Altaids computer

Software Manual 09/10/18

MONITOR

The operating system of the Altaids computer is a program known as a "monitor program." This program allows for interacting with the computer over a serial port and debugging the programs. Commands consist of simple one or two letter command followed by hexadecimal values. Monitor programs were very popular with early computers. These programs allowed values to be written directly to memory, displaying memory, reading and writing Intel Hex files, etc. The monitor prompt is an asterisk '*'.

Monitor Start Up and Prompt

```
Altaids Monitor Ver 05
Copyright (C) 2018 David R. Hunter
This program comes with ABSOLUTELY NO WARRANTY
This is free software, and you are welcome to redistribute it
under certain conditions; see the GNU GPL Version 3 for details
Type "B" for BASIC, "H" for help
Type ^X to return to the Monitor
*
```

MONITOR COMMANDS

CMD: `A`

Enter Altair Turnkey Monitor (see below)

CMD: `B`

Start Tiny BASIC

CMD: `Dssss eeee`

Dump memory from address `ssss` to `eeee`. After ssss is entered, a space will automatically be generated.

CMD: `Eaaaa`

Write to memory starting at address `aaaa`. The address will be printed with the current byte. Type a new byte (in hexadecimal) to change. Press <SPACE> to skip the address leaving it unchanged. Type any other character to exit.

CMD: `Gaaaa`

Go (execute) a program from address `aaaa`

CMD: `H`

Help / Display monitor commands

CMD: `I`

Input a byte from port `60H` and display

CMD: `L`

Load an Intel Hex file. Send the text file from the terminal emulator.

A '.' is printed for each record received correctly, a '?' if there is a checksum error, a '!' is printed when done

CMD: `Obb`
Output byte `bb` to port `61H`

CMD: `R`
Display the values of the registers. Program Counter, Stack Pointer, Accumulator, Processor flags, BC,DE,HL. The flags are: Sign, Zero, Auxiliary carry, Parity, and Carry. An example is below

```
PC   :SP   :A   SZ_A_P_C:B C :D E :H L
11CB 3F9E 22 00000100 01D2 1082 11C8
```

CMD: `Taaaa`
Single Step (trace) instructions starting at address `aaaa`. Type any key to step. Type ^C to exit. Each step shows the registers and a disassembled instruction. See the example below for details.

MONITOR ADDRESS VECTORS
The ROM subroutine addresses below may be useful for writing programs on the Altaids computer.

MONITOR JUMP ADDRESSES

1000	COLD	Monitor cold start
1003	WARM	Tiny BASIC warm start (return to BASIC without losing program in memory with *G1003*)

MONITOR VECTORS TO USEFUL ROUTINES (ENTER WITH CALL OPCODE)

1006	COUT	Output char in A to serial port
1009	CIN	Input char from serial port to A
100C	PRTWD	Output H,L as 4 Hex digits to serial port
100F	PRTBY	Output the byte in A as 2 Hex digits to serial port
1012	GETWD	Get Hex word from serial port, return in H,L
1015	GETBY	Get Hex byte from serial port, return in A
1018	PIN	Get byte from input port, return in A
101B	POUT	Send byte in A to output port
101E	DSPREG	Display registers (registers are restored upon return)
1021	PUTSTR	Send null terminated string pointed to by DE to serial port

ALTAIR MONITOR

The Altair monitor was adapted from the Altair Turnkey Monitor (TURMON) for the Altaids Computer. It is more limited than the Altaids monitor and operates with Octal values only. It is provided to “relive” the experience of operating an original Altair computer.

Altair Monitor Prompt character: '.'

Commands:

Mxxxxxx	- change memory location xxxxxx (octal)
	- enter 3 octal digits for new data
	- press <SPACE> to skip to next location
	- any other character to exit
Dxxxxxx xxxxxx	- dump data from start to end address
	- originally in Altair 8800b binary Absolute Tape Format
	- now changed to a more standard memory dump
Jxxxxxx	- jump to location and begin executing

TINY BASIC

Palo Altaids Tiny BASIC is an integer only BASIC. It is a modified version of Palo Alto Tiny BASIC by LiChen Wang. This was one of the most popular Tiny BASIC implementations for the Intel 8080.

All numbers are 16 bit values between -32767 and 32767. There are 26 variables A-Z. There is a single large one dimensional array '@(n)'. The @ array uses all remaining memory, so it's size depends on the length of the program. The number of elements is from 0 to SIZE/2. Also, one dimensional array variables may be defined using the DIM statement. Defined arrays reduce the size of memory available for the @ array.

Note, all values are decimal, so address values are expressed as decimal values. (e.g. 8192 = 0x2000)

Line numbers are from 1 to 32767. To add a line, type the line number and statement(s). To delete a line, enter the line number by itself.

