THE 8K BASIC—IN—ROM
REFERENCE MANUAL

Ohio Scientific's BASIC-in-ROM was written by Microsoft, Inc. and is very compatible with the numerous other BASICS written by Microsoft including the original ALTAIR 8080 and 6800 BASIC, the Commodore PET BASIC, the Apple floating POINT BASIC, and Radio Shack Level II BASIC. The 6502 BASIC is considerably faster than 8080 and 6800 BASICS because of the 6502's superior instruction execution time. It is faster than competitive personal computer BASICS because it is a 6½ digit implementation of BASIC and Ohio Scientific machines are typically operated at higher clock speeds that comparable 6502 based personal computers. As of the writing of this manual, Ohio Scientific 8K BASIC-in-ROM is considered the fastest floating POINT BASIC available for personal computers.

The following manual provides detailed information about the features in the language which are unique in this particular version of BASIC. It further provides a handy reference for the standard syntax of Microsoft BASIC. It is not intended however, to be a tutorial or teaching aid. The user is directed to several of any of the excellent texts on BASIC to learn the language such as BASIC AND THE PERSONAL COMPUTER by Dwyer and Crithfield which is available through most Ohio Scientific dealers.

Ohio Scientific 8K BASIC-in-ROM has been extensively tested and has been in use for several years. It is believed to be reasonably bug free, however, no warranty is made for its accuracy or usability.

8K BASIC-in-ROM is copyrighted by Microsoft, Inc. The BASIC I/O handlers or support code are copyrighted by Ohio Scientific, Inc. The duplication, copying or publication of the code of the 8K BASIC-in-ROM is strictly prohibited without specific written consent from Ohio Scientific.

Ohio Scientific 8K BASIC-in-ROM Reference Manual

Introduction

The following discussion pertains to Ohio Scientific computers which utilize 8K BASIC-in-ROM in conjunction with an internal video display interface, specifically, the C2-4P, C2-8P, Superboard II and Challenger 1P systems. When the computer is reset, the message D/C/W/M or simply C/W/M will appear on the screen. D stands for boot from disk, M stands for exit to monitor, C and W refer to BASIC options. When the machine is first turned on, the user always selects C which stands for "cold start". Cold start clears the work space and initializes the BASIC interpreter. Once the machine has been running, the user can optionally select W for "warm start". Warm start is used to restart the BASIC interpreter when an executable program is present in memory. It can be utilized specifically when the break key is inadvertently depressed to return to a program and in instances when the control C has been disabled for graphics displays for instance. For example, when the Ohio Scientific Tiger Tank game is running in the computer and the user wishes to examine the program, he would depress the break key then the W key which will return him to the immediate mode of BASIC with his program intact.

Memory Size?

When cold starting BASIC, the computer asks the question "Memory Size?". By answering carriage return to this question, the machine performs a memory test such that it finds all usable memory. It reports the number of bytes available and sets up the basic work space to use all available memory. The user can optionally type in a decimal number which corresponds to the upper limit of the memory he wants BASIC to use. For example, an 8K BASIC-in-ROM computer has 8192 bytes of RAM (approximately 768 bytes are used for system overhead). If the user wishes to allow BASIC to utilize 4K thus leaving 4K available for machine code and other applications, he would answer, 4096 to memory size then carriage return. This would limit BASIC to utilize the first 4K of memory. Furthermore, the BASIC memory test would not be implemented, consequently, memory above 4K would be preserved. Because of this, the user can preload machine code routines before cold starting the computer.

Terminal Width?

The next question BASIC asks is "Terminal Width?". If the user simply types carriage return to this question, the terminal width defaults to 72 characters. This terminal width specification is part of BASIC's drivers. The video display software also contains terminal width parameters. These parameters are fixed at 24 characters on the 600 board based systems and 64 characters on the 540 basad systems. The user can, in fact, specify narrower

screen width in response to the terminal width question. For instance, the user could answer 20 ⟨carriage return⟩ , so the terminal width question on a 600 based system would confine the display width to 20 characters. This option is useful primarily to limit display width for television sets which are overscanning and for use in conjunction with narrow width printers. However, a better solution to overscanning problems of the video display is to reduce the overscan of the monitor.

