BASIC2 (12K)

Using this Manual.

This manual is intended to explain BASIC2 to users with some experience of using simple BASIC on an interactive disc-based system. It will give the syntax of the BASIC2 commands, statements and functions as well as describing in detail those features of BASIC2 over and above what has come to be known as standard BASIC. If you have never used BASIC before, you should use this manual in conjunction with one of the many books describing BASIC. In the manual, the term DOS means the Disc Operating System you are using (usually SCIDOS) and the symbol ~ will be used to indicate the depression of the RETURN key. In describing the BASIC2 syntax, the characters <, >, [and] are not part of BASIC2 but indicate the intended construction of the statements thus:

Words enclosed in angular brackets < and > suggest a construct at that point e.g.

SAVE <filename> REM <remark> DELETE <line range>

Constructs enclosed in brackets are optional e.g.

LIST [<line range>] RUN [<start line>] PRINT [<I/O list>]

BASIC2 ignores spaces, except when they are in strings. Any spacing given in the syntax examples will therefore be optional and only included here for the sake of clarity. Note that string quantities should be delimited by a quotes, e.g. SAVE "TEST"

Making a Copy of BASIC2.

The file name of BASIC2 on the master disc is BAS.COM. Copy this file onto another disc and store the master. Do not lose the master disc as it may be necessary for you to return it as proof of purchase, should the need arise.

General.

Name of File	BAS.COM	
Version	2.2	
Length	45 pages	
START Address	00100H	
Restart Address	00103H	
USR	00106H	Jump to USR subroutine
USROUT	00109H	Called to get USR variable
USRIN	0010CH	Called to pass USR variable
INTIN	0010FH	Called to pass an integer
EXTERR	00115H	Display "External Error" message

Reserved Words

The following words have a special meaning in BASIC and must not appear as, or as part of, the names you use for variables.

ABS	AND	ASC	ATN	BYE	CALL	CHR$	CLEAR	CLOSE
CONT	COS	CREATE	DATA	DEF	DELETE	DIM	DIR	EDIT
ELSE	END	EOF	ERASE	ERL	ERR	ERROR	EXP	FN
FOR	FRE	FSAVE	GET	GET$	GRAPH	GOSUB	GOTO	HEX$
IF	INP	INPUT	INT	LEFT$	LEN	LET	LINE	LIST
LLIST	LLVAR	LNULL	LOAD	GO	LOG	LOOKUP	LPOS	LPRINT
LTRACE	LVAR	LWIDTH	MERGE	MID$	NEW	NEXT	NOT	NULL
ON	BREAK	OPEN	OR	OUT	PEEK	PLOT	POINT	POKE
POS	PRINT	PUT	QUOTE	RANDOMIZE		READ	REM	RENAME
RENUMBER	RESET	RESTORE	RESUME	RETURN	RIGHT$	RND	SAVE	SGN
SIN	SPC	SQR	STEP	STOP	STR$	TAB	TAN	TEXT
THEN	TO	TRACE	USR	VAL	VARADR	VPEEK	VPOKE	WAIT
WIDTH	?							

Using BASIC2.

To load and use BASIC2 from SCIDOS, type BAS<return>. To load BASIC2 and execute a program immediately, type BAS<space><filename><ret> e.g. BAS DEMO In the former mode, once it has completed its initialisation, the prompt Ready: will appear. This means that the Basic is ready for you to type something. All input to Basic is buffered, that is it is stored in a particular area of memory and only passed to the interpreter after you press return. This means that you may alter the line using Delete or BS (Control H on some keyboards) as often as is necessary during the input of a line. The action taken by BASIC2 when you press return will depend upon how the line begins. Lines which start with a digit are taken to be lines of a program being entered and are partially compiled and then put into the area of memory being used for program storage. Otherwise the line is executed immediately. Most Basic statements may be used in this DIRECT mode. Thus typing:

```
PRINT 12+9~
```

will produce the answer 21 as soon as the return is typed. Certain statements are meaningless in the direct mode, e.g. NEXT X. Instead of acting on these types, Basic prints an error message.

BASIC2 character set.

BASIC2 responds to the full ASCII character set as well as to certain control codes, viz:

Ctrl E - Toggles the printer echo flag Ctrl U - Abandons an input line
Ctrl Z - Interrupts the interpreter

In addition, most of the non-alphanumeric characters have special meanings under certain circumstances. These are explained at various points in the manual and are also summarised here:

" - delimits a string value & - precedes a hexadecimal constant
- precedes a unit number : - links statements on the same line
$ - indicates a string variable ? - is an alternative for PRINT
, and ; separate items in the same statement

I/O in BASIC2.