There is a "one line screen editor" available which requires an ANSI compatible terminal or emulator (e.g. VT100). To edit a line, type "EDIT <line num>". The prompt will change to ']' and the line to edit will be displayed. Use the left and right arrows to move about the line. The <DEL> key will delete the character under the cursor. The <BACKSPACE> key will delete the character to the left (unless at the line number). Any characters typed will automatically be inserted. The <ENTER> key will erase to the end of the line and exit the editor saving the line. To exit the editor and save the line press <F1>. Note, the line number cannot be changed with the editor.

Tiny BASIC can use an external computer running a terminal emulator (like TeraTerm) to load and store programs. The files are transferred using the XMODEM protocol (checksum mode only). This allows another computer to provide storage for programs. Files transferred via XMODEM typically use a ".TBI" extension. The files transferred are binary files.

When Tiny BASIC is ready, the prompt is '>'. When a command is successful there terminal will show "OK" followed by a prompt. When Tiny BASIC is entered, the following is displayed.

```
PALO ALTAIDS TINY BASIC VER 1.0

OK
>
```

BASIC COMMANDS

DIRECT COMMANDS	**OPERATORS**	**KEYWORDS**	**FUNCTIONS**
NEW	+	REM	RND
LIST	-	LET	ABS
RUN	*	IF	PEEK
EDIT	/	GOTO	USR
XLOAD	=	GOSUB / RETURN	INP
XSAVE	<>	INPUT	SIZE
BYE	<	PRINT	FREE
	>	STOP / END	*"c"*
	<=	CLS	
	>=	FOR / TO / STEP / NEXT	
		GET$	
		PUT$	
		POKE	
		OUT	
		DIM	
		RANDOMIZE	

Direct Commands

NEW	- clear the memory for a new program
LIST	- list the program
LIST num	- list the program starting at line "num"
RUN	- start running the program
EDIT num	- edit line "num" [ANSI terminal (or emulator) required]
XLOAD	- receive a file using XMODEM protocol (in TeraTerm: use *File->Transfer->XMODEM->Send...*)
XSAVE	- send a file using XMODEM protocol (in TeraTerm: use *File->Transfer->XMODEM->Receive...*)
BYE	- exit Tiny BASIC and return to the monitor

Operators

The four fundamental math operations are available in addition to comparison operators. The compare operators result in a 1 if TRUE and a 0 if FALSE. The compare operators results can be used for complex operations. For example:

```
20 IF (U=1)*(V<2)+(U>V)*(U<99)*(V>3) PRINT "YES"
```

The '*' operator acts as a logical AND operation and the '+' operator acts as a logical OR operation.

Keywords

Blanks can be used freely except in numbers, keywords and function names. Keywords may be abbreviated when truncated with a period. Note, the abbreviation must be unambiguous (e.g. “GOS.” vs “GOT.” for GOSUB vs GOTO) Statements on the same line are separated by a colon (':'), however “GOTO”, “STOP”, “END” and “RETURN” must be the last command in any given statement. Also, LET is optional.

Note: *expr* = expression or variable *var* = variable *num* = number *str* = string

REM	- remark statement, ignored by Tiny BASIC
LET *var=expr*	- set the variable to the expression
LET *var=“char”*	- set the variable to the ASCII value of the character between the double quotes
IF *expr stmt*	- check if expression is not zero. If so, execute the statement. (note “THEN” is not used)
	- if the expression is zero, the remainder of the statement is skipped
GOTO *num*	- jump to statement at line number “num”
GOTO *expr*	- jump to the line number value determined by the expression (e.g. computed GOTO)
GOSUB *num*	- call the subroutine at the line number “num”
GOSUB *expr*	- similar to GOTO except it is a call rather than a jump
RETURN	- signal the end of a subroutine and return to the statement that followed the GOSUB command
INPUT *var*	- input a numeric value to a variable. When executed, Tiny BASIC will print the variable name followed by a colon as a prompt. If the input is not valid, Tiny BASIC will print “WHAT?”
INPUT “*str*”*var*	- use the string “str” as the prompt rather than the variable name.
PRINT	- if nothing follows the PRINT statement, output a CR/LF only
PRINT *expr*	- print the result of an expression followed by a CR/LF on the output
PRINT expr,	- print the result of an expression without a CR/LF when a comma ',' is at the end of the line
PRINT *expr*_	- an underscore in the line (_) means generate a CR without a LF
PRINT *#n,expr*	- print the variable with a fixed number of digits. The default is 6.
PRINT *$expr*	- print the ASCII character whose value is determined by the expression. e.g. $42 = '*'
PRINT *“str”*	
PRINT *'str'*	- a string is quoted by either double or single quotes. Note, they cannot be mixed
	- note, multiple values can be printed if separated by commas.
STOP	- exit the program and return to the user prompt
END	- same as STOP
CLS	- generate an ANSI clear screen command [ANSI terminal (or emulator) required]
FOR *var=expr1* TO *expr2* STEP *expr3*	
	- execute a loop from *expr1* to *expr2* updating variable *var*. The STEP parameter is optional.
	- When STEP is not present, the STEP is 1
NEXT *var*	- end of the loop, check if *expr2* has been reached, if not update var by adding *expr3* (or 1) and loop
GET$ *expr*	- input a string from the keyboard to successive memory locations.
	- Pressing ENTER stores a NULL after the string. The length of the string is returned in variable Z
PUT$ *expr*	- output the null terminated string starting from address *expr*.
POKE *expr1,expr2*	
	- put data from expr2 into the memory location *expr1*
OUT expr1,expr2	
	- output value of *expr2* to microprocessor output port *expr1* (note: Altaids output port = 97)