LOAD and SAVE Commands

ROM BASIC contains two special reserved words for use with I/O devices, LOAD and SAVE.

LOAD Command

The LOAD command can be executed in the immediate mode or as part of a stored program. When the BASIC interpreter encounters a LOAD command, it switches input from the keyboard to serial input port 1. Input to BASIC continues from this port until the user depresses the space bar on the terminal or a program modifies a flag in memory by the statement POKE 515,0 (POKE 515,255 also turns on LOAD). Serial port 1 is normally connected to an audio cassette interface but can easily be expanded to accept input from a modem or RS-232 terminal in conjunction with or in lieu of the audio cassette port. The LOAD command in conjunction with the video cassette interface has the capability to read programs from cassette and to read in data from cassette files. By the addition of hardware, the same LOAD command can be utilized to support an external terminal or modem.

SAVE Command

The SAVE command can be executed in the immediate mode or as part of the stored program. When the BASIC interpreter encounters the SAVE command, it routes output to both the video screen and serial port 1. This mode of operation continues until a LOAD command is encountered which automatically clears the SAVE condition. The serial port is normally connected to an audio cassette output interface so that the SAVE command can normally be used for saving programs and storing data in cassette files. By the addition of hardware, it can also support output to a modem, external terminal and printer. (POKE 517,0 turns off SAVE, POKE 517,255 also turns on SAVE.)

LOADing and SAVEing BASIC Programs

To SAVE a program on cassette:

    1. Rewind the tape.
    2. Type SAVE ⟨carriage return⟩ .
    3. Type LIST but not ⟨carriage return⟩ .
    4. Start the recorder in the record mode.
    5. As soon as the leader passes over the tape head, type
        ⟨carriage return⟩ .
    6. When the listing is complete, turn off the tape recorder
        and optionally type LOAD ⟨carriage return⟩   ⟨space bar⟩
        ⟨carriage return⟩ to revert to normal computer operation.

To LOAD programs which are stored on tape into the computer, proceed as follows:

1. Rewind the tape.
2. Cold start the machine or type NEW ⟨carriage return⟩ .
3. Type LOAD but not ⟨carriage return⟩ .
4. Start the tape in play-back mode.
5. As soon as the leader passes over the head, type ⟨carriage return⟩ .
6. Upon completion of a LOAD, turn off the tape recorder type ⟨space bar⟩ and then ⟨carriage return⟩ .

Cassette Data Files

The simplest way to store data on cassette is to store the data imbedded in data statements which have line numbers as part of the program. Thus, the data comes along with the program when it is loaded in but requires that the entire program be re-stored when the data is changed. This is not an unreasonable handicap in small programs. To further simplify the use of data statements for data storage, BASIC allows the LIST command to be imbedded as a statement as part of the BASIC program. For example, a short BASIC program utilizes data in data statements between lines 100 and lines 200. To allow the user to change these data statements, a portion of the program can have a statement such as LIST 100-200 which when executed will selectively LIST that portion of the program. And following this statement, it can have a statement such as PRINT" CHANGE THE ABOVE STATEMENTS AS NECESSARY BY TYPING A LINE OVER AGAIN": STOP. This statement when executed will prompt the user to change data statements and will then discontinue program execution by a break.

