MITS ALTAIR BASIC

REFERENCE MANUAL

Table of Contents:

PRINTED IN U.S.A.

MITS
"Creative Electronics"
P.O. BOX 8636
ALBUQUERQUE, NEW MEXICO 87108

ALTAIR MITS BASIC

"Creative Electronics"

Supplement & Errata

The following are additions and corrections to the ALTAIR BASIC REFERENCE MANUAL. Be sure to read this over carefully before continuing.

1) If you are loading BASIC from paper tape, be sure your Serial I/O board is strapped for eight data bits and no parity bit.

2) On page 53 in Appendix C, the meaning for an "OS" error should read:

> Out of String Space. Allocate more string space by using the "CLEAR" command with an argument (see page 42), and then run your program again. If you cannot allocate more string space, try using smaller strings or less string variables.

3) On page 42, under the "CLEAR" command, It is stated that "CLEAR" with no argument sets the amount of string space to 200 bytes. This is incorrect. "CLEAR" with no argument leaves the amount of string space unchanged. When BASIC is brought up, the amount of string space is initially set to 50 bytes.

4) On page 30, under the "DATA" statement, the sentence "IN THE 4K VERSION OF BASIC, DATA STATEMENTS MUST BE THE FIRST STATEMENTS ON A LINE," should be changed to read, "IN THE 4K VERSION OF BASIC, A DATA STATEMENT MUST BE ALONE ON A LINE."

5) If you desire to use a terminal interfaced to the ALTAIR with a Parallel I/O board as your system console, you should load from the ACR interface (wired for address 6). Use the ACR load procedure described in Appendix A, except that you should raise switches 15 & 13 when you start the boot. The Parallel I/O board must be strapped to address 0.

6) If you get a checksum error while loading BASIC from a paper tape or a cassette, you may be able to restart the boot loader at location 0 with the appropriate sense switch settings. This depends on when the error occurs. The boot loader is not written over until the last block of BASIC is being read; which occurs during approximately the last two feet of a paper tape, or the last 10 to 15 seconds of a cassette. If the checksum error occurs during the reading of the last block of BASIC, the boot will be overwritten and you will have to key it in again.

7) The number of nulls punched after a carriage return/line feed does not need to be set >=3 for Teletypes or >=6 for 30 CPS paper tape terminals, as described under the "NULL" command on page 23 of the BASIC manual. In almost all cases, no extra nulls need be punched after a CR/LF on Teletypes, and a setting of nulls to 3 should be sufficient for 30 CPS paper tape terminals. If any problems occur when reading tape (the first few characters of lines are lost), change the null setting to 1 for Teletypes and 4 for 30 CPS terminals.

8) If you have any problems loading BASIC, check to make sure that your terminal interface board (SIO or PIO) is working properly. Key in the appropriate echo program from below, and start it at location zero. Each character typed should be typed or displayed on your terminal. If this is not the case, first be sure that you are using the correct echo program. If you are using the correct program, but it is not functioning properly, then most likely the interface board or the terminal is not operating correctly.

In the following program listings, the number to the left of the slash is the octal address and the number to the right is the octal code for that address.

FOR REV 0 SERIAL I/O BOARDS WITHOUT THE STATUS BIT MODIFICATION

```
 0 / 333          1 / 000          2 / 346
 3 / 040          4 / 312          5 / 000
 6 / 000          7 / 333         10 / 001
11 / 323         12 / 001         13 / 303
14 / 000         15 / 000
```

FOR REV 1 SERIAL I/O BOARDS (AND REV 0 MODIFIED BOARDS)

```
 0 / 333          1 / 000          2 / 017
 3 / 332          4 / 000          5 / 000
 6 / 333          7 / 001         10 / 323
11 / 001         12 / 303         13 / 000
14 / 000
```

FOR PARALLEL I/O BOARDS

```
 0 / 333          1 / 000          2 / 346
 3 / 002          4 / 312          5 / 000
 6 / 000          7 / 333         10 / 001
11 / 323         12 / 001         13 / 303
14 / 000         15 / 000
```

For those of you with the book, MY COMPUTER LIKES ME when i speak in BASIC, by Bob Albrecht, the following information may be helpful.

1) ALTAIR BASIC uses "NEW" instead of "SCR" to delete the current program.

2) Use Control-C to stop execution of a program. Use a carriage-return to stop a program at an "INPUT" statement.

3) You don't need an "END" statement at the end of a BASIC program.

Introduction

Before a computer can perform any useful function, it must be "told" what to do. Unfortunately, at this time, computers are not capable of understanding English or any other "human" language. This is primarily because our languages are rich with ambiguities and implied meanings. The computer must be told precise instructions and the exact sequence of operations to be performed in order to accomplish any specific task. Therefore, in order to facilitate human communication with a computer, programming languages have been developed.

