# UBMON

# Altair
# Universal Boot Loader
# and PROM Monitor
## User's Guide

By Martin Eberhard

August 17, 2014

## Revision History

| Revision | Date | Author | Notes |
|---|---|---|---|
| 1.00 | 11 SEP 2013 | M. Eberhard | Created |
| 1.01 | 16 SEP 2013 | M. Eberhard | Shorten leader 20 60 bytes, add 60 byte trailer for D command. This version was hosed. |
| 1.02 | 17 SEP 2013 | M. Eberhard | Fixed mistake in 1.01 |
| 1.03 | 17 AUG 2014 | M. Eberhard | Punch 60 bytes of 20h before the leader, for MBL compatibility |

## ABSTRACT

This document describes the operation of the Altair Universal Boot Loader and PROM Monitor (UBMON). UBMON is a system program that allows you to examine and change the contents of any memory address or series of addresses, start execution of a program at any specified address, punch a tape from memory in Altair Binary Absolute Load format, and boot from any Altair boot device.

UBMON is an improvement to the TURMON PROM by MITS. All commands function the same way as TURMON, and three boot commands (B,L, and T) have been added. In addition, the D command now includes a short null trailer after the data dump is completed. Important subroutines remain at the same addresses as in TURMON, for compatibility.

## TABLE OF CONTENTS

## 1. SYSTEM REQUIREMENTS

UBMON has the same requirements for I/O Ports, available RAM, and PROM addressing as does the TURMON PROM from MITS.

### Terminal I/O Port

UBMON requires a 6850-based serial port addressed at 020 and 021 octal (10h and 11h), such as port A of an 88-2SIO, or the serial port on an 88-UIO or on an 8800b Turnkey Module.

### PROM Addressing

UBMON is 256 decimal (400 octal) bytes long and is assembled to operate with a starting address of 176400 octal (FD00h). This is socket K1 on an 8800b Turnkey Module, and socket F on an 88-PMC.

### Available RAM

UBMON requires a small amount of RAM for its stack. There are three versions of UBMON. The only difference between these versions is in which RAM page UBMON establishes its stack:

| UBMON Version | PROM Board | RAM Page Address Range | |
|---|---|---|---|
| | | Octal | Hex |
| UBMONb | e.g. 8K Bytesaver | 157400-157777 | DF00-DFFF |
| UBMONp | 88-PMC | 173400-173777 | F700-F7FF |
| UBMONt | 8800b Turnkey Module[1] | 175400-175777 | FB00-FBFF |

UBMON actually only uses the highest eight bytes in its RAM page.

### Associated Loader PROMs

The three new load commands (B, L, and T) each require an associated loader PROM to be installed, to enable the command:

| UBMON Command | Boot Device | Loader PROM | PROM Address | | IC Socket | |
|---|---|---|---|---|---|---|
| | | | Octal | Hex | Turnkey | 88-PMC |
| B | 88-DCDD | DBL or CDBL | 177400 | FF00 | H1 | H |
| B | 88-MDS | MDBL or CDBL | 177400 | FF00 | H1 | H |
| L | 88-HDSK | HDBL | 176000 | FC00 | L1 | E |
| T | Various | MBL or MBLe | 177000 | FE00 | J1 | G |

[1] UBMONt requires either the Turnkey Module's RAM to be enabled or the Altair-style "Phantom" rework to be installed on the Turnkey Module, and 64K of RAM installed in the Altair. Otherwise, use one of the other UBMON versions.

## 2. STARTING UBMON

To run UBMON in a Turnkey Module:

a) Install the UBMON PROM in PROM socket K1.
b) Set the AUTO-START address and PROM address switches on your Turnkey Module to 176400 octal (FD00h).
c) If your Turnkey Module has its RAM enabled, then set the RAM address switches to 174000 octal (F800h).
d) Otherwise, make sure you have RAM in the Altair at the address that UBMON will put its stack.
e) Set the Serial Port address switches to 020 octal (10h).
f) Turn the Altair’s power on.
g) If you are using an Altair with a front panel, then reset the Altair and press RUN.
h) UBMON prints its prompt character, a period (.).
i) The START switch (on an Altair 8800bt) or the RESET switch (on an Altair with a front panel) restarts UBMON.

You can use an 88-PMC PROM board if your Altair has a front panel and a RAM card as noted above. You must also have a compatible serial port (e.g. an 88-2SIO or an 88-UIO.) addressed at 020 octal (10h). To run UBMON in an 88-PMC PROM board:

a) Install the UBMON PROM in socket F.
b) Set the PROM address switches to 174000 octal (F800h).
c) Make sure you have a RAM card with memory for UMBON’s stack.
d) Make sure your serial port is addressed at 020 octal (10h).
e) Turn the power on.
f) Use the Altair front panel to go to address 176400 octal:
    a. Hold the STOP switch and press the RESET switch
    b. Enter 176400 octal (FD00h) on the address switches
    c. press the EXAMINE switch
    d. press the RUN switch
g) UBMON prints its prompt character, a period (.).
h) To return control to the UBMMON, repeat step f above.

