor, upon turn-on, establishes an entry at 38 H from which it saves ALL CPU registers and status. This command is intended to permit the examination of all CPU registers and status while in the process of executing a program. The RST 7 instruction saves the PC (Program Counter) on the stack and jumps to location 38 H. From here, it jumps to a routine within the monitor which copies all registers into a special RAM area. When finished, the address of the initial RST 7 instruction is typed out as:

EXIT 232E (hexadecimal)

A prompting period is then issued by the monitor. At this point, the I (Insert) command can be used to examine and/or change individual registers. The memory location used to store the register values is listed under the RETURN command.

The most valuable use of the exit command is accomplished by inserting a RST 7 (0FF H) instruction in the program sequence being de-bugged and an automatic exit will be executed. The RETURN command can be used to return to the program sequence. An interrupt will also cause the exit command to be executed since a RST 7 is used as the interrupt vector.

3.2 Monitor Subroutines

Several general purpose subroutines are included in the software monitor. Some of these subroutines are:

3.2.1 BEGIN (address 8000 H)

This subroutine allows general entrance to the monitor mode. It initializes all parameters and the USART.

3.2.2 CI (Console Input - address 8003 H)

CI is a console input routine that will return an ASCII character (standard serial I/O) from the console control device and place the ASCII code in the A register. The contents of the A and PSW registers are modified. Three levels of the stack are used by this operation.

3.2.3 RI (address 8006 H)

This routine is the same as the CI routine except that the character is originated by the reader input device instead of by the console. Serial ASCII I/O is standard.
3.2.4 CO (address 8009 H)

This subroutine causes an ASCII character in the C register to be output to the console device (serial I/O is standard). The contents of the A and PSW registers are modified and three stack levels are used by this operation.

3.2.5 PO (address 800C H)

This routine is the same as the CO subroutine except that the ASCII character is output to the punch device (serial ASCII I/O is standard).

3.2.6 LO (address 800F H)

This routine is also similar to the CO routine with the exception that the data is output to the list output device. As before, serial ASCII I/O is the standard code format.

3.2.7 CSTS (address 8012 H)

This is a console status request subroutine which evaluates the status of the console input device and returns A = Ø (zero value in the A register) or A = 0FF H if an input character is waiting. Since the CI subroutine will only return if a character is input, a call to CSTS can be used to determine if a call to CI is successful (will result in a character being input and returned).

3.2.8 IOCHK (address 8015 H)

A single memory location in RAM is used to define the four input / output (I/O) devices. The logical devices available are:

<table>
<thead>
<tr>
<th>Logical Device</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSOLE:</td>
<td>CI, CO, CSTS</td>
</tr>
<tr>
<td>READER:</td>
<td>RI</td>
</tr>
<tr>
<td>PUNCH:</td>
<td>PO</td>
</tr>
<tr>
<td>LIST:</td>
<td>LO</td>
</tr>
</tbody>
</table>

These logical devices can be "assigned" to any one of the following physical devices:

- **Serial I/O:** Uses the 8251 USART
- **Keyboard / Display:** Optional MCEM-KB/VDU Keyboard/Video Display Unit
- **Parallel I/O:** Uses the 8255 Programmable Peripheral Interface IC
- **User Defined I/O:**

  - **USRIN (address 40 H):** A user input subroutine which will return an ASCII character in the A register, similar in operation to the CI subroutine.
  - **USROTX (address 43 H):** A user subroutine, similar in function to the CO subroutine, which will allow output to the user I/O of an ASCII character in the C register.
USRST (address 46 H): A user status routine which returns
A = 0 if USRIN will not return a character immediately and
A = OFF H if USRIN will immediately return a character.

Serial I/O data is processed through the 8251 USART integrated circuit
and may be either serial Baudot (5-unit) code (DSR pin = "1") or serial
ASCII (8-unit) code (DSR pin = "0"). In Baudot code, the code conversion
to and from ASCII code is performed by the I/O subroutine and need not be
done otherwise. For instance, if a call to CO is performed while the console
is assigned to the serial I/O, an ASCII character should always be present
in the C register. The monitor routine checks the status of the DSR line
and performs a code conversion if necessary.

The Keyboard / Video Display Unit is a HAL Communications option
available for the MCEM-8080. If a logical device is assigned to the Key-
board / Video Display Unit, the monitor will automatically write the dis-
play screen (output), read the keyboard (input), and check the keyboard
status.

Parallel I/O data is processed through the 8255 PPI integrated
circuit. Port A of the 8255 is used for data output, Port B for input,
and Port C for control. The seven-bit ASCII code (bit 8 = "0", space)
is used for parallel I/O. Mode 1 of the 8255 is used (Intel; p 5-123).

The user defined I/O capability is provided so that the user can write
his own I/O subroutines to service particular devices (such as an electrically
controlled Selectric (© IBM) typewriter, etc.). The monitor automatically
calls a set of routines which start at location 40 H (USRIN, USROUT, and
USRST) for user I/O applications. When the monitor requests a character
(CI, RI), a call to 40 H is executed. To output a character, a call to
43 H is executed; if the status of the I/O device is needed, a call to
46 H is executed. The routines in these locations should conform to the
CI, CO, and CSTS format. For example:

Address: 40 H JMP INPUT
43 H JMP OUTPUT

\[
\begin{align*}
46 H & \quad \text{INPUT STATUS ROUTINE} \\
& \quad \text{USER INPUT ROUTINE} \\
& \quad \text{OUTPUT USER OUTPUT ROUTINE}
\end{align*}
\]

Memory location 3 is used to store the I/O device assignments. The
format of the assignment byte is:

\[
\begin{align*}
D7 & \quad D6 \quad D5 \quad D4 \quad D3 \quad D2 \quad D1 \quad D0 \\
\text{LIST} & \quad \text{PUNCH} \quad \text{READER} \quad \text{CONSOLE}
\end{align*}
\]

Contents of Memory Location 3

3-7
D0 and D1 define the console device (CO, CI, CSTS)
D2 and D3 define the reader device (RI)
D4 and D5 define the punch device (PO)
D6 and D7 define the list device (LO)

Each two bit set can have one of the following four values:

00 assigns serial I/O
01 assigns Keyboard Video Display option
10 assigns parallel I/O
11 assigns user I/O.

For example:

Memory location 3 = 10110001 B (61 H) defines that:

(a) Console operations are via the optional Keyboard / Video Display Unit,
(b) Reader operations are via the serial I/O device,
(c) Punch operations are via the user I/O device,
(d) List operations are via the parallel I/O.

The monitor automatically sets memory location to 0000 0000 B (00 H) upon turn-on and thus assigns all logical devices to the serial I/O port.
The monitor also checks to see if the optional Keyboard / Video Display Unit has been attached to the UPB. If so, memory location 3 is set to 01010101 B (55 H) which assigns all logical devices to the Keyboard / Video Display Unit.

If, at any time, it is desired to reassign the I/O system, the \texttt{I} command can be used. For example:
\texttt{.I 03} \texttt{\uparrow}

will result in a request to change location 3 which contains the I/O assignments.

A call to \texttt{IOCHK} will return the value of memory location 3 in the A register. A call to \texttt{I0SET} will write the contents of the C register into memory location 3. If it is desired to change the I/O system assignments, these routines should be used.

3.2.9 MEMCK (address 801B H)

This routine returns the contents of memory location 5 into the B register and memory location 6 into the A register. These locations are intended to hold the address of the first non-RAM memory address and are used by the resident assembler and editor to determine how much memory is available to them. The I (INSERT) command should be used to set these values if this routine is used.
4. System Address Assignments

The MCEM-8080 uses Random Access Memory (RAM), Read Only Memory (ROM), Input ports, and Output ports. The address assignments for these sections are discussed below.

4.1 Random Access Memory (RAM)

The random access memory is used by both the monitor and for user storage. The standard MCEM-8080 systems are furnished with 1024 ("1 K") bytes of RAM - this can be doubled to "2 K" of RAM by the addition of more integrated circuits to the circuit board.

4.1.1 Monitor RAM Usage

Memory locations between \( \emptyset \) and \( 3FF \) are reserved for RAM memory. The area between \( \emptyset \) and \( 40 \) H is used by the software monitor for the stack and for temporary storage. User programs should not use these storage locations to avoid interference with the monitor. As explained in section 3.2.8, the entry points for user I/O assignments are \( 40 \) H (User Input), \( 43 \) H (User Output), and \( 46 \) H (User Input Status).

4.1.2 User RAM Usage

All RAM in locations higher than \( 40 \) H is available to the user for program storage. The monitor stack does not take the possible requirements of a user stack into account. Therefore, user programs should establish a stack in the free RAM area (higher than \( 40 \) H). The EXIT command and RETURN commands assume that at least three levels (6 bytes) of user stack are available and that the user stack is not the same as the monitor stack.

4.1.3 Optional RAM

The standard MCEM-8080 circuit board has 1024 bytes ("1 K") of RAM integrated circuits installed. However, additional circuit board space and connections are provided that 1024 bytes of RAM ICs can be added, for a total of "2 K" (2048) bytes of RAM. Only type 8102A-4 integrated circuits should be used to assure compatibility with the rest of the MCEM-8080. It is highly recommended that these integrated circuits be installed by the HAL Communications factory to assure proper system operation. When the second "1 K" of RAM is used, it occupies the address space between \( 400 \) H and \( 7FF \) H. The installation of this additional RAM does not affect the monitor RAM usage and therefore, all of the additional RAM storage is available for user programs.

4.2 Read Only Memory (ROM)

The MCEM-8080 uses Read Only Memory (ROM) for non-volatile program stage. (Non-volatile = stored data is retained even when power is removed from the MCEM-8080. RAM is a volatile memory; ROM is non-volatile.) Typical uses of the ROM include storage of the monitor, support subroutines for peripheral devices, and user programs.
4.2.1 Monitor ROM

The memory locations between 8000 H and 83FF H are occupied by the software monitor program. This program uses a part of the ROM storage space on the main MCEM-8080 circuit board. It is contained in either 2 - 3624 PROM's, 1 - 8708 EPROM, or 1 - 8308 ROM integrated circuits. When the type 3624 ICs are used, the monitor program consumes one-half of the available on-board ROM space. When either the 8708 or 8308 ICs are used, the monitor consumes one-quarter of the on-board ROM space.

4.2.2 Peripheral ROM

Many MCEM-8080 peripherals require support programs ("software") to operate. Typical such peripheral devices include the Keyboard / Video Display Unit option and the PROM Programmer option. The Keyboard / Video Display Unit support software is physically resident on its circuit board and logically located between memory locations FB00 H and F9FF H. Similarly, the ROM containing the software to support the PROM Programmer is also resident on the programmer circuit board and the program is located in memory locations between F200 H and F3FF H. As additional peripheral devices are developed, they will be assigned RAM and/or ROM storage in descending locations below F200 H.

4.2.3 User ROM

Space is provided on the main MCEM-8080 circuit board for user defined ROM storage. These ROMs, however, must be of the same type as that used for the monitor software. For instance, if the monitor has been supplied in type 3624 ROMs, all four ROM positions on the MCEM-8080 board must use the 3624 ROM. However, types 8708 and 8308 ROMs can be intermixed. User ROM storage starts at location 8400 H and extends to 87FF H (for 3624s) or 87FF H (for 8708/8308 ICs). HAL Communications provides ROM programming services to MCEM-8080 owners - please consult the factory if it is desired to program a PROM.

