

Chapter 2 - BASIC Language Vocabulary

This chapter is a reference to every word in BASIC. The words are listed in alphabetical order. Each entry shows the syntax form, a short description, and an example. Hardware-related keywords give the chapter where the screen, sound, or input system they belong to is described in full.

Do not read this chapter as a lesson. On a first pass, skim the headings so you know what BASIC can say, then continue to Chapter 3 and make the machine draw. Return here when you need the exact spelling of a keyword or the argument order for a command. BASIC is the common door into the same bus machine used by the later graphics, sound, file, and processor chapters.

2.1 Notation

The following conventions appear throughout this chapter and the rest of the book:

Notation	Meaning
UPPERCASE	Type this word exactly.
<i>italics</i>	A piece of variable information you supply.
[...]	Optional. May be omitted.
[...]*	Optional and may be repeated.
a \ b	One of a or b.
<i>expr</i>	Any numeric expression.
<i>str-expr</i>	Any string expression.
<i>var</i>	A numeric variable name.
<i>str-var</i>	A string variable name (ends with \$).
<i>line</i>	A line number (0-4294967295).
<i>addr</i>	An integer expression interpreted as an address.

2.2 Direct-mode commands

These commands work only in direct mode. They are not part of the stored programme language and may not appear inside a stored programme line.

- **DIR** - list the names of files that BASIC can LOAD. See Chapter 35.
- **TYPE** - print a text file from the disk volume. See Chapter 35.
- **RUN AOT** - compile the stored programme to native IE64 code and run it immediately.
- **COMPILE** - write the stored programme as a standalone .ie64 image. See Chapter 35.
- **TRANSPILE** - write the stored programme as IE64 assembly text. See Chapter 35.
- **ASSEMBLE** - assemble IE64 source text into a standalone .ie64 image. See Chapter 35.

2.3 Alphabetical reference

ABS

ABS(*expr*)

Returns the absolute value of *expr*.

```
PRINT ABS(-5.5)
5.5
```

AHX

AHX*args*

Play an AHX music file. See Chapter 18.

AND

*expr*AND*expr*

Bitwise AND of two integers. Each operand is truncated to a signed 32-bit integer before the operation; the result is converted back to a number.

```
PRINT 12 AND 10
8
```

ANTIC

ANTIC*args*

Program the ANTIC display list. See Chapter 7.

ASC

ASC(*str-expr*)

Returns the byte value of the first character of *str-expr*. If the string is empty, returns 0. See Appendix C for the character code table.

```
PRINT ASC("A")
65
```

ASSEMBLE

ASSEMBLE "*filename*"

Direct-mode command. Read the assembly source matching *filename* from the disk volume, assemble it inside the machine as IE64 source at PROG_START, and write *filename*.ie64. The stored BASIC programme is not changed.

ASSEMBLE accepts the IE64 instruction set, including MOV_T, labels, branch and call targets, dc.b, dc.w, dc.l, dc.q, align, and the symbolic constants known to the IE64 assembler. It also accepts the zero-test branch source forms BEQZ, BNEZ, BLTZ, BGEZ, BGTZ, and BLEZ; these assemble as the ordinary IE64 compare-and-branch instructions with the second register set to R0. See Chapter 35 for the file behaviour and Chapter 25 for the IE64 image form.

ATN

ATN(*expr*)

Returns the arctangent of *expr* in radians.

BIN\$

BIN\$(*expr*)

Returns the binary representation of the integer part of *expr* as a string of 0 and 1 digits.

```
PRINT BIN$(10)
1010
```

BITCLR

BITCLR*addr, bit*

Clears bit *bit* (0-7) of the byte at *addr*. *bit* outside that range gives an unspecified result.

BITSET

BITSET*addr, bit*

Sets bit *bit* (0-7) of the byte at *addr*.

BITTST

BITTST(*addr, bit*)

Returns -1 if bit *bit* (0-7) of the byte at *addr* is set, 0 if it is clear.

