SCITEX BASIC (18K)

SCITEX Basic is an extended BASIC interpreter some 18 kbytes in length. Its main features are:

* Extensive I/O facilities.
* PRINT USING allowing formatting of string and numeric output.
* 12 digit internal precision.
* Buffered I/O, made entirely through the disc operating system.

Using this Manual.

This manual is intended to explain SCITEX Basic with its extensive facilities, to users with some experience of using simple BASIC on an interactive disc-based system. It will give the syntax of the SCITEX Basic commands, statements and functions as well as describing in detail those features of SCITEX Basic 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 and the symbol ~ will be used to indicate the depression of the RETURN key. In describing the SCITEX Basic syntax, the characters <, >, [and] are not part of SCITEX Basic 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>]
```

SCITEX Basic 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.

Making a Copy of SCITEX Basic.

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

Using SCITEX Basic.

To use SCITEX Basic, type BASIC followed by a return. SCITEX Basic allows you to reserve space between the bottom of BDOS and the top of its workspace for USR routines etc. In response to the prompt Highest Memory? which is given when BASIC is loaded into memory, type the highest memory address (in decimal) that you want it to use. If you do not wish to reserve any space, just a ~ will do. SCITEX Basic will then tell you how much memory you have available for program use and fully initialise itself. If the message Out of Memory appears, enter a different value for the highest memory.

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 SCITEX Basic 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.

SCITEX Basic character set.

SCITEX Basic responds to the ASCII character set and to certain control codes, viz:

Ctrl E - Interrupts the interpreter
Ctrl I - Moves to the next multiple of 8 position on the I/O line
Ctrl J - Moves the I/O position to the start of the next line
Ctrl R - Prints the current input buffer contents
Ctrl T - Prints the number of the line being executed at that moment
Ctrl U - Abandons an input line

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:

! - begins a format specifying line
" - delimits a string value
- precedes a unit number and is a format character
$ - indicates a string variable
% - precedes the output of an unformattable quantity
& - precedes a hexadecimal constant
' - precedes a remark appended to a statement, or a string format field
. - means "the current line"
: - separates statements on the same line
, and ; separate items in the same statement
? - is an alternative for PRINT
@ - precedes a random address in disc I/O

Data I/O in SCITEX Basic.

Input and output (I/O) of data in SCITEX Basic is based upon the concept of UNITS. Each unit is given a number in the range 0 to 255. The current assignment of unit numbers is:

UNIT	DEVICE	
0	VDU/Keyboard	(The SCIDOS CON: device)
1	LOAD/SAVE Device	
2	LIST Device	(The SCIDOS LST: device)
3	2nd Input Device	(The SCIDOS AXI: device)
4	2nd Output Device	(The SCIDOS AXO: device)
5-9	Reserved	
10-255	Disc I/O	

In an I/O statement the unit is designated by the syntax

#<unit>,

where <unit> is any expression evaluating to a valid unit number, e.g.,