## 3. OPERATION

UBMON has six commands:

| | |
|---|---|
| D | Memory dump in “Altair Binary Absolute Load” format |
| J | Jump to another program |
| M | Memory examine and modify |
| L | Boot from 88-HDSK hard disk |
| B | Boot from 88-DCDD floppy disk or 88-MDS minidisk |
| T | Boot from paper tape or cassette tape |

## The D Command

The D command allows you to dump (on the Terminal port) the contents of the Altair’s memory between any two addresses, in Altair Binary Absolute Load Format. The D command has the following form:

```
Dxxxxxx yyyyyy
```

To use the D command, type D in response to the prompt. UBMON will then wait for the starting address xxxxxx (zero to six valid octal digits). If six digits are input, UBMON prints a space and then waits for the ending address yyyyyy (zero to six valid octal digits). Fewer than six digits may be entered for either address, by typing a space when done.

The data is dumped to port 021 octal (11h), which is also the Altair’s Terminal port. If you have a paper tape punch (for example, the punch on a Teletype), then turn the punch on before the last character of the ending address is typed.

Once UBMON receives a valid starting and ending address, it punches a leader of 60 space characters (040 octal, 20h) followed by 60 null characters. It then punches out the contents of memory starting at the starting address, up to and including the ending address in the Altair Binary Absolute Load format, as shown in Appendix A. (The word “punch” is used to refer to the output of the D command, no matter what output device is used.) If the number of bytes to be punched is greater than 255 (377 octal), then UBMON punches as many 255-byte blocks as necessary until the number of bytes left to punch is less than 255. The last block punched may have fewer than 255 bytes, but a block of zero bytes will not be punched. Upon completion of the dump, UBMON prints a trailer of 60 null characters, and then performs a carriage return and line feed, and finally returns to the prompt.

## The J Command

The J command allows you to transfer control to another program. The J command has the following form:

```
Jxxxxxx
```

where xxxxxx is the starting address (in octal) of the program to run. Once all 6 address digits are entered (fewer, if terminated by a space), UBMON will jump to the address you entered.

## The M Command

The M command allows you to examine and modify address in the Altair memory. The M command has the following form:

```
Mxxxxxx
```

where xxxxxx stands for a memory address expressed as six valid octal digits. UBMON reads the address specified and displays the three digit octal contents of that address. UBMON then waits for three valid octal digits. When this valid data has been received, UBMON attempts to write the data into the same address. Once the write has been made and verified, UBMON reads and displays the following address.

If you attempt to write information into nonexistent memory, ROM, or protected RAM, the bad write will cause “?” to be printed on the terminal and the UBMON prompt to be printed.

Assuming a valid write, this sequence continues until a non-valid character (any character except the digits 0-7) is typed. This non-valid character is flagged with a “?” and the prompt is printed. This is the normal way to return to the prompt.

If a space is typed instead of any valid octal characters, UBMON moves the next address without writing.

### The B Command

The B command initiates loading from either an 88-DCDD floppy disk or an 88-MDS minidisk, via the DBL or MDBL PROM. The B command has the following form:

```
B
```

UBMON examines memory at address 177400 octal (FF00h), which is the first address in the DBL, MDBL, and CDBL PROMs. If this address contains anything except 377 octal (FFh), then UBMON assumes a valid floppy disk boot PROM is installed, and control is passed to the PROM at this address to boot from either an 88-DCDD or an 88-MDS.

If this address contains 377 octal (FFh), then UBMON assumes no valid floppy disk loader PROM exists, and the command is ignored.

### The L Command

The L command initiates loading from an 88-HDSK hard disk, via the HDBL PROM. The L command has the following form:

```
L
```

UBMON examines memory at address 176000 octal (FC00h), which is the entry point for the HDBL PROM. If this address contains anything except 377 octal (FFh), then UBMON assumes a valid HDBL PROM is installed, and control is passed to the HDBL PROM at this address to boot from an 88-HDSK.

If this address contains 377 octal (FFh), then UBMON assumes no HDSK PROM exists, and the command is ignored.

### The T Command

The T command initiates loading from paper tape or cassette tape, via the MBL (or MBLe) PROM. The T command has the following form:

```
T
```

UBMON examines memory at address 177000 octal (FE00h), which is the first address in the MBL PROM. If this address contains anything except 377 octal (FFh), then UBMON assumes a valid MBL PROM is installed, and control is passed to the MBL PROM at this address to boot or load a file from paper tape or cassette tape.

If this address contains 377 octal (FFh), then UBMON assumes no MBL PROM exists, and the command is ignored.

### Entering Numeric Values

When waiting for an address or data, UBMON will accept only valid octal digits (0-7) and the “space” character. UBMON expects 6 digits for an address, and 3 digits for data. All of the expected digits need not be typed. The first “space” character terminates input and may be used to delimit separate inputs. If no digits have been typed before the delimiting space character, UBMON will assume a value of zero, except as noted above, when entering data for the M command.

Errors in numeric value entry can be corrected easily before the last character is typed. Simply enter a non-octal character and UBMON will print a question mark followed by its prompt. The command may then be typed again.

## 4. LOADING BASIC WITH UBMON

UBMON greatly speeds up the process of loading Altair BASIC, and can be used whether or not a loader PROM is installed.

### Without a Loader PROM

The usual procedure for loading BASIC involves toggling in a bootstrap loader program from the front panel and using it to load a paper tape or cassette tape version of BASIC. If the UBMON PROM is installed, the bootstrap loader can be entered from the terminal in octal instead of from the front panel switches in binary.

