

# MEMON/80 USERS MANUAL

Version 2.4

26 December 2018

Martin Eberhard



## TABLE OF CONTENTS

Introduction.....	1
RAM Requirements.....	1
Console and Transfer I/O Ports.....	1
Supported Disk Controllers.....	2
Memon/80 Commands.....	3
? (Print Help Message) .....	3
Memory Commands.....	3
CO <SOURCE> <DEST> <COUNT> [<REPEAT>] (Copy Memory) .....	3
DU [<ADDRESS> [<COUNT>]] (Dump Memory) .....	3
EN [<ADDRESS>] (Enter Memory Data) .....	3
FI [<ADDRESS> [<COUNT> [<VALUE>]]] (Fill Memory) .....	3
MT <ADDRESS> <COUNT> {ROM2K} (Test Memory) .....	3
SE <ADDRESS> <BYTE1> <BYTE2>.. <BYTEN> {ROM2K} (Search Memory) .	4
SE <ADDRESS> 'Text string' {ROM2K} (Search Memory) .....	4
VE <ADDRESS1> <ADDRESS2> <COUNT> (Verify Memory) .....	4
8080 I/O Port Commands.....	4
IN <PORT> (Input from Port) .....	4
OT <PORT> <DATA> (Output to Port) .....	4
Execution Commands.....	4
BO [<Drive>] (Boot from Floppy Disk) .....	4
CE <Command Line Text> {ROM2K} (Execute CP/M program) .....	4
EX [<ADDRESS>][<Option>] (Execute) .....	5
Transfer Port Commands.....	5
HD <ADDRESS> <COUNT> (Dump Memory as Intel Hex File) .....	5
HL [<OFFSET>] (Load Intel Hex File into memory) .....	5
TB <BAUD> {ROM2K} (Set Transfer Port Baud Rate) .....	5
TE [<EXITCHR>] (Terminal Mode) .....	6
TP [<0/1>] (Set Transfer Port) .....	6
Configuring Memon/80.....	6
1.Specify the CPU Speed: CPUMHZ equ <1-6> .....	6
2. Specify the ROM Size: ROM2K equ TRUE/FALSE .....	6
3.Specify Memon/80's Memory Utilization.....	7
Memon/80's EPROM address: MEBASE equ <Address> .....	7
RAM Execution: RAMCOD equ TRUE/FALSE .....	7
RAM Hunt: RAMHNT equ TRUE/FALSE .....	7

Memon/80's RAM Page:	RAMEND equ <Address> .....	7
Include Help Command:	HELPC equ TRUE/FALSE .....	7
Allow Lowercase Input:	LOWERC equ TRUE/FALSE .....	8
Enable Record Count for HL Command:	HLRECS equ TRUE/FALSE .....	8
Enable Repeat Option for CO Command:	CORPT equ TRUE/FALSE .....	8
Enable ROM-Disable Option for EX Command:	EXOPT equ TRUE/FALSE ..	8
Prettier Output for the DU Command:	DUPRTY equ TRUE/FALSE .....	8
4. Specify the Console Port:	<PORTNAME> equ TRUE/FALSE .....	8
5. Specify the Transfer Port:	<PORTNAME> equ TRUE/FALSE .....	8
6. Set The Serial Port Base Addresses .....		9
Console Port I/O Address:	CBASE equ <Address> .....	9
Transfer Port I/O Address:	TBASE equ <Address> .....	9
7. Set The Serial Port Baud Rates .....		9
Console Port Baud Rate:	CBAUD equ <Value> .....	9
Transfer Port Default Baud Rate:	TBAUD equ <Value> .....	9
8. Specify The Disk Controller:	<Controller> equ TRUE/FALSE .....	10
9. Set the Disk Controller Base Addresses .....		10
Disk Controller I/O BASE Address:	DIBASE equ <address> .....	10
Disk Controller Memory BASE Address:	DMBASE equ <address> .....	10
10. Specify whether or not interrupts will be enabled .....		10
Example: Using Memon/80 to Build a CP/M Disk .....		11
Example: Programming an EPROM .....		12
Software Entry Points .....		13
MEINIT (xx00) (Restart Memon/80) .....		13
MECSTA (xx06) (Get Console Status) .....		13
MECIN (xx09) (Get Console Input) .....		13
MECOUT (xx0C) (Console Output) .....		13
MEPBSI (xx0F) (Get Transfer Port Input Status) .....		14
MEPBDI (F412) (Get Transfer Port data Input) .....		14
MEPBDO (F415) (Transfer Port Output) .....		14
MEPBSO (F418) (Get Transfer Port Output Status) .....		14
Memon/80 Version 2.3 Source Code Listing .....		15

## **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<sup>2</sup>I/O
- Heathkit H8-4 serial/cassette interface
- Heathkit H8-5 quad serial port
- IMSAI SIO-2
- Ithaca Intersystems Series II VIO's serial ports
- Jade Serial/Parallel I/O Board's serial ports
- Micromation Doubler disk controller's serial port
- Processor Technology 3P+S's serial port
- Salota I/O 2+1's serial ports
- Solid State Music IO4's SERIAL 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) <sup>1</sup>
- SD Systems Versafloppy single-density FDC<sup>1</sup>
- SD Systems Versafloppy II double-density FDC<sup>1</sup>
- Tarbell 1011 8" Single-density FDC
- Tarbell 2022 8" Double-density FDC

---

<sup>1</sup> 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.

## 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.

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,0D,'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 *<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 any 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 last 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 does 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: TBAUD 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 B0 (boot from floppy disk) command. If you do not want the B0 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 set TRUE, and Memon/80 will run with interrupts enabled, only masking them when timing is critical. Otherwise, set ENINTS set 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<sup>2</sup>
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.<sup>3</sup>
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

---

<sup>2</sup> An 8K Bytesaver has eight sockets, each of which can read or program a 2708 EPROM.

<sup>3</sup> 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.

##### ***MECSTA (xx06) (GET CONSOLE 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.

##### ***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.

**MEPBSI (xx0F) (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.

**MEPBDI (F412) (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.

**MEPBDO (F415) (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.

**MEPBSO (F418) (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.

#### MEMON/80 VERSION 2.4 SOURCE CODE LISTING

Example configuration:

- 2KROM = TRUE
- Console port = CCS 2410 CPU's serial port
- Transfer Port = CCS 2719 Port A
- Disk controller = CCS 2422

# MEMON80.PRN

=====

; Simple, configurable 1K-byte or 2K-byte ROM-based monitor for  
; an 8080- or Z80-based system, supporting a variety of serial  
; ports and floppy disk controllers.

; See comments further down for a list and description of  
; Memon/80's commands.

; Memon/80 uploads and downloads data either the console or a  
; second serial port (called the Transfer Port).

; Memon/80 supports the following serial ports for either the  
; console or the Transfer Port, as selected with assembly  
; options below.

; 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 Comouter Systems 2718's serial ports  
; California Computer Systems 2719  
; California Computer Systems 2810 CPU's serial port  
; Compupro Interfacer 1  
; Compupro Interfacer II  
; Cromemco 4FDC/16FDC/64FDC disk controller's serial port  
; Cromemco TU-ART  
; Electronic Control Technology (ECT) R2IO  
; Heathkit H8-4 quad serial port  
; Heathkit H8-5 Console/Cassette controller's console port  
; (assuming a PAM-8 or XCON-8 ROM is installed in the H8)  
; IMSAI SIO-2  
; Ithaca Intersystems Series II VIO's serial ports  
; Jade Serial/Parallel I/O Board's serial ports  
; Micromation Doubler disk controller's serial port  
; Processor Technology 3P+S's serial port  
; Salota I/O 2+1's serial ports  
; Solid State Music IO4's serial ports (Configured as one of  
; the other supported serial port 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

; Memon/80's BO command boots from the following floppy  
; disk controllers, selected with assembly options below.

; Altair 88-DCDD 8"  
; Altair 88-MDS Minidisk  
; California Computer Systems 2422  
; Cromemco 4FDC/16FDC/64FDC  
; Heathkit H17 Minifloppy  
; IMSAI FIF (FIB & IFM)  
; IMSAI MDC-DIO  
; Micromation Doubler  
; Micropolis - FD Control B  
; Northstar MDC-A Minidisk  
; Northstar MDS-A Minidisk  
; Northstar MDS-D Double-density Minidisk  
; Salota FDC  
; SD Systems Versafloppy (Single-density)  
; SD Systems Versafloppy II (Double-density)  
; Tarbell 1011 8" (Single-density)

```

MEMON80.PRN
; Tarbell 2022 8" (Double-density)
;
; Formatted to assemble with digital Research's ASM.
;=====
0000 = FALSE equ 0
FFFF = TRUE equ NOT FALSE

;=====
;= Memon/80 Option Selection =
;=
;= Set the required options in the following 10 sections =
;=
;= Select Memon/80 functionality by setting these =
;= variables according to the guidelines in each section. =
;=====

;=====
;= 1. Specify the CPU Speed =
;=====
; (Only used to adjust time-out loops for floppy disk booting)
;-----
0004 = CPUHZ equ 4 ;Integer CPU speed in MHZ

;=====
;= 2. Specify the ROM Size =
;=====
; 2K-Byte EPROM option adds several commands,
; and extends a few other commands.
; ROM2K equ TRUE selects a 2K EPROM (e.g. 2716)
; ROM2K equ FALSE selects a 1K EPROM (e.g. 2708)
;-----
FFFF = ROM2K equ TRUE

;=====
;= 3. Specify Memon/80's Memory Utilization =
;=====
; These specify where Memon/80 is place in memory and where
; its buffers, stack, and variables are. Additionally, you
; can disable some Memon/80 features to reduce the code size.
;-----
F000 = MEBASE equ 0F000h ;Base address of Memon/80 code
0000 = RAMCOD equ FALSE ;Set to true if executing from RAM in
; low memory (e.g. at 0100h for CP/M).
FFFF = RAMHNT equ TRUE ;TRUE causes Memon to hunt for the
; highest contiguous RAM page for its
; stack, buffer, and variables.
; FALSE uses RAM just below RAMEND,
; and saves at least 26 bytes. Must be
; FALSE for I/O boards using interrupts.

F49F = RAMEND equ 0F49Fh ;Last allowed RAM address for MEMON/80.
; All of Memon's RAM must fit in one
; 256-byte page. This means the low 2
; hex digits must be larger than 70h. (a
; few more if your I/O ports use
; interrupts.) -->Must be 0xFFh for
; Altair disk drives<--

FFFF = HELPC equ TRUE ;TRUE includes a minimal help command.
; FALSE saves 40 to 60 bytes.

```

```

MEMON80.PRN
FFFF =      LOWERC  equ      TRUE      ;TRUE allows lowercase input too.
                                           ;FALSE saves 11 bytes.

FFFF =      HLRECS  equ      TRUE      ;TRUE: HL command reports total number
                                           ;of records received when done.
                                           ;FALSE: saves 25 bytes.

FFFF =      CORPT   equ      TRUE      ;TRUE includes the <RPT> option for the
                                           ;CO (copy) command.
                                           ;FALSE saves 36 bytes.

0000 =      EXOPT   equ      FALSE     ;TRUE includes the EX command option to
                                           ;disable the ROM on IN from FFh (for
                                           ;Altair-type boards).
                                           ;FALSE saves 17 bytes.

FFFF =      DUPRTY  equ      TRUE      ;TRUE makes the DU output a little
                                           ;prettier
                                           ;FALSE saves 17 bytes

;=====
;= 4. Specify the Console Port =
;=====
;Exactly one of these must be TRUE
;-----
0000 =      CALSIO  equ      FALSE     ;Altair 88-SIO
0000 =      CA2SIO  equ      FALSE     ;Altair 88-2SIO (Port A or B)
                                           ;or 8800b Turnkey Module or 88-UIO
0000 =      CC2718A equ      FALSE     ;CCS 2718 Serial Port A
0000 =      CC2718B equ      FALSE     ;CCS 2718 Serial Port B
0000 =      CC2719A equ      FALSE     ;CCS 2719 Port A
0000 =      CC2719B equ      FALSE     ;CCS 2719 Port B
FFFF =      CC2810  equ      TRUE      ;CCS 2810 CPU'S serial port
0000 =      CIFAC   equ      FALSE     ;Compupro Interfacer 1 (Port A or B)
0000 =      CIFAC2  equ      FALSE     ;Compupro Interfacer II
0000 =      CTUART  equ      FALSE     ;Cromemco TU-ART (Port A or B)
0000 =      CCFDCS  equ      FALSE     ;Cromemco 4FDC/16FDC/64FDC serial port
0000 =      CER2IO  equ      FALSE     ;ECT R2IO (Port A, B, or C)
0000 =      CH84     equ      FALSE     ;Heathkit H8-4 Serial (any of the 4)
0000 =      CH85     equ      FALSE     ;Heathkit H8-5's console port
0000 =      CISIO2A equ      FALSE     ;IMSAI SIO-2 Port A
0000 =      CISIO2B equ      FALSE     ;IMSAI SIO-2 Port B
0000 =      CIVIO2  equ      FALSE     ;Ithaca Intersystems VIO-II Port A or B
0000 =      CJADSP  equ      FALSE     ;Jade Serial/Parallel I/O (port A or B)
0000 =      CMDUBLR equ      FALSE     ;Micromation Doubler serial port
0000 =      CPT3PS  equ      FALSE     ;Processor Technology 3P+S
0000 =      CSAL21  equ      FALSE     ;Salota I/O 2+1 (Port A or B)
0000 =      CSIO5   equ      FALSE     ;Solid State Music IO5 Serial (A or B)
0000 =      CTARBL  equ      FALSE     ;Tarbell 4044 (Port A, B, C, or D)
0000 =      CBITS1  equ      FALSE     ;Vector Graphic BitStreamer
0000 =      CBITS2  equ      FALSE     ;Vector Graphic BitStreamer II Port A,B,C
0000 =      CWIOB1  equ      FALSE     ;Wameco IOB-1's serial port

```

```

;=====
;= 5. Specify the Transfer Port =
;=====
;At most one of these must be TRUE. If none is TRUE then the
;following commands will be deleted: TB (set Transfer Port
;baud rate), TE (Terminal mode), and TP (Select transfer
;Port). The Transfer Port must not also be the Console Port
;-----

```

```

0000 =      TALSIO  equ      FALSE     ;Altair 88-SIO
0000 =      TA2SIO  equ      FALSE     ;Altair 88-2SIO (Port A or B)

```

# MEMON80.PRN

```

0000 = TC2718A equ FALSE ;or 8800b Turnkey Module or 88-UIO
0000 = TC2718B equ FALSE ;CCS 2718 Serial Port A
FFFF = TC2719A equ TRUE ;CCS 2718 Serial Port B
0000 = TC2719B equ FALSE ;CCS 2719 Port A
0000 = TC2810 equ FALSE ;CCS 2719 Port B
0000 = TIFAC equ FALSE ;CCS 2810 CPU'S serial port
0000 = TIFAC2 equ FALSE ;Compupro Interfacer 1 (Port A or B)
0000 = TTUART equ FALSE ;Compupro Interfacer II
0000 = TCFDCS equ FALSE ;Cromemco TU-ART (Port A or B)
0000 = TER2IO equ FALSE ;Cromemco 4FDC/16FDC/64FDC serial port
0000 = TH84 equ FALSE ;ECT R2IO (Port A, B, or C)
0000 = TH85 equ FALSE ;Heathkit H8-4 (any of the 4)
0000 = TISIO2A equ FALSE ;Heathkit H8-5's console port
0000 = TISIO2B equ FALSE ;IMSAI SIO-2 Port A
0000 = TIVIO2 equ FALSE ;IMSAI SIO-2 Port B
0000 = TJADSP equ FALSE ;Ithaca Intersystems VIO-II Port A or B
0000 = TMDUBLR equ FALSE ;Jade Serial/Parallel I/O (port A or B)
0000 = TPT3PS equ FALSE ;Micromation Doubler serial port
0000 = TSAL21 equ FALSE ;Processor Technology 3P+S
0000 = TSIO5 equ FALSE ;Salota I/O 2+1 (Port A or B)
0000 = TTARBL equ FALSE ;Solid State Music IO5 Serial (A or B)
0000 = TBITS1 equ FALSE ;Tarbell 4044 (Port A, B, C, or D)
0000 = TBITS2 equ FALSE ;Vector Graphic BitStreamer
0000 = TWIOB1 equ FALSE ;Vector Graphic BitStreamer II Port A,B,C
0000 = Wameco IOB-1's serial port

```

```

=====
;= 6. Set the Serial port base addresses =
=====

```

```

;Here are the default addresses for various boards. Make sure
;that CBASE is not the same as TBASE, and that their address
;ranges do not overlap, unless they are on the same serial
;board, sharing address space. (some board combinations will
;require you to change the address of at least one board to a
;non-default address.) Also make sure that neither serial port
;interferes with the addresses used by the disk controller,
;if you are enabling the BO (boot from floppy) command.

```

Serial Port Board	Port	Port Base Address	Board Address Range
Altair 88-SIO		00h	00h-01h
Altair 88-2SIO	Port A	10h	10h-13h
	Port B	12h	
Altair 8800b Turnkey	Serial	10h	10h-11h
MITS 88-UIO	Serial	10h	10h-11h
CCS 2718	Ser A	00h	00h-03h
(Jumpers SPR=N,SAI=N,SBI=I)	Ser B	00h	
CCS 2719	Port A	50h	50h-57h
	Port B	50h	
CCS 2810	Serial	20h	20h-26h
Compupro Interfacer 1	Port A	00h	00h-03h
	Port B	02h	
Compupro Interfacer II		00h	00h-01h

# MEMON80.PRN

;				
;	Cromemco TU-ART	Port A	20h	20h-23h
;		Port B	50h	50h-53h
;				
;	Cromemco 4FDC/16FDC/64FDC	Serial	00h	00h-03h
;				
;	ECT R2I/O	Port A	00h	00h-07h
;		Port B	02h	
;		Port C	04h	
;				
;	Heathkit H8-4	Console	0E8h	0E8h-0EDh
;		Lineprinter	0E0h	0E0h-0E6h
;		Alt Term 0	0C0h	0C0h-0C6h
;		Alt Term 1	0C8h	0C8h-0CDh
;		Alt Term 2	0D0h	0D0h-0D6h
;		Alt Term 3	0D8h	0D8h-0DDh
;				
;	Heathkit H8-5	Console	0FAh	0F8h-0FBh
;				
;	IMSAI SIO-2	Port A	02h	02h-06h
;		Port B	04h	
;				
;	Ithaca Intersystems VIO-II	Port A	00h	00h-1Fh
;		Port B	04h	
;				
;	Jade Serial/Parallel I/O	Port A	00h	00h-02h
;		Port B	01h	& 80h-82h
;				
;	Micromation Doubler	Serial	0F800h	0F800h-0FFFFh
;	(memory-mapped)			
;				
;	Processor Technology 3P+S	C & D	00h	00h-03h
;				
;	Salota I/O 2+1	Serial A	80h	80h-8Eh
;		Serial B	82h	
;				
;	Solid State Music IO5	Serial A	00h	00h-0Dh
;		Serial B	02h	
;				
;	Tarbell 4044 4S2P IO	Port A	11h	10h-18h
;		Port B	13h	
;		Port C	15h	
;		Port D	17h	
;				
;	VG BitStreamer	-	02h	02h-03h
;				
;	VG BitStreamer II	Port A	02h	02h-09h
;		Port B	04h	
;		Port C	06h	
;				
;	Wameco IOB-1	Serial	04h	00h-05h

00FA =  
00E8 =

CBASE equ 0FAh ;Console Port base address  
TBASE equ 0E8h ;Transfer Port base address

=====

;= 7. Set the Serial Port Baud Rates =

=====

;Set the Console baud rate and the initial Transfer Port  
;baud rate, for boards that support software-controlled  
;baud rates. (Note that the TB command is only available  
;if ROM2K is TRUE and a Transfer Port has been specified.)  
;The following supported boards allow software to set the



# MEMON80.PRN

;baud rate:

Serial Port Board	Max Rate	Notes
CCS 2718	19200	
CCS 2719	57600	
CCS 2810	38400	
Cromemco TU-ART	76800	no 600 baud
Cromemco 4FDC/16FDC/64FDC	76800	no 600 baud
Heathkit H8-4	38400	
Ithaca Intersystems System II VIO	19200	

;Specify the baud rates from this table:

Baud Rate	Stop Bits	CBAUD/TBAUD Value
110	2	0
150	1	1
300	1	2
600	1	3
1200	1	4
2400	1	5
4800	1	6
9600	1	7
19200	1	8
38400	1	9
57600	1	10
76800	1	11

0007 = CBAUD equ 7 ;Console baud rate  
0007 = TBAUD equ 7 ;Initial Transfer Port baud rate

;=====

;= 8. Specify the Disk controller =

;=====

;The disk controller is only used for the BO command.  
;At most one of these may be TRUE. If none is TRUE,  
;then the BO (boot) command will be deleted.

0000 = A88DCD equ FALSE ;MITS Altair 88-DCDD  
0000 = A88MCD equ FALSE ;MITS Altair 88-MDS  
FFFF = CC2422 equ TRUE ;CCS 2422 Disk controller  
0000 = C4FDC equ FALSE ;Cromemco 4FDC  
0000 = C16FDC equ FALSE ;Cromemco 16FDC  
0000 = C64FDC equ FALSE ;Cromemco 64FDC  
0000 = H17R equ FALSE ;Heathkit H17 (onboard boot ROM),  
0000 = IFIF equ FALSE ;IMSAI FIF  
0000 = IMD8R equ FALSE ;IMSAI MDC-DIO with 8" disk (onboard  
;..boot ROM  
0000 = IMDMR equ FALSE ;IMSAI MDC-DIO with minidisk (onboard  
;..boot ROM  
0000 = MDUBLR equ FALSE ;Micromation Doubler (onboard boot  
;..ROM  
0000 = MICROP equ FALSE ;Micropolis FD controller B  
0000 = MICROR equ FALSE ;Micropolis FD Controller B (onboard  
;..boot ROM  
0000 = NSTARS equ FALSE ;Northstar MDC-A and MDS-A  
0000 = NSTRSR equ FALSE ;Northstar MDC-A and MDS-A (onboard  
;..boot ROM  
0000 = NSTARD equ FALSE ;Northstar MDS-D  
0000 = NSTRDR equ FALSE ;Northstar MDS-D (onboard boot ROM)  
0000 = SALFDC equ FALSE ;Salota FDC (same as Versafloppy II)

```

                                MEMON80.PRN
0000 =   VERSA1 equ      FALSE   ;SD Systems Versafloppy
0000 =   VERSA2 equ      FALSE   ;SD Sales Versafloppy II
0000 =   TARBL1 equ      FALSE   ;Tarbell 1011
0000 =   TARBL2 equ      FALSE   ;Tarbell 2022

```

```

;=====
;= 9. Set the Disk Controller's Base Adresse(s) =
;=====
;Here are the default base addresses for various boards. Note
;that some disk controllers use I/O address, some use memory
;addresses, and some use both. Also, some of these boards do
;not allow you to change their I/O addresses and/or memory
;addresses.

```

	I/O Base Address	I/O Top Address	Memory Base Address	Memory Top Address
Disk Controller				
-----				
; MITS Altair 88-DCDD/88-MDS	08h	0Ah		
; CCS 2422 Disk controller	{04h & 30h-34h}			
; Cromemco 4FDC/16FDC/64FDC	{04h & 30h-34h}			
; Heathkit H17			01800h	01FFFh
; IMSAI FIF	0FDH	0FDH		
; IMSAI MDC-DIO	0D0h	0DFh	0E000h	0EFFFh
; Micromation Doubler			0F800h	0FFFFh
; Micropolis FD controller B			0F400h	0F7FFh
; Northstar MDC-A and MDS-A			0E800h	0EBFFh
; Northstar MDS-D			0E800h	0EBFFh
; Systems Versafloppy	060h	067h		
; Tarbell 1011/2022	0F8h	0FFh		
-----				

```

0004 =   DIBASE equ      004h   ;Disk Controller I/O Base Address
0000 =   DMBASE equ      0000h  ;Disk controller memory Base Address

```

```

;=====
;= 10. Specify whether or not interrupts are enabled =
;=====
;"ENINTS equ TRUE" will generally leave interrupts enabled,
;masking them only during initialization and when time-critical
;code is running. Unless your computer requires interrupts to
;be enabled, set this to FALSE.
;
;"ENINTS equ FALSE" will mask interrupts always, and save
;several bytes of codespace, and protect you from random
;interrupts and potential code crashes in a computer where
;interrupts are not set up.
;
;Some I/O ports (e.g. the Heathkit H8-5) require interrupts.
;If such a board is specified above, then interrupts will get
;enabled anyway, (and the necessary interrupt code will be
;installed) regardless of what you enter here.

```

```

0000 =   ENINTS equ      FALSE

```

```

;=====
;= End of Assembly Options =
;= No user options below here. =
;=====

```

```

;=====
;Memon/80 Commands (all values are in HEX)
;
; BO <DRIVE> {Some controllers require ROM2K}

```

```

; MEMON80.PRN
; Boot from floppy disk. The <DRIVE> option ( a number
; between 0 and 3) is only available for the Cromemco disk
; controllers, because CDOS is one of the few S100 operating
; systems that allows booting from anything but the first
; drive.
;
; CE [command line] {Requires ROM2K}
; Execute CP/M program: Copy Command line (any text string)
; to RAM at 0080h, install WBOOT jump at 0000h, and jump to
; program at 0100h.
;
; CO <SRC> <DST> <BCNT> [<RPT>]
; Copy <BCNT> bytes of memory from address <SRC> to address
; <DST>. The <RPT> option is available only if CORPT=TRUE.
; This optionally repeats the copy <RPT> times. (This is for
; programming EPROMS with e.g. a Cromemco bytesaver).
;
; DU [<ADR> [<BCNT>]]
; Dump <BCNT> (which defaults to 1) bytes of memory in both
; hex and ASCII, starting at address <ADR> (defaults to 0).
;
; EN [<ADR>]
; Enter hex data into memory at <ADR>, which defaults to 0.
; values are separated with spaces or CR'S. Quit EN command
; with a blank line.
;
; EX [[<ADR>] <OPT>]
; Execute at <ADR>, which defaults to 0. Programs can RET
; to Memon/80's main loop.
; <OPT> is available only if ROM2K=TRUE. If <OPT>=1 then
; Memon/80 executes an IN from port FFh before jumping to
; the specified address. (This disables PROM on the
; MITS 8800b Turnkey Module and on my 88-2SIOJP.)
;
; FI [<ADR> [<BCNT> [<VAL>]]]
; Fill <BCNT> bytes of memory starting at <ADR> with <VAL>,
; <VAL> and <ADR> default to 0. <BCNT> defaults to all RAM.
; Stop when fill reaches Memon/80's RAM page.
;
; HD <ADR> <BCNT>
; Intel hex dump <BCNT> bytes of memory starting at <ADR>,
; to the Transfer Port
;
; HL [<OFST>]
; Load Intel hex file to memory from the Transfer Port. Add
; optional address offset <OFST> to each record address.
; This will generate a "Mem" error if a record will write
; over Memon/80's RAM page (or RAM image, if RAMCOD=TRUE).
; HL will report the total number of records received when
; an Intel hex EOF record is received, if ROM2K=TRUE.
;
; IN <PORT>
; Input from <PORT> and print result on Console
;
; MT <ADR> <CNT> {Requires ROM2K}
; Test <CNT> bytes of memory, starting at <ADR>. 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 uses for its stack.
;
; OT <PORT> <data>
; Output <data> to <PORT>
;

```

```

; MEMON80.PRN
; SE <ADR> <Byte1> [<byte2> [<byte3> [...<Byten>]]]
; or
; SE <ADR> 'text string' {Requires ROM2K}
; Search for string in memory, starting at address <ADR>
; Can also mix forms e.g.
; SE 100 'Hello world' 0D 0A 'second line'
;
; TB <BAUD> {Requires ROM2K}
; {deleted if no Transfer Port specified}
; Set Transfer Port baud rate, <BAUD> from this table:
; Value Baud Rate
; 0 110 (2 stop bits)
; 1 150
; 2 300
; 3 600
; 4 1200
; 5 2400
; 6 4800
; 7 9600
; 8 19200
; 9 38400
; A 76800
;
; This command is only available if the serial port board
; that's selected to be the Transfer Port allows software to
; set its baud rate. Not all baud rates are available for
; every board. Attempting to select an unsupported baud rate
; will result in a command error.
;
; TE [<EXCHR>] {deleted if no Transfer Port specified}
; Terminal mode: Console keyboard input goes to Port B
; output, and Port B input goes to the Console output.
; Type CTRL-<EXCHR> (which defaults to CTRL-C) to exit.
;
; TP [<0/1>] {deleted if no Transfer Port specified}
; Enable the Transfer Port. "TP 0" disables the Transfer
; port, causing Transfer Port operations to use the Console
; Port instead. defaults to 1 (enabled).
;
; VE <SRC> <DST> <BCNT>
; Verify <BCNT> bytes of memory, starting at <SRC> and <DST>
; =====
; REVISION HISTORY
;
; VERS. 2.4 by M. Eberhard 26 December 2018
; Work around ASM bug: "set" variables that control
; conditional assembly must be declared as 0 (FALSE) first,
; before any other value gets assigned to them. Some
; additional cleanup as well.
;
; VERS. 2.3 by M. Eberhard 3 November 2018
; Support the Heathkit H8-4. Support the H8-5, which requires
; interrupts for receiving data. Allow interrupts to be
; enabled, generally. Allow for no transfer port to be
; specified and eliminate all transfer port commands if no
; transfer port is defined. Support H17 disk controller. Allow
; simple disk boots to work without ROM2K=FALSE. Allow more
; assembly options to save code space.
;
; VERS. 2.2 by M. Eberhard 1 October 2018
; Fix pseudo-op bug when assembling with ROM2k=FALSE
;
; VERS. 2.1 by M. Eberhard 26 April 2018

```

```

MEMON80.PRN
; Correct comments for IMSAI 2SIO ports. (Thanks, Len B.)
; Support many more I/O boards and disk controllers. Correct a
; few comments. Add CE command.
;
;VERS. 2.0 by M. Eberhard 31 December 2016
; Eliminate Module interface. default TP to enabled.
; ROM2K option adds a bunch of commands, requires 2K EPROM.
; (Unreleased version)
;
;VERS. 1.1 by M. Eberhard 9 November 2016
; Add IN and OT commands. Better defaults and RAM overwrite
; protection for FI. check each byte for RAM page overwrite
; during HL. disable ints for the IMSAI SIO-2. add support for
; several more serial ports. Squeeze code. Eliminate record-
; type test on HL. Improve comments & labels a lot.
;
;VERS. 1.00 by M. Eberhard 14 April 2014
; Created
;=====

0024 = VERSION equ 24h ;BCD-encoded version number

        if not ROM2K
ROMSIZ equ 400h ;1K EPROM size (e.g. 2708)
        endif ;not ROM2K
        if ROM2K
0800 = ROMSIZ equ 800h ;2K EPROM size (e.g. 2716)
        endif ;ROM2K

;*****
;General program Equates
;*****
0003 = CTRLC equ 03h ;ASCII control-C
0008 = BS equ 08h ;ASCII backspace
000A = LF equ 0Ah ;ASCII linefeed
000D = CR equ 0DH ;ASCII carriage return
0027 = QUOTE equ 27h ;ASCII single quote
007F = DEL equ 7Fh ;ASCII delete

003E = PROMPT equ '>' ;prompt character
0003 = CABKEY equ CTRLC ;command abort character
0003 = DTEXTIT equ CTRLC ;default Terminal mode exit chr
002E = PCFIER equ '.' ;Console pacifier chr
0010 = HDRLEN equ 16 ;Intel hex record length for HD

0020 = STKSIZ equ 32 ;Max stack size
0050 = LBSIZE equ 80 ;input line buffer size

;*****
;RAM Page Organization
;*****
        if RAMHNT and not (A88DCD or A88MCD)
00B0 = RAMBUF equ 100h-LBSIZE ;buffer offset in RAM page
        endif ;RAMHNT and not (A88DCD or A88MCD)

        if not RAMHNT and not (A88DCD or A88MCD)
RAMBUF equ RAMEND-LBSIZE+1 ;buffer address in RAM page
        endif ;not RAMHNT and not (A88DCD or A88MCD)

;The Altair disk controller code requires the buffer to be at
;the end of a 256-byte page

        if RAMHNT and (A88DCD or A88MCD)