```
PRINT#2,X,Y,A$
```

Except for LOAD, SAVE and MERGE operations (for which it is not necessary to specify a unit), all I/O operations may be made via any unit capable of supporting it. (e.g. it is not possible to input from the LIST device). In most cases, the unit specification is optional, the default being zero. This maintains compatibility with standard BASICs.

Disc I/O.

SCITEX Basic places no restriction on the mix of data being supplied to disc files. String, numeric and binary data may be output to or input from files in exactly the same way it is with other I/O devices. Both Sequential Access (i.e. starting at the beginning of the file and working through to the end) and Random Access (i.e. being able to drop straight into a file at the start of any sub-division or "record") files are possible, as well as combinations of these two techniques. For disc operations, any unit number from 10 to 255 may be used, the only limitation within this range being the user memory available (see CLEAR). The association between a particular disc file and a unit number is made by the OPEN statement. Note that the units refer to files, not discs or disc drives. Thus several units may be associated with files on the same disc - indeed, even with the same file.

When the file is to be accessed randomly, it is possible to specify the actual point in the file at which I/O is to occur in the I/O statement. The syntax for this fuller specification is

#<unit>@<addr>,

where <addr> is any expression evaluating to a positive integer less than 4194304

OPEN #<unit>,<mode>[,<file name>[,<record size>]]

OPEN is primarily intended for disc units although it will have an effect upon units 0 to 4. <mode> is a single character string. Mode options are:

"I" - only input operations are allowed via this unit

"O" - only output operations are allowed via this unit

"R" - full I/O operations possible (disc units only)

"U" - as for "R" except that, unless otherwise specified, each ouput operation commences at the same point in the file as the start of the previous input and each input operation follows on from the end of the last operation. This particular mode is intended as a means of updating record files.

Note that units 0 to 4 are always open. OPENs to these units re-initialise the parameters (see OPTION) for that particular unit. In addition, for units 0 and 2, Ctrl L (Form Feed) is sent to the device; for unit 4, 16 OFFH bytes are sent.

In the OPEN statement, <file name> is a string expression evaluating to a valid DOS file specification. If the file type is omitted it defaults to .BAS. If the drive code is omitted it defaults to the currently logged drive. The action taken on opening a disc file depends upon the mode selected. For mode:

I - If the file exists the association is made. Otherwise an error occurs.

O - If the file exists an error occurs. Otherwise the file is created and the association is made.

R - If the file does not exist it is created. The association is the made.

U - As for I

The effect of an OPEN is to set a "file pointer" to zero, i.e. on the first byte of the file, so that unaddressed access to the file starts at its beginning. The final argument, <record size>, is an integer in the range 1 to 32767 and is a means by which files may be accessed randomly. When addressed access is made to the file, the file pointer is made equal to the record size times the random address <addr>, given in the I/O statement. The default record size is 1, so if a record size is not given in the OPEN statement for that unit, the file pointer and the random address are the same. The default for the random address is 0 so if it is not given, the file pointer is zero whatever the record size, giving standard, sequential, file access. For example:

After declaring a file for random access, based on a record size of 80 characters, thus:

```
OPEN#11,"R","ALFRED",80
```

INPUT#11,A will use the current value of the file pointer as the point in the file from which data is to be obtained. Data will thus be read sequentially from the file - unless SETLOC is used to directly modify the pointer.

INPUT#11@N,A will set the file pointer to N times 80, i.e. it will read data from the eightieth record in the file ALFRED.BAS

After,

OPEN#27,"R","CAKE"

INPUT#27,A will use the current value of the file pointer as the point in the file from which data is to be obtained. Data will thus be read sequentially from the file - unless SETLOC is used to modify the pointer.

INPUT#27@N,A will set the file pointer to N times 1.

CLOSE [#<unit 1>[,#<unit 2> ...]]

CLOSE is primarily intended for disc units although it has an effect upon units 0 to 4. It is impossible to close units 0 to 4. A CLOSE to units 0 or 2 results in an output of a Ctrl L (Form Feed) and to unit 4, the output of Ctrl Z (End of File) followed by 16 0FFH bytes. For disc units the action depends upon the mode set for the unit. For mode:

I - No action is taken.

O - A Ctrl Z (End of File) is written to the disc buffer. Then as U.

R - As U.

U - The buffer for that unit is updated to the disc and the disc directory is updated.

In all modes, the workspace allocated to the unit is released for re-assignment by an OPEN statement. Note that if no units are specified in the CLOSE statement, all disc units are closed.

CLEAR [<string space>][,<number of units>]

Each open disc unit requires 181 bytes of workspace. This workspace, together with that for string storage, is allocated by the CLEAR statement. As unit space is returned to the system whenever a unit is closed, <number of units> need only be the maximum number of units to be simultaneously open. When using CLEAR, either argument may be omitted, in which case it remains the same. Note that when the number of units is set there must be no open files as CLEAR clears the total unit workspace and would thus destroy any data not yet sent to a unit.

```
OPTION [#<unit>,]<option>,<arg 1>[,<arg 2>]
```

OPTION allows the system parameters for the particular unit to be modified. <option> is a single character string thus:

W WIDTH - <arg 1> is the line length required.

N NULL - <arg 1> is the number and <arg 2> the code for the NULL character.

Q QUOTE - <arg 1> is the string delimiter. 0 means no delimiter.

P PROMPT - <arg 1> is the prompt character for INPUT operations.

The default values for these parameters are presently:

UNIT	W	N	Q	P
0	96	0,0	0	63
2	96	3,0	0	
3		not applicable		
4	253	0,0	34	
10-255	253	0,0	34	

Several I/O statements and functions have been provided in SCITEX Basic to augment unit-based I/O.

