MEMON/80 USERS MANUAL

Version 3.05

12 March 2022

Martin Eberhard

TABLE OF CONTENTS

Memon/80

Simple, Extensible ROM Monitor for an 8080 Microcomputer

INTRODUCTION

Memon/80 is a ROM-based monitor for an 8080-based computer, which has a serial port as its console and optionally a second serial port for transferring data. It provides basic commands for examining, modifying, saving, loading, searching, and testing memory. It can also be used to program EPROMs, using a memory-based EPROM programmer such as any of the Cromemco Bytesavers.

Memon/80 supports booting from a variety of floppy disk controllers, when assembled with the 2K-byte option.

Memon/80 provides standardized software interfaces for the serial ports, suitable for a CP/M BIOS, and compatible with programs that call CP/M's BIOS for I/O.

An assembly option allows construction of either a tight, 1K-byte monitor, or a larger 2K-byte monitor. The 2K-byte version includes a few additional commands, such as memory test and search.

RAM REQUIREMENTS

Memon/80 requires the upper 128 bytes of RAM in the highest 256-byte page of contiguous RAM, or optionally the upper 128 bytes of a specified RAM page. Unless the specified RAM page option is selected, Memon/80 will search and find the highest RAM page during initialization. The Beginning address of Memon/80's RAM usage will be printed immediately following the sign-on banner.

CONSOLE AND TRANSFER I/O PORTS

Memon/80 can be assembled to work with a variety of serial port boards. One serial port is used as Memon/80's console, and the other is the "Transfer Port." (If your computer has only one port, then Memon/80 will work with only a console port.

The two Intel Hex transfer functions, HD and HL, transfer hex data through the Transfer Port or the Console port.

TE (Terminal mode) sends data between the Console and the Transfer Port, if it exists.

The following serial I/O boards are supported:

- Altair 88-SIO
- Altair 88-2SIO or my own 88-2SIOJP
- Altair 8800b Turnkey Module's serial port
- Altair (MITS) 88-UIO's serial port
- California Computer Systems 2410 CPU'S serial port
- California Computer Systems 2718
- California Computer Systems 2719
- Compupro Interfacer 1

- Compupro Interfacer II
- Cromemco 4FDC/16FDC/64FDC disk controller's serial port
- Cromemco TU-ART
- Electronic Control Technology (ECT) R^2I/O
- Heathkit H8-4 serial/cassette interface
- Heathkit H8-5 quad serial port
- IMSAI SIO-2
- IMS International 440’s serial ports
- IMS International 480 quad serial port
- Ithaca Intersystems Series II VIO's serial ports
- Jade Serial/Parallel I/O Board's serial ports
- Micromation Doubler disk controller's serial port
- Morrow Designs Multi-I/O
- Morrow Designs Wunderbus I/O
- Processor Technology 3P+S's serial port
- Salota I/O 2+1's serial ports
- Seattle Computer Products SCP400
- Solid State Music IO4's ser**IAL PORTS (CONFIGURED AS ONE OF** the other supported boards)
- Solid State Music IO5's serial ports
- Tarbell 4044 4S2P's serial ports
- Vector Graphic BitStreamer
- Vector Graphic BitStreamer II
- Wameco (WMC) IO-1B's serial port

SUPPORTED DISK CONTROLLERS

Memon/80 can be assembled to boot from a variety of floppy disk controllers. (As of rev 2.1, not all of these disk controllers have been tested...)

- Altair 88-DCDD 8-inch disk subsystem
- Altair 88-MDS minidisk subsystem
- California Computer Systems 2422 FDC
- Cromemco 4FDC/16FDC/64FDC
- Heathkit H17 Minidisk Controller
- IMSAI FIF (FIB & IFM)
- IMSAI MDC-DIO (with either 8” disks or minidisks)
- Micromation Doubler
- Micropolis - FD Control B
- Northstar MDC-A FDC
- Northstar MDS-A FDC
- Northstar MDS-D Single- or double-density minidisk
- Salota FDC (which is the same as a Versafloppy II) [1]

- SD Systems Versafloppy single-density FDC[1]
- SD Systems Versafloppy II double-density FDC[1]
- Tarbell 1011 8" Single-density FDC
- Tarbell 2022 8" Double-density FDC

MEMON/80 COMMANDS

Memon/80 commands may be typed at the Memon/80 prompt, '>'. Commands are executed once you type the Return key. You can correct typing mistakes with the DEL or the BACKSPACE key.