Cassette Based Sequential Files

The most sophisticated cassette based data storage technique is to utilize sequential data files on cassette. To construct a sequential data file on tape, the program must simply execute a SAVE command followed by a series of PRINT commands which print out the desired information. This information will appear on the video screen and will also be stored in sequential fashion on cassette. The individual entries will be delineated or separated by carriage returns. To input from cassette data files, the BASIC program must execute a LOAD command then execute INPUT statements. These INPUT statements will be answered by the cassette instead of the keyboard. This technique is straightforward with two minor tricks. The first problem is that the programmer must be certain that the information on cassette is presented to the computer after each INPUT statement is executed. Obviously, if the information was outputted from the cassette before the BASIC program executed an INPUT statement, it would be lost. This is not a problem in simple programs because the SAVE command automatically places 10 nulls before each output before it is placed on cassette. These 10 pad characters will provide sufficient

delay for normal programs. If time consuming calculations are performed such as dimensioning a large array, it will be necessary for the programmer to provide additional delays when the file is printed out to insure proper timing on play back. The second problem or trick associated with cassette based data files deals with noise pulses in the leader portion of the tape and between data files when the tape recorder is turned on and off. This situation can be handled effectively as follows. At the beginning of every data file, the outputting program should place a 72 character string followed by a carriage return. This string will be utilized as a sync field. The program that then reads the data file will throw away its first input, that is, it is assumed that the play back process will start in the middle of this long sync string which will be discarded. In cases of severe noise, a beginning of record string can be utilized which must be input and verified by the INPUT program before it accepts valid data. This procedure should not be necessary under normal circumstances.

Outputting to Printers

If a printer is connected to serial port 1, the user can directly LIST programs and send print outs to the printer by executing the SAVE command followed by the operations desired. The resulting print outs will appear on the video display and the printer.

Other Devices

The same procedure is utilized for the audio cassette system and can be directly applied to modems, terminals and other devices which are placed on serial port 1 in lieu of or in conjunction with the audio cassette interface.

The following section is a reference manual which characterizes the syntax of the 8K BASIC-in-ROM. This section is not meant to be a tutorial on the use of BASIC but simply a reference manual to be used in conjunction with a standard BASIC text book.

SPECIAL CHARACTERS

Character	Use
@ (shift P)	Erases line being typed
(shift O)	Erases last character being typed
Carriage return	Must be used after each line typed
Control C	Interrupts program execution or listing returns to command mode.
: (colon)	Allows multiple statements per line
Contol O	Typing a control O once surpresses output until another control O is typed.
?	? can be used instead.of print.

OSI 8K BASIC is a "standard" BASIC with additional string handling capability and I/O commands, as well as the following features.

OSI BASIC allows multiple statements per line via ":". Next without a variable can be used when FOR-NEXT statements are not nested. END statements are not necessary. Question marks can be used instead of "PRINT". "LET" is optional. No spaces are required in BASIC. These features allow highly efficient memory usage when necessary.

Variables can be two characters long. Longer variables can be used but only the first two characters will be utilized. The first character must be alphabetic, the second can be alphabetic or numeric. Long variables can not contain words used by BASIC such as NEW, SIN, and so on. Since spaces are not necessary BASIC would interpert a variable such as "ANEW" as a variable A and the command "NEW" and would erase the program.

EXAMPLES:

LEGAL	ILLEGAL
A	IA
A1	#B
AZ	TOO
BEQ	RGOTO
APPLE	NEW 1
TUESOAY	FREQUENCY

Note: that variables AZ1 and AZ2 would be treated the same since BASIC looks only at the first two characters.

COMMANDS

NAME	EXAMPLE	COMMENTS
LIST	LIST	Lists program
	LIST 100	Lists program from line 100. Control C stops program listing at end of current line.
NULL	NULL 3	Inserts 3 nulls at the start of each line to eliminate change return bounce problems. Null should be 0 when entering paper tapes from Teletype readers. When punching tapes Null>=3. Higher settings are required on faster mechanical terminals.
RUN	RUN	Starts program execution at first line. All variables are reset. Use an immediate GOTO to start execution at a desired line.
	RUN 200	GOTO 200 with variables reset.
NEW	NEW	Deletes current program.
CONT	CONT	Continues program after Control C or STOP if the program has not been modified. For instance a STOP followed by manually printing out variables and then a CONT is a useful procedure in program debugging.
LOAD	LOAD	Used in cassette and Disk BASIC only.

