altair™ 8800b Turnkey

PROM Monitor User's Guide

mits

a subsidiary of **Pertec Computer Corporation**

altair 8800b Turnkey

PROM Monitor User's Guide

©MITS, Inc. 1977
First Printing August, 1977

mits
a subsidiary of **Pertec Computer Corporation**
2450 Alamo S.E. /Albuquerque, New Mexico 87106

TABLE OF CONTENTS

1. ABSTRACT

This document describes the function and operation of the Altair 8800b Turnkey PROM Monitor. The PROM Monitor is a system program that allows the user to examine and change any memory location or series of locations, punch the contents of any range of memory locations in Altair Absolute Load Tape format and start execution of a program at any specified address. A source listing of the 8800b Turnkey PROM Monitor is provided so that its I/O and octal conversion routines can be used in other programs.

2. STARTING THE PROM MONITOR

a) The Monitor PROM must be installed in PROM socket K1 on the Turnkey Module.

b) The AUTO-START address switches on the Turnkey Module must be set to 176400 octal and the PROM address switches to 174000 octal.

c) Turn power on.

d) The PROM Monitor prints its prompt character, a period (.).

e) At any time, pressing the START switch causes control to return to the Monitor and the prompt to be printed.

NOTE

The input routines in the PROM monitor will accept only valid octal digits (0-7) and the "space" character. When waiting for input, the routines expect either three or six digits. All of the expected digits need not be input. The first space character terminates the input routine and may be used to delimit separate inputs. If no digits have been entered before the delimiting space is entered, the input routine will return a value of zero. Whenever the delimiting space is used, the carry bit is set, and the return is made. During a normal return (i.e., one in which no space was used), the carry bit is always clear.

3. OPERATION

The Prom Monitor has three commands:

M	Memory examine and change
D	Memory dump
J	Jump to user program

a) The M command. The M command allows the user to examine and change any location in the Altair 8800b memory. The form of the M command is as follows:

Mxxxxxx

where xxxxxx stands for from zero to six valid octal digits. The PROM Monitor opens the location specified and displays the three digit octal contents of that location. The Monitor then waits for three valid octal digits. Three complete octal digits must be input; the space character cannot be used as a delimiter in this case. When this valid data has been received, the Monitor attempts to place the data into the opened location. Once the deposit has been made and verified, the M function closes the current location and opens the following location. If the user tries to deposit information into nonexistent memory, ROM, or protected RAM, the bad deposit causes "?" to be printed on the terminal and control to return to the Monitor. Assuming a valid deposit, this sequence continues until a non-valid character (any character except the digits 0-7) is input. This non-valid character is flagged with a "?" and control returns to the Monitor. This is the normal way to return to the Monitor.

If a space is input instead of a valid octal character, the M function closes the present location without making any changes and then opens the next consecutive location. While the M command is looking for input, the space character may be used at any time to close the current location without change, and open the following location. Therefore, even though one or two valid octal digits may have been input, when the space has been received, the location is closed without change. To deposit new data, three complete valid octal digits must be input.

b) The D command. The D command allows the user to dump the contents of the Altair 8800b's memory between any two locations. The D command has the following form:

Dxxxxxx xxxxxx

To use the D command, type a D in response to the Monitor's prompt character. The D function will then wait for the starting address (zero to six valid octal digits). If six digits are input, the D function prints a space and then waits for the ending address (zero to six valid octal digits). The ending address must be greater than or equal to the starting address. If less than six digits are input during the starting address, the D function echoes the delimiting space character, but does not print one of its own.

Once the D function has received valid starting and ending addresses, it punches a leader of 60 octal 302's followed by 60 nulls (zero bytes). It then punches out the contents of memory starting at the first address up to and including the end address in the Altair 8800b binary Absolute Load Tape format, as shown in Table A. (The word "punch" is used here to refer to the output of the D command, no matter what output device is actually used.) If the number of bytes to be punched is greater than 377 octal, the D function punches as many blocks of 377 octal bytes as necessary until the number of bytes left to punch is less than 377 octal bytes. The last block punched may have less than 377 octal bytes. If the number of bytes to be punched in the last block is equal to zero, a zero block is not punched. Upon completion of the dump, the D function performs a carriage return and line feed and then returns to the Monitor.

c) The J command. The J command allows the user to transfer control between the monitor and another program. The J command has the following form:

Jxxxxxx

where xxxxxx is the starting address of the user routine (zero to six valid octal digits). Once the J function has received a valid address, it will load the program counter with the address and start execution of the user program at that address.