4.3 Input / Output (I/O) Assignments

Various input / output ports have been preassigned in the MCEM-8080 system. Among these are the 8251 USART IC, the 8255 PPI IC, the Keyboard / Video Display Unit option, and the PROM Programmer option.

4.3.1 8251 USART Integrated Circuit

The 8251 IC requires two input and two output ports; one input and one output port for control and one input and one output port for data. The control port has been assigned to port 0B H and the data port is assigned to 0A H.

4.3.2 8255 Programmable Peripheral Interface (PPI) IC

The 8255 IC requires four output and three input ports. Three of the ports map directly to the three parallel I/O ports of the IC. The fourth output port is used for PPI mode selection. The 8080 ports
corresponding to the 8255 ports are:

<table>
<thead>
<tr>
<th>8080A Port</th>
<th>8255 Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0C H</td>
<td>Port A input</td>
</tr>
<tr>
<td>Input 0D H</td>
<td>Port B input</td>
</tr>
<tr>
<td>Input 0E H</td>
<td>Port C input</td>
</tr>
<tr>
<td>Output 0C H</td>
<td>Port A output</td>
</tr>
<tr>
<td>Output 0D H</td>
<td>Port B output</td>
</tr>
<tr>
<td>Output 0E H</td>
<td>Port C output</td>
</tr>
<tr>
<td>Output 0F H</td>
<td>8255 Mode Select</td>
</tr>
</tbody>
</table>

NOTE: There is no Input port 0F H.

4.3.3 MCEM-KB/VDU Keyboard/Video Display Unit

The optional Keyboard / Video Display Unit requires one output port and two input ports. These are assigned as 8080A ports 0, 4, and 6. See the Keyboard / Video Display Unit manual for further information on this option.

4.3.4 MCEM-7K PROMPROG PROM Programmer

The optional PROM Programmer requires four output ports and three input ports. These are assigned as 8080A ports 80, 81, 82, and 83. See the PROM Programmer manual for further information on this option.
5. Operating Hints

Much of the versatility of the MCEM-8080 and the software monitor system will be best understood only after practical experience with the computer has been gained. This section of the manual contains some examples that will help to gain this needed experience.

5.1 Power-on Start Up

Several items should be checked out and possibly changed when initially installing the MCEM-8080. Among these are the power supplies, baud rate, I/O connections, etc. Once these items have been checked and corrected (if necessary), the following sequence can be used to "power-up" the system:

a. Set RUN/STOP switch to STOP
b. Set the DATA BUS OVERRIDE switch (DBO) to ON
c. Set the DATA BUS REGISTER switches to all zeros (front of rocker switch down)
d. Set the break point register (ADDRESS switches) to 8000 H
e. Apply DC power.

The address indicators should momentarily light and then extinguish. When all address lamps are on (logical "1"), the 8080A is being RESET. Once the address indicators are off, the WAIT lamp (far right-hand side of the circuit board) should come on and all DATA lamps should be off.

f. Set the RUN/STOP switch to RUN.
The 8080A will now run and automatically stop at location 8000 H (the setting of the break-point register).
g. Set the Data Bus Override (DBO) to OFF.
The DATA indicators should now indicate C3 H (1100 0011 B) which represents the first instruction in the software monitor, a JMP instruction.
h. Press and release the STEP switch. This causes the 8080A to begin executing the software monitor.
i. If the proper console device is operational (serial I/O if the Keyboard / Video Display Unit is not attached), the monitor will send the character sequence: "CR, LF, blank, blank, blank, period" to indicate that the monitor is ready to accept a command. The system is now ready to use.

NOTE:

The software monitor writes a jump to monitor instruction into location 0 (0, 1, 2) as it is initializing so that once the monitor has been entered (at 8000 H), a RESET (set the program counter to zero) will automatically cause an entry into the software monitor. Therefore, once an initial entry has been made, it is no longer necessary to go through the DBO = ON, Data Bus = 0, Break-point = 8000 H routine again. If power is removed or a user program writes data in location 0, 1, or 2, the automatic monitor entry on RESET will not operate.
5.2 Changing the Monitor Mode

The software monitor has several operational options. Some options should be selected before power turn-on and some after.

5.2.1 Baudot / ASCII Code

The serial I/O processing routines can be operated in either BAUDOT or ASCII units. The DSR terminal on the 8251 USART (pin 22 of the IC, terminal three of the I/O connector) is used to indicate to the monitor which code is being used. For ASCII code, terminal three is connected to ground; for BAUDOT, to +5 volts. The ASCII connection is normally furnished on the MCEM-8080. This connection must be made BEFORE power is applied to the MCEM-8080. If the state of the DSR connection is changed with power on and without first performing a RESET, the result may be indeterminate (for example transmitting a 5 bit character to an 8 bit USART, etc.). The placement of this jumper is shown in Figure B.1.

5.2.2 Half / Full Duplex (Echo / No echo)

In normal operation, characters that are input to the software monitor in the form of commands or parameters are retransmitted out to the output device so that the operator can view and verify them. This is called echoing of the character. If however, the MCEM-8080 is to be used in a system which automatically echos the input character, external to the MCEM-8080, this feature may be defeated (otherwise a double echo would result, causing repeating of the input characters). An example of a self-echoing I/O system is the serial loop-connected teleprinter in which the keyboard and printer are connected in series. To defeat the echo feature, FF H should be written into location 0D H. This location is normally initialized to 00 upon monitor entry and will be re-initialized upon each new entry into the monitor. The I (Insert) command can be used to perform the change by typing:

```
. I D 
     ___
000D 00,FF
000E 00S
```

The characters to be typed by the operator are underlined. If the double character transmission is occurring, it will appear as:

```
. I I D D 
   ___
000D 00,F,F,F
000E 00S
```

As above, underlined characters indicate those typed. Notice that only one S appeared because the echo has been turned off by that time.
5.2.3 Changing I/O Device Assignments

As discussed in section 3.2.8 with regard to monitor routines IOSET and IOCHK, an eight bit byte is reserved to hold the system I/O assignment. Changing this byte will change the device assignments. Use the I (Insert) command to change the byte as explained in section 3.1.3 and as in the preceding section (5.2.2). Remember that if the console device (Ports 0 and 1) is changed, the new console will be polled for new command strings. The IOBYT is set to zero (all devices set to serial I/O) upon initialization by the monitor. If, however, the Keyboard / Video Display Unit option is attached, all devices are set to it (the monitor checks for the presence of the Keyboard / Video Display Unit). In this case, IOBYT = 55.

5.3 Manually Writing a Memory Location

Two methods can be used to write a memory location, the easiest being to use the I (Insert) command. If it is impractical to use the I command, the following procedure can also be used:

a. Set the RUN / STOP switch to STOP
b. Set the DB0 switch to ON
c. Set the DATA switches to zero
d. Set the break-point register to the desired address
e. Press and release the RESET switch
f. Set the RUN / STOP switch to RUN
   (The Address indicators should now equal the desired memory address)
g. Set the DATA switches to the desired new memory value
   (number to be stored)
h. Press and release the MEMORY WRITE switch
i. Set the DB0 switch to OFF
j. Go to the desired address to proceed with program execution.

5.4 Manually Jumping to a Program Address

If, for some reason, the monitor JMP (Jump) command is unavailable to perform a jump to a desired program point, the following sequence can be used. Note that this is the same sequence as described in section 5.1 for initial entry into the monitor.

a. Set the RUN / STOP switch to STOP
b. Set the DB0 switch to ON
c. Set the DATA switches to zero
d. Set the break-point register to the desired address
e. Press and release the RESET switch
f. Set the RUN / STOP switch to RUN
   (The address indicators should now equal the desired address)
g. Set the DB0 switch to OFF
h. The program counter is now set. Press the STEP switch to begin program execution.
5.5 Manually Writing to an Output Port

At times it is desirable to be able to manually write data into an output port. This can be accomplished by:

1. Follow steps a. through h. of the previous two examples using the output port address instead of the memory address. Remember that I/O addresses are copied twice, once as the high order address and once as the low order address. For instance, Output Port 23 H is represented on the address bus (and break-point register) as 2323 H.

2. Press and release the OUTPUT WRITE switch.

3. Go to the desired address to proceed with program execution.

5.6 Using the Break-point Register for Debugging

The break-point register provides a mechanism for selectively stopping the 8080A. During the course of debugging a program, it may be desirable to determine when and if a particular string of instructions is executed. Setting the break-point register to this address (or I/O port) will provide this information. Another use of the break-point register allows use of the STEP switch as a "loop execute" switch. If the software being debugged contains a loop, the break-point register can be set to an address within the loop and the RUN / STOP switch set to RUN. At this point, the 8080A will stop each time it passes through the loop and will continue each time the STEP is pressed and released.

5.7 Using the E and R Commands for Debugging

The software monitor provides two very powerful commands to aid in debugging programs. The E (EXIT) command is a mechanism for saving complete context at any point in a user program and entering into the monitor. The R (RETURN) command allows return to the user program after restoring the complete context previously saved by the E command. The E command is invoked by executing RST 7. For example:

5.7.1 Manual EXIT Command

When a program is being debugged by manually stepping through the program steps (using the STEP switch), it is sometimes desirable to examine the contents of some of the internal registers of the 8080A (for instance the B, A, or PSW registers). However, since these registers are internal to the 8080A, they can not be directly examined on the console. The following procedure can be used to examine these internal registers:

a. Set RUN / STOP switch to STOP.
   (If the STEP switch is being used for debugging, the RUN / STOP switch is probably already set to STOP.)

b. Press and release the STEP switch as many times as necessary to bring the execution to the first byte of an instruction.
The E command can only be invoked during the fetch cycle of an instruction. For instance, JMP 23F2 is represented by

\[ C3 \\
F2 \\
2B \]

in memory. The E command can only be invoked when C3 is being read (indicated on the data indicators).

c. Set the DBO switch to ON.

d. Set the DATA switches to FF (all "ones" = RST 7).

e. Press and release the STEP switch once.

f. Set the DBO switch to OFF.

g. Set the RUN / STOP switch to RUN.

h. At this point, the following character stream should be typed on the console device:

\texttt{EXIT xxxx}

Remember that if the users program reassigns the I/O assignments or disturbs the USART mode, the console operation may be inhibited. 'xxxx' in the above character stream represents the next address after the one in which RST 7 was inserted.

i. Type

\[ .D2C,38] : \]

The saved register values will now be displayed in the following format:

\[ :0C \ 002C \ 00 \ SPL \ SPH \ PCL \ PCH \ L \ H \ PSW \ A \ E \ D \ C \ B \ SC \]

\[ :00 \ 003B \ 00 \]

where:

- \texttt{SPL} = low order stack pointer, stored at \texttt{2C}
- \texttt{SPH} = high order stack pointer, stored at \texttt{2D}
- \texttt{PCL} = low order program counter, stored at \texttt{2E}
- \texttt{PCH} = high order program counter, stored at \texttt{2F}
- \texttt{L} = L register, stored at \texttt{30}
- \texttt{H} = H register, stored at \texttt{31}
- \texttt{PSW} = Processor Status Word, stored at \texttt{32}
- \texttt{A} = A register, stored at \texttt{33}
- \texttt{E} = E register, stored at \texttt{34}
- \texttt{D} = D register, stored at \texttt{35}
- \texttt{C} = C register, stored at \texttt{36}
- \texttt{B} = B register, stored at \texttt{37}
- \texttt{SC} = Sum Check Character.
The format of the Processor Status Word (PSW) is:

\[
\begin{array}{cccccccc}
D7 & D6 & D5 & D4 & D3 & D2 & D1 & D0 \\
S & Z & 0 & AC & 0 & P & 1 & CY \\
\end{array}
\]

where S, Z, AC, P, and CY are the corresponding flags.

j. If, at this point, it is desired to set a particular register to a new value, the I command can be used. For example, .133 will allow the A register to be modified.

k. After the registers have been examined and changed (if desired), the R (RETURN) command can be used to return to the original program. The return address, however, is not the same as the saved address in this case. (Recall that a RST 7 instruction was inserted instead of a valid instruction and the saved address is one more than the address of the substituted RST 7.) To use the R command, mentally calculate

\[ yyyyy = xxxx - 1 \]

where "xxxx" is the address stored and the address displayed on the console after the E identifier (see step h). Now type .Ryyyy

Leading zeros can be omitted.