OPERATORS

SYMBOL	EXAMPLE	COMMENTS
=	A=10 LET B=10	LET is optional
-	C=-B	Negation
↑ (Shift/n)	X↑4	X to the 4th power
		C↑D with C negative and D not an integer gives an FC error.
*	C=A*B	Multiplication
/	D=L/M	Division
+	Z=L+M	Addition

-8-

	J=255.1-X	Subtraction
<>	10 IF A<>B THEN 5	Not equal
>	B>A	B greater than A
<	B<A	B less than A
<=, =<	B<=A	B less than or equal to A
=>,>=	B=>A	B greater than or equal to A
AND	IF B>A AND A>C THEN 7	If both expressions are true then--.
OR	IF B>A OR A>C THEN 7	If either expression is true then--.
NOT	IF NOT B>A THEN 7	If B<=A then--.

AND, OR, and NOT can also be used in Bit manipulation mode for performing Boolean operations of 16 bit 2s complement numbers (-32768 to +32767)

EXAMPLES

EXPRESSION	RESULT
63 AND 16	16
-1 AND 8	8
4 OR 2	6
10 OR 10	10
NOT 0	-1
NOT 1	-2

OPERATOR EVALUATION RULES: Math statements evaluated from left to right with * and / evaluated before + and - Parentheses explicitly determine order of evaluation.

Precedence for evaluation

1) By parenthese
2) ↑
3) Negation
4) * /
5) + -
6) =,<>, <,>, <=, >=
7) NOT
8) AND
9) OR

STATEMENTS

In the following examples

V or W is a numeric variable, X is a numeric expression,
X$ is a string expression, I or J is a truncated integer.

NAME	EXAMPLE	COMMENTS
DATA	10 DATA 1,3,7	Data for REAO statements must be in order to be read. Strings may be read in DATA statements.
OEF	10 OEF FNA (V)=V*B	User defined function of one argument.
OIM	110 OIM A (12)	Allocates space for Matrices and sets all matrix variables to zero. Non dimensioned variables default to 10.
ENO	999 ENO	Terminates program (optional)
FOR, NEXT	10 FOR x=.1 to 10 STEP.1 20_____ 30 NEXT X	STEP is needed only if X is not incremented by 1. NEXT X is needed only if FOR NEXT loops are nested if not NEXT alone can be used variables and functions can be used in FOR statements.
GOTO	50 GOTO 100	Jumps to line 100
GOSUB, RETURN	100 GOSUB 500 500 . . . . 600 RETURN	Goes to subroutine, RETURN goes back to next line number after the GOSUB
IF...THEN	10 If X=5 THEN 5 10 If x=5 THEN PRINT X 10 If X=5 THEN PRINT X:Y=Z	If the statement is true Then the following will be executed including multiple statements of that line.
IF...GOTO	10 IF X=5 GOTO5	Same as if THEN with line number
ON... GOTO	100 ON I GOTO 10, 20, 30	Computed GOTO If I=1 then 10 If I=2 then 20 If I=3 then 30

PRINT	10 PRINT X 20 PRINT "Test"	Prints value of expression Standard BASIC syntax with , ; " formats
READ	490 READ V, W	Reads data consecutively from DATA statements in program
REM	10 REM	This is a comment for non- executed comments.
RESTORE	SOO RESTORE	Restores Intial values of all DATA statements
STOP	100 STOP	Stops program execution re- ports a BREAK. Program can be restarted via CONT.

