The Reference Section provides a guide to all the BASIC words available on the Enterprise, along with their purposes and methods of use. Some of them are mentioned only briefly in the Tutorial section, others are not mentioned at all.

It is to be hoped you will experiment with all these words and discover for yourself the full extent of the Enterprise's potential. If you know BASIC already, you will find, in the main, that this section is your best guide to the Intelligent Standard BASIC (copyright Intelligent Software Ltd, 1984) provided on the Enterprise.

GENERAL RULES

Upper and lower case letters are interchangeable in BASIC keywords and identifiers, e.g. FOR, For, for and fOr are all the same word.

A program line may be up to 250 characters long, with a line-number 1 to 9999. It may include several statements, separated by colons; anything permitted after THEN in an IF/THEN statement may be incorporated on a multi-statement line.

An identifier can be up to 31 characters long, and all characters are significant. The identifier can contain letters, numbers, full stops and 'underline' characters; the first character must be a letter.

! is used to mark off the rest of the line as a comment.

The interpreter deletes spaces before and after the line-number and first keyword, and at the end of the line. It then indents the program for every new block. FOR, DEF, DO, HANDLER, SELECT and WHEN will indent the next line by 2 spaces. ELSE and CASE inside an indented block are placed 2 characters to the left. LOOP, END and NEXT terminate the indentation. Line-numbers are printed with leading spaces to maintain a straight edge.

```
  1     LET A=0
 10     DO WHILE A<10
100       LET A=A+1
110       SELECT CASE A
120       CASE 1
130         PRINT "first time"
140       CASE ELSE
150         PRINT "not first time"
160       END SELECT
170       PRINT A
```

```
180    LOOP
190    GOTO 1
1000   END
```

For further reference on the syntax and conventions of the BASIC, see the Draft Proposal for Standard BASIC from ANSI committee X3J2/82-17.

Keywords are given in **BOLD CAPITALS** in the left-hand margin of the page. The formats of the commands using the keyword are given in normal print, and examples are given in *italics*.

MULTIPLE PROGRAMS

IS-BASIC on the Enterprise gives the facility for several programs to be in the computer at one time. Each program has its own line numbers and its own variables.

A program can be referred to either by number, or by a name given on a PROGRAM line. See in particular the commands **CHAIN, EDIT** and **PROGRAM**.

At any particular time, one of the programs (by default, program Ø) is the 'current' one, on which commands such as LIST and RENUMBER will operate. The number of this program is shown on the 'status line' at the top of the screen.

Program Ø can use approximately 42 k of memory. Other programs are limited to 32 k each.

EXTENSIONS

The facility exists to provide extensions to BASIC, which may be either loaded from cassette or disk or included in an add-on stack unit for the computer. Explanation of the extra commands or functions will be provided in the instructions accompanying such products.

ABBREVIATIONS

The following abbreviations are used in this reference:

chan — channel-number
id — identifier (e.g. variable name)
str — string
var — variable
expr — expression
relop — relational operator (i.e. >, > =, etc.)
para — parameter

line-number

line-number text
line-number space
line-number

Adds or replaces a program line. If the line-number is followed by only a space, then a line containing an '!' is inserted. If the line-number is followed by nothing, then the line is deleted. Only executed in immediate mode. Clears variables.

Note that all commands or statements which clear variables also close any open channels in the range 1-99 inclusive (see **OPEN**).

ALLOCATE

ALLOCATE expr

Used in connection with machine code subroutines. Moves up the program source to create a gap of the specified number of bytes, where the user's machine code will go. Sets the location counter to the first free byte in the gap. Note that this destroys all variables, so it should only be used at the start of the program.

ASK

ASK machine-option var

Enquires about some option (e.g. KEY RATE); see 'Machine Options', 'Video Options' and 'Sound Options' sections. Compare also **SET** and **TOGGLE**. The variable will take on the current value of the machine-option.

e.g. *ASK KEY RATE A*

assigns the current keyboard repeat rate to the variable A.

AUTO

Special editing command which prints line-numbers automatically. Only works in immediate mode.

AUTO
AUTO AT 10 STEP 1
AUTO STEP 10

Default starting line-number is 10. The default step size is 1. New lines replace old ones with the same line-numbers.

AUTO can be cancelled by pressing 'stop'.

CALL

CALL function
CALL function (para-list)

Used to call a function (either built-in, or defined by DEF), when no result is required from the function.

Any expression following CALL will be evaluated, and the result ignored. *CALL USR (A,B)+USR(C,D)* will therefore call two machine code USR programs.

Can be executed in immediate mode.

CAPTURE

CAPTURE FROM £ chan TO £ chan

Captures input from first channel and substitutes it for input expected from second channel. Input from second channel is locked out until 'stop' is pressed, an end-of-file condition arises on the first channel, or an error occurs. CAPTURE FROM a particular channel can also be terminated by giving £255 (normally invalid) as the TO channel, in a later statement. Default first channel is £1Ø7. Default second channel is £Ø.

CASE

See **SELECT** block.

CAUSE EXCEPTION

CAUSE EXCEPTION expr

Defines an error and assigns it to the category denoted by the expression; user values should be between 1 and 999, since these will never be used by BASIC.

CHAIN

CHAIN £ chan: program-number
CHAIN £ chan: "name"

Used for executing BASIC programs from the current program, and for calling other application ROMs in the machine.

Parameters may be passed by value from one program to the other, e.g.:

CHAIN "My Program" (1, "Fred")

See also **PROGRAM**.

CLEAR

CLEAR £ chan:
CLEAR ENVELOPE
CLEAR GRAPHICS
CLEAR QUEUE sound-source-number

CLEAR SCREEN
CLEAR SOUND
CLEAR TEXT

Clears various options. Can be executed in immediate mode.

CODE

CODE string
CODE variable-name = string

Used in connection with machine code subroutines. Copies a string to the position indicated by the current location counter. If a variable is given, this takes the value of the location counter. The location counter is left pointing to the byte following the string, which is assumed to contain the machine code. The variable-name can later be used to call the routine, or to form the destination address of jumps etc.

CLOSE

CLOSE £ chan

Closes the channel and frees any buffers.

CONTINUE

As a command in immediate mode, it restarts the program at the next line after a STOP command or press of the 'stop' key.

Used in a program as an exit from an exception handler, it resumes at the statement following the one which caused the exception.

COPY

COPY FROM £ chan TO £ chan

Copies the contents of one channel to a second channel (both channels must be open). The copy terminates on end-of-file, an error, or the 'stop' key. Default input is channel Ø. Default output is channel 1Ø4.

COPY

copies from £Ø to £1Ø4.

COPY FROM £5

copies from £5 to £1Ø4. Requires channel 5 to have been opened.

DATA

DATA data-list

Allows the inclusion of a list of constants, numbers and/or strings, for subsequent READing. See **READ**.

DEF

DEF numeric-id = expression
DEF numeric-id(parameter-list) = expression
DEF string-id = string-expression
DEF string-id(parameter-list) = string-expression

One-line function definition:

DEF AVERAGE (X,Y) = (X + Y)/2

DEF block: this is a group of statements that can be called as a function returning a value in the expression, or as a procedure statement. There are several small changes from the ANSI definition. See also **CALL, EXIT DEF, NUMERIC** and **STRING**.

def-line
  any number of statements of blocks
end-def-line

def-line:
DEF numeric-id
DEF numeric-id (parameter-list)
DEF string-id
DEF string-id(parameter-list)

end-def-line:
END DEF

```
DEF ANSWER (A$)
  IF UCASE$(A$(1:1)) = "Y" THEN
    ANSWER = 1
  ELSE IF UCASE$(A$(1:1)) = "N" THEN
    ANSWER = 0
  ELSE
    ANSWER = -1
  END IF
END DEF
```

The scope of variables at any point in a program is dynamic—that is, it depends upon the history of which lines have been executed, and not upon the static

layout of the program.

```
100     NUMERIC FRED
110     LET FRED = 1
120     CALL Q
130     PRINT FRED
140     END
200     DEF P
210       LET FRED = 123! This FRED is a global.
220     END DEF
300     DEF Q
310       NUMERIC FRED! A local FRED.
320       LET FRED = 0
330       PRINT FRED
350     END DEF
```

In this example, FRED is used both as a global and as a local. When line 210 is executed, the FRED at 310 gets changed to 123 and not the one at 100. The program will print 123 and 1. In a static scope language, the program would print 0 and 123; this may happen if the same program is run under a compiler BASIC.