5.7.2 Interrupt EXIT Command

The monitor software and the MCEM-8080 hardware combine to cause interrupts to execute E commands (the interrupt vector is RST 7). Therefore, if it is desired to execute an E command, it can be instituted by placing +5 volts on the INTR (INTERRUPT) line (pin 4 of the 36 pin I/O connector). If the user has not disabled the interrupt or written into low memory (below 3F H), the following should appear on the console device:

\[ \text{EXIT xxxx} \]

All of the techniques for examining and modifying registers listed above may be used. However, when a R command is desired, it is not necessary to recompute the address because the interrupt method saves the proper address.

A return to zero command, R0

will return the CPU to the program, restoring the registers to their states just prior to the interrupt.
5.7.3 Programmed EXIT Command

Many programs have error testing subroutines and the E command can be used to perform error exits from these programs. If a RST 7 instruction is inserted in the program in the error branch, it will cause the following to be displayed on the console:

```
EXIT xxxx
```

If the user tabulates the addresses of all of the RST 7 instructions, it is then a simple matter to correlate the "xxxx" typed against the list. The techniques explained previously can be used to evaluate and modify the CPU parameters that existed at the time of the interrupt (RST 7).

A useful feature that results when the RST 7 instruction is used as the E command driver is prevention of transfer to non-existent areas of memory. Since non-existent memory is generally FF, a RST 7 will be immediately encountered and control then transferred to the E command process. This feature helps prevent the "run away" condition that could conceivably rewrite all of memory otherwise.
6. Program Examples

This section contains several example programs to demonstrate the features and capabilities of the MCEM-8080. In no case should any of these example programs be considered "optimum" or "required procedure". They are, however, working routines that can be used as starting points for more elaborate programs, as subroutines in user programs, or simply for ideas as to typical procedures to be used with the MCEM-8080A. Except as noted, all example programs will operate in the basic '1 K' memory furnished with the MCEM-8080.
7. Software Monitor Listings

The following pages contain a complete listing of the MCEM-8080 Microcomputer System software monitor. This listing is provided for the sole benefit of owners of HAL Communications Model MCEM-8080 systems and remains the sole property of HAL Communications Corp. The listing may not be duplicated for any use without the prior permission of HAL Communications Corp. HAL Communications reserves the right to make changes, additions, or deletions to these computer programs without prior notification or obligation to incorporate such changes in prior versions of the programs.
The following are valid commands for the
MCEM monitor:

**L<BIAS>**
- Load a hex formatted file.
- Check for sumcheck errors.
- And type an 'X' if error.
- The value of the bias is added to the load address before the data is written.
- To memory, the reader device is used as input.

**D<START>,<END>**
- Dump memory from <start> memory locations from <start> to but not including <end> is dumped.
- The format of this dump is compatible with the load command so memory areas can be dumped and loaded at a later time. The punch device is used as output.

***The form for both loads and dumps is:***

```
<L<LENGTH>,<ADDRESS>,<TYPE>,<DATA,BYTES>,<SUMCHECK>
```

- All records are preceded by a colon. All characters between the sumcheck and the colon are ignored. All spaces are ignored (i.e., spaces can be contained in the record with no effect).
- **<LENGTH>** is the number of data bytes in the record (00-FF).
- **<ADDRESS>** is the load address (0000-FFFF).
- **<TYPE>** is not used at this time and is 00 (ignored).
- **<DATA,BYTES>** are the actual data. **<LENGTH>** of them.

COPYRIGHT 1976 (C)

BY

HAL COMMUNICATIONS CCPF
807 E GREEN STREET
URBANA, ILLINOIS 61801

MCEM MONITOR — DECEMBER 1976
HAL COMMUNICATIONS CCPF.

THIS PROGRAM IS CONTAINED IN PROMS (03:0) (FIRST HALF)
AND (02:0) (SECOND HALF)
<SCHECK> IS THE NEG SUM OF ALL BYTES (ADDRESS
IS TWO BYTES (HIGH AND LOW))
EXCLUDING THE COLCN, I.E. THE SUM OF
ALL BYTES INCLUDING THE SUMCHECK IS 0.
FOR NO ERROR, DURING LOAD, IF A
SUMCHECK ERROR IS ENCOUNTERED, AN 'X'
IS PRINTED ON THE CONSOLE DEVICE.

<LOCATION> INSERT IN THE SPECIFIED
LOCATION. THE PREVIOUS
CONTENTS OF THE LOCATION
IS TYPED. THE CONSOLE DEVICE IS
USED FOR INPUT AND OUTPUT.
A ' ' IS USED TO OPEN A CELL. I.E.

0065 58,74

WOULD BE THE FORMAT FOR CHANGING
LOCATION 65 FROM 56 (OLC VALUE) TO
74 (NEW VALUE)

<LOCATION> A UNCONDITIONAL
GO TO <LOCATION>. A UNCONDITIONAL
JUMP IS EXECUTED TO THE IN-
DICATED LOCATION. INTERRUPT (EXIT
COMMAND) IS ENABLED BEFORE THE
JUMP.

<LOCATION> RETURN TO LOCATION. A RESTORE
REGISTER JUMP IS EXECUTED TO
LOCATION UNLESS LOCATION = 0 IN
WHICH CASE, THE CONTENTS OF THE
PCSAV IS USED AS THE ADDRESS.
INTERRUPT (EXIT COMMAND) IS ENABLED
BEFORE THE RETURN.

EXIT COMMAND: A RST 7 WILL EXECUTE AN EXIT
COMMAND. ALL REGISTERS ARE SAVED
IN_RAM FOR EXAMINATION AND/OR
MODIFICATION. THE R COMMAND IS
THE COMMAND OF THE EXIT COMMAND
AND WILL RETURN THE PROCESSOR TO
ITS ORIGINAL STATE. AN EXIT
COMMAND SHOULD BE PERFORMED PRIOR
TO AN R COMMAND BECAUSE THE
STORED VALUE OF THE SP SHOULD BE
INTACT (I.E. POINT TO A VALID
STACK AREA) (OTHER THAN THE MONITOR
STACK).
I/O ASSIGNMENT: An eight bit I/O assignment byte is stored at location IBYTE (03). The value of this byte directs the console, reader, and punch (list to 10) to one (each) of four possible I/O devices. The format of I/O byte is:

D7 D6 D5 D4 D3 D2 D1 D0
/LIST DEV/PUNCH DEV/READ DEV/CONSOLE / 

Each device can be assigned to:

00: Serial input/output (8251, ASCII or baudot)
01: Keyboard display module (optional device)
10: Parallel input/output (8255)
11: User input/output (user defined routines)

All numeric entries can be terminated by typing any of the characters G-Z

TITLE 'MCEM-8680 MONITOR 1.1'

CONSTANT DEFINITIONS

KEYBOARD DISPLAY I/O CONSTANTS

F6C0 DSPCK EQU C6803H ;DISPLAY PRESENT CHECK
F6CA SDFIO EQU DSPCK+0AH ;SET DISPLAY IO BYTE ENTRY
F6C1 KBIN EQU DSPCK+1 ;KBIN ENTRY
F6C7 KBST EQU DSPCK+7 ;KBST ENTRY
F6C4 DSPOT EQU DSPCK+4 ;DISPLAY OUTPUT ENTRY

SERIAL I/O CONSTANTS

C83C CSR EQU 80H ;CSR BIT IN UART (0=BAD)
C83F URTHM EQU 63H ;LATH MODE FOR BAUDOT
001F LTHS EQU 1FH ;BAUDOT LETTERS
0018 FIGS EQU 1BH ;BAUDOT FIGURES
C037 BCASI EQU 7 ;BAUDOT CASE (INPUT)
C03F BCASQ EQU 6EH ;BAUDOT CASE (OUTPUT)
00C2 FXRDY EQU 2H ;RX READY TEST MASK
C001 TXRDY EQU 1H ;TX READY TEST MASK
0027 TXRXE EQU 27H ;TX RX ENABLE
C0FA UTIMO EQU 0FEAH ;7 BITS, EVEN_PARITY, 2 STOP
C0EE LARTI EQU 6EH ;INITIAL UART MODE WORD
C055 UARTR EQU 55H ;UART RESET COMMAND
C002 URTC EQU 6BH ;UART CONTROL PORT
80EC MACRO ASSEMBLER, VER 2.2 MC80-8080 MC80TC 1.1 ERRORS = 0 PAGE 4

CCCA URTDA EQU 0AH ; UART DATA PORT

; PARALLEL I/O CONSTANTS

COAE PARMD EQU 0A6H ; PPI MODE FOR PARALLEL I/O
COAF PARCT EQU 0FH ; PARALLEL CONTROL PORT
CO02 PIDY EQU 2 ; PARALLEL INPUT READY BIT
COGE PSTA EQU 0EH ; PARALLEL STATUS PORT
CO0C RORDY EQU 80H ; PARALLEL OUTPUT READY BIT
CO0C P0RT EQU 0CH ; PARALLEL OUTPUT PORT

; USER I/O CONSTANTS

0640 LSRIN EQU 40H ; ENTRY FOR USER INPUT
0043 USRGT EQU 43H ; ENTRY FOR USER OUTPUT
004E USRST EQU 46H ; ENTRY FOR USER INPUT STATUS

; MISC MONITOR CONSTANTS

CCCA LF EQU 0AH ; ASCII LINE FEED
G00D CR EQU CDH ; ASCII CARRIAGE RETURN

; MONITOR MEMORY ALLOCATIONS

CCCE MG21 EQU 5 ; NUMSIZE STRC
CC06 MG22 EQU 6 ; NUMSIZE STORE (HIGH)
CG04 IFLAG EQU 4 ; INCIR REF TO EIT_FLAG
C02E KSAP EQU 38H ; START OF RESTART STORAGE
C02C MDST EQU KSAP-12 ; MONITOR STACK AREA
C02E PGSAC EQU PGSAC-10 ; LOCATION OF PC SAVE
C022 PSSAC EQU PGSAC-6 ; LOCATION OF PSW SAVE
C002 IOBYT EQU 3 ; I/O ASSIGNMENT STORAGE
G00D ECHCM EQU 0CH ; STERAGE FOR ECHM MODE FLAG