To do this, type M000000 (or M<space>) in response to UBMON’s prompt. After UBMON displays the current contents of the first address in memory, type the first entry in the “OCTAL DATA” column of the applicable loader program. After three digits are typed, UBMON closes the current address and opens the next address. (The loaders are found in Appendix B of the Altair BASIC Reference Manual.) Repeat this process until the entire bootstrap loader program is entered. The program can be checked by typing a non-octal character to return to UBMON, and then again typing M000000 (or M<space>). As the contents of each address are displayed, type a space to display the contents of the contents of the next address without making any modifications. Any mistakes can be corrected by typing a new octal value, instead of a space.

Once the loader program has been entered, position the paper tape or cassette tape of BASIC in the load device, and set the front panel switches according to the directions in the BASIC reference manual. Start the loader by typing J000000, and start the tape device. The terminal should print BASIC’s “MEMORY SIZE” initialization question after BASIC has loaded. At this point, BASIC is in control.

With a Standard Loader PROM

If you have installed a DBL, MDBL or CDBL PROM, then you can load BASIC directly from floppy disk by typing B at UBMON’s prompt.

If you have installed an MBL PROM, then you can load BASIC directly from paper tape or cassette tape by setting the front panel switches according to the Appendix B of the BASIC Reference Manual, and typing T at UBMON’s prompt. Wait about 5 seconds after typing T, and then start the tape transport.

If you have installed an HDBL PROM, then you can load BASIC directly from hard disk, by typing L at UBMON’s prompt.

With a Nonstandard Loader Program

You can execute any non-standard loader program by typing Jxxxxxx at UBMON’s prompt, where xxxxxx is the execution address of the loader program, in octal.

## 5. LOADING A FILE THAT WAS PUNCHED WITH UBMON

Like files punched with TURMON, files punched with UBMON can be loaded using the MBL PROM, using the T command. Start the tape anywhere in the leader portion that is punched with 40 octal (20h). (You can skip ahead and start loading from the null portion of the leader if you have the MBLe PROM instead of the MBL PROM.) Set the Altair front panel switches (or Turnkey Module sense switches for an Altair 8800b) correctly to load from your paper tape reader, as described in Appendix B of the Altair BASIC Reference Manual or in the MBL manual.

Note that MBL takes about 5 seconds to initialize, so do not start the tape transport until about 5 seconds after you type T.

MBL is written to skip over a checksum loader at the beginning of a tape - such as an Altair Basic tape. On such a tape, the leader character is the length (in bytes) of the checksum loader, so this many bytes get skipped. The 40 octal (20h) leader that UBMON produces will cause the first 32 bytes of the null leader to be “skipped”, before hunting for the first Altair Binary Absolute Load record.

Note that TURMON punches 15 octal (0Dh) instead of 40 octal at the beginning of its leader, causing 13 bytes of the nulls to be “skipped.” This will have no effect on file loading.

# APPENDIX A - ALTAIR BINARY ABSOLUTE LOAD FORMAT

The Altair Binary Absolute Load Format comprises a series of Records. The format defines several Record types, although only one Record type (the Program Load Record) is ever written by UBMON.

The Altair MBL PROM code recognizes the End-of-File Record as well, using it to initiate execution after a successful file load. Begin/Name Records are always ignored.

The following Record types are defined:

Begin/Name Record (ignored by MBL, not supported by UBMON or TURMON)

| Byte # | Contents | Comments |
|---|---|---|
| 1 | 125 Octal(ASCII U) | Begin Sync Byte |
| 2-4 | Name | Program Name |
| 5-N | Comments | Program version and date, etc.<br>NO CRs allowed. |
| N+1 | 15 Octal (ASCII CR) | Terminates Begin/Name Record |

Program Load Record

| Byte # | Contents | Comments |
|---|---|---|
| 1 | 74 Octal (ASCII <) | Load Sync Byte |
| 2 | 0-377 Octal | Number of load bytes |
| 3 | Least-significant byte | of load address |
| 4 | Most-significant byte | Of load address |
| 5-N | Data bytes | |
| N+1 | Checksum byte | Sum of all bytes except the first 2 bytes, with carries discarded |

End-of-file Record (not supported by UBMON or TURMON)

| Byte # | Contents | Comments |
|---|---|---|
| 1 | 170 Octal (ASCII x) | EOF Sync Byte |
| 2 | Least-significant byte | of execution start address |
| 3 | Most-significant byte | of execution start address |

# APPENDIX B - UBMON ENTRY POINTS

UBMON has several software subroutines that are at the same addresses and compatible with equivalent subroutines in the Altair TURMON PROM. The entry points for subroutines in UBMON are as follows:

| Address | | | |
|---|---|---|---|
| Octal | Hex | Name | Function |
| 176400 | FD00 | MONTOR | Cold-start entry for UBMON. Destroys all registers and the stack. |
| 177750 | FFE8 | INCH | Get, strip parity, and echo a Terminal character. Result is in A, all registers except the flags are preserved. |
| 177762 | FFF2 | OUTCH | Print A on the Terminal. All registers except the flags are preserved. (Note: TURMON also adds the value in A to C for checksum calculation. UBMON does not.) |
| 177717 | FFCF | PRINT3 | Convert binary value in H into octal and print it as 3 octal digits on the Terminal, followed by a space. Destroys all registers and flags except D and E. |
| 177743 | FFE3 | SPACE | Print a space on the Terminal. All registers except the PSW (A and the flags) are preserved. |

Note that if the UBMON PROM is installed in a Turnkey Module that has the Altair-style “Phantom” rework, then UBMON will become disabled upon the first read of the Altair’s Sense Switches (Port 377 octal) by any program, and the above routines will no longer be available.