DIM *var*(*num*)	- define a one dimensional array variable of length *num*. A DIM command is required for each variable e.g. DIM J(4): DIM K(5): DIM L(6)
RANDOMIZE	- seed the random number generator with a hardware generated value Tiny BASIC initializes with the same seed value after a reset. Each call to RND() will generate the same sequence. This is handy for games that generate the same sequence each time (e.g. mazes). The RANDOMIZE function will generate a random seed value (based on keypress intervals). This is useful for games in which a new random sequence is desired (e.g. card games).

Functions

RND(*expr*)	- return a random number between 1 and *expr* (inclusive)
ABS(*expr*)	- return the absolute value of *expr*
PEEK(*expr*)	- return the value of the byte at address *expr*
USR(expr)	- call the machine language subroutine at expr
USR(expr1,expr2)	- call the machine language subroutine at expr with value expr2 passed in H,L registers - the value of the function from the machine language subroutine is returned in H,L
INP(expr)	- return value read from microprocessor input port *expr* (note: Altaids input port = 96)
SIZE	- return the number of bytes of memory unused by the program
FREE	- return the address of the first free byte of memory (useful for string addresses)
"*char*"	- return the ASCII value of a character between quotes (ASCII to decimal conversion)

Error Messages

WHAT?	- indicates that Tiny BASIC does not understand
HOW?	- indicates that the statement cannot be executed (e.g. a GOTO to a line number that doesn't exist)
SORRY?	- indicates that there is not enough memory to execute the statement

Example 1 – Tiny BASIC program

```
5 REM SIEVE OF ERATOSTHENES
10 INPUT "ENTER NUMBER TO SEARCH TO " L
15 IF L > 3500 GOTO 140
20 FOR I=0 TO L: @(I)=0: NEXT I
25 GOSUB 165
28 PR."STARTING CALCULATION"
30 FOR N = 2 TO X
40 IF @(N) <> 0 GOTO 80
50 FOR K = N*N TO L STEP N
60 @(K) = 1
70 NEXT K
80 NEXT N
85 PR."DONE"
90 REM DISPLAY THE PRIMES
100 FOR N = 2 TO L
110 IF @(N) = 0 PR.#5,N,
120 NEXT N
130 STOP
140 PR."ERROR: MAXIMUM NUMBER IS 3500"
150 GOTO 10
160 REM SQUARE ROOT CALCULATION
165 X = L
170 Y = 1
180 IF (X-Y)<=0 RETURN
190 X = (X+Y)/2
200 Y=L/X
210 GOTO 180
220 RETURN
```