; MONITOR RAM ALLOCATION

C-2 JUMP TO MONITOR C00E
3 IOBYT (I/O DEVICE ASSIGNMENT)
A IFLAG (INCIR REF TO EIT_FLAG)
5 MG21 (LOW CREIF MEMORY SIZE BYTE)
6 MG22 (HIGH CREIF MEMORY SIZE BYTE)
7 ECHC1 (BAUDOUT CASE FOR INPUT)
8-9 CRSRP (CURSOR POSITION FOR CRT DISPLAY)
A CRSAV (UPSET REGISTER COPY FOR CRT DISPLAY)
B HIOD (CHARACTER HIDDEN UNDER CURSOR)
C RPTFG (KEPEAT MODE (KEYBOARD) FLAG)
D ECHCM (ECHCM MODULE MAP FLAG)
E BCASO (BAUDOUT CASE FOR OUTPUT)
F SFAKE
10-2C MONITOR STACK
MACRO DEFINITIONS

<table>
<thead>
<tr>
<th>TEST</th>
<th>MACRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>C32880</td>
</tr>
<tr>
<td>8000</td>
<td>C35381</td>
</tr>
<tr>
<td>8003</td>
<td>C3AD01</td>
</tr>
<tr>
<td>6C66</td>
<td>C36E61</td>
</tr>
<tr>
<td>6C75</td>
<td>C3F180</td>
</tr>
<tr>
<td>6C7F</td>
<td>C3F981</td>
</tr>
<tr>
<td>6012</td>
<td>C3182</td>
</tr>
<tr>
<td>6C15</td>
<td>C36F82</td>
</tr>
<tr>
<td>6018</td>
<td>C35382</td>
</tr>
<tr>
<td>601E</td>
<td>C35882</td>
</tr>
<tr>
<td>601E</td>
<td>C31CE3</td>
</tr>
<tr>
<td>6021</td>
<td>C35A83</td>
</tr>
<tr>
<td>6024</td>
<td>C37283</td>
</tr>
<tr>
<td>6027</td>
<td>C3CA81</td>
</tr>
<tr>
<td>8G2A</td>
<td>C9</td>
</tr>
</tbody>
</table>

INITIALIZE UART

802B EEO:

E60:

LXI   H,40H ;CLEAR MONITOR RAM AREA

802E EGO:

LXI   H,40H ;CLEAR MONITOR RAM AREA

802F 74 ;MONITOR ENTRY

8030 C32E80 ;CONSOLE INPUT

8033 3EC3 ;CONSOLE OUTPUT

8035 3660 ;MEMORY SIZE CHECK

803E 2E30 ;MESSAGE TYPE

803D 36C3 ;MESSAGE TYPE

803F 23 ;MESSAGE TYPE

8040 3645 ;MESSAGE TYPE

8042 23 ;MESSAGE TYPE

8043 3681 ;MESSAGE TYPE

8045 3EA6 ;MESSAGE TYPE

8047 D38 ;MESSAGE TYPE

8049 3EE0 ;MESSAGE TYPE

804B 214000 ;CLEAR MONITOR RAM AREA

802E 2D ;CLEAR MONITOR RAM AREA

802F 74 ;CLEAR MONITOR RAM AREA

8030 C32E80 ;CLEAR MONITOR RAM AREA

8033 3EC3 ;CLEAR MONITOR RAM AREA

8035 3660 ;CLEAR MONITOR RAM AREA

803E 2E30 ;CLEAR MONITOR RAM AREA

803D 36C3 ;CLEAR MONITOR RAM AREA

803F 23 ;CLEAR MONITOR RAM AREA

8040 3645 ;CLEAR MONITOR RAM AREA

8042 23 ;CLEAR MONITOR RAM AREA

8043 3681 ;CLEAR MONITOR RAM AREA

8045 3EA6 ;CLEAR MONITOR RAM AREA

8047 D38 ;CLEAR MONITOR RAM AREA

8049 3EE0 ;CLEAR MONITOR RAM AREA

804B 214000 ;CLEAR MONITOR RAM AREA
8080 MACRO ASSEMBLER, VER 2.2 MCLM-8080 MONITOR 1.1 ERRORS = 0 PAGE 6

8048 D3CD OUT URTCT ; ISSUE MODE
804D 3E55 MVI A, UARTR ; UART RESET INSTRUCTION
804F D30B OUT URTCT ; ISSUE INTERNAL & ERROR RESET
8051 3FFA MVI A, URM0 ; SET FINAL MODE
8053 D30B CUT URTCT ; ISSUE PREDEFINED PRESENTLY
8055 CB00 IN URTCT ; ISSUE LARM MODE INSTRUCTION
8057 E680 ANI DSR ; DSR=0 IS BAUDOT (+5VOLTS)
8059 C5480 JNZ BG2 ; NO, MODE IS ASCII
805C 3E55 MVI A, UAPTR ; RESET CSART
805E D30B CUT URTCT
8060 3E63 MVI A, URTBM ; SET MODE FOR 5 EIT, 1 1/2
8062 D30B OUT URTCT ; STCP, NO PARITY, X64 CLK
8064 3E27 JNZ ; TMRXE : ENABLE TX AND RX
8066 D30B COT URTCT
8068 312C00 LXI SP, MONST ; SET STACK POINTER
806E 3AC0F8 LDA DFCK ; IS DISPLAY ATTACHED?
806E FEAS CPI Q5H ; THIS IS THE TEST BYTE
8070 CCCAF8 CZ SDPID ; USE THE DISPLAY ROUTINE

END OF INITIALIZATION SEQUENCE

MONITOR MAIN LOOP

8073 312C00 LXI SP, MONST ; RESET MONITOR STACK POINTER
8076 21F6E3 LXI H, PMTMG ; FRCMPT WITH PERIOD
8079 CD9A83 CALL TYPMG
807C CDEB81 CALL ECHO ; GET INPUT
807F E6C2 MVI B, 2' ; DEFAULT PARAMETER COUNT
8081 FE4C CPI 'L' ; LOAD
808C CACEF8 JZ LEAD ; YES
808E FE4A CPI 'J' ; START EXECUTION (JUMP)?
808E CAE880 JZ GO ; YES
808E FE49 CPI 'I' ; INSERT?
808E CADD60 JZ INSR ; YES
8090 E632 CPI 'R' ; RETURN TO PROGRAM?
8092 CAF6E1 JZ RETRN ; YES
8096 FE46 CPI 'D' ; DISPLAY/DUMP?
8097 C273D0 JNZ MAIN ; NO, EAD COMMAND, TRY AGAIN

COMMAND SERVICE ROUTINES

DISPLAY (DUMP) MEMORY TO SERIAL OUTPUT
THE FORMAT OF THIS DUMP IS COMPATIBLE
WITH THE LOAD (HEX) ROUTINE

<table>
<thead>
<tr>
<th>DEC</th>
<th>COD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC5A</td>
<td>CDIC83</td>
<td>CALL EXPRH ; GET 2 PARAMETERS</td>
</tr>
<tr>
<td>EC5C</td>
<td>D1</td>
<td>POP D ; PUT &lt;END&gt; IN DL</td>
</tr>
<tr>
<td>EC9E</td>
<td>E1</td>
<td>POP H ; PUT &lt;START&gt; IN FL</td>
</tr>
<tr>
<td>EC9F</td>
<td>70</td>
<td>MOV A,L ; LCBYTE OF CURRENT POINTER</td>
</tr>
<tr>
<td>E044</td>
<td>C610</td>
<td>ADD 16 ; ENC-TEST_ADDRESS</td>
</tr>
<tr>
<td>ECA2</td>
<td>47</td>
<td>MOV B,A ; GOES INTO (CE)( )</td>
</tr>
<tr>
<td>E0A3</td>
<td>7C</td>
<td>MOV A,H ; FIGE BYTE OF CURRENT POINTER</td>
</tr>
<tr>
<td>E0A4</td>
<td>CEC0</td>
<td>ADI C ; ADD CARRY OF PREVIOUS ADI 16</td>
</tr>
<tr>
<td>E0A5</td>
<td>AF</td>
<td>MOV C,A ; (CB)=(HL)+(16)</td>
</tr>
<tr>
<td>ECA7</td>
<td>D7380</td>
<td>JC MAIN ; EXIT IF WRAP AROUND</td>
</tr>
<tr>
<td>ECA8</td>
<td>70</td>
<td>MOV A,E ; LOW BYTE OF &lt;END&gt;</td>
</tr>
<tr>
<td>ECAE</td>
<td>9C</td>
<td>SUB B ; E-(B+16)</td>
</tr>
<tr>
<td>ECA4</td>
<td>47</td>
<td>MOV B,A ; SAVE DISPLACEMENT-16</td>
</tr>
<tr>
<td>E0AD</td>
<td>7A</td>
<td>MOV A,D ; FIGE BYTE OF &lt;END&gt;</td>
</tr>
<tr>
<td>E0AE</td>
<td>56</td>
<td>MOV B,C ; D-C-CY OF E-(L+16)</td>
</tr>
<tr>
<td>E0AF</td>
<td>D2E980</td>
<td>JNC DISP5 ; SKIP IF &gt;15 LEFT</td>
</tr>
<tr>
<td>ECD2</td>
<td>78</td>
<td>MOV A,B ; UPDATE RECORD LENGTH</td>
</tr>
<tr>
<td>E0E3</td>
<td>C610</td>
<td>ADD 10H ; (L-16)+16</td>
</tr>
<tr>
<td>E0E5</td>
<td>47</td>
<td>MOV B,A ; E=DISPLACEMENT(IF &lt;16)</td>
</tr>
<tr>
<td>E0E6</td>
<td>2880</td>
<td>JMP DISP2 ; SKIP SINCE &lt;16</td>
</tr>
<tr>
<td>E0E8</td>
<td>0610</td>
<td>MOV A,10H ; ECO 16 BYTES PER ITERATE</td>
</tr>
<tr>
<td>E0F7</td>
<td>E6</td>
<td>PUSH H ; SAVE ADDRESS FOR MSG</td>
</tr>
<tr>
<td>E0FC</td>
<td>218483</td>
<td>LX L,H,CREC ; CR,LE           ...</td>
</tr>
<tr>
<td>E0F7</td>
<td>2DBD82</td>
<td>CALL TYPMP ; TYPE IT</td>
</tr>
<tr>
<td>E0CC</td>
<td>E1</td>
<td>POP H ; RETRIEVE ADDRESS</td>
</tr>
<tr>
<td>E0C3</td>
<td>D5</td>
<td>PUSH D ; SAVE &lt;END&gt;</td>
</tr>
<tr>
<td>E0C4</td>
<td>16C0</td>
<td>MVI D,0 ; ZERC SUMCHECK BYTE</td>
</tr>
<tr>
<td>E0C6</td>
<td>70</td>
<td>MOV A,B ; GET LENGTH</td>
</tr>
<tr>
<td>E0CC</td>
<td>CDA082</td>
<td>CALL BYTOP ; PRINT THE LENGTH</td>
</tr>
<tr>
<td>E0CC</td>
<td>CDE482</td>
<td>CALL WROD ; TYPE BEGIN ADDRESS</td>
</tr>
<tr>
<td>E0CD</td>
<td>AF</td>
<td>XRA A ; MAKE TYPE ZERO</td>
</tr>
<tr>
<td>E0CE</td>
<td>CDA082</td>
<td>CALL BYTOP ; CUTFUT TYPE</td>
</tr>
<tr>
<td>E0D1</td>
<td>78</td>
<td>MOV A,B ; TEST FOR FINISHED</td>
</tr>
<tr>
<td>E0D2</td>
<td>17</td>
<td>AN A ; ZERC LENGTH IS END</td>
</tr>
<tr>
<td>E0D3</td>
<td>7A</td>
<td>CALL BYTOP ; SET FLAGS ; CY=0</td>
</tr>
<tr>
<td>E0D3</td>
<td>CA7830</td>
<td>JZ MAIN ; GET NEXT COMMAND IF DONE</td>
</tr>
<tr>
<td>E0D6</td>
<td>7E</td>
<td>MOV A,M ; TYPE RECORDS</td>
</tr>
<tr>
<td>E0D7</td>
<td>CDA082</td>
<td>CALL BYTOP ; PRINT IT</td>
</tr>
<tr>
<td>E0CA</td>
<td>22</td>
<td>INX H ; BUMP POINTER</td>
</tr>
<tr>
<td>E0CC</td>
<td>D5</td>
<td>DCR B ; BUMP PARM COUNTER</td>
</tr>
<tr>
<td>E0CE</td>
<td>C2D680</td>
<td>JNZ DISP3 ; FORM SUMCHECK</td>
</tr>
<tr>
<td>E0CF</td>
<td>AF</td>
<td>XRA A ; FORM SUMCHECK</td>
</tr>
<tr>
<td>E0EC</td>
<td>96</td>
<td>SUB D ; SUMCHECK = -D</td>
</tr>
<tr>
<td>E0E1</td>
<td>CDA082</td>
<td>CALL BYTOP ; CUTFUT SUMCHECK</td>
</tr>
<tr>
<td>E0E4</td>
<td>D1</td>
<td>POP D ; RETRIEVE &lt;END&gt; SAVED</td>
</tr>
</tbody>
</table>
GO ROUTINE