Everything declared within a DEF block is local to that block, and allocated at each first execution of the declaration after the call. Anything not declared may be local or global depending on the history.

It is best to declare all variables at the start of each program or function in order to avoid unexpected results.

```
100     CALL P ! This call of P has I as local to P.
110     LET I = 9
120     CALL P ! This call of P changes the global I.
130     END
200     DEF P
210       LET I = 6
220     END DEF
```

In order to give consistent results, a line

```
90      NUMERIC I
```

should be added to the program; this will make I global in both calls of P.

The memory used for the storage of local variables is released when a function is exited. This

characteristic can be exploited for the efficient use of computer memory—for example, a temporary data array can be within a function.

Almost anything can be passed as a reference parameter. Normally parameters are passed by value, which means that copies are passed to the function and any operation inside the function does not change the external variables. Reference parameters take their type from the actual parameter, and any changes inside the function change the external variables also.

```
100     DEF SWAP (REF A,REF B)
110       NUMERIC T
120       LET T=A
130       LET A=B
140       LET B=T
150     END DEF
200     LET X=99
210     LET Y=23
220     CALL SWAP (X,Y)
230     PRINT X, Y

prints    23      99
```

Arrays and functions must always be passed by reference.

```
100     NUMERIC A(10)
110     OPTION ANGLE DEGREES
120     DEF P (REF FN, X)
130       PRINT FN(X),
140     END DEF
150     LET A(2)=66
160     CALL P(A,2)
170     CALL P (SIN,30)

prints    66      .5
```

Passing built-in and user functions can be very useful for library software. A graph-drawing function can have the function to be plotted passed as a parameter, a sort function can have the exchange and compare routines passed as functions.

Functions can call themselves recursively.

DELETE

DELETE line-description TO line-description,...

DELETE line-description—line-description,...
DELETE block-name

Deletes lines from the program. Only executed in immediate mode. Clears variables.

DELETE LAST
DELETE FIRST TO 100
DELETE 1 TO 199, 300, 500 TO 9999

Acceptable syntax is to use '-' instead of TO. If the first (or last) number in a range is omitted, it defaults to the first (or last) line of the program.

e.g. *DELETE FIRST-100, 500-LAST*
or *DELETE TO 100, 500-*
for *DELETE FIRST TO 100, 500 TO LAST*

Lines defining a function P can be deleted with *DELETE P*. DELETE on its own will remove all program lines; can be halted with 'stop' key.

DIM DIM array-list

Declares numeric or string arrays; lower bound defaults to 0 if not specified. One or two dimensions are allowed. Maximum length cannot be specified for a string by using DIM, so the default of 132 characters is used. (Compare **STRING**.)

DIM A(1 TO 10), FRED$(9), B(-7899 TO-7890)

Note: all the above have 10 elements.

DISPLAY DISPLAY £ chan: AT a FROM b TO c

Defines a window to display a segment of a text or graphics video page. Screen-row 'a' is the position where the top line of the segment will be placed. Parameters 'b' and 'c' are character-rows on the page which is to be displayed, and define the top and bottom lines of the segment. The numbering of character-rows follows the conventions for text, whether the page displayed is text or graphics. See **PRINT**.

DISPLAY GRAPHICS

Sets up 20 lines as graphics, and displays previous graphics page if one was open. Does not clear text page.

DISPLAY TEXT

Sets up full screen in text mode and displays full page of text if it was previously open. Does not clear graphics screen.

If only a small text page was previously open, then this is cleared, and a new full-size text page is opened.

DO

do-line
  any number of statements or blocks
loop-line

do-line:
DO
DO WHILE relational-expression
DO UNTIL relational-expression

loop-line:
LOOP
LOOP WHILE relational-expression
LOOP UNTIL relational-expression

The structure of a loop is defined as a block, with a DO line, the loop body, and a LOOP line. DO or LOOP cannot be placed on a conditional line.

```
DO WHILE A>3 AND A<10
  LET A=A+1
  PRINT A
LOOP
```

Control cannot be transferred from outside to inside of a loop. See also **EXIT DO**.

EDIT

EDIT program-number
EDIT "name"

Makes the specified program into the current one, so that LIST, RENUMBER, RUN etc. will operate on it.

ELSE

See **IF**.

END

Halts execution, marks the end of the program. Also **END DEF, END HANDLER, END IF, END SELECT** and **END WHEN** mark the end of their relevant blocks.

ENVELOPE

ENVELOPE £chan: NUMBER
a;b,c,d,e;f,g,h,i;...;RELEASE j,k,l,m;...

Defines a sound envelope to be used in conjunction with a controlling SOUND statement. The number 'a' which identifies the envelope must be in the range Ø-254.

Parameters 'b', 'c', 'd' and 'e' define the first phase of the envelope; 'b' gives the change of pitch in semitones (decimal places allowed), 'c' and 'd' specify the change in volume for the left and right speakers respectively, and 'e' gives the duration of the phase, in 'ticks' (one tick is 1/5Ø second).

The values for 'c' and 'd' are in the range Ø-63; they specify the change in volume as a proportion of the overall maximum volume allowed by the SOUND statement. A value of -63 will turn the sound off (any overshoot is ignored); the sound is assumed to be 'off' at the beginning of the envelope. If stereo equipment is not in use, the volume at any moment will be determined by the sum of the values (dependent on SOUND and ENVELOPE statements) for the left and right speakers.

The next phase is defined by 'f', 'g', 'h' and 'i'... For the number of possible phases, see **SOUND BUFFER**, under 'Sound Options'.

RELEASE is optional; it may be followed by any number of phases with their separate parameters. The 'release' phases are performed after the conclusion of the previous phases, or at the expiry of the SOUND duration if there is no following sound on the same channel.

EXIT DO
EXIT FOR
EXIT DEF

Breaks out of FOR, DO or DEF block. Not valid unless inside the right sort of block.

EXIT HANDLER

Breaks out of an exception handler, which propagates the exception to the surrounding environment. This will cause another exception handler to be activated,

either a user handler or the default handler.

FOR

for-line
  any number of statements or blocks
next-line

for-line:
FOR simple-variable = expression TO expression STEP expression

STEP can be omitted—the default STEP value is 1.

next-line:
NEXT
NEXT variable

The structure of a FOR loop is defined as a block, with a FOR line, the loop body, and a NEXT line. FOR and NEXT cannot be placed on a conditional line. Allowed in Minimal BASIC.

```
FOR Y=0 TO 10 STEP 2
  PRINT Y
NEXT Y
```

The value of the control variable after the loop has ended is the terminating value plus the STEP expression, i.e. Y will have the value 12 in the example above.

Nested FOR loops cannot use the same control variable. The limit and increment expressions are copied to hidden local memory on execution of the FOR line; these values cannot be changed by the body of the loop. Control cannot be transferred from outside to inside of a loop. See also **EXIT FOR**.

GOSUB

GOSUB line-number

Calls subroutine beginning at the line-number specified.

GOTO

GOTO line-number

Program execution is continued at the line-number specified. Can be used to exit FOR, DO, HANDLER or DEF blocks, but this is not recommended.

GRAPHICS

GRAPHICS
GRAPHICS HIRES/LORES colour-quantity-number
GRAPHICS ATTRIBUTE

The command GRAPHICS has the effect of closing and re-opening the default graphics and text pages (£1Ø1 and £1Ø2); it displays the default graphics page over most of the screen, but with four lines of text at the bottom.

GRAPHICS also establishes the default channel (1Ø1) for video machine options such as VIDEO MODE.

Valid colour-quantity numbers are 2,4,16 and 256. If nothing is specified for the colour quantity or the HIRES/LORES option, the values that were used for the previous GRAPHICS command will be re-used. For the significance of these values, see 'Video Mode', in the 'Video Options' section. Initially, GRAPHICS selects a high-resolution graphics page with 4 colours.

GRAPHICS ATTRIBUTE selects an 'attribute' mode of graphics in which each colour-cell (8 dots wide by 1 dot deep) can contain one 'ink' colour and one 'paper' colour. This mode combines a 16-colour palette with the same resolution as 4-colour HIRES graphics (resolution and colour-quantity cannot be specified by the user). Both printing and plotting commands may be given, although there can be interactive effects between the colours. Line modes 4-7 (see 'Video Options' section) are used for plotting in 'paper' colour instead of 'ink' colour.