## APPENDIX C - SOURCE CODE LISTING

The following pages list the source code for UBMON. This code was assembled using Digital Research’s ASM assembler. As such, all values are in hexadecimal, rather than in octal as is normal for MITS software.

```
;***************************************************************
;* UBMON - UNIVERSAL BOOT/MONITOR          BY MARTIN EBERHARD  *
;*                                                             *
;* UBMON a 256-byte PROM monitor for use with the Altair 8800b *
;* Turnkey Module or the 88-PMC PROM card. UBMON is very       *
;* similar in operation to the Altair TURMON PROM monitor,     *
;* with the addition of 3 commands to boot from the various    *
;* Altair boot devices, including floppy disks, tapes, and     *
;* hard disk. UBMON provides the following functions:          *
;*                                                             *
;*  M xxxxxx  Memory Examine and Modify: Allows you to examine *
;*            and change the contents of memory starting at    *
;;*           address xxxxxx octal                             *
;*                                                             *
;*  D xxxxxx yyyyyy   Memory Dump: Dumps memory contents in    *
;*            Altair binary punch format from address xxxxxx   *
;*            to address yyyyyy, inclusive (octal addresses)   *
;*                                                             *
;*  J xxxxxx  Jump To Adddress xxxxxx octal                    *
;*                                                             *
;*  B         Boot from floppy: Requires the DBL, the MDBL, or *
;*            the CDBL PROM at address FF00h                   *
;*                                                             *
;*  T         Boot from Tape or Load Tape File: Requires the   *
;*            MBL PROM at address FE00h                        *
;*                                                             *
;*  L         Boot from Hard Disk: Requires a hard-disk PROM   *
;*            (e.g. HDBL) at address FC00h                     *
;*                                                             *
;* Differences compared to TURMON functionality:               *
;* 1) UBMON's 'D' command's leader is a string of 60 bytes of  *
;*    20h, followed by a string of 60 nulls, while TURMON      *
;*    punches a string of 60 bytes of 0Dh followed by a string *
;*    of 60 nulls.  Both of these allow MBL to load the file   *
;*    by fooling it to think it is skipping a checksum loader. *
;* 2) UBMON's 'D' command also follows the memory dump with a  *
;*    trailer of 60 nulls, while TURMON prints no trailer.     *
;*                                                             *
;* As with TURMON, the Terminal for UBMON is a 6850-based      *
;* serial port at address 020 octal (10h). An 88-2SIO, as well *
;* as the serial ports on the Turnkey Monitor and the 88-UIO,  *
;* are all suitable.                                           *
;*                                                             *
;* Also like TURMON, UBMON requres RAM for its stack. Several  *
;* assembly options support various memory and PROM board      *
;* configurations.                                             *
;*                                                             *
;* This monitor provides subroutine compatibility with TURMON  *
;* at the following entry points:                              *
;*                                                             *
;* Address  Name   Function                                    *
;*  FD00h  MONTOR  Cold-start entry to the monitor             *
;*  FD08h  ENTER   Warm start, does not init ACIA              *
;*                 Result in A. All regs except F preserved.   *
;*  FDF2h  XOUTCH  Print A. All regs except F preserved.       *
;*  FDCFh  XPRNT3  Print value in H in octal, followed by a    *
;*                 space. Trashes AF,BC,HL.                    *
;*  FDE3h  XSPACE  Print a SPACE. Trashes AF.                  *
;*  FDE8h  XINCH   Get, strip parity & echo a character        *
;***************************************************************
```

```
               ; REVISION HISTORY
               ;
               ; Ver. 1.00  12 SEP 2013 M.Eberhard
               ;   Created from MITS TURMON
               ;
               ; Ver. 1.01  16 SEP 2013 M. Eberhard
               ;   Hosed version. Released only for about 1 hour - hopefully
               ;   no distribution.
               ;
               ; Ver. 1.02  17 SEP 2013 M. Eberhard
               ;   Fixed. Improved comments. Squeeze code more, and add trailer
               ;   after 'D' command. Set leader and trailer to 60 bytes
               ;
               ; Ver. 1.03  17 AUG 2014 M. Eberhard
               ;   Punch 60 bytes of 20h before leader, so that file can also
               ;   be loaded by MITS's MBL loader PROM. This required some
               ;   further code compression.

               ;***************************************************************
               ;* NOTES                                                       *
               ;*                                                             *
               ;* The assembler will not check for overlapping code. If you   *
               ;* make any changes, be sure to check that your new code does  *
               ;* not overlap fixed-location subroutines near the end.        *
               ;*                                                             *
               ;* Forcing this code to fit into one 256-byte PROM required    *
               ;* making some assumptions about the address of this code, as  *
               ;* well as the addresses of the  DBL, MBL, and HDBL PROMS. if  *
               ;* you change the address of any of the PROM programs, then    *
               ;* this program will require some changes. Fortunately, the    *