FUNCTIONS

Function	Comment
A8S (X)	For X=>0 ABS(X)=X For X<0 ABS(X)=-X
INT (X)	INT (X) = largest integer less than X
RND (X)	Generates a random number between 0 and 1
	RND (O) generates the same number always
	RNO (X) with the same X always generates the same sequence of random numbers NOTE (B-A)* RND (1)+A generates a random number between B and A
SGN (X)	IF X>0 SGN(X)=1 IF X=<0 SGN(X)=0
SIN (X) COS (X) TAN (X) ATN (X)	Sine of X where X is in radians Same for COS, TAN, and ATN (ARC TAN)
SQR (X)	Square root
TAB (I)	Spaces the print head I.
USR (I) See I/O section	
EXP (X)	E↑X where E is 2.71828
FRE (X)	Gives number of 8ytes left in the workspace.
LOG (X)	Natural LOG to obtain base 10 logs use LOG(X)/LOG (10)

POS (I)	Gives current location of terminal print head.
SPC (I)	Prints I spaces, can only be used in print statements.

STRINGS

Strings can be from 0 to 255 characters long. All string variables end in $ ex. A$,B9$, HELLO$.

Strings can be dimensioned equated, printed, read from Data statements, etc.

STRING FUNCTIONS

ASC (X$)	Returns ASCII value of first character in string.
CHR$ (I)	returns a I character string equivalent the ASCII value above.
LEFT$ (X$,I) RIGHT$(X$,I)	Gives left most I characters of string X$ Gives right most I character of string X$
MID $ (X$,I,J)	Gives string subset of string X$ starting at I^{th} character for J characters. If J is omitted, goes to end of string.
LEN (X$)	Gives length of string in bytes.

STR$ (X)	Gives a string which is the character representation of the numeric expression of X. Example X=3:1
	                    X$=STR$(X)
	                    X$="3.1"

VAL (X$)	Returns string variable converted to number. Opposite of STR$(X)

I/O

The following features of OSI 8K BASIC are usefull primarily for I/O control. The user should be extremely careful with these statements and functions since they manipulate the memory of the computer directly. An improper operation with any of these commands can cause a system crash, wiping out BASIC and the users program, thus requiring a complete reload of the computer.

STATEMENT/FUNCTION	COMMENT
PEEK (I)	Returns the decimal value of the specified memory or I/O location. (Oecimal) Example: X=PEEK (64256) Loads variable X with the 430 Board's A/D converter output. ($FBOO_{hex}$)
POKE I,J	Loads memory location I (decimal) with J (Oecimal) I must be between 0 and 65536 and J must be between 0 and 255 Example: 10 Poke 64256, 255 loads FBOO with FF (Hex) thus loads the 430 Board's O/A port 0 such that its output is +2 volts.
WAIT I,J,K	Reads status of memory location I (Oecimal) exclusive OR's with K then ANO's the result with J until a non zero result is obtained. If K is omitted, it is zero. Wait is used for fast service of input status flags. Example: Wait X,1 will wait until Bit zero of memory location X goes low then BASIC will continue.

The high speed servicing of flags via the WAIT command allows the programmer to service medium speed devices such as line printers or industrial equipment directly in BASIC.

USR: The USR function allows linkage to machine language routines such as ultra-fast device handlers, etc. The USR function calls only one machine language routine and can pass one integer value to the machine language routine so that 65,000 actual user routines are possible.

The beginning of the user subroutine must be poked into $23E_{hex}$ (low) and $23F_{hex}$ (high). The USR routine can use up to 8 levels of subroutines (16 stack locations) without page swapping.

The USR function can obtain the argument of the function by calling the routine pointed to by 6 (low) and 7 (high). This routine will

place the value of the argument in $AE_{(hex)}$ (high part) and $AF_{(hex)}$ (low part). To pass a value back to BASIC, the high part is placed in A and the low part is placed in Y and the subroutine pointed to by 8 and 9 should be called. If this function is not called USR (X) will equal X. An RTS returns from USR to BASIC. All registers can be modified by the user routine without affecting BASIC, however, no page zero locations can be modified! The POKE instruction can also be used to change the USR function call.

INTERRUPTS

For Interrupting routines of any significant length, page zero and page one should be swapped out to higher memory, or memory partitioning (A_{16} and A_{17} on late model OSI memory boards) should be used.