EXITS VIA JUMP TO THE USERS ROUTINE
AFTER RESTORING THE USERS REGISTERS

GO:

CALL WRDIN ;GET JUMP ADDRESS

ENABLE EXIT COMMAND

JUMP THERE

INSERT DATA COMMAND
ALLOWS HEX DATA TO BE SEQUENTIALLY ENTERED
THE INSERT FUNCTION IS TERMINATED BY A CHAR
BETWEEN G AND Z.

INST:

CALL WRDIN ;GET INSERT ADDRESS

INST:

PUSH F ;SAVE ADDRESS

CALL TYP1

POP F ;RETRIEVE ADDRESS

CALL WRDOT ;TYPE ADDRESS

MOV A,M ;GET PRESENT CELL VALUE

CALL BTOT1 ;TYPE IT

CALL EIT ;CHECK IF CHANGE IS DESIR\ED

USE , TO OPEN THE CELL

SKIF BYTE READ IF NOT

CALL EXTIN ;GET NEW VALUE (EXIT FROM HERE)

STORE IT

MOVE TO NEXT LOCATION

START NEXT LINE

LOAD HEX ROUTINE. THIS ROUTINE IS COMPATIBLE WITH
BOTH THE DATA GENERATED BY THE D COMMAND AS WELL
AS THE HEX FILES GENERATED BY THE ASSEMBLERS AND
COMPILERS.

LOAD:

GET BIAS VALUE

USE NORMAL PARAM

SEARCH FOR "":

ZERO SUMCHECK
CALL BYTIR ;GET RECORD LENGTH
JZ MAIN ;EXIT IF ZERO LENGTH
PDP P ;GET BIAS
PUSH PSW ;SAVE RECORD LENGTH
CALL WPDIR ;GET LOAD ADDRESS
DAD B ;ADD BIAS
PDP PSW ;RETRIEVE RECORD LENGTH
PUSH A ;SAVE BIAS
MOV B,A ;PUT LENGTH IN B
CALL CDC682 ;GET RECORD TYPE.
LD1:
CALL BYTIR ;GET BYTE
INX M,A ;STORE IT IN MEMORY
MOV H ;MOVE TO NEXT ADDRESS
DEC B ;DECREMENT RECORD LENGTH
JNZ LD1 ;LOOP UNTIL RECORD DONE
CALL BYTIR ;GET SUMCHECK BYTE
JZ LD0 ;SUMCHECK WAS ZERO
LD0
MOVA C,'X' ;TYPE E FOR SUMCHECK ERROR
CALL CDC682
JMP LD0

STACK FORMATS FOR RESTARTS:
USER STACK
PCF
PFL
H
L
A
PSW

MONITOR STACK
B (37)
C (36)
D (35)
E (34)
A (33)
PSW (32)
H (31)
L (30)
PCH (2F)
PCL (2E)
SPM (2D)
SPL (2C)

MONITOR STACK STARTS AT 2B

EXIT COMMAND PROCESSOR. WE GOT HERE BY DOING A
RST 7 (INIT SEQUENCE WRITES THIS ADDRESS INTO RAM
AT 3EH). GC_PC IS ALREADY ON USER STACK. SAVE HL
AND PSW ON USER STACK TO MAKE ROOM TO WORK. SAVE
REMAINING REGISTERS IN RAM IN SAVE AREA AND THEN
TRANSFER THOSE STILL ON USER STACK TO SAVE AREA.
A TOTAL OF 6 BYTES OF USER STACK ARE NEEDED. USER
STACK MUST NOT BE THE SAME AS THE MONITOR STACK
ONCE FULL RECOVERY HAS BEEN ACCOMPLISHED, WRITE THE
ADDRESS THAT GOT US HERE ON THE CONSOLE.

RSTRT:

8145  E5  PUSH  H  ;SAVE HL ON USER STACK
8146  F5  PUSH  PSW  ;SAVE PSW
8147  210000  LXI  H,0  ;GET SP VALUE IN HL
814A  35  DAD  SP
8148  213800  LXI  SP,RSAV;SET UP SP FOR SAVE AREA
814E  C6  PUSH  B  ;SAVE BC
814F  D5  PUSH  D  ;SAVE DE
8150  5E  MOV  E,M  ;MOVE A,PSW TO SAVE AREA
8151  23  INX  H
8152  56  MOV  D,M
8153  23  INX  H
8154  D5  PUSH  D  ;THEY ARE IN DE
8155  5E  MOV  E,M  ;MOV HL TO SAVE AREA
8156  23  INX  H
8157  56  MOV  D,M
8158  23  INX  H
8159  D5  PUSH  D  ;THEY ARE IN DE
815A  5E  MOV  E,M  ;MOV PC TO SAVE AREA
815B  23  INX  H
815C  56  MOV  D,M
815D  23  INX  H
815E  D5  PUSH  D  ;THEY ARE IN DE
815F  E5  PUSH  H  ;SAVE SP IN SAVE AREA
8160  21B883  LXI  H,RSMG;TYPE EXIT IDENTIFIER
8163  CG9A83  CALL  TYPMG
8166  2A2EC0  LHLD  PCSAV;GET THE CALLING ADDRESS
8169  301A281  CALL  WRDOUT;TYPE IT ON CONSOLE
816C  C37380  JMP  MAIN;GO TO MAIN LOOP

RETURN PROCESSOR RECOVER ALL REGISTERS FROM THE
SAVE AREA AND THEN RETURN TO WHERE EVER THE PC SAV
INDICATES. ONE PARAMETER IS GATHERED, AND IF IT
IS NON-ZERO, ITS VALUE IS SUBSTITUTED FOR THE PC
SAV BEFORE RETURNING.

RETRN:

816F  CD2683  CALL  WRDIN;GET ONE PARAMETER IN HL
8172  7C  MOV  A,H  ;TEST IF IT IS ZERO
8173  BE  CRA  L
8174  C47A81  JZ  RETR1;HL = 0, DONT WRITE PC SAV
8177  222000  SHLH  PCSAV;STORE NEW DESTINATION VALUE

RETR1:

817A  E1  POP  H  ;GET STACK POINTER
817B  D1  POP  D  ;GET PC
817C  C1  POP  B  ;GET HL
8060 MACRO ASSEMBLER, VER 2.2 MCEM-8060 MONITOR 1.1 ERRORS = 0 PAGE 11

817C FS SPHL ; RESTORE STACK POINTER
817C DS PUSh D ; PUT PC ON STACK
817E CS PUSh H ; PUT HL ON STACK (TEMP)
8180 212200 LXI M, PSAlV ; MOVE A, PSW FROM STACK
8193 5E MOV E, M ; USE DE AS BUFFER PAIR
8194 23 INX H
8195 56 MOV D, M
8196 23 INX H
8197 DS PUSh D ; SAVE FOR A WHILE ON SACK
8198 5E MOV E, M ; RECEIVE DE
8199 23 INX H
819A 56 MOV D, M
819B 23 INX H
819C 4E MOV C, M ; RECEIVE BC
819D 23 INX H
819E 46 MOV E, M
819F F1 POP PSW ; RECEIVE A, PSW
81A0 E1 POP H ; RECEIVE HL
81A1 FB EI ; ENABLE INTERRUPTS (RST 7)
81A2 CS RET ; RECEIVE PC

---

CCNSCLE INPUT ROUTINE. THE DEVICE INDICATED BY
THE BOTTOM TWO BITS OF IOBYT IS CALLED AND EXPECTED
TO RETURN A CHARACTER. THE CHARACTER RETURNS IN A

81A3 3A3C00 LDA IOBYT ; GET IO ASSIGNMENT
81A4 3C0300 IURCH: LDA IOBYT ; GET IO ASSIGNMENT
81A5 E63E ANI 3 ; TEST BOTTOM BITS
81A6 CA3882 JZ CHI ; ZRC IS SERIAL INPUT
81A7 3D DCR A
81A8 CAC1F8 JZ KBI N ; CNE IS KEYBOARD INPUT
81A9 3D DCR A
81A0 C240C0 JNZ USRIN ; NOT TWC IS THREE (PARALLEL)

---

PARALLEL INPUT ROUTINE. 7-BIT ASCII IS EXPECTED
ON THE B PORT OF THE PPI. TRUE DATA IS EXPECTED.

81AC DECE IN PSSTAT ; GET THE PORT STATUS
81AE E6C2 ANI PIRDY ; TEST INPUT READY BIT
81AF CA381 JZ PARIN ; IORP TIL IT IS READY
81A0 DPOD IN PINPT ; READ THE CHARACTER
81AC CS RET

READER INPUT. BITS 2 AND 3 OF IOBYT ARE USED TO DIRECT
CONTROL TO THE PROPER DEVICE. AN ASCII CHAR IS
RETURNED IN THE A REGISTER.

81AD 3A3C00 LDA IOBYT ; GET IO ASSIGNMENT
81AE 1F RAR ; MOVE THE BITS IN QUESTION
808C MACRO ASSEMBLER, VER 2.2 MCEM-8080 MONITOR 1.1 ERRORS = 0 PAGE 12

81E1 1F  KAR  ; TIE THE BOTTOM TWO BITS
81E2 C36681  JMP  IBRCH  ; BRANCH TO PROPER ROUTINE

; CONSOLE ECHO ROUTINE. GET CHARACTER AND THEN IF
; THE MODE IS ASCII, ECHO IT.