               ;* addresses of these PROMs haven't changed in 38 years.       *
               ;*                                                             *
               ;* Apologies for the somewhat convoluted code. Squeezing an    *
               ;* additional 3 commands (on top of TURMON's 3 commands) into  *
               ;* a 256-byte PROM space, while also keeping the externally-   *
               ;* accessible subroutines at their historical addresses        *
               ;* required some twisted code. Hopefully my comments will help *
               ;* explain how it works. Be careful if you change this code.   *
               ;*                                                -M. Eberhard *
               ;***************************************************************

               ;-------------------------------------
               ;Stack address options
               ; (All but one must be commented out)
               ;-------------------------------------
               ;STACK  equ     0FC00h          ;UBMONt Turnkey board 1K RAM
F800 =         STACK   equ     0F800h          ;UBMONp Last RAM before 88-PMC
               ;STACK  equ     0E000h          ;UBMONb RAM before 8K Bytesaver

               ; Program Equates

003C =         LDRLEN  equ     60              ;'D' cmd leader/trailer length
002E =         PROMPT  equ     '.'             ;Command prompt
003F =         ERRMSG  equ     '?'             ;1-byte error message

               ; Terminal port equates - same for 88-2SIO port 0, Turnkey
               ; Module, and 88-UIO (all based on the Motorola 6850 ACIA)

0010 =         ACCTRL  equ     10h             ;ACIA Control output port
0010 =         ACSTAT  equ     10h             ;ACIA Status input port
0011 =         ACTXD   equ     11h             ;ACIA TX Data register
0011 =         ACRXD   equ     11h             ;ACIA RX Data register

0003 =         ACRST   equ     00000011b       ;Master reset
```

```
0001 =         ACRDF   equ     00000001b       ;RX Data register full
0002 =         ACTDE   equ     00000010b       ;TX Data register empty
0011 =         ACINIT  equ     00010001b       ;/16, 8bit, No Parity, 2Stops

               ; Altair File Equate

003C =         LBSYNC  equ     3CH     ;Altair file Load block synch chr

               ; Fixed locations in UBMON, for TURMON compatibility
               ; MONTOR must be FD00h, because FDh is also -3.

FD00 =         MONTOR  equ     0FD00h          ;Execution beginning
FDCF =         XPRNT3  equ     0FDCFh          ;Print h in octal on Terminal
FDE3 =         XSPACE  equ     0FDE3h          ;Print a space on the Terminal
FDE8 =         XINCH   equ     0FDE8h          ;Get Terminal chr in A
FDF2 =         XOUTCH  equ     0FDF2h          ;Print A on Terminal

               ; Code enrty points in external PROMs. The low address byte of
               ; these all must be 00. FBOOT must be 100h greater than TBOOT.

FC00 =         HBOOT   equ     0FC00h          ;HDBL boot ROM
FE00 =         TBOOT   equ     0FE00h          ;MBL multi-tape boot ROM
FF00 =         FBOOT   equ     0FF00h          ;DBL or MDBL floppy boot ROM

               ; ASCII characters

000D =         CR      equ     0Dh
000A =         LF      equ     0Ah

               ;*****************
               ;* Start of Code *
               ;*****************
FD00                   ORG     MONTOR

               ;-------------------------
               ; Initialize Terminal ACIA
               ;-------------------------
FD00 3E03              mvi     a,ACRST
FD02 D310              out     ACCTRL
FD04 3E11              mvi     a,ACINIT
FD06 D310              out     ACCTRL

               ;===Main Loop=========================================
               ;Get Command a command and dispatch.
               ; This portion of the code is particulary
               ; convoluted, to keep it short. Here we assume:
               ;    high(MAIN)=FDh
               ;    high(TBOOT)=FEh
               ;    low(HBOOT)=low(TBOOT)=low(FBOOT)=0
               ;    high(FBOOT)=high(TBOOT)+1
               ;    low(EXTCMD)<=FDh
               ;=====================================================
FD08 3100F8    MAIN:   lxi     sp,STACK        ;create stack
FD0B 0108FD            lxi     b,MAIN          ;create return address
FD0E C5                push    b               ;..also set b=high(EXTCMD)

               ; Print prompt on a new line

FD0F CDA1FD            call    PCRLF           ;d=0ah, e=0dh
FD12 3E2E              mvi     a,PROMPT
FD14 CDF4FD            call    PRINTA

FD17 2100FC            lxi     h,HBOOT         ;Entry address for HDBL PROM
```

```
                                               ;also l=0 for GETADR

                ; Get user input and dispatch command
                ; On Entry:
                ;  b = high(EXTCMD) = FDh = -3
                ;  d = 0Ah (10)
                ;  e = 0Dh (13)
                ; hl = HBOOT = FC00h
                ;  l = 0

FD1A CDE8FD            call    GETCHR          ;Get user command

FD1D D64A              sui     'J'             ;Jump command?
FD1F CA81FD            jz      DOJCMD

FD22 80                add     b               ;(b=FDh=-3) 'M' command?
                                               ;c<FDh, and l=0 here
FD23 CCACFD            cz      GETADR          ;M: get hl=addr from user, set Z
FD26 CA86FD            jz      DOMCMD          ;M: do M command

                ; The next 3 commands all go through EXTCMD. Code is
                ; shorter if we use conditional returns, rather than
                ; conditional jumps.
                ; On Entry:
                ;  a = user input - 'J' - 3 = user input - 'M'
                ;  b = high(EXTCMD)
                ;  d = 0Ah (10)
                ; hl = HBOOT = FC00h
                ;  l = 0