4. MEMORY SPACE AND STACK CONSIDERATIONS

The PROM Monitor is 256 decimal or 377 octal bytes long and is assembled to operate with a starting address of 176400 octal. It must be located at this point in memory or it will not operate correctly. The PROM Monitor establishes a stack with a top address of 176000 octal when it is entered. The Monitor never has more than four levels of subroutine calls at any one time, so only eight bytes are actually used in the stack. The stack itself usually resides in the 1K of RAM that is part of the Turnkey Module. It is the user's responsibility to see that there is RAM available at the stack location. Otherwise, the Monitor cannot operate correctly, if at all.

All necessary registers and the stack pointer should be saved before jumping from a program to the Monitor, since the Monitor destroys the contents of the stack pointer and all registers upon entry. Restoration of the registers must be handled by the user's program.

5. ERROR CONSIDERATIONS

Errors in data input can be corrected easily before the last character is typed. Simply type a non-octal character (except space) and the monitor will print a question mark and a period. The command may then be typed again.

When the octal input routines are requesting input, they do not check for over-range conditions on the input data. For example, when using the M function, three complete valid octal digits must be input in order to deposit new data into a memory location. Since the Altair 8800b is organized around an eight bit byte, the largest valid octal number that can be input is 377. In fact, 777 can be input without the Monitor detecting an error. The actual value that is deposited in the memory location in that case is not equal to 777 octal, but depends upon the binary representation of the most significant digit input to the routine. For example, 477 causes the routine to deposit octal 077 into the memory location. The same possible error condition is present when addresses are input, except that the maximum value that may be typed is 1777777. Anything larger will not be flagged as an error, but the effective address will depend upon the binary representation of the highest order digit.

6. RUNNING BASIC WITH THE PROM MONITOR

The Altair 8800b PROM Monitor greatly speeds the process of loading Altair BASIC and can be used whether or not the Multi-Boot Loader or Disk Boot Loader PROMs are in use.

A. Without the Loader PROMS. The usual procedure for loading BASIC involves toggling a loader program in from the front panel and using it to load a paper tape or cassette version of BASIC. If the PROM Monitor is installed, this bootstrap loader can be entered from the terminal in octal instead of from the front panel switches in binary.

To do this, type MOOOOOO (or M <space>) in response to the Monitor's prompt. After the Monitor displays the current contents of the first location in memory, type the first entry in the "OCTAL DATA" column in the applicable loader program. The loaders are found in Appendix B of the Altair BASIC Reference Manual. After three digits are typed, the Monitor closes the current location and opens the next location. This process is repeated until the entire loader program is entered. The program can be checked by typing a non-octal character to return to the Monitor and again typing MOOOOOO (or M <space>). As the contents of each location are displayed, typing a space causes the Monitor to display the contents of the next location without making any modifications.

Once the loader program has been entered and verified, the paper tape or cassette tape of BASIC is loaded and positioned in the load device according to the directions in the BASIC Reference Manual. Then the loader is started by typing J000000 (or J <space>). The terminal should print BASIC's "MEMORY SIZE?" initialization question after BASIC has been loaded. At that point, BASIC is in control.

B. With a bootstrap loader PROM. If either the Multi-Boot Loader or Disk Boot Loader PROM is installed, the response to the Monitor's prompt should be Jxxxxxx, where xxxxxx is the starting address for the loader in use. For the Multi-Boot loader, the starting address is 177000. For the Disk Boot Loader, the starting address is 177400. For more information, see the Multi-Boot Loader Manual and the Altair 8800 BASIC Reference Manual.