HANDLER

HANDLER handler-name
  exception handler statements
END HANDLER

The HANDLER block is used for dealing with program exceptions caused by errors, the CAUSE EXCEPTION command, or machine interruptions.

The handler to be used is specified by the handler-name given in the current WHEN block.

See **RETRY**, **EXIT HANDLER**, and the functions **EXLINE** and **EXTYPE**.

Control can be transferred into an exception handler only as the result of an exception (not by a GOTO or GOSUB).

If an exception occurs inside the exception handler

the effect is similar to EXIT HANDLER, since control passes to the next outer level of handler (as specified by the next outer level of WHEN block). However, the former values of EXTYPE and EXLINE will have been replaced by new ones.

IF

IF relational-expression THEN line-number
IF relational-expression THEN simple-statement

Statements not allowed on an IF line are DATA, DEF, END, DIM, NUMERIC, STRING, a further IF, or any statement which introduces a block.

IF A>=3 AND A<=9 THEN 100
IF A>=3 THEN GOTO 100

if-line
  any number of statements or blocks
else-if-lines option
  any number of statements or blocks
else-line option
  any number of statements or blocks
end-if-line

if-line:
IF relational-expression THEN

else-if-line:
ELSE IF relational-expression THEN

There can be any number of ELSE IF lines.

else-line:
ELSE

end-if-line:
END IF

IF blocks can contain any statement which is not restricted to immediate mode.

```
IF A<10 THEN
  PRINT A
ELSE IF A>30 AND A<=40 OR A>50 THEN
  PRINT A+100
ELSE
```

PRINT B
END IF

The ELSE and ELSE IF lines can be used to break the block into sub-blocks with the usual meanings. ELSE may only be used once, but ELSE IF can be used as often as needed. Control cannot be transferred from outside to inside of an IF block.

IMAGE

IMAGE: format-specification

Used in conjunction with PRINT commands, to control the format of the output. The format-specification is a string containing characters which, in this context, have the following meaning.

Numeric format characters: —

, — prints a comma in the number.
$ — prints a floating dollar-sign preceding the sign.
+ — prints a floating space or '-' sign.
− — prints a floating '+' or '−' sign.
% — prints a digit, including leading zeros.
£ — prints a digit or space, trailing zeros after a decimal point.
* — prints a digit or leading '*'.
. — prints a decimal point.
^ — prints exponent part; minimum 4 characters.

If the number does not fit in the format space, an error is generated.

String format characters: —

< — left-justification of the string, in the field defined by '£' characters.
£ — prints a character.
> — right-justification of the string.

The 'justify' format character must start the field; if no 'justify' character is used, the string is centred.

The format in the IMAGE line starts immediately after the ':' and ends with the last printed character on the line, or the exclamation mark starting an end-of-line comment.

INPUT

INPUT £chan, IF MISSING action, AT row-expr, column-expr, PROMPT string: variable-list.

Reads data from channel into a list of variables. Default channel is the editor (channel Ø). Items of data read in to match with variables in the variable-list must be separated by commas.

INPUT PROMPT K$&"Enter next number please? ":N
INPUT A(I), B$

The IF MISSING and PROMPT parts can be in either order, or absent. The default input prompt is "? ". PROMPT replaces the default prompt with the string.

The AT option (with row-expr, column-expr) is independent of the PROMPT option.

IF MISSING is used if less data is received from the channel than is required by the variable-list. The action then taken follows the same rules as with READ.

See also **LINE INPUT**.

INFO

Prints out the amount of memory in the system and the number of unused bytes. A table of information about the programs in memory is also printed, in the following form:

program-number	number of bytes in program	first line of program

INFO clears all variables. Only executed in immediate mode.

LET

LET variable-list = expression

Simple assignment; LET is optional unless the variable name is the same as a keyword. Listing or saving the program causes the LET to be inserted so that the program conforms to the standard. Can be executed in immediate mode.

One value can be assigned to several variables:

LET A, B(4), C=Ø
A_VAR=A_VAR+1
A$,FRED$="He said"&"Don't"&". "&FRED$(1:J)
LET INPUT=3

LINE INPUT

Similar to INPUT, but reads a whole line (including commas, etc.) for each item in the variable-list—which may only contain string variables.

LIST

LIST £chan:line-description TO line-description
LIST £chan:line-description—line-description
LIST block-name

Lists all or part of the program. Can be stopped by 'stop' key or paused by 'hold' key. Only executed in immediate mode. The default channel is £Ø.

LIST 3ØØ
LIST 3ØØ TO 4ØØ
LIST FIRST TO 9ØØ, 1ØØØ, 2ØØØ TO LAST
LIST TO 5ØØ, 7ØØ TO
LIST MY_FUNCTION
LIST LAST

TO may be replaced with '-'. Compare **DELETE**.

e.g. *LIST FIRST-1ØØ,5ØØ-LAST*
for *LIST FIRST TO 1ØØ, 5ØØ TO LAST*

LLIST

LLIST list-expression

Identical to LIST, but defaults to £1Ø4, the printer listing channel.

LOAD

LOAD £chan:filename

Throws away the current program file and loads a new file from the given channel, or, if no channel is specified, from channel 1Ø6 (cassette, or disks if attached). Only executed in immediate mode. Clears variables.

LOOK

LOOK £chan AT x,y:v

Assigns to variable 'v' the palette colour at point (x,y) on the standard graphics page or other page specified by the channel expression. Both the channel expression and the AT part are optional. If the AT part is omitted, the current beam (cursor) position will be used. Note that the use of AT will turn off the beam and move it to (x,y).

LOOP See **DO**.

LPRINT LPRINT print-expression

Identical to PRINT, but defaults to £1Ø4, the printer listing channel.

MERGE MERGE £chan:filename

Merges the file from disk, tape or other channel with the current file. Lines from the new program will replace lines of the same number in the current file. Only executed in immediate mode. Clears variables.

NEW Deletes all the current program. Only executed in immediate mode. Clears variables.

NEW ALL Deletes all programs from computer memory, and returns to program Ø.

NEXT See **FOR**.

NUMERIC NUMERIC variable/array-list

Declares numeric variables or arrays (which are local if declared within a DEF function). The default lower bound will be Ø. Compare **DIM**.

NUMERIC I,A(1Ø),B(-1Ø TO 2Ø)

ON ON expr GOTO line-number-list
ON expr GOSUB line-number-list

Evaluates expression, converts result to an integer, and uses integer result N to choose Nth line-number from the list (the count starts from 1). Program execution then resumes from that line. If there is no Nth line-number, no action is taken. Use SELECT or IF block for a more readable program.

ON A +2 GOTO 1ØØ,2ØØ,3ØØ,4ØØ,99,7ØØ

OPEN OPEN £chan:NAME device/filename ACCESS mode
OPEN £chan:device/filename

The access mode is either INPUT or OUTPUT.

ACCESS OUTPUT attempts to create a new file (if on tape or disk); ACCESS INPUT attempts to use an existing file. For devices such as VIDEO:, either can be used. The default is INPUT.

Connects a device, or a file in the case of tape and disk, to a channel. Commands may then read, write or otherwise manipulate data from and to the device (or file) by referring to the channel number.

OPEN £8:"DISK(1):TEST_PROGRAM" ACCESS OUTPUT

Only one device (or file) may be connected to a given channel at any one time, although a single channel may be used to access several devices (files) one after the other.

To disconnect the channel from a device (or file), use the CLOSE command.

Channel numbers range from Ø to 254. (255 is an invalid channel number which is used for special purposes.)

The BASIC system uses several channels as defaults when channels are not specified in statements. These channels are: —

Ø —used for command input and normal text output (e.g. for LIST and PRINT). This channel is connected at reset (or power on) to the device "EDITOR:".

The device "EDITOR:" itself uses the devices "KEYBOARD:" and "VIDEO:", set up in video-mode Ø with page-size 24, 4Ø.

This channel is the default assumed for CAPTURE TO, COPY FROM, and REDIRECT FROM.

Channel Ø is automatically opened at reset, and remains opened until explicitly closed.

Note that channel Ø is specified as the default command channel for ANSI compatibility. Other default channels are numbered over 1ØØ to leave simple channel numbers available for user definition.

1Ø1 —used for graphics input and output statements. This channel is connected at first use of GRAPHICS command to device "VIDEO:",

which is set up in video-mode 1, colour-mode 1, with page-size 20,40. Channel 101 remains open until explicitly closed, e.g. by a TEXT command.

102 —the standard 'text' page. Automatically opened at reset, with page size 24,40.