81E5  CALL  CI  ; GET CHARACTER FROM CONSOLE
81E6  MOV  C,A  ; MOVE IT INTO C FOR OUTPUT
81E7  F5  PUSH  PSW  ; OUTPUT MIGHT DESTROY IT
81E8 A0D00  LDA  ECHOM  ; SHOULD WE ECHO?
81E9  TEST  ; ZERO IS YES
81E10  A7  +  ANA  A  ; SET FLAGS, CY=0
81E11  C2C881  JNZ  ECHO0  ; DON'T ECHO ANYTHING
81E12  3AC4C0  LDA  IFLAG  ; DON'T ECHO IF INDIRECT MODE
81E13  TEST  ; ZERO IS YES
81E14  A7  +  ANA  A  ; SET FLAGS, CY=0
81E15  CCCC81  CO  ; ECHO IT IF NOT INDIRECT
81E16  ECHO0:  POP  PSW  ; RECOVER THE CHARACTER
81E17  C5  RET

; WORD OUTPUT RTN
; OUTPUTS A ASCII CODED HEX DIGITS

81C9  CALL  EE03  ; EECODES AND PRINTS HEX
81CA  C5  PUSH  B  ; SAVE BC REGISTERS
81CB  7C  MOV  A,H  ; 16 BIT # IN (HL)
81CC  C078B3  CALL  BTCT1  ; SPIT OUT HIGH BYTE
81CD  7C  MOV  A,L  ; GET LOW BYTE
81CE  C078B3  CALL  BTOT1  ; SPIT OUT LOW BYTE
81CF  CI  POP  BC  ; RETRIEVE BC REGISTERS

; SPACE OUTPUT RTN

81D4  0E20  ; CONSOLE OUTPUT ROUTINE. THE IOBYTE IS EXAMINED
81D5  MVI  C,, ' '  ; TO DETERMINE THE PROPER IO DEVICE. THE VALUE OF
81D6  3A0300  LDA  IOBYTE  ; THE C REGISTER IS THEN OUTPUT VIA THAT DEVICE.
81D7  CBRCH:  JZ  CH0  ; ZER0 IS SERIAL IO
81D8  E603  ANI  3  ; BRANCH BASED ON BOTTOM
81DB  CA4882  JZ  CH0  ; ZER0 IS SERIAL IO
81DE  3C  DCR  A  ; CONSOLE OUTPUT ROUTINE. THE IOBYTE IS EXAMINED
81DF  CA44F8  JZ  DSPOT  ; CNE IS CRT DISPLAY
81EE  3D  DCR  A  ; CONSOLE OUTPUT ROUTINE. THE IOBYTE IS EXAMINED
81EF  C24300  JNZ  USROT  ; NOT TWO IS THREE (PARALLEL)
PARALLEL OUTPUT ROUTINE. ALPHA CHARACTERS ARE
OUTPUT AND CONTROL CHARACTERS ARE NOT. A LINE FEED IS
USED TO DETERMINE THE END OF THE LINE AND DRIVES A
RUN OUTPUT. THIS INTERFACE IS INTENDED TO DRIVE
EASY LINE PRINTERS.

81E6        DECE        IN  PSTAT       ;GET THE PORT STATUS
81E6        E800        ANI  PCSRY       ;TEST OUTPUT READY BIT
81EA        CAE681      JZ  PAROT       ;LOCP TIL IT IS READY
81FC        75          MOV  A,C       ;THE CHARACTER IS IN C
81EE        D30C        CUT  F0PTP     ;CUTPUT THE DATA
81FO        C9          RET          ;

PUNCH OUTPUT ROUTINE. BITS 4 AND 5 OF THE IOBYT
DETERMINE THE PUNCH DEVICE. A CHARACTER IS EXPECTED
IN THE C REGISTER

81F0        3A0300      LDA  IOBYT      ;GET 10 ASSIGNMENT
81F4        07          PLC          ;MOVE BITS 4 AND 5 TO 6 AND 7
81F5        07          RLC          ;SO THAT THEY CAN BE MOVED
81F6        C3EC81      JMP  L01       ;TO THE CONSOLE POSITION

LIST OUTPUT ROUTINE. BITS 6 AND 7 OF THE IOBYT
DETERMINE THE LIST DEVICE.

81F5        3AC300      LDA  IOBYT      ;GET THE 10 ASSIGNMENT
81FC        L01:       RLC          ;MOVE BITS 6 AND 7 TO THE
81FD        07          RLC          ;CONSOLE POSITION
81FE        C3D981      JMP  OBRCH     ;BRANCH TO PROPER ROUTINE

CONSOLE INPUT STATUS ROUTINE. THE SELECTED
CONSOLE DEVICE IS INTERCATEG TO DETERMINE IF A
CHARACTER IS WAITING. A ZERO IS RETURNED IF NO
CHARACTER AND A -1 IF THERE IS A CHARACTER.

8201        3AC300      LDA  IOBYT      ;WHAT IS CONSOLE DEVICE?
8204        E603        ANI  3         ;BRANCH ON CONSOLE ASSIGN
8206        C21182      JNZ  CSTS2     ;ZERO IS SERIAL. DO IT
8205        DB0B        IN  URTCT     ;GET USART INPUT STATUS
8205        CSTS1:     CSTS:       LDA  IOBYT      ;WHAT IS CONSOLE DEVICE?
820E        E602        ANI  RXRDY     ;ZERO IS NO CHAR.
820E        CE          RZ           ;NON ZERO IS CHAR, USE -1
8210        3EFF        MVI  A,OFFH     ;NON ZERO IS CHAR, USE -1
8210        CS          RET          ;
**PARALLEL INPUT STATUS ROUTINE**

**CHARACTER INPUT ROUTINE (SERIAL). IF THE MODE IS BAUDOT, THE INPUT CHARACTER IS CONVERTED TO ASCII.**

**IF THE CHARACTER INPUT IS A CASE SHIFT CHARACTER, THAT SHIFT IS DONE AND ANOTHER CHARACTER IS WAITED FOR.**

**CH1:**

**CHI:**

**SERIAL CHARACTER OUTPUT ROUTINE. THE MODE IS CHECKED AND IF IT IS BAUDOT, THE CHARACTER IN THE C REGISTER IS CONVERTED TO BAUDOT PFIC TO SENDING. OTHERWISE, IT IS SIMPLY OUTPUT TO THE USART. IF A CASE CHANGE IS INDICATED, THE CASE CHARACTER IS **
; TRANSMITTED PRIOR TO THE ACTUAL CHARACTER (BAUDOT
; MODE ONLY).

; CH0:
824E DEC8 IN URTCT ;Determine mode first
824E E6EC ANI DSR ;0=BAUDOT
824F C5E282 JZ CH02 ;Baudot mode, go convert it

; CH01:
8252 DBC8 IN URTCT ;Test for ready and output
8254 E6C1 ANI TXRY ;The character in the C reg
8256 C5E282 JZ CH01
8255 75 MOV A,C

; CH02:
825A C3CA OUT URTDA
825C C9 RET

; CH03:
825C E5 PUSH H ;Save some registers
825E D5 PUSH D
825F 21EE83 LXI H,EDTAB ;Search the table for a match
8262 1640 MVI C,64 ;The table is 64 chars long
8264 75 MOV A,C

; CH04:
8265 EE CMP M ;Does this one match?
8266 CA7082 JZ CH04 ;Yes, exit loop
8268 23 INX H ;Bump Address
8269 15 DCR C ;Decrement count
826E C26582 JNZ CH03 ;Continue to loop till zero
826F 1640 MVI D,64 ;If no match, pretend blank

; CH05:
8270 3E40 MVI A,64 ;Actual value is 64-count
8272 92 SUB D ;Calculate actual value
8273 57 MOV D,A ;Save a copy in D for later
8274 21EFO0 LXI H,BCASQ ;Test and set case if appropriate
8277 E620 ANI 20H ;6th bit is case (0=LETTERS)
8278 8E CMP M ;Compare to previous case
827A C5E882 JZ CH06 ;match, no need to send case
827C 77 MOV M,A ;No match, set case right

; CH06:
827E 0E1F + MVI C,LTRS ;Send case (0=LETTERS)
8280 1 + TEST ;Case 0 or 20H?
8281 A7 + ANA A ;Set flags; CY=0
8282 CA6882 JZ CH05 ;Letters; case, send it
8284 0E18 MVI C,FIGS ;Not letters; send figures
8286 CD5282 CALL CH01 ;Send the case shift char

; CH07:
8289 4A MOV C,D ;Recover the original char
828A D1 POP D ;Recover some registers
828E E1 POP H
829C C3E282 JMP CH01 ;Go send the character

; ID Check routine. The value of the ID byte is
; returned in the A register
0830 MACRO ASSEMBLER, VER 2.2 MCLN-8080 MONITOR 1.1 ERRORS = 0 PAGE 16

829F
829F 3A0300 LDA IOBYT
829F CS RET

IO SET ROUTINE. THE VALUE OF THE C REGISTER IS
SUBSTITUTED FOR THE VALUE OF THE IOBYT.

8293
8293 75 MOV A,C
8294 32C300 STA IOBYT
8297 CS RET

MEMORY CHECK ROUTINE. TWO RAW LOCATIONS ARE USED
TO STORE THE TOP OF RAM. THEY ARE RETURNED IN
A AND B. LEAST SIGNIFICANT IN A.

829E
829E 3AC6C0 LDA MSZ2 ;MOST SIGNIFICANT BYTE HERE
829F 47 MOV E,A ;PUT IT IN E
829C 3AC5C0 LDA MSZ1 ;LEAST SIGNIFICANT BYTE HERE
829F CS RET

PUNCH BYTE OUTPUT ROUTINE. THE CONSOLE BYTE OUTPUT
ROUTINE IS USED EXCEPT THAT THE IOBYT IS FIRST
ROTATED SUCH THAT THE CONSOLE ASSIGNMENT AND PUNCH
ASSIGNMENT ARE INTERCHANGED. WHEN FINISHED, THEY
ARE AGAIN INTERCHANGED SO THAT ALL IS OK.

82AC
82A0 CDA682 CALL PUCO ;INTERCHANGE PUNCH AND CONSOLE
82A3 CD7263 CALL BYTOUT ;IO BYTE OUTPUT ROUTINE

ROUTINE TO INTERCHANGE CONSOLE AND PUNCH ASSIGN.