FD29 0E7F              mvi     c,EXTCMD AND 0FFh ;bc=EXTCMD
FD2B C5                push    b               ;prepare for conditional returns

FD2C 3C                inr     a               ;'L' command?
                                               ;hl=HBOOT here
FD2D C8                rz                      ;L: jump to EXTCMD

                ;  a = user input - 'J' - 3 + 1 = user input - 'L'
                ;  d = 10
                ;  l = 0

FD2E 26FF              mvi     h,FBOOT/256     ;entry address for DBL/MDBL PROM
FD30 82                add     d               ;d=10 = -('B'-'L')
                                               ;Boot from floppy cmd?

FD31 C8                rz                      ;B: jump to EXTCMD

                ;  a = user input - 'J' - 3 + 1 + 10 = user input - 'B'
                ; hl = FBOOT = FF00
                ;  l = 0

FD32 25                dcr     h               ;hl=TBOOT: address for MBL PROM
FD33 FE12              cpi     ('T'-'B')       ;boot from tape cmd?
FD35 C8                rz                      ;T: jump to EXTCMD

FD36 C1                pop     b               ;done with EXTCMD commands

                ;  a = user input - 'J' - 3 + 1 + 10 = user input - 'B'
                ; hl = TBOOT = FE00h
                ;  h = FEh = -2
                ;  l = 0

FD37 84                add     h               ;'D' Dump memory cmd?
```

```
FD38 C0                rnz                     ;anything else is invalid

               ;===Command Routine==============================
               ; Process D (dump memory) Command
               ; Punch specified address range as
               ; an Altair-style data file
               ; on entry:
               ;   c=low(EXTCMD) < 0FEh
               ;   l=0
               ;================================================

               ; Get the start & end addresses from the user

FD39 5D                mov     e,l             ;e=0 too, for 2nd GETADR
FD3A CDACFD            call    GETADR          ;Get start addr, print space
FD3D EB                xchg                    ;de=start address
FD3E CDACFD            call    GETADR          ;Get hl=end address
                                               ;returns with a=20h

FD41 23                inx     h               ;hl = one past end address

               ; Punch a pre-leader so that MITS's MBL can load this file
               ; a=20h here.

FD42 47                mov     b,a             ;punch 20h as the pre-leader
FD43 CD75FD            call    LEADER          ;returns b=0

               ; Punch null leader

FD46 AF                xra     a               ;Punch null leader
FD47 CD75FD            call    LEADER          ;returns with b=0

               ; Loop to punch all the requested data
               ; (b=0 here, both on initial entry and upon looping)

               ; Compute b=data byte count of the next block, max=255

FD4A 05        NXTBLK: dcr     b               ;b=FFh=255

FD4B 7D                mov     a,l             ;compute least sig byte
FD4C 93                sub     e
FD4D 4F                mov     c,a             ;save least sig byte
FD4E 7C                mov     a,h             ;compute most sig byte
FD4F 9A                sbb     d               ;>255 bytes remaining?
FD50 C254FD            jnz     BLKSIZ          ;y: then do 255 bytes
FD53 41                mov     b,c             ;n: byte count = lsb
               BLKSIZ:

               ; Punch the the block header info:
               ; sync chr, byte count, & 2-byte load address
               ;  b = block size
               ; de = starting memory address for block data
               ; hl = last address of file + 1

FD54 D5                push    d               ;save load address

FD55 1E3C              mvi     e,LBSYNC        ;Punch load-block sync chr
FD57 50                mov     d,b             ;and block byte count
FD58 CDA4FD            call    PRNTDE

FD5B D1                pop     d               ;restore load address
FD5C CDA4FD            call    PRNTDE          ;Punch de=load address
                                               ;ends with a=d
```

```
FD5F 83                add     e               ;a=checksum of the address

               ; Punch b bytes of block data, computing checksum as we go
               ;  a = checksum so far
               ;  b = block size
               ; de = starting memory address for block data
               ; hl = last address of file + 1

FD60 4F        BDATLP: mov     c,a             ;temp save checksum
FD61 1A                ldax    d               ;get memory data
FD62 CDF4FD            call    PRINTA          ;...and punch it

FD65 81                add     c               ;update checksum

FD66 13                inx     d               ;Next address
FD67 05                dcr     b               ;Loop 'til done with block data
FD68 C260FD            jnz     BDATLP          ;ends with b=0

               ; a = block checksum
               ; b = 0

FD6B CDF4FD            call    PRINTA          ;Punch the block checksum

               ; Continue until all the data has been punched
               ;  b = 0
               ; de = next address to punch
               ; hl = last address of file + 1
               ; Test for de<hl, meaning there are more bytes to punch

FD6E 7B                mov     a,e             ;compute de-hl
FD6F 95                sub     l
FD70 7A                mov     a,d
FD71 9C                sbb     h               ;set carry if hl>de
                                               ;ends with hl=de so a=0

FD72 DA4AFD            jc      NXTBLK          ;Y: Do another block

               ; Fall into LEADER (with a=0) to punch the trailer

               ;---Subroutine---------------------
               ; Punch a leader
               ; On Entry:
               ;   a = leader character