All parameters are 4 hex digits, unless otherwise noted. Additional upper hex digits are ignored. Leading 0's need not be typed.

Commands marked with {ROM2K} require the ROM2K option to be true, and require a 2K-byte ROM for Memon/80.

? (PRINT HELP MESSAGE)

Prints a minimal list of Memon/80 commands on the console. (Requires HELPC option to be TRUE.)

MEMORY COMMANDS

CO <SOURCE> <DEST> <COUNT> [<REPEAT>] (COPY MEMORY)

Copies <COUNT> bytes memory starting at address <SOURCE> to memory starting at address <DEST>. Optionally, repeats the copy <REPEAT> times. (Max value for <REPEAT> is FFh.)

Verifies the copy when done, using the VE command.

The <REPEAT> option is only available if the CORPT option is TRUE.

Each period printed on the Console represents a completed pass through the copy.

Press CONTROL-C to abort a Copy.

This command can be used to program an EPROM with (for example) a Cromemco Bytesaver board. See example below.

DU [<ADDRESS> [<COUNT>]] (DUMP MEMORY)

Dumps <COUNT> bytes memory on the Console in hexadecimal, starting at <ADDRESS>, which defaults to 0. If no <COUNT> is specified, then dump all 64K bytes of memory.

Press the space bar to pause the dump, and press CONTROL-C to abort the memory Dump.

EN [<ADDRESS>] (ENTER MEMORY DATA)

Allows you to enter 2-digit hex data into memory starting at <ADDRESS>, using a space or Return as a separator between bytes. Type Return on a blank line to exit. If no address is provided, then the starting address will be 0.

---

[1] The Versafloppy and Versafloppy II normally work with a PROM board that contains the entire CP/M BIOS and boot code. Memon/80 loads track 0, sector 1 into RAM at 80h and then executes the loaded code. This implies that you have written a loader and a BIOS for the Versafloppy or Versafloppy II FDC.

CONTROL-C aborts without saving the current line of data.

FI [<ADDRESS> [<COUNT> [<VALUE>]]] (FILL MEMORY)

Fills <COUNT> BYTES OF memory, starting at <ADDRESS>, with <VALUE>,which is a 2-digit hex value. <VALUE> and <ADDRESS> default to 00. If <COUNT> is not specified, then the fill will stop when it reaches Memon/80's RAM page. Thus, "FI" will clear all RAM.

MT <ADDRESS> <COUNT> {ROM2K} (TEST MEMORY)

Test <COUNT> bytes of memory, starting at <ADDRESS>. This is a destructive test - RAM will be filled with garbage. Errors are reported to the console. This will skip over the portion of RAM that Memon/80 uses for its stack.

SE <ADDRESS> <BYTE1> <BYTE2>.. <BYTEN> {ROM2K} (SEARCH MEMORY)

SE <ADDRESS> 'TEXT STRING' {ROM2K} (SEARCH MEMORY)

Search for string in memory, starting at <ADDRESS>. You can also mix hex bytes and text strings, e.g.:

SE 100 'Hello World',0A,OD,'Second Line'

VE <ADDRESS1> <ADDRESS2> <COUNT> (VERIFY MEMORY)

Compares a block of memory starting at <ADDRESS1> that is <COUNT> bytes long, to an equal-sized block of memory starting at <ADDRESS2>. Differences are reported on the Console, with the address and data from the first data block, followed by the data found in the second block.

Press CONTROL-C to abort a verify operation. Any other key will pause until you press another key.

8080 I/O PORT COMMANDS

IN <PORT> (INPUT FROM PORT)

Inputs from <PORT> and prints the result on the console.

OT <PORT> <DATA> (OUTPUT TO PORT)

Outputs <DATA> to <PORT>.

EXECUTION COMMANDS

BO [<DRIVE>] (BOOT FROM FLOPPY DISK)

Boots from Drive 0 (or A). Only if a Cromemco disk controller is specified, you can optionally boot from one of the other drives, by specifying the Drive, between 0 and 3.

Depending on the particular disk controller, various error messages will be printed, if booting fails.

Many disk controllers have long enough boot code that they require ROM2K to be TRUE to fit.