103 —used for standard sound output. The channel is connected at reset to device "SOUND:".

Channel 103 is automatically opened at reset, and remains opened until explicitly closed.

104 —used for assumed 'hard-copy' operations. This channel is connected at reset to device "PRINTER:". It is the default channel assumed for COPY TO.

Channel 104 is automatically opened at reset, and remains open until explicitly closed.

105 —used for keyboard operations (connected at reset to device "KEYBOARD:"). Remains open until explicitly closed.

106 —used for file-based input and output operations. Whenever required, the channel is connected to "DISK(1):" if attached; If disks are not attached, it is connected to "TAPE:".

Standard file operations include LOAD, MERGE and VERIFY.

Channel 106 is only opened when necessary, and is closed following the completion of every operation—unless an OPEN command has been explicitly given.

107 —used for assumed network operations. This is the default channel assumed for SET CAPTURE FROM.

Channel 107 is only opened automatically by a command which assumes this channel for the default, and is closed following completion of the operation.

108 —used for word processor operations. This is the default channel used for documents being

typed or edited.

Channels 1ØØ-254 remain open unless specifically closed, but channels 1-99 are always closed when RUN is typed, or if any other operation takes place which clears all variables. If BASIC discovers a default channel closed, then it will close all channels (Ø-254) and attempt to re-open its default channels. If it cannot do this, BASIC assumes that an unrecoverable error has occurred and flashes the screen border until the computer is reset.

Device names passed through to the operating system are terminated by a colon so that they can be recognized. Where more than one device is known by the same name, a number enclosed in brackets is appended to the name, e.g. "DISK(2):".

The valid names are:—

"DISK(N):"	Disk drives.
"EDITOR:"	Screen editor. This in turn uses devices "VIDEO:" and "KEYBOARD:".
"KEYBOARD:"	Transparent keyboard. Includes external joysticks.
"NET:"	Built-in local net.
"PRINTER:"	'Centronics-style' printer port.
"SERIAL:"	Serial RS423 I/O.
"SOUND:"	Sound generator.
"TAPE(N):"	Tape drives.
"VIDEO:"	Video pages.

As other devices are attached to the computer, they are likely to define additional names within the operating system.

In most cases, only a device name is required for an OPEN operation. When file-based input/output is used, a filename must be given.

The full specification of a filename is:

"device(n):name"

—"device" is optional; if it is omitted, the system mass-storage default device will be used—for an unexpanded system, this is "TAPE:".

"n" is the device number, and defaults to 1 if omitted; e.g. "DISK:" would go to "DISK(1):".

"name" is the description of the file within a device. It follows the same rules of format as a BASIC identifier, except that only the first 28 characters are significant.

If no colon is included in the filename, it is assumed that the device name has been omitted. So, for example, "SOUND" is a file on "TAPE:", but "SOUND:" is the sound generator device.

The "name" part of a filename is ignored by all currently-defined devices except "TAPE:" and "DISK:". So, for instance, "PRINTER:PRETTY-LISTING" is equivalent to "PRINTER:".

There are some commands which allow you to specify both channels and filenames within the one statement; e.g. LOAD and SAVE.

The full specification in these cases takes the form:

£chan:filename

If £chan is missing, then a default channel is used.

OPTION OPTION ANGLE DEGREES/RADIANS

Selects the base unit for subsequent operations using angles. The default is radians.

OUT OUT n,a

Writes byte 'a' to the I/O port 'n'.

PING Produces 'ping' sound.

PLOT PLOT £chan:point-list
PLOT £chan:ANGLE expr
PLOT £chan:FORWARD/BACK expr
PLOT £chan:LEFT/RIGHT expr

PLOT £chan:ELLIPSE expr, expr
PLOT £chan:PAINT

PLOT followed by a point-list plots points and/or lines. When a PLOT command ends in a semicolon, the beam will be left 'on' after the command has been executed, otherwise it will be turned 'off'.
Thus:—

PLOT x, y

will move the beam—drawing a line, if the beam was 'on'—to position (x,y), and then turn the beam off.

PLOT x,y;

will leave the beam 'on'.

The last two statements both plot a point at (x,y). If the co-ordinate pair is followed by a comma, the beam is moved to the specified position without plotting a point there (and is left 'off').

PLOT x1,y1;x2,y2;...

will draw lines with the beam 'on' between the specified points, and leave it on if the command ends in a semicolon. If the beam was 'on' before the command is executed, a line will also be drawn from the previous beam-position to the point (x1,y1).

Plotting is done in the current ink colour and according to the current line style and line mode (see the Video Options section).

The co-ordinates used in PLOT statements follow the conventions for GRAPHICS plotting. The bottom left-hand corner of the video page is (Ø,Ø). In the co-ordinate specification (x,y), x is the horizontal position counting from the left, and y is the vertical position counting from the bottom.

ELLIPSE plots an ellipse with its centre at the current beam position. The two parameters that follow give the horizontal and vertical distances from centre to circumference, in graphic screen positions. The ellipse must be plotted with the beam 'off' if a dot is not to appear in the centre.

e.g. *PLOT 300,350,*
PLOT ELLIPSE 200,300,

will avoid plotting the dot.

PAINT fills an enclosed area (that contains the current beam position) with the current ink colour. The area painted is bounded by a continuous line differing in colour from the original colour of the beam position.

If the beam is in a position where a point, of the current ink colour, has been plotted, then PAINT will have no effect, as it will detect a boundary condition immediately. As with ELLIPSE, precautions should be taken to avoid plotting a point.

e.g. *PLOT 400, 300, PAINT*

A PLOT command will by default go to channel 101.

POKE

POKE address, value

Sets the value of the specified Z80 memory location.

PRINT

PRINT £chan, AT row-expr, column-expr:output-list
PRINT £chan, USING line-number:output-list
PRINT £chan, USING string:output-list

An item in the output-list can be either an expression or the word TAB followed by a column-number in brackets. Items may be separated by commas or semicolons. A semicolon generates a null string; a comma inserts spaces up to the start of the next print zone. TAB inserts spaces up to the specified column. An output list ending with a comma or semicolon does not generate an end-of-line sequence. Can be executed in immediate mode.

The AT option positions the cursor at the specified row and columm before printing the list. The optional channel number redirects the output (default channel is the standard text page).

The row and column co-ordinates for the AT specification follow the conventions for text positioning. The top left-hand corner of the video page has text co-ordinates (1,1). The fifteenth column in the second line has text co-ordinates (2,15).

PRINT "VALUE = ";A
PRINT AT x,y:"o";

The USING option controls the format of the output. The line-number must be the number of an IMAGE statement. See IMAGE for the details of the format specification.

PROGRAM

PROGRAM name (variable-list)

Defines the name of the current program, for use in CHAIN statements. The program name must be in the standard form required for an identifier (see 'Rules of Basic').

PROGRAM "My Program" (A,B$)

The variable-list (if included) allows the specified parameters to be passed by value from another program. See **CHAIN** and **EDIT**.

RANDOMIZE

Normally each run of a program starts with the same random number sequence. RANDOMIZE changes the random numbers to a fresh sequence.

READ

READ variable-list
READ IF MISSING line-number:variable-list
READ IF MISSING EXIT DO: variable-list

Reads data from the DATA statements; the IF MISSING action is executed on an attempt to read past the end of the data.

READ A,B$(i)

REDIRECT

REDIRECT FROM £chan TO £chan

Reads input from the first channel and directs it to the second, until the end of a file is reached, the 'stop' key is pressed, or there is an error from one of the channels. The redirection can also be halted by use of the invalid channel number £255 as the FROM channel in a later REDIRECT statement.

REM

REM comment-line

Remark line.

REM must be at beginning of line. For greater flexibility, '!' is recommended.

RENUMBER

RENUMBER line-description TO line-description AT expr STEP expr

Renumbers all or a part of the program. Only executed in immediate mode.

RENUMBER FIRST TO 10Ø
RENUMBER 1Ø TO 1ØØ AT 3ØØ STEP 1Ø
RENUMBER STEP 1ØØ

STEP and AT can be in either order or omitted. If STEP is unspecified, the default is 1Ø. If AT is omitted, then the first line-number in the segment to be renumbered is used. If no line-number range is given, then the whole program is renumbered and the default for AT is 1ØØ. For the syntax of the line-descriptions, compare **DELETE**.

All references in the program block to renumbered lines are changed.