               ; On exit:
               ;   b=0
               ;   all other registers preserved
               ;----------------------------------
FD75 063C      LEADER: mvi     b,LDRLEN        ;leader length

FD77 CDF4FD    LEADLP: call    PRINTA
FD7A 05                dcr     b
FD7B C277FD            jnz     LEADLP          ;ends with b=0

FD7E C9                ret

               ;===Command Routine==================================
               ; Process External Command (B,L,T), only if the PROM
               ; for that command appears to exist
               ; Note: this assumes ROM - i.e. writing does
               ; not effect contents.
               ; On entry:
               ;   hl=start address of external command
               ; On branch to external command:
```

```
               ;   Top of Stack = MAIN (for potential return)
               ;====================================================
FD7F 34        EXTCMD: inr     m               ;Does location = FF?
FD80 C8                rz                      ;Y:ignore command

               ; Fall into DOJCMD, with Z cleared

               ;===Command Routine==================================
               ; Process J (jump) Command
               ; On Entry:
               ;   C=low(EXTCMD) << 0FFh
               ;   l=0
               ;   Z flag set
               ;====================================================
FD81 CCACFD    DOJCMD: cz      GETADR          ;Get jump address if J
FD84 E9                pchl                    ;and go there

               ;===Command Routine==================================
               ; Process M (Memory examine and edit) Command
               ; On Entry at (DOMCMD):
               ;   hl = start address from GETADR
               ;====================================================
FD85 23        MMODLP: inx     h               ;loop for next address

FD86 CDA1FD    DOMCMD: call    PCRLF           ;on a new line

FD89 E5                push    h               ;save memory address

               ; Print address (in hl) and data (at memory address hl)

FD8A CDC8FD            call    PADRDA          ;sets l=0 too

               ; Get user data, carry clear if space (means don't change data)

FD8D CDADFD            call    GETDAT          ;l=user data, carry means none
FD90 7D                mov     a,l             ;a=8-bit user data

               ; Prepare to use conditional returns, recover memory address

FD91 2185FD            lxi     h,MMODLP        ;addr for conditional returns
FD94 E3                xthl                    ;..recover hl=memory address
FD95 D0                rnc                     ;no carry means user typed space

               ; Modify memory, and test it

FD96 77                mov     m,a             ;write new data
FD97 BE                cmp     m               ;Verify write
FD98 C8                rz                      ;Next address if okay

               ; Fall into BADINP to report memory-write failure

               ;---Subroutine Abort----------------------------
               ; Bad input from user or memory write failure.
               ; Print error message and start over. There is
               ; junk on the stack, which MAIN will repair.
               ;-----------------------------------------------
FD99 3E3F      BADINP: mvi     a,ERRMSG        ;Error message
FD9B CDF4FD            call    PRINTA
FD9E C308FD            jmp     MAIN

               ;---Subroutine--------------------
               ; Print Carriage Return/Line Feed
               ; On Exit:
                ;   d=0Ah
                ;   e=0Dh
                ; trashes a
                ;---------------------------------
 FDA1 110D0A    PCRLF:  lxi     d,LF*256+CR     ;load de with CR LF

                ; Fall into PRNTDE

                ;---Subroutine---------------
                ; Print E then D
                ; On Exit:
                ;   a=d
                ;----------------------------
 FDA4 7B        PRNTDE: mov     a,e
 FDA5 CDF4FD            call    PRINTA
 FDA8 7A                mov     a,d
 FDA9 C3F4FD            jmp     PRINTA

                ;---Subroutine----------------------------------------------
                ; Get 3 or 6 octal digits from user, and then print a space.
                ;
                ; Entry at GETADR gets a 6-digit address
                ; Entry at GETDAT gets a 3-digit data value
                ; A space typed at any time terminates the get
                ; with all upper digits defaulting to 0.
                ; On Entry:
                ;  c<FEH so that entry at GETADR will not bump B
                ;  l=0
                ; On Exit:
                ;   a = 20h (space character)
                ;   l = 8-bit binary value of input
                ;  hl = 16-bit binary value of input
                ;   b trashed.
                ;   c incremented only for entry at GETADR.
                ;   Z flag set
                ;   Carry flag cleared if user typed a space
                ;------------------------------------------------------------
 FDAC 06        GETADR: db      06      ;Entry here: '0606 mvi b,06'
                                        ;            '03   inx b' (increments c)

 FDAD 06        GETDAT: db      06      ;Entry here: '0603 mvi b,03'
 FDAE 03                db      03

 FDAF 65                mov     h,l             ;hl=0 init value for <6 digits

 FDB0 CDE8FD    GETNXT: call    GETCHR          ;get a digit, Z set if space
 FDB3 C8                rz                      ;return with carry clear if

space

 FDB4 D630              sui     '0'             ;subtract ASCII offset
 FDB6 FE08              cpi     8               ;valid octal digit?
 FDB8 D299FD            jnc     BADINP          ;N: error

 FDBB 29                dad     h               ;shift new value into hl
 FDBC 29                dad     h