```

```

                                MEMON80.PRN
RAMBUF equ    100h-SECSIZ      ;Exactly room for 1 sector
endif ;RAMHNT and (A88DCD or A88MCD)

    if (not RAMHNT) and (A88DCD or A88MCD)
RAMBUF equ    RAMEND-SECSIZ+1 ;Exactly room for 1 sector
endif ;(not RAMHNT) and (A88DCD or A88MCD)

;The stack goes just below the RAM buffer

00B0 =    STACK equ    RAMBUF          ;SP decrements before writing

;Assign for interrupt flags, even if we don't use them
;Note: if interrupts are used then RAMHNT MUST be FALSE

008F =    CIFLAG equ    STACK-STKSIZ-1 ;console interrupt Rx flag
008E =    CRXBUF equ    STACK-STKSIZ-2 ;console interrupt Rx buffer
008D =    TIFLAG equ    STACK-STKSIZ-3 ;T-port interrupt Rx flag
008C =    TRXBUF equ    STACK-STKSIZ-4 ;T-port interrupt Rx buffer

;*****
;CP/M Equates (for CE command)
;The CE command creates a CP/M BIOS-like jump to Memon/80's
;warm-boot entry point, which is followed by a CP/M BIOS-
;like jump table to its I/O routines. The CE command also
;moves the command-line text entered by the user to the
;CP/M-like COMBUF. These make execution possible for simple
;CP/M programs that only call the BIOS for console I/O, and
;make no BDOS calls. This command is especially useful for
;building a CP/M disk from scratch, allowing you to load
;(via the HL command) and run (via the CE command) programs
;like FORMAT and PUTSYS. Once you have a diskette with a bare
;CP/M and a working BIOS, you can use these commands to load
;CP/M's non-built in commands onto the diskette.
;*****
    if ROM2K
0000 =    WBOOT equ    0000H          ;Jump to "BIOS" warm boot
0001 =    WBOOTA equ    WBOOT+1       ;Address of Warm Boot
0080 =    COMBUF equ    WBOOT+80H     ;command line buffer
0100 =    USAREA equ    WBOOT+100H    ;User program area
    endif ;ROM2K

;=====
;= Option Selection Logic                                     =
;=                                                           =
;= The following long section of equates performs         =
;= the logic for supporting the various serial             =
;= ports and disk controllers, as selected above.         =
;= Generally, these should not be modified.                =
;=====

;-----
;Initialize some assembler variables that
;may get changed, based on options set above
;-----
TMAP set    FALSE
CMEMAP set   FALSE
TMRCTC set   FALSE
T5511 set    FALSE
C6850 set    FALSE
T6850 set    FALSE
C8250 set    FALSE
T8250 set    FALSE
C8251 set    FALSE

```

# MEMON80.PRN

```

T8251    set    FALSE
DART     set    FALSE
CISTAT   set    FALSE
TISTAT   set    FALSE
CSBAUD   set    FALSE
TSBAUD   set    FALSE
BD192    set    FALSE
BD384    set    FALSE
BD576    set    FALSE
BD768    set    FALSE
CRXINT   set    FALSE
TRXINT   set    FALSE
TPORT    set    FALSE

```

```

;-----
;Altair 88-SIO Equates
;
;The 88-SIO is built around a 2502-type generic UART,
;with a jumper selectable baud rate generator and with
;external logic providing status and control registers.
;
;Note that the 88-SIO requires no initialization.
;-----
if CALSIO

```

```

CSTAT    equ    CBASE          ;1st SIO status input port
CDATA    equ    CBASE+1        ;1st SIO Rx & Tx data regs

CRXRDY    equ    00000001B      ;Console Rx data reg full
CTXRDY    equ    10000000B      ;Console Tx data reg empty

CISTAT    set    TRUE           ;Status bits are active-low
endif ;CALSIO

if TALSIO
TPORT     set    TRUE           ;Transfer port is defined

TSTAT     equ    TBASE          ;2nd SIO status input port
TDATA     equ    TBASE+1        ;2nd SIO Rx & Tx data regs

TRXRDY    equ    00000001B      ;Transfer Port Rx data reg full
TTXRDY    equ    10000000B      ;Trans. port Tx data reg empty

TISTAT    set    TRUE           ;Status bits active-low
endif ;TALSIO

```

```

;-----
;Altair 88-2SIO, 8800b Turnkey Module, and MITS 88-UIO Equates
;
;The 88-2SIO, the 8800b Turnkey Module, the 88-UIO, and my own
;88-2SIOJP are all built around a pair of MC6850 ACIA'S with a
;jumper-selectable baud rate generator.
;-----

```

```

if CA2SIO
C6850     set    TRUE           ;Altair 6850-type UART present
CB6850    equ    CBASE
endif ;CA2SIO

if TA2SIO
TPORT     set    TRUE           ;Transfer port is defined
T6850     set    TRUE           ;Altair 6850-type UART present
TB6850    equ    TBASE
endif ;TA2SIO

```

```

;-----
;CCS 2718 Equates
;
;The CCS 2718's serial port A is built around an 1883/1602/1611
;generic UART. Its serial port B is built around an 8251 UART.
;Both UARTs are clocked by an external baud rate generator that
;can be forced to a chosen baud rate for each port with
;jumpers, or jumpered for software baud rate control. This
;code assumes that software can control the baud rates.
;Additionally, the control register for serial port A is
;configured with jumpers. This code assumes the configuration
;is as shown on page 2-8 of the CCS 2718 manual.
;-----

```

```

if CC2718A
CSTAT equ CBASE ;serial port A status
CCTRL equ CBASE ;channel a control
CDATA equ CBASE+1 ;serial port A data

CTXRDY equ 01h ;Console Tx ready bit
CRXRDY equ 02h ;Console Rx ready bit

CBD110 equ 10Fh ;0= 110 baud
CBD150 equ 10Eh ;1= 150 baud
CBD300 equ 10Dh ;2= 300 baud
CBD600 equ 106h ;3= 600 baud
CBD1200 equ 10Bh ;4= 1200 baud
CBD2400 equ 10Ch ;5= 2400 baud
CBD4800 equ 109h ;6= 4800 baud
CBD9600 equ 108H ;7= 9600 baud
CBD192 equ 0 ;8= 19200 baud
CBD384 equ 0 ;9= 38400 baud (not supported)
CBD576 equ 0 ;a= 57600 baud (not supported)
CBD768 equ 0 ;b= 76800 baud (not supported)

CSBAUD set TRUE ;software Console baud rate control
endif ;CCS2718A

```

```

if CC2718B
C8251 set TRUE
CB8251 equ CBASE+2 ;Base of Console 8251 registers

CBD110 equ 1F0h ;0= 110 baud
CBD150 equ 1E0h ;1= 150 baud
CBD300 equ 1D0h ;2= 300 baud
CBD600 equ 160h ;3= 600 baud
TBD1200 equ 1B0h ;4= 1200 baud
CBD2400 equ 1C0h ;5= 2400 baud
CBD4800 equ 190h ;6= 4800 baud
CBD9600 equ 180H ;7= 9600 baud
CBD192 equ 0 ;8= 19200 baud
CBD384 equ 0 ;9= 38400 baud (not supported)
CBD576 equ 0 ;a= 57600 baud (not supported)
CBD768 equ 0 ;b= 76800 baud (not supported)

CSBAUD set TRUE ;software Console baud rate control
endif ;CCS2718B

```

```

if TC2718A
TPORT set TRUE ;Transfer port is defined

TSTAT equ TBASE ;serial port A status
TCTRL equ TBASE ;channel a control

```



```

MEMON80.PRN
TDATA    equ    TBASE+1 ;serial port A data

TTXRDY    equ    01h    ;Transfer Port Tx ready bit
TRXRDY    equ    02h    ;Transfer Port Rx ready bit

TBD110    equ    10Fh    ;0= 110 baud
TBD150    equ    10Eh    ;1= 150 baud
TBD300    equ    10Dh    ;2= 300 baud
TBD600    equ    106h    ;3= 600 baud
TBD1200   equ    10Bh    ;4= 1200 baud
TBD2400   equ    10Ch    ;5= 2400 baud
TBD4800   equ    109h    ;6= 4800 baud
TBD9600   equ    108H    ;7= 9600 baud
TBD192    equ    100h    ;8= 19200 baud
TBD384    equ    0       ;9= 38400 baud (not supported)
TBD576    equ    0       ;a= 57600 baud (not supported)
TBD768    equ    0       ;b= 76800 baud (not supported)

TSBAUD    set     TRUE    ;software transfer port baud control
endif ;TCS2718A

    if TC2718B
T8251    set TRUE
TB8251    equ     CBASE+2    ;Base of T-port 8251 registers

TBD110    equ    1F0h    ;0= 110 baud
TBD150    equ    1E0h    ;1= 150 baud
TBD300    equ    1D0h    ;2= 300 baud
TBD600    equ    160h    ;3= 600 baud
TBD1200   equ    1B0h    ;4= 1200 baud
TBD2400   equ    1C0h    ;5= 2400 baud
TBD4800   equ    190h    ;6= 4800 baud
TBD9600   equ    180H    ;7= 9600 baud
TBD192    equ    100h    ;8= 19200 baud
TBD384    equ    0       ;9= 38400 baud (not supported)
TBD576    equ    0       ;a= 57600 baud (not supported)
TBD768    equ    0       ;b= 76800 baud (not supported)

TSBAUD    set     TRUE    ;software transfer port baud control
BD192     set     TRUE    ;19200 baud supported
endif ;CCS2718B

;-----
;CCS 2719 Equates
;
;The CCS 2719 is built around a Z80-DART (dual UART) and
;a Z80-CTC (counter/timer, which generates baud rates).
;-----
    if CC2719A or CC2719B or TC2719A or TC2719B

DART      set     TRUE    ;DART present

0001 =    DWR1     equ    1    ;Write Register 1
0002 =    DWR2     equ    2    ;Write Register 2
0003 =    DWR3     equ    3    ;Write Register 3
0004 =    DWR4     equ    4    ;Write Register 4
0005 =    DWR5     equ    5    ;Write Register 5

0044 =    DART1S   equ    44h   ;1 stop bit
004C =    DART2S   equ    4Ch   ;2 stop bits

;DART initialization sequence, both ports
;note: INIT code assumes that DI7 is UNIQUE

```

# MEMON80.PRN

```

0028 =      DI1      equ      28h      ;channel Reset
0001 =      DI2      equ      DWR1     ;Access WR1
0000 =      DI3      equ      00h      ;Disable interrupts
0003 =      DI4      equ      DWR3     ;Access WR3
00C1 =      DI5      equ      0C1h     ;Rx 8-bit bytes, enable Rx
0005 =      DI6      equ      DWR5     ;Access WR5
00EA =      DI7      equ      0EAh     ;Tx 8-bit bytes, enable Tx,
                                ;handshakes true

        endif ;CC2719A or CC2719B or TC2719A or TC2719B

        if CC2719A
CCTC      equ      CBASE      ;CTC channel for Console baud
CDATA     equ      CBASE+4    ;channel A data
CSTAT     equ      CBASE+5    ;channel A status
CCTRL     equ      CBASE+5    ;channel A control
        endif ;CC2719A

        if CC2719B
CCTC      equ      CBASE+1    ;CTC channel for Console baud
CDATA     equ      CBASE+6    ;channel B data
CSTAT     equ      CBASE+7    ;channel B status
CCTRL     equ      CBASE+7    ;channel B control
        endif ;CC2719B

        if TC2719A
00E8 =      TCTC     equ      TBASE     ;CTC channel for TR port baud
00EC =      TDATA     equ      TBASE+4   ;channel A data
00ED =      TSTAT     equ      TBASE+5   ;channel A status
00ED =      TCTRL     equ      TBASE+5   ;channel A control
        endif ;TC2719A

        if TC2719B
TCTC      equ      TBASE+1    ;CTC channel for TR port baud
TDATA     equ      TBASE+6    ;channel B data
TSTAT     equ      TBASE+7    ;channel B status
TCTRL     equ      TBASE+7    ;channel B control
        endif ;TC2719B

        if CC2719A or CC2719B
CTXRDY     equ      04h      ;Console Tx ready bit
CRXRDY     equ      01h      ;Console Rx ready bit

;Console Port baud rate values
;high byte must be written 1st

CBD110     equ      078DH     ;0= 110 baud 2 stop bits
CBD150     equ      0768h     ;1= 150 baud
CBD300     equ      0734h     ;2= 300 baud
CBD600     equ      47C0h     ;3= 600 baud
CBD1200    equ      4760h     ;4= 1200 baud
CBD2400    equ      4730h     ;5= 2400 baud
CBD4800    equ      4718h     ;6= 4800 baud
CBD9600    equ      470CH     ;7= 9600 baud
CBD192     equ      4706h     ;8= 19200 baud
CBD384     equ      4703h     ;9= 38400 baud
CBD576     equ      4702h     ;a= 57600 baud
CBD768     equ      0         ;b= 76800 baud (not supported)

CSBAUD     set      TRUE      ;software Console
                                ;..baud rate control
CSTOP1     equ      DART1S

```

```

MEMON80.PRN
CSTOP2 equ DART2S

CSTOPS set CSTOP2
endif ;CC2719A or CC2719B

    if (CC2719A or CC2719B) and ((0-CBAUD) SHR 15)
CSTOPS set CSTOP1
    endif ;(CC2719A or CC2719B) and ((0-CBAUD) SHR 15)

    if TC2719A or TC2719B
TPORT set TRUE ;Transfer port is defined

0004 = TTXRDY equ 04h ;Transfer Port Tx ready bit
0001 = TRXRDY equ 01h ;Transfer Port Rx ready bit

;Transfer Port baud rate values
;high byte must be written 1st

078D = TBD110 equ 078DH ;0= 110 baud 2 stop bits
0768 = TBD150 equ 0768h ;1= 150 baud
0734 = TBD300 equ 0734h ;2= 300 baud
47C0 = TBD600 equ 47C0h ;3= 600 baud
4760 = TBD1200 equ 4760h ;4= 1200 baud
4730 = TBD2400 equ 4730h ;5= 2400 baud
4718 = TBD4800 equ 4718h ;6= 4800 baud
470C = TBD9600 equ 470CH ;7= 9600 baud
4706 = TBD192 equ 4706h ;8= 19200 baud
4703 = TBD384 equ 4703h ;9= 38400 baud
4702 = TBD576 equ 4702h ;a= 57600 baud
0000 = TBD768 equ 0 ;b= 76800 baud (not supported)

BD192 set TRUE ;19200 baud supported
BD384 set TRUE ;38400 baud supported
BD576 set TRUE ;57600 baud supported

00E8 = BRATE0 equ TCTC ;LOW baud byte goes here
00E8 = BRATE1 equ TCTC ;high baud byte goes here

TMRCTC set TRUE ;Z80-CTC used as timer
TSBAUD set TRUE ;software Transfer Port

0044 = TSTOP1 equ DART1S
004C = TSTOP2 equ DART2S

TSTOPS set TSTOP2
endif ;TC2719A or TC2719B

    if (TC2719A or TC2719B) and ((0-TBAUD) SHR 15)
TSTOPS set TSTOP1
    endif ;(TC2719A or TC2719B) and ((0-TBAUD) SHR 15)

;-----
;CCS 2810 Serial Port Equates
;
;The serial port on the CCS 2810 CPU board is built
;around the National Semiconductor INS8250 UART,
;which has a buad rate generator built into it.
;-----

;CCS2810 serial port as the Console

    if CC2810
C8250 set TRUE

```

00FA =

```
MEMON80.PRN
CB8250 equ CBASE
endif ;CC2810

;CCS2810 serial Port As the Transfer Port

if TC2810
T8250 set TRUE
TB8250 equ TBASE
endif ;TC2810

;-----
;Compupro Interfacer 1 Equates
;
;The Interfacer 1 is built around a pair of 1602-type
;generic UARTs, with jumper selectable baud rate
;generators and with external logic providing status
;and control registers.
;
;Control bits are configured with jumpers. the board will
;XOR whatever you write to the control port with the
;jumper setting. This assumes all jumpers are set to 0.
;-----
if CIFAC
CSTAT equ CBASE ;Console status
CCTRL equ CBASE ;Console control
CDATA equ CBASE+1 ;Console data

CTXRDY equ 01h ;channel a Tx ready bit
CRXRDY equ 02h ;channel a Rx ready bit
endif ;CIFAC

if TIFAC
TPORT set TRUE ;Transfer port is defined

TSTAT equ TBASE ;Transfer Port status
TCTRL equ TBASE ;Transfer Port control
TDATA equ TBASE+1 ;Transfer Port data

TTXRDY equ 01h ;channel B Tx ready bit
TRXRDY equ 02h ;channel B Rx ready bit
endif ;TIFAC

if CIFAC or TIFAC
IFRST equ 0ACH ;reset: 8 data, no parity, flow
;...control outputs high, no ints
endif ;CIFAC or TIFAC

;-----
;Compupro Interfacer II Equates
;The Interfacer II is built around a 1602-type generic
;UART, with jumper selectable baud rate generator and with
;external logic providing status and control registers.
;
;Control bits are configured with jumpers. The board will
;XOR whatever you write to the control port with the
;jumper setting. This assumes all jumpers are set to 0.
;-----
if CIFAC2
CSTAT equ CBASE ;Console status
CCTRL equ CBASE ;Console control
CDATA equ CBASE+1 ;Console data

CTXRDY equ 01h ;Console Tx ready bit
```

```

MEMON80.PRN
CRXRDY    equ    02h    ;Console Rx ready bit
CIFRST    equ    0ACH    ;reset: 8 data, no parity, flow
                    ;...control outputs high, no ints

endif ;CIFAC2

if TIFAC2
TPORT     set     TRUE    ;Transfer port is defined

TSTAT     equ     TBASE    ;Transfer Port status
TCTRL     equ     TBASE    ;Transfer Port control
TDATA     equ     TBASE+1 ;Transfer Port data

TTXRDY    equ     01h    ;Transfer Port Tx ready bit
TRXRDY    equ     02h    ;Transfer Port Rx ready bit

TIFRST    equ     0ACH    ;reset: 8 data, no parity,
                    ;...controls high, no ints
endif ;TIFAC2

;-----
;Cromemco TU-ART Equates
;Cromemco 4FDC/16FDC/64FDC serial port Equates
;
;The TU-ART is built around a pair of TMS5511 UARTs
;with integral software-settable baud rate generators.
;
;The serial ports on the Cromemco xFDC disk controllers
;are built around the (very similar) TMS5501 UART
;-----
if CTUART or TTUART or CCFDCS or TCFDCS
TURST     equ     01h    ;reset UART
TUHBD     equ     10h    ;high baud rate
endif ;CTUART or TTUART or CCFDCS or TCFDCS

if CTUART or CCFDCS
CSTAT     equ     CBASE    ;Console status
CDATA     equ     CBASE+1 ;Console data
TCPBAUD   equ     CBASE    ;Console baud rate
TCCTRL    equ     CBASE+2 ;Console control
TCINTE    equ     CBASE+3 ;Console interrupt enable

CRXRDY    equ     40h    ;Console Rx data buffer full
CTXRDY    equ     80h    ;Console Tx data buf available

CSTOP1    equ     80h    ;1 stop bit
CSTOP2    equ     0      ;2 stop bits

;Console Port TU-ART baud rates
; high byte is the command register value
; low byte is the baud rate register value
; bit 7 of the low byte selects 2 stop bits if 0

CBD110    equ     0001h or CSTOP2 ;110 baud
CBD150    equ     0002h or CSTOP1 ;150 baud
CBD300    equ     0004h or CSTOP1 ;300 baud
CBD600    equ     0      ;600 baud (not supported)
CBD1200   equ     0008h or CSTOP1 ;1200 baud
CBD2400   equ     0010h or CSTOP1 ;2400 baud
CBD4800   equ     0020h or CSTOP1 ;4800 baud
CBD9600   equ     0040h or CSTOP1 ;9600 baud
CBD192    equ     1010h or CSTOP1 ;19200 baud

```

```

                                MEMON80.PRN
CBD384 equ    1020h or CSTOP1 ;38400 baud
CBD576 equ    0                ;57600 baud (not supported)
CBD768 equ    1040h or CSTOP1 ;76800 baud

CSBAUD set    TRUE             ;software Console
                                ;...baud rate control
CSTOPS set    CSTOP2
endif ;CTUART or CCFDCS

    if (CTUART or CCFDCS) and ((0-CBAUD) SHR 15)
CSTOPS set    CSTOP1
endif ;(CTUART or CCFDCS) and ((0-CBAUD) SHR 15)

    if TTUART or TCFDCS
TPORT set     TRUE             ;Transfer port is defined

TSTAT equ     TBASE            ;Transfer Port status
TDATA equ     TBASE+1          ;Transfer Port data
TTPBAUD equ    TBASE            ;Transfer Port baud rate
TTCTRL equ     TBASE+2          ;Transfer Port control
TTINTE equ     TBASE+3          ;Transfer Port interrupt enable

TRXRDY equ     40h             ;Tx Port Rx data buffer full
TTXRDY equ     80h             ;Tx Port Tx data buf available

BRATE0 equ     TTPBAUD          ;LOW baud byte goes here
BRATE1 equ     TTCTRL           ;high baud byte goes here

TSTOP1 equ     80h             ;1 stop bit
TSTOP0 equ     0               ;2 stop bits

;Transfer Port TU-ART baud rates
; high byte is the command register value
; low byte is the baud rate register value

TBD110 equ     0001h or TSTOP2 ;110 baud
TBD150 equ     0002h or TSTOP1 ;150 baud
TBD300 equ     0004h or TSTOP1 ;300 baud
TBD600 equ     0                ;600 baud (not supported)
TBD1200 equ    0008h or TSTOP1 ;1200 baud
TBD2400 equ    0010h or TSTOP1 ;2400 baud
TBD4800 equ    0020h or TSTOP1 ;4800 baud
TBD9600 equ    0040h or TSTOP1 ;9600 baud
TBD19200 equ   1010h or TSTOP1 ;19200 baud
TBD38400 equ   1020h or TSTOP1 ;38400 baud
TBD57600 equ   0                ;57600 baud (not supported)
TBD76800 equ   1040h or TSTOP1 ;76800 baud

BD192 set      TRUE            ;19200 baud supported
BD384 set      TRUE            ;38400 baud supported
BD768 set      TRUE            ;76800 baud supported

TSBAUD set     TRUE            ;software T-port baud rate control
T5511 set      TRUE            ;Transfer port baud rates via TMS5511

TSTOPS set     TSTOP2
endif ;TTUART or TCFDCS

    if (TTUART or TCFDCS) and ((0-TBAUD) SHR 15)
TSTOPS set     TSTOP1
endif ;(TTUART or TCFDCS) and ((0-TBAUD) SHR 15)

```

```

MEMON80.PRN
;Electronic Control Technology (ECT) R2I/O Equates
;
;The ECT R2I/O is built around three TMS6011
;generic UARTS with jumper-selectable baud rates.
;-----
; if CER2IO
CSTAT equ CBASE ;Console status
CDATA equ CBASE+1 ;Console data

CRXRDY equ 01h ;Console Rx data buffer full
CTXRDY equ 80h ;Console Tx data buf available

CISTAT set TRUE ;Status bits are active-low
endif ;CER2IO

; if TER2IO
TPORT set TRUE ;Transfer port is defined

TSTAT equ TBASE ;Transfer Port status
TDATA equ TBASE+1 ;Transfer Port data

TRXRDY equ 01h ;Transfer port Rx data buffer full
TTXRDY equ 80h ;Transfer port Tx data buf available

TISTAT set TRUE ;Status bits are active-low
endif ;TER2IO

;-----
;Heathkit H8-4 Serial Port Equates
;
; The Heathkit H8-4 has 4 independent, seperately addressable
; serial ports, each built around a National Semiconductor
; INS8250 UART.
;
; CPU Interrupts must be enabled so that the Heathkit H8's
; front panel will work correctly. The H8-4 may or may not be
; connected to one of the interrupt lines. However, this code
; disables the H8-4's interrupts in software, and polls its
; UART(s) as with most other supported ports.
;-----

;H8-4 serial port as the Console

; if CH84
C8250 set TRUE
CB8250 equ CBASE
endif ;CH84

;H8-4 serial Port As the Transfer Port

; if TH84
T8250 set TRUE
TB8250 equ TBASE
endif ;TH84

;-----
;Heathkit H8-5 Serial Port Equates
;
; The H8-5 is built around an Intel 8251 (not the 8251A) UART,
; which has some peculiarities when operating with interrupts.
; In particular, disabling the transmitter (which is the only
; way to disable the transmit interrupt) will stop transmitting
; mid-byte, if the transmitter is busy. This makes the transmit

```

```

MEMON80.PRN
; interrupt pretty worthless. Heathkit's recommended jumpers
; only enable interrupts for receive data, not transmit. I
; suspect it is for this reason.
;
; Memon/80 expects the H8-5 to be jumpered the standard way
; (per the H8-5 manual):
;
;   Port Select          Serial Interrupt Select
;   T-3                  (Jumper group labeled "SERIAL INT. S")
;   W-7                  RXR
;   X-2                  S-/I3
;   Y-0                  INT ON
;
; Interrupts must be enabled so that the Heathkit H8's front
; panel will work correctly. The H8-5 must be connected to
; Interrupt level 3 so that other H8 software will work. There
; is no way in software to disable the h8-5's interrupt, so
; Memon/80 must use interrupts for receiving data from the
; H8-5. Memon/80 assumes that either the PAM-8 or the XCON-8
; ROM is installed in the H8 computer and maps the interrupts
; to the standard PAM-8 vector addresses.
;-----
if CH85
C8251 set TRUE ;8251 UART
CRXINT set TRUE ;Interrupt-driven receive (only)
CB8251 equ CBASE ;Base of Console 8251 registers
endif ;CH85

if TH85
TPORT set TRUE ;Transfer port is defined
T8251 set TRUE ;8251 UART
TRXINT set TRUE ;Interrupt-driven receive (only)
TB8251 equ CBASE ;Base of T-port 8251 registers
endif ;TH85

if CH85 or TH85
P8INT3 equ 2025h ;Address of PAM-8/XCON-8 Int 3 vector
endif ;CH85 or TH85

;-----
;IMSAI SIO-2 I/O Equates
;
;The SIO-2 is built around a pair of 8251A UARTs
;with a jumper-selectable baud rate generator and
;logic to provide an interrupt-control port.
;The SIO-2 can be jumpered for either memory-mapped
;I/O or port-mapped I/O. this assumes port-mapped I/O.
;-----
if CISIO2A or CISIO2B
C8251 set TRUE ;8251 UART
CB8251 equ CBASE ;Base of Console 8251 registers
endif ;CISIO2A or CISIO2B

if TISIO2A or TISIO2B
TPORT set TRUE ;Transfer port is defined
T8251 set TRUE ;8251 UART
TB8251 equ TBASE ;Base of T-port 8251 registers
endif ;TISIO2A or TISIO2B

if CISIO2A or TISIO2A
SIOCTL equ CBASE+6 ;SIO-2 int control port
endif ;CISIO2A or TISIO2A

```



# MEMON80.PRN

```

if CISIO2B and not TISIO2A
SIOCTL equ    CBASE+2 ;SIO-2 int control port
endif ;CISIO2B and not TISIO2A

if TISIO2B and not CISIO2A
SIOCTL equ    TBASE+2 ;SIO-2 int control port
endif ;TISIO2B and not CISIO2A

;-----
;Ithaca Intersystems Series II VIO Serial Port Equates
;
;The two serial ports on the Series II VIO are built
;around Signetics 2651 UARTS, which have internal baud
;rate generators.
;-----
if CIVIO2 or TIVIO2

IVSTOP1 equ    04E00h ;1 stop bit
IVSTOP2 equ    0CE00h ;2 stop bits

CTL2651 equ    27h    ;Ctrl reg initialization value
endif ;CIVIO2 or TIVIO2

if CIVIO2
CDATA equ      CBASE
CSTAT equ      CBASE+1
CMODE equ      CBASE+1
CCTRL equ      CBASE+3

CTXRDY equ     01h    ;Console transmitter ready
CRXRDY equ     02h    ;Console receiver ready

;Console Port 2651 baud rates

CBD110 equ     IVSTOP2 or 32h ;110 baud (2 stop bits)
CBD150 equ     IVSTOP1 or 34h ;150 baud
CBD300 equ     IVSTOP1 or 35h ;300 baud
CBD600 equ     IVSTOP1 or 36h ;600 baud
CBD1200 equ    IVSTOP1 or 37h ;1200 baud
CBD2400 equ    IVSTOP1 or 3Ah ;2400 baud
CBD4800 equ    IVSTOP1 or 3Ch ;4800 baud
CBD9600 equ    IVSTOP1 or 3Eh ;9600 baud
CBD192 equ     IVSTOP1 or 3Fh ;19200 baud
CBD384 equ     0       ;38400 baud (Not supported)
CBD576 equ     0       ;57600 baud (not supported)
CBD768 equ     0       ;76800 baud (Not supported)

CSBAUD set     TRUE    ;software Console baud control

endif ;CIVIO2

if TIVIO2
TPORT set      TRUE    ;Transfer port is defined

TDATA equ      TBASE
TSTAT equ      TBASE+1
TMODE equ      TBASE+1
TCTRL equ      TBASE+3

BRATE0 equ     TMODE    ;Low byte of baud rate goes here
BRATE1 equ     TMODE    ;High byte of baud rate goes here

```

```

MEMON80.PRN
TTXRDY equ 01h ;Transfer Port transmitter ready
TRXRDY equ 02h ;Transfer Port receiver ready

;Transfer Port 2651 baud rates
;(The high byte of these words gets written first)

TBD110 equ IVSTOP2 or 32h ;110 baud (2 stop bits)
TBD150 equ IVSTOP1 or 34h ;150 baud
TBD300 equ IVSTOP1 or 35h ;300 baud
TBD600 equ IVSTOP1 or 36h ;600 baud
TBD1200 equ IVSTOP1 or 37h ;1200 baud
TBD2400 equ IVSTOP1 or 3Ah ;2400 baud
TBD4800 equ IVSTOP1 or 3Ch ;4800 baud
TBD9600 equ IVSTOP1 or 3Eh ;9600 baud
TBD192 equ IVSTOP1 or 3Fh ;19200 baud
TBD384 equ 0 ;38400 baud (Not supported)
TBD576 equ 0 ;57600 baud (not supported)
TBD768 equ 0 ;76800 baud (Not supported)

BD192 set TRUE ;19200 baud supported

TSBAUD set TRUE ;software transfer port
;...baud rate control
endif ;TIVIO2

;-----
;Jade Serial/Parallel I/O Port Equates
;
;The serial port of the Jade Serial/Parallel board are
;built around the AY61013A or TR1602B "generic" UART.
;-----
if CJADSP or TJADSP
JRESET equ 10110000B ;UART reset value
endif ;CJADSP or TJADSP

if CJADSP
CDATA equ CBASE
CSTAT equ CBASE or 80h
CCTRL equ CBASE or 80h

CTXRDY equ 80h ;Console transmitter ready
CRXRDY equ 10h ;Console receiver ready
endif ;CJADSP

if TJADSP
TPORT set TRUE ;Transfer port is defined

TDATA equ TBASE
TSTAT equ TBASE or 80h
TCTRL equ TBASE or 80h

TTXRDY equ 80h ;T-port transmitter ready
TRXRDY equ 10h ;T-port receiver ready
endif ;TJADSP

;-----
;Micromation Doubler Serial Port Equates
;
;The serial port on the Doubler is built around the
;Intel 8251 UART, with jumper-selectable baud rates.
;This UART is accessed via memory accesses, rather
;than the usual I/O accesses.
;-----

```

```

MEMON80.PRN

    if CMDUBLR
C8251  set      TRUE      ;8251 UART
CMEMAP set      TRUE      ;memory-mapped I/O

CDATA  equ      CBASE+0602h ;Console data port
CSTAT  equ      CBASE+060Ah  ;Console status port
CCTRL  equ      CBASE+060Ah  ;Console control port

CTXRDY  equ      01h      ;Console transmitter ready
CRXRDY  equ      02h      ;Console receiver ready

    endif ;CMDUBLR

    if TMDUBLR
TPORT  set      TRUE      ;Transfer port is defined
T8251  set      TRUE      ;8251 UART
TMEMAP set      TRUE      ;memory-mapped I/O

TDATA  equ      TBASE+0602h ;Transfer data port
TSTAT  equ      TBASE+060Ah  ;Transfer status port
TCTRL  equ      TBASE+060Ah  ;Transfer control port

TTXRDY  equ      01h      ;Transfer transmitter ready
TRXRDY  equ      02h      ;Transfer receiver ready

    endif ;TMDUBLR

;-----
;Tarbell 4044 4P2S Serial Port Equates
;
;The Tarbell 4044 is built around four Intel 8251A
;UARTs, with switch-selectable baud rates.
;-----
    if CTARBL
C8251  set      TRUE      ;8251 UART
CB8251 equ      CBASE     ;Base of Console 8251 registers
    endif ;CTARBL

    if TTARBL
TPORT  set      TRUE      ;Transfer port is defined
T8251  set      TRUE      ;8251 UART
TB8251 equ      TBASE     ;Base of T-port 8251 registers
    endif ;TTARBL

;-----
;Salota I/O 2+1 Serial Port Equates
;
;The Salota I/O 2+1's serial ports are built around two
;Intel 8251A UARTs, with jumper-selectable baud rates.
;-----
    if CSAL21
C8251  set      TRUE      ;8251 UART
CB8251 equ      CBASE     ;Base of Console 8251 registers
    endif ;CSAL21

    if TSAL21
TPORT  set      TRUE      ;Transfer port is defined
T8251  set      TRUE      ;8251 UART
TB8251 equ      TBASE     ;Base of T-port 8251 registers
    endif ;TSAL21

```

```

MEMON80.PRN
;-----
;Solid State Music IO5 Serial Port Equates
;
;The IO5's serial ports are built around two Intel
;8251A UARTs, with switch-selectable baud rates.
;-----
if CSIO5
C8251 set TRUE ;8251 UART
CB8251 equ CBASE ;Base of Console 8251 registers
endif ;CSIO5

if TSIO5
TPORT set TRUE ;Transfer port is defined

T8251 set TRUE ;8251 UART
TB8251 equ TBASE ;Base of T-port 8251 registers
endif ;TSIO5

;-----
;Processor Technology 3P+S Serial Port Equates
;
;The 3P+S is built around a TMS6011 or AMI S1883
;UART, with a jumper selectable baud rate generator.
;This board has many jumpers on it, allowing you to
;set up the bits within the control port any way you
;like. Memon/80 assumes it is set up the way that
;Processor Technology'S CUTER expects it to be.
;-----
if CPT3PS
CSTAT equ CBASE ;Console status
CCTRL equ CBASE ;Console control
CDATA equ CBASE+1 ;Console data

CTXRDY equ 80h ;transmitter buffer empty
CRXRDY equ 40h ;receive data available

endif ;CPT3PS

if TPT3PS
TPORT set TRUE ;Transfer port is defined

TSTAT equ TBASE ;Transfer Port status
TCTRL equ TBASE ;Transfer Port control
TDATA equ TBASE+1 ;Transfer Port data

TTXRDY equ 80h ;transmitter buffer empty
TRXRDY equ 40h ;receive data available
endif ;TPT3PS

;-----
;Vector Graphic BitStreamer Equates
;
;The BitStreamer is built around an 8251 UART,
;with a jumper selectable baud rate generator.
;-----
if CBITS1
C8251 set TRUE ;8251 UART
CB8251 equ CBASE ;Base of Console 8251 registers
endif ;CBITS1

if TBITS1
TPORT set TRUE ;Transfer port is defined