RENUMBER cannot change the order of lines in a program. So if the renumbered lines would overlay or surround lines not renumbered, or would be put into a new place in the sequence, or would create too high a line-number—then the RENUMBER command is not executed, and the text of the program is left unchanged.

RESTORE

RESTORE
RESTORE line-number

Resets the start of DATA (for READ statements) to the start of the program or the given line-number.

RETRY

Used as an exit from an exception handler, this returns control to the line or statement which caused the exception. Compare **CONTINUE**.

If an exception handler is used to trap the 'stop' key, then RETRY should be used to continue the program.

RETURN

Returns from a subroutine called by GOSUB.

RUN

RUN
RUN line-number
RUN £chan:file-name

RUN on its own runs the current program from the first line. If a line-number is given, then execution starts from that line-number. If a filename is given (with optional channel), the program is loaded and then run. Only executed in immediate mode. Clears variables.

SAVE

SAVE £chan:filename
SAVE PAGE

Saves the current program. By default it is saved via channel 1Ø6.

SET

Sets current machine-option values. See 'Machine Options', 'Video Options' and 'Sound Options'. Compare **ASK** and **TOGGLE**.

SELECT

select-line
case-line
  any number of statements or blocks
case-line option
  any number of statements or blocks
end-select-line

select-line:
SELECT CASE expression

case-line:
CASE expression
CASE expression TO expression
CASE IS relop expression
CASE ELSE

end-select-line:
END SELECT

The SELECT block is a group of statements to test the variable or expression against a number of alternative conditions.

The word CASE in the SELECT line is optional unless the expression begins with an identifier CASE.

e.g. *SELECT CASE CASE+23*

There can be any number of CASE lines. The cases are tested in order of line-numbers. There is no point in having additional case-lines after a CASE ELSE, since they cannot normally be reached. Several cases can be combined on one line by separating them with commas.

e.g. *CASE 1,2,3 TO 6, 99*

```
SELECT CASE 1
CASE 1
  PRINT "first case"
CASE 2 TO 9,11,21
  PRINT "some more cases"
CASE IS< =A +20
  PRINT "even more cases"
CASE ELSE
  PRINT "rest of cases"
END SELECT
```

The CASE ELSE line can only be used once, and must follow all the other CASE lines. The other CASE lines can be used in any order as necessary, the lines in between two CASE lines forming a block. Control cannot be transferred from outside to inside of a SELECT block.

String SELECTs are also available.

SOUND

SOUND £chan:PITCH expr, DURATION expr, LEFT expr, RIGHT expr, SOURCE expr, STYLE expr, ENVELOPE expr, SYNC expr, INTERRUPT

Provides overall control of a sound. The parameters may be listed in any order.

The number specified by PITCH may be anything from Ø to 127, although good results are normally obtained only in the range Ø-83. Within that range, an increase of 1 will raise the pitch by one semitone; pitch value 37 (the default) is equivalent to middle C.

DURATION gives the duration of the sound (allocated to the non-release phases of the envelope), in 'ticks' (one tick is 1/5Ø second). The default is 5Ø ticks.

The LEFT and RIGHT parameters specify the overall volume of the sound for the two stereo output

channels. The values range from Ø (no sound) to 255 (maximum volume of the machine—the default). If stereo equipment is not being used, the volume will be determined by the sum of the values given for the left and right channels.

SOURCE specifies the tone generator used; the values are Ø-3 (default:Ø). Tone generator 3 is the 'noise generator' (which ignores pitch values).

The STYLE parameter is in the range Ø-255 (default: Ø); for its effects, see the 'Sound Options' section.

ENVELOPE specifies the number of the envelope to be applied to the sound. See the **ENVELOPE** statement 255 (the default) is a built-in envelope.

SYNC allows the start of the sound to be precisely synchronized with 1, 2 or 3 other sounds from different 'sources'. If, for example, three sounds are to start together, each one can be given the instruction SYNC 2, causing it to be synchronized with the two others. (Default value is Ø).

INTERRUPT, if included, causes the new sound to replace any sound (from the same source) which may currently be going.

SPOKE

SPOKE segment, address, value

As POKE, but writes the value to the system address within the specified segment.

START

If no program is currently loaded, this command loads and runs the first file on channel 1Ø6. If a program is loaded, then START acts as RUN.

STOP

Halts execution (prints STOP message).

Note that CONTINUE is allowed after a STOP instruction.

STRING

STRING variable/array-list*n

Declares string variables or arrays with maximum length. Default length is 132. Adding *n after the word STRING or the variable declaration sets the length to n. The default lower bound for an array is Ø.

*STRING*8 LAST_NAME$*2Ø,FIRST_NAME$, MIDDLE_NAME$*

In this example, LAST_NAME$ is given a maximum length of 20; FIRST_NAME$ and MIDDLE_NAME$ are up to 8 characters long.

STRING NAME$

Here, NAME$ has a maximum length of 132.

*STRING NAME$ (4 TO 99)*10*

This array has 96 elements, each of 10 characters.

Note: a DIM statement cannot be used to define the length of a string variable.

TEXT

TEXT
TEXT 40
TEXT 80

Opens a text page covering the entire display except for the area of the status line. Closes the standard graphics page if it was open.

40 or 80 specifies the number of columns on the screen. If this is not specified, then the previous value will be used.

THEN

See **IF**.

TOGGLE

Acts on machine options that have only two possible values (e.g. 'on' and 'off'), by switching from the current value to the alternative. See 'Machine Options', 'Video Options' and 'Sound Options' sections; compare SET and ASK.

TRACE

TRACE ON TO £chan
TRACE OFF

After TRACE ON, the number of the line currently being executed is reported. The output is directed to the video display unless redirected to a specific channel number.

VERIFY

VERIFY £chan:filename

Verifies that a program has been saved correctly; compares the current program file with the specified file, and gives an error message if the two files are not

identical. Channel 1Ø6 is used by default. Only executed in immediate mode.

WHEN

WHEN EXCEPTION USE handler-name
  statements
END WHEN

Specifies the exception handler to be used when an exception caused by program execution occurs inside the WHEN block.

The program statements can include additional nested WHEN blocks.

See **HANDLER**.

Certain system variables and machine functions can be controlled directly from BASIC; these are called machine options. To assign a value to an option, the command SET is used. Where stated, the options listed below may also be handled in conjunction with ASK or TOGGLE.

EDITOR BUFFER

SET EDITOR BUFFER expr

Defines the size of the editor's buffer, in 256-byte chunks. Can be used with ASK.

EDITOR KEY

SET EDITOR KEY channel-number

Allows the specified channel to be used as the editor's keyboard input. Can be used with ASK.

EDITOR VIDEO

SET EDITOR VIDEO channel-number

Allows the specified channel to be used as the text page for the editor. Can be used with ASK.

FKEY

SET £chan:FKEY key-number string

Sets the function key to produce the specified string each time it is pressed (a null string will cause an exception). The default channel is 1Ø5.

The function keys are numbered 1-16. Numbers 1-8 are the unshifted function keys; numbers 9-16 are the shifted equivalents of keys 1-8.

The function keys are set up with default strings by the system, and re-definition of the keys will remove the default settings.

To create automatic 'enter', use &CHR$(13).

INTERRUPT

ASK INTERRUPT CODE

Asks the software interrupt code for the last interrupt.

SET INTERRUPT KEY ON/OFF

When 'on', causes a software interrupt from any key-press. Can be used with TOGGLE.

SET INTERRUPT NET ON/OFF

Turns on or off the software interrupt caused by receiving data from the network.

SET INTERRUPT STOP ON/OFF

Turns on or off the software interrupt from the 'stop' key. Can be used with TOGGLE.

KEY CLICK

SET KEY CLICK ON/OFF

Determines whether a click is heard with each key-press. Can be used with TOGGLE.

KEY DELAY

SET KEY DELAY expr

Sets the initial keyboard delay before auto-repeat starts, in units of 1/50 second. Can be used with ASK.

KEY RATE

SET KEY RATE expr

Specifies the keyboard auto-repeat rate, in units of 1/50 second. Can be used with ASK.