To provide for future expansion of BASIC,a system of 'unit numbers' is used in BASIC I/O operation. Each I/O device has a unit number and typical I/O syntax is

PRINT[#<unit>,]<I/O list>

At present the unit numbers are:

0	VDU/Keyboard
2	Listing device
10	Disc file

<unit> can be any expression evaluating to a valid unit number. If the unit number specification is omitted it defaults to zero so that VDU/Keyboard I/O is the default, maintaining syntax compatibility with BASICs which do not use unit numbers. BASIC version 2 offers "L" versions of various commands (viz. LLIST, LLVAR, LNULL, LPRINT, LPOS, LTRACE and LWIDTH) which direct the output to the listing device but these are just for compatibility and may not appear in future BASICs. For this reason, it is recommended that unit numbers are used in writing new programs.

Data I/O.

INPUT [LINE][#<unit>,][<prompt>,]<I/O list>

INPUT supplies data to a program. INPUT LINE provides increased flexibility in inputting. It implies that there will only be one string variable in the I/O list and that every character typed in, including commas and spaces, up to the ~ will be placed into that string.

PRINT [#<unit>,][<I/O list>] and LPRINT [<I/O list>]

Outputs data to the specified unit. ? is an alternative to PRINT

DATA <constant list> READ <variable list> RESTORE[<line number>]

READ is used in the same way as INPUT, except that the data required is obtained from one or more DATA statements in the program itself. RESTORE may specify the line to which the data pointer is set. The default is the first DATA line.

GET(<expression>) and GET$(<expression>)

These functions are for timed, single character input. If the expression is present, BASIC waits for the value of the expression times ten milliseconds. If no expression is given, BASIC waits indefinitely. If no key is pressed in the time specified, 0 is returned by GET and a null string by GET$. Otherwise the value of, or a one-byte string containing, the key pressed is returned.

PUT [#<unit>,]<I/O list>

PUT is used for single character output. Strings in the I/O list are output one character at a time.

WIDTH [#<unit>,]<line width> and NULL [#<unit>,]<no of chars>,<char>

or LWIDTH <line width> and LNULL <no of chars>,<char>

The statements allow the I/O parameters for units to be independently set. Default width for all I/O devices is infinite.

QUOTE [#<unit>,]<parameter>

QUOTE is primarily intended for use with disc I/O but can be made to apply to any unit. When executed, it causes several changes on that unit's output stream:

1. All I/O list items are output close-spaced (i.e. as if separated by ; in the list).
2. Commas are automatically inserted between items as they are output.
3. Strings quantities output are delimited by the ASCII character specified by <parameter>.

At initialisation, none of the above conditions is set. To reset to the initial state after using QUOTE, execute QUOTE [#<unit>,] 0

INP(<port number>)

This function returns the value read from the given Z80 port.

OUT <port number>,<value>

Outputs the given value (0 to 255) to the given Z80 port.

WAIT <port number>,<bit mask>,<bit state>

Inputs from the given port, ANDs this with the second argument to mask unwanted bits and then exclusive ORs the result with the third argument. If the result of this is not zero, i.e. the masked input did not match the bit state pattern, the entire process is repeated. The three arguments must be in the range 0 to 255 and are used as 8 bit bytes in the logical operations. Care must be taken in using WAIT, as it is possible for the mask and match to be impossible, in which case control can only be restored by resetting the computer.

PEEK(<memory address>)

This function returns the value found at the specified memory location.

POKE <memory address>,<data>

Writes the data (0 to 255) into the specified memory location (0 to 65535).

VPEEK(<memory address>)

This function returns the value found at the specified VDU location.

VPOKE <memory address>,<data>

Writes the data (0 to 255) into the specified VDU location. Both VPEEK and VPOKE accept addresses in the range 0 to 8191 (denary). The organisation of video memory is that location 0 is the CRTC pointer register, location 1 is the CRTC register and locations for the VDU memory are 5120-8191 in mode 0 and 2048-8191 in mode 1.

TAB(<line position>)

This function-like expression is used in PRINT statements to set the printing position in the current ouptut line. It cannot be used to move the position backwards.