CE <COMMAND LINE TEXT> {ROM2K} (EXECUTE CP/M PROGRAM)

Use this command after you have loaded a program (such as FLEXER or FORMAT) into RAM using the HL command. This will install a jump to Memon/80's jump table in RAM at address 0000, copy any Command Line

Text into CP/M's command line buffer (where a CP/M program would expect it to be), and then jumps to address 0100h, the normal CP/M program execution address. This command will work only for CP/M programs that perform basic I/O via calls to the BIOS (not the BDOS). The following BIOS entry points are supported by the CE command:

- BIOS + 00h Cold Boot (Restarts Memon/80)
- BIOS + 03h Warm Boot (Also restarts Memon/80)
- BIOS + 06h Console Status (returns status in a)
- BIOS + 09h Console Input (Returns keyboard character in a)
- BIOS + 0Ch Console Output (Prints character in c on console)
- BIOS + 12h Punch Output (sends byte in c to the Transfer Port)
- BIOS + 15h Reader Input (Returns Transfer Port byte in a)

EX [<ADDRESS>][<OPTION>] (EXECUTE)

Calls <ADDRESS>, which defaults to 0.

Programs can return to Memon/80 with a RET instruction, or by jumping to MEINIT or MEWARM. (Obviously, this won't work if <Option> is used to disable the Memon/80 ROM.)

<Option> is only available if EXOPT is TRUE. If <Option> is set to 1 then Memon/80 will execute an 'IN FFh' prior to executing at the specified address. (Some computers, such as later Altairs, will disable ROM and enable RAM in its place when an IN FF is executed. This option, which executes from RAM, will allow you to disable the Memon/80 ROM when you execute external code.)

TRANSFER PORT COMMANDS

HD <ADDRESS> <COUNT> (DUMP MEMORY AS INTEL HEX FILE)

Dumps <COUNT> bytes of memory to the Transfer Port, starting at <ADDRESS>, in Intel Hex format.

HL [<OFFSET>] (LOAD INTEL HEX FILE INTO MEMORY)

Loads an Intel Hex file from the Transfer Port into memory at the addresses specified in the hex file. If optional offset <OFFSET> is specified, then this value is added to the record addresses.

A period is printed on the Console for each record received.

Any hex record that would overwrite Memon/80's RAM page will cause an address error. Memon/80 will report hex errors and checksum errors as well. Any error will abort the load.

Loading terminates with any record with 0 data bytes. You can also abort the load by typing CONTROL-C.

If ROM2K is true, then Memon/80 will perform a read-back test when writing to RAM, to catch any RAM errors.

If HLRECS is true, then Memon/80 will print a count of the total number of records received when the end-of-file record is received.

TB <BAUD> {ROM2K} (SET TRANSFER PORT BAUD RATE)

Set Transfer Port baud rate, <BAUD> from this table:

Value	Baud Rate	Value	Baud Rate
0	110 (2 stop bits)	6	4800
1	150	7	9600
2	300	8	19200
3	600	9	38400
4	1200	A	57600
5	2400	B	76800

This command is only available if the selected serial port board allows software to set the baud rate. Not all baud rates are available for every serial portboard. Attempting to select an unsupported baud rate will result in a command error.

TE [<EXITCHR>] (TERMINAL MODE)

Enters Terminal Mode. Console keyboard data is sent to the Transfer Port, and Transfer Port data is sent to the Console. (This command is useful for verifying the Transfer Port connection.)

<EXITCHR> specifies the Exit Character, which is a control character that defaults to CONTROL-C. Control characters may be entered without the CONTROL. For example, you may type Z instead of CONTROL-Z. (Note: if you type CONTROL-C as as <EXITCHR>, the TE command will immediately abort.)

Type the Exit Character to exit Terminal Mode.

TP [<0/1>] (SET TRANSFER PORT)

The Transfer Port is the port used for transferring Intel hex files with the HD and HL commands. TP 1 (the default) transfers via the

CONFIGURING MEMON/80

Memon/80 is configured via a series of TRUE/FALSE equates at the beginning of the source code. Set these appropriately for your system, and then assemble Memon/80 using Digital Research's ASM or an equivalent 8080 assembler.

Here are the choices you must make during configuration. I have provided space where you can write down your configuration choices for future reference.

1. SPECIFY THE CPU SPEED: CPUMHZ EQU <1-6>

Your Configuration: CPUMHZ equ _____

This parameter specifies the CPU speed in MHz, and is used only for timeout loops when booting from floppy disk.