SERIAL BAUD

SET SERIAL BAUD expr

The parameter (in the range 0-15) determines the baud rate for the RS232 port, according to the code given below. Can be used with ASK.

```
0 = > 50    baud      6 = > 300   baud
1 = > 75     "        7 = > 600    "
2 = > 110    "        8 = > 1200   "
3 = > 134.5  "        9 = > 1800   "
4 = > 150    "       10 = > 2400   "
5 = > 200    "
```

```
11 = > 3600   baud
12 = > 4800    "
13 = > 7200    "
14 = > 9600    "
15 = > 19200   "
```

SERIAL FORMAT

SET SERIAL FORMAT expr

Defines the word format for the serial device driver. The format is controlled by the binary bits in the

number, as follows:

BIT	VALUE	EFFECT	
Ø	Ø	8 bits	
	1	7 bits	
1	Ø	no parity	
2	Ø	even parity	ignored if bit 1 is Ø
	1	odd parity	
3	Ø	two stop bits	
	1	one stop bit	

Bits 4 and upwards must be Ø.

STATUS SET STATUS ON/OFF

Turns the 'status line' (at the top of the display) on or off. Can be used with TOGGLE.

REM1 SET REM1 ON/OFF

Controls remote control switch 1. (Also controlled by tape operations.)

REM2 SET REM2 ON/OFF

As above, but for remote control switch 2.

TAPE SOUND SET TAPE SOUND ON/OFF

Controls transmission of sound from tape input to sound output. Allows direct throughput of music or speech from the tape onto the internal speaker or hi-fi output. Can be used with TOGGLE.

These work on the built-in video device, which can contain many video pages each with different parameters. The commands which work on individual pages can be given a channel specification, but if this is left out, some of them default to the standard text page (£1Ø2), others to the standard graphics page (£1Ø1)—as detailed below.

Note that COLOR is always acceptable in place of COLOUR.

BEAM SET £chan:BEAM ON/OFF

The current graphics plotting position is called the 'beam' position. Whenever the beam is moved, it may or may not leave a line behind it, depending on whether it is 'on' or 'off'.

BIAS SET £chan:BIAS colour-number

Establishes which group of colours will figure as numbers 8-15 within the palette. The number specified in the command is the standard code-number of any colour within the desired group; there are 32 effective values. The bias may also be specified using the COLOUR function.

SET BIAS COLOUR (Ø,.6,.4)

The channel number defaults to £1Ø1. The bias is, however, applied to every palette used on the display.

BORDER SET £chan:BORDER colour-number

Changes the border to the colour corresponding to the specified standard code-number. Channel number defaults to £1Ø1.

CHARACTER SET £chan:CHARACTER n,a,b,c,d,e,f,g,h,i

Defines the pattern of the character with ASCII code 'n'. Each of the parameters a-i defines one row of the pattern, starting from the top.

To assist in creating characters, the BIN function can be used to specify each pixel in a row as a Ø or 1.

Although a channel number is specified, the

command will affect all video pages. The channel number defaults to £1Ø2.

CURSOR

SET £chan:CURSOR CHARACTER code
SET £chan:CURSOR COLOUR palette-number

Specifies the ASCII code of the character, and/or the palette-number of the colour, to be used for the cursor. Channel number defaults to £1Ø2.

INK

SET £chan:INK colour-number

Sets the current plotting colour. The colour number is a palette-number except in colour mode 3 (256 colours), when it is a standard code number. The channel number defaults to £1Ø1.

LINE STYLE

SET £chan:LINE STYLE parameter

The current line-style may be set to any value in the range 1-14, enabling various types of broken line to be plotted. Channel number defaults to £1Ø1.

LINE MODE

SET £chan:LINE MODE parameter

Determines the interaction between the colours on the existing display and the new lines which are plotted. In mode Ø (the default mode), a new line overwrites anything plotted before. In modes 1-3, the colour used for any part of the new line will be determined by combining the palette numbers of the old and new ink-colours, in the following ways:

mode 1—'or'
mode 2—'and'
mode 3—'exclusive or'

Modes 4-7 are analogous to modes Ø-3, but are used on the 'attribute' graphics page for plotting in 'paper' colour instead of 'ink' colour.

Channel number defaults to £1Ø1.

PALETTE

SET £chan:PALETTE a,b,c,d,e,f,g,h

Sets the values of the first 8 colours in the palette, which are then used by video options such as SET

PAPER and SET INK. Channel number defaults to £1Ø1.

Only the first four colours can be used in colour-mode 1, and a graphics page in colour-mode Ø can only use the first two. If only the first 2 or 4 colours are specified, the remainder default to colour Ø.

The colours to be placed in the palette are specified by standard code-numbers in the range Ø-255, or by the COLOUR function (see 'Built-in Functions and Variables'). The 'Teletext primary' colours can be specified by name (e.g. MAGENTA).

The palette contains 16 colours in all, although only the first 8 can be chosen entirely freely. See the BIAS option for details on the remaining 8 colours.

PAPER

SET PAPER £chan:colour-number

Selects the colour which will be used as a background for printing or plotting. In colour-mode 3, the paper colour is defined by a standard code-number; in other modes, by a palette-number. The channel number defaults to £1Ø1.

For a graphics video page (mode 1—see **VIDEO MODE** option), the PAPER command will only take effect when the page is cleared—when a new background is selected for the graphics display.

For an 8Ø-column text page (video mode 2), the valid paper colours are palette numbers Ø, 2, 4 and 6. These are paired with ink colours 1, 3, 5 and 7 respectively; a character printed in a specific ink colour will automatically be given the associated paper colour for its own individual background. A colour-pair for ink and paper is selected by typing SET PAPER or SET INK, followed by either of the two relevant palette-numbers.

A 4Ø-column text page (video mode Ø) is similar except that there are only 2 available colour-pairs.

SCROLL

SET £chan:SCROLL ON/OFF

Turns automatic scroll on or off. Channel number defaults to £1Ø2.

SET £chan:SCROLL UP/DOWN n,m

Scrolls the screen up or down from line (n-32) to (m-32).

Channel number defaults to £1Ø2.

VIDEO COLOUR SET VIDEO COLOUR expr

Sets the colour-mode for video pages that are subsequently to be opened. (Channel number is ignored.)

When defining a text video page, colour mode Ø must always be selected. For high-resolution graphics pages, the colour modes have the following significance:—

mode Ø — 2 colours; horizontal resolution 64Ø
mode 1 — 4 colours; horizontal resolution 32Ø
mode 2 — 16 colours; horizontal resolution 16Ø
mode 3 — 256 colours; horizontal resolution 8Ø

On a LORES graphics page (using half as much memory as HIRES), the colour quantity for each mode is as above, but the horizontal resolution is halved.

VIDEO MODE SET VIDEO MODE expr

Sets the video mode for pages that are subsequently to be opened. (Channel number is ignored.)

Parameter values are as follows:—

mode Ø — 4Ø-column text page (2 colour-pairs)
mode 1 — high resolution graphics page
mode 2 — 8Ø-column text page (4 colour-pairs)
mode 5 — low resolution graphics page
mode 15 — 'attribute' graphics screen

VIDEO X SET VIDEO X expr

Defines the horizontal size of video pages subsequently to be opened. (Channel number ignored.) The size is specified as a number of character positions in the range 2-42, using the co-ordinate conventions for text pages.

VIDEO Y SET VIDEO Y expr

As above, only defines the vertical size of the page as a number of character-rows in the range 1-255.

These work on the built-in sound generator.

SOUND BUFFER

SET SOUND BUFFER expr

Sets the size of the sound envelope storage area, for a subsequent open to the "SOUND:" device. The expression is the number of phases. Possible values are 1-255; the default is 20. Can be used with ASK.

SOUND STYLE

The values for the STYLE parameter in a SOUND statement (see 'Commands and Statements' section) have the following effects.

On tone channel 0:—

16 — Low distortion.
32 — Medium distortion.
48 — High distortion.
64 — Use high pass filter. Tone channel 1 is clock.
128 — Ring modulation with channel 2.

On tone channel 1:—

As channel 1, but high pass filter uses tone channel 2; ring modulator uses noise channel (channel 3).

On tone channel 2:—

As channel 1, but high pass filter uses noise channel (channel 3); ring modulator uses tone channel 0.

On tone channel 3 (noise channel):—

1,2,3 — Use tone channel 0, 1 or 2 as clock frequency, instead of the standard 31.25 KHz frequency.

4,8,12 — Select noise frequency from 15, 11 or 9-bit polynomial counters, instead of standard 17-bit counter.

16 — Substitute a 7-bit polynomial counter for the 17-bit counter.

32 — Use low pass filter on noise channel, using tone channel 2 as the clock.

64 — Use high pass filter on noise channel, using tone channel Ø as the clock.

128 — Use ring modulator with tone channel 1.