BLIT

BLIT COPY*src, dst, w, h* [, *src-stride, dst-stride*]

BLIT FILL*dst, w, h, colour* [, *dst-stride*]

BLIT LINE*x0, y0, x1, y1, colour* [, *dst-stride*]

BLIT MEMCOPY*src, dst, len*

BLIT M*src, dst, len*

BLIT MODE7*src, dst, w, h, u0, v0, du-col, dv-col, du-row, dv-row, u-mask, v-mask* [, *src-stride, dst-stride*]

BLIT WAIT

Use the VideoChip blitter. COPY copies pixels, FILL fills a rectangle, LINE draws one line, MEMCOPY copies a byte span, M is the short form of MEMCOPY, MODE7 performs affine texture mapping, and WAIT waits until the blitter is idle. See Chapter 4.

BLOAD

BLOAD*filename, addr*

Load a binary file into memory beginning at *addr*. Both arguments are required; omitting the comma and address loads nothing. The destination is carried through the File I/O block's 32-bit data pointer; an address of 2^{32} or greater reports ? FC ERROR. See Chapter 35.

BOX

`BOXx0,y0,x1,y1[,colour]`

Draw a rectangle. See Chapter 5.

CALL

`CALLaddr`

Call IE64 machine code at *addr*. Returns to BASIC when the called code executes RTS. See Chapter 25 for the calling convention.

CHR\$

`CHR$(expr)`

Returns a one-character string whose single byte equals the low eight bits of `INT(expr)`. `CHR$(n)` and `CHR$(n+256)` produce the same string.

```
PRINT CHR$(65); CHR$(66); CHR$(67)
ABC
```

CIRCLE

`CIRCLEx,y,r[,colour]`

Draw a circle. See Chapter 5.

CLEAR

`CLEAR`

Reset all variables and arrays. Reset the DATA read pointer. Reset the GOSUB and structured-control stack. The program text is preserved.

CLS

`CLS`

Clear the screen. See Chapter 5.

COMPILE

`COMPILE "filename"`

Direct-mode command. Compile the stored BASIC programme into a standalone IE64 image. If *filename* has no `.ie64` suffix, BASIC adds it. If the current programme was loaded from a subdirectory, the compiled image is written beside that loaded programme; otherwise it is written at the root of the disk volume.

COMPILE is a prompt command, not a stored-programme statement. It rejects direct-mode-only commands and other raw prompt forms. When a statement cannot be compiled, BASIC prints `?COMPILE ERROR IN` followed by the line number and the reason. See Chapter 35 for the File I/O side and Chapter 25 for flat IE64 images.

COLOR

COLOR*fg* [, *bg*]

Set the VGA text-mode foreground and optional background colour used by PRINT. See Chapter 5.

CONT

CONT

Resume execution after a STOP. Has no effect if there is no saved state. After RUN A0T, a top-level STOP saves a native IE64 continuation, and CONT re-enters the compiled code. Editing the program, NEW, LOAD, or a fresh RUN or RUN A0T discards that saved continuation.

COPPER

COPPER*args*

Program the copper list. See Chapter 4.

COS

COS (*expr*)

Returns the cosine of *expr*, where *expr* is in radians.

COSTART

COSTART*args*

Start a coprocessor program. See Chapter 32.

COSTOP

COSTOP*args*

Stop a coprocessor program. See Chapter 32.

COWAIT

COWAIT*args*

Wait for a coprocessor to reach a synchronisation point. See Chapter 32.

DATA

DATA*item* [, *item*]*

Declare numeric or string constants for later use with READ. The items are stored as raw text from DATA to the end of the line or to the next colon, and parsed when READ consumes them.

```
10 DATA 1, 2, "HELLO", 3.5
20 READ A, B, C$, D
```

DEC

DEC*var*

Subtract 1 from *var*. Equivalent to `LETvar=var- 1`, but shorter to type.