Example 2 – Tiny BASIC program

```
2000 REM ACEY-DUECEY
2100 PR.
2101 PRINT "ACEY-DUECEY IS PLAYED IN THE FOLLOWING MANNER:"
2102 PRINT "THE DEALER (COMPUTER) DEALS TWO CARDS FACE UP."
2103 PRINT "YOU THEN BET AN AMOUNT DEPENDING ON WHETHER"
2104 PRINT "OR NOT YOU FEEL THE NEXT CARD WILL HAVE"
2105 PRINT "A VALUE BETWEEN THE FIRST TWO."
2106 PRINT "THE MINIMUM BET IS 10 DOLLARS."
2110 PR.
2120 RANDOMIZE
2170 Q=100
2190 PRINT "YOU NOW HAVE ",Q," DOLLARS."
2195 PR.
2200 GOTO 2260
2210 LET Q=Q+M
2220 GOTO 2190
2240 LET Q=Q-M
2250 GOTO 2190
2260 PRINT "HERE ARE YOUR NEXT TWO CARDS..."
2270 LET A=RND(14)
2280 IF (A<2)+(A>14) GOTO 2270
2300 LET B=RND(14)
2310 IF (B<2)+(B>14) GOTO 2300
2330 IF A>=B GOTO 2270
2340 IF B=A+1 GOTO 2270
2350 LET D=A
2360 GOSUB 3100
2500 LET D=B
2510 GOSUB 3100
```

```
2560 PR.
2660 INPUT "WHAT IS YOUR BET?"M
2670 IF M>=10 GOTO 2680
2675 PRINT "SORRY MY FRIEND, THE MINIMUM BET IS 10 DOLLARS"
2676 PR.
2677 GOTO 2660
2680 IF M<=Q GOTO 2730
2690 PRINT "SORRY MY FRIEND, BUT YOU BET TOO MUCH"
2700 PRINT "YOU HAVE ONLY ",Q," DOLLARS TO BET"
2710 GOTO 2560
2730 LET C=RND(14)
2740 IF (C<2)+(C>14)+(C=A)+(C=B) GOTO 2730
2755 PRINT "THE DRAW CARD IS :",
2760 LET D=C
2770 GOSUB 3100
2910 IF C>A GOTO 2930
2920 GOTO 2970
2930 IF C>=B GOTO 2970
2950 PRINT "YOU WIN!!!"
2955 PR.
2960 GOTO 2210
2970 PRINT "SORRY, YOU LOSE."
2975 PR.
2980 IF M<Q GOTO 2240
3010 PRINT "SORRY FRIEND, BUT YOU LOST IT ALL."
3020 PRINT
3030 PRINT "DO YOU WANT TO PLAY AGAIN?"
3040 INPUT "TYPE 1 TO PLAY AGAIN AND 0 TO EXIT"I
3060 IF I=0 GOTO 3400
3070 IF I>1 GOTO 3040
3080 IF I=1 GOTO 2110
3100 REM PRINT CARD VALUE FROM D
3110 IF D<11 PRINT #2,D:GOTO 3250
3120 IF D=11 PRINT "JACK":GOTO 3250
3130 IF D=12 PRINT "QUEEN":GOTO 3250
3140 IF D=13 PRINT "KING":GOTO 3250
3150 IF D=14 PRINT "ACE":GOTO 3250
3250 RETURN
3400 END
```

Example 3 – assembly language program

```
                         ; DEMO.ASM
                         ;
                         ; ALTAIDS computer demo
                         ; based on Altair demo from Popular Electronics, Feb 1975
                         ;
                         ; COMMAND TO ASSEMBLE:
                         ;  a85 demo.asm -l demo.prn -o demo.hex
                         ;
                         ;
   2000                           ORG    2000H
   2000    3a 80 20      DEMO:    LDA    VAR1
   2003    47                     MOV    B,A
   2004    3a 81 20               LDA    VAR2
   2007    80                     ADD    B
   2008    32 82 20               STA    VAR3
   200b    c3 00 20               JMP    DEMO
                         ;
   2080                           ORG    2080H
   2080    1c            VAR1:    DB     1CH
   2081    08            VAR2:    DB     08H
   2082    00            VAR3:    DB     0

   2083                           END
```

Example 4 – using the monitor

```
Type ^X to return to the Monitor

*L                                              <= send file "demo.hex" from TeraTerm
..                                              <= dots show status of loading the file  (one dot per line)
2083                                            <= first free address after program was loaded
*T2000                                          <= single step the program from 0x2000

Press any key to step to the next instruction
Press ^C to return to the Monitor
PC   :SP   :A   SZ_A_P_C:B C :D E :H L          <= identifies register values and flags
2000 3F02 00 01000100 0005 15B8 2000   LDA   2080   <= shows current register values (in hex) and next instruction
                                                <= press a key (e.g. <SPACE>) to show the next instruction
PC   :SP   :A   SZ_A_P_C:B C :D E :H L
2003 3F02 1C 01000100 0005 15B8 2000   MOV   B,A

PC   :SP   :A   SZ_A_P_C:B C :D E :H L
2004 3F02 1C 01000100 1C05 15B8 2000   LDA   2081

PC   :SP   :A   SZ_A_P_C:B C :D E :H L
2007 3F02 08 01000100 1C05 15B8 2000   ADD   B

PC   :SP   :A   SZ_A_P_C:B C :D E :H L
2008 3F02 24 00010100 1C05 15B8 2000   STA   2082

PC   :SP   :A   SZ_A_P_C:B C :D E :H L
200B 3F02 24 00010100 1C05 15B8 2000   JMP   2000

PC   :SP   :A   SZ_A_P_C:B C :D E :H L
2000 3F02 24 00010100 1C05 15B8 2000   LDA   2080
```