```

```

MEMON80.PRN
T8251 set TRUE ;8251 UART
TB8251 equ TBASE ;Base of T-port 8251 registers
endif ;TBITS1

;-----
;Vector Graphic BitStreamer II Equates
;
;The BitStreamer II is built around three 8251A
;UARTs with a jumper-selectable baud rate generator.
;-----
if CBITS2
C8251 set TRUE ;8251 UART
CB8251 equ CBASE ;Base of Console 8251 registers
endif ;CBITS2

if TBITS2
TPORT set TRUE ;Transfer port is defined

T8251 set TRUE ;8251 UART
TB8251 equ TBASE ;Base of T-port 8251 registers
endif ;TBITS2

;-----
;Wameco IOB-1 Equates
;
;The serial port of the Wameco IOB-1 is built around
;the Intel 8251 UART with a switch-selectable baud rate.
;-----
if CWIOB1
C8251 set TRUE ;8251 UART
CB8251 equ CBASE ;Base of Console 8251 registers
endif ;CWIOB1

if TWIOB1
TPORT set TRUE ;Transfer port is defined

T8251 set TRUE ;8251 UART
TB8251 equ TBASE ;Base of T-port 8251 registers
endif ;TWIOB1

;-----
;General MC6850 ACIA Equates
;-----
if C6850 or T6850
;MOTOROLA 6850 ACIA control/status values
U68RES equ 00000011B ;master reset
U68ST2 equ 00010001B ;2 stop bits, /16
U68ST1 equ 00010101B ;1 stop bit, /16
endif ;C6850 or T6850

; MC6850 as the Console Port

if C6850
CCTRL equ C6850 ;ACIA 0 control output port
CSTAT equ C6850 ;ACIA 0 status input port
CDATA equ C6850+1 ;ACIA 0 Rx & Tx data regs

CRXRDY equ 00000001B ;Console Rx data reg full
CTXRDY equ 00000010B ;Console Tx data reg empty
endif ;C6850

;MC6850 as the Transfer Port

```

```

MEMON80.PRN

if T6850
TCTRL equ TB6850 ;ACIA 1 control output port
TSTAT equ TB6850 ;ACIA 1 status input port
TDATA equ TB6850+1 ;ACIA 1 Rx & Tx data regs

TRXRDY equ 00000001B ;Transfer Port Rx data reg full
TTXRDY equ 00000010B ;Tr. Port Tx data reg empty
endif ;T6850

;-----
;General INS8250 UART Equates
;-----

;INS8250 UART as the Console Port

if C8250

0001 = CRXRDY equ 00000001b ;Rx data ready
0020 = CTXRDY equ 00100000b ;Tx buffer empty
0080 = CDLAB equ 80h ;Baud rate access

CSBAUD set TRUE ;software Console
;...baud rate control
;Console Port baud rates

0417 = CBD110 equ 1047 ;0= 110 baud (2 stop bits)
0300 = CBD150 equ 768 ;1= 150 baud
0180 = CBD300 equ 384 ;2= 300 baud
00C0 = CBD600 equ 192 ;3= 600 baud
0060 = CBD1200 equ 96 ;4= 1200 baud
0030 = CBD2400 equ 48 ;5= 2400 baud
0018 = CBD4800 equ 24 ;6= 4800 baud
000C = CBD9600 equ 12 ;7= 9600 baud
0006 = CBD192 equ 6 ;8= 19200 baud
0003 = CBD384 equ 3 ;9= 38400 baud
0000 = CBD576 equ 0 ;a= 57600 baud (not supported)
0000 = CBD768 equ 0 ;b= 76800 baud (not supported)

0003 = CSTOP1 equ 03h ;1 stop bit
0007 = CSTOP2 equ 07h ;2 stop bits

00FA = CDATA equ CB8250 ;data port
00FB = CSINTEN equ CDATA+1 ;interrupt enable port
00FC = CSIDENT equ CDATA+2 ;interrupt ID port
00FD = CSLCTRL equ CDATA+3 ;line control port
00FE = CSMDMCT equ CDATA+4 ;modem control port
00FF = CSTAT equ CDATA+5 ;line status port
0100 = CMSTAT equ CDATA+6 ;modem status port

CSTOPS set CSTOP2
endif ;C8250

if C8250 and ((0-CBAUD) SHR 15)
CSTOPS set CSTOP1
endif ;CC8250 and ((0-CBAUD) SHR 15)

;INS8250 UART as the Transfer Port

if T8250

TRXRDY equ 00000001b ;Rx data ready
TTXRDY equ 00100000b ;Tx buffer empty
TDLAB equ 80h ;Baud rate access

```

# MEMON80.PRN

```

TPORT    set        TRUE    ;Transfer port is defined
TSBAUD   set        TRUE    ;software Transfer Port
                               ;..baud rate control

;Transfer Port baud rates

TBD110   equ        1047    ;0= 110 baud (2 stop bits)
TBD150   equ        768     ;1= 150 baud
TBD300   equ        384     ;2= 300 baud
TBD600   equ        192     ;3= 600 baud
TBD1200  equ        96      ;4= 1200 baud
TBD2400  equ        48      ;5= 2400 baud
TBD4800  equ        24      ;6= 4800 baud
TBD9600  equ        12      ;7= 9600 baud
TBD192   equ        6       ;8= 19200 baud
TBD384   equ        3       ;9= 38400 baud
TBD576   equ        0       ;a= 57600 baud (not supported)
TBD768   equ        0       ;b= 76800 baud (not supported)

BD192    set        TRUE    ;19200 baud supported
BD384    set        TRUE    ;38400 baud supported
TSTOP1   equ        03h     ;1 stop bit
TSTOP2   equ        07h     ;2 stop bits

TDATA    equ        TB8250  ;data port
TSINTEN  equ        TDATA+1 ;interrupt enable port
TSIDENT  equ        TDATA+2 ;interrupt ID port
TSLCTRL  equ        TDATA+3 ;line control port
TSMDMCT  equ        TDATA+4 ;modem control port
TSTAT    equ        TDATA+5 ;line status port
TMSTAT   equ        TDATA+6 ;modem status port

BRATE0   equ        TDATA    ;Baud rate divisor low byte
BRATE1   equ        TSINTEN  ;Baud rate divisor high byte

TSTOPS   set        TSTOP2
endif ;T8250

if T8250 and ((0-TBAUD) SHR 15)
TSTOPS   set        TSTOP1
endif ;T8250 and ((0-TBAUD) SHR 15)

;-----
;General 8251/8251A UART Equates
;-----

;Port Relative Addresses and Ready Bits for I/O-mapped 8251s

if C8251 and (not CMEMAP)
CDATA    equ        CB8251    ;Console data
CSTAT    equ        CB8251+1  ;Console status
CCTRL    equ        CB8251+1  ;Console control

CTXRDY   equ        01h      ;Console transmitter ready
CRXRDY   equ        02h      ;receiver ready
CTXMTY   equ        04h      ;Console Transmitter empty
endif ;C8251 and (not CMEMAP)

if T8251 and (not TMAP)
TDATA    equ        TB8251    ;Transfer port data
TSTAT    equ        TB8251+1  ;Transfer port status

```

```

MEMON80.PRN
TCTRL    equ      TB8251+1      ;Transfer port control

TTXRDY    equ      01h          ;Transfer port transmitter ready
TRXRDY    equ      02h          ;Transfer port receiver ready
TTXMTY    equ      04h          ;Transfer port Transmitter empty
endif ;T8251 and (not TMAP)

;initialization values for all 8251s

    if C8251 or T8251
UMOD16    equ      4Eh          ;8 bits, 1 stop, no parity, X16
UCMDRE    equ      40h          ;reset UART
UCMDEN    equ      37h          ;enable Tx & Rx, DTR, RTS ON
UCMDEI    equ      36h          ;enable Rx, disable Tx

;8251 reset sequence
;note: code assumes UCMDEN is UNIQUE here

SI21      equ      0AAh        ;Fake SYNC chr
SI22      equ      UCMDRE      ;reset command
SI23      equ      UMOD16      ;Clock divisor, etc.
SI24      equ      UCMDEN      ;enable Rx & Tx
endif ;C8251 or T8251

;-----
;Compute Baud Rate Constants
;-----
    if CSBAUD
CPBAUD    set      CBD110
endif ;CSBAUD

    if CSBAUD and ((0-CBAUD)SHR 15)
CPBAUD    set      CBD150
endif ;CSBAUD and ((0-CBAUD)SHR 15)

    if CSBAUD and ((1-CBAUD)SHR 15)
CPBAUD    set      CBD300
endif ;CSBAUD and ((1-CBAUD)SHR 15)

    if CSBAUD and ((2-CBAUD)SHR 15)
CPBAUD    set      CBD600
endif ;CSBAUD and ((2-CBAUD)SHR 15)

    if CSBAUD and ((3-CBAUD)SHR 15)
CPBAUD    set      CBD1200
endif ;CSBAUD and ((3-CBAUD)SHR 15)

    if CSBAUD and ((4-CBAUD)SHR 15)
CPBAUD    set      CBD2400
endif ;CSBAUD and ((4-CBAUD)SHR 15)

    if CSBAUD and ((5-CBAUD)SHR 15)
CPBAUD    set      CBD4800
endif ;CSBAUD and ((5-CBAUD)SHR 15)

    if CSBAUD and ((6-CBAUD)SHR 15)
CPBAUD    set      CBD9600
endif ;CSBAUD and ((6-CBAUD)SHR 15)

    if CSBAUD and ((7-CBAUD)SHR 15)
CPBAUD    set      CBD192
endif ;CSBAUD and ((7-CBAUD)SHR 15)

```



```

MEMON80.PRN
if CSBAUD and ((8-CBAUD)SHR 15)
CPBAUD set CBD384
endif ;CSBAUD and ((8-CBAUD)SHR 15)

if CSBAUD and ((9-CBAUD)SHR 15)
CPBAUD set CBD576
endif ;CSBAUD and ((9-CBAUD)SHR 15)

if CSBAUD and ((0Ah-CBAUD)SHR 15)
CPBAUD set CBD768
endif ;CSBAUD and ((0Ah-CBAUD)SHR 15)

if TSBAUD
TPBAUD set TBD110
endif ;TSBAUD

if TSBAUD and ((0-TBAUD)SHR 15)
TPBAUD set TBD150
endif ;TSBAUD and ((0-TBAUD)SHR 15)

if TSBAUD and ((1-TBAUD)SHR 15)
TPBAUD set TBD300
endif ;TSBAUD and ((1-TBAUD)SHR 15)

if TSBAUD and ((2-TBAUD)SHR 15)
TPBAUD set TBD600
endif ;TSBAUD and ((2-TBAUD)SHR 15)

if TSBAUD and ((3-TBAUD)SHR 15)
TPBAUD set TBD1200
endif ;TSBAUD and ((3-TBAUD)SHR 15)

if TSBAUD and ((4-TBAUD)SHR 15)
TPBAUD set TBD2400
endif ;TSBAUD and ((4-TBAUD)SHR 15)

if TSBAUD and ((5-TBAUD)SHR 15)
TPBAUD set TBD4800
endif ;TSBAUD and ((5-TBAUD)SHR 15)

if TSBAUD and ((6-TBAUD)SHR 15)
TPBAUD set TBD9600
endif ;TSBAUD and ((6-TBAUD)SHR 15)

if TSBAUD and ((7-TBAUD)SHR 15)
TPBAUD set TBD192
endif ;TSBAUD and ((7-TBAUD)SHR 15)

if TSBAUD and ((8-TBAUD)SHR 15)
TPBAUD set TBD384
endif ;TSBAUD and ((8-TBAUD)SHR 15)

if TSBAUD and ((9-TBAUD)SHR 15)
TPBAUD set TBD576
endif ;TSBAUD and ((9-TBAUD)SHR 15)

if TSBAUD and ((0Ah-TBAUD)SHR 15)
TPBAUD set TBD768
endif ;TSBAUD and ((0Ah-TBAUD)SHR 15)

;*****
;Compute Memon80's RAM beginning for the banner message
;*****

```

# MEMON80.PRN

008F =

```

if not (TRXINT or CRXINT)
    RAMBEG equ    STACK-STKSIZ-1
endif ;not (TRXINT or CRXINT)

if TRXINT or CRXINT
    RAMBEG equ    TRXBUF
endif ;TRXINT or CRXINT

;*****
;Disk Controller Definitions
;*****
;Initialize some assembler variables
BOOTER set      FALSE           ;Assume no BO command for now
JMPBOT set      FALSE
WD179X set      FALSE

;-----
;Altair 8800 Floppy Disk controller Equates (These are the
;same for the 88-DCDD controller and the 88-MDS controller.)
;-----
if A88DCD or A88MCD
    BOOTER set    TRUE           ;Boot command exists

    DENABL equ    DIBASE         ;Drive enable output
    DDISBL equ    80h           ;disable disk controller

    DSTAT equ    DIBASE         ;status input (active low)
    ENWDAT equ    01h           ;-enter write data
    MVHEAD equ    02h           ;-Move Head OK
    HDSTAT equ    04h           ;-Head status
    DRVRDY equ    08h           ;-Drive Ready
    INTSTA equ    20h           ;-interrupts enabled
    TRACK0 equ    40h           ;-Track 0 detected
    NRDA equ    80h           ;-new Read data Available

    DCTRL equ    DIBASE+1       ;Drive control output
    STEPIN equ    01h           ;Step-In
    STEPOT equ    02h           ;Step-Out
    HEDLOD equ    04h           ;8" disk: load head
                                ;Minidisk: restart
                                ;..6.4 S timer
    HDUNLD equ    08h           ;unload head (8" only)
    IENABL equ    10h           ;enable sector interrupt
    IDSABL equ    20h           ;Disable interrupts
    WENABL equ    80h           ;enable drive write circuits

    DSECTR equ    DIBASE+1       ;Sector position input
    SVALID equ    01h           ;Sector valid (1st 30 us
                                ;..of sector pulse)
    SECMSK equ    3Eh           ;Sector mask for MDSEC

    DDATA equ    DIBASE+2       ;Disk data (input/output)

;Floppy Disk Parameters

    BPS equ    128             ;data bytes/sector
    MDSPT equ    16            ;Minidisk sectors/track
                                ;this code assumes SPT for 8"
                                ;disks=MDSPT * 2.

    HDRSIZ equ    3             ;header bytes before data

```

```

MEMON80.PRN
TLRSIZ equ 2 ;trailer bytes read after data
SECSIZ equ BPS+HDRSIZ+TLRSIZ ;total bytes/sector
MAXTRYS equ 16 ;max retries per sector

;sector buffer component offsets
SFSIZE equ RAMBUF+1 ;file size
SDATA equ RAMBUF+HDRSIZ ;sector data
SMARKR equ SDATA+BPS ;marker byte
SCKSUM equ SMARKR+1 ;checksum byte
endif ;A88DCD or A88MCD

;-----
;CCS 2422 Disk controller
;-----
if CC2422

BOOTER set TRUE ;Boot command exists
WD179X set TRUE ;WD1793-type controller

0010 = MAXTRYS equ 16 ;max retries
0080 = SECSIZ equ 128 ;bytes/sector
0000 = BOTDSK equ 0 ;Boot disk
0001 = BOTSEC equ 1 ;Boot sector
0080 = BOTADR equ 80h ;Load & Run boot sector here
;Code assumes xx80h

;CCS 2422 floppy disk controller addresses

0004 = BCTRL equ 4 ;Control port 2
0004 = BSTAT equ 4 ;status port 2
0030 = DSTAT equ 30h ;disk status port
0030 = DCMMMD equ DSTAT ;disk command port
0031 = DTRCK equ DSTAT+1 ;track port
0032 = DSCTR equ DSTAT+2 ;sector port
0033 = DDATA equ DSTAT+3 ;data port
0034 = DFLAG equ DSTAT+4 ;disk flag port
0034 = DCTRL equ DSTAT+4 ;disk control port

;Control register 1 (DCTRL) output bits

0001 = DCDS1 equ 00000001b ;drive select 1
0002 = DCDS2 equ 00000010b ;drive select 2
0004 = DCDS3 equ 00000100b ;drive select 3
0008 = DCDS4 equ 00001000b ;drive select 4
0010 = DCMAXI equ 00010000b ;0=Minidisk, 1=8" disk
0020 = DCMDMO equ 00100000b ;Minidisk motor on
0040 = DCDDEN equ 01000000b ;double density enable
0080 = DCAUTOW equ 10000000b ;autowait

;Control register 2 (BCTRL) output bits

0004 = BCEJECT equ 00000100b ;remote eject control (PerSci)
0010 = BCFSEEK equ 00010000b ;fast seek: voicecoil drives
0040 = BCSIDE0 equ 01000000b ;0=side 1, 1=side 0
0080 = BCROMEN equ 10000000b ;enable onboard ROM

;Composite BCTRL values used to select disk side
; Note: ROM is actually disabled via the page register
; Note: fast seek disabled via hardware jumper

00D0 = SIDE0 equ BCROMEN+BCFSEEK+BCSIDE0

```

```

MEMON80.PRN
0090 =      SIDE1      equ      BCROMEN+BCFSEEK

;Status register 1 (DFLAG) input bits

0001 =      DFINTRQ  equ      00000001b      ;WD1793 INTRQ signal,
;active high
0002 =      DFDS1    equ      00000010b      ;DS1 in control reg 1
0004 =      DFDS2    equ      00000100b      ;DS2 in control reg 1
0008 =      DFDS3    equ      00001000b      ;DS3 in control reg 1
0010 =      DFDS4    equ      00010000b      ;DS4 in control reg 1
0020 =      DFSHLD   equ      00100000b      ;WD1793 HLD signal, active high
0040 =      DFAUTOB   equ      01000000b      ;autoboot jumper, 0=autoboot
0080 =      DFSDRQ   equ      10000000b      ;WD1793 DRQ signal, 1=ready

;Status reg 2 (BSTAT) input bits

0001 =      BTRK0    equ      00000001b      ;when on track 0: 1=8" disk
0002 =      BMINI    equ      00000010b      ;-DCTRL reg bit: 1=Minidisk
0004 =      BWPRT    equ      00000100b      ;0=disk is write-protected
0008 =      BSIDE0   equ      00001000b      ;SIDE0 bit in control reg 2
0010 =      BINDEX   equ      00010000b      ;index signal, active low
0020 =      BDDEN    equ      00100000b      ;DDEN signal in ctrl register 1
0040 =      BSIDED   equ      01000000b      ;0=selected disk is 2-sided

      endif ;CC2422

;-----
;Cromemco 4FDC/16FDC/64FDC Equates
;-----
      if C4FDC or C16FDC or C64FDC

      BOOTER      set      TRUE              ;Boot command exists
      WD179X      set      TRUE              ;WD1793-type controller

      MAXDRV      equ      3                  ;drives 0-3 allowed
      BOTADR      equ      0080h              ;boot and run address
; (code assumes xx80h)
      BOTSEC      equ      1                  ;boot sector
      MAXTRY      equ      10                 ;boot retries

;16FDC Ports & Bits

      ASTAT      equ      04h                 ;Aux stat input
      ACTRL      equ      04h                 ;Auxiliary cmd output
      DSTAT      equ      30h                 ;Status input
      DCMMMD      equ      30h                 ;cmd output
      DTRCK      equ      31h                 ;Track I/O
      DSCTR      equ      32h                 ;Sector I/O
      DDATA      equ      33h                 ;Data I/O
      DFLAG      equ      34h                 ;Flags input
      DCTRL      equ      34h                 ;Control output

;DCTRL output bits

      DCDS0      equ      00000001B ;Drive Select 0
      DCDS1      equ      00000010B ;Drive Select 1
      DCDS2      equ      00000100B ;Drive Select 2
      DCDS3      equ      00001000B ;Drive Select 3
      DCSMSK     equ      11110000B ;Drive Select mask
      DCMAXI     equ      00010000B ;0=Mini, 1=Maxi
      DCMOTO     equ      00100000B ;Motor On
      DCMMSK     equ      11011111B ;Motor mask
      DCDDEN     equ      01000000B ;Enable Double Density

```

```

                                MEMON80.PRN
DCAUTO equ      10000000B ;Enable auto-wait

;ACTRL output bits (These are active-low)

ACSID0 equ      00000010B ;-Side select
ACCTRL equ      00000100B ;-Control Out
ACRSTR equ      00001000B ;-Restore
ACFSEK equ      00010000B ;-Fast Seek
ACDSO equ       00100000B ;-Drive Select Override
ACEJCT equ      01000000B ;-EJECT

;DFLAG input bits

DFSE0J equ      00000001B ;Disk cmd complete
DFAWTO equ      00000010B ;Auto-wait timeout
DFSMT0 equ      00000100B ;Motor timeout
DFSMON equ      00001000B ;1793 is requesting motors on
DSFhLD equ      00100000B ;1793 is requesting head load
DSFDRQ equ      10000000B ;1793 is requesting data

;ASTAT bits

DASSIP equ      01000000B ;Seek in progress
DASDRQ equ      10000000B ;Data Request

endif ;C4FDC or C16FDC or C64FDC

;-----
;IMSAI FIF Disk Controller
;-----
if IFIF

BOOTER set      TRUE                ;Boot command exists
MAXTRYS equ     16                  ;max retries

;Disk Command Port

IDISK equ       DIBASE

;Command Port Commands

IEXEC0 equ      0                   ;Execute string command 0
ISSPTR equ      10h                 ;Set string pointer 0
IRESTR equ      21h                 ;Reestore to track 0

;RAM string commands

IRDSEC equ      21h                 ;Read sector

;RAM location for command strings

IBCMD equ       80h                 ;Beginning of disc command string
                                      ;..and command byte therEOF
IBSTAT equ      IBCMD+1             ;Status byte
IBTRK equ       IBCMD+2             ;Track (2 bytes)
IBSECT equ      IBCMD+4             ;Sector
IBUFAD equ      IBCMD+5             ;Buffer Address

;Disk status

ISUCCS equ      01h                 ;Success

```

```
endif ;IFIF
```

```
-----
;Micropolis FD controller B Disk Controller
;-----
if MICROP
```

```
BOOTER set TRUE ;Boot command exists
```

```
FDCMD equ DMBASE+200h ;Command register
FDSECT equ DMBASE+200h ;Sector register
FDSTAT equ DMBASE+201h ;Status register
FDDATA equ DMBASE+203h ;Data in/out
```

```
SCLEN equ 268 ;sector len without sync byte and checksum
PTROFF equ 010Ah ;offset of load address
;..pointer within sector 0
EXEOFF equ 0Ch ;offset into sector data for execution
MAXTRYS equ 16 ;max retries
```

```
;Sector register is at FDCBASE+0
```

```
SMASK equ 00Fh ;Sector mask
SFLG equ 080h ;sector start flag
SIFLG equ 040h ;sector interrupt flag
DTMR equ 020h ;2mhz/4mhz jumper for timers
```

```
;Status register is at FDCBASE+1
```

```
TFLG equ 80h ;transfer flag
INTE equ 40h ;S-100 INTE signal
RDY equ 20h ;disk ready
WPT equ 10h ;write protect
TK0 equ 08h ;track zero
USLT equ 04h ;unit selected
```

```
;Command register at FDCBASE+0 (and again at FDCBASE+1)
```

```
; bits 1-0 command modifier
; bits 7-5 command
```

```
SLUN equ 020h ;select unit
;Modifier is unit address
```

```
SINT equ 040h ;set interrupt
;Modifier=1 enable int
; 0 disable int
```

```
STEP equ 060h ;step carriage
STEPO equ 000h ;step out modifier
STEPI equ 001h ;step in modifier
```

```
WTCMD equ 080h ;enable write
;No modifier used
```

```
FRESET equ 0A0h ;reset FDC
;No modifier used
```

```
;RAM locations that must be initialized for DOS
```

```
ROMADR equ 00A0h ;onboard ROM base address
FDCADR equ 00A2h ;Save FDC address here for DOS
LODADR equ 00A4h ;load address of sector
```

```

MEMON80.PRN
RDSEC    equ    00A6h    ;Read Sector subroutine
                        ;...(once its moved)

RDSECH   equ    RDSEC+3  ;enter here with hl=FDC address
                        ;(entry point used during boot)

endif ;MICROP

;-----
;Northstar MDC-A and MDS-A Single-Density Disk Controller
;-----
if NSTARS

BOOTER   set     TRUE           ;Boot command exists

;The Northstar MDC-A and MDS-A disk controllers work by reading
;and writing to memory addresses - no INs or outs occur.
;Commands are encoded into the low byte of the address.

;Disk controller memory space allocation

ROMADR   equ     DMBASE+100h    ;Standard ROM address
CTLADR   equ     DMBASE+300h    ;Disk Control base address

;controller commands (are at memory addresses)

CTLDS0   equ     CTLADR+00h    ;Select drive 0 (illegal)
CTLDS1   equ     CTLADR+01h    ;Select drive 1
CTLDS2   equ     CTLADR+02h    ;Select drive 2
CTLDS3   equ     CTLADR+03h    ;Select drive 3
CTLWRT   equ     CTLADR+04h    ;Initiate write
CTLSTC   equ     CTLADR+08h    ;Clear track-step f-f
CTLSTS   equ     CTLADR+09h    ;Set track-step f-f
CTLDI    equ     CTLADR+0Ch    ;Disarm interrupts
CTLEI    equ     CTLADR+0Dh    ;Arm interrupts
CTLNOP   equ     CTLADR+10h    ;No operation
CTLRSF   equ     CTLADR+14h    ;Reset sector flag
CTLRES   equ     CTLADR+18h    ;Reset controller
CTLSTO   equ     CTLADR+1Ch    ;select step out
CTLSTI   equ     CTLADR+1Dh    ;Select step in
CTLBST   equ     CTLADR+20h    ;Read B status
CTLRD    equ     CTLADR+40h    ;Read Data
CTLMO    equ     CTLADR+80h    ;Drive motors on command

;A-Status Bits

SATRO    equ     01h          ;Track 0
SAWP     equ     02h          ;Write protect
SABDY    equ     04h          ;Sync chr found
SAWRT    equ     08h          ;Ready for write data
SAMO     equ     10h          ;Motor is on
SAWN     equ     40h          ;Status read during window
SASF     equ     80h          ;Sector flag

;B-Status Bits

SBSECT   equ     0Fh          ;Sector position mask
SBMO     equ     10h          ;Motor is on
SBWN     equ     40h          ;Status read during window
SBSF     equ     80h          ;Sector flag

;Constants

```

```

MEMON80.PRN
UNINIT equ 59h ;uninitialized RAM value
MAXTRYS equ 10 ;Max retries
BDRIVE equ 1 ;Boot drive
;...(Code assumes this is 01.)
BTRACK equ 0 ;Boot track
BSECTR equ 4 ;Boot sector
LODADR equ 2000h ;Standard MDS-A load address
EXEADR equ 2000h ;..and execution address

```

```
endif ;NSTARS
```

```

;-----
;Northstar MDS-D Double-Density Disk Controller
;-----
if NSTARD

```

```
BOOTER set TRUE ;Boot command exists
```

```

;The Northstar MDS-D disk controller works by reading and
;writing to memory addresses - no INs or outs occur.
;Commands are encoded into the low byte of the address.

```

```

;-----
;MDS-D Disk Controller Equates
;-----

```

```

CTLWD equ DMBASE+100h ;write data
CTLORD equ DMBASE+200h ;controller Orders
CTLCMD equ DMBASE+300h ;controller Commands

```

```
;Order Register Bits
```

```

ORDDS0 equ 00 ;No drive selected
ORDDS1 equ 01h ;Select drive 1
ORDDS2 equ 02h ;Select drive 2
ORDDS3 equ 04h ;Select drive 3
ORDDS4 equ 08h ;Select drive 4
ORDST equ 10h ;Step level
ORDSIN equ 20h ;Step in
ORDSS equ 40h ;Disk side
ORDDD equ 80h ;Double-density

```

```
;Control register bits
```

```

CTLRSF equ 01h ;reset sector flag
CTLDI equ 02h ;Disarm interrupt
CTLEI equ 03h ;Arm interrupt
CTLsb equ 04h ;Set Body (diag)
CTLMO equ 05h ;Motors on
CTLWR equ 06h ;Begin write
CTLRES equ 07h ;reset controller

CTLAS equ 10h ;Select A-Status
CTLBS equ 20h ;Select B-Status
CTLCS equ 30h ;Select C-Status
CTLRD equ 40h ;Select read data

```

```
;A-Status Bits
```

```

SABD equ 01h ;sync chr detected (body)
SARE equ 04h ;Read enabled
SAWI equ 08h ;Sector window
SAMO equ 10h ;Motor on
SADD equ 20h ;Double density detected

```



```

MEMON80.PRN
SAIX    equ    40h    ;Index hole on previous sector
SASF    equ    80h    ;Sector flag
                        ;...(cleared by software)

;B-Status Bits

SBT0    equ    01h    ;Track 0 detected
SBWP    equ    02h    ;Write protected
SBWR    equ    08h    ;Write operation in progress
SBMO    equ    10h    ;Motor on
SBDD    equ    20h    ;Double density detected
SBIX    equ    40h    ;Index hole on previous sector
SBSF    equ    80h    ;Sector flag
                        ;...(cleared by software)

;C-Status Bits

SCSM    equ    0Fh    ;Sector mask
SCMO    equ    10h    ;Motor on
SCDD    equ    20h    ;Double density detected
SCIX    equ    40h    ;Index hole on previous sector
SCSF    equ    80h    ;Sector flag
                        ;...(cleared by software)

;-----
; Constants
;-----
MAXTRYS equ    10    ;max retries
DDBSCR   equ    4    ;Double-density boot sector
SDBSCR   equ    8    ;Single-density boot sector

endif ;NSTARD

;-----
;SD Systems Versafloppy and Versafloppy II
;Single- or Double-Density Disk Controller
;for minidisks or 8" disks
;-----
if VERSA1 or VERSA2 or SALFDC

BOOTER   set    TRUE    ;Boot command exists
WD179X   set    TRUE    ;WD179x (or WD177x)

SECSIZ   equ    128    ;boot sector size
VLOAD    equ    0800h   ;Boot code Load address
VBOOT    equ    0080H   ;Boot code exec address
MAXTRYS   equ    16    ;Max retries

;Port addresses

VDRSEL    equ    DIBASE+3    ;Drive select port
VDCOM     equ    DIBASE+4    ;WD1793 Command port
VDSTAT    equ    DIBASE+4    ;WD1793 Status port
VTRACK    equ    DIBASE+5    ;WD1793 Track port
VSECT     equ    DIBASE+6    ;WD1793 Sector Register
VDDATA    equ    DIBASE+7    ;WD1793 Data Register

endif ;VERSA1 or VERSA2 or SALFDC

if VERSA1
;Versafloppy VDRSEL bit assignments (all active-low)

VDSEL0N   equ    00000001b    ;select drive 0

```

```

                                MEMON80.PRN
VDSSEL1N equ    00000010b      ;select drive 1
VDSSEL2N equ    00000100b      ;select drive 2
VDSSEL3N equ    00001000b      ;select drive 3
VDSIDE1N equ    00010000b      ;Select side 1
VRSTORN equ     00100000b      ;optional Restore
VWAITN equ      01000000b      ;Enable auto-wait circuit
VINTEN equ      10000000b      ;Interrupt Enable
endif ;VERSA1

    if VERSA2 or SALFDC
;Versafloppy II VDRSEL bit assignments (all active-high)

VDSSEL0 equ     00000001b      ;select drive 0
VDSSEL1 equ     00000010b      ;select drive 1
VDSSEL2 equ     00000100b      ;select drive 2
VDSSEL3 equ     00001000b      ;select drive 3
VDSIDE1 equ     00010000b      ;Select side 1
VMINI equ       00100000b      ;Set up for minidisk
VDDEN equ       01000000b      ;Enable double-density
VWAIT equ       10000000b      ;Enable auto-wait circuit
endif ;VERSA2 or SALFDC

;-----
;Tarbell Single- or Double-Density Disk Controller
;-----
    if TARBL1 or TARBL2

BOOTER set      TRUE           ;Boot command exists
WD179X set      TRUE           ;WD179x (or WD177x)

SBOOT equ       007DH          ;Boot code exec address
SLOAD equ       0              ;Code load address
MAXTRYs equ     16             ;Max retries

TDCOM equ       DIBASE         ;WD1793 Command port
TDSTAT equ      DIBASE         ;WD1793 Status port
TTRACK equ      DIBASE+1       ;WD1793 Track port
TSECT equ       DIBASE+2       ;WD1793 Sector Register
TDDATA equ      DIBASE+3       ;WD1793 Data Register
TWAIT equ       DIBASE+4       ;Wait Input Register
endif ;TARBL1 or TARBL2

    if TARBL1
TEXTP equ       DIBASE+4h      ;Extended Output Port
;Extended Output Port bits

TPADE32 equ     00000000b      ;Write to option pad E-32
TOUTSO equ     00000001b      ;Write to SOn line
TDSEL equ       00000010b      ;Write to Drive Select latch
TDSEL1N equ     00010000b      ;Drive Select 1, active low
TDSEL2N equ     00100000b      ;Drive Select 2, active low
TDSEL3N equ     01000000b      ;Drive Select 3, active low
TDSEL4N equ     10000000b      ;Drive Select 4, active low

TSEL0 equ       TDSEL + (TDSEL1N xor 0F0h) ;select drive 0
endif ;TARBL1

    if TARBL2
TEXTP0 equ      DIBASE+4h      ;Extended Output Port 0
TEXTP1 equ      DIBASE+5h      ;Ext. Out Port 1 (bank select)
;Extended Output Port 0 bits

```

# MEMON80.PRN

```

TDDEN    equ    00001000b    ;Double density enable
TDSSEL   equ    00110000b    ;Drive select mask
TSIDE1    equ    01000000b    ;side 1 select
endif ;TARBL2

```

```

;-----
;Common Western digital WD179x Constants
;(Also correct for WD177x-type controllers.)
;-----

```

```

if WD179X

```

```

;WD1793 Commands

```

```

0000 =    RSTRCM    equ    00000000B ;(I)Restore
0010 =    SEEKCM    equ    00010000B ;(I)Seek
0020 =    STEPCM    equ    00100000B ;(I)Step
0040 =    STPICM    equ    01000000B ;(I)Step In
0060 =    STPOCM    equ    01100000B ;(I)Step Out
0080 =    RSECCM    equ    10000000B ;(II)Read Record
00A0 =    WSECCM    equ    10100000B ;(II)Write Record
00C0 =    RADRCM    equ    11000000B ;(III)Read addr
00E0 =    RTRKCM    equ    11100000B ;(III)Read track
00F0 =    WTRKCM    equ    11110000B ;(III)Write track
00D0 =    FINTCM    equ    11010000B ;(IV)Force int

```

```

;1793 Command Options (1st letter matches 1793 spec)

```

```

0000 =    RS03MS    equ    00000000b ;(I) 3 mS step rate
0001 =    RS06MS    equ    00000001b ;(I) 6 mS step rate
0002 =    RS10MS    equ    00000010b ;(I) 10 mS step rate
0003 =    RS15MS    equ    00000011b ;(I) 15 mS step rate
0004 =    VERTRK    equ    00000100b ;(I) verify on destination track
0008 =    HDLOAD    equ    00001000b ;(I) head load
0010 =    UPDTTR    equ    00010000b ;(I) update track reg
0002 =    F1SCMP    equ    00000010b ;(II)enable side compare
0008 =    F2SCS1    equ    00001000b ;(II)side compare, side 1
0010 =    MULTCM    equ    00010000b ;(II) read/write multiple
0004 =    ED15MS    equ    00000100b ;(II & III) settling delay
0001 =    A0DDM     equ    00000001B ;(III)write deleted-data mark

```

```

;Composite 1793 commands

```

```

;Note: For some controllers, disk will always appear
;ready if not HDLOAD.