2. SPECIFY THE ROM SIZE: ROM2K EQU TRUE/FALSE

Your Configuration: ROM2K equ ☐TRUE ☐FALSE

- **ROM2K equ FALSE**: the assembler will generate an error message if the resulting code is larger than 1K-byte, the size of one 2708 EPROM.

- **ROM2K equ TRUE**: the assembler will generate an error message if the resulting code is larger than 2K-bytes, the size of one 2716 EPROM.

3. Specify Memon/80's Memory Utilization

These parameters allow you to customize Memon/80, to control what memory addresses it uses, and to tradeoff features for EPROM space.

Memon/80's EPROM address: MEBASE equ <Address>

Your Configuration: MEBASE equ _____

This parameter specifies the assembly address for Memon/80. The low byte of this parameter must be 00 (e.g. 0F800h)

RAM Execution: RAMCOD equ TRUE/FALSE

Your Configuration: RAMCOD equ ☐TRUE ☐FALSE

- **RAMCOD equ FALSE**: This is the normal setting. Memon/80 is assembled to execute from EPROM.
- **RAMCOD equ TRUE**: (Primarily for debugging) This allows you to create Memon/80 suitable for executing from RAM, adding a few extra tests to prevent FI and MT commands from overwriting Memon/80 itself. If you also set MEBASE to 0100h, then Memon/80 can be run directly from CP/M.

RAM Hunt: RAMHNT equ TRUE/FALSE

Your Configuration: RAMHNT equ ☐TRUE ☐FALSE

- **RAMHNT equ FALSE**: Uses the RAM specified by the MEMRAM parameter for Memon/80's RAM needs (stack, buffer, variables). This option uses less code space than the TRUE option, and is mandatory if either of your I/O boards requires interrupts.
- **RAMHNT equ TRUE**: Causes Memon/80 to hunt for the highest RAM page during initialization, and use that page for its RAM needs. This uses more code space than does the FALSE option. Also, Memon/80 will not find RAM that is at a higher address than any EPROM in the system.

Memon/80's RAM Page: RAMEND equ <Address>

Your Configuration: RAMEND equ _____

This parameter specifies the last address for Memon/80's RAM, if RAMHNT equ FALSE. (The value doesn't matter if RAMHNT equ TRUE) All of MEMON/80's RAM must fit within the same 256-byte RAM page, or an error message will be printed when you assemble Memon/80. If Eemon/80 will be booting from an Altair disk controller, then RAMEND must be of the form xxFFh.

Include Help Command: HELPC equ TRUE/FALSE

Your Configuration: HELPC equ ☐TRUE ☐FALSE

- **HELPC equ FALSE**: Saves code space by eliminating the '?' help command
- **HELPC equ TRUE**: Includes a minimal help command, '?'

Allow Lowercase Input: LOWERC equ TRUE/FALSE

Your Configuration: LOWERC equ ☐TRUE ☐FALSE

- **LOWERC equ FALSE**: Saves 11 bytes, but requires all input to be uppercase.
- **LOWERC equ TRUE**: allows input to be uppercase or lowercase

Enable Record Count for HL Command: HLRECS equ TRUE/FALSE

Your Configuration: HLRECS equ ☐TRUE ☐FALSE

- **HLRECS equ FALSE**: Saves 25 bytes, and eliminates reporting the record count for the HL (Intel hex load) command.
- **HLRECS equ TRUE**: enables the record count for the HL command

Enable Repeat Option for CO Command: CORPT equ TRUE/FALSE

Your Configuration: CORPT equ ☐TRUE ☐FALSE

- **CORPT equ FALSE**: Saves 36 bytes, and eliminates the <RPT> option for the CO (copy) command.
- **CORPT equ TRUE**: enables the <RPT> option for the CO command

Enable ROM-Disable Option for EX Command: EXOPT equ TRUE/FALSE

Your Configuration: EXOPT equ ☐TRUE ☐FALSE

- **EXOPT equ FALSE**: Saves 17 bytes, and eliminates the <OPT> option for the EX (execute) command.
- **EXOPT equ TRUE**: enables the <OPT> option for the EX command

Prettier Output for the DU Command: DUPRTY equ TRUE/FALSE

Your Configuration: DUPRTY equ ☐TRUE ☐FALSE

- **DUPRTY equ FALSE**: Saves 17 bytes, but the ASCII output does not line up for lines with other than 16 bytes of data
- **DUPRTY equ TRUE**: pads out the ASCII output to line up nicely

4. Specify the Console Port: <PORTNAME> equ TRUE/FALSE

Your Configuration: _____ equ TRUE

Memon/80's source code includes a long list of supported serial ports that can be used as its console. Select exactly one of these by setting its equate to TRUE, and all other console port equates to FALSE. Note that the amount of code to support these serial ports varies, resulting in a larger or smaller code set, depending on your serial port choice.