CONVERTING OTHER BASICS TO RUN
ON OSI 6502 8K BASIC

MATRIX subscripts: Some BASICS use [ ]
OSI BASIC used ( ).

Strings:

OTHER	OSI
DIM A$(I,J)	DIM A$ (J)
A$ (I)	MID$ (A$,I,1)
AS (I,J)	MID$ (A$,I,J-I+1)

Multiple assignments: B=C=0 must be rewritten as B=0:C=0. Some BASICs use / to delimit multiple statements per line. Use ";". Some BASICs have MAT functions which will have to be rewritten with FOR-NEXT loops.

TABLE 2-1. BASIC ERROR CODES.

CODE		DEFINITION
DD	D	Double Dimension: Variable dimensioned twice. Remember subscripted variables default to dimension 10.
FC	F	Function Call error: Parameter passed to function out of range.
ID	I	Illegal Direct: Input or DEFIN statements can not be used in direct mode.
NF	N	NEXT without FOR:
OD	O	Out of Data: More reads than DATA
OM	O	Out of Memory: Program too big or too many GOSUBs, FOR NEXT loops or variables
OV	O	Overflow: Result of calculation too large for BASIC.
SN	S	Snytax error: Typo, etc.
RG	R	RETURN without GOSUB
US	U	Undefined Statement: Attempt to jump to non-existent line number
/∅	/	Division by Zero
CN	C	Continue errors: attempt to inappropriately continue from BREAK or STOP
LS	L	Long String: String longer than 255 characters
OS	O	Out of String Space: Same as OM
ST	S	String Temporaries: String expression too complex.
TM	T	Type Mismatch: String variable mismatched to numeric variable
UF	U	Undefined Function

Ohio Scientific BASIC for the 6502, 8K version 3.2

Commands

CONT ✓　　LIST ✓　　　　NEW ✓　　　　NULL ✓　　　　RUN ✓

Statements

CLEAR ✓　　DATA ✓　　　OEF ✓　　　DIM　　　　END ✓　　FOR ✓

GOTO ✓　　GOSUB ✓　　IF...GOTO　　IF...THEN ✓　　INPUT ✓　　LET

NEXT ✓　　ON...GOTO　　ON...GOSUB　　POKE ✓　　　PRINT ✓　　READ

REM ✓　　RESTORE ✓　　RETURN ✓　　STOP ✓

Expressions

Operators

-, +, *, /, ↑, NOT, AND, OR, ><, <>, > =,< =,=　　　RANGE 10-32 to 10+32

Variables

A, B, C, . . ., Z and two letter variables
The above can all be subscripted when used in an array
String variables use above names plus $, eg. A$

Functions

ABS(X) ✓　　ATN(X) ✓　　COS(X) ✓　　EXP(X) ✓　　FRE(X) ✓　INT(X) ✓

LOG(X) ✓　　PEEK(I) ✓　　POS(I) ✓　　RND(X) ✓　　SGN(X) ✓　SIN(X) ✓

SPC(I) ✓　　SQR(X) ✓　　TAB(I) ✓　　TAN(X) ✓　　USR(I) ✓

String Functions

ASC(X$) ✓　　CHR$(I) ✓　　FRE(X$)　　LEFT$(X$,I) ✓　LEN(X$) ✓　MID$(X$, I, J) ✓

RIGHT$(X$,I) ✓　　　　　　STR$(X) ✓　　　　　　VAL(X$) ✓

Special Characters

● Erases line being typed, then provides carriage return, line feed.
← Erases last character typed.
CR Carriage Return – must be at the end of each line
: Separates statements on a line.
CONTROL/C Execution or printing of a list is interrupted at the end of a line.
　"BREAK IN LINE XXXX" is printed, indicating line number of next statement
　to be executed or printed.
CONTROL/O No outputs occur until return made to command mode. If an Input
　statement is encountered, either another CONTROL/O is typed, or an error occurs.
? Equivalent to PRINT