DEF

`DEF FNname(var) =expr`

Define a user function with one parameter. The function is called later with `FNname(expr)`. Use the exact `DEF FN` form for user functions.

NOTE: `DEF` shares stored token byte \$97 with `TROFF`. Type `DEF FN` exactly. If you `LIST` a stored function definition, the listing may show `TROFF` in place of `DEF`; the line still behaves as `DEF FN`. Appendix A records the token map.

DIM

`DIMname(size) [,name(size[,size])]`

Declare a numeric array. *size* is the highest legal subscript; the array has *size* + 1 elements per dimension, indexed from 0. Only numeric arrays are supported. There is no string-array form.

```
DIM A(10), GRID(7,7)
```

DO

`DO`

Begin a structured loop. End the loop with `LOOP UNTILcondition` or `LOOP WHILEcondition`. See **LOOP**.

ELSE

`IFconditionTHENthen-partELSEelse-part`

Mark the alternative branch of an `IF` statement. See **IF**.

END

`END`

Stop the running program and return to direct mode. There is no saved state for `CONT`.

ENVELOPE

`ENVELOPEargs`

Set a volume envelope. See Chapter 11.

EOR

`exprEORexpr`

Bitwise exclusive-OR of two integers.

```
PRINT 12 EOR 10  
6
```

EXP

EXP(*expr*)

Returns *e* raised to the power *expr*.

FN

FN*name*(*expr*)

Call a user function defined with DEF FN.

FOR

FOR*var*=*start*TO*end*[STEP*step*]

Begin a numeric loop. *var* takes the value *start* on first entry. After each pass through the loop body, the matching NEXT adds *step* (default 1) to *var* and compares it with *end*. The loop ends when *var* passes *end*.

```
10 FOR I = 1 TO 10
20   PRINT I,
30 NEXT
40 PRINT
```

FRE

FRE(*expr*)

Returns the approximate number of free bytes in the internal BASIC arena. The argument is ignored; use FRE(0).

GATE

GATE*args*

Open or close a sound-channel gate. See Chapter 11.

GET

GET*var* | GET*str-var*

Read one character from the keyboard without waiting for RETURN. *var* receives the byte value; *str-var* receives the character as a one-byte string. If no key is available, *var* is 0 and *str-var* is the empty string.

GOSUB

GOSUB*line*

Call the subroutine starting at *line*. Execution resumes at the statement after the matching RETURN. Subroutines may call other subroutines.

GOTO

GOTO*line*

Branch to *line*. Execution continues at the first statement of that line.

GTIA

GTIAargs

Program the GTIA. See Chapter 7.

HEX\$

HEX\$(*expr*)

Returns the hexadecimal representation of INT(*expr*) as a string of uppercase hex digits.

```
PRINT HEX$(255)
FF
```

HOST

HOST*subverb*

Issue a HOST command. *subverb* is one of NET, UPDATE, REBOOT, POWEROFF, HELP. See Chapter 36.

IF

IF*condition*THEN*then-part*[ELSE*else-part*]

If *condition* is non-zero, execute *then-part*; otherwise execute *else-part* if present, or fall through. *then-part* may be a line number (equivalent to GOToline) or a statement list.

```
10 INPUT N
20 IF N < 0 THEN PRINT "MINUS" ELSE PRINT "PLUS OR ZERO"
```

INC

INC*var*

Add 1 to *var*.

INPUT

INPUT [*prompt*;]*var* [, *var*]*

Print *prompt* (a quoted string) if present, then ?, then read values typed on the terminal. Values are separated by commas.

```
10 INPUT "WHAT IS YOUR NAME"; N$
20 PRINT "HELLO, "; N$
```

INT

INT(*expr*)

Returns the integer part of *expr*, truncated toward 0.