If the selected serial port requires interrupts (e.g. the Heathkit H8-5), then you must specify RAMHNT equ FALSE, and specify a lat RAM address for Memon/80 with the RAMEND setting.

5. Specify the Transfer Port: <PORTNAME> equ TRUE/FALSE

Your Configuration: _____ equ TRUE

Memon/80's source code includes a long list of supported serial ports that can be used as its Transfer Port. Select at most one of these by

setting its equate to TRUE, and all other console port equates to FALSE. Note that the amount of code to support these serial ports varies, resulting in a larger or smaller code set, depending on your serial port choice.

If you do not select ant port to be the transfer port, then the following commands will be deleted: TB, TE, TP.

Some of the supported serial ports have software-selectable baud rates. If you select such a serial port as the Transfer Port and ROM2K is TRUE, then the TB command (which lets you set the Transfer Port baud rate) will be added to Memon/80 automatically.

If the selected serial port requires interrupts (e.g. the Heathkit H8-5), then you must specify RAMHNT equ FALSE, and specify a lat RAM address for Memon/80 with the RAMEND setting.

6. Set The Serial Port Base Addresses

Memon's source code includes a list of the standard addresses for the supported serial ports. However, you may decide to change the address of your console and/or transfer port (with jumpers or switches on the board) to avoid conflicts with other hardware in your system.

Note that while most serial boards use I/O mapping for their port addresses, some use memory mapping instead. Port addresses are 8-bit (2-digit) values, while memory addresses are 16-bit (4-digit) values.

Console Port I/O Address: CBASE equ <Address>

Your Configuration: CBASE equ _____

Set CBASE to the base address of the serial port that you have assigned to be the console.

Transfer Port I/O Address: TBASE equ <Address>

Your Configuration: TBASE equ _____

Set TBASE to the base address of the serial port that you have assigned to be the transfer port. Make sure that TBASE doe not equal CBASE, or overlap any of the address space occupied by the console port.

7. Set The Serial Port Baud Rates

If the serial port board that you have selected for the console or the transfer port allow software to set their baud rates, then choose the baud rates from the table in the TB command above.

Console Port Baud Rate: CBAUD equ <Value>

Your Configuration: CBAUD equ _____

Set the console serial port baud rate by setting CBAUD from the baud rate table. For example, to set the baud rate to 9600, CBAUD equ 7.

Transfer Port Default Baud Rate: TBAUD equ <Value>

Your Configuration: TBASE equ _____

Set the default Transfer Port baud rate by setting TBAUD from the baud rate table. For example, to set the Transfer Port default baud

rate to 57600, TBAUD equ 0Ah (or TBAUD equ 10). Note that you can later change the Transfer Port baud rate using the TB command.

8. Specify The Disk Controller: <Controller> equ TRUE/FALSE

Your Configuration: _____ equ TRUE

This selects the floppy disk controller used by the BO (boot from floppy disk) command. If you do not want the BO command, then set all floppy disk controller equates to FALSE.

Memon/80's source code includes a long list of supported floppy disk controllers. Select exactly one of these by setting its equate to TRUE, and all other floppy disk controller equates to FALSE.

Note that the amount of code to support these floppy disk controllers varies (a lot!), resulting in a larger or smaller code set, depending on your floppy disk controller choice. For some disk controllers, you have a choice of using the controller's onboard ROM to boot, or to use Memon/80 code to boot. Using the controller's ROM to boot saves a lot of code space. Using Memon/80's code will give you meaningful error messages during boot, and will return to Memon/80 if the boot fails.

9. Set the Disk Controller Base Addresses

Some floppy disk controllers use I/O addressing, some use memory-mapped addressing, and some use both. Memon/80's source code includes a list of the default I/O and/or memory addresses for the supported disk controllers. However, you may decide to change the address of your disk controller (with jumpers or switches on the board) to avoid conflicts with other hardware in your system. (Note that some disk controllers don't allow you to change their addresses.)

Disk Controller I/O BASE Address: DIBASE equ <address>

Disk Controller Memory BASE Address: DMBASE equ <address>

Your Configuration: DIBASE equ _____

DMBASE equ _____

Set DIBASE and DMBASE for your disk controller, making sure these addresses do not conflict with other hardware in your system.