82AE
82AE F5 PUSH PSW ;SAVE SOME REGISTERS
82A7 E5 PUSH H
82AE 21C300 LXI H,IOBYT ;POINT HL AT IOBYTE
82AE 7E MOV A,IOBYT ;INTERCHANGE TOP AND BOTTOM
82AC 0F RRC
82AC 0F RRC
82AE 0F RRC
82AE 0F RRC
82AC 77 MOV M,A ;PUT IT BACK
82B1 E1 POP H ;RECOVER SOME REGISTERS
82B2 F1 POP PSW
82B3 CS RET

WORD PUNCH OUTPUT ROUTINE. PUNCH AND CONSOLE
ASSIGNMENTS ARE INTERCHANGED TO USE CONSOLE
**8080 MACRO ASSEMBLER, VER 2.2**

**MC680 MC680 MCMCF 1.1**

**ERRORS = 0 PAGE 17**

---

**ROUTINES.**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>82E4</td>
<td>CCA662</td>
<td>CALL PUCO</td>
</tr>
<tr>
<td>82E7</td>
<td>CCA681</td>
<td>CALL WRDOT</td>
</tr>
<tr>
<td>82EA</td>
<td>C3A662</td>
<td>JMP PUCO</td>
</tr>
</tbody>
</table>

**PUNCH MESSAGE TYPES AGAIN, CONSOLE AND PUNCH ASSIGNMENTS ARE INTERCHANGED TO USE CONSOLE ROUTINES.**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>82ED</td>
<td>CCA662</td>
<td>CALL PUCO</td>
</tr>
<tr>
<td>82C0</td>
<td>CDS6A63</td>
<td>CALL TYPMG</td>
</tr>
<tr>
<td>82C3</td>
<td>C3A662</td>
<td>JMP PUCO</td>
</tr>
</tbody>
</table>

**READER BYTE INPUT ROUTINE. THE READER ASSIGNMENT IT ROTATED INTO THE CONSOLE POSITION SO THAT THE CONSOLE ROUTINES CAN BE USED. THE ASSIGNMENTS ARE RESTORED WHEN DONE. SINCE AN EXIT CAN BE DONE FROM THE EXIT ROUTINE, A FLAG IS SET TO INDICATE THAT THE SWITCH HAS BEEN MADE. IF AN EXIT IS TAKEN THE PROPER ASSIGNMENT WILL BE REGISTERED.**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>82C6</td>
<td>CCE462</td>
<td>CALL RDRCO</td>
</tr>
<tr>
<td>82CC</td>
<td></td>
<td>MOVE READER DEVICE TO CONSOLE</td>
</tr>
<tr>
<td>82C9</td>
<td>CCE663</td>
<td>CALL BYTIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USE CONSOLE BYTE INPUT</td>
</tr>
</tbody>
</table>

**ROUTINE TO RECOVER THE CONSOLE ASSIGNMENT AND OLD READER ASSIGNMENT AFTER USING CONSOLE ROUTINES.**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>82CC</td>
<td>F6</td>
<td>PUSH PSW</td>
</tr>
<tr>
<td>82CD</td>
<td>F6</td>
<td>SAVE SCME REGISTERS</td>
</tr>
<tr>
<td>82CE</td>
<td>21300</td>
<td>LXI H,10BYT</td>
</tr>
<tr>
<td></td>
<td>7E</td>
<td>RFCNT HL AT 10BYTE</td>
</tr>
<tr>
<td>82C2</td>
<td>07</td>
<td>RLC</td>
</tr>
<tr>
<td>82C3</td>
<td>07</td>
<td>RLC</td>
</tr>
<tr>
<td>82C4</td>
<td>77</td>
<td>MOV M,A</td>
</tr>
<tr>
<td>82C5</td>
<td>2C</td>
<td>RFCNT HL AT IFLAG</td>
</tr>
<tr>
<td>82DE</td>
<td>36CO</td>
<td>MVC M,O</td>
</tr>
<tr>
<td>82DA</td>
<td>E1</td>
<td>RESET IT TO ZERO</td>
</tr>
<tr>
<td>82DB</td>
<td>F1</td>
<td>RECOVER SOME REGISTERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RET</td>
</tr>
</tbody>
</table>

**READER WORD INPUT ROUTINE. AGAIN, THE CONSOLE ROUTINE IS UTILIZED BY MOVING THE 10BYTE.**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>82DB</td>
<td>CCE482</td>
<td>CALL RDRCO</td>
</tr>
<tr>
<td>82DE</td>
<td>C02683</td>
<td>CALL WRDIN</td>
</tr>
</tbody>
</table>
`8080 MACRO ASSEMBLER, VER 2.2 MCM-8080 MCRITCH 1.1 ERRORS = 0 PAGE 18`  

```assembly
82E1  C3CC82   JMP   RCVR1
       ; READER TO CONSOLE ASSIGNMENT SWITCHER.

82E4  RDCRC:
82E4  ES
82E6  21C300  LXI   H,10BYT ; POINT HL AT 10BYTE
82E6  7E      MOV   A,M ; SHIFT IOBYTE 2 LEFT
82E8  0F      RRC
82E9  77      MOV   M,A ; RESTORE IT
82EC  2C      INR   L ; POINT AT IFLAG
82ED  3FF     MVI   H,-1 ; SET FLAG TO -1
82EF  E1      POP   H ; RECOVER HL
82F0  C5      RET

; ECHC INPUT AND TEST

82F1  CDB581  CALL   ECHO   ; GET CHAR AND ECHO
82F4  FE20    CPI    ' '; STORE IT
82F6  CA102   JZ     ECHC ; IGNORCE BLANKS
82F9  FE2C    CPI    ','; COMMA IS A DELIMITER
82FE  37      STC
82FC  CE      PZ
82FD  FECD    CPI    CR ; CR IS A DELIMITER
82FF  37      STC
8300  C8      RZ
8301  FE47    CPI    'G' ; STCF IF LARGER THAN P
8303  3F      CMC   ; INVERT CARRY BIT
8304  DC      RNC   ; NO CARRY IS OK CHARACTER
8305  3AC400  LDA    IFLAG
     + TEST
8308  1A7     ANA    A ; SET FLAGS , CY=0
830B  CA7380  JZ     MAIN
830C  C3CC82  CALL   RCVR1
830F  C37380  JMP    MAIN

; NIBBLE ROUTINE, CONVERY ASCII TO HEX

8312  D641    SUI    'A' ; COMPARING FOR >=10
8314  F21983  JP     GTA
8317  C6C7    ADI    7 ; ADJUST FOR GAP BETWEEN 9 & A
8319  C6CA    ADI    10 ; MAKE IT BINARY
831E  CG      RET ; THATS ALL FOR PERFECT INPUT

; EXPRESSION (PARAMETER LIST) GRABBER

831C  EXPR:
```
B21C CC2683 CALL WRDIN ; READ IN 16 BIT GROUP INTO HL
8326 210000 LXI H,0 ; CLEAR BUFFER
8328 C5 PUSH B ; SAVE PARAM COUNT
832A 06C4 MVI B,4 ; GET 4 HEX DIGITS IN ASCII
832C WRD1:
832C CDF182 CALL EIT ; READ/CHECK CHARACTER
8332 D23B83 JNC WRD2 ; CARRY SET IMPLIES DELLIMITER
8333 7B MOV A,B ; LIST FOR NEW WORD
8335 E4 CIJ4 CPI 4 ; B=4 MEANS JUST STARTED
8337 CA2C83 J2 WRD1 ; IGN.CRE. AT BEGINNING
8338 C34863 JMP EXIT ; VALID DELIMITER IF NOT 1ST
833B WRD2:
833E CD1283 CALL NIBBL ; EAT SOME CHARs
8340 25 DAD H ; HL*2
8342 25 DAD H ; HL*2
8344 25 DAD H ; HL*2
8346 25 DAD H ; LONG LEFT SHIFT 4 KIDDIES
8348 B5 GRA L ; BRING IN NEW 4 BITS
834A 6F MOV L,A ; HL NOW HAS GOOD DATA
834C 0E DCF B ; REDUCE CHAR COUNT
834E C22C83 JNZ WRD1 ; 3,2,1,0 AND OUT
8350 EXIT:
8354 7A MOV A,D ; UPDATE CHECKSUM
8356 85 ADD L ; CERDER IMMATERIAL
835A 84 ADD H ; CHK SUM IN A CORRECT
835E 57 MOV D,A ; CHECKSUM UPDATE COMPLETE
8362 C1 POP U ; RESTORE
8364 C5 RET ;

BYTE INPUT RIN

BYTIN:
8366 C5 PUSH B ; SAVE IT
836F 0600 MVI E,0 ; INITIAL BINARY VALUE BUFFER
8371 CDF182 CALL EIT ; READ NEW DIGIT
8374 DAEC83 JC EXITB ; LEAVE IF DELIMITER, A=B=0
8377 CD1263 CALL NIBBL ; GET BINARY VALUE IN A
8379 4A MOV B,A ; SAVE BINARY VALUE IN B
837B CCF182 CALL EIT ; READ 2ND (LOW) DIGIT, IF ANY
837E DAEC83 JC EXITB ; LEAVE IF DELIMITER, A=B=0
8381 CD1263 CALL NIBBL ; CONVERT 2ND DIGIT TO BINARY
8080 MACRO ASSEMBLER, VER 2.2 MCEM-8080 MCNEIR 1.1 ERRORS = 0 PAGE 20

8364 4F MOV C,A ;SAVE CHAR
8365 78 MOV A,B ;GET FIRST CHAR
8366 07 PLC ;MOVE TO TOP CHAR
8367 07 RLC
8368 07 RLC
8369 07 RLC
836A 81 CP A,B ;OR IN FIRST CHAR
836C 47 MOV B,A ;SAVE IN B
836C 78 MOV A,B ;IN CASE A HOLDS DELIM CHAR
836D 82 ADD D ;ADD CHECKSUM
836E 57 MOV D,A ;RETURN UPDATED CKSUM
836F 7E MOV A,B ;RESTORE SA FROM $B
8370 C1 POP B ;RESTORE B
8371 C5 RET ;RET DONE ELSEWHERE

BYTOUT:
8372 C5 PUSH E ;CALLS USE $B
8373 CD7B63 CALL BTOT1 ;DECODE 2 HEX DIGITS
8375 CD481 CALL SPACE ;PRINT A SPACE TRAILER
837A C1 POP B ;RESTORE
837B C9 RET

ETOT1:
837E 47 MOV B,A ;SAVE BYTE IN B
837F 82 ADD D ;ADD TO CHECKSUM.
837C 57 MOV D,A ;SAVE NEW SUM
837E 7B MOV A,B ;SWAP HEX DIGITS
837F 07 RLC
837C 07 RLC
8381 07 RLC
8382 07 RLC
8383 CD7B83 CALL HXOUT ;PRINTED w/HIGH DIGIT
8386 78 MOV A,B ;INSTEAD OF SHIFT. MOV

HEXOUT RTN
8387 HXOUT: ;C-9,A-F NOT CONTIG-
<table>
<thead>
<tr>
<th>Offset</th>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>83E7</td>
<td>E6CF</td>
<td>ANI</td>
<td>0FH: STRIP HIGH DIGIT</td>
</tr>
<tr>
<td>83E9</td>
<td>DECA</td>
<td>SUI</td>
<td>0AH: &gt;9 TEST</td>
</tr>
<tr>
<td>83EB</td>
<td>F29483</td>
<td>JP</td>
<td>HX&lt;1: BRANCH GE 10</td>
</tr>
<tr>
<td>83ED</td>
<td>C63A</td>
<td>ADI</td>
<td>'9'+1: ADD ASCII '9' FOR ASCII CHAR CODE OF #</td>
</tr>
<tr>
<td>83F0</td>
<td>4F</td>
<td>MOV C,A</td>
<td>PRIME ARGUMENT FOR CALL</td>
</tr>
<tr>
<td>83F1</td>
<td>C3D681</td>
<td>JMP CO</td>
<td>OUTPUT IT AND RET</td>
</tr>
<tr>
<td>83F4</td>
<td>C641</td>
<td>ADI</td>
<td>'A': ADD ASCII 'A' FOR ASCII CODE OF DIGIT</td>
</tr>
<tr>
<td>83F6</td>
<td>4F</td>
<td>MOV C,A</td>
<td>PRIME ARGUMENT FOR CALL</td>
</tr>
<tr>
<td>83F7</td>
<td>C3D681</td>
<td>JMP CO</td>
<td>OUTPUT AND RET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NO ERRCH ON INPUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POSSIBLE, SO NO CHKs</td>
</tr>
</tbody>
</table>