SPC(<number of spaces>)

This function-like expression is used in PRINT statements to insert extra spaces in the output line.

POS and LPOS

These functions return the position of the cursor or print-head in the current output line.

File I/O.

There are several commands and functions which are only used when inputting from or outputting to a data file on disc. These are:

OPEN [#<unit>,]<filespec>

OPEN opens an already existing data file for input to the computer.

CREATE[#<unit>,]<filespec>

CREATE creates an empty file, ready for output from the computer. If the file already exists you will be asked to confirm that you want to over-write it.

CLOSE[#<unit>]

CLOSE should be used to end output to a file.

ON EOF [GOTO <line number>]

ON EOF sets up a monitoring condition so that the statement need only be executed once and from that point onwards, BASIC monitors input from the file for the end of file and jumps as specified in the GOTO part of the statement on finding the end of file character - even if it occurs in the middle of an INPUT statement involving several variables. The form ON EOF without the GOTO part cancels the EOF monitoring.

EOF

EOF as a statement in a program simulates the reception of an end of file character by BASIC.

Several SCIDOS-type commands are available is BASIC2. They are:

ERASE <file name>

RENAME <old name>,<new name>

DIR [<filespec>]

DIR list all or part of the disc directory as specified in <filespec>. If omitted, the listing is of all .BAS type files on the currently logged drive.

RESET [<drive number>]

RESET updates the DOS system parameters. RESET must be used whenever a disc in the current drive is changed or when you want to "log" a different drive as the current drive. In the latter case, the drive to be logged is given as the <drive number>, 0 corresponding to drive A:, 1 to drive B: etc.

Program I/O in BASIC2.

SAVE <filespec> and FSAVE <filespec>

The SAVE command saves programs in ASCII. FSAVE saves the program in the compressed, internal, format i.e. as the program is actually stored in computer memory. This results in several differences which can make FSAVE of value. Programs stored in internal format load and are saved faster and take up less space on the disc. They cannot be listed using the SCIDOS LIST command and so are more secure. A disadvantage of internal format is that it is specific to the particular interpreter that created it, so FSAVE would not be used for a program stored on a disc which might then be put into another machine using a different interpreter. Also, see MERGEGO.

LOAD <filename> and LOADGO [<line number>]

LOAD clears the program area and then loads the specified file, whatever its format. Default file type is .BAS. LOADGO combines the LOAD and RUN functions. The default line number is the first.

MERGE and MERGEGO [<line number>]

MERGE loads a program into memory without erasing the program currently in memory. MERGEGO combines the MERGE and RUN functions. Default line number is the first. Note that MERGE and MERGEGO only function with ASCII format files.

LOOKUP(<filename>)

LOOKUP tests to see if the file exists. Default file type is "BAS", default drive is the current one. -1 is returned if the file exists, 0 if it does not.

LIST [<line range>] and LLIST [<line range>]

Lists all or part of a program.

NEW

Clears all program statements and the variable storage area.

DELETE <line range>

Deletes one or more program lines.

RENUMBER [<new number>][,<increment>]

RENUMBER renumbers a program. Default for the new number and the increment is 10.

Graph Plotting.

GRAPH [<parameter>]

GRAPH clears the VDU and selects the graph plotting mode where scrolling is restricted to the bottom 4 lines of the VDU and the rest becomes a non-scrolling plotting area. The following parameters have the following effects:

0 or no parameter	Selects 192x84 plotting mode
1	Selects 192x180plotting mode
8	Enables reverse video plotting of PLOT strings
9	Disables reverse video plotting of PLOT strings

TEXT [<parameter>]

TEXT restores full screen scrolling. The following parameters have the following effects:

0 or no parameter	Selects 96 columns, 32 rows display mode
1	Selects 96 columns, 64 rows display mode
8	Enables reverse video display
9	Disables reverse video display

In both GRAPH and TEXT, any other value of parameter returns an error.

PLOT <X co-ordinate>,<Y co-ordinate>[,<parameter>]

In the PLOT statement, parameter may be a string or numeric expression. If it is a string, it is plotted, starting at the specified co-ordinate pair. If it is numeric, 0 plots a black point, 1 plots a white point and 2-255 plot the character generator equivalent of that value. Default for the parameter is the previously used numeric value.

LINE <X co-ordinate>,<Y co-ordinate>,[<numeric expression>]

LINE allows the plotting of as straight a line as is possible between the last plotted point and that specified. The numeric expression is as for the parameter in PLOT.