```

```

000B =    RESTOR    equ    RSTRCM+RS15MS+HDLOAD ;Restore
0086 =    RDSECT    equ    RSECCM+ED15MS+F1SCMP ;read sector, side 0

```

```

;WD1793 status after a command

```

```

0001 =    SBUSY     equ    00000001b ;(All) busy
0008 =    SCRCER    equ    00001000b ;(All) CRC error
0040 =    SWPROT    equ    01000000b ;(All) write protect
0080 =    SNORDY    equ    10000000b ;(All) drive not ready
0002 =    SINDEXX   equ    00000010b ;(I) index
0004 =    STRAK0    equ    00000100b ;(I) track 0
0010 =    SSEKER    equ    00010000b ;(I) seek error
0020 =    SHEADL    equ    00100000b ;(I) head loaded
0020 =    SWFALT     equ    00100000b ;(write cmds) write fault
0002 =    SDATREQ   equ    00000010b ;(II & III) data request state
0004 =    SLOSTD    equ    00000100b ;(II & III) lost data error
0010 =    SRNFER    equ    00010000b ;(II & III) record not found
0020 =    SRCTYF    equ    00100000b ;(II & III) record type fault

```

```

                                MEMON80.PRN
0020 =      SWFALT equ      00100000b ;(II & III) write fault
                                endif ;WD179x
                                ;=====
                                ;End of Definitions
                                ;=====

                                ;=====
                                ;= Memon/80 fixed-location Entry points =
                                ;= These are organized so that most of =
                                ;= them align with CP/M BIOS offsets:  =
                                ;= MEINIT ~ BOOT                        =
                                ;= MECSTA ~ CONST                      =
                                ;= MECIN ~ CONOUT                      =
                                ;= MECOUT ~ CONOUT                     =
                                ;= METPDO ~ PUNCH                      =
                                ;= METPDI ~ READER                     =
                                ;=====

F000                                org      MEBASE                ;Memon/80 ROM start
F000 F3      MEINIT: di                                ;(00)Cold-start entry point
                                                ;ints off for a little while

F001 AF      xra      a
F002 57      mov      d,a                                ;So code can recognize Memon

F003 C31FF0   MEWARM: jmp      INIT

                                ;the following 7 entry points can be called by CP/M,
                                ;and have been designed with minimal stack usage.

F006 C397F6   MECSTA: jmp      KSTAT                ;(06)Test Console keyboard
                                                ;Z set, 0 means no data
                                                ;Z clear, a=FF if data AVAIL
F009 C38CF6   MECIN:  jmp      KDATA                ;(09)Get Console chr in a
F00C C37FF6   MECOUT: jmp      PRINTC               ;(0C)Send C to Console
                                                ;a=c=chr on ret

                                if TPORT
F00F C39FF6   METPSI: jmp      TPISTA                ;(0F)Test Transfer Port input
                                                ;Z set, a=0 if no data
                                                ;Z clear, a=FF if data
F012 C3C7F6   METPDO: jmp      TPDATC               ;(12)Send C to Transfer Port
F015 C3A6F6   METPDI: jmp      TPIDAT               ;(15)Get Transfer Port chr in a
METPSO:                                ;Get T-Port output status
                                                ;a=0 & Z set if not ready
                                                ;a=FF & Z clear if ready

                                ;Fall into TPOSTA
                                endif ;TPORT

                                ;***CP/M External Subroutine*****
                                ;Get Transfer Port Output Status
                                ;On Exit:
                                ; Z clear and a=FFh if the printer can accept a chr
                                ; Z set and a=0 of the print queue is full
                                ;*****
TPOSTA:

                                if TPORT and (not TMAP)
F018 DBED      in      TSTAT
                                endif ;TPORT and (not TMAP)

```

# MEMON80.PRN

```

if TPORT and TMAP
    lda    TSTAT
endif ;TPORT and TMAP

if TPORT and TISTAT
    cma                    ;inverted status bit
endif ;TPORT and TISTAT

if TPORT
    ani    TTXRDY
    jmp    DOSTAT
endif ;TPORT

F01A E604
F01C C39BF6

;=====
;= Cold-Start Initialization =
;=====
;On Entry:
; a=0
; Interrupts are disabled
;-----
INIT:

;*****
;Serial Port Initialization
;(conditional assembly depending on which
;serial port options are selected above)
;*****

;-----
;Initialize RAM flags for any interrupt-driven I/O port
;On Entry and Exit:
; a=0
;-----
if CRXINT
    sta    CIFLAG
endif ;CRXINT

if TRXINT
    sta    TIFLAG
endif ;TRXINT

;-----
;Initialize any/all INS8250 UARTs
;On Entry:
; a=0
;-----
;INS8250 as Console

if C8250 and (not T8250)
    out    CSINTEN        ;Interrupts off
    out    CSTAT
    mvi    a,0Fh          ;modem control reg
    out    CSMDMCT        ;Handshakes true
    mvi    a,CSTOPS or CDLAB ;baud rate divisor access
    out    CSLCTRL        ;and default stop bits
    mvi    a,CPBAUD SHR 8 ;Divisor high byte
    out    CSINTEN
    mvi    a,CPBAUD and 0FFh

```

```

MEMON80.PRN
F031 D3FA          out      CDATA          ;Divisor low byte
F033 3E03          mvi      a,CSTOPS        ;stop pointing to divisor
F035 D3FD          out      CSLCTRL
endif ;C8250 and (not T8250)
;INS8250 as Transfer Port
if T8250 and (not C8250)
    out      TSINTEN          ;Interrupts off
    out      TSTAT
    mvi      a,0Fh            ;modem control reg
    out      TSMDMCT          ;Handshakes true
    mvi      a,TSTOPS or TDLAB ;baud rate divisor access
    out      TSLCTRL          ;and default stop bits
    mvi      a,TPBAUD SHR 8   ;Divisor high byte
    out      TSINTEN
    mvi      a,TPBAUD and 0FFh
    out      TDATA            ;Divisor low byte
    mvi      a,TSTOPS          ;stop pointing to divisor
    out      TSLCTRL
endif ;T8250 and (not C8250)
if C8250 and T8250
    out      CSINTEN          ;Interrupts off
    out      CSTAT
    out      TSINTEN
    out      TSTAT
    mvi      a,0Fh            ;modem control reg
    out      CSMDMCT          ;Handshakes true
    out      TSMDMCT          ;Handshakes true
    mvi      a,CSTOPS or CDLAB ;baud rate divisor access
    out      CSLCTRL          ;and default stop bits
    mvi      a,CPBAUD SHR 8   ;Divisor high byte
    out      CSINTEN
    mvi      a,CPBAUD and 0FFh
    out      CDATA            ;Divisor low byte
    mvi      a,CSTOPS          ;stop pointing to divisor
    out      CSLCTRL
    mvi      a,TSTOPS or TDLAB ;baud rate divisor access
    out      TSLCTRL          ;and default stop bits
    mvi      a,TPBAUD SHR 8   ;Divisor high byte
    out      TSINTEN
    mvi      a,TPBAUD and 0FFh
    out      TDATA            ;Divisor low byte
    mvi      a,TSTOPS          ;stop pointing to divisor
    out      TSLCTRL
endif ;C8250 and T8250

```

```

MEMON80.PRN
;-----
;Initialize any/all I/O or memory-mapped 8251
;UARTs for async polled access, no interrupts.
; On Exit:
;   a=0
;-----
if C8251 or T8251
    lxi    h,IN8251        ;8251 INIT table
SI2LUP: mov    a,m
endif ;C8251 or T8251

if C8251 and (not CMEMAP)
    out    CTRL            ;I/O-mapped 8251
endif ;C8251 and (not CMEMAP)

if C8251 and CMEMAP
    sta    CTRL            ;memory-mapped 8251
endif ;C8251 and CMEMAP

if T8251 and (not TMAP)
    out    CTRL            ;I/O-mapped 8251
endif ;T8251 and (not TMAP)

if T8251 and TMAP
    sta    CTRL            ;memory-mapped 8251
endif ;T8251 and TMAP

if C8251 or T8251
    inx    h
    xri    SI24            ;last one?
    jnz    SI2LUP
endif ;C8251 or T8251

if CISIO2A or CISIO2B or TISIO2A or TISIO2B
    out    SIOCTL          ;a=0: disable SIO-2 interrupts
endif ;CISIO2A or CISIO2B or TISIO2A or TISIO2B
;-----
;Initialize any/all MC6850 ACIAs
;-----
if C6850 or T6850
    mvi    a,U68RES        ;ACIA reset
endif ;C6850 or T6850

if C6850
    out    CTRL            ;2SIO port
endif ;C6850

if T6850
    out    CTRL            ;2SIO port
endif ;T6850

if C6850 or T6850
    mvi    a,U68ST2        ;Port set up
endif ;C6850 or T6850

if C6850
    out    CTRL            ;2SIO port
endif ;C6850

if T6850
    out    CTRL            ;2SIO port
endif ;T6850

```

```

endif ;T6850
;-----
;Initialize the baud rates for both CCS 2718 serial channels,
;including setting their baud rates. (If port B is used, it
;will be initialized in the 8251 initialization below.) If
;one of the CCS 2718 serial ports is not used, then its baud
;rate will default to a value of 0, which is 19200 baud.
;-----
CBD18  set    0
TBD18  set    0
      if CC2718A or CC2718B
CBD18  set    CPBAUD
BASE18 set    CBASE
      endif ;CC2718A or CC2718B

      if TC2718A or TC2718B
TBD18  set    TPBAUD
BASE18 set    TBASE
      endif ;TC2718A or TC2718B

      if CC2718A or CC2718B or TC2718A or TC2718B
          mvi    a,(CBD18 or TBD18) and 0FFh
          out     BASE18
      endif ;CC2718A or CC2718B or TC2718A or TC2718B

;-----
;Initialize both CCS 2719 channels,
;including setting their baud rates
;-----
if DART

;Initialize both DART channels

F037 21D3F6          lxi    h,DITAB

F03A 7E             DILOOP: mov    a,m
F03B 23             inx     h
                      endif ;DART

                      if CC2719A or CC2719B
                          out     CTRL
                      endif ;CC2719A or CC2719B

F03C D3ED           if TC2719A or TC2719B
                          out     CTRL
                      endif ;TC2719A or TC2719B

F03E FEEA           if DART
F040 C23AF0          cpi     DI7          ;last one?
                      jnz     DILOOP

;set the stop bits for each channel

F043 3E04           mvi     a,DWR4
                      endif ;DART

                      if CC2719A or CC2719B
                          out     CTRL
                      endif ;CC2719A or CC2719B

F045 D3ED           if TC2719A or TC2719B
                          out     CTRL
                      endif ;TC2719A or TC2719B

```



# MEMON80.PRN

```

if CC2719A or CC2719B
    mvi    a,CSTOPS
    out    CTRL
endif ;CC2719A or CC2719B

if TC2719A or TC2719B
    mvi    a,TSTOPS
    out    TCTRL
endif ;TC2719A or TC2719B

;Initialize baud rates in the CTC

if CC2719A or CC2719B
    mvi    a,CPBAUD SHR 8 ;set up Console baud
    out    CCTC
    mvi    a,CPBAUD and 0FFh
    out    CCTC
endif ;CC2719A or CC2719B

if TC2719A or TC2719B
    mvi    a,TPBAUD SHR 8 ;set up Transfer Port baud
    out    TCTC
    mvi    a,TPBAUD and 0FFh
    out    TCTC
endif ;TC2719A or TC2719B

;-----
;Initialize both Compupro Interfacer channels
;-----
if CIFAC or TIFAC
    mvi    a,IFIRST
endif ;CIFAC or TIFAC

if CIFAC
    out    CTRL
endif ;CIFAC

if TIFAC
    out    TCTRL
endif ;TIFAC

;-----
;Initialize Compupro Interfacer II
;-----
if CIFAC2
    mvi    a,CIFIRST
    out    CTRL
endif ;CIFAC2

if TIFAC2
    mvi    a,TIFIRST
    out    TCTRL
endif ;TIFAC2

;-----
;Initialize both Cromemco TU-ART channels
;-----
if CTUART or TTUART
    mvi    a,TURST ;reset
endif ;CTUART or TTUART

if CTUART

```

```

MEMON80.PRN
    out    TCCTRL
endif ;CTUART

if TTUART
    out    TTCTRL
endif ;TTUART

if CTUART or TTUART
    xra    a                    ;disable interrupts
endif ;CTUART or TTUART

if CTUART
    out    TCINTE
endif ;CTUART

if TTUART
    out    TTINTE
endif ;TTUART

if CTUART
    mvi    a,CPBAUD SHR 8 ;Console high baud rate bit
    out    TCCTRL

    mvi    a,CPBAUD and 0FFh ;set Console baud rate
    out    TCPBAUD
endif ;CTUART

if TTUART
    mvi    a,TPBAUD SHR 8 ;Transfer PT high baud rate bit
    out    TTCTRL

    mvi    a,TPBAUD and 0FFh ;set Transfer Port baud rate
    out    TTPBAUD
endif ;TTUART

;-----
;Install jump vectors for H8-5 interrupts
;-----
if CH85
    mvi    a,JMP
    sta    P8INT3 ;addrss of PAM-8/XCON-8 Int 3 vector
    lxi    h,CONINT
    shld   P8INT3+1
endif ;CH85

if TH85
    mvi    a,JMP
    sta    P8INT3 ;addrss of PAM-8/XCON-8 Int 3 vector
    lxi    h,TBINT
    shld   P8INT3+1
endif ;TH85

;-----
;Initialize the Ithaca Intersystems
;Series II VIO serial port(s)
;-----
if CIVIO2
    in      CCTRL                    ;reset mode byte flipflop

    mvi    a,CPBAUD shr 8 ;set Console baud rate
    out    CMODE                    ;<MR17:MR10>

    mvi    a,CPBAUD and 0FFh

```

```

MEMON80.PRN
    out    CMODE    ;<MR27:MR20>
endif ;CIVIO2

if TIVIO2
    in      TCTRL    ;reset mode byte flipflop

    mvi     a,TPBAUD shr 8 ;default t-port baud rate
    out     TMODE    ;<MR17:MR10>

    mvi     a,TPBAUD and 0FFh
    out     TMODE    ;<MR27:MR20>
endif ;TIVIO2

if CIVIO2 or TIVIO2
    mvi     a,CTL2651    ;Control reg initialization
endif ;CIVIO2 or TIVIO2

if CIVIO2
    out     CCTRL
endif ;CIVIO2

if TIVIO2
    out     TCTRL
endif ;TIVIO2

```

```

;-----
;Initialize any/all Jade Serial/Parallel I/O serial ports
;-----

```

```

if CJADSP or TJADSP
    mvi     a,JRESET
endif ;CJADSP or TJADSP

```

```

if CJADSP
    out     CCTRL
endif ;CJADSP

```

```

if TJADSP
    out     TCTRL
endif ;TJADSP

```

```

;*****
;End of Serial Port Initialization
;*****

```

```

;-----
;Hunt for the highest contiguous RAM page, and create
;a stack at its top. (This assumes memory comes in
;256-byte pages.) Leave all memory as we found it.
;If Memon/80 is running in low memory (e.g. for
;debugging) then don't hunt in Memon's memory space.
;-----

```

```

HUNTA    set     STACK    ;Hunt starting at 0

    if RAMCOD    ;Running in low memory?
HUNTA    set     (CODEND and 0FF00h) or STACK
    ;y: Hunt after Memon/80
    endif ;RAMCOD

    if RAMHNT
        lxi     h,HUNTA

F053 21B000
F056 24        FSLOOP: inr     h    ;next RAM page
F057 CA63F0    jz      FSEND    ;end of RAM?

```

# MEMON80.PRN

```

F05A 7E      mov     a,m           ;get original RAM
F05B 47      mov     b,a           ;and remember it
F05C 2F      cma
F05D 77      mov     m,a           ;write the inverse
F05E BE      cmp     m             ;Did it write right?
F05F 70      mov     m,b           ;Put original RAM back
F060 CA56F0  jz      FSLOOP        ;keep looking if RAM OK

FSEND:

F063 25      dcr     h             ;last RAM page
F064 F9      sphl
F065 2E8F    mvi     l,RAMBEG      ;For announcement

endif ;RAMHNT

if not RAMHNT
;Set up stack in RAM, and load hl with our 1st RAM address

        lxi     sp, STACK
        lxi     h,RAMBEG          ;For announcement
endif ;not RAMHNT

if TPORT
;set up default: Transfer Port enabled (H<>0 here)
F067 E5      push    h             ;h=port flag at top of stack
endif ;TPORT

;-----
;print Signon Banner
;-----
F068 CD11F6  call    CILPRT        ;Enables ints too, if ENINTS
F06B 4D656D6F6E db      'Memon/80 '
F074 322E    db      ((VERSION and 0F0h)/16)+'0','.'
F076 34      db      (VERSION and 0Fh) +'0'
F077 206279204D db      ' by M.Eberhard',CR

        if not HELPC ;HELPC option also adds LF's after CR's
        db      LF
endif ;not HELPC

F086 52414D3AA0 db      'RAM:', ' '+80h

;Announce RAM location. hl currently points
;to the address of MEMON's 1st RAM usage

F08B CD55F6  call    PHLCHX        ;Trashes bc

;Fall into MAIN
;*****
;Command Processor Main Loop
;Get and process commands
;*****

;Print the prompt, and get a line of keyboard input

F08E 018EF0  MAIN:   lxi     b,MAIN      ;create command-return
F091 C5      push    b               ;..address on the stack

F092 CD11F6  call    CILPRT        ;print CR,LF, prompt

```

```

MEMON80.PRN
F095 BE          db      PROMPT+80h      ;Enables ints too, if ENINTS
F096 CDCCF5      call    GETLIN           ;get user input line
                                           ;de=beginning of line
                                           ;Z set if no chr found
                                           ;0 at end of line

F099 D8          rc              ;No command? just ignore.

;Search the command list, and execute a command if found

F09A EB          xchg      d,h           ;command address to h1
F09B 118FF7      lxi       d,COMTAB-1     ;point to command table

F09E 4E          mov       c,m           ;1st command byte in c
F09F 23          inx       h             ;2nd command byte in m

;Search through table at de for a 2-chr match of c,m. Note
;that c & m have their parity bits stripped. The msb of the
;2nd table chr is the high bit of the address offset, so
;that the range of address offsets is 512 bytes.

F0A0 13          NXTCOM: inx      d       ;skip over address offset
F0A1 1A          ldax      d
F0A2 B7          ora       a             ;test for table end
F0A3 CAC1F5      jz        CMDERR        ;not in table

F0A6 A9          xra       c             ;test 1st chr

F0A7 47          mov       b,a           ;temp save result
F0A8 13          inx       d             ;2nd table chr

F0A9 1A          ldax      d
F0AA AE          xra       m             ;test 2nd chr
F0AB 13          inx       d             ;point to address offset

F0AC B0          ora       b             ;both chrs match?

if LOWERC
F0AD E6DF        ani       ('a'-'A') xor 0FFh ;lowercase ok
endif ;LOWERC

F0AF 07          rlc              ;address offset msb to lsb
F0B0 FE02        cpi       2            ;0 or 1 means match
F0B2 D2A0F0      jnc       NXTCOM

;Got a match. Compute routine address and put it on stack
;a=msb of address offset. Z is clear here, due to cpi above

F0B5 47          mov       b,a           ;offset msb
F0B6 23          inx       h             ;skip past 2nd command letter

F0B7 EB          xchg              ;(h1)=offset addr of routine
                                           ;de=input buffer pointer

F0B8 4E          mov       c,m           ;bc=address offset (b=0 or 1)
F0B9 21EAF0      lxi       h,CMDBAS      ;base of command routines
F0BC 09          dad       b             ;h1=address of routine

F0BD E5          push      h             ;command address on stack

;Special case: the TE command gets a non-hex parameter.

```

```

MEMON80.PRN
;The address offset for the TE command is 0. a=msb, c=lsb.
    if (TPORT and (not ROM2K)) or (ROM2K and (not TPORT))
        ora    c
        rz
    endif ;(TPORT and (not ROM2K)) or (ROM2K and (not TPORT))
;ROM2K and TPORT special case: the 1st 2 commands have a non-
;hex 1st parameter, and the 1st 1 is only a jmp (3 bytes).
; CE <command line text> (execute a CP/M program)
; TE <exit chr> (terminal mode)
    if ROM2K and TPORT
        rrc                    ;put msb in high bit
        ora    c              ;combine low bits
        ani    0FCh          ;0 if offset = 0 or 3
        rz
    endif ;ROM2K and TPORT
;Get following hex parameter (if any) and put it in hl. Set the
;carry flag if no parameter present. Leave de pointing to the
;1st chr after the 1st parameter. 'return' to the command
;routine on the stack.
F0BE 0F                    db    21h                ;"LXI H" opcode skips 2
F0BF B1
F0C0 E6FC
F0C2 C8
;Skip over PHFHEX and into FNDHEX
;***Subroutine*****
;Push h, then scan past spaces and get a hex value
;On Entry:
; de=address of next item in the input line buffer
;On Exit:
; Top-of-stack=original HL value
; hl=value from input buffer
; de advanced past chr
; Z set, carry clear if value found
; carry set, a=0 if no value found
;*****
F0C4 E3    PHFHEX: xthl        ;push hl
F0C5 E5        push    h        ;..beneath return address
;Fall into FNDHEX
;***Subroutine*****
;Scan past spaces and get a hex value
;On Entry:
; de=address of next item in the input line buffer
;On Exit:
; hl=value of last 4 digits found, defaults to 0
; Z set
; de advanced past last digit found
; carry clear if value found
; carry set if no value found
;*****
F0C6 210000    FNDHEX: lxi    h,0        ;default & initial value
F0C9 CDDEF5    FHExLP: call    SSPACE    ;skip spaces, get 1st digit
F0CC D8        rc                ;carry set if no digits
F0CD 29        dad    h            ;make room for the new digit
F0CE 29        dad    h
F0CF 29        dad    h

```

```

MEMON80.PRN
F0D0 29          dad      h

        if LOWERC
F0D1 FE61          cpi      'a'          ;Lowercase letter?
F0D3 DAD8F0        jc      FXHL1
F0D6 E6DF          ani      ('a'-'A') xor 0FFh ;y: convert to uppercase
FXHL1:
        endif ;LOWERC

F0D8 CD04F6        call     HEXCON        ;convert a to binary
F0DB D2C1F5        jnc      CMDERR        ;not valid hexadecimal value?

F0DE 85           add      l
F0DF 6F           mov      l,a            ;move new digit in
F0E0 13           inc      d            ;bump the pointer

F0E1 1A           ldax     d            ;get next character
F0E2 B7           ora      a            ;End of line?
F0E3 C8           rz              ;Y: a=0, carry clear

F0E4 D620          sui      ' '          ;value separator?
F0E6 C2C9F0        jnz      FHEXLP        ;N: it's another hex digit

F0E9 C9           ret              ;a=0 for exit, keep carry clear

;=====
;base address for commands
;=====
F0EA =            CMDBAS equ $

;***ROM2K Command Routine*****
;CE Command Entry
;Jump to actual routine, more than 512 bytes from CMDBAS
; ---> Must be first, at CMDBAS <---
;*****
        if ROM2K
F0EA C3CEF2        CECMD: jmp      GCECMD
        endif ;ROM2K

        if TPORT
;***Command Routine*****
;TE [<EXchr>] (Simple Terminal mode)
;
; --->not ROM2K: TECMD must be first, at CMDBAS<---
; --->ROM2K: TECMD must be second, at CMDBAS+3<---
;
;Send all Console keyboard data to Transfer Port, and
;send all Transfer Port data to the Console. <EXchr>
;from the keyboard exits. (default to DTEXTIT.)
;On Entry:
; de=points to the 1st chr in RAMBUF past the 'TE' cmd
;*****
F0ED CD11F6        TECMD: call     CILPRT        ;announce exit chr
F0F0 4578697420    db      'Exit ', '^'+80h

F0F6 CDDEF5        call     SSPACE        ;get optional exit character
F0F9 B7           ora      a            ;nothing entered?
F0FA C2FFF0        jnz      TMNL1        ;got an exit value in a
F0FD 3E03          mvi      a,DTEXTIT    ;default exit chr

;Convert exit chr to uppercase, non-control,
;and print exit character message

```

```

MEMON80.PRN
F0FF E61F    TMNL1:  ani    1Fh    ;make it a CTRL chr
F101 6F      mov    1,a    ;remember exit chr

F102 F640    ori     'C'-CTRLC ;make it printable
F104 CD80F6  call    PRINTA

F107 CD11F6  call    CILPRT    ;CR,LF,CR to be pretty
F10A 8D      db     CR+80h    ;(start on a new line)

;Be a Terminal until we get an exit character=1

F10B CD97F6  TLOOP:  call    KSTAT    ;anything typed?
F10E C48CF6  cnz     KDATA    ;Y:get the kbd data

F111 BD      cmp     1        ;exit chr?
F112 C8      rz      ;Y: done

F113 B7      ora     a        ;typed chr? (ignore null)
F114 C4C8F6  cnz     PBODAT    ;kbd data to Transfer Port

F117 CD9FF6  call    TPISTA    ;any Transfer Port data?
                                ;NZ if so

F11A C4A6F6  cnz     TPIDAT    ;get Transfer Port data
                                ;always returns w/ NZ
F11D C480F6  cnz     PRINTA    ;and send it to Console
F120 C30BF1  jmp     TLOOP

endif ;TPORT

;***Command Routine*****
;? Command Entry
;Jump to actual routine, more than 512 bytes from CMDBAS
;*****
if HELPC
F123 C306F7  HELP:  jmp     GHELP
endif ;HELPC

;***ROM2K Command Routine*****
;MT Command Entry
;Jump to actual routine, more than 512 bytes from CMDBAS
;*****
if ROM2K
F126 C3CAF3  MTCMD: jmp     GMTCMD
endif ;ROM2K

;***ROM2K Command Routine*****
;SE Command Entry
;Jump to actual routine, more than 512 bytes from CMDBAS
;*****
if ROM2K
F129 C33CF3  SECMD: jmp     GSECMD
endif ;ROM2K

;***ROM2K Command Routine*****
;TB Command Entry
;Jump to actual routine, more than 512 bytes from CMDBAS
;*****
if ROM2K and TSBAUD
F12C C3EBF2  TBCMD: jmp     GTBCMD
endif ;ROM2K and TSBAUD

;***Command Routine*****
;BO Command Entry

```



```

MEMON80.PRN
;Jump to actual routine, more than 512 bytes from CMDBAS
;*****
BOCMD:
JMPBOT set TRUE

if H17R or MDUBLR or MICROR or NSTRSR or NSTRDR or IMD8R or IMDMR
BOOTER set TRUE
JMPBOT set FALSE
endif ;H17R or MDUBLR or MICROR or NSTRSR or NSTRDR or IMD8R or
IMDMR

if ROM2K and BOOTER
F12F CD11F6 call CILPRT ;2K ROM, so be verbose
F132 426F6F7469 db 'Booting',CR+80h
endif ;ROM2K and BOOTER

;-----
;Boot using disk controller's onboard ROM
;-----
if H17R or MDUBLR or MICROR or NSTRSR or NSTRDR
jmp DMBASE ;use the H17's onboard boot code
endif ;H17R or MDUBLR or MICROR or NSTRSR or NSTRDR

;-----
;Boot from IMSAI MDC-DIO Floppy Controller's onboard ROM
;(Boot address depend on 8" or minidisk)
;-----
if IMD8R
DBOOT equ DMBASE ;Boot from 8" disk
endif ;IMD8R

if IMDMR
DBOOT equ DMBASE+3 ;Boot from minidisk
endif ;IMDMR

if IMD8R or IMDMR
IDRSEL equ DIBASE+0Eh ;disable controller, enable RAM
IDCSEL equ DIBASE+0Fh ;disable RAM, enable controller

out IDCSEL ;disable RAM, enable controller
;a's value doesn't matter

jmp DBOOT ;Use onboard boot code

endif ;IMD8R or IMDMR

;-----
;The boot code might be too big to fit here
;-----
if BOOTER and JMPBOT
F13A C3AEF4 jmp GBOCMD
endif ;BOOTER and JMPBOT

;***Command Routine*****
;EN [<ADR>] (enter data into memory)
;
;Get hex values from the keyboard and enter them
;Sequentially into memory, starting at <ADR>. a blank
;line ends the routine and returns control to the
;command mode. values may be separated by spaces or CR's.
;Print the current address at the beginning of each line.
;On Entry:
; hl=address, defaulting to 0

```

```

MEMON80.PRN
; carry set if none entered
;*****
F13D CD48F6 ENCMD: call    PHLADR          ;print hl as an address
;trash c, b=0

F140 CDCCF5          call    GETLIN          ;get a line of user input
F143 D8              rc              ;Z=blank line terminates

;Get hex data from the user input line and write it to memory

F144 CDC4F0 ENLOOP: call    PHFHEX          ;push memory address and
;..get/convert next value

F147 7D              mov     a,l            ;get low byte as converted

F148 E1              pop     h              ;recover memory address
F149 77              mov     m,a            ;put in the value
F14A 23              inc     h              ;next address

F14B CDDEF5          call    SSPACE          ;scan to next input value
F14E D244F1          jnc     ENLOOP          ;not end of line: continue

F151 C33DF1          jmp     ENCMD          ;END of line: start new line

;***Command Routine*****
;FI [<ADR> [<BCNT> [<VAL>]]] (fill memory)
;
;Fill <BCNT> bytes of memory with <VAL> from <ADR>.
;If <VAL> is not provided, then fill the specified
;range with 00. If <BCNT> is not provided, then fill
;memory until fill reaches the RAM page. If <ADR> is
;not provided, then fill from 0.
;On Entry:
;  hl=<ADR>
;  Carry set if none entered
;  de points to <BCNT>, <VAL> follows, if any
;*****
FICMD:

    if RAMCOD                      ;Running in low memory?
        push    h
        lxi     b,-CODEND          ;will this wipe out Memon/80?
        dad     b                   ;16-bit compare
        pop     h
        jnc     CMDERR              ;y: error
    endif ;RAMCOD

F154 CDC4F0          call    PHFHEX          ;push address and
;..get hl=byte count default=0

F157 CDC4F0          call    PHFHEX          ;push byte count and
;..get fill data, default 0
F15A 55              mov     d,l        ;fill data in d

F15B C1              pop     b              ;bc has byte count

    if RAMHNT
F15C CDC9F2          call    RAMPAG          ;get h=RAM page
F15F 5C              mov     e,h
    endif ;RAMHNT

    if not RAMHNT
        mvi     e,RAMEND/256
    endif