To select a combination of sound style options, add together the values for the individual options and specify the resulting number as the STYLE parameter.

SPEAKER SET SPEAKER ON/OFF

Controls sound output from the internal speaker; SET SPEAKER OFF is used for silencing the machine quickly.

Trigonometric functions work in degrees or radians (see OPTION statement). Minimal BASIC functions are ABS, ATN, COS, EXP, INT, LOG, RND, SGN, SIN, SQR, TAB and TAN

ABS(X) The absolute value of a number. This just means removing the sign from it. So *ABS(−9)* would be 9.

ACOS(X) The angle associated with cosine X, i.e. the opposite of COS. Thus, *ACOS(COS(X))* is X.

ANGLE(X,Y) The angle between the positive x-axis and the line joining point (Ø,Ø) to point (X,Y).

ASIN(X) The angle of which X is the sine.

ATN(X) The angle of which X is the tangent.

BIN(X) Returns the number corresponding to the given binary representation, e.g. *BIN(11ØØ1)* is 25.

BLACK The colour black, equivalent to COLOUR (Ø,Ø,Ø).

BLUE The colour blue, equivalent to COLOUR (Ø,Ø,1).

CEIL(X) Gives the smallest whole number not less than X. In other words, X is 'rounded up' to the nearest whole number. *CEIL(3.45)* would be 4, and *CEIL(−3.45)* would be −3.

CHR$(X) Returns the character of which X is the ASCII code-number.

COLOUR(R,G,B) Returns the machine-dependent colour number equivalent to the specified mixture of red, green and blue colours. R specifies the proportion of red (Ø to 1), G specifies green (Ø to 1), and B specifies blue (Ø to 1).

e.g. *SET INK COLOUR (1/2,1/3,1/4)*

COLOR(R,G,B) Identical with COLOUR (R,G,B).

COS(X) The cosine of X.

COSH(X) The hyperbolic cosine of X.

COT(X) The cotangent of X.

CSC(X) The cosecant of X.

CYAN The colour cyan, equivalent to COLOUR (Ø,1,1).

DEG(X) Converts X from radians to degrees. DEG(X)=X*18Ø/PI.

EPS(X) The smallest quantity that can be added to or subtracted from X to make the computer register a change in the value of X.

EXLINE Returns the number of the last statement that caused an exception.

EXP(X) Returns the value of e raised to the power of X. The number known as 'e' (2.71828...) is the base for natural logarithms.

EXTYPE Returns the category-number of the last exception.

FP(X) FP stands for fractional part. *FP(1.23)* would be Ø.23, and *FP(−1.23)* would be −Ø.23. FP is the opposite of IP.

FREE The amount of memory free inside the computer. This will depend on how much program and/or data the computer is 'remembering' at any one time. This function provides a useful way of checking on very long programs as you write them. FREE can be used in immediate mode or within a program. PRINT FREE will display the amount of memory (in bytes—see the glossary) on the screen.

GREEN The colour green, equivalent to COLOUR (Ø,1,Ø).

HEX$(X$) Returns a string of bytes given the hex values of the bytes in X$. The hex bytes are in upper or lower case and separated by commas, e.g. *HEX$("21,E3,ff")*

IN(N) Reads a byte from I/O port N.

INF The largest positive number the Enterprise can handle—its idea of infinity. This number is

9.999999999*1Ø^62.

INKEY$ Returns the character from the keyboard if a key is pressed; otherwise returns a null string ("").

INT(X) The largest whole number not bigger than X. So *INT(3.4)* would be 3, and *INT(−3.4)* would be −4.

IP(X) The integer part of X. This means that all figures following the decimal point are chopped off. *IP(9.9)* would be 9, and *IP(−9.9)* would be −9.

LEN(A$) The number of characters (length) of A$.

LCASE$(A$) Converts all upper case alphabetic characters (capitals) to lower case (small letters).

LBOUND(A) Lower bound of the dimension of a one-dimensional array A.

LBOUND(A,N) Lower bound of dimension N of an array A.

LOG(X) The natural logarithm (logarithm to base e) of number X.

LOG1Ø(X) The logarithm of X to base 1Ø.

LOG2(X) Logarithm of X to base 2.

LTRIM$(A$) Removes all spaces which are at the beginning of the string A$. So *LTRIM$(" Hello")* would be "Hello".

MAGENTA The colour magenta, equivalent to COLOUR (1,Ø,1).

MAX(X,Y) Returns the bigger number of X and Y. So *MAX(6,99)* is 99.

MAXLEN(A$) Gives the maximum length that was specified for a string variable or array.

MIN(X,Y) As MAX(X,Y), but returns the smaller number.

MOD(X,Y) X modulo Y. Or, in simpler terms, the remainder of X divided by Y.

ORD(A$) Gives the ASCII code for the character in quotes, or

the ASCII code of the first character of a string variable. ORD stands for ordinal, and means the number associated with the character, in the character-set used by the computer. Since the Enterprise uses ASCII, the ASCII value is returned.

PEEK(N) Returns the byte at Z80 address N.

POS(X$,Y$) Gives the position in X$ (counting the characters from left to right) where Y$ first occurs. If Y$ cannot be found in X$, the result is Ø. By adding a number after the second string (i.e. POS(X$,Y$,X)), you can tell the machine to begin looking for Y$ from a specific place in X$. If X$ is "LONDON" AND Y$ is "ON", then POS(X$,Y$) is 2. But POS(X$,Y$,4) would tell the computer to start from the "D" in "LONDON" when looking for "ON", and would give the result 5.

POS(A$,B$,M) Alternative version of POS. See above.

RAD(X) Converts X from degrees to radians. RAD(X)=X*PI/18Ø.

RED The colour red, equivalent to COLOUR (1,Ø,Ø).

RND Generates a random number between Ø and 1. For practical use, random numbers are multiplied and made into bigger numbers. *INT(RND*1ØØ)* would give a whole (integer) random number between Ø and 99 inclusive. (RND is never 1.)

ROUND(X,N) Rounds X to N decimal places. *ROUND(1.7668,2)* would be 1.77. *ROUND(−1.7668,2)* would be −1.76.

RTRIM$(A$) Cuts off spaces from the end of the string. As LTRIM$, but removes spaces from the right.

SEC(X) The secant of X.

SGN(X) Returns the sign of X. Returns −1 if X is negative, Ø if X is Ø, and 1 if X is bigger than Ø.

SIN(X) The sine of X.

SINH(X) The hyperbolic sine of X.

SIZE(A) The number of elements in the array A.

SIZE(A,N) The number of elements allowed in dimension N of the array.

SPEEK(S,N) As PEEK, but returns the byte at system address N within the segment S.

STR$(X) Converts value X into a string of digits without leading or trailing spaces, but with a '–' sign if X is negative.

SQR(X) The square root of X. X must be positive.

TAB(X) Only allowed in PRINT statements. Moves the cursor position to column X of the current row.

TAN(X) Tangent of X.

TANH(X) Hyperbolic tangent of X.

TRUNCATE (X,N) Cuts N decimal places from X.

UCASE$(A$) Converts all letters in string A$ to upper case (capitals).

UBOUND(A) Upper bound of the dimension of a one-dimensional array A.

UBOUND(A,N) Upper bound of dimension N of an array A.

USR(N,X) Calls an address N (which will probably have been defined using CODE), and passes the integer X in HL to the machine code routine. The value left in HL will be the value returned by USR.

VAL(A$) Converts a string to a number (i.e. the opposite of STR$). VAL starts converting at the first digit in the string, and stops when it gets to the first non-digit character.

WHITE The colour white, equivalent to COLOUR (1,1,1).

WORD$(N) Returns a two-byte string containing the upper and lower bytes of N, which is assumed to be an address. N will usually be an address defined by a CODE statement, and allows backward jumps etc. to be

formed using labels. The first byte of the string will be the LSB.

YELLOW The colour yellow, equivalent to COLOUR (1,1,Ø).

EXOS is short for Enterprise Expandable Operating System. An operating system is a program that attempts to enable the best and easiest possible use to be made of a computer and its facilities. It forms an interface between high-level programs (such as the BASIC language) and the computer.

The main facilities of a computer are its devices and peripherals. These are such things as the screen, the tape interface, a printer and so on. Thus the main part of an operating system handles the devices and peripherals: the input/output system. Other facilities handled by the operating system include the sharing of memory.