10. Specify whether or not interrupts will be enabled

If your system requires interrupts to be enabled, set ENINTS equ TRUE, and Memon/80 will run with interrupts enabled, only masking them when timing is critical. Otherwise, set ENITS equ FALSE to mask interrupts always, and save some code space.

Some I/O boards (e.g. the Heathkit H8-5) require interrupts to be enabled. If you have specified such an I/O board for either the console or the transfer port, then Memon/80 will enable interrupts anyway, regardless of how ENINTS is set.

Your Configuration: ENINTS set ☐TRUE ☐FALSE

- **ENINTS set FALSE**: saves code space by masking interrupts
- **ENINTS set TRUE**: enables interrupts, masking only when necessary

EXAMPLE: USING MEMON/80 TO BUILD A CP/M DISK

You can use Memon/80 to build a bootable CP/M disk from files that you load over the serial port. Here is how you might do that:

Checklist:

A CP/M disk format utility (such as my own FORMAT.COM for the CCS 2422 disk controller) that performs its console I/O via BIOS calls, and makes no BDOS calls. (Such a program can run almost stand-alone, using Memon/80 for its console I/O.)

A custom PUTSYS program that can write to your disk controller. (See *CP/M Operating Systems Manual*, Appendix C, for a description of PUTSYS.) In this example, PUTSYS expects to find the complete CP/M image (including the boot loader and the BIOS) in RAM starting at address 3380h. (Note that the example PUTSYS program in the CP/M manual executes at address 200h.)

CPM.HEX: A hex file of an unconfigured version of CP/M, or even a version of CP/M that's configured for some other computer. If you have a binary version of CP/M (e.g. CPM,COM), then you can use a PC-based BIN2HEX converter to create CPM.HEX.

MYBOOT.HEX: A custom 128-byte boot loader for your disk controller

MYBIOS.HEX: A custom BIOS for your disk controller and I/O devices

The last three must be assembled for the same CP/M memory location. This example assumes you are building a "standard" a 20K CP/M system. (See *CP/M Operating Systems Manual*, Section 6.2.)

1. Format a blank disk:
   1. Use the HL function to load FORMAT.HEX into RAM (at 100h)
   2. Initiate FORMAT using the CE command, with any command line options it might need, for example:

      %CE A: {This will tell my own FORMAT.COM to format drive A}
2. Assemble the CP/M components in RAM:
   1. Use the HL function to load CPM.HEX into RAM at 3400h.
   2. Use the HL program to load MYBOOT.HEX into RAM at 3380h.
   3. Use the HL command to load your MYBIOS.HEX into RAM at 4A00h.
3. Write CP/M onto the formatted disk:
   1. Load your PUTSYS.HEX program into RAM at 200h.
   2. Execute PUTSYS to install CP/M on disk:

      %EX 200
4. Now you can load CP/M's non-built-in utilities onto disk, using HL. I recommend first loading file transfer program (such as my own XMODEM.com and its configuration file, XMODEM.CFG), so that you can use that program to transfer the rest of CP/M's files using just CP/M.
   1. Boot the CP/M disk, to load CP/M into memory

2. Reset your computer to restart Memon/80 (either with the front panel or with a jump-start board)
3. Load the next file into RAM at 100h, using HL
4. Restart CP/M

   %EX 0

5. Use CP/M's SAVE to save the file to disk like this:

   >SAVE XMODEM.COM 10

6. Repeat for as many files as you want to transfer

EXAMPLE: PROGRAMMING AN EPROM

As an example, suppose:

1. We have a Cromemco 8K Bytesaver board, which occupies addresses E000h through FFFFh[2]
2. We have assembled code whose target address is E400h (which is Socket 1 in this 8K Bytesaver)
3. We actually use the Socket 7 (starting at FC00h) for programming EPROMS.[3]
4. We will use 400h bytes of RAM, starting at 1000h, as a buffer
5. We plan to load the hex file via the Transfer Port

Step 1: Select the Transfer Port

>TP

(You can verify that the Transfer Port is working, by using the TE command.)