TABLE A

ABSOLUTE LOAD TAPE FORMAT

Begin/Name Record

Byte #	Contents	Comments
1	125 Octal	Begin Sync
2-4	Name	Program name
5-N	Comments	Program version and date, etc.
N+1	15 Octal	Terminates program name record

Program Load Record

Byte	Contents	Comments
1	74 octal	Load sync byte
2	0-377 octal	Number of load bytes
3	L.S. Byte	of Load address
4	M.S. Byte	of Load address
5-N	Data Bytes	
N+1	Checksum Byte	Generated by adding all bytes except the first two without carry

End-of-file record

Byte	Contents	Comments
1	170 octal	Paper tape/Audio Cassette EOF
2-3		Execution start address

```
00010   ;*****************************************************************
00020   ;*                                                               *
00030   ;*  THIS IS A 256 BYTE PROM MONITOR FOR USE WITH THE ALTAIR       *
00040   ;*  8800B TURNKEY MODULE.  THIS MONITOR PROVIDES THE USER WITH    *
00050   ;*  THE FOLLOWING FUNCTIONS:                                     *
00060   ;*                                                               *
00070   ;*      1)  MEMORY EXAMINE AND CHANGE FUNCTION                   *
00080   ;*              YOU CAN EXAMINE AND CHANGE THE CONTENTS OF ANY   *
00090   ;*              VALID MEMORY LOCATION                            *
00100   ;*      2)  MEMORY DUMP FUNCTION                                 *
00110   ;*              YOU CAN DUMP IN THE ALTAIR BINARY PUNCH FORMAT   *
00120   ;*              BETWEEN ANY TWO VALID MEMORY LOCATIONS           *
00130   ;*      3)  JUMP TO FUNCTION                                     *
00140   ;*              YOU CAN CAUSE THE MONITOR TO JUMP TO ANY         *
00150   ;*              LOCATION AND START EXECUTING THE PROGRAM THERE   *
00160   ;*                                                               *
00170   ;*  THE MONITOR CAN BE REENTERED FROM THE USER'S PROGRAM          *
00180   ;*  SO THAT THE FEATURES OF THE MONITOR ARE ALWAYS AVAILABLE      *
00190   ;*  TO ANY USER PROGRAM.                                         *
00200   ;*                                                               *
00210   ;*****************************************************************
00220   ;
00230   TITLE TURMON - MITS TURNKEY MONITOR PROM
00240   ;
00250   ;MITS TURNKEY MONITOR
00260   ;C.W.  VERTREES          01/13/77
00270   ;REVISED                 01/17/77
00280   ;                        01/19/77
00290   ;                        01/20/77
00300   ;
00390   ;CONSTANTS
176000  00400   STACK=176000
```

```
                00010   ;
                00020   ;MONITOR STARTS AT THIS LOCATION
                00030   ;BEGINNING OF PROM
                00040   ;
                00050   ;MONITOR CONTROL STRUCTURE
                00060   ;
176400'         00070   RELOC 176400
176400'         00080   MON:    MVI     A,003           ;RESET 2SIO
176401'
                00090   IFN REALIO,<
176402'         00100           OUT     20              ;AND INITIALIZE
176403'
176404'         00110           MVI     A,021
176405'
176406'         00120           OUT     20
176407'
                00130   >
176410'         00140   ENTER:  LXI     SP,STACK        ;LOAD STACK
176411'
176412'
176413'         00150           CALL    CRLF            ;FORMAT OUTPUT
176414'
176415'
176416'         00160           MVI     A,"."           ;HELLO MONITOR
176417'
176420'         00170           CALL    OUTCHK
176421'
176422'
176423'         00180           CALL    INCH            ;WHAT TO DO?
176424'
176425'
176426'         00190           CPI     "M"
176427'
176430'         00200           JZ      MEM             ;DO MEMORY EXAMINE
176431'
176432'
176433'         00210           CPI     "D"
176434'
176435'         00220           CZ      DMP             ;DO A MEMORY DUMP
```

```
176436'         00230           CPI     "J"
176437'
176440'         00240           JNZ     ENTER           ;NOT A VALID CMD
176441'
176442'
176443'         00250           CALL    OCTL6           ;DO JUMP, GET ADDR
176444'
176445'
176446'         00260           PCHL                    ;LOAD PC AND GO
176447'
176450'
                00010   ;THIS CONTROL STRUCTURE HANDLES THE MEMORY
                00020   ;EXAMINE AND CHANGE FUNCTION
                00030   ;
176451'         00040   MEM:    CALL    OCTL6           ;GET ADDRESS
176452'
176453'
176454'         00050           INST    076             ;"MVI A," SKIP NEXT (BOMB A)
176455'         00060   CONT:   INX     H               ;INCREMENT ADDRESS
176456'         00070           CALL    CRLF            ;NEW LINE
176457'
176460'
176461'         00080           MOV     D,H             ;STORE ADDRESS IN D/E
176462'         00090           MOV     E,L
176463'         00100           CALL    PRINT6          ;PRINT ADDRESS
176464'
176465'
176466'         00110           LDAX    D               ;LOAD DATA