INPUT/OUTPUT SYSTEM

The Enterprise microcomputer is extremely complex; to perform even simple functions like printing a string on the screen requires thousands of machine-level instructions, and to print the same string on a printer requires hundreds more instructions. The Enterprise operating system rationalizes the interface between a program and the microcomputer, making it as easy to print a string on a printer as it is to print a string on the screen. This is achieved by allowing programs to treat all input and output devices in an identical fashion. All input and output is performed through 'channels' (a channel simply connects the program to a device). The channels are numbered from Ø to 254. The operating system provides the following functions on channels:

Code number	Function
Ø	System reset
1	Open a channel (connect a device)
2	Create and open a channel
3	Close a channel (disconnect)
4	Close and delete a channel
5	Read a character from a channel
6	Read a block
7	Write a character to a channel
8	Write a block
9	Return the status of the channel
1Ø	Set and read the channel status
11	Perform a special function
16	Read, write or toggle a system variable
17	Capture input from channel to channel

18	Re-direct channel
19	Set default device name
20	Return system status
21	Link device
22	Read system boundary
23	Set user boundary
24	Allocate a segment
25	Free a segment
26	Locate ROMs
27	Allocate channel buffer

These functions are used by BASIC to provide input/output facilities. They are available for all languages to use, and thus provide a uniform method of communicating with devices. They are also available to the machine code programmer, making it very simple to write programs in machine code.

To call the operating system from a machine code program, a single instruction is required, followed by the code for the function. For example, to open a channel, the following code is needed:

Machine code	Assembler code
F7	RST 30H
01	DB 1

The Enterprise operating system provides many more functions than those listed above. A full list of functions and the calling conventions can be found in the Enterprise Technical Manual.

MEMORY USAGE

The operating system is based in Read Only Memory. This means that the program is stored in ROM but still requires RAM space to store its data. The Enterprise is capable of managing a vast amount of RAM and ROM storage.

This storage is manipulated by dividing it up into 'pages' (not to be confused with video pages); each page is 16K bytes long, and there are 256 pages altogether, giving a maximum store capacity of 4M bytes. The Z80 in the Enterprise can only use four of these pages at any one time (rather like reading a book, where you can only see and use two pages at once).

ERROR MESSAGES

Every so often, you are bound to make the odd mistake in a program. It may be difficult to find where the mistake is—or even what it is.

The computer helps you here by providing messages to tell you as much as possible about what's wrong. If you run a program which contains a BASIC error, the computer will stop when it reaches the point at which it can no longer understand the program, and will display a short statement indicating the cause of the problem.

Remember—the computer can't tell you about other kinds of mistake in the same way. If, for instance, you forget about 'operator priority', or think the result of a calculation would be different from what it really is, this may not stop the program from running all the way through—the program will then simply be doing something other than what you thought it should. The computer can only detect errors in the syntax or organization of your BASIC, or problems caused because an action requested by the program is impossible.

The message displayed by the computer when you have typed something it doesn't understand will include a code-number giving the category of your error (these error categories are explained below).

```
*** Not understood.                    Error 2000
```

If the program is already running when a problem arises which makes it impossible to continue, the error message will contain the relevant line-number; for example:

```
*** Mathematical                       Error 3005
300 PRINT SQR(Y)
```

You can now move the cursor up and edit line 300.

Here are the error category numbers. They're grouped into ranges of 1000.

0-999—A number in this range is given by a CAUSE EXCEPTION statement (see page 127).

Overflow errors (1000-). A number or string is too long or too big/small:

1000 — Numeric expression out of range.
1001 — Overflow in evaluating numeric constant.
1002 — Overflow in evaluating numeric expression.
1051 — Overflow in evaluating string expression.
1106 — Overflow in string assignment.

Note that an overflow in the case of a string usually means that the string is too long for use in the given context.

Subscript errors (2000-):

2001 — Subscript out of bounds (not within the range specified by the declaration of an array).

Mathematical errors (3000-):

3001 — Division by zero.
3004 — Attempt to evaluate LOG(X) where X is 0 or a negative number.
3005 — Attempt to evaluate SQR(X) where X is negative.
3007 — Attempt to evaluate ASIN(X) or ACOS(X) where X is in the range −1 to 1.

Parameter errors (4000-):

4000 — Error in evaluating DEF parameters.
4002 — Argument of CHR$(X) out of range.
4003 — Argument of ORD(X) invalid.
4004 — Index of SIZE(X) out of range.
4005 — Invalid TAB index.
4008 — Index of LBOUND(X) out of range.
4009 — Index of UBOUND(X) out of range.
4301 — Error in evaluating CHAIN parameters.

Storage exhausted (5000-):

5000 — The computer's memory is full up.

File errors (7000-):

7000 — Invalid file format (SAVE, LOAD, MERGE, INPUT £chan, etc.), or files do

not match (VERIFY).

7001 — Invalid channel number.

7003 — OPEN statement when channel already open.

7004 — Channel not open.

Note that other file-related problems are covered by device and operating-system errors (9000-). See below.

Input/output errors (8000-):

8001 — READ/INPUT beyond end of data, and IF MISSING action not specified.

8101 — Non-numeric data given for numeric READ or INPUT.

8201 — Invalid format string in PRINT USING.

8203 — Format item in format string too short for output.

Device (and EXOS) errors (9000-):

Editor errors:

9207 — Invalid co-ordinates for positioning cursor.

9208 — Trouble in communicating with keyboard.

9209 — Problems in communicating with video.

Serial errors:

9210 — Invalid baud rate.

Video errors:

9211 — Can't display character on graphics page.

9212 — Line mode too big.

9213 — Line style too big.

9214 — Attempt to move beam off page.

9215 — Not enough rows in page to DISPLAY.

9216 — Invalid parameter to DISPLAY.

9217 — Invalid video mode to OPEN.

9218 — Invalid 'x' or 'y' size to OPEN.

9219 — Invalid colour passed to INK or PAPER.

9220 — Attempt to move cursor off page.

9221 — Invalid row number to scroll.

Sound errors:

9222 — Envelope storage requested is too small (i.e. less than 2).
9223 — Invalid for user-defined envelope number (i.e. 255).
9224 — Not enough room to define envelope.
9225 — Envelope is too big.

Keyboard errors:

9226 — Run out of function key space.
9227 — Invalid function key number.

File-related errors:

9230 — File is too big.
9231 — End of file met in read.
9232 — File already open.
9233 — File already exists.
9234 — File does not exist.

General errors returned by various devices:

9236 — Invalid escape character.
9238 — Attempt to open second channel.

General errors returned by the operating system (EXOS) kernel:

9245 — Insufficient video RAM.
9246 — Insufficient RAM for buffer.
9249 — Channel already exists.
9250 — Device does not exist.
9251 — Invalid channel number.
9252 — Insufficient stack.
9253 — Invalid name string.

Control errors (10000-):

10002 — RETURN without corresponding GOSUB.
10004 — No CASE line in SELECT block has been selected.
10005 — Non-existent program specified in

CHAIN or EDIT.

BASIC syntax errors (20000-):

20000 — Syntax error.
20001 — Invalid line-number.
20002 — Invalid range of line-numbers given.
20003 — Second line-number in range is lower than first.
20004 — Non-existent line-number specified.
20010 — RENUMBER cannot be executed as specified.
20020 — Program cannot be CONTINUEd from this point.
20030 — Identifier (variable name) expected.
20031 — String identifier expected.
20032 — Array identifier expected.
20033 — Function identifier expected.
20034 — Type mismatch.
20040 — Identifier uninitialized (no value has been assigned to the variable).
20041 — Identifier declared twice.
20042 — Identifier too long.
20043 — Closing inverted commas missing from string.
20050 — Missing end of block (LOOP, NEXT, END DEF, END IF, etc.).
20051 — Invalid end of block or EXIT (LOOP/NEXT/EXIT without a corresponding DO/FOR etc.).
20052 — Too many nested blocks.
20060 — Invalid SET, ASK or TOGGLE operation.
20070 — Statement not allowed after THEN or on a multi-statement line.

System errors (30000-):

These should never happen, and mean something is wrong with the computer. It's very unlikely you'll ever see one of these.

Error messages can be trapped, if desired, by using WHEN EXCEPTION and a handler block (page 127). An exception handler can be used to trap any error, even those such as a memory overflow or a syntax

error (a keyword mis-spelled, for example). This must be handled with care, as a RETRY to a permanent error will cause the program to loop indefinitely.

Errors like a division by zero, or a negative SQR argument, can be caught without crashing the program.