ALTAIR BASIC* is a programming language both easily understood and simple to use. It serves as an excellent "tool" for applications in areas such as business, science and education. With only a few hours of using BASIC, you will find that you can already write programs with an ease that few other computer languages can duplicate.

Originally developed at Dartmouth University, BASIC language has found wide acceptance in the computer field. Although it is one of the simplest computer languages to use, it is very powerful. BASIC uses a small set of common English words as its "commands". Designed specifically as an "interactive" language, you can give a command such as "PRINT 2 + 2", and ALTAIR BASIC will immediately reply with "4". It isn't necessary to submit a card deck with your program on it and then wait hours for the results. Instead the full power of the ALTAIR is "at your fingertips".

Generally, if the computer does not solve a particular problem the way you expected it to, there is a "Bug" or error in your program, or else there is an error in the data which the program used to calculate its answer. If you encounter any errors in BASIC itself, please let us know and we'll see that it's corrected. Write a letter to us containing the following information:

1) System Configuration

2) Version of BASIC

3) A detailed description of the error
   Include all pertinent information such as a listing of the program in which the error occurred, the data placed into the program and BASIC's printout.

All of the information listed above will be necessary in order to properly evaluate the problem and correct it as quickly as possible. We wish to maintain as high a level of quality as possible with all of our ALTAIR software.

* BASIC *is a registered trademark of Dartmouth University.*

We hope that you enjoy ALTAIR BASIC, and are successful in using it to solve all of your programming needs.

In order to maintain a maximum quality level in our documentation, we will be continuously revising this manual. If you have any suggestions on how we can improve it, please let us know.

If you are already familiar with BASIC programming, the following section may be skipped. Turn directly to the Reference Material on page 22.

NOTE: MITS ALTAIR BASIC is available under license or purchase agreements. Copying or otherwise distributing MITS software outside the terms of such an agreement may be a violation of copyright laws or the agreement itself.

ALTAIR 680 BASIC

Altair 680 BASIC is identical to Altair 8800 8K BASIC, Revision 3.2, with a few exceptions. The BASIC Reference Manual for 8800 8K BASIC is included here. Listed below are the sections of the BASIC Reference Manual that are applicable to 680 BASIC:

Pages 1-43
Appendices C-H
Appendix K
Appendix M

These introductory pages to the "BASIC Reference Manual" (pages I through VI) contain several additions to the manual that are needed when using 680 BASIC.

Loading Altair 680 BASIC

Altair BASIC is loaded into the 680 using the PROM Monitor's L command. (See the Altair 680 System Monitor Manual for details.)

Initialization Dialog

Starting BASIC:

Use the PROM Monitor's J command to begin execution of BASIC at address Ø.

```
.J ØØØØ
```

After you have started BASIC, it will respond:

```
MEMORY SIZE?
```

If you type a Carriage Return to MEMORY SIZE?, BASIC will use all the contiguous memory upwards from location zero that it can find. BASIC will stop searching when it finds one byte of ROM or non-existent memory.

If you wish to allocate only part of the Altair's memory to BASIC, type the number of bytes of memory you wish to allocate in decimal. This might be done, for instance, if you were using part of the memory for a machine language subroutine.

BASIC will then ask:

```
TERMINAL WIDTH?
```

This is to set the output line width for PRINT statements only. Type in the number of characters for the line width for the particular terminal or other output device you are using. This may be any number from 1 to 255, depending on the terminal. If no answer is given (i.e. a Carriage Return is typed) the line width is set to 72 characters.

Now Altair BASIC will enter a dialog which will allow you to delete some of the arithmetic functions. Deleting these functions will give more memory space to store your programs and variables. However, you will not be able to call the functions you delete. Attempting to do so will result in an FC error. The only way to restore a function that has been deleted is to reload BASIC.

The following is the dialog which will occur:

```
WANT SIN-COS-TAN-ATN?
```

Answer "Y" to retain all four of the functions, "N" to delete all four, or "A" to delete ATN only.

Now BASIC will type out:

```
XXXX BYTES FREE
MITS ALTAIR 680 BASIC
VERSION Y.Y REV Z.Z
COPYRIGHT, 1976 BY MITS, INC.
```

"XXXX" is the number of bytes available for program, variables, matrix storage and the stack. It does not include string space. Y.Y is the version number. Z.Z is the revision number.

```
OK
```

INP and OUT

The INP and OUT facilities of Altair 8800 BASIC are unnecessary in 680 BASIC because of the I/O structure of the 6800 microprocessor. The 6800 handles I/O by addressing certain memory locations as ports. Therefore, to do an input, the PEEK function is used. For example, suppose memory location 65000_{10} is an input port for a peripheral. Then to get the information, simply use

```
I = PEEK (65000)
```

either as a direct or indirect command.