POINT(<X co-ordinate>,<Y co-ordinate>)

This function returns a 0 if the point referenced is blank, 1 if it is lit and the ASCII value if there is a non-graphic character at that point.

User Defined Activities.

USR(<variable>)

USR allows BASIC2 to execute a machine code subroutine and to receive and return a 16 bit value. A jump to the subroutine must be patched into BASIC2 at address 00106H. The value of the variable, converted to a 16 bit integer, may be obtained in the DE registers by making a call to 00109H, and a 16 bit value may be returned as the value of the function, in the AB registers, by making a call to 0010CH. To terminate the subroutine an ordinary RET instruction is used.

CALL <address>[,<I/O list>]

CALL provides a more powerful technique than USR for calling machine code subroutines. A call is made to the specified address, at which point registers BC will hold a count of the number of variables to be passed to the routine which will all have been converted to 16 bit notation and pushed onto the Z80 stack in their order in the I/O list (i.e. last one POPs off the stack first).

DEF FN<label> (<dummy variable>)=<expression>

This statement is used to define a function for later use in a program. <label> is any valid variable name. The function is used as:

FN<label>(<values to be passed to the function>)

For example, a definition of:

DEF FNZ9(X)=X+1/X means that FNZ9(5) will return 5.2

Line Editing.

EDIT <line number>

This command invokes the line editor which copies the specified line into a buffer, displays its number, positions a pointer at the start of the text of the line and then enters the "review" mode. In this mode, certain keys have special functions and some of these may be made to repeat their action by prefacing them with a number, represented here by n. These keys do not appear on the VDU when depressed. In the review mode:

A	reloads the buffer with the line.
nD	deletes and echoes between backslashes, n characters.
nFx	moves the pointer to just before the nth occurence of x.
H	deletes all undisplayed text and enters the insert mode.
I	enters the insert mode.
L	lists the current edited line and resets the pointer to its start.
Q	abandons an editing session.
nR	replaces the next n characters from the keyboard.
n<space>	moves the pointer n positions to the right.
n<DEL>	moves the pointer n positions to the left.
~	leaves EDIT, replacing the old line with the new version.
ESC	abandons a command.

In the "insert" mode, every key acts normally on the line, with the exception of:

ESC	which returns to the review mode, and
~	which leaves EDIT, replacing the old line with the new version.

Program Control

RUN [<line number>]

Clears the variable space, initialises all variables to zero or the null string and starts execution at the specified line. The default is the first line of the program.

BYE

Ctrl C can be used to exit from BASIC, although BASIC intercepts a Ctrl C and checks that you really want to exit. The command BYE is the preferred method for leaving BASIC though, as before leaving it closes and updates any file which might have been left open, thus preventing accidental loss of file data.

GOTO <line number> GOSUB <line number> RETURN

These statements transfer control unconditionally to the specified line.

ON <variable><term><list of line numbers>

ON transfers control to the nth line number where n is the integral part of the variable. The <term> may be either GOTO or GOSUB. The only limit on the list of line numbers is the length of the program line. If n exceeds the number of line numbers given, execution passes to the next program statement.

ON BREAK [GOTO <line number>]

ON BREAK intercepts keyboard interrupts with Ctrl Z. The form without the GOTO clears the function.

FOR <variable>=<starting value> TO <limit value> [STEP <step value>]

The default for <step value> is +1. A FOR loop is always executed once.

NEXT [<loop variable list>]

The default for <loop variable list> is the innermost loop.

IF <conditional expression><true s'ment(s)> [ELSE <false s'ment(s)>]

The <conditional expression> may contain the relational operators = < > <> <= or >= linked by the logical operators NOT AND or OR. The true statements are of the form:

GOTO <line number> or THEN <line number> or THEN <statement(s)>

If the ELSE clause does not transfer control, execution passes to the next statement.

END

Terminates execution.

STOP

Causes a break in program execution.

CONT or CONTINUE

Continues execution after a Ctrl Z or a STOP. During such a break in the program, the variables may be listed using LVAR and variable values altered using direct commands. The program itself may also be listed, but must not be altered.

LVAR and LLVAR

Lists the values of the variables in use.

TRACE <logical value> and LTRACE <logical value>

Causes all line numbers to be listed as the lines are executed. A logical 0 disables TRACE. All other values enable it.

Error Handling

ON ERROR [GOTO <line number>]

This statement alters the response to an error in BASIC. The form without the GOTO cancels the pending ON ERROR condition.