```

```

MEMON80.PRN
endif ;not RAMHNT

F160 E1          pop     h          ;hl has start address

F161 7C          FILOOP: mov     a,h          ;filling Memon/80's RAM page?
F162 BB          cmp     e
F163 C8          rz          ;Y: done

F164 72          mov     m,d          ;fill data

F165 23          inc     h
F166 0B          dcx     b          ;done yet?
F167 78          mov     a,b
F168 B1          ora     c
F169 C261F1      jnz     FILOOP

F16C C9          ret

;***Command Routine*****
;HD <ADR> <BCNT> (Intel hex dump)
;
;Dump the specified memory range to the
;Transfer Port as an Intel hex file
;On Entry:
;  hl=ADR
;  de points to <BCNT>
;*****
F16D CDBC F5      HDCMD: call    GETHEX          ;save 1st address
                                           ;get hl=byte count

F170 D1          pop     d          ;recover start address
F171 EB          xchg          ;hl has start, de has count

;Loop to send requested data in HDRLEN-byte records
;Send record-start

F172 CDB9 F6      HDLINE: call    TPCRLF          ;send CRLF to Transfer Port
F175 010010      lxi     b,HDRLEN*256          ;b=bytes/line,
                                           ;..b<>0 for TPOUT,
                                           ;..c=0 initial checksum

F178 3E3A          mvi     a,':'          ;record start
F17A CDC0 F6      call    TPOUT

;Compute this record length (b=HDRLEN here)

F17D 7B          mov     a,e          ;short last line?
F17E 90          sub     b          ;normal bytes/line
F17F 7A          mov     a,d          ;16-bit subtract
F180 99          sbb     c          ;c=0 here
F181 D285 F1      jnc     HDLIN1          ;N: full line

F184 43          mov     b,e          ;Y:short line
HDLIN1:

;If byte count is 0 then go finish EOF record

F185 78          mov     a,b
F186 B7          ora     a
F187 CAAFF6      jz     HDEOF

```

```

MEMON80.PRN
;Send record byte count=a to the Transfer Port (b<>0)
;checksum=0 in c here

F18A CD5EF6          call    PAHCSM          ;send byte count

;Send the address at the beginning of each line,
;computing the checksum in c

F18D CD57F6          call    PHLHEX          ;hl=address, b<>0

;Send the record type (00), checksum in c

F190 AF              xra      a
F191 CD5EF6          call    PAHCSM

;Send b bytes of hex data on each line, computing
;the checksum in c

F194 CD5DF6          HDLOOP: call    PMHCSM          ;send chr to Transfer Port

F197 1B              dcx      d
F198 23              incx     h
F199 05              dcr      b                ;next
F19A C294F1          jnz      HDLOOP

;Compute & send the checksum

F19D AF              xra      a
F19E 91              sub      c
F19F 04              inc      b                ;To Transfer Port
F1A0 CD5EF6          call    PAHCSM

;Give the user a chance to break in at the end of each line

F1A3 CD8BF5          call    CHKKBD
F1A6 C372F1          jmp      HDLINE          ;next record

;***Command Routine*****
;IN <PORT> (input from port)
;
;On Entry:
;  l=PORT
;Creates this routine on the stack, then executes
;it, then returns through PCAHEX to print value
;
;  nop
;  in    <PORT>
;  ret
;  <PCAHEX address for return>
;*****
F1A9 CD4FF6          INCMD: call    PCCOLS          ;': ' as separator

F1AC 1163F6          lxi      d,PCAHEX          ;create return address
F1AF D5              push     d                ;ret through PCAHEX

F1B0 26C9            mvi      h,RET            ;opcode
F1B2 E5              push     h                ;l=<PORT>

F1B3 2100DB          lxi      h,IN*256          ;NOP, in opcodes

F1B6 C3C2F1          jmp      IOCMD            ;recycle some code

;***Command Routine*****

```

```

MEMON80.PRN
;OT <PORT> <data> (output to port)
;
;On Entry:
;  l=PORT
;  de points to data
;
;Creates this routine on the stack, then executes it
;  NOP
;  OUT <PORT>
;  RET
;*****
F1B9 26C9  OTCMD:  mvi    h,RET          ;ret opcode, l=PORT
F1BB CDBC5          call    GETHEX          ;push <PORT>, RET opcode
;get l=<data>
F1BE 7D          mov     a,l          ;port data
F1BF 2100D3       lxi     h,out*256      ;NOP, out opcodes
;Fall into IOCMD
;-----
;Recycle some code
;On Entry:
;  l=0
;  h=IN or OUT opcode
;-----
F1C2 E5  IOCMD:  push   h          ;install IN or OUT opcode
;
;  if EXOPT
;      jmp     EXSTK          ;recycle some code
;  endif ;EXOPT
;
;  if not EXOPT
F1C3 65          mov     h,l          ;hl=0
F1C4 39          dad     sp          ;hl points to routine
F1C5 D1          pop     d          ;repair sp so that
F1C6 D1          pop     d          ;..we can return
;Fall into EXCMD to execute RAM routine at (hl)
endif ;not EXOPT
;***Command Routine*****
;EX [<ADR> [<OPT>]](execute)
;
;EXOPT=TRUE adds <OPT> to optionally disable the EPROM on an
;Altair-type board (such as on the Turnkey Module or my own
;88-2SIOJP) on the way to executing at the requested address.
;On Entry:
;  hl=address, default to 0
;  Carry set if none entered
;  Top-of-stack has main address
;*****
;  if not EXOPT
F1C7 E9  EXCMD:  pchl          ;push <ADR>, get l=<OPT>
;endif ;not EXOPT
;
;  if EXOPT
EXCMD:  call    PHFHEX          ;Anything but 1 just
;      dcr     l              ;..executes at <ADR>
;      rnz

```

# MEMON80.PRN

;Fall into EXECDP

;\*\*\*Subroutine\*\*\*\*\*  
;Execute 'IN FF' from RAM to disable PROM, then return  
;(Some boards disable PROM on any IN FF)  
;\*\*\*\*\*

EXECDP: mvi e,RET ;ret, don't care  
push d  
  
lxi d,0FFDBh ;IN FF opcode  
push d

;Fall into EXSTK

;\*\*\*Subroutine\*\*\*\*\*  
;Execute code on the stack  
;\*\*\*\*\*

EXSTK: lxi h,0 ;Find our RAM page to find  
dad sp ;..the code we just pushed  
  
pop d ;restore stack  
pop d

pchl ;execute the code we pushed  
endif ;EXOPT

if TPORT  
;\*\*\*Command Routine\*\*\*\*\*  
;TP [<0/1>] enable/disable Transfer Port  
;  
;0 means disabled (use Console), anything else  
;means Transfer Port enabled.  
;On Entry:  
; hl=enable state  
; Top-of-stack=return address to MAIN  
; next-on-stack=TP flag word  
;\*\*\*\*\*

F1C8 7D TPCMD: mov a,1 ;get state  
F1C9 210300 lxi h,3 ;TP flag is 3 back  
F1CC 39 dad sp ;..on the stack  
F1CD 77 mov m,a ;New state

F1CE C9 ret  
endif ;TPORT

;\*\*\*Command Routine\*\*\*\*\*  
;DU [<ADR> [<BCNT>]] (dump memory)  
;  
;Print <BCNT> bytes of memory contents from <ADR> on  
;the Console in HEX. If no count is specified, then  
;then print the contents of all memory, 10000h bytes.  
;Pause with the space bar, abort with control-C.  
;  
;On Entry:  
; hl=address  
; de points to <BCNT>, if any  
;\*\*\*\*\*

F1CF CDC4F0 DUCMD: call PHFHEX ;push address and  
;..get hl=byte count or 0  
;a=0  
F1D2 8D adc l ;carry set if none entered  
F1D3 6F mov l,a ;hl defaults to 1

# MEMON80.PRN

```

F1D4 D1          pop      d          ;recover start address
F1D5 EB          xchg                     ;hl has start, de has count

;Print the address at the beginning of each line

F1D6 CD48F6      DULINE: call    PHLADR          ;print hl as an address
F1D9 48          mov      c,b          ;Trashes c, sets b=0
                                   ;line byte counter=0

;Print up to 16 bytes of hex data
;(separated by spaces) on each line

F1DA E5          push     h          ;save mem pointer for ASCII dump

F1DB 7E          DULoop: mov     a,m
F1DC CD63F6      call    PCAHEX          ;chr to Console in hex, trash b
F1DF 0C          inc      c          ;line byte counter

F1E0 CD15F6      call    ILPRNT          ;space separates bytes
F1E3 A0          db      ' '+80h
F1E4 23          inc      h

F1E5 1B          dcx      d          ;all done with DU data?
F1E6 7A          mov     a,d
F1E7 B3          ora      e
F1E8 CAF1F1      jz       DLDONE          ;Y: print last ASCII too

F1EB 3E0F          mvi     a,0Fh          ;all done with row?
F1ED A5          ana      l          ;new line Every XXX0 hex
F1EE C2DBF1      jnz      DULoop

DLDONE:

    if DUPRTY
;Pad out the rest of the line so that the ASCII lines up
;for short lines. c=number of bytes printed so far
;(This just makes the display a little prettier.)

F1F1 3E10          mvi     a,10h          ;a full line has 16 bytes
F1F3 91          sub      c          ;how short is this line?
F1F4 CA02F2      jz       DNOPAD          ;No padding for full lines

F1F7 47          mov     b,a          ;loop counter

F1F8 CD15F6      DPLoop: call    ILPRNT
F1FB 2020A0      db      ' ',' '+80h          ;3 bytes per hex digit
F1FE 05          dcr      b
F1FF C2F8F1      jnz      DPLoop

DNOPAD:
    endif ;DUPRTY

;print up to 16 ASCII characters, or '.' if unprintable

F202 E1          pop      h          ;recover this line's address

F203 7E          ADLoop: mov     a,m          ;get a character
F204 3C          inc      a          ;DEL (7Fh) is also non-printing
F205 E67F        ani      7Fh          ;strip high (parity) bit
F207 FE21        cpi      ' '+1          ;Everything below space
F209 D20EF2      jnc      PRNTBL          ;..is non-printing

F20C 3E2F          mvi     a,'.'+1          ;A dot for non-printing chr

```

```

MEMON80.PRN
F20E 3D      PRNTBL: dcr      a          ;undo inc
F20F CD80F6      call     PRINTA

F212 23      inx      h          ;next byte on this ROW
F213 0D      dcr      c          ;this line's byte count
F214 C203F2      jnz     ADLOOP

F217 7A      mov      a,d        ;all done with dump?
F218 B3      ora      e
F219 C8      rz

;Give the user a chance to break in at the end of each line.
;Pause if anything except ^C typed. abort if ^C.
F21A CD7BF5      call     CPAUSE
F21D C3D6F1      jmp      DULINE      ;next line

;***Command Routine*****
;HL [<OFST>] (Intel Hex Load)
;
;Load an Intel hex file from the Transfer Port into memory
;AT the addresses specified in the hex file, with optional
;address offset <OFST>. Ends with any record with 0 data
;bytes, or if control-C is typed.
;
;HLRECS=TRUE adds record counting, with a report of the
;total number of records received when done.
;
;ROM2K will enable RAM read-back test when writing to RAM
;
;On Entry:
; hl=offset (defaults to 0)
;
;Register usage during hex load:
; b: Scratch
; c: record byte count
; d: record checksum
; e: RAM page address
; hl: memory address
; top of stack: address offset
; Next on stack: Record count
;*****
HLCMD:
    if HLRECS
F220 110000      lxi      d,0      ;initialize record count
F223 D5      push     d          ;keep it on the stack
    endif ;HLRECS

F224 E5      push     h          ;address offset onto stack

;Eat all chrs until we get record-start colon

F225 CD34F6      GETCOL: call     GETTPD
F228 FE3A      cpi      ':'
F22A C225F2      jnz     GETCOL

;Restart checksum, then get 4-byte record header: (a=0 here)
; c gets 1st byte=data byte count
; h gets 2nd byte=address high byte
; l gets 3rd byte=address low byte
; b gets 4th byte=record type (ignored)
; d computes checksum

```



```

MEMON80.PRN
F22D 110400      lxi      d,4      ;Initialize checksum in d
                                   ;get e=4 header bytes

;Shift in the four header bytes: c <- h <- l <- b

F230 4C          HLHDR:  mov     c,h      ;c=byte 1: byte count
F231 65          mov     h,l      ;h=byte 2: address msb
F232 68          mov     l,b      ;l=byte 3: address lsb
F233 CD98F5      call    GTPBYT      ;get header byte, do checksum
F236 1D          dcr     e
F237 C230F2      jnz     HLHDR

;Offset the record address by the value on top of the stack
;a=checksum so far here

F23A D1          pop     d      ;get offset

if HLRECS
F23B E3          xthl
F23C 23          inx     h      ;bump record count
F23D E3          xthl
endif ;HLRECS

F23E 19          dad     d      ;offset the address
F23F D5          push    d      ;save offset

if RAMCOD
push     h      ;Running in low memory?
lxi     d,-CODEND ;wipe out Memon/80?
dad     d      ;16-bit compare
pop     h
jnc     OVMERR   ;y: error
endif ;RAMCOD

F240 57          mov     d,a      ;d=checksum so far

;c=byte count
;d=checksum so far
;hl=RAM address for this record=record address+offset

;Test for 0-byte record, which is the EOF

F241 79          mov     a,c      ;c=byte count
F242 B7          ora     a      ;0-byte record?
F243 CA60F2      jz      GETCSM    ;Y: done

;Get e=RAM page address to test for overwriting. this
;bocks out a 256-byte region of memory wherever this
;program found RAM for its stack.

if RAMHNT
F246 E5          push    h
F247 CDC9F2      call    RAMPAG      ;get h=RAM page
F24A 5C          mov     e,h
F24B E1          pop     h
endif ;RAMHNT

if not RAMHNT
mvi     e,RAMEND/256
endif ;not RAMHNT

;Loop to get data into memory at hl.

```

```

MEMON80.PRN
F24C 7C      HLLLOOP: mov    a,h      ;overwrite our RAM?
F24D BB      cmp      e              ;
F24E CA8CF4  jz        OVMERR        ;Y: abort load

F251 CD98F5  call     GTPBYT        ;data byte in b, cksum in d
F254 70      mov      m,b

      if ROM2K
F255 78      mov      a,b          ;readback compare
F256 BE      cmp      m            ;successful write?
F257 C295F4  jnz      MEMERR        ;N: error
      endif ;ROM2K

F25A 23      inc      h
F25B 0D      decr     c
F25C C24CF2  jnz      HLLLOOP

F25F 0C      incr     c              ;remember >0 data bytes

;Get and test record checksum

F260 CD98F5  GETCSM: call    GTPBYT        ;get checksum in Z
F263 C283F4  jnz      CSMERR        ;NZ means checksum error

;All done with this record. Check its data byte count
;On Entry: c=0 for 0-byte record, c=1 otherwise

F266 CD15F6  call     ILPRNT        ;pacifier on console
F269 AE      db       PCFIER+80h

F26A 0D      decr     c              ;0-byte record?
F26B CA25F2  jz        GETCOL        ;N: go get another record

F26E E1      pop      h              ;remove offset from stack

      if HLRECS
F26F E1      pop      h              ;record count
F270 CD11F6  call     CILPRT
F273 526563733A db      'Recs:', ' '+80h
F279 C355F6  jmp      PHLCHX          ;hl in hex on console
      endif ;HLRECS

      if not HLRECS
      ret
      endif ;not HLRECS

;***Command Routine*****
;VE <SRC> <DST> <BCNT> (verify memory)
;
;Compare <BCNT> bytes of memory from
;<SRC> to <DST> and report differences
;On Entry:
; hl=<SRC>
; carry set if none entered
; de points to <DST>, <BCNT> follows
;*****
F27C CDB8F5  VECMD: call    GETHX2        ;save source address
                                           ;get & save destination
                                           ;get byte count

F27F E5      push     h              ;save byte count

F280 C3AEF2  jmp      VERIFY        ;actually verify

```

# MEMON80.PRN

```

;***command Routine Continuation*****
;CO <SRC> <DST> <BCNT> [<RPT>](copy memory)
; (RPT only available if CORPT is TRUE)
;
;Copy <BCNT> bytes of memory from <SRC> to <DST>
;repeat <RPT> times (for EPROM programming)
;On Entry:
;  hl=<SRC>
;  carry set if none entered
;  de points to <DST>, <BCNT>, <RPT> follow
;*****
F283 CDB8F5  COCMD:  call    GETHX2          ;save source address
                                           ;get & save destination
                                           ;get byte count

                                if CORPT
F286 CD11F6          call    CILPRT
F289 436F707969      db      'Copyin','g'+80h

F290 CDC4F0          call    PHFHEX          ;push byte count and
                                           ;..get repeat count
F293 8D              adc     1              ;On ret, a=0
                                           ;carry set if no value given

;Repeat copy the specified number of times (for
;programming an EPROM, e.g. a 2708 on a bytesaver)

CORLUP:
endif ;CORPT

F294 C1              pop     b              ;bc=count
F295 D1              pop     d              ;de=destination
F296 E1              pop     h              ;hl=source

F297 E5              push    h              ;save source
F298 D5              push    d              ;save destination
F299 C5              push    b              ;save count

                                if CORPT
F29A F5              push    psw            ;save a=repeat count
                                endif ;CORPT

;Loop to copy bc bytes from (hl) to (de)

F29B 7E              COLLOOP: mov    a,m
F29C 12              stax    d
F29D 23              inx     h
F29E 13              inx     d
F29F 0B              dcx     b
F2A0 78              mov     a,b
F2A1 B1              ora     c
F2A2 C29BF2          jnz     COLLOOP

                                if CORPT
;Repeat the copy as requested by the user

F2A5 CD15F6          call    ILPRNT          ;pacifier on Console
F2A8 AE              db      PCFIER+80h

F2A9 F1              pop     psw            ;recover repeat count

```

```

MEMON80.PRN
F2AA 3D          dcr      a          ;repeat as requested
F2AB C294F2      jnz      CORLUP
                endif ;CORPT

;Fall into VERIFY to verify the copy

;***Subroutine End*****
;Verify memory. Report errors to Console.
;On Entry:
;  top of stack=byte count
;  next on stack=destination address
;  next on stack - source address
;  next on stack=return address (to MAIN)
;*****
F2AE C1          VERIFY: pop     b          ;byte count
F2AF D1          pop     d          ;de=destination
F2B0 E1          pop     h          ;hl=source

                if CORPT
F2B1 CD11F6      call     CILPRT
F2B4 636865636B db      'checkin','g'+80h
                endif ;CORPT

;Loop to compare memory, reporting mismatches

F2BC 1A          VELOOP: ldax     d
F2BD BE          cmp      m          ;match?
F2BE C467F5      cnz      VERROR     ;N:done

F2C1 23          inx      h
F2C2 13          inx      d
F2C3 0B          dcx      b
F2C4 78          mov      a,b
F2C5 B1          ora      c
F2C6 C2BCF2      jnz      VELOOP

;Fall into RAMPAG to save a byte on the way
;TO returning to MAIN

;***Subroutine*****
;Get high byte of RAM page address
;On Entry:
;  sp is valid
;On Exit:
;  h=RAM page
;Trashes 1
;*****
F2C9 210000      if RAMHNT
F2CC 39          RAMPAG: lxi      h,0
                dad      sp
                endif ;RAMHNT

F2CD C9          ret

;***ROM2K Command Routine Continuation*****
;CE [command line]
;
;Execute CP/M program
; 1. Move command line into COMBUF
; 2. Install WBOOT jump at 0
; 3. Jump to USAREA
;On Entry:
;  de=points to the 1st chr in RAMBUF past the 'CE' cmd

```

```

                                MEMON80.PRN
; top-of-stack points to MAIN
;*****
; if ROM2K

GCECMD:

;Copy null-terminated command line to CP/M's
;command line buffer

F2CE 218000      lxi      h,COMBUF
F2D1 E5          push    h
F2D2 06FF        mvi      b,0FFh      ;chr count (minus null term)

F2D4 23          CLLOOP:  inx      h
F2D5 1A          ldax    d
F2D6 77          mov     m,a
F2D7 13          inx     d
F2D8 04          inr     b      ;count chrs

F2D9 B7          ora     a      ;copy 'till null termination
F2DA C2D4F2      jnz     CLLOOP

F2DD E1          pop     h      ;install chr count
F2DE 70          mov     m,b    ;..at 1st position in buffer

;Install WBOOT jump at 0
;a=0 here

F2DF 6F          mov     l,a      ;hl=0
F2E0 36C3        mvi     m,JMP

F2E2 2103F0      lxi     h,MEWARM
F2E5 220100      shld    WBOOTA

;Execute

F2E8 C30001      jmp     USAREA
endif ;ROM2K

;***ROM2K Command Routine Continuation*****
;TB <BAUD>
;
;Set and announce the Transfer Port's baud Rate
;(Only available for serial port boards with
;software-set baud rates)
;On Entry:
; l=<BAUD>
;*****
; if TSBAUD and ROM2K

F2EB DAC1F5      GTBCMD:  jc      CMDERR      ;Must supply baud rate

;Look up entry in baud rate table

F2EE 7D          mov     a,l      ;User selection
F2EF 87          add     a      ;Table has 4 bytes
F2F0 87          add     a      ;..per entry
F2F1 FE2C        cpi     BTEND-BTABLE
F2F3 D2C1F5      jnc     CMDERR

F2F6 4F          mov     c,a      ;c=selection*4 (0 for 110 baud)
F2F7 0600        mvi     b,0
F2F9 21DAF6      lxi     h,BTABLE

```

```

MEMON80.PRN
F2FC 09          dad      b          ;hl points to entry
;Get c=BRATE0 and m=BRATE1 from the table. check
;for unsupported baud rate (BRATE0 & BRATE1 are 0)
;b=0

F2FD 56          mov      d,m          ;value for BRATE0
F2FE 23          inx      h

F2FF 7E          mov      a,m          ;value for BRATE1
F300 B2          ora      d
F301 CAC1F5      jz       CMDERR

;Set baud rate, and set the stop bits to 1
;except for 110 baud, which gets 2 stop bits
;b=0

endif ;TSBAUD and ROM2K

if TSBAUD and ROM2K and (TC2718A or TC2718B)
;Both CCS2718 serial ports use the same I/O port for
;setting their baud rates. Here, we combine the fixed
;Console baud rate (if any) with the new transfer port
;baud rate.
    mvi      a,CBD18 and 0FFh ;default Console baud
    ora      d                ;new baud rate
    out      BASE18
endif ;TSBAUD and ROM2K and (TC2718A or TC2718B)

if TSBAUD and ROM2K and (TIVIO2)
;Reset mode byte flipflop before setting baud rate

    in      TCTRL          ;reset mode byte flipflop

    mov      a,m            ;stop bits, etc,
    out      BRATE1         ;<MR17:MR10>

    mov      a,d            ;baud rate
    out      BRATE0         ;<MR27:MR20>

endif ;TSBAUD and ROM2K and (TIVIO2)

if TSBAUD and ROM2K and T5511
;Stop bits are encoded with the baud rates.

    mov      a,m
    out      BRATE1         ;Divisor high byte
    mov      a,d
    out      BRATE0         ;Divisor low byte
endif ;TSBAUD and ROM2K and T5511

if TSBAUD and ROM2K and TMRCTC
F304 3E04      mvi      a,DWR4          ;point to DART
F306 D3ED      out      TCTRL          ;..write register 4

F308 79          mov      a,c          ;110 baud (c=0)?
F309 B7          ora      a
F30A 3E44      mvi      a,DART1S        ;1 stop bit
F30C C211F3     jnz      TBC1
F30F 3E4C      mvi      a,DART2S        ;110 baud gets 2 stop bits
F311 D3ED      TBC1: out      TCTRL      ;set the DART stop bits

F313 7E          mov      a,m

```

```

MEMON80.PRN
F314 D3E8          out      BRATE1          ;CTC Divisor high byte
                                           ;(must be 1st)
F316 7A            mov      a,d
F317 D3E8          out      BRATE0          ;CTC Divisor low byte
endif ;TSBAUD and ROM2K and TMRCTC

if TSBAUD and ROM2K and T8250
    mov      a,c          ;110 baud (c=0)?
    ora      a
    mvi      a,TDLAB or TSTOP1 ;baud rate access, 1 stop bit
    jnz      TBC1
    mvi      a,TDLAB or TSTOP2 ;110 baud gets 2 stop bits
TBC1: out      TSLCTRL
    mov      e,a          ;temp save

    mov      a,m
    out      BRATE1        ;Baud rate divisor high byte
    mov      a,d
    out      BRATE0        ;Baud rate divisor low byte

    mov      a,e          ;stop pointing to divisor
    xri      TDLAB
    out      TSLCTRL

endif ;TSBAUD and ROM2K and T8250

if TSBAUD and ROM2K

;Print resulting baud rate, using BCD from BTABLE,
;suppressing up to 2 leading zeros, and appending
;a 0. b=0 here (for PAHEX1)

F319 CD11F6        call     CILPRT
F31C 8D            db      CR+80h

F31D 23            inx      h          ;point to the rate BCD

F31E 7E            mov      a,m

F31F 0F            rrc      ;high nibble
F320 0F            rrc
F321 0F            rrc
F322 0F            rrc
F323 E60F          ani      0Fh      ;Leading 0 to suppress?
F325 C46EF6        cnz      PAHEX1    ;n: print it

F328 7E            mov      a,m
F329 B7            ora      a          ;2 leading 0s to suppress?
F32A C46EF6        cnz      PAHEX1    ;(Strips off high nibble)

F32D 23            inx      h          ;low byte
F32E 7E            mov      a,m
F32F CD65F6        call     PAHEX2    ;..always gets printed

F332 CD15F6        call     ILPRNT
F335 3020426175    db      '0 Bau', 'd'+80h

F33B C9            ret
endif ;TSBAUD and ROM2K

;***ROM2K Command Routine Continuation*****
;SE <ADR> <byte1> [<byte2> [<byten>]]
;

```

```

MEMON80.PRN
; Search for string of bytes, starting at <ADR>
; <byten> can be either hex byte or 'text string'
;
;On Entry:
; hl=<ADR>
; Carry set if none entered
; de points to <bytes>
;*****
if ROM2K
F33C E5 GSECMD: push h ;remember <ADR>

;Get search string from input buffer, convert each byte
;to binary, and save result in the RAM buffer
endif ;ROM2K

if RAMHNT and ROM2K
F33D CDC9F2 call RAMPAG ;Find our line buffer
F340 2EB0 mvi l,RAMBUF ;Overwrite with converted data

endif ;RAMHNT and ROM2K

if not RAMHNT and ROM2K
lxi h,RAMBUF
endif ;not RAMHNT and ROM2K

if ROM2K
F342 E5 push h ;binary string address
F343 012700 lxi b,QUOTE ;b=byte count, c=QUOTE

;-----
;Loop to get either a 2-digit hex value
;or a text string (in quotes) each pass
;-----
F346 CDDEF5 SCHLUP: call SSPACE ;returns a=found chr, 0 if none

F349 B9 cmp c ;is 1st chr a quote?
F34A CCAEF3 cz SSTRNG ;y:search for a string
F34D C4BBF3 cnz SCHHEX ;n: search for hex
;returns carry set if end
F350 D246F3 jnc SCHLUP ;loop to get all input

;-----
;Search RAM for the requested string
; b=string length
; top-of-stack=binary string address
; next-on-stack=starting search address
;-----
F353 D1 pop d ;binary string address
F354 E1 pop h ;search start address

F355 78 mov a,b ;anything to search for?
F356 B7 ora a
F357 CAC1F5 jz CMDERR ;error if not

F35A E5 SLOOP1: push h ;search start address
F35B D5 push d ;binary string address

F35C 48 mov c,b ;string byte count

;Loop through all bytes of the requested string
;until either all bytes match or 1st non-matching byte

```



```

MEMON80.PRN
F35D 7A      SLOOP2: mov    a,d
F35E BC      cmp     h      ;don't search our own RAM page
F35F CA99F3  jz      NOMTCH

F362 1A      ldax    d      ;search string
F363 BE      cmp     m      ;current RAM
F364 C299F3  jnz     NOMTCH

F367 23      inx     h      ;test next byte
F368 13      inx     d
F369 0D      dcr     c      ;tested all bytes yet?
F36A C25DF3  jnz     SLOOP2

;String match found. Print address, ask to continue search

F36D D1      pop     d      ;binary string address
F36E E1      pop     h      ;search start address

F36F CD11F6  call    CILPRT
F372 466F756E64 db     'Found', ' '+80h
F378 C5      push    b
F379 CD55F6  call    PHLCHX    ;print match address, trash bc
F37C C1      pop     b

F37D CD11F6  call    CILPRT
F380 4D6F726520 db     'More (Y/N)?', ' '+80h

F38C CD7FF5  call    GETKBD    ;user response
F38F CD80F6  call    PRINTA    ;echo
endif ;ROM2K

if ROM2K and LOWERC
F392 E6DF    ani     ('y'-'Y') XOR 0FFh    ;make it uppercase
endif ;ROM2K and LOWERC

if ROM2K
F394 FE59    cpi     'Y'
F396 C0      rnz                    ;anything but y ends

F397 E5      push    h      ;search start address
F398 D5      push    d      ;binary string address

;Search again, starting at the next byte after hl.
;Quit if we've reached the end of memory, FFFFh

F399 D1      NOMTCH: pop    d      ;binary string address
F39A E1      pop     h      ;search start address

F39B 23      inx     h      ;next RAM
F39C 7C      mov     a,h
F39D B5      ora     l      ;End of memory?
F39E C25AF3  jnz     SLOOP1

F3A1 CD11F6  call    CILPRT
F3A4 4E6F742066 db     'Not foun', 'd'+80h

F3AD C9      ret

;---Local Subroutine-----
;Get a text string from user input at (de),
;store string at (hl), bump count in b
;On Entry:
;  b=byte count

```

# MEMON80.PRN

```

; c=QUOTE
; de points to initial quote
; hl points to destintion for binary
;On Exit:
; b=b+number of string chrs
; de incremented past the string, and past
; the terminating quote, if it's there
; hl incremented for each string chr
; Z set
; Carry clear
;-----
F3AE 13      SSTRNG: inx      d          ;skip over quote

F3AF 1A      STLOOP: ldax    d
F3B0 B7      ora      a          ;end quote is not required
F3B1 C8      rz

F3B2 13      inx      d          ;point past this input chr

F3B3 B9      cmp      c          ;end of string?
F3B4 C8      rz

F3B5 77      mov      m,a        ;store a string byte
F3B6 23      inx      h
F3B7 04      inr      b

F3B8 C3AFF3  jmp      STLOOP      ;get more of this string

;---Local Subroutine-----
;Get one hex value from user input at (de), convert
;it to binary, store it at (hl), bump count in b
;On Entry:
; b=byte count
; de points to next inut chr
; hl points to destintion for binary
;On Exit:
; b=b+1 if hex byte found
; de incremented past the hex byte, if found
; hl incremented once of hex found
; Carry set if no hex digit found
;-----
F3BB CDC4F0  SCHHEX: call    PHFHEX      ;push next string addr byte,
;..and get a hex value.
;hl=0 & carry set if none

F3BE 24      inr      h          ;no high byte allowed
F3BF 25      dcr      h          ;does not change carry
F3C0 C2C1F5  jnz      CMDERR

F3C3 7D      mov      a,l        ;binary value from hex
F3C4 E1      pop      h          ;next string address byte

F3C5 D8      rc                ;carry set means end of input

F3C6 77      mov      m,a        ;store the hex digit
F3C7 23      inx      h
F3C8 04      inr      b

F3C9 C9      ret
endif ;ROM2K

```

;\*\*\*ROM2K Command Routine Continuation\*\*\*\*\*  
 ;MT [<ADR> [<CNT>]] (Test Memory)

```

MEMON80.PRN
;
;On Entry:
; Carry set if no parameters provided
; hl=<ADR>
; de points to <CNT>
;*****
if ROM2K
GMTCMD:
endif ;ROM2K

if RAMCOD and ROM2K ;Running in low memory?
    push    h
    lxi     b,-CODEND ;wipe out Memon/80?
    dad     b          ;16-bit compare
    pop     h
    jnc     CMDERR     ;y: error
endif ;RAMCOD and ROM2K

if ROM2K
F3CA CDC4F0      call    PHFHEX      ;push <ADR>, get hl=<CNT>
                                ;hl=0 if none entered
F3CD CD11F6      call    CILPRT
F3D0 5465737469  db      'Testin','g'+80h
F3D7 EB          xchg
endif ;ROM2K ;de=byte count

if RAMHNT and ROM2K
F3D8 CDC9F2      call    RAMPAG      ;get RAM page
F3DB 7C          mov     a,h         ;a=RAM page
endif ;RAMHNT and ROM2K

if not RAMHNT and ROM2K
    mvi     a,RAMEND/256
endif ;not RAMHNT and ROM2K

if ROM2K
F3DC E1          pop     h           ;hl=start address
F3DD 0165F4      lxi     b,MTPAT     ;Test pattern sequence

;Loop until all memory locations have seen each pattern byte

F3E0 E5          MTLOOP: push    h           ;Start address
F3E1 D5          push    d           ;Byte count
F3E2 C5          Push    b           ;Pattern position
F3E3 F5          push    psw         ;a=RAM page

;-----
;Fill memory with pattern, avoiding the stack. Do
;a read/invert/write twice to stress the memory.
;-----
F3E4 F1          FIL0: pop     psw         ;a=RAM page address
F3E5 F5          push    psw

F3E6 BC          cmp     h           ;On RAM page?
F3E7 C2F0F3      jnz     FIL1

F3EA 7D          mov     a,l         ;y: stack part of RAM page?
F3EB FE8F        cpi     RAMBEG and 0FFh ;Can't test above the stack
F3ED D200F4      jnc     FIL3