```
OUTBYTE [#<unit>[@<addr>],]<I/O list>
```

OUTBYTE is used for single byte output. In the I/O list, numeric quantities must be in the range 0 to 255. String quantities are output byte by byte.

```
SETLOC #<unit 1>@<addr 1>[,#<unit 2>@<addr 2> ... ]
```

SETLOC provides a means of setting the address for disc I/O independently of the actual I/O operation, simplifying the structure of the I/O statements as well as making the one I/O statement suitable for both sequential and random use.

ON EOF [#<unit>][GOTO [<line number>]]

ON EOF sets the action to be taken when an end-of-file is encountered during input. If no unit is given, the condition is assumed to apply to all open units without specific ON EOF conditions pending. If the GOTO or the line number is missing, the ON EOF condition for that unit is cancelled.

EOF [#<unit>]

EOF as a statement simulates an end-of-file from that unit.

EOF(<unit>)

This function returns -1 if an end-of-file occurred on the last I/O operation at that unit. Otherwise 0 is returned.

ERR(<unit>)

This function returns -1 if an error occurred on the last I/O operation at that unit. Otherwise 0 is returned.

LOC(<unit>)

This function returns the value of the file pointer for that unit.

LOF(<unit>)

This function returns the number of bytes in the current extent of the disc file.

POS(<unit>)

This function returns the number of bytes output to the current line.

BYTEPOLL(<unit>)

This function returns -1 if a byte is available from the specified unit. Otherwise 0 is returned.

BYTE(<unit>)

This function returns the value of the next byte read from the unit.

BYTE$(<unit>)

This function returns a one byte string consisting on the next byte read from the unit.

Program I/O in SCITEX Basic.

In entering program lines, Ctrl I (tab) and Ctrl J (line feed) may be used to format the text without terminating the program line, these two characters being ignored during execution. A line entered as:

```
100<Ctrl I>IF X=5 THEN<Ctrl J><Ctrl I>I=0:<Ctrl J><Ctrl I>W=W+X
```

would list as:

```
100     IF X=5 THEN
        I=0:
        W=W+X
```

Each line may be up to 255 characters long.

AUTO [<starting line>[+]][,<increment>]

AUTO aids program input by providing the line numbers for you. The + option adds the increment to the starting line before the first number is produced. Default for both values is 10. In the AUTO mode, a + after the displayed line number means that a line with that number already exists. To keep the old line, just press ~. To leave AUTO, terminate the current line and type Ctrl E.

FIND <delimited text>[,<line range>]

FIND finds a string of up to 32 characters delimited by any printing character other than a comma which does not appear within the string itself. The lines in which the string appear are listed. The default for the line range is the entire program area. Example:

```
FIND ?X=LOG(?,100-200
```

REPLACE <delimited text 1><delimited text 2>,<line range>

REPLACE is an extension of FIND, replacing <delimited text 1> with <delimited text 2>, unless this would make the line exceed 254 characters, in which case an error occurs and the command is terminated. Note that the line range is not optional. Example:

```
REPLACE /SIN(//COS(/,10-500
```

SAVE <filename>[,A]

SAVE saves the current program area under the DOS file specification given. The default file type is .BAS. The file is saved in the compressed format used in SCITEX Basic itself unless the ,A suffix is added, in which case the file is saved as ASCII characters. Files saved in internal format are more compact and secure, the DOS being unable to list them. The drawback of internal format is that it is specific to SCITEX Basic and so the file could not be loaded using another interpreter.

RESAVE <filename>[,A]

As SAVE, but no error occurs if the file already exists.

LOAD <filename>

LOAD clears the program area and then loads the specified file, whatever its format. Default file type is .BAS.

LOADGO <filename>[,<start line>]

LOADGO loads and executes a program. The default start line is the first one.

MERGE <filename>

MERGE will load an ASCII program file into the memory without initially clearing the program area.

PRIVACY <password>

When this statement is included in a program, no lines may be added to or deleted from the program and the commands AUTO, COPY, DELETE, EDIT, FIND, LIST, RENUMBER, REPLACE, SAVE and RESAVE must be followed by the password, which can be any string value. The PRIVACY statement must be the first one in a program line. To simplify this manual, the syntax for a command using a password is only shown for LIST, but is the same for other commands.

LIST [<password>,][#<unit>[@<addr>],][<line range>]

Lists all or part of a program.

COPY <new line>[,<increment>]=<line range>

Providing that the new lines will not overwrite existing ones or fall within the range of lines to be copied, the specified lines will be copied to the new limits. The default increment is 10.

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

RENUMBER renumbers part or all (if no start line is given) of a program. Default for the new number and the increment is 10. By specifying a start line, a "window" may be opened up in the middle of a program.

NEW

Clears all program statements and the variable storage area.

DELETE <line range>

Deletes one or more program lines.

. is a shorthand for "the current line". If a line has just been type incorrectly, it may be corrected using EDIT.~ If an error has halted a program, the error line may be examined with LIST.~ AUTO may be re-entered after a Ctrl E by using AUTO+.~

DOS Operations.

Several DOS-type commands are available is SCITEX Basic. They are:

ERASE <file name>

RENAME <old name>,<new name>

DIR [Ø<unit>[@<addr>],][<file name>]

RESET

RESET updates the DOS system parameters. RESET must be used whenever a disc is changed.

LOOKUP(<file name>)

This function returns -1 if the file exists, 0 if it does not.

BYE

Leaves SCITEX Basic and returns to DOS. All disc units must be closed before BYE is used, as otherwise information may be lost.

In all file name specifications, any string constant or variable may be used which meets the DOS specification. Default drive is the currently logged one. Default file type is .BAS.

Data I/O.

INPUT [LINE][#<unit>[@<addr>],][<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>[@<addr>],][USING <format>;][<I/O list>]

<format> may consist of a string containing a single field description to be used repetitively until the I/O list is exhausted, or a line number at which will be found the complete specification for the formatted output. In the latter case, the line begins with an exclamation mark (!).

When formatting numeric variables,

`#` represents a digit position, several adjacent # characters specifying a field in which the displayed variable will be right justified, spaces being used to fill any vacant positions in the field.

`.` may be inserted in the field, in which case the result is positioned and, if necessary, rounded to align its decimal position with the point.

`+` may be placed at the beginning or end of the field, in which case a sign will always be displayed in that position.

`-` may be placed at the end of the field, in which cases negative results will be displayed with a trailing sign.

`**` two or more asterisks preceding the field represent digit positions and also indicate that vacant positions in the field are to be filled with asterisks rather than spaces.

`$$` two or more dollar symbols preceding the field represent digit positions and also result in a dollar symbol being displayed at the immediate left of the first digit. The last two features may be combined thus: **$

`,` may be inserted to the left of a decimal point in a field to indicate that commas are to be inserted after every three digits. Note that each comma inserted will need a digit position in the overall field specification.

`^^^^` at the end of a field indicates that the number is to be printed in scientific notation.

If a numeric quantity cannot be fitted to a field, it is printed in its entirety, preceded by a % sign.

String fields begin with an apostrophe (') which itself becomes part of the field. This may then be followed by the number of L E R or C characters (not mixed) required to make up the field length. The significance of the different letters is:

L The string is left justified in the field, extra characters being lost at the right.

E As L, except that the field is extended if necessary to display the entire string.

R The string is right justified in the field, extra characters being lost at the left.

C The string is centre justified, extra characters being lost at the right. Any other characters which could not be part of a valid field terminate field "blocks" and are themselves displayed literally.

```
10 PRINT USING "$$#.##___";1.238,22,PI,12345

 $1.24___$22.00___ $3.14___%$12345.00___
```

```
20 PRINT USING 200;123456789,"string",PI,PI,"OK!"
200 !
**#############,.# and then:'CCCCCCCCCC +#.#### #.######^^^^'LLLL