176467'         00120           MOV     H,A
176470'         00130           CALL    PRINT3          ;PRINT DATA BYTE
176471'
176472'
176473'         00140           CALL    OCTL3           ;GET NEW DATA
176474'
176475'
176476'         00150           XCHG                    ;RESTORE ADDRESS
176477'         00160           JC      CONT            ;NO NEW DATA
176500'
176501'
```

```
176502'          00170              MOV     M,A             ;STORE DATA
176503'          00180              CMP     M               ;COMPARE DEPOSIT
176504'          00190              JZ      CONT            ;OK, DO NEXT
176505'
176506'
176507'          00200    ERR:      MVI     A,"?"           ;FLAG BAD DEPOSIT
176510'
176511'          00210              CALL    OUTCHK          ;PRINT "?"
176512'
176513'
176514'          00220              JMP     ENTER           ;RETURN TO MONITOR
176515'
176516'
                 00230    ;ERROR CONDITIONS RETURN TO MONITOR VIA "ERR"
                 00010    ;THIS CONTROL STRUCTURE RUNS THE MEMORY DUMP FUNCTION.
                 00020    ;
176517'          00030    DMP:      CALL    OCTL6           ;GET START
176520'
176521'
176522'          00040              XCHG                    ;STORE IN D/E
176523'          00050              CNC     SPACE
176524'
176525'
176526'          00060              CALL    OCTL6           ;GET END
176527'
176530'
176531'          00070              MVI     A,015           ;LOAD LEADER CHAR
176532'
176533'          00080    X1:       MVI     B,^D060         ;LOAD LEADER CNTR
176534'
176535'          00090    X2:       CALL    OUTCHK          ;PUNCH LEADER
176536'
176537'
176540'          00100              DCR     B
176541'          00110              JNZ     X2
176542'
176543'
176544'          00120              CMP     B               ;THROUGH WITH LEADER?
176545'          00130              MOV     A,B
176546'          00140              JNZ     X1              ;PUNCH NULLS
```

```
176547'
176550'
176551'  00150          MOV
176552'  00160          SUB
176553'  00170          MOV
176554'  00180          MOV
176555'  00190          SBB
176556'  00200          MOV
176557'  00210          INX
176560'  00220  BLOCK:  DCR
176561'  00230          MOV
176562'  00240          ORA
176563'  00250          JNZ
176564'
176565'
176566'  00260          MOV
176567'  00270  NOTLST:  MVI
176570'
176571'  00280          CALL
176572'
176573'
176574'  00290          MOV
176575'  00300          CALL
176576'
176577'
176600'  00310          MVI
176601'
176602'  00320          MOV
176603'  00330          CALL
176604'
176605'
176606'  00340          MOV
176607'  00350          CALL
176610'
176611'
176612'  00360  DATA:    LDAX
176613'  00370          CALL
176614'
176615'
```

```
A,L             ;SUB START FROM END
E
L,A
A,H
D
H,A             ;HL CONTAINS TOT BYTES
H               ;INCREMENT TOT BYTES
B               ;B=377Q
A,H
A               ;MORE THAN ONE BLOCK?
NOTLST          ;NOT LAST BLOCK


B,L             ;LAST BLOCK
A,074

OUTCHK          ;PUNCH "START OF BLOCK"


A,B             ;B=BYTE CNTR
OUTCHK          ;PUNCH BYTE COUNT


C,0             ;CLEAR CHECKSUM

A,E             ;PUNCH LOAD ADDR
OUTCHK          ;L.S. BYTE


A,D
OUTCHK          ;M.S. BYTE


D               ;GET DATA BYTE
OUTCHK          ;PUNCH IT
```

```
176616'          00380              INX     D                ;INCREM ADDR
176617'          00390              DCX     H                ;TOTBYTES=TOTBYTES-1
176620'          00400              DCR     B                ;THROUGH W/BLOCK?
176621'          00410              JNZ     DATA             ;NO
176622'
176623'
176624'          00420              MOV     A,C              ;YES, PUNCH CKSUM
176625'          00430              CALL    OUTCHK
176626'
176627'
176630'          00440              MOV     A,H              ;THROUGH W/ALL BYTES?
176631'          00450              ORA     L
176632'          00460              JNZ     BLOCK            ;NO, PUNCH NXT BLOCK
176633'
176634'
176635'          00470    CRLF:     MVI     A,015            ;DO A CRLF
176636'
176637'          00480              CALL    OUTCHK
176640'
176641'
176642'          00490              MVI     A,012
176643'
176644'          00500              JMP     OUTCHK
176645'
176646'
                 00510    ;RETURN TO MONITOR THROUGH OUTCHK
                 00010    ;THIS SUBROUTINE BUILDS 3/6 OCTAL DIGITS IN H&L
                 00020    ;
                 00030    ;SPECIAL RETURN PROVIDED BY A "SPACE", CARRY BIT SET.
                 00040    ;ONLY VALID OCTAL OR "SPACE" ACCEPTED, ALL OTHER FLAGED AND
                 00050    ;CONTROL RETURNS TO THE MONITOR.
                 00060    ;
176647'          00070    OCTL6:    INST    6                ;LOAD B WITH 6, SKIP NEXT