 FDBD 29                dad     h
 FDBE B5                ora     l               ;install new digit
 FDBF 6F                mov     l,a

 FDC0 05                dcr     b               ;more digits to get?
 FDC1 C2B0FD            jnz     GETNXT          ;Y: go get it.
```

```
FDC4 37                stc                     ;carry set: complete input
FDC5 C3E3FD            jmp     PSPACE          ;follow with a space
                                               ;preserve psw

               ;---Subroutine--------------------------------------------
               ; Print address in hl, a space, and then the data at (hl)
               ; On Exit:
               ;   a=20h (space)
               ;   b=0
               ;   c=memory data
               ;   h=memory data
               ;   l=0
               ;---------------------------------------------------------
FDC8 4E        PADRDA: mov     c,m             ;remember memory data

               ; Print address in hl as 6 octal digits

FDC9 0606              mvi     b,6             ;6 digits in PR6OCT
FDCB AF                xra     a               ;initial value for PR6OCT
FDCC CDD4FD            call    PR6OCT          ;Print HL in octal
                                               ;returns with h=c

               ; Fall into PR3OCT to Print data in h=c as 3 octal digits

               ;**Fixed-Location Externally Accessable Subroutine**************
               ;
               ; Print octal digits followed by a space
               ;
               ; Entry at PR3OCT will print 3 octal digits from h.
               ; The first digit is from the 2-bit value in the bits 7:6 of c.
               ; Entry at PR6OCT will print 6 octal digits from hl.
               ; the first digit is from the 1-bit value in the bit 7 of h.
               ; Subsequent digits are 3-bit values in decending order from hl.
               ;
               ; On Entry at PR3OCT:
               ;   h = value to print
               ;
               ; On Entry at PR6OCT:
               ;   b = 6 (digit count)
               ;   a = 0
               ;   hl = value to print
               ;
               ; On Exit:
               ;   a=20h (space)
               ;   b=0
               ;   h=c
               ;   l=0
               ;***************************************************************
FDCF                   org     XPRNT3

               ; Entry for 8-bit value: 1st digit shifts only twice

FDCF AF        PR3OCT: xra     a               ;initial value
FDD0 0603              mvi     b,3             ;3 digits

FDD2 29        OCTLUP: dad     h               ;2nd and 3rd shifts
FDD3 17                ral

               ; Entry for 16-bit value: 1st digit shifts only once

FDD4 29        PR6OCT: dad     h
FDD5 17                ral
```

```
FDD6 C630              adi     '0'             ;Convert to ASCII

FDD8 CDF4FD            call    PRINTA          ;Print digit
FDDB AF                xra     a               ;start new digit

FDDC 29                dad     h               ;1st shift for next digit
FDDD 17                ral

FDDE 05                dcr     b               ;More digits?
FDDF C2D2FD            jnz     OCTLUP          ;N: the next digit is 3 bits

FDE2 61                mov     h,c             ;get memory data
                                               ;(this is doen here to use
                                               ;..a memory location.)
               ; Fall into PSPACE

               ;**Fixed-Location Externally Accessable Subroutine**************
               ;
               ; Print a space
               ;
               ; On Exit:
               ;   a = space = 20h
               ;   all other registers preserved
               ;***************************************************************
FDE3                   org     XSPACE

FDE3 3E20      PSPACE: mvi     a,' '
FDE5 C3F4FD            jmp     PRINTA

               ;**Fixed-Location Externally Accessable Subroutine**************
               ;
               ; Get & echo a character from the Terminal
               ;
               ; On Exit:
               ;    a = received character, with parity stripped
               ;    Z is set if the character is a space
               ;***************************************************************
FDE8                   org     XINCH

FDE8 DB10      GETCHR: in      ACSTAT  ;Wait for chr available
FDEA 0F                rrc             ;test bit 0=S2DS1
FDEB D2E8FD            jnc     GETCHR

FDEE DB11              in      ACRXD   ;Get the chr
FDF0 E67F              ani     7Fh     ;strip parity

               ;  Fall through to XOUTCH to echo

               ;**Fixed-Location Externally Accessable Subroutine**************
               ;
               ; Print A on the Terminal
               ;
               ; (The cpi below is here to make to do the routine the right
               ; length, doing a test that we need for GETCHR anyway.)
               ; On Exit:
               ;    Z set if printed chr is a space
               ;    All other registers preserved
               ;***************************************************************
FDF2                   org     XOUTCH

FDF2 FE20              cpi     ' '     ;is it a space?

               ; Fall into PRINTA to print
               ;---Subroutine---------------
               ; Print A on the Terminal
               ; All registers preserved
               ;----------------------------
FDF4 F5        PRINTA: push    psw     ;temp save chr

FDF5 DB10      PAWAIT: in      ACSTAT  ;Wait for ACIA TX to be ready
FDF7 E602              ani     ACTDE
FDF9 CAF5FD            jz      PAWAIT

FDFC F1                pop     psw     ;recover chr, flags
FDFD D311              out     ACTXD   ;send chr

FDFF C9                ret             ;with chr in a

FE00                   END
```