```

```

MEMON80.PRN
F3F0 0A      FIL1:  ldax    b          ;Get a pattern byte
F3F1 B7              ora     a          ;Pattern end?
F3F2 C2F8F3      jnz     FIL2
F3F5 0165F4      lxi     b,MTPAT      ;y: restart pattern
F3F8 03      FIL2:  inx     b
                ;High-frequency memory byte test while
                ;we fill memory with the pattern
F3F9 77              mov     m,a        ;Write pattern to memory
F3FA 7E              mov     a,m        ;Invert & write
F3FB 2F              cma
F3FC 77              mov     m,a
F3FD 7E              mov     a,m        ;twice
F3FE 2F              cma
F3FF 77              mov     m,a
F400 23      FIL3:  inx     h          ;next address
F401 1B              dcx     d          ;byte count
F402 7A              mov     a,d        ;end?
F403 B3              ora     e
F404 C2E4F3      jnz     FIL0        ;n: keep filling
F407 F1              pop     psw        ;a=RAM page address
F408 C1              pop     b          ;Pattern position
F409 D1              pop     d          ;Byte count
F40A E1              pop     h          ;Start address
                ;-----
                ;compare memory to the pattern, avoiding the stack
                ;-----
F40B E5              push    h          ;Start address
F40C D5              push    d          ;Byte count
F40D C5              push    b          ;Pattern position
F40E F5              push    psw        ;RAM page address
F40F F1      CML0OP: pop     psw        ;a=RAM page
F410 F5              push    psw
F411 BC              cmp     h          ;On RAM page?
F412 C21BF4      jnz     CML1        ;n: okay
F415 7D              mov     a,l        ;y: stack part of RAM page?
F416 FE8F      cpi     RAMBEG and 0FFh ;Can't test above the stack
F418 D228F4      jnc     CML3
F41B 0A      CML1:  ldax    b          ;Get pattern byte
F41C B7              ora     a          ;Pattern end?
F41D C223F4      jnz     CML2
F420 0165F4      lxi     b,MTPAT      ;y: restart pattern
F423 03      CML2:  inx     b          ;next pattern byte
F424 BE              cmp     m          ;compare pattern to memory
F425 C442F4      cnz     CMPERR      ;report any mismatch

```

```

MEMON80.PRN
F428 23      CML3:  inx      h      ;next RAM location
F429 1B      dcx      d      ;next byte count
F42A 7A      mov      a,d      ;end?
F42B B3      ora      e
F42C C20FF4  jnz      CMLOOP

;-----
;Done with one pass. Print pacifier, test for abort,
;and do another pass, unless we are done.
;-----
F42F CD15F6  call     ILPRNT      ;print pacifier
F432 AE      db      PCFIER+80h
F433 CD8BF5  call     CHKKBD      ;Chance to abort
F436 E1      pop      h      ;h=RAM page address
F437 C1      pop      b      ;Pattern position
F438 D1      pop      d      ;Byte count
F439 03      inx      b      ;rotate pattern once
F43A 0A      ldax     b      ;end of pattern?
F43B B7      ora      a
F43C 7C      mov      a,h      ;restore RAM page address
F43D E1      pop      h      ;Start address
F43E C2E0F3  jnz      MTLOOP
F441 C9      ret              ;to main

;----Local Subroutine-----
;Report memory error, and give
;user an opportunity to abort
;On Entry:
;  a=expected data
;  hl=address
;  (hl)=found data
;-----
F442 C5      CMPERR: push    b      ;PHLADR trashes bc
F443 F5      push    psw
F444 CD48F6  call     PHLADR      ;address:
F447 CD15F6  call     ILPRNT
F44A 57726F7465 db      'wrote',' '+80h
F450 F1      pop      psw      ;expected data
F451 CD5EF6  call     PAHCSM
F454 CD15F6  call     ILPRNT
F457 2C20726561 db      ', read',' '+80h
F45E CD5DF6  call     PMHCSM      ;memory data
F461 C1      pop      b
F462 C37BF5  jmp      CPAUSE      ;Abort or pause from user?

;-----
; Memory Test Pattern Sequence
;-----
F465 00FF55AA33MTPAT: db      000h,0FFh,055h,0AAh,033h,0CCH,0F0h,00Fh
F46D C33C669978 db      0C3h,03CH,066h,099h,078h,001h,0FEh,002h

```

```

                                MEMON80.PRN
F475 FD04FB08F7      db      0FDH,004h,0FBh,008h,0F7h,010h,0EFh,020h
F47D FD40BF807F      db      0FDH,040h,0BFh,080h,07Fh
F482 00              db      00h      ;End of table mark
endif ;ROM2K

```

```

;***Command Routine Continuation*****
;BO (Boot from 8" or 5=1/4" Altair Floppy Disk)
;Automatically detects 8" versus minidisk. Note
;that Altair disks do not contain a boot sector.
;This loader loads the entire operating system.
;*****
if (A88DCD or A88MCD)

```

GBOCMD:

```

;-----
;Wait for user to insert a diskette into the drive 0, and
;then load that drive's head. Do this first so that the disk
;has plenty of time to settle. Note that a minidisk will
;always report that it is ready. Minidisks will hang (later
;on) waiting for sector 0F, until a few seconds after the
;user inserts a disk.
;-----
WAITEN: xra      a      ;boot from disk 0
        out      DENABL ;..so enable disk 0

        call     CHKKBD ;abort from user?

        in       DSTAT  ;Read drive status
        ani      DRVRDY ;Diskette in drive?
        jnz      WAITEN ;no: wait for drive ready

        mvi      a,HEDLOD ;load 8" disk head, or enable
        out      DCTRL   ;..minidisk for 6.4 Sec

;-----
;Step in once, then step out until track 0 is detected.
;The first time through, delay at least 25 ms to force
;a minimum 43 ms step wait instead of 10ms. This meets
;the 8" spec for changing seek direction. (Minidisk
;step time is always 50ms, enforced by the minidisk
;controller hardware.) See the 88-DCDD documentation
;for details. This loop ends with hl=0.
;-----
        lxi      h,(25000/24)*CPUMHZ ;25 ms delay 1st time thru
        mvi      a,STEPIN           ;step in once first

SEEKT0: out      DCTRL           ;issue step command

        inr      l               ;After the 1st time, the
                                ;..following loop goes 1 time.

TODELY: dcx      h               ;(5)
        mov      a,h             ;(5)
        ora      l               ;(4)
        jnz      TODELY          ;(10)24 cycles/pass

WSTEP:  in       DSTAT           ;wait for step to complete
        rrc                      ;put MVHEAD bit in Carry
        rrc                      ;is the servo stable?
        jc       WSTEP           ;no: wait for servo to settle

        ani      TRACK0/4        ;Are we at track 00?
        mvi      a,STEPOT        ;Step-out command

```

```

                                MEMON80.PRN
jnz      SEEKTO                ;no: step out another track

;-----
;Determine if this is an 8" disk or a minidisk, and set
;c to the correct sectors/track for the detected disk.
;an 8" disk has 20h sectors, numbered 0-1Fh. a minidisk
;has 10h sectors, numbered 0-0Fh.
;-----

;wait for the highest minidisk sector, sector number 0Fh

endif ;(A88DCD or A88MCD)
if (A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
    di                    ;disable interrupts
endif ;(A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
if (A88DCD or A88MCD)

CKDSK1: in      DSECTR          ;Read the sector position

        ani      SECMSK+SVALID ;mask sector bits, and hunt
        cpi      (MDSPT-1)*2   ;..for minidisk last sector
        jnz      CKDSK1        ;..only while SVALID is 0

;wait for this sector to pass

CKDSK2: in      DSECTR          ;Read the sector position
        rrc                      ;wait for invalid sector
        jnc      CKDSK2

;wait for and get the next sector number

CKDSK3: in      DSECTR          ;Read the sector position
        rrc                      ;put SVALID in Carry
        jc       CKDSK3         ;wait for sector to be valid

endif ;(A88DCD or A88MCD)
if (A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
    ei                    ;enable interrupts
endif ;(A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
if (A88DCD or A88MCD)

;The next sector after sector 0Fh will be 0 for a minidisk,
;and 10h for an 8" disk. Adding MDSPT (10h) to that value
;will compute c=10h (for minidisks) or c=20h (for 8" disks).

        ani      SECMSK/2      ;mask sector bits
        adi      MDSPT         ;compute SPT
        mov      c,a           ;..and save SPT in c

;-----
;Set up to load
;On Entry:
;  hl=0 (DMA address & execution address)
;  c=SPT (for either minidisk or 8" disk)
;-----

        push     h              ;execution address=0 onto stack
        push     h              ;INIT DMA address
endif ;(A88DCD or A88MCD)

if RAMHNT and (A88DCD or A88MCD)
    dad      sp                ;recover RAM page address

```

```

MEMON80.PRN
    mvi    1,RAMBUF    ;Cleverly aligned buffer
endif ;RAMHNT and (A88DCD or A88MCD)

if (not RAMHNT) and (A88DCD or A88MCD)
    lxi    h,RAMBUF    ;Cleverly aligned buffer
endif ;(not RAMHNT) and (A88DCD or A88MCD)

if (A88DCD or A88MCD)
    xthl                                ;push buffer address, recover
                                        ;DMA address=0

    mov     b,1                        ;initial sector number=0

;-----
;Read current sector over and over, until either the
;checksum is right, or there have been too many retries
; b=current sector number
; c=sectors/track for this kind of disk
; hl=current DMA address
; top-of-stack=buffer address
; next on stack=execution address
;-----
NXTSEC: mvi     a,MAXTRYS                ;(7)Initialize sector retries

;-----
;Begin Sector Read
; a=Remaining retries for this sector
; b=Current sector number
; c=Sectors/track for this kind of disk
; hl=current DMA address
; top-of-stack=RAMBUF address
; next on stack=execution address=0
;-----
RDSCTR: pop     d                        ;(10)get RAMBUF address
        push    d                        ;(11)keep it on the stack
        push    psw                      ;(11)Remaining retry count

endif ;(A88DCD or A88MCD)
if (A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
    di                                ;disable interrupts
endif ;(A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
if (A88DCD or A88MCD)

;-----
;Sector Read Step 1: hunt for sector specified in b.
;Data will become available 250 uS after -SVALID goes
;low. -SVALID is low for 30 uS (nominal).
;-----
FNDSEC: in      DSECTR                    ;(10)Read the sector position

        ani     SECMSK+SVALID            ;(7)yes: mask sector bits
                                        ;..along with -SVALID bit
        rrc     b                        ;(4)sector bits to bits <4:0>
        cmp     b                        ;(4)found the desired sector
                                        ;..with -SVALID low?
        jnz     FNDSEC                  ;(10)no: wait for it

;-----
;Test for DMA address that would overwrite the sector buffer
;or the stack. Do this here, while we have some time.
;-----
        mov     a,h                      ;(5)high byte of DMA address

```



```

MEMON80.PRN
    cmp     d           ;(4)high byte of RAM code addr
    jz      OVMERR      ;(10)report overlay error
;-----
;Set up for the upcoming data move
;Do this here, while we have some time.
;-----
    push    h           ;(11)DMA address for retry
    push    b           ;(11)Current sector & SPT
    lxi     b,BPS       ;(10)b= init checksum,
                        ;c= byte count for MOVLUP
;-----
;Sector Read Step 2: Read sector data into RAMBUF at de.
;RAMBUF is positioned in memory such that e overflows
;exactly at the end of the buffer. Read data becomes
;available 250 us after -SVALID becomes true (0).
;
;This loop must be << 32 us per pass.
;-----
DATLUP: in     DSTAT      ;(10)Read the drive status
        rlc          ;(4)new Read data Available?
        jc     DATLUP    ;(10)no: wait for data

        in     DDATA      ;(10)Read data byte
        stax   d          ;(7)store it in sector buffer
        inr    e          ;(5)Move to next buffer address
                        ;..and test for end
        jnz    DATLUP    ;(10)loop if more data
;-----
;Sector Read Step 3: Move sector data from RAMBUF
;into memory at hl. compute checksum as we go.
;
;8327 cycles for this section
;-----
        mvi     e,SDATA and 0FFh ;(7)de= address of sector data
                        ;..within the sector buffer
MOVLUP: ldax    d          ;(7)get sector buffer byte
        mov     m,a       ;(7)store it at the destination
        cmp     m         ;(7)Did it store correctly?
        jnz     MEMERR    ;(10)no: abort w/ memory error

        add     b         ;(4)update checksum
        mov     b,a       ;(5)save the updated checksum

        inx     d          ;(5)bump sector buffer pointer
        inx     h          ;(5)bump DMA pointer
        dcr     c          ;(5)more data bytes to copy?
        jnz     MOVLUP    ;(10)yes: loop
;-----
;Sector Read Step 4: check marker byte and compare
;computed checksum against sector's checksum. Retry/
;abort if wrong marker byte or checksum mismatch.
;On Entry and Exit:
;    a=computed checksum
;134 cycles for for this section
;-----
        xchg                    ;(4)hl=1st trailer byte address
                                ;de=DMA address
        mov     c,m            ;(7)get marker, should be FFh

```

```

MEMON80.PRN
    inr    c                ;(5)c should be 0 now

    inx    h                ;(5)(hl)=checksum byte
    xra    m                ;(7)compare to computed cksum
    ora    c                ;(4)..and test marker=ff

    pop    b                ;(10)Current sector & SPT
    jnz    BADSEC          ;(10)NZ: checksum error

;Compare next DMA address to the file byte count that came
;from the sector header. Done if DMA address is greater.

    mvi    l,SFSIZE and 0FFh ;(7)hl=address of file size
    mov    a,m              ;(7)low byte
    inx    h                ;(5)point to high byte
    mov    h,m              ;(7)high byte
    mov    l,a              ;(5)hl=SFSIZE

    xchg                     ;(4)put DMA address back in hl
                           ;..and file size into de

    mov    a,l              ;(4)16-bit subtraction
    sub    e                ;(4)
    mov    a,h              ;(5)..throw away the result
    sbb    d                ;(4)..but keep Carry (borrow)

    pop    d                ;(10)chuck old DMA address
    pop    d                ;(10)chuck old retry count

    jnc    FDEXEC          ;(10)done loading if hl >= de

;-----
;Next Sector: the sectors are interleaved by two.
;Read all the even sectors first, then the odd sectors.
;
;44 cycles for the next even or next odd sector
;-----
    lxi    d,NXTSEC        ;(10)for compact jumps
    push   d                ;(10)

    inr    b                ;(5)sector=sector + 2
    inr    b                ;(5)

    mov    a,b              ;(5)even or odd sectors done?
    cmp    c                ;(4)c=SPT
    rc     (5/11)no: go read next sector
                           ;..at NXTSEC

;Total sector-to-sector = 28+8327+134+44=8533 cycles=4266.5 us
;one 8" sector time = 5208 us, so with 2:1 interleave, we will
;make the next sector, no problem.

endif ;(A88DCD or A88MCD)
if (A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
    ei                ;enable interrupts
endif ;(A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
if (A88DCD or A88MCD)

    mvi    b,01h           ;1st odd sector number
    rz                     ;Z: must read odd sectors now
                           ;..at NXTSEC

```

```

MEMON80.PRN
;Next Track: Step in, and read again.
;Don't wait for the head to be ready (-MVHEAD), since we just
;read the entire previous track. Don't need to wait for this
;step-in to complete either, because we will definitely blow
;a revolution going from the track's last sector to sector 0.
;(One revolution takes 167 ms, and one step takes a maximum
;of 40 uS.) Note that NXTRAC will repair the stack.
;-----
        mov     a,b                ;STEPIN happens to be 01h
        out     DCTRL
        dcr     b                  ;start with b=0 for sector 0
        ret                     ;go to NXTSEC
;-----
;Execute successfully loaded code, after disabling
;the floppy drive and disabling the PROM
;On Entry:
;  Top of stack=RAMBUF address
;  Next on stack=execution address
;-----
FDEXEC: mvi     a,DDISBL           ;Disable floppy controller
        out     DENABL
        pop     d                  ;chuck RAMBUF address
                                   ;..to expose exec address

endif ;(A88DCD or A88MCD)
if (A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
        di                     ;disable interrupts
endif ;(A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
if (A88DCD or A88MCD)

        jmp     EXECDP            ;disable PROM and execute code
;---Error Routine-----
;Checksum error: attempt retry if not too many retries
;already. otherwise, abort, reporting the error
;On Entry:
;  Top of stack=address for first byte of the failing sector
;  next on stack=retry count
;-----
BADSEC: mvi     a,HEDLOD           ;Restart Minidisk 6.4 uS timer
        out     DCTRL

endif ;(A88DCD or A88MCD)
if (A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
        ei                     ;enable interrupts
endif ;(A88DCD or A88MCD) and (ENINTS or CRXINT or TRXINT)
if (A88DCD or A88MCD)

        pop     h                ;Restore DMA address
        pop     psw              ;get retry count
        dcr     a                ;Any more retries left?
        jnz     RDSCTR           ;yes: try reading it again

;Irrecoverable error in one sector: too many retries.
;these errors may be either incorrect marker bytes,
;wrong checksums, or a combination of both.

;Fall into CSMERR

```

# MEMON80.PRN

```

endif ;A88DCD or A88MCD

;---Error Routine-----
;Checksum Error (for both HL and BO commands)
;On Entry:
;  if ROM2K: hl=RAM address for first byte of the failed
;    sector, or last address of the Intel Hex block
;-----
F483 CD11F6 CSMERR: call    CILPRT
F486 4373ED      db      'Cs','m'+80h

      if ROM2K
F489 C39BF4      jmp      RPTERR      ;go give details
      endif ;ROM2K

      if not ROM2K
      jmp      CMDERR      ;error handler
      endif ;not ROM2K

;---Error Routine-----
;Memory Overwrite Error: Attempt to overwrite stack
;(for both HL and BO commands)
;On Entry:
;  if ROM2K: hl=offending RAM address
;-----
F48C CD11F6 OVMERR: call    CILPRT
F48F 4164F2      db      'Ad','r'+80h

      if ROM2K
F492 C39BF4      jmp      RPTERR      ;go give details
      endif ;ROM2K

      if not ROM2K
      jmp      CMDERR      ;error handler
      endif ;not ROM2K

      if ROM2K
;---Error Routine-----
;Memory Error: memory readback failed
;On Entry:
;  if ROM2K: hl=offending RAM address
;-----
F495 CD11F6 MEMERR: call    CILPRT
F498 5241CD      db      'RA','M'+80h

;Fall into RPTERR

;---Error Routine-----
;Report An error: turn the disk controller off, report
;the error on the Console and jump to the Console loop.
;On Entry:
;  hl=offending RAM address
;  sp=valid address in RAM page
;-----
RPTERR:

      endif ;ROM2K
      if A88DCD or A88MCD

          mvi    a,DDISBL      ;Disable floppy controller
          out     DENABL

```

```

MEMON80.PRN
endif ;A88DCD or A88MCD
if ROM2K

F49B CD15F6      call    ILPRNT
F49E 206572726F  db      ' error at', ' '+80h
F4A8 CD55F6      call    PHLCHX      ;print hl in hex on Console

F4AB C3C5F5      jmp     CABORT      ;Repair stack, go to MAIN

endif ;ROM2K

;***Command Routine Continuation*****
;BO (Boot from CCS 2422 Floppy Controller)
;*****
if CC2422

F4AE 0E11      GBOCMD: mvi     c,MAXTRYS+1      ;sector retry counter

F4B0 3E21      mvi     a,DCMDMO+(1 shl BOTDSK)
F4B2 D334      out     DCTRL      ;Assume minidisk for restore
F4B4 57        mov     d,a

F4B5 3ED0      mvi     a,SIDE0      ;set side 0
F4B7 D304      out     BCTRL

F4B9 3E0B      mvi     a,RESTOR      ;Restore to track 0
F4BB D330      out     DCMMMD
F4BD 0680      mvi     b,SNORDY      ;(7)The only possible error
F4BF CD23F5      call    EXCCHK      ;(17)test WD1793 status (17)

;Check for 8" disk and change setup if that's what we have

F4C2 DB04      in      BSTAT      ;read BSTAT while on track 0
F4C4 E601      ani     BTRK0      ;1 if 8" disk while on track 0
F4C6 CACFF4      jz     REB01

F4C9 3E10      mvi     a,DCMAXI      ;set up for an 8" disk by
F4CB B2        ora     d      ;..setting the maxidisk bit
F4CC D334      out     DCTRL
F4CE 57        mov     d,a      ;remember DCTRL state

REB01:

;Set up to read the boot sector

F4CF 3E01      RBRTY: mvi     a,BOTSEC      ;Sector to boot
F4D1 D332      out     DSCTR      ;WD1793 sector port

F4D3 218000     lxi     h,BOTADR      ;Put boot sector here

;Read boot sector into RAM at hl

endif ;CC2422
if CC2422 and (ENINTS or CRXINT or TRXINT)
di      ;disable interrupts
endif ;CC2422 and (ENINTS or CRXINT or TRXINT)
if CC2422

F4D6 7A        mov     a,d      ;(13)select
F4D7 F680      ori     DCAUTOW      ;(7)enable auto-wait
F4D9 D334      out     DCTRL      ;(11)output DCTRL value

F4DB 3E86      mvi     a,RDSECT      ;read command
F4DD D330      out     DCMMMD      ;(11)disk command port

```

MEMON80.PRN

```

F4DF DB33      RDSLUP: in      DDATA      ;(10) 32 cycles in-to-in
F4E1 77        mov      m,a          ;(7)
F4E2 2C        inr      l          ;(4)
F4E3 FADFF4     jm      RDSLUP      ;(10)

F4E6 069C      mvi      b,SNORDY+SRNFER+SCR CER+SLOSTD
F4E8 CD23F5     call    EXCCHK      ;test WD1793 status

;Execute the loaded code if it's ok

F4EB CA8000     jz      BOTADR

;-----
;Read fail. Try again if possible.
;Give user feedback about retries.
; c=retry count-down
;-----
F4EE 47        mov      b,a          ;error code

F4EF 3E11      mvi      a,MAXTRYS+1
F4F1 B9        cmp      c          ;first retry?
F4F2 C200F5     jnz      RBR1       ;n: just print dot

F4F5 CD11F6     call    CILPRT      ;Enables ints too, if ENINTS
F4F8 5265747279 db      'Retryin','g'+80h

F500 CD15F6     RBR1: call    ILPRNT
F503 AE        db      '.'+80h

F504 0D        dcr      c          ;too many retries?
F505 C2CFF4     jnz      RBRTRY

;Fall into BOTERR

;---Fail-----
;Fatal error when trying to boot.
;Report, and return to prompt
;On Entry:
; b=error code
;-----
F508 CD11F6     BOTERR: call    CILPRT
F50B 426F6F7420 db      'Boot failed:', ' '+80h

F518 78        mov      a,b          ;error code
F519 CD63F6     call    PCAHEX

F51C CD15F6     call    ILPRNT
F51F E8        db      'h'+80h

F520 C3C5F5     jmp      CABORT

;---Local Subroutine-----
;Check disk controller status
;On Entry:
; b=error mask
;On Exit:
; Z set if no errors
; Z clear if error or timeout
;Trashes a,e,hI
;-----
F523 214006     EXCCHK: lxi      h,400*CPUMHz ;(7)about 2 seconds

```

```

MEMON80.PRN
;(e a little random at first)

F526 2B      ECLLOOP: dcx      h          ;(5)
F527 7D      mov      a,l          ;(5)
F528 B4      ora      h          ;(4)
F529 CA56F5  jz      TIMEOUT      ;(10)controller timeout?

F52C 1D      ECLUP2: dcr      e          ;(5)inner loop: 9984 cycles
F52D CA26F5  jz      ECLLOOP      ;(10)

F530 DB34      in      DFLAG      ;(10)wait for command to end
F532 0F      rrc      ;(4)test DFSE0J
F533 D22CF5  jnc      ECLUP2      ;(10) 39 cycles/pass

;Delay enough that we have at least 56 us.
;We have 94 cycles already in the code.
;(See WD179x Application Notes, November 1980)

F536 1E0C      mvi      e,3*CPUMHz      ;(7)

F538 1D      ELDLP: dcr      e          ;(4)
F539 C238F5  jnz      ELDLP      ;(10) 14 cycles/pass

;check controller status

F53C DB30      in      DSTAT      ;get status
F53E E600      ani      b          ;mask errors
F540 F0      rp          ;msb is drive not ready bit

;Fall into DNRERR

;---Fail-----
;Drive not ready.
;Report, and return to prompt
;-----
F541 CD11F6  DNRERR: call      CILPRT
F544 4472697665 db      'Drive not read','y'+80h
F553 C3C5F5  jmp      CABORT

;---Fail-----
;Controller timeout.
;Report, and return to prompt
;-----
F556 CD11F6  TIMEOUT: call      CILPRT
F559 4644432074 db      'FDC timeou','t'+80h

F564 C3C5F5  jmp      CABORT

endif ;CC2422

;***Command Routine Continuation*****
;BO (Boot from Cromemco 4FDC/16FDC/64FDC Floppy Controller)
;On Entry:
; hl=1st value after the command
; We allow booting from any valid drive because CDOS
; (and probably Cromix) can boot from any drive.
;*****
; if (C4FDC or C16FDC or C64FDC)

GBOCMD: mvi      c,MAXTRYS+1      ;sector retry counter

;Get the specified boot drive, if any, and set up the
;DCTRL value in d

```

# MEMON80.PRN

```

        mov     a,l
        cmp     MAXDRV+1      ;valid drive specified?
        jnc     CMDERR        ;n: bogus

        mvi     a,80h         ;compute drive select

DSLOOP: rlc
        dcr     1
        jp      DSLOOP

        ori     DCMAXI or DCMOTO ;complete select bits
        mov     d,a

RBRTY:  mov     a,d
        out     DCTRL         ;select disk & maxi/mini

        mvi     a,7Fh         ;side 0, slow-seeking, etc.
        out     ACTRL

;Restore to track 0

        mvi     a,RESTOR      ;restore to track 0
        out     DCMMMD

        mvi     b,SNORDY      ;(7)The only possible error
        call    EXCCHK        ;test WD1793 status (17)

;Set up to read boot sector

        mvi     a,BOTSEC      ;Boot sector
        out     DSCTR        ;set sector reg

        lxi     h,BOTADR      ;address to load & run

        mov     a,d           ;select bits
        ori     DCAUTO        ;turn on auto-wait
        out     DCTRL

;Read the sector

endif ;(C4FDC or C16FDC or C64FDC)
if (C4FDC or C16FDC or C64FDC) and (ENINTS or CRXINT or TRXINT)
    di          ;disable interrupts
endif ;(C4FDC or C16FDC or C64FDC) and (ENINTS or CRXINT or TRXINT)
if (C4FDC or C16FDC or C64FDC)

        mvi     a,RDSECT
        out     DCMMMD

;32 cycles=16 uS per byte (no problem for SD track)

DDREAD: in      DDATA         ;(10)get a byte
        mov     m,a          ;(7)store it in RAM
        inr     l             ;(5)
        jm      DDREAD       ;(10)

;check status

        mvi     b,SNORDY+SRNFER+SCR CER+SLOSTD
        call    EXCCHK        ;test WD1793 status

;Execute the loaded code if it's ok

```



# MEMON80.PRN

jz BOTADR

```

;-----
;Read fail. Try again with the other size
;size disk (maxi vs mini) if possible.
;Give user feedback about retries.
; a=error code
; c=retry count-down
; d=select bits with mini/maxi
;-----
        mov     b,a                ;error code

        mvi     a,DCMAXI
        xra     d                  ;toggle maxi/mini
        mov     d,a

        ani     DCMAXI             ;report only when
        jz      RBRTRY             ;..retrying maxidisk

        mvi     a,MAXTRYS+1
        cmp     c                  ;first retry?
        jnz     RBR1              ;n: just print dot

        call    CILPRT
        db      'Retryin','g'+80h

RBR1:    call    ILPRNT
        db      '.'+80h

        dcr     c                  ;too many retries?
        jnz     RBRTRY

;Fall into BOTERR

;---Fail-----
;Fatal error when trying to boot.
;Report, and return to the prompt.
;On Entry:
; b=error code
;-----
BOTERR: call    CILPRT
        db      'Boot failed:', ' '+80h

        mov     a,b                ;error code
        call    PCAHEX

        call    ILPRNT
        db      'h'+80h

        jmp     CABORT

;---Local Subroutine-----
;Check disk controller status
;On Entry:
; b=error mask
;On Exit:
; Z set if no errors
; Z clear and a=error code if error or timeout
;Trashes a,e,h
;-----
EXCCHK: lxi     h,400*CPUMHZ        ;(7)about 2 seconds
                                           ;(e a little random at first)

```

# MEMON80.PRN

```

ECLOOP: dcx      h          ;(5)
        mov      a,l        ;(5)
        ora      h          ;(4)
        jz       TIMEOUT    ;(10)controller timeout?

ECLUP2: dcr      e          ;(5)inner loop: 9984 cycles
        jz       ECLOOP     ;(10)

        in       DFLAG      ;(10)wait for command to end
        rrc      ;(4)test DFSE0J
        jnc      ECLUP2     ;(10) 39 cycles/pass

;Delay enough that we have at least 56 us (224 cycles)
;for a 4 MHz CPU. We have 94 cycles already in the code.
;(See WD179x Application Notes, November 1980)

        mvi      e,10       ;(7)

ELDLP:  dcr      e          ;(4)
        jnz      ELDLP      ;(10) 14 cycles/pass

;Check controller status

        in       DSTAT      ;get status
        ani      b          ;mask errors
        rp       ;msb is drive not ready bit

;Fall into DNRERR

;---Fail-----
;Drive not ready.
;report, and return to Memon/80
;-----
DNRERR: call     CILPRT
        db       'Drive not read','y'+80h

        jmp      CABORT

;---Fail-----
;controller timeout.
;report, and return to Memon/80
;-----
TIMOUT: call     CILPRT
        db       'FDC timeou','t'+80h

        jmp      CABORT

endif ;(C4FDC or C16FDC or C64FDC)

;***Command Routine Continuation*****
;BO (Boot from IMSAI FIF Floppy controller)
;(This code based on ABOOTSIM.ASM in the "IMSAI
;CP/M System User's Guide Version 1.31 Rev.2",
;3/21/77, page CP/M 2 - 18.)
;*****
if IFIF
GBOCMD: mvi      c,MAXTRYS+1 ;retry counter

        mvi      a,ISSPTR    ;Set string pointer 0
        out      IDISK

;Set Floppy Disk Interface string pointer

```

# MEMON80.PRN

```

        lxi    h,IBCMD      ;Point hl at command string
        mov    a,l          ;lo
        out    DISK
        mov    a,h          ;hi
        out    DISK

;Set up string in RAM

        mvi    m,IRDSEC     ;Read sector 0, unit 0
        inx    h            ;Point at status byte
        xra    a            ;Get 0 in a
        mov    m,a          ;Zero high order track
        inx    h
        mov    m,a          ;Zero lo order track
        inx    h
        mvi    m,1          ;Sector 1
        inx    h
        mov    m,a          ;Zero low order buffer address
        inx    h
        mov    m,a          ;Zero high order buf address

;Initialization complete. Now read sector.

RBRTRY: lxi    h,IBSTAT
        xra    a

        mov    m,a          ;Stat must be 0 before command

;N.B. a=0 is disk command IEXEC0 to execute string 0

IDWAIT: out    DISK        ;Tell disk to go
        add    m            ;Look for non-0 status
        jz     IDWAIT       ;Keep looking 'till it comes

        cpi    ISUCCS
        jz     0             ;On success, go to bootstrap
                                ;routine read from disk

;-----
;Read fail. Restore the disk and try again if
;we can. Give user feedback about retries.
; m=error code
; c=retry count-down
;-----
        mvi    a,IESTR      ;Restore the disk
        out    DISK

        mvi    a,MAXTRYS+1
        cmp    c            ;first retry?
        jnz    RBR1         ;n: just print dot

        call   CILPRT
        db     'Retryin','g'+80h

RBR1:   call   ILPRNT
        db     '.'+80h

        dcr    c            ;too many retries?
        jnz    RBRTRY

;Fall into BOTERR

```

```

MEMON80.PRN
;---Fail-----
;Fatal error when trying to boot.
;report, and return to the prompt.
;On Entry:
; m=error code
;-----
BOTERR: call    CILPRT
        db      'Boot failed:', ' '+80h

        mov     a,m                ;error code
        call    PCAHEX

        call    ILPRNT
        db      'h'+80h

        jmp     CABORT
endif ;IFIF

;***Command Routine Continuation*****
;BO (Boot from Micropolis Floppy Controller)
;*****
if MICROP

;Copy Read-Sector subroutine into RAM,
;since DOS expects it there.

GBOCMD: lxi     d,RDSROM            ;source
        lxi     h,RDSEC            ;destination
        mvi     c,RDSLEN          ;byte count

COPYB:  ldax    d
        mov     m,a
        inx     h
        inx     d
        dcr     c
        jnz     COPYB

        lxi     h,FDCMD            ;command register address
        mvi     m,FRESET          ;reset FDC

        mvi     b,MAXTRYS+1       ;b=max read retries

;-----
;Find and load track 0, sector 0, using
;the FDC we just found.
;On Entry:
; b=retry counter
; c=0
; hl=command register address
;-----

;select drive 0 and give the drive time to spin up.
;The drive may not be spinning if the FDC has been
;modified with the motor-off mod. DELSTA will read
;the FDC's status register every pass, resetting
;the motor-off mod's 4-second timer.

        mvi     m,SLUN+0          ;select drive zero

        mvi     d,85              ;delay for 85*5.89=500 ms
        call    DELSTA            ;returns a=status reg value
                                   ;returns de=0