****123,456,789.0 and then:  string    +3.1416 0.314159E+01OK!
```

PRECISION [<digits>]

PRECISION causes all numeric output, except via PRINT USING, to be rounded to the number of digits specified. 0 or no figure sets the maximum precision, 11 digits.

READ #<unit>[@<addr>],<I/O list>

READ with a unit number requests binary data from that unit.

WRITE #<unit>[@<addr>],<I/O list>

Writes binary data to that unit. In both READ and WRITE, numeric data is restricted to 0-255 and the unit specified must be capable of binary I/O.

MAT READ #<unit>[@<addr>],<I/O list>

Loads binary data to a matrix. For multi-dimensional matrices the last subscript varies most rapidly.

MAT WRITE #<unit>[@<addr>],<I/O list>

Outputs matrices as binary data. For multi-dimensional matrices the last subscript varies most rapidly.

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>

SCITEX Basic 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 amd match to be impossible, in which case control can only be restored by completely 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).

LVAR [#<unit>[@<addr>]]

Lists the values of the variables in use.

TRACE [#<unit>[@<addr>],]<logical value>

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

TAB(<line position>)

This 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>)

SPC is used in PRINT statements to insert extra spaces in the output line.

__User Defined Activities.__

USR(<variable>)

USR allows SCITEX Basic to execute a machine code subroutine and to receive and return a 16 bit value. A jump to the subroutine must be patched into SCITEX Basic at address 0106H. The value of the variable, converted to a 16 bit integer, may be obtained in the DE registers by making a call to address 0109H, and a 16 bit value may be returned as the value of the function, in the AB registers, by making a call to address 010CH. To terminate the subroutine an ordinary RET instruction is used.

CALL <address>[,<arg 1>[, ... ,<arg n>]]

CALL calls a subroutine at the given address. Before making the call, SCITEX Basic pushes the return address onto the Z80 stack, followed by each of the listed arguments, converted to a 16 bit value. When the call is made, the BC registers will contain the number of pushed variables and the HL registers the stack pointer value prior to the pushing of all the variables. The purpose of having this latter value available is that should the subroutine wish to return before all of the variables have been used, a LD SP,HL followed by a RET will "clean up" the stack.

DEF FN<label> (<list of dummy variables>)=<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,Y)=X+1/Y   means that   FNZ9(12,5) will return 12.2

It is possible to spread the definition of a function over several lines, with the possibility of several exit points and returned values. The syntax is:

```
DEF FN<label> (<list of dummy variables>)
        .
    <function body>
        .
    FNEND [<function value>]
```

The function body may contain any SCITEX Basic statement except for another multi-line function definition. FNEND serves to mark the end of the function body as well as to provide a point at which the value to be returned can be computed. If there is to be more than one exit point from the definition, FNRETURN has the same effect as FNEND, but may occur anywhere in the function body. As an example, this function calculates a factorial using an FNRETURN to return a value of 1 for the special case of factorial 0:

```
100 DEF FNF(I)
110 IF I=0 THEN FNRETURN 1
120 FNEND FNF(I-1)*I
```

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.

Error Handling.

ON ERROR GOTO [<line number>]

Allows the user to control the action taken when an error occurs. This statement, once executed, is valid until an error has occurred, or until the statement without the line number is executed.

RESUME [<line number>] or RESUME NEXT

RESUME is the last statement in an error handling routine. On its own, execution returns to the statement at which the error occurred. Otherwise a line number may be specified. RESUME NEXT returns execution to the statement after the one in which the error occurred.

ERL

A function (no argument) which returns the line number in which an error occurred.

ERR

A function (no argument) which returns the error code for the last error (see appendix A).

ERR(<unit>)

A function which returns -1 if an I/O error was encountered during the last I/O operation via that unit. Otherwise 0 is returned.

ERROR <error number>

This forces an error specified by the error number. (see appendix A).

General Commands and functions.

CONT or CONTINUE

Continues execution after a Ctrl E 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.

KILL <matrix>[, ... ,<matrix n>]

Releases space allocated to the named matrices. No subscripts should be given.

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.

LET <variable>=<expression>

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

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.

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. RESTORE may specify the line to which the data pointer is set. The default is the first DATA line.

EXCHANGE <variable 1>,<variable 2>

High speed exchange of variables.

GOTO <line number> GOSUB <line number> RETURN [<line number>]

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, GOSUB or RETURN. The only limit on the list of line numbers is the length of the program line. If n exceeds the number of line number, execution passes to the next program statement.

FOR <loop 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>]

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

EXIT [<line number>][,][<variable name>]

EXIT is the correct statement for early termination of a FOR-NEXT loop. The comma is only required if both arguments are present. EXIT with no arguments terminates the innermost loop with the loop variable holding its current value and execution passing to the statement after the NEXT. If <line number> is given, then execution passes to this line instead. If <variable name> is given, its loop and all those within it are simultaneously terminated.

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 XOR EQV or IMP. 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.

REM

Prefaces a user remark to be ignored during execution.

A remark can be added to a statement, preceded by ' and terminated by a : or a ~.

STOP

Causes a break in program execution.

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)

RND may be used without an argument in which case it is as though X=1 in RND(X).

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

EE returns the constant 2.7182818285

PI returns the constant 3.1415926536

FIX(X) returns the truncated integer part 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 6 consecutive bytes. The first 5 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 sixth byte is the binary exponent, offset by 128. As an example, the hex string 00 00 00 00 E0 81 represents the number -1.75.

If X is a string variable, the address returned is that of a 6 byte data block. The first is the string length, the next is zero, the next two contain the memory address at which the string starts, least significant byte first and the last two are undefined.

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

String Functions.

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

MID$(<string variable>,<starting char>[,<length>])=<string value>

This is a second and quite different use of MID$. The <string value> will replace the characters in the <string variable>, starting at the <starting char> position. A <length> can be given, and if the <string value> is not of this length, it is either truncated at the right or padded with spaces to make it so. For example, if A$="ABCDEFGHIJ" then