To output, use the POKE command. To output an ASCII "A" to the port addressed by 64000_{10}, use the statement

```
POKE 64000, ASC("A")
```

either as a direct or indirect command.

User Machine Language Interface

A user can call his own machine language function by using the USR function. The user must reserve some memory for this function by typing a number in response to BASIC's initialization question

```
MEMORY SIZE?
```

For example, to reserve 1K of memory for user functions in a 17K machine, the response would be

```
16383
```

The user function may be loaded into memory via the PROM Monitor, the front panel switches, or the POKE function in BASIC. Prior to calling the user function, the starting address of the function must be stored in locations $287_{(10)}$ (high order byte of address) and $288_{(10)}$ (low order byte of address). For example, if the user function begins at 17K, the following procedure should be used.

17K is $4000_{(16)}$
so $40_{(16)}$ must be stored in $287_{(10)}$
and $00_{(16)}$ must be stored in $288_{(10)}$
$40_{(16)}$ is $4 \times 16 = 64_{(10)}$
and Ø is Ø in any base

so the commands

```
POKE 287, 64
```

and

```
POKE 288, Ø
```

would be used to store the starting address of the user function. The call to a user function is

```
Y = USR(X)
```

X is the argument and must be a number in the range $+32767_{(10)}$ and $-32768_{(10)}$. The result of the USR function is returned in Y and must also be in the range $+32767_{(10)}$ and $-32768_{(10)}$.

The argument is obtained for use in the user function by calling the routine whose address is given in locations Ø115 and Ø116 (hexadecimal). Therefore, the instructions

```
LDX #Ø115
JSR X
```

cause the argument to be converted to a signed two byte integer with the high order byte stored location $174_{(10)}$ and the low order byte stored in $175_{(10)}$.

The result of the function is returned to BASIC by storing the high order byte in accumulator A, the low order byte in accumulator B, and calling the routine whose address is given in locations Ø117 and Ø118 (hexadecimal).

For example, the instructions

```
CLR A
LDA B #3
LDX #Ø117
JSR X
```

will return a value of 3 to BASIC. Program control is returned to BASIC by executing an RTS instruction.

Example USR Function

The USR function described below generates a program delay of 1 second times the argument. The function assumes the argument is between 1 and $255_{(10)}$. The value returned to BASIC is always zero. It is assumed the user answered the memory size question with 16383.

```
00001                        NAM     USRFN
00002 4000                   ØRG     #4000
00003 4000 FE 0115           LDX     #0115     GET LOW BYTE OF ARG
00004 4003 AD 00             JSR     X         INTØ B
00005 4005 D6 AF             LDA B   175       WE ASSUME HIGH BYTE IS 0
00006 4007 CE F424 WAIT1     LDX     #62500    THIS LØØP GENERATES
00007 400A 09      WAIT2     DEX               A DELAY OF ARG*1 SEC
00008 400B 26 FD             BNE     WAIT2
00009 400D 5A                DEC B             DECREMENT THE ARG
00010 400E 26 F7             BNE     WAIT1
```

```
00011 4010 4F               CLR A
00012 4011 FE 0117          LDX         #0117     A&B ARE ZERØ
00013              *RETURN THE VALUE TO BASIC
00014 4014 6E 00            JMP         X         (JSR AND RTS)
00015                       END
TØTAL ERRØRS 00000
```

NOTE: This function was developed using the Altair 680 Assembly Language Development System.

The following BASIC program rings the Teletype bell at 10 second intervals by calling the USR function above.

```
 5 REM   SET UP USR ADDRESS
10 POKE 287,64
20 POKE 288,0
25 REM RING THE BELL
30 PRINT CHR$(7);
35 REM DELAY 10 SECONDS
40 X = USR(10)
45 REM DO IT AGAIN
50 GOTO 30
```

680 BASIC I/O Patch Points

Altair 680 BASIC calls routines in the 680 PROM Monitor to perform I/O transfers. The following routines are necessary and the address of their calls are given.

1) INCH - Input character routine. Reads character from terminal, strips parity, and returns the resultant 7 bit ASCII character in accumulator B. Called from location Ø41F (hex).
2) OUTCH - Output character routine. Sends the ASCII character in accumulator B to the terminal. Called from location Ø8AD (hex).
3) POLCAT - Poll for character routine. Checks input status of terminal. Returns carry set if character has been typed. Returns carry clear if no character has been typed. Called from location Ø618 (hex).

Baudot Control - C

The Baudot version of the PROM Monitor supports only those Baudot Teletypes wired for half-duplex operation. It is therefore impossible to type a control - C while BASIC is doing output. Consequently, BASIC checks the Baudot bit at location FØØ2 and if it indicates the presence of a Baudot terminal, any character typed while BASIC is executing a program will be interpreted as a control - C.