; TYPE MESSAGE ROUTINE
; (HL) POINTS TO MESSAGE ADDRESS
; MESSAGE TERMINATED BY NEGATIVE BYTE

; TYPE mg:
<table>
<thead>
<tr>
<th>Offset</th>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>83FA</td>
<td>E5</td>
<td>PUSH</td>
<td>H: PRINT PREFACE MSSG</td>
</tr>
<tr>
<td>83FB</td>
<td>21AD83</td>
<td>LXI H, CRMG</td>
<td>POINT TO PREFACE</td>
</tr>
<tr>
<td>83FC</td>
<td>CDA263</td>
<td>CALL</td>
<td>TYPI: CR.LF... ****</td>
</tr>
<tr>
<td>83FD</td>
<td>E1</td>
<td>POP H</td>
<td>POINT TO 1ST MSSG AGN</td>
</tr>
<tr>
<td></td>
<td>7E</td>
<td>MOV A,M</td>
<td>FALL THRU AND PRINT</td>
</tr>
<tr>
<td>83FA</td>
<td>1</td>
<td>TEST</td>
<td>CHK IT 1ST</td>
</tr>
<tr>
<td>83FB</td>
<td>1</td>
<td>ANA A</td>
<td>SET FLAGS, CY=0</td>
</tr>
<tr>
<td>83FC</td>
<td>F4</td>
<td>RM</td>
<td>NEGATIVE IMPLIES END</td>
</tr>
<tr>
<td>83FD</td>
<td>4F</td>
<td>MOV C,A</td>
<td>PRIME WITH ARG</td>
</tr>
<tr>
<td>8400</td>
<td>CDD681</td>
<td>CALL</td>
<td>PUT IT OUT</td>
</tr>
<tr>
<td>8402</td>
<td>23</td>
<td>INX H</td>
<td>POINT TO NEXT CHAR</td>
</tr>
<tr>
<td>8404</td>
<td>C3A283</td>
<td>JMP H</td>
<td>LOCATE TILL NEG CHAR</td>
</tr>
<tr>
<td></td>
<td>0CCA000</td>
<td>CRMG:</td>
<td>DB 0,0,0,0,-1</td>
</tr>
<tr>
<td></td>
<td>C0C0FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3AFF</td>
<td>CRCO:</td>
<td>DB '':-1</td>
</tr>
<tr>
<td></td>
<td>2EFF</td>
<td>RTMAG1</td>
<td>DB '':-1</td>
</tr>
<tr>
<td></td>
<td>4ES84954</td>
<td>RSMG:</td>
<td>DB 'EXIT ':-1</td>
</tr>
<tr>
<td></td>
<td>2FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0E550A41</td>
<td>EDTAB:</td>
<td>DB 0,'E',LF,'A SIU',CR,'DRJNFCK'</td>
</tr>
<tr>
<td></td>
<td>20E34955</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0D44524A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E6434B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>54A45C7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E55051</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4F4247A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D556580</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C330A2D</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'TZYWHYFGCBG',0A0H,'MXV',80H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>83E2</td>
<td>2C2A3937</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83E6</td>
<td>0D143427</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83EA</td>
<td>2C21A2A8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83EE</td>
<td>3E2B2932</td>
<td>DB '5+12#60157&amp;t+0A0H+*/=80H</td>
<td></td>
</tr>
<tr>
<td>83E2</td>
<td>232E3031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83E6</td>
<td>392F26A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83EA</td>
<td>2E2F3D80</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

NC PROGRAM ERRORS
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A007</td>
<td>0000</td>
<td>BCASI 0007</td>
</tr>
<tr>
<td>BCTAE</td>
<td>E3E6</td>
<td>BEGIN 802B</td>
</tr>
<tr>
<td>BCTEI</td>
<td>E378</td>
<td>BYTN 834E</td>
</tr>
<tr>
<td>BCTIR</td>
<td>E37E</td>
<td>BYTIR 82C6</td>
</tr>
<tr>
<td>CHDC</td>
<td>824E</td>
<td>CHD1 8252</td>
</tr>
<tr>
<td>CHDC4</td>
<td>E27C</td>
<td>CHDS 8266</td>
</tr>
<tr>
<td>CHD2</td>
<td>E106</td>
<td>CRCC 63C4</td>
</tr>
<tr>
<td>CSTS</td>
<td>E201</td>
<td>CSTM1 6225</td>
</tr>
<tr>
<td>DEF</td>
<td>E09F</td>
<td>DISP 802F</td>
</tr>
<tr>
<td>DSFCK</td>
<td>E0CC</td>
<td>DSPO 8084</td>
</tr>
<tr>
<td>ECFC</td>
<td>E1B5</td>
<td>ECFC 81C6</td>
</tr>
<tr>
<td>EXIT</td>
<td>E346</td>
<td>EXITB 833C</td>
</tr>
<tr>
<td>GC</td>
<td>E0CE</td>
<td>GTA 8219</td>
</tr>
<tr>
<td>HXCT1</td>
<td>E387</td>
<td>HXCT 8196</td>
</tr>
<tr>
<td>INST1</td>
<td>E0F0</td>
<td>INST 811B</td>
</tr>
<tr>
<td>ICST</td>
<td>8293</td>
<td>ICST 8801</td>
</tr>
<tr>
<td>LOC</td>
<td>E113</td>
<td>LD1 812E</td>
</tr>
<tr>
<td>LGC</td>
<td>E1FC</td>
<td>LGC 810F</td>
</tr>
<tr>
<td>MAIN</td>
<td>E073</td>
<td>MAIN 829F</td>
</tr>
<tr>
<td>M522</td>
<td>0006</td>
<td>M522 8312</td>
</tr>
<tr>
<td>PARIN</td>
<td>E1A3</td>
<td>PARIN 09A6</td>
</tr>
<tr>
<td>PINFT</td>
<td>0000</td>
<td>PINFT 0002</td>
</tr>
<tr>
<td>PERDY</td>
<td>00EC</td>
<td>PERDY 00C7</td>
</tr>
<tr>
<td>PS4</td>
<td>0026</td>
<td>PS4 82A6</td>
</tr>
<tr>
<td>RETR1</td>
<td>E17A</td>
<td>RETR1 816F</td>
</tr>
<tr>
<td>REMG</td>
<td>E38E</td>
<td>REMG 8145</td>
</tr>
<tr>
<td>SF</td>
<td>0006</td>
<td>SF 8114</td>
</tr>
<tr>
<td>TXRXE</td>
<td>E027</td>
<td>TXRXE 83A2</td>
</tr>
<tr>
<td>UART</td>
<td>00FF</td>
<td>UART 0C55</td>
</tr>
<tr>
<td>UARTC</td>
<td>000A</td>
<td>UARTC 000F</td>
</tr>
<tr>
<td>USFST</td>
<td>0046</td>
<td>USFST 832C</td>
</tr>
<tr>
<td>WRC</td>
<td>E2DB</td>
<td>WRC 82A4</td>
</tr>
</tbody>
</table>

* 01

* 02

* 03

* 04

* 05

* 06

* 07

* 08
8. Circuit Board Layout and Schematic Diagrams

The following pages contain complete diagrams of the HAL Communications Corp. Model MCEM-8080 Microcomputer System. These diagrams reflect the current circuit connections as of the printing date of this manual. HAL Communications reserves the right to make changes in the circuitry without incurring any obligation to make such changes in previously sold units. The diagrams may not be duplicated in any form without the express permission of HAL Communications Corp.
Use of the UPB cable connections:

The MCEM-8080 is furnished with a two foot length of 40 conductor ribbon cable with a connector on one end that mates to the UPB (Universal Processor Bus) connector of the MCEM. The cable connections are explained in Table 2.6 on page 2-12 of the MCEM manual (the red stripe on the cable corresponds to pin 1). When plugging the cable into the MCEM, be sure to align it correctly as indicated by the small arrows embossed on the plastic connectors. If your cable connector has NOT been polarized, it should be by putting a small piece of bare wire into the cable connector pin 32 location. This should correspond to the missing pin of the MCEM UPB connector.

Connection to the cable can be made by simply separating the conductors of the ribbon cable, stripping and tinning each one required, and then connecting the wires as required. The cable SHOULD NOT BE EXTENDED BEYOND THE TWO FOOT LENGTH FURNISHED. Alternately, the same 3M connector can be attached to the ribbon cable. The cable connector is a 3M part no. 3417-0000, which can be obtained from a 3M distributor or from HAL Communications Corp. for $6.00. The mating circuit board connector is a 3M no. 3432-1002 connector at $4.00 each from HAL. The cable connector is designed so that it clamps directly over the cable and several can therefore be attached to the same cable. The following procedure should be used to attach the 3M connector to the cable.

Installation of additional connectors to the UPB cable:

The 40 conductor cable polarity is distinguished in two ways:

1. The RED stripe side corresponds to pin 1 of the UPB connector
2. The ribbing of the cable is heavier on one side than the other. This can best be determined by looking at the END of the cable, although dragging of your finger-nail across the two sides will also indicate which is roughest and therefore has the heavier ribbing.

Refer to the attached Figure 2 for the following instructions.

The 40 pin connector (3M No. 3417-0000) is a two-piece assembly, the larger section with the connector pins and a smaller clamp strip. The clamp strip has a protective paper covering over an adhesive. In assembly, the protective strip is removed to expose the adhesive and the cable is "sandwiched" between the forks of the connector pins and the clamp strip. Attach the connector to the cable using the following procedure:
1. Locate and mark the desired connector location on the 40 conductor cable. Note that several connectors can be placed on the same cable since the connectors simply clamp around the cable, allowing it to pass through. Therefore, if several connectors are intended, do not cut the cable until the end-most connector has been installed.

2. Carefully remove the protective paper BUT NOT the adhesive from the clamp piece.

3. Put the adhesive surface of the clamp piece on the the heavily ribbed side of the ribbon cable, taking care to remove only the paper and not the adhesive with it.

4. Locate the embossed arrow on the connector pin section. This indicates the location of pin 1 of the connector.

5. Place the pin section of the connector on the opposite side of the cable from the clamp assembly, aligning the arrow with the red stripe on the cable.

6. Align the guide pins of the clamp piece into the mating holes of the pin section and press the two pieces together with your fingers until the forks of the connector pins start to "bite" into the cable.

7. RECHECK THE CABLE AND CONNECTOR ALIGNMENT TO BE SURE THAT:
   a. The red stripe of the cable is adjacent to the arrow
   b. The heavily ribbed side of the cable is against the clamp.
   c. The connector is perpendicular to the cable.

8. After checking, the two sections can be completely pressed together in a bench vice. Use only enough pressure to close the gaps between the cable - too much pressure will break the connector. If the vise has rough surfaced jaws, you may wish to prevent scratching of the connector by using cardboard protective shims.

9. If additional connectors are required, they can be attached at any cable location using the above procedure. If it is desired to end the cable after the connector, use a VERY SHARP knife or razor-blade and cut the cable off flush with the outside edge of the connector. Be careful to cut on the "scrap" end of the connector and not on the processor end! After cutting, inspect the cut edge to be sure that adjacent wires have not been shorted together in the process of cutting.

10. The completed connector should now be keyed by inserting a short piece of No. 22 bare wire into connector pin no. 32 position. Notice that the numbers are marked on the face of the connector - odd numbers down the arrow side (starting at the arrow) and even numbers on the other side.
To processor

"Scrap-end" of cable

 Cut along this line with razor blade

d. Cutting-off excess cable

Figure 2. Preparation of UPB Cable Connector