```

```

                                MEMON80.PRN
RETRY:  dcr      b                ;too many read retries?
        jz      RDERR            ;y: read error
        push    b                ;b=retries

;Verify drive is ready. If the controller board has been
;modified with the motor-off mod, then its RDY line is
;always true. So rather than check the drive ready bit,
;look at the sector register to see that it is changing,
;indicating that a disk is spinning. (We don't care if the
;sector is valid - just that it is changing.)

        mov     c,m              ;c=1st check of sector register

        mvi     d,17             ;delay for 17*5.89=100.3 ms
        call    DELSTA           ;..1/2 revolution

        mov     a,m              ;a=2nd check of sector register
        xra     c                ;did sector register change?
                                ;any bits changing in sect reg
                                ;..indicate a spinning disk.

        jz      DNRERR           ;same: drive not ready

;Home the head: Step in to get off track 0, and then seek to
;track zero. The number of step-ins is equal to the remaining
;retry counter, meaning it will try more times to get off the
;track 0 sensor on early retries than on late retries. In
;fact, it should always take just one step to get off the
;track 0 sensor. If we happen to start out on the last track,
;then we will do one more step out, which is okay.
;hl=FDC command register address for HSTEP
;b=retries here
;e=0 here

;step away from track 0

        lxi     b,900h+STEP+STEP1 ;b=8 steps, c=step in
UNSTUK: call    HSTEP            ;exits with de=0,
                                ;..a=status reg value
        ani     TK0             ;still at track zero?
        jnz     UNSTUK          ;y: try stepping away again

;seek track 0

        lxi     b,(90*256)+STEP+STEP0 ;b=90 steps max,
                                ;..c=step out
SEEK0:  call    HSTEP            ;exits with de=0,
                                ;..a=status reg value
        ani     TK0             ;at track zero yet?
                                ;returns with de=0
        jz      SEEK0           ;n: step out again

;Read sector zero into a RAM buffer (MICBUF)
;d=0
;hl=FDC command register address for RDSECH

        mvi     e,MICBUF         ;de=RAM buffer addr in page 0
        call    READ0           ;hl=control register
                                ;Z set if ok

;Unless the 1st read failed, get the load address for

```

```

MEMON80.PRN
;this sector from the sector data and re-read the same
;sector into the requested address

        xchg          ;de=FDC address
        lhl          MICBUF+PTROFF ;requested load address
        xchg
        cz          READ0          ;hl=control register

        pop          b          ;b=retry counter, c=0
        jnz          RETRY        ;failed, start over

;Successful load. Create stack for DOS, save addresses
;DOS uses (Note: we can't create DOS's stack until we
;are sure we will not return to MAIN.)

        lxi          sp,LODADR+2 ;create stack for DOS
        push         d          ;LODADR gets load address
        push         h          ;FDCADR gets FDC address
        dcr          h          ;compute ROM address
        dcr          h          ;200h below FDC reg address
        push         h          ;ROMADR

;Go execute the loaded code, with registers and the sp
;where the onboard ROMs would have left them.

        xchg          ;hl=LODADR
        pop          d          ;de=ROMADR (for DOS)

        lxi          b,EXEOFF    ;execute 12 bytes into sector
        dad          b          ;hl=execution address

; sp=ROMADR+2=00A2
; de=address of onboard ROM
; hl=code exec address (12-bytes into sector data in RAM)
;
; 00a0: ROMADR=address of onboard ROM
; 00A2: FDCADR=address of FDC registers
; 00A4: LODADR=address beginning of sector in RAM

        endif ;MICROP
        if MICROP and (ENINTS or CRXINT or TRXINT)
            di          ;disable interrupts
        endif ;MICROP and (ENINTS or CRXINT or TRXINT)
        if MICROP

            pchl          ;execute boot loader from disk

;---Local Subroutine-----
;Read and validate sector 0
;On Entry:
;    hl=FDC data register address
;    de=target address
;On Exit:
;    a=0 and Z set = read successful
;    a<>0 and Z clear = read fail (checksum or wrong sector)
;    c=0
;Trashes b
;-----
READ0:   push         d          ;buffer address

        lxi          b,SCLen/2 ;b=sector 0, c=bytes to read/2
        call         RDSECH    ;hl=FDC command reg
                                ;read the sector into (de)

```

```

MEMON80.PRN
;returns a<>0 if read error
;returns c=0

    pop     h                ;buffer address
    mov     e,c              ;de=FDC address (c=0)

;Verify metadata from the loaded sector says track 0, sector 0
;(a=0 if RDSECH thought the read was successful)

    ora     m                ;a=track number from disk data
    inx     h                ;(hl)=sec number from disk data
    ora     m                ;verify track and sec are zero
    dcx     h                ;hl=buffer address

    xchg
    ret                    ;put pointers back

;---Local Subroutine-----
;Step head Once
;On Entry:
;  b=retry counter
;  c=command (Step in or step out)
;  e=0
;  hl=command register address
;On Exit:
;  Abort (rudely) if b decremetned to 0
;  a=status register value
;  b decremented
;  de=0
;-----
HSTEP:  dcr     b
        jz      T0FAIL      ;Can't find track 0
                                ;..or get off track 0

        mov     m,c          ;Step in or out now

;delay for >40 mS

        mvi     d,7          ;delay for 7*5.89=41.2 mS

;Fall into DELSTA to delay and read the status register

;---Local Subroutine-----
;Delay 5.89 mS times value specified in d, and return
;the FDC status register value. Looks at 2mhz/4mhz
;jumper on the FDC to determine CPU speed.
;
;On Entry:
;  d=delay value in 5.89 mS units (max value=127)
;  e=0 (low byte of delay counter, really)
;  hl=address of FDC control register
;On Exit:
;  a=status register value
;  de=0
;-----
DELSTA:  mov     a,m          ;a=sector register from FDC
        ani     DTMR         ;see if 4mhz jumper in place
        mov     a,d          ;a=requested delay
        jnz     NOT4M        ;not 4mhz
        rlc                 ;double the count for 4mhz CPU

NOT4M:   mov     d,a          ;save adjusted delay count

```

```

MEMON80.PRN
;we re-read the status register every pass for delay
DLOOP:  dcx      d                ;(5)
        mov      a,e              ;(5)
        ora      d                ;(4)
        inc      h                ;(5) point to status register
        mov      a,m              ;(7) read status register
        dcx      h                ;(5) point to control register
        rz                     ;(5 not taken)
        jmp      DLOOP            ;(10) 46 cycles/pass

```

```

;---Error Routines-----
;These all repair the stack
;and return to the prompt.
;-----

```

```

TFAIL:  call     CILPRT            ;stuck or lost
        db       'Track ', '0'+80h
        jmp      PFAIL            ;adds ' error' and aborts

RDERR:  call     CILPRT            ;too many retries on reading
        db       'Rea', 'd'+80h
        jmp      PFAIL            ;adds ' error' and aborts

NCERR:  call     CILPRT            ;can't find FDC
        db       'FD', 'C'+80h

```

```

;Fall into PFAIL

```

```

PFAIL:  call     ILPRNT
        db       ' fai', 'l'+80h
        jmp      CABORT

DNRERR: call     CILPRT            ;no disk probably
        db       'Drive not read', 'y'+80h
        jmp      CABORT

```

```

;---RAM Subroutine-----
;Read sector into RAM
;This code gets moved into RAM at RDSEC
;
;On Entry at RDSEC:
; de=RAM buffer
; b=sector to read
; c=number of bytes to read/2
;On Entry at RDSEC+3:
; hl=FDC control/sector register address
; de=RAM buffer
; b=sector to read
; c=number of bytes to read/2
;On Exit:
; a=0 and Z set = read successful
; a<>0 and Z clear = checksum error
; c=0
; de=FDC data register address
; hl=next RAM address past this sector's data
;Trashes b
;-----

```

```

RDSROM: lhld     FDCADR            ;FDC registers
                                           ;(sector register)

wtSec:  mov      a,m                ;sector status
        ani      SFLG              ;wait for sector true

```



```

                                MEMON80.PRN
jz      wtSec-OFFSET

mov     a,m                    ;re-read to be safe
ani     SMASK                  ;get sector number alone
xra     b                      ;right sector number?
jnz     wtSec-OFFSET          ;no, keep looking

inx     h                      ;hl=status register address

endif ;MICROP
if MICROP and (ENINTS or CRXINT or TRXINT)
    di                        ;disable interrupts
endif ;MICROP and (ENINTS or CRXINT or TRXINT)
if MICROP

wtXfer: ora     m              ;wait for transfer flag (bit 7)
        jp     wtXfer-OFFSET ;(10)

        inx     h              ;(5)hl=data register address
        mov     a,m            ;(7)read & chuck sync byte
        xra     a              ;(4)start with carry clear
        xchg                    ;(4)de=data register address
                                ;hl=target RAM address

        mvi     b,0            ;(7)initialize checksum
        nop                    ;(4)timing for FDC stall logic
        nop                    ;(4)

;Read sector data from disk into RAM two bytes at a time
;de=FDC data register
;hl=destination RAM address

movSec: ldax    d              ;read data byte
        mov     m,a            ;save in buffer
        inx     h
        adc     b              ;update checksum
        mov     b,a

        ldax    d              ;repeat for 2nd byte
        mov     m,a
        inx     h
        adc     b
        mov     b,a

        dcr     c              ;decrement byte count
        jnz     movSec-OFFSET ;repeat until done
endif ;MICROP
if MICROP and (ENINTS or CRXINT or TRXINT)
    ei                        ;enable interrupts
endif ;MICROP and (ENINTS or CRXINT or TRXINT)
if MICROP

;Read and verify checksum

        ldax    d              ;a=checksum from disk
        xra     b              ;compare to computed checksum
        ret                    ;return result

RDSEND equ     $
RDSLEN equ     RDSEND-RDSROM
OFFSET equ     RDSROM-RDSEC ;address offsets
MICBUF equ     RDSEND-OFFSET ;Micropolis Sec RAM buffer

```

```
endif ;MICROP
```

```
***Command Routine Continuation*****
;BO (Boot from Northstar Double-Density Floppy Disk drive,
;   with either a single- or double-density disk)
;*****
if NSTARD
```

```
GBOCMD: lxi      b,0D00h+MAXTRYS ;Initialize c=retry counter
;and b=index hunt counter
```

```
-----
;Start the spindle motors, select drive 1, and delay enough
;time for the motor to spin up and the head to settle
;-----
```

```
    lda      CTLCMD+CTLAS+CTLM0 ;Motor on

    lxi      d,3008h            ;d=48 secs: motor spin-up delay
;e=8: step-ins
    call     WSECTD

    lxi      h,CTLORD+ORDSIN+ORDDS1 ;select drive 1
;..(step in mode)
    mov      a,m
```

```
-----
;Wait for the index mark to ensure a disk is installed.
;(The MDS-D will generate fake sector pulses if not.)
;b=0DH here, for 12 sectors to try
;h1=CTLORD+ORDSIN+ORDDS
;-----
```

```
IWLOOP: dcr      b              ;Too many sectors?
        jz       INFERR        ;y: no index hole

        call     WSECT1        ;wait for next sector

        lda      CTLCMD+ORDST   ;Get A-Status
        ani      SAIX          ;Index hole?
        jz       IWLOOP        ;n: keep looking
```

```
-----
;Home: Seek track 0
;e=8 here
;h1=CTLORD+ORDSIN+ORDDS
;-----
```

```
;If already on track 0, step away.
```

```
HOME:   lda      CTLCMD+CTLBS   ;Read B-Status
        rar      ;test SBT0: On track 0?

        cnc      STEP

        jnc      T0STUK        ;Stuck on track 0?
```

```
;Step out to track 0
;h=E8h here
```

```
        mvi      l,ORDDS1      ;step out
        mov      e,h           ;max number of tries
        call     STEP          ;Step out to track 0
        jc       T0FAIL        ;can't find track 0
```

```
-----
```

```

                                MEMON80.PRN
;Hunt: wait for the double-density boot sector
;h=E8h here
;-----
                mov     b,h                ;long timeout for several loops
HUNT:          dcr     b                    ;timeout?
                jz      SNFERR
                call    WSECT1              ;wait for next sector
                lda     CTLCMD+CTLCS+CTLMO    ;Read C-Status
                ani     SCSM                  ;Sector mask
                cpi     DDBSCR                ;Found our boot sector?
                jnz     HUNT                  ;N: Keep looking

;wait for hardware to say that read is enabled
;(timer b should be between DBh and E8h - huge)
WAITRD:        dcr     b                    ;(5)timeout?
                jz      SNFERR              ;(10)
                lda     CTLCMD+CTLAS         ;(13)Read A-Status
                ani     SARE                 ;(7)Read enabled?
                jz      WAITRD              ;(10)n: wait more

;Stall for about 68 uS on a 2 MHz 8080
;or about 34 uS on a 4 MHz Z80
                mvi     e,9
STAL68:         dcr     e                    ;(5)
                jnz     STAL68              ;(10)end with e=0

;-----
;Check for single-density disk, which
;loads from a different sector
;e=0 here
;h=E8h here
;-----
                lda     CTLCMD+CTLAS         ;Read A-Status
                ani     SADD                 ;Double-density disk?
                jnz     GETSEC               ;y: go read the sector

;-----
;Single-density boot
;Step in to track 1 and hunt for sector 8
;(Sector 8 is a special 512-byte boot sector.)
;-----
                inr     e                    ;e=1
                mvi     l,ORDSIN+ORDD51     ;Step in once to track 1
                call    STEP
SDRSCT:         call    WSECT1              ;wait for 1 sector time
                lda     CTLCMD+CTLCS+CTLMO    ;Read C-Status
                ani     SCSM                  ;Sector mask
                cpi     SDBSCR                ;single-density boot sector
                jnz     SDRSCT

;Fall into GETSEC

;-----
;wait for the sector body.
;Time out after a ridiculously long time.

```

```

MEMON80.PRN
;b still has a large timeout value here
;h=E8h here
;-----
GETSEC: lxi      d,CTLCMD+CTLRD  ;Select Read Data

      endif ;NSTARD
      if NSTARD and (ENINTS or CRXINT or TRXINT)
          di          ;disable interrupts
      endif ;NSTARD and (ENINTS or CRXINT or TRXINT)
      if NSTARD

WATSEC: dcr      b              ;(5)timeout?
        jz        NSERR        ;(10)

        lda      CTLCMD+CTLAS  ;(7)Read A-Status
        rrc      ;(4)SABD set?
        ;..(Sync chr detected)
        jnc      WATSEC        ;(10)n: keep waiting

;-----
;Read and validate the sector data
;  Byte 0: <PA> High byte of target RAM address
;  Bytes 1-511: Sector data
;  Byte 512: CRC of Bytes 0-511
;Bytes1-511 get written into RAM at <PA>01 through <PA+1>FF.
; <PA> gets written to <PA>01 before getting overwritten by
; byte 1. Note that <PA>00 never gets written
;-----
        ldax     d              ;read 1st disk data byte
        mov      h,a           ;This is the target RAM page

        mvi      l,1           ;byte 1 in that page
        mov      m,a           ;save page address (why?)

        rlc          ;Start CRC calculation
        mov      b,a           ;b accumulates CRC

;Read the next 255 bytes (the remainder of the data that goes
;into the 1st RAM page) into the specified RAM page

RLOOP1: ldax     d              ;get a disk data byte
        mov      m,a           ;write it to RAM
        xra      b             ;compute CRC
        rlc
        mov      b,a           ;..in b
        inr      l             ;end of page data?
        jnz      RLOOP1        ;n: get another byte

;Read the final 256 bytes into the next RAM page

        inr      h             ;Next RAM page

RLOOP2: ldax     d              ;get a disk data byte
        mov      m,a           ;write it to RAM
        xra      b             ;compute CRC
        rlc
        mov      b,a           ;..in b
        inr      l             ;end of sector data?
        jnz      RLOOP2        ;n: keep reading

        ldax     d              ;Get CRC byte
        xra      b             ;Does it match?
        jnz      NSERR        ;n: retry if possible

```

# MEMON80.PRN

```

;-----
;Success. Go execute the loaded code image
;Like the MDS-D boot ROM, d contains the
;high byte of the disk controller address,
;in case the loaded code looks for this.
;-----
        dcr     h           ;get execution address
        mvi     1,0ah       ;10th byte of load page

        pchl                    ;Go execute loaded code

;---Local Subroutine-----
;Wait 1 sector time (for next sector)
;On Exit:
;   a=0
;   d=0
;   Sector flag is cleared
;-----
WSECT1: mvi     d,1

;Fall into WSECTD

;---Local Subroutine-----
;Wait d sector times
;On Entry:
;   d=number of sector times to wait
;On Exit:
;   a=0
;   d=0
;   Sector flag is cleared
;-----
WSECTD: lda     CTLCMD+CTLAS+CTLRSF ;Reset sector flag

WSECTR: lda     CTLCMD+CTLAS         ;Read A-Status
        ora     a                   ;Test SASF (Sector flag)
                                   ;clears carry too
        jp      WSECTR              ;wait for sector

        dcr     d                   ;next sector
        lda     CTLCMD+CTLAS+CTLRSF ;Reset sector flag
        jnz     WSECTD              ;more sectors to wait?

        ret

;---Local Subroutine-----
;Step e tracks
;On Entry:
;   e=number of steps
;   hl=step-in or step-out command
;On Exit:
;   Carry set if on track 0
;Trashes a,de,l
;-----
STEP:   mov     a,m                 ;issue command (Step level low)
        push    h                   ;remember original command

        mov     a,l                 ;set step level high
        ori     ORDST
        mov     l,a
        mov     a,m                 ;Step level high

        pop     h                   ;original command

```

```

                                MEMON80.PRN
mov     a,m                     ;step level low again
mvi     d,2                     ;wait for 2 sector times
call    WSECTD
lda     CTLCMD+CTLBS            ;Read B-Status
rar                                           ;test SBT0: On track 0?
rc                                           ;y: done with Carry set

dcr     e
jnz     STEP

ret                               ;with carry clear

;---Error Routines-----
;These all repair the stack
;and return to the prompt.
;-----
NSERR:  dcr     c                ;More retries left?
jnz     HOME                   ;y: try again

SNFERR: call    CILPRT
db      'Secto','r'+80h
jmp     NFERR                   ;recycle some code

INFERR: call    CILPRT
db      'Inde','x'+80h
jmp     NFERR                   ;recycle some code

T0STUK:T0FAIL: call    CILPRT
db      'Track ','0'+80h

;Fall into NFERR

NFERR:  call    ILPRNT
db      ' erro','r'+80h
jmp     CABORT

endif ;NSTARD

;***Command Routine Continuation*****
;BO (Boot from Northstar Single-Density Floppy Disk)
;*****
if NSTARS

GBOCMD: mvi     b,MAXTRYS        ;INit retry count (10)

;(Re)load the boot sector
;b=remaining retries. Stack is valid
;turn on the drive motor and select the boot drive

RETRY:  lda     CTLMO or CTLNOP  ;Motors on & a status
ani     SAMO                    ;Motor already on? (retry?)
jnz     SELECT                  ;Yes, no need to wait

mvi     d,32h                   ;wait 32 sector times
call    WSECTS                  ;..for motor to spin up

SELECT: lda     CTLDS0 or BDRIVE ;Select boot drive
mvi     d,0DH                   ;Wait 13 sector tiems
call    WSECTS                  ;..for head to settle

;Step in a few tracks in case we are lost

```

# MEMON80.PRN

```

        mvi        e,17                ;max step-ins+1
STIN:   dcr        e                    ;
        jz         T0STUK              ;track 0 sensor stuck?

        lxi        h,CTLSTI            ;step-in command
        mvi        c,1                 ;step 1 track inward
        call       STEP                ;
        jnz        STIN                ;still on track 0?

;Step out to track 0

        lxi        h,CTLSTO            ;step-out command
        mvi        c,59h               ;max 59 steps outward
        call       STEP                ;
        jz         T0FAIL              ;Can't find track 0

;-----
;Hunt for the boot sector
;-----
HUNT:   mvi        d,1                 ;Wait one sector time
        call       WSECTS

        lda        CTLBST or CTLNOP    ;Read B-status
        ani        SBSECT              ;Get sector number
        cpi        BSECTR              ;Desired sector?
        jnz        HUNT                ;N: wait for it

;-----
;Set up for load
;-----
        lxi        h,LODADR            ;RAM load address
        mvi        c,8dh               ;Timeout count

;-----
;Issue read command, and wait for the boot sector's data
;-----
        lxi        d,CTLRD or CTLNOP  ;Read data command

WATSEC: dcr        c                    ;timeout?
        jz         SNFERR

        lda        CTLNOP              ;Get A-status
        ani        SABDY              ;Wait for Body
        jz         WATSEC              ;n: keep waiting

endif ;NSTARS
if NSTARS and (ENINTS or CRXINT or TRXINT)
        di                    ;disable interrupts
endif ;NSTARS and (ENINTS or CRXINT or TRXINT)
if NSTARS
;-----
;Read 256-byte sector data into RAM
;This loop assumes the load address
;low byte is 00.
;-----
        mov        c,l                ;initial CRC (l=0)

RDLOOP: ldax       d                    ;Read data byte
        mov        m,a                ;Write to RAM

        xra        c                  ;Compute CRC

```

```

                                MEMON80.PRN
                                rlc
                                mov     c,a

                                inr     l           ;Next RAM address
                                jnz     RDLOOP       ;Read them all

                                ldax    d           ;Read CRC
                                xra     c           ;Does it match?
                                jnz     CRCERR       ;n: CRC error

;-----
;Execute the loaded code
;-----
                                jmp     EXEADR

;-----
;CRC fail: retry sector if we can
;-----
CRCERR: dcr     b           ;any retries left?
                                jnz     RETRY

;Too many retries

                                call    CILPRT
                                db      'CR','C'+80h
                                jmp     RPTERR

;---Local Subroutine-----
;Step specified number of tracks
;On Entry:
;  c=number of tracks to step
;  hl=step in or step out command
;On Exit:
;  Z clear if on track 0
;trashes a,c,d
;-----
STEP:  mov     a,m           ;Set direction

STPLUP: lda     CTLSTS       ;Set track-step flip flop
                                xthl
                                xthl
                                lda     CTLSTC       ;Clear track-step flip flop

                                mvi     d,2           ;wait 2 sector times
                                call    WSECTS

                                lda     CTLNOP       ;Read A-status
                                ani     SATR0        ;At track 0?
                                jz      STEP1        ;No, keep stepping

                                ret                 ;Z clear: On track 0

STEP1: dcr     c           ;More steps to do?
                                jnz     STPLUP       ;n: step again

                                ret                 ;Z set: not on track 0

;---Local Subroutine-----
;wait d sector times
;On Entry:
;  d=number of sector times to wait
;Trashes psw,d
;-----

```



```

MEMON80.PRN
WSECTS: lda      CTLRSF      ;Reset sector flag

WSECT:  lda      CTLMO or CTLNOP ;Read A-status
        ani      SASF        ;Sector pulse?
        jz       WSECT      ;Wait for it
        dcr      d           ;enough sectors?
        rz              ;y: done

        jmp      WSECTS      ;n: Keep waiting

;---Error Routines-----
;These all repair the stack
;and return to the prompt.
;-----
TOSTUK:
T0FAIL: call     CILPRT
        db       'Track ', '0'+80h
        jmp      NFERR

SNFERR: call     CILPRT
        db       'Secto', 'r'+80h

;Fall into NFERR

NFERR:  call     ILPRNT
        db       ' not foun', 'd'+80h
        jmp      CABORT

endif ;NSTARS

;***Command Routine Continuation*****
;BO (Boot from SD Systems Versafloppy disk or
;Versafloppy II disk, either minidisk or 8")
;*****
if VERSA1 or VERSA2 or SALFDC

GBOCMD: lxi      b,4000h+MAXTRYS ;b=home error code
                                           ;c=Retry counter

        endif ;VERSA1 or VERSA2 or SALFDC
        if VERSA1

                mvi      a,(VDSEL0N + VINTEN) xor 0FFh ;select drive 0

        endif ;VERSA1
        if VERSA2 or SALFDC

                mvi      a,VDSEL0 + VMINI ;select drive 0,
                                           ;..assume mini for now

        endif ;VERSA2 or SALFDC
        if VERSA1 or VERSA2 or SALFDC

                out      VDRSEL

                mvi      a,RESTOR      ;Issue HOME command, load head
                out      VDCOM

        WDELAY: mvi      a,4*CPUMHZ    ;delay for 52 us min
                dcr      a              ;(5)
                jnz      WDELAY        ;(10)

        HOME1: in       VDSTAT      ;wait for restore to complete

```

```

                                MEMON80.PRN
    mov     d,a
    rrc
    jc      HOME1                ;test WD1793 busy bit

    mvi     a,SNORDY             ;drive ready?
    ana     d
    jnz     DNRERR               ;Fail if not ready

    mvi     a,STRAK0
    ana     d
    jz      T0FAIL               ;Error if can't find home

;Assume successful home without checking
;other error bits. Load at address 0.
TRETRY: lxi     h,VLOAD          ;load address

    mvi     a,1                  ;sector 0
    out     VSECT

;Enable auto-wait circuit

endif ; VERSA1 or VERSA2 or SALFDC
if VERSA1

    mvi     a,(VDSEL0N + VINTEN + VWAITN) xor 0FFh

endif ;VERSA1
if VERSA2 or SALFDC

    in      VDRSEL               ;get drive sel, mini/maxi
    ori     VWAIT

endif ;VERSA2 or SALFDC
if (VERSA1 or VERSA2 or SALFDC)

    out     VDRSEL               ;enable auto-wait

endif ;(VERSA1 or VERSA2 or SALFDC)
if (VERSA1 or VERSA2 or SALFDC) and (ENINTS or CRXINT or TRXINT)
    di                      ;disable interrupts
endif ;(VERSA1 or VERSA2 or SALFDC) and (ENINTS or CRXINT or
TRXINT)
if (VERSA1 or VERSA2 or SALFDC)

    mvi     a,RDSECT             ;Read sector command
    out     VDCOM

;Loop to read the sector data into memory at hl

    mvi     b,SECSIZ            ;byte count

RLOOP: in      VDDATA            ;read the data
    mov     m,a                  ;write it into memory
    inc     h                    ;Next memory address
    dcr     b
    jnz     RLOOP

;Turn off auto-wait

endif ;VERSA1 or VERSA2 or SALFDC
if VERSA1

```

```

                                MEMON80.PRN
        mvi        a,(VDSEL0N + VINTEN) xor 0FFh

endif ;VERSA1
if VERSA2 or SALFDC

        in         VDRSEL                ;get drive sel, mini/maxi
        ani        VWAIT xor 0FFh

endif ;VERSA2 or SALFDC
if VERSA1 or VERSA2 or SALFDC

        out        VDRSEL                ;disable auto-wait

;Wait for command to complete

RWAIT:  in         VDSTAT                ;Disk status
        mov        b,a
        rrc                     ;test WD1793 busy bit
        jc         RWAIT

;Check status for any errors

        mvi        a,SNORDY+SRNFER+SCRCER+SLOSTD
        ana        b                ;any errors?
        jz         VBOOT                ;N: go execute loaded code

;-----
;Read fail. Try again if possible.
;Give user feedback about retries.
; c=retry count-down
;-----
        mov        b,a                ;error code

endif ;VERSA1 or VERSA2 or SALFDC
if VERSA2 or SALFDC

;If we tried a minidisk, then try 8" instead
        in         VDRSEL
        xri        VMINI
        out        VDRSEL

        ani        VMINI                ;Was it mini?
        jz         RBRTRY

endif ;VERSA2 or SALFDC
if VERSA1 or VERSA2 or SALFDC

        mvi        a,MAXTRYS+1
        cmp        c                ;first retry?
        jnz        RBR1                ;n: just print dot

        call       CILPRT
        db         'Retryin','g'+80h

RBR1:   call       ILPRNT
        db         '.'+80h

        dcr        c                ;too many retries?
        jnz        RBRTRY

;Fall into BOTERR

;---Error Routines-----

```

```

MEMON80.PRN
;These all repair the stack
;and return to the prompt.
;On Entry:
;  b=error code
;-----
BOTERR: call    CILPRT
        db      'Boot failed:', ' '+80h

        mov     a,b                ;error code
        call    PCAHEX

        call    ILPRNT
        db      'h'+80h

        jmp     CABORT

TOFAIL: call    CILPRT
        db      'Track 0 fai','l'+80h
        jmp     CABORT

DNRERR: call    CILPRT
        db      'Drive not read','y'+80h
        jmp     CABORT

endif ;VERSA1 or VERSA2 or SALFDC

;***Command Routine Continuation*****
;BO (Boot from Tarbell Floppy Disk)
;*****
if TARBL1
GBOCMD: mvi     a,TSEL0            ;select drive 0
        out     TEXTP
endif ;TARBL1

if TARBL2
GBOCMD: xra     a
        out     TEXTP0            ;select drive 0
        out     TESTP1            ;Memory bank 0
endif ;TARBL2

if (TARBL1 or TARBL2)
        lxi     b,4000h+MAXTRYS ;b=home error code
                                ;c=Retry counter

TRETRY: mvi     a,FINTCM           ;Cancel any pending command
        out     TDCOM

WDELAY: mvi     a,4*CPUMHZ         ;delay for 52 us min
        dcr     a                  ;(5)
        jnz     WDELAY             ;(10)

HOME1:  in      TDSTAT              ;wait for cancel to complete
        rrc                      ;test WD1793 busy bit
        jc      HOME1

        mvi     a,RESTOR           ;Issue restore command
        out     TDCOM

        in      TWAIT              ;wait for restore to complete

        ora     a                  ;Test for success
        jm      DNRERR             ;Fail if not ready

```

```

                                MEMON80.PRN
    in      TDSTAT              ;controller status
    ani     STRAK0              ;b is home error code too
    jz      TOfAIL              ;Error if can't find home

;Assume successful home without checking
;other error bits. Load at address 0.

    lxi     h,SLOAD            ;Load address

    mvi     a,1                 ;Load sector 1
    out     TSECT

    mvi     a,RDSECT           ;Read sector command
    out     TDCOM

;Loop to read the sector data into memory at hl

    endif ;(TARBL1 or TARBL2)
    if (TARBL1 or TARBL2) and (ENINTS or CRXINT or TRXINT)
        di                     ;disable interrupts
    endif ;(TARBL1 or TARBL2) and (ENINTS or CRXINT or TRXINT)
    if (TARBL1 or TARBL2)

RLOOP:  in      TWAIT           ;(10)wait for DRQ or INTRQ

        ora     a               ;(4)
        jp      RDONE           ;(10)Done if INTRQ

        in      TDDATA          ;(10)read the data
        mov     m,a             ;(7)write it into memory
        inx     h               ;(5)Next memory address
        jmp     RLOOP           ;(10) 56 cycles=28 us

;Check status for any errors

RDONE:  in      TDSTAT          ;Disk status
        ani     SNORDY+SRNFER+SCRCER+SLOSTD
        jz      SBOOT          ;N: go execute loaded code

;-----
;Read fail. Try again if possible.
;Give user feedback about retries.
; c=retry count-down
;-----
        mov     b,a             ;error code

        mvi     a,MAXTRYS+1
        cmp     c               ;first retry?
        jnz     RBR1           ;n: just print dot

        call    CILPRT
        db      'Retryin','g'+80h

RBR1:   call    ILPRNT
        db      '.'+80h

        dcr     c               ;too many retries?
        jnz     RBRTRY

;Fall into BOTERR

;---Error Routines-----
;These all repair the stack

```

```

MEMON80.PRN
;and return to the prompt.
;On Entry:
;  b=error code
;-----
BOTERR: call    CILPRT
        db      'Boot failed:', ' '+80h

        mov     a,b                ;error code
        call    PCAHEX

        call    ILPRNT
        db      'h'+80h

        jmp     CABORT

T0FAIL: call    CILPRT
        db      'Track 0 fai','l'+80h
        jmp     CABORT

DNRERR: call    CILPRT
        db      'Drive not read','y'+80h
        jmp     CABORT

endif ;TARBL1 or TARBL2

;=====
;= Memon/80 Subroutines =
;=====

;***Subroutine*****
;Report a verify error
;On Entry:
;  de=destination address of fail
;  hl=source address of fail
;Trashes a
;*****
F567 C5  VERROR: push    b
F568 CD48F6      call    PHLADR                ;Fail address on new line
                                                ;also sets b=0, trashes c
F56B CD5DF6      call    PMHCSM                ;b=0 so source data to Console
F56E CD15F6      call    ILPRNT
F571 A0          db      ' '+80h
F572 1A          ldax    d                    ;destination data
F573 CD5EF6      call    PAHCSM                ;b=0 so print on Console
F576 CD15F6      call    ILPRNT
F579 BF          db      '?' +80h                ;flag the error
F57A C1          pop     b

;Fall into CPAUSE for Pause and abort opportunity

;***Subroutine*****
;Get a keyboard character, abort if control-C, pause
;until the next keyboard character if anything else.
;On Exit:
;  a=keyboard chr
;*****
F57B CD8BF5  CPAUSE: call    CHKKBD                ;anything from the keyboard?