176650'          00080    OCTL3:    INST    6                ;LOAD B WITH 3
176651'          00090              INST    3
176652'          00100              LXI     H,$CODE+0        ;CLEAR H/L FOR LESS THAN 6 DIG RET
176653'
176654'
176655'          00110    AGN:      CALL    INCH             ;GET CHARACTER
```

```
176656'
176657'
176660'  00120          MOV     C,A             ;STORE IN C
176661'  00130          CPI     " "             ;COMPARE TO "SPACE"
176662'
176663'  00140          STC                     ;SET THE CARRY
176664'  00150          RZ                      ;RETURN IF "SPACE"
176665'  00160          ANI     270             ;TEST FOR VALID OCTAL
176666'
176667'  00170          XRI     060
176670'
176671'  00180          JNZ     ERR             ;BAD, FLAG & RET TO MON
176672'
176673'
176674'  00190          MOV     A,C             ;RESTORE CHAR
176675'  00200          ANI     007             ;STRIP ASCII
176676'
176677'  00210          DAD     H               ;SHIFT H&L LEFT 3 BITS
176700'  00220          DAD     H
176701'  00230          DAD     H
176702'  00240          ADD     L
176703'  00250          MOV     L,A             ;PUT OCTAL IN H
176704'  00260          DCR     B               ;THROUGH ?
176705'  00270          JNZ     AGN             ;NO, DO AGAIN
176706'
176707'
176710'  00280          RET                     ;YES, NORM RETURN
         00010  ;THIS SUBROUTINE PRINTS 3 OCTAL DIGITS FROM H
         00020  ;OR 6 OCTAL DIGITS FROM H AND L
         00030  ;
         00040  ;DIGITS ARE FOLLOWED BY A SPACE
         00050  ;
176711'  00060  PRINT6: MVI     B,6             ;LOAD CNTR W/6
176712'
176713'  00070          XRA     A               ;CLEAR A
176714'  00080          JMP     NEXT1           ;SHIFT ONE BIT
176715'
176716'
176717'  00090  PRINT3: MVI     B,3             ;LOAD CNTR W/3
```

```
176720'
176721'         00100                   INST    346             ;SKIP NEXT, SHIFT 2 BITS
176722'         00110   NEXT3:  DAD     H                       ;SHIFT H/L LEFT 3 INTO A
176723'         00120           RAL
176724'         00130           DAD     H
176725'         00140           RAL
176726'         00150   NEXT1:  DAD     H
176727'         00160           RAL
176730'         00170           ANI     7                       ;STRIP OFF OCTAL
176731'
176732'         00180           ORI     060                     ;ADD ASCII
176733'
176734'         00190           CALL    OUTCHK                  ;PRINT IT
176735'
176736'
176737'         00200           DCR     B                       ;THROUGH ?
176740'         00210           JNZ     NEXT3                   ;NO, SHIFT NEXT THREE
176741'
176742'
176743'         00220   SPACE:  MVI     A,040                   ;YES, PRINT SPACE
176744'
176745'         00230           JMP     OUTCHK                  ;AND RETURN
176746'
176747'
                00240   ;RETURN TO CALLING PROG THROUGH OUTCHK
                00010   ;THIS SUBROUTINE WILL INPUT A CHARACTER, STRIP
                00020   ;PARITY AND AUTOMATICALLY ECHO THE CHARACTER.
                00030   ;IT WILL ALSO OUTPUT A CHARACTER WITH CHECKSUM CALCULATIONS.
                00040   ;
                00050   IFN REALIO,<
176750'         00060   INCH:   IN      20                      ;READ STATUS
176751'
176752'         00070           RRC
176753'         00080           JNC     INCH                    ;NOT READY
176754'
176755'
176756'         00090           IN      21                      ;READ CHARACTER
176757'
                00100   >
                00110   IFE REALIO,<
```

```
                    00120   INCH:   IN
                    00130   >
176760'             00140           ANI
176761'
176762'             00150   OUTCHK: PUSH
176763'             00160           ADD
176764'             00170           MOV
                    00180   IFN REALIO,<
176765'             00190   LOOP:   IN
176766'
176767'             00200           RRC
176770'             00210           RRC
176771'             00220           JNC
176772'
176773'
176774'             00230           POP
176775'             00240           OUT
176776'
                    00250   >
                    00260   IFE REALIO,<
                    00270           POP
                    00280           OUT
                    00290   >
176777'             00300           RET
                    00080   END


NO ERRORS DETECTED

PROGRAM BREAK IS 177000
CPU TIME USED 00:05.334

4K CORE USED
```

```
1

177                 ;STRIP PARITY

PSW                 ;SAVE CHARACTER
C                   ;ADD IN CHECKSUM
C,A                 ;RESTORE CHECKSUM

20                  ;READ STATUS


LOOP                ;READY ?


PSW                 ;YES, GET CHAR
21                  ;PRINT CHARACTER


PSW
1

                    ;FROM WHENCE YE CAME
```