```
MID$(A$,2,4)="1234"    =>    A$="A1234FGHIJ"
MID$(A$,2,4)="Q"       =>    A$="AQ   FGHIJ"
```

This special form of MID$ is a very fast way of splitting and combining strings.

FIX$(A$,X) returns A$, truncated or padded with spaces to length X

HEX$(X) returns the hexadecimal value of X

SPACE$(X) returns a string of X spaces

STRING$(A$,X) returns a string consisting of A$ repeated X times

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

INSTR(<string value 1>,<string value 2>[,<starting char>[,<length>]])

INSTR is a function which searches <string value 1> to see if it contains the <string value 2>. If it is found, the function returns the position in <string value 1> at which <string value 2> starts, otherwise 0 is returned. It is possible to specify the character position in <string value 1> at which the search starts and also to truncate or pad <string value 2> to a specified length before the search is made (as in MID$ above). For example,

```
INSTR("ABCDEFGHIJ","DEF")       => 4
INSTR("ABCDEFGHIJ","DEG")       => 0
INSTR("ABCDABCD","C")           => 3
INSTR("ABCDABCD","C",6)         => 7
```

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. \ (integer division)
6. MOD (modulus)
7. + and - (add and subtract)
8. relational operators
9. NOT (logical complement)
10. AND (logical AND)
11. OR (logical OR)
12. XOR (logical EXclusive OR)
13. EQV (logical EQuiValence)
14. IMP (logical IMPlication)

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 and so:

AND is true if any of the sixteen bit pairs are both at logic 1

OR is true if any of the 32 bits are at logic 1

XOR is true any of the 16 bit pairs are different

EQV is true if all of the bit pairs are the same

IMP is true if any of the first term's 16 bits are at logic 0, or if any of the second term's 16 bits are at logical 1

Integer division produces the integer part of the division only, i.e. 17\5 produces the result 3, exactly. MOD produces the remainder of an integer division, i.e. 17 MOD 5 produces the result 2, as 17 divided by 5 is 3 remainder 2.

Miscellaneous.

RANDOMIZE is a statement to randomly change the seed used by the pseudo-random number generator.

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. Initial unit allocation is 0.

....

Appendix A Error Codes and Messages

1. NEXT Without FOR
2. Syntax Error
3. RETURN Without GOSUB
4. Out of Data
5. Illegal Function
6. Arithmetic Overflow
7. Out of Memory
8. Undefined Statement
9. Subscript Out of Range
10. Re-DIMensioned Array
11. Cant Divide by Zero
12. Illegal Direct
13. Type Mis-match
14. No String Space
15. String Too Long
16. Expression Too Complex
17. Cant CONTinue
18. Undefined User FN Call
19. Illegal EOF
20. Bad Statement Number
21. Recovered
22. FNRETURN Without FN Call
23. Missing Statement Number
24. Record Too Large
25. Undefined Matrix
26. Invalid Unit Number
27. RESUME Without ERROR
28. Directory Full
29. Extension Error
30. No Disk Space
31. Input File Not Found
32. Duplicate Output File
33. Output CLOSE Error
34. Invalid OPEN Type
35. Invalid File ID
36. Invalid OPTION
37. Not OPEN Input
38. Not OPEN Output
39. Not Random Device
40. File Not OPEN
41. Not Binary Unit
42. No File Space
43. Invalid Record Number
44. Disk PROTECTion Violation
45. EXIT without FOR
46. Not Available
47. Not Implemented
48. Unknown ERROR