```

```

MEMON80.PRN
F57E C8          rz          ;N: done

;Fall into GETKBD to wait for the user

;***Subroutine*****
;Get a keyboard character, abort if CABKEY (control-C)
;On Exit:
;  a=keyboard chr
;  Z set if BS or delete
;*****
; if not TPORT
GETTPD:          ;No transfer port
endif ;not TPORT ;..so use the console

F57F CD8BF5      GETKBD: call    CHKKBD      ;get kbd chr, test for ^C
F582 CA7FF5      jz          GETKBD      ;wait for chr

F585 FE7F        cpi         DEL          ;delete?
F587 C8          rz
F588 FE08        cpi         BS           ;backspace?
F58A C9          ret

;***Subroutine*****
;Get keyboard status. if a chr is waiting, then
;return it in a with parity stripped. Abort if
;CABKEY (control-C).
;On Exit:
;  if a chr is waiting, then chr is in a
;  if no chr waiting, Z set, a=0
;*****
F58B CD97F6      CHKKBD: call    KSTAT      ;anything typed?
F58E C8          rz          ;N: ret w/ Z set

F58F CD8CF6      call        KDATA        ;Y: get chr, strip parity
F592 FE03        cpi         CABKEY       ;abort character typed?
F594 C0          rnz

F595 C3C5F5      jmp         CABORT        ;repair stack, go to MAIN

;***Subroutine*****
;Get 2 hex digits from the Transfer Port, combine them
;into one byte, and add the result to the checksum in d
;On Entry:
;  d=checksum so far
;On Exit:
;  b=byte of data
;  a=d=new checksum value
;  Z flag set if checksum is now 0
;  all other registers preserved, unless error abort
;*****
F598 CDA8F5      GTPBYT: call    GTPNIB     ;get high nibble
F59B 87          add         a            ;shift high nibble up
F59C 87          add         a
F59D 87          add         a
F59E 87          add         a
F59F 47          mov         b,a
F5A0 CDA8F5      call        GTPNIB     ;get low nibble

F5A3 B0          ora         b            ;combine nibbles
F5A4 47          mov         b,a         ;save result for return
F5A5 82          add         d            ;Compute checksum
F5A6 57          mov         d,a         ;ret with checksum in a & d
F5A7 C9          ret

```

# MEMON80.PRN

```

;---Local Subroutine-----
;Get a hex digit from the Transfer Port,
;validate it, and return it in a<3:0>
;-----
F5A8 CD34F6 GTPNIB: call GETTPD ;get Tx port byte
F5AB CD04F6 call HEXCON ;convert hex to binary
F5AE D8 rc ;carry means okay

;Abort: ASCII character error - not a valid hex digit

F5AF CD11F6 call CILPRT
F5B2 4368F2 db 'Ch','r'+80h

F5B5 C3C1F5 jmp CMDERR ;error handler

;***Subroutine*****
;Get two hex values from the keyboard input line buffer
;abort to CMDERR if none provided
;On Entry:
; de=address of next item in the input line buffer
;On Exit:
; 1st hex value is at top of the stack
; prior h1 value is next on the stack
; h1=2nd hex value
; de advanced past both hex values
; abort to CMDERR if no value found
;*****
F5B8 E3 GETHX2: xthl ;save old h1
F5B9 CDBC F5 call GETHEX ;get 1st hex value

;Fall into GETHEX to get 2nd hex value

;***Subroutine*****
;Get a hex value from the keyboard input line buffer
;abort to CMDERR if none provided
;On Entry:
; de=address of next item in the input line buffer
;On Exit:
; prior h1 value is on the stack
; h1=value
; de advanced past chr
; abort to CMDERR if no value found
;*****
F5BC E3 GETHEX: xthl ;save old h1
F5BD CDC4 F0 call PHFHEX ;restore our return address
;get h1=hex value
F5C0 D0 rnc

;Fall into CMDERR if no hex value provided

;*****
;command Error Handler
;*****
F5C1 CD11F6 CMDERR: call CILPRT
F5C4 BF db '?' +80h

;Fall into CABORT

;*****
;Command abort: repair stack and go to MAIN
;On Entry:
;

```



```

MEMON80.PRN
; if TPORT:
;   Bottom of stack (XXFF)=Transfer Port flag
;   (XXFE)=junk
;   next on stack (XXFc, XXFD)=MAIN
;
; if not TPORT:
;   Bottom of stack (XXFE, XXFF)=MAIN
;*****
CABORT:
    if RAMHNT and TPORT
F5C5 CDC9F2        call    RAMPAG        ;get hl=stack address
F5C8 2EAC          mvi     1,(STACK-4) and 0FFh ;return address on stack
F5CA F9           sphl                    ;fix stack
F5CB C9           ret                     ;return to MAIN
    endif ;RAMHNT and TPORT

    if RAMHNT and (not TPORT)
        call    RAMPAG        ;get hl=stack address
        mvi     1,(STACK-2) and 0FFh ;return address on stack
        sphl                    ;fix stack
        ret                     ;return to MAIN
    endif ;RAMHNT and (not TPORT)

    if (not RAMHNT) and TPORT
        lxi     sp,STACK-4
        ret                     ;return to MAIN
    endif ;(not RAMHNT) and TPORT

    if (not RAMHNT) and (not TPORT)
        lxi     sp,STACK-2
        ret                     ;return to MAIN
    endif ;not RAMHNT and (not TPORT)

;***Subroutine*****
;Read a command line from the keyboard, echoing and
;saving it in the input line buffer. CR input ends
;the sequence. LBCHR replaces the terminating CR
;with 0 for easy testing later.
;On Exit:
; Complete command line is in the input line buffer
; de=address of the first non-space chr on the line
; Carry set if nothing but spaces found
;*****
F5CC E5          GETLIN: push    h

    if RAMHNT
F5CD CDC9F2        call    RAMPAG        ;get RAM address
F5D0 2EB0          mvi     1,RAMBUF      ;buffer location in page
    endif ;RAMHNT

    if not RAMHNT
        lxi     h,RAMBUF
    endif ;not RAMHNT

F5D2 E5          push    h                ;save input line buffer's
                                           ;..start address

;Get & echo chrs, stashing them in the input line buffer
;at hl, until a CR is encountered or the line overflows

F5D3 3EFF          GLLOOP: mvi     a,(RAMBUF+LBSIZE-1) and 0FFh ;input buf full?

```

```

MEMON80.PRN
F5D5 95          sub    l          ;carry set if full
F5D6 D4E6F5     cnc     LBCHR      ;N:get another character
F5D9 D2D3F5     jnc     GLLLOOP    ;carry means CR found

F5DC D1         pop     d          ;de=line buffer address
F5DD E1         pop     h          ;restore original hl

;Fall into SSPACE to skip leading spaces

;***Subroutine*****
;Scan past spaces in line buffer, looking for the
;first non-space character
;On Entry:
;  de=starting address within the input line buffer
;On Exit:
;  Carry set if no chr or only control chrs found
;  a=character value if found, 0 if end of line
;  de advanced past spaces
;*****
F5DE 1A         SSPACE: ldax    d          ;get next character
F5DF FE20       cpi     ' '        ;another space?
F5E1 C0         rnz          ;carry set for any ctrl chr
F5E2 13         inx     d          ;next scan address
F5E3 C3DEF5     jmp     SSPACE    ;keep skipping

;***Subroutine*****
;Get, echo, and store a Console character in the input
;line buffer. Handle deletes and backspaces. This
;assumes that the line buffer is all in one page. CR's
;are not echoed, but are replaced with 0.
;On Entry:
;  hl=next free spot in the input line buffer
;On Exit (no deletes, no CR):
;  chr stored in buffer at original hl
;  hl=hl+1
;  Carry clear
;On Exit (CR)
;  (hl)=0
;  Carry set
;On Exit (BS or DEL)
;  hl=hl-1
;  Carry clear
;*****
F5E6 CD7FF5     LBCHR: call    GETKBD    ;get a chr, test BS & DEL
F5E9 CAF8F5     jz      LDELET    ;BS or delete?

F5EC 3600       mvi     m,0        ;replace CR with a null
F5EE FE0D       cpi     CR        ;
F5F0 37         stc          ;flag CR found
F5F1 C8         rz          ;

F5F2 3F         cmc          ;clear carry for ret
F5F3 77         mov     m,a        ;enqueue
F5F4 2C         inr     l          ;Bump line buffer pointer
F5F5 C380F6     jmp     PRINTA    ;echo & ret with carry clear

;Backspace if possible

F5F8 7D         LDELET: mov     a,l          ;buffer size

```

```

MEMON80.PRN
F5F9 FEB0      cpi      RAMBUF and 0FFh ;back up if we can
F5FB C8        rz              ;done if not (carry clear)

F5FC CD15F6    call     ILPRNT          ;back up, clear Carry
F5FF 082088    db       BS,' ',BS+80h  ;erase old chr & back up

F602 2D        dcr       1              ;back up

F603 C9        ret

```

```

;***Subroutine*****
;Convert ASCII hex digit to binary
;On Entry:
;  a=chr to convert
;On Exit:
;  a=binary
;  Carry set if OK, clear if bogus chr
;*****

```

```

F604 D630      HEXCON: sui    '0'          ;remove ASCII bias
F606 FE0A      cpi      10
F608 D8        rc              ;if 0-9 then we're done

F609 D611      sui      9+('A'-'9')      ;should be 0-5 now
F60B FE06      cpi      6                ;gap chr or too high?
F60D D0        rnc              ;error: return w/o carry

F60E D6F6      sui      0F6h             ;add 0Ah, set Carry
F610 C9        ret

```

```

;***Subroutine*****
;Print CR LF, followed by inline message at (sp)
;on Console. (Calls to CILPRT are followed by the
;string.) The last string byte has its msb set
;On Exit:
;  carry & Z cleared
;  Interrupts enabled if ENINTS=TRUE
;Trashes a. All other registers preserved
;*****

```

```

CILPRT:
  if (ENINTS or CRXINT or TRXINT)
    ei          ;enable interrupts
  endif ;(ENINTS or CRXINT or TRXINT)

```

```

F611 CD15F6    call     ILPRNT
               if not HELPC
               db       CR,LF+80h
               endif ;not HELPC

```

```

               if HELPC
               db       CR+80h
               endif ;HELPC

```

```

;Fall into ILPRNT

```

```

;***Subroutine*****
;Print inline message at (sp) on Console.
;(Calls to ILPRNT are followed by the string.)
;The last string byte has its msb set.

```

```

;On Exit:
;  carry & Z cleared
;Trashes a. All other registers preserved
;
;The help screen has a lot of lines, making it

```

```

MEMON80.PRN
;more efficient to follow all CRs with LFs
;automatically.. Otherwise, it's not worth it.
;*****
F615 E3      ILPRNT: xthl          ;save hl, get msg addr

F616 7E      IPLOOP: mov      a,m          ;LOOP through message
F617 E67F    ani      7Fh          ;strip end-marker
F619 CD80F6  call     PRINTA

        if HELPC
F61C FE0D    cpi      CR
F61E 3E0A    mvi      a,LF
F620 CC80F6  cz       PRINTA
        endif ;HELPC

F623 B6      ora      m          ;End? (clears carry too)
F624 23      inx      h
F625 F216F6  jp       IPLOOP      ;test msb in memory string

F628 E3      xthl          ;restore hl
F629 C9      ret           ;..get ret address

;***Subroutine*****
;Test which port is the Transfer Port
;On Entry:
;  sp points into the RAM page
;  RAM page flag=0 for Console, <>0 for Transfer Port
;On Exit:
;  Z set only if the Transfer Port is the Console
;*****
        if TPORT
F62A E5      TESTTP: push     h
        endif ;TPORT

        if TPORT and RAMHNT
F62B CDC9F2  call     RAMPAG      ;find RAM page
F62E 2EAF    mvi      l,(STACK-1) and 0FFh ;get flag from RAM page
        endif ;TPORT and RAMHNT

        if TPORT and (not RAMHNT)
                lxi      h,STACK-1
        endif ;TPORT and (not RAMHNT)

        if TPORT
F630 34      inr      m          ;test flag
F631 35      dcr      m          ;Z means disabled

F632 E1      pop      h
F633 C9      ret
        endif ;TPORT

;***Subroutine*****
;Get a byte from the Transfer Port
;Strips parity, checks for control-C abort
;from the Console keyboard
;On Entry:
;  sp points into the RAM page
;  RAM page byte FE=0 for Console, <>0 for Transfer Port
;On Exit:
;  chr in a, with parity stripped
;*****
        if TPORT

```

```

MEMON80.PRN
F634 CD2AF6 GETTPD: call TESTTP ;which port?
F637 CA7FF5      jz   GETKBD  ;Z means Console

F63A CD8BF5 GTPLUP: call CHKKBD ;user abort?
F63D CD9FF6      call TPISTA ;Transfer Port chr?
F640 CA3AF6      jz   GTPLUP
      endif ;TPORT

      if TPORT and (not TMAP) and (not TRXINT)
F643 DBEC        in     TDATA ;get Transfer Port chr
F645 E67F        ani    7Fh   ;strip parity
F647 C9          ret
      endif ;TPORT and (not TMAP) and (not TRXINT)

      if TPORT and TMAP and (not TRXINT)
        lda     TDATA ;get Transfer Port chr
        ani    7Fh   ;strip parity
        ret
      endif ;TPORT and TMAP and (not TRXINT)

      if TPORT and TRXINT
        call    TPIDAI
        ani    7Fh   ;strip parity
        ret
      endif ;TPORT and TRXINT

;***Subroutine*****
;Print hl on a new line in hex on
;the Console, followed by ': '
;Trashes psw,bc
;*****
F648 CD11F6 PHLADR: call CILPRT ;CR LF space begins line
F64B A0      db     '+'80h

F64C CD55F6      call PHLCHX ;hl=address on Console
                        ;Trashes bc

;Fall into PCCOLS to print ': '

;***Subroutine*****
;Print ': ' on the Console
;Trashes psw
;*****
F64F CD15F6 PCCOLS: call ILPRNT ;print colon space
F652 3AA0      db     ':',' '+80h
F654 C9          ret

;***Subroutine*****
;print hl as 4 hex digits on the
;Console & accumulate the checksum
;On Entry:
; (c=checksum so far)
; hl=2 bytes to print
;On Exit:
; b=0
; (c=updated checksum)
;Trashes psw
;*****
F655 0600 PHLCHX: mvi b,0 ;print on Console

;Fall into PHLHEX

;***Subroutine*****

```

```

MEMON80.PRN
;Print hl as 4 hex digits & accumulate checksum
;On Entry:
;  b=0 for Console, <>0 for Transfer Port
;  c=checksum so far
;  hl=2 bytes to print
;On Exit:
;  c=updated checksum
;Trashes psw
;*****
F657 7C      PHLHEX: mov     a,h           ;h first
F658 CD5EF6   call     PAHCSM          ;returns with carry clear
F65B 7D      mov     a,l           ;then l

F65C FE      db         CPI           ;skip over PMHCSM

;Skip into PAHCSM

;***Subroutine*****
;Print m as 2 hex digits & accumulate checksum
;On Entry:
;  (HL)=byte to print
;  b=0 for Console, <>0 for Transfer Port
;  c=checksum so far
;On Exit:
;  c=updated checksum
;Trashes psw
;*****
F65D 7E      PMHCSM: mov     a,m

;Fall into PAHCSM

;***Subroutine*****
;Print a as 2 hex digits & accumulate checksum
;On Entry:
;  a=byte to print
;  b=0 for Console, <>0 for Transfer Port
;  c=checksum so far
;On Exit:
;  c=opdated checksum
;Trashes psw
;*****
F65E F5      PAHCSM: push    psw
F65F 81      add     c           ;Compute checksum
F660 4F      mov     c,a
F661 F1      pop     psw        ;recover chr

        if TPORT
F662 FE      db         CPI           ;CPI opcode skips 1

;Skip into PAHEX2 (executing a NOP on the way)
endif ;TPORT

;***Subroutine*****
;Print a on Console as 2 hex digits
;On Entry:
;  a=byte to print
;Trashes psw, b
;*****
F663 0600    PCAHEX:      if TPORT
                        mvi     b,0           ;print on the Console
                        endif ;TPORT

```

# MEMON80.PRN

```

;Fall into PAHEX2
;***Subroutine*****
;Print a as 2 hex digits
;On Entry:
;  a=byte to print
;  b=0 for Console, <>0 for Transfer Port
;    (only if TPORT is TRUE)
;Trashes psw
;*****
F665 F5    PAHEX2: push    psw            ;save for low digit
F666 0F                rrc                ;move the high four down
F667 0F                rrc
F668 0F                rrc
F669 0F                rrc
F66A CD6EF6          call    PAHEX1        ;print high digit
F66D F1                pop     psw         ;this time the low four

;Fall into PAHEX1 to print low digit
;***Subroutine*****
;Print low nibble of a as 1 hex digit
;On Entry:
;  b=0 for Console, <>0 for Transfer Port
;    (only if TPORT is TRUE)
;On Exit:
;  psw trashed
;*****
F66E E60F          PAHEX1: ani     0Fh        ;Four on the floor
F670 C630                adi     '0'         ;We work with ASCII here
F672 FE3A                cpi     '9'+1       ;0-9?
F674 DA79F6          jc      PNIB1          ;Yup: print & return

F677 C607                adi     'A'-'9'-1    ;make it a letter
PNIB1:

    if TPORT
F679 04                inr     b              ;which port?
F67A 05                dcr     b
F67B C2C0F6          jnz     TPOUT          ;print on Transfer Port
    endif ;TPORT

F67E FE                db      CPI          ;CPI opcode skips PRINTC

;Skip into PRINTA
;***CP/M External Subroutine*****
;Print C on the Console
;On Entry:
;  the character for output is in c
;On Exit:
;  a=c=chR
;  all other regs preserved
;*****
F67F 79    PRINTC: mov     a,c

;Fall into PRINTA
;***Subroutine*****
;Print a on the Console
;On Entry:

```

```

MEMON80.PRN
; a=the character to print
;all regs preserved
;*****
F680 F5 PRINTA: push    psw

        if not CMAP
F681 DBFF PAWAIT: in      CSTAT
F683 E620      ani      CTXRDY      ;wait for transmitter ready

        endif ;not CMAP

        if CMAP
PAWAIT: lda      CSTAT
      ani      CTXRDY      ;wait for transmitter ready
        endif ;CMAP

        if CISTAT
      jnz      PAWAIT

        endif ;CISTAT

F685 CA81F6      if not CISTAT
      jz      PAWAIT
        endif ;not CISTAT

        if not CMAP
F688 F1      pop      psw      ;recover chr
F689 D3FA      out      CDATA
F68B C9      ret
        endif ;not CMAP

        if CMAP
      pop      psw      ;recover chr
      sta      CDATA
      ret
        endif ;CMAP

;***CP/M External Subroutine*****
;Wait for and get Console keyboard data
;On Exit:
; a=keyboard character, parity stripped
; Z clear
;*****
F68C CD97F6 KDATA: call    KSTAT
F68F CA8CF6      jz      KDATA

        if (not CMAP) and (not CRXINT)
F692 DBFA      in      CDATA      ;get keyboard chr
        endif ;(not CMAP) and (not CRXINT)

        if CMAP and (not CRXINT)
      lda      CDATA      ;get keyboard chr
        endif ;CMAP and (not CRXINT)

        if CRXINT
      xra      a      ;clear flag
      di      ;mask while we work
      sta      CIFLAG
      lda      CRXBUF      ;get data
      ei
        endif ;CRXINT

```



```

MEMON80.PRN
F694 E67F      ani      7Fh      ;strip parity
F696 C9        ret

;***CP/M External Subroutine*****
;Get Console keyboard status
;On Exit:
;  a=0 and Z set if nothing waiting
;  a=FF and Z cleared if kbd chr waiting
;*****
KSTAT:
F697 DBFF      if (not CMAP) and (not CRXINT)
                in      CSTAT
endif ;(not CMAP) and (not CRXINT)

if CMAP and (not CRXINT)
    lda      CSTAT
endif ;CMAP and (not CRXINT)

if CISTAT and (not CRXINT)
    cma
endif ;CISTAT and (not CRXINT) ;inverted status bit

F699 E601      if not CRXINT
                ani      CRXRDY
endif ;not CRXINT

if CRXINT
    lda      CIFLAG
    ora      a
endif ;CRXINT

;Fall into DOSTAT

;***Subroutine*****
;Create CP/M-style return value
;On Exit:
;  if Z then a=0
;  if NZ then a=0FFh
;*****
F69B C8        DOSTAT: rz      ;CP/M-style return values
F69C 3EFF      mvi      a,0FFh
F69E C9        ret

;***CP/M External Subroutine***
;Get Transfer Port Rx status
;On Exit:
;  a=0 & Z set if no data
;  a=FF & Z clear if data
;*****
TPISTA:
F69F DBED      if TPORT and (not TMAP) and (not TRXINT)
                in      TSTAT
endif ;TPORT and (not TMAP) and (not TRXINT)

if TPORT and TMAP and (not TRXINT)
    lda      TSTAT
endif ;TPORT and TMAP and (not TRXINT)

if TPORT and TISTAT and (not TRXINT)
    cma
endif ;TPORT and TISTAT and (not TRXINT) ;inverted status bit

```

```

MEMON80.PRN
if TPORT and (not TRXINT)
    ani    TRXRDY
    jmp    DOSTAT
endif ;TPORT and (not TRXINT)

if TPORT and TRXINT
    lda    TIFLAG
    ora    a        ;a=0 or FF
    ret
endif ;TPORT and TRXINT

;***CP/M External Subroutine***
;Get Transfer Port Rx data
;On Exit:
;  a=byte from port
;  Z cleared
;*****
if TPORT
F6A6 CD9FF6 TPIDAT: call    TPISTA        ;wait for chr
F6A9 CAA6F6        jz      TPIDAT
endif ;TPORT

if TPORT and (not TMAP) and (not TRXINT)
F6AC DBEC    in      TDATA
F6AE C9      ret
endif ;TPORT and (not TMAP) and (not TRXINT)

if TPORT and TMAP and (not TRXINT)
    lda    TDATA
    ret
endif ;TPORT and TMAP and (not TRXINT)

if TPORT and TRXINT
TPIDAI: xra    a        ;clear flag
        di      ;mask while we work
        sta    TIFLAG
        lda    TRXBUF    ;Get data
        ei
        ret
endif ;TPORT and TRXINT

;***Subroutine*****
;Finish writing Intel end-of-file hex record
;(the CRLF and colon have already been sent.)
;*****
F6AF 0605 HDEOF: mvi    b,5        ;5 bytes for EOF

F6B1 AF HDELUP: xra    a
F6B2 CD65F6 call    PAHEX2        ;B<>0 for Transfer Port
F6B5 05 dcr    b
F6B6 C2B1F6 jnz    HDELUP

;Fall into TPCRLF for a final CR LF

;***Subroutine*****
;Send CR LF to the Transfer Port
;On Entry:
;  sp points into the RAM page
;  RAM page byte FE=1 for channel B, 0 for Console
;Trashes psf
;*****
F6B9 3E0D TPCRLF: mvi    a,CR        ;CR,LF to Transfer Port
F6BB CDC0F6 call    TPOUT

```

```

F6BE 3E0A          mvi      a,LF
;Fall into TPOUT
;***Subroutine*****
;Send a to the Transfer Port
;On Entry:
;  sp points into the RAM page
;  RAM page byte FE=1 for channel B, 0 for Console
;Trashes flags
;*****
TPOUT:
  if TPORT
F6C0 CD2AF6      call      TESTTP          ;which physical port?
F6C3 CA80F6      jz        PRINTA          ;Z means Console

F6C6 FE          db        CPI              ;CPI opcode skips TPDATC
;..TO Fall into PBODAT

  endif ;TPORT

  if not TPORT
    jmp          PRINTA
  endif ;not TPORT

;***CP/M External Subroutine***
;Send c to Transfer Port
;On Exit:
;  a=c=chr
;All other regs preserved
;*****
  if TPORT
F6C7 79          TPDATC: mov      a,c

;Fall into PBODAT
  endif ;TPORT

;***Subroutine*****
;Send a to Transfer Port
;All regs preserved
;*****
  if TPORT
F6C8 F5          PBODAT: push     psw

F6C9 CD18F0      PBODA1: call     TPOSTA      ;wait for transmitter
F6CC CAC9F6      jz          PBODA1

  endif ;TPORT

  if TPORT and (not TMAP)
F6CF F1          pop          psw            ;recover chr
F6D0 D3EC      out          TDATA
F6D2 C9          ret

  endif ;TPORT and (not TMAP)

  if TPORT and TMAP
    pop          psw            ;recover chr
    sta          TDATA
    ret
  endif ;TPORT and TMAP

;*****
;8251 initialization table
;*****

```

```

MEMON80.PRN
if C8251 or T8251
IN8251: db      SI21,SI22,SI23,SI24
endif ;C8251 or T8251
;*****
;Z80-DART initialization table
;(Gets sent to both DART channels)
;*****
if DART
F6D3 28010003C1DITAB: db      DI1,DI2,DI3,DI4,DI5,DI6,DI7
endif ;DART
;*****
;Transfer Port Baud Rate Table
;Each entry has 4 bytes:
;  Byte 0=value for BRATE0 port (written 1st)
;  Byte 1=value for BRATE1 port (written 2nd)
;  Byte 3-4=Decimal value for baud rate string,
;              encoded as 4 hex digits. Leading zeros
;              get suppressed, and a "0" gets appended
;              to these digits when printed.
;*****
if TSBAUD and ROM2K
F6DA 8D07      dw      TBD110 ;0= 110 baud 2 stop bits
F6DC 0011      db      00h,11h ;string

F6DE 6807      dw      TBD150 ;1= 150 baud
F6E0 0015      db      00h,15h

F6E2 3407      dw      TBD300 ;2= 300 baud
F6E4 0030      db      00h,30h

F6E6 C047      dw      TBD600 ;3= 600 baud
F6E8 0060      db      00h,60h

F6EA 6047      dw      TBD1200 ;4= 1200 baud
F6EC 0120      db      01h,20h

F6EE 3047      dw      TBD2400 ;5= 2400 baud
F6F0 0240      db      02h,40h

F6F2 1847      dw      TBD4800 ;6= 4800 baud
F6F4 0480      db      04h,80h

F6F6 0C47      dw      TBD9600 ;7= 9600 baud
F6F8 0960      db      09h,60h
endif ;TSBAUD and ROM2K

if BD192 and TSBAUD and ROM2K
F6FA 0647      dw      TBD192 ;8= 19200 baud
F6FC 1920      db      19h,20h
endif ;BD192 and TSBAUD and ROM2K

if BD384 and TSBAUD and ROM2K
F6FE 0347      dw      TBD384 ;9= 38400 baud
F700 3840      db      38h,40h
endif ;BD383 and TSBAUD and ROM2K

if BD576 and TSBAUD and ROM2K
F702 0247      dw      TBD576 ;a= 57600 baud
F704 5760      db      57h,60h

```

```

MEMON80.PRN
endif ;BD576 and TSBAUD and ROM2K

if BD768 and TSBAUD and ROM2K
    dw    TBD768    ;b= 76800 baud
    db    76h,80h
endif ;BD768 and TSBAUD and ROM2K

if TSBAUD and ROM2K
BTEND:
endif ;TSBAUD and ROM2K

;***Command Routine Continuation***
;? Print (Minimal) Help Screen
;*****
if HELPC
F706 CD11F6    GHHELP: call    CILPRT

;Memory commands

F709 434F207320    db 'CO s d c [r]',CR
F716 4455206120    db 'DU a c',CR
F71D 454E20610D    db 'EN a',CR
F722 4649205B61    db 'FI [a [c [v]]]',CR
endif ;HELPC

if HELPC and ROM2K
F731 4D54206120    db 'MT a c',CR
F738 5345206120    db 'SE a v v ',QUOTE,'t',QUOTE,'...',CR
endif ;HELPC and ROM2K

if HELPC
F748 5645207320    db 'VE s d c',CR,LF

;I/O commands

F752 494E20700D    db 'IN p',CR
F757 4F54207020    db 'OT p v',CR,LF
endif ;HELPC

;Disk commands

if HELPC and BOOTER
F75F 424F          db 'BO'
endif ;HELPC and BOOTER

if HELPC and (C4FDC or C16FDC or C64FDC)
    db ' d' ;Cromemco can boot from any disk
endif ;HELPC and (C4FDC or C16FDC or C64FDC)

if HELPC and BOOTER
F761 0D            db CR
endif ;HELPC and BOOTER

if HELPC and ROM2K
F762 434520636C    db 'CE cl',CR
endif ;HELPC and ROM2K

if HELPC
F768 45582061      db 'EX a'
endif ;HELPC

if HELPC and EXOPT
    db ' [0/1]'

```

```

endif ;HELPC and EXOPT

if HELPC
F76C 0D0A      db CR,LF
;Transfer Port Commands

F76E 4844206120      db 'HD a c',CR
endif ;HELPC

if TPORT and HELPC
F775 484C205B6F      db 'HL [o]',CR
endif ;TPORT and HELPC

if (not TPORT) and HELPC
      db 'HL [o]',CR+80h
endif ;(not TPORT) and HELPC

if HELPC and TSBAUD
F77C 544220302D      db 'TB 0-'
endif ;HELPC and TSBAUD

if (HELPC and TSBAUD) and not (BD192 or BD384 or BD576 or BD768)
      db '7'
endif ;(HELPC and TSBAUD) and not (BD192 or BD384 or BD576 or
BD768)

if (HELPC and TSBAUD and BD192) and not (BD384 or BD576 or BD768)
      db '8'
endif ;(HELPC and TSBAUD and BD192) and not (BD384 or BD576 or
BD768)

if (HELPC and TSBAUD and BD384) and not (BD576 or BD768)
      db '9'
endif ;(HELPC and TSBAUD and BD384) and not (BD576 or BD768)

if (HELPC and TSBAUD and BD576) and not BD768
      db 'A'
endif ;(HELPC and TSBAUD and BD576) and not BD768

if HELPC and TSBAUD and BD768
      db 'B'
endif ;TSBAUD and HELPC and BD768

if TPORT and HELPC and TSBAUD
F782 0D      db CR
endif ;TPORT and HELPC and TSBAUD

if TPORT and HELPC
F783 544520650D      db 'TE e',CR
F788 545020302F      db 'TP 0/1',CR+80h
endif ;TPORT and HELPC

if HELPC
F78F C9      ret
endif ;HELPC

```

```

;=====
;command table
;EACH entry is 3 bytes:
;  byte 0: 1st command character
;  byte 1 <6:0>: 2nd command character
;  byte 1 <7>: address offset msb

```

```

MEMON80.PRN
; byte 2: address offset from CMDBAS
;the table terminates with byte 0=0.
;=====
F790 4455 COMTAB: db 'DU' ;Dump
F792 E5 db DUCMD-CMDBAS

F793 454E db 'EN' ;Enter
F795 53 db ENCMD-CMDBAS

F796 4558 db 'EX' ;Execute
F798 DD db EXCMD-CMDBAS

F799 4649 db 'FI' ;Fill memory
F79B 6A db FICMD-CMDBAS

F79C 494E db 'IN'
F79E BF db INCMD-CMDBAS ;In from port

F79F 4F54 db 'OT'
F7A1 CF db OTCMD-CMDBAS ;Out to port

if TPORT
F7A2 5445 db 'TE' ;Terminal mode
F7A4 03 db TECMD-CMDBAS

F7A5 5450 db 'TP' ;Set Transfer Port
F7A7 DE db TPCMD-CMDBAS
endif ;TPORT

0192 = VED equ VECMD-CMDBAS
F7A8 56C5 db 'V','E'+((VED/2) and 80h) ;verify
F7AA 92 db VED and 0FFh

0199 = COD equ COCMD-CMDBAS
F7AB 43CF db 'C','O'+((COD/2) and 80h) ;COPY
F7AD 99 db COD and 0FFh

0083 = HDD equ HDCMD-CMDBAS
F7AE 4844 db 'H','D'+((HDD/2) and 80h) ;Intel hex dump
F7B0 83 db HDD and 0FFh

0136 = HLD equ HLCMD-CMDBAS
F7B1 48CC db 'H','L'+((HLD/2) and 80h) ;Intel hex load
F7B3 36 db HLD and 0FFh

TBD set 0
if TPORT and TSBAUD and ROM2K
TBD set TBCMD-CMDBAS
F7B4 5442 db 'T','B'+((TBD/2) and 80h) ;Set T Port Baud Rate
F7B6 42 db TBD and 0FFh
endif ;TPORT and TSBAUD and ROM2K

BOD set 0
if BOOTER
BOD set BOCMD-CMDBAS
F7B7 424F db 'B','O'+((BOD/2) and 80h) ;Boot frm disk
F7B9 45 db BOD and 0FFh
endif ;BOOTER

SED set 0
if ROM2K
F7BA 4345 db 'CE' ;Execute CP/M program
F7BC 00 db CECMD-CMDBAS

```

# MEMON80.PRN

```

F7BD 4D54      db      'MT'                ;Memory Test
F7BF 3C        db      MTCMD-CMDBAS

SED           set      SECMD-CMDBAS
F7C0 5345      db      'S','E'+((SED/2) and 80h) ;Search
F7C2 3F        db      SED and 0FFh
endif ;ROM2K

if HELPC
F7C3 3F00      db      '?',0                ;Help
F7C5 39        db      HELP-CMDBAS
endif ;HELPC

F7C6 00        db      0                    ;End of table mark

```

```

;=====
;Interrupt code for the H8-5 Rx Interrupt, as the console
;This code assumes RAMHNT = FALSE
; RAM page variables
; CRXBUF: 1-byte receive queue
; CIFLAG: 0FFh means CRXBUF has a character to send
;=====

```

```

if CH85
CONINT: push    psw                        ;onto MEMON's stack

;Receive and enqueue a character, ignoring any overflow

        in      CDATA                    ;get chr now
        sta     CRXBUF                    ;enqueue chr

        mvi     a,0FFh                    ;flag received chr
        sta     CIFLAG

;Restore registers and return from the interrupt

        pop     psw
        ei
        ret

```

endif ;CH85

```

;=====
;Interrupt code for the H8-5 Rx Interrupt, as the transfer port
;This code assumes RAMHNT = FALSE
; RAM page variables
; TRXBUF: 1-byte receive queue
; TIFLAG: 0FFh means TRXBUF has a character to send
;=====

```

```

if TH85
TBINT: push    psw                        ;onto MEMON's stack

;Receive and enqueue a character, ignoring any overflow

        in      TDATA                    ;get chr now
        sta     TRXBUF                    ;enqueue chr

        mvi     a,0FFh                    ;flag received chr
        sta     TIFLAG

;Restore registers and return from the interrupt

        pop     h

```