ERROR <error number>

This statement forces the specified error. (See appendix)

ERL and ERR

These functions aid error processing. ERL returns the line in which the error occurred (0 for a direct error) and ERR returns the number of the error (See appendix).

RESUME

RESUME returns control from the error-processing routine, re-executing the statement that caused the error.

String Functions.

ASC(A$)	CHR$(X)	LEFT$(A$,X)	LEN(A$)
VAL(A$)	STR$(X)	RIGHT$(A$,X)	MID(A$,X,Y)

HEX$(X) returns the hexadecimal value of X

In all functions, the resulting string must not be longer than 255 characters.

Numeric Functions.

ABS(X)	ATN(X)	COS(X)	EXP(X)	FRE(X)	INT(X)
LOG(X)	RND(X)	SGN(X)	SIN(X)	SQR(X)	TAN(X)

As well as these standard numeric functions, the following are available:

ALG(X) returns the common anti-logarithm of X

CLG(X) returns the common logarithm of X

VARADR(X) returns the memory address of the first byte of the variable X.

Numeric variables are stored in 4 consecutive bytes. The first 3 bytes hold the mantissa of the number, least significant byte first. The most significant bit of the mantissa will always be a 1 and so this bit is assumed, and the actual bit is used to represent the sign of the number, 0 meaning positive. The fourth byte is the binary exponent, offset by 128. As an example, the hex string 00 00 E0 81 represents the number -1.75.

If X is a string variable, the address returned is that of a 4 byte data block. The first two bytes are the string length, least significant byte first. The second two contain the address at which the string is stored, least significant byte first,

Apart from VARADR, the X in the above syntax represents a numeric constant, variable or expression.

Arithmetical and Logical Operators.

The following list gives the execution priority of operators in descending order:

1. Expressions in parentheses
2. ^ (raise to the power)
3. - (negation)
4. * and / (multiply and divide)
5. + and - (add and subtract)
6. relational operators
7. NOT (logical complement)
8. AND (logical AND)
9. OR (logical OR)

Operators with the same priority are evaluated from left to right in an expression. When logical operations are made, the operands are converted to a 16 bit integer form. The operands must therefore both be in the range 0 to 65535 or -32768 to 32767 if an error is not to occur.

Logical operations may be included as mathematical operations, e.g

```
PRINT 4 AND 6          => 2
```

as 0000000000000100 logically ANDed with 0000000000000110 is 0000000000000010.

In IF statements a result of 0 is taken to mean logical false and anything else to be true.

General Commands and functions.

CLEAR <string space>

Reserves memory space for string storage.

DIM <matrix list>

Reserves memory space for matrices. Matrices may have up to 255 dimensions. It is not possible to re-declare matrices in the same program.

LET <variable>=<expression>

LET assigns values to variables. The word LET is optional.

RANDOMIZE

RANDOMIZE is a statement to randomly change the seed used by the pseudo-random number generator. The Random Number generator is re-initialised by RUN so that the same sequence of numbers is generated each time a program runs - unless RANDOMIZE is used.

REM

Prefaces a user remark to be ignored during execution.

Miscellaneous

Hexadecimal constants, preceded by an ampersand (&), may be used anywhere in SCITEX Basic except as line numbers, They must be in the range 0 to FFFF.

Initial string allocation is 256 characters.

Appendix - Error Messages

1.	External Error	22.	Recovered
2.	NEXT without FOR	23.	Name not found
3.	Syntax Error	24.	Can't verify ASCII files
4.	RETURN without GOSUB	25.	Can't MERGE internal files
5.	Out of data	26.	Missing statement number
6.	Illegal function	27.	Read error
7.	Arithmetic overflow	28.	RESUME without error
8.	Out of memory	29.	Record too large
9.	Undefined statement	30.	Invalid unit number
10.	Subscript out of range	31.	Missing file name
11.	Redimensioned array	32.	No input file
12.	Can't divide by zero	33.	No output file
13.	Illegal direct	34.	Invalid device
14.	Type mismatch	35.	Invalid file name
15.	No string space	36.	Write error
16.	String too long	37.	Wrong internal format
17.	Too complex	38.	File not found
18.	Can't continue	39.	Directory full
19.	Undefined user call	40.	No disc space
20.	Illegal EOF	41.	Output close error
21.	Files Different	>41.	Unknown error