2450 Alamo S.E.
Albuquerque, New Mexico 87106

USER'S DOCUMENTATION REPORT

In order to improve the quality and usefulness of our publications, user feedback is necessary. Your comments will help us effectively evaluate our documentation.

Please limit your remarks to the document, giving specific page and line references when appropriate. Specific hardware or software questions should be directed to the MITS Customer Service or Software Departments, respectively.

NAME OF PUBLICATION: _____

SUGGESTIONS FOR IMPROVEMENT: _____

_____

_____

_____

_____

ERRORS: _____

_____

_____

_____

_____

Name _____ **Date** _____

Organization _____

Street _____

City _____ **State** _____ **Zip** _____

First Fold Here

Second Fold Here and Staple

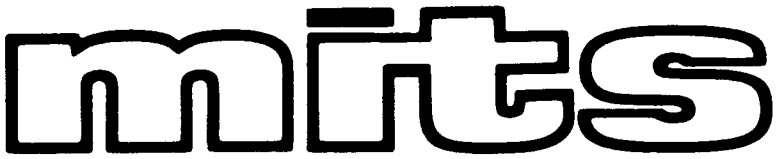
No Postage Stamp
Necessary If Mailed in
the United States

BUSINESS REPLY MAIL

First Class Permit No. 2114, Albuquerque, New Mexico

Postage Will be Paid by:

MITS, Inc.
2450 Alamo S.E.
Albuquerque, New Mexico 87106



2450 Alamo S.E. / Albuquerque, New Mexico 87106