Step 2: Load the Intel Hex file into the RAM buffer:

The Intel Hex file that was generated by our assembler has address fields starting at E400. The address offset to our buffer is:

1000h - E400h = -D400h

To create a negative number, compliment, and add one:

-D400h = 2BFFh+1 = 2C00h

Load the Intel Hex file with this offset:

>HL 2C00

{Send the Intel Hex file to the Transfer port}

The file should now be in RAM, starting at 1000h. You can see it using the Memory Dump command:

>DU 1000 400

Step 3: Program the EPROM

The 8K Bytesaver uses 2708 EPROMs, which have 400h bytes of data, and require 60 (3Ch) programming passes on a Cromemco 8K Bytesaver.

Note that Cromemco recommends removing the Programming Diodes on the 8K Bytesaver, for any EPROM sockets that contain code that you don't

[2] An 8K Bytesaver has eight sockets, each of which can read or program a 2708 EPROM.

[3] We might do this because we have a ZIF socket installed in the 8K Bytesaver's Socket 7.

want to overwrite accidentally. Make sure that the socket that you plan to use for programming has its Programming Diode installed. (These diodes are just above the sockets, near pin 24.)

To program and verify our EPROM:

1. Insert a blank EPROM in 8K Bytesaver Socket 7
2. Turn on the red programming switch on the 8K Bytesaver
3. Issue a Copy command:

   >CO 1000 FC00 400 3C

Programming will take about 35 seconds. When done, the EPROM will be verified, and any mismatches will be reported on the Console.

4. Turn off the red programming switch on the 8K Bytesaver.

Step 4: Move the EPROM to its target socket

Remove the EPROM from Socket 7 and insert it in Socket 1.

Alternatively, we could have just put the EPROM in the 8K Bytesaver's Socket 1 in the first place (assuming that Socket 1 has its Programming Diode installed), and programmed it there:

   >CO 1000 E400 400 32

SOFTWARE ENTRY POINTS

Memon/80 provides the following set of fixed-address entry points for software access:

MEINIT (xx00) (RESTART MEMON/80)

This assumes that Memon/80 is still resident in memory.

MEWARM (xx03) (RESTART MEMON/80)

(The same as MEINIT)

MECSTA (xx06) (GET CONSOLE INPUT STATUS)

Call this address to get the Console keyboard status. This is subroutine compatible with CP/M BIOSes.
On Return:
   A=0 and Z flag set if no Console keyboard character waiting
   A=FFh and Z flag cleared if a Console keyboard character is waiting.
   All other registers are preserved

MECIN (xx09) (GET CONSOLE INPUT)

Call this address to get one Console keyboard character. This is subroutine compatible with CP/M BIOSes.
This subroutine waits for a Console keyboard character, and returns it in A. The Z flag is always cleared. All other registers are preserved.
NOTE: Parity is not cleared.

MECOUT (xx0C) (CONSOLE OUTPUT)

Call this address to send one character to the Console. This is subroutine compatible with CP/M BIOSes.
On Entry:

C=character to print on the Console
On Return:
A=C
All other registers are preserved.

MECTSO (XX0F) (GET CONSOLE OUTPUT STATUS)

Call this address to get the Console's output status.
On Return:
A=0 and Z flag set if not ready (meaning that the transmit queue is full)
A=FFh and Z flag cleared if a ready (meaning that the transmit queue is not full)
All other registers are preserved.
NOTE: Parity is not cleared.

METPDO (XX12) (TRANSFER PORT OUTPUT)

Call this address to send one character to the Transfer Port.
On Entry:
C=character to send to the Transfer Port
On Return:
A trashed
All other registers are preserved.

METPDI (XX15) (GET TRANSFER PORT DATA INPUT)

Call this address to get one Transfer Port character.
Waits for a Transfer Port character, and returns it in A. The Z flag is always cleared. All other registers are preserved.

METPSI (XX18) (GET TRANSFER PORT INPUT STATUS)

Call this address to get the Transfer Port's input status.
On Return:
A=0 and Z flag set if not ready (meaning that the receive queue is empty)
A=FFh and Z flag cleared if a ready (the receive queue is not empty)
All other registers are preserved.

METPSO (XX1B) (GET TRANSFER PORT OUTPUT STATUS)

Call this address to get the Transfer Port's output status.
On Return:
A=0 and Z flag set if not ready (meaning that the transmit queue is full)
A=FFh and Z flag cleared if a ready (meaning that the transmit queue is not full)
All other registers are preserved.

MECFG (XX1E) (GET MEMON CONFIGURATION)

Call this address for various MEMON/80 configuration details.
Currently returns a=CPU speed in units of 125 KHz.