LCASE\$

LCASE\$(*str-expr*)

Returns *str-expr* with every uppercase letter replaced by its lowercase equivalent. Non-letter bytes are unchanged.

LEFT\$

LEFT\$(*str-expr*, *n*)

Returns the first *n* bytes of *str-expr*. If *n* is negative, raises ILLEGAL QUANTITY.

LEN

LEN(*str-expr*)

Returns the number of bytes in *str-expr*.

LET

[LET] *var*=*expr*

Assign *expr* to *var*. The LET keyword is optional.

LINE

LINE *x0*, *y0*, *x1*, *y1* [, *colour*]

Draw a line. See Chapter 5.

LIST

LIST

Print the whole stored programme. Any arguments after LIST are ignored; BASIC always lists every line.

LOAD

LOAD *filename*

Load a program file. See Chapter 35.

LOCATE

LOCATE *row*, *col*

Move the text cursor. See Chapter 5.

LOG

LOG(*expr*)

Returns the natural logarithm of *expr*. *expr* must be positive.

LOOP

LOOP UNTIL *condition* | LOOP WHILE *condition*

End a DO block. LOOP UNTIL repeats until *condition* is non-zero; LOOP WHILE repeats while *condition* is non-zero.

```
10 N = 0
20 DO
30   N = N + 1
40   PRINT N
50 LOOP UNTIL N >= 5
```

MAX

`MAX(expr, expr)`

Returns the larger of the two arguments.

MEMALLOC

`MEMALLOC(size [, align])`

Allocates a public low-memory buffer and returns its address. Use it when a BASIC programme needs a buffer that hardware, the copper, a coprocessor request, or a DMA-style register block can read.

size is the number of bytes to reserve. *align* is optional; it defaults to 4096. If supplied, *align* must be a power of two from 1 through 1048576. A zero size, zero alignment, fractional value, negative value, or out-of-range value gives an FC error. If the public BASIC allocation ranges are exhausted, BASIC reports an OM error.

```
10 B=MEMALLOC(256,16)
20 POKE32 B,&H12345678
30 PRINT HEX$(PEEK32(B))
```

The final line prints 12345678, showing that the allocated buffer can be used by the ordinary memory helpers.

The address returned by MEMALLOC is an exact integer value. It may be stored in a numeric variable and used later with PEEK, POKE, PEEK32, POKE32, or POKE64.

MIDI

`MIDI NOTEch, note, velocity`

Send a live MIDI note event to the RawlandMini synth. *ch* is masked to 0 through 15, and *note* and *velocity* are masked to 0 through 127. A velocity of 0 turns the note off.

`MIDI PROGch, programme`

Select the live MIDI programme for a channel.

`MIDI CTRLch, controller, value`

Send a live MIDI controller event. Controller 7 changes channel volume, and controller 11 changes expression.

`MIDI SENDbyte [, byte] ...`

Write raw MIDI bytes to the live MIDI stream parser. This form is for running-status byte streams; ordinary BASIC programmes usually use MIDI NOTE, MIDI PROG, and MIDI CTRL instead.

`MIDI RESET`

Turn off live MIDI notes and reset the live stream parser. See Chapter 21.

MID\$

MID\$(*str-expr*, *start* [, *len*])

Returns *len* bytes of *str-expr* starting at position *start* (1-based). If *len* is omitted, returns the substring from *start* to the end.

MIN

MIN(*expr*, *expr*)

Returns the smaller of the two arguments.

MOD

MOD STATUS

Return the MOD player's status byte. Use SOUND MOD PLAY, SOUND MOD STOP, and SOUND MOD FILTER to control playback. See Chapter 19.

NEW

NEW

Erase the program, all variables, and all arrays. After NEW, the program is empty.

NEXT

NEXT [*var*]

Mark the end of a FOR loop. *var* identifies which loop to close; if omitted, the innermost loop is used.

NOT

NOT*expr*

Bitwise complement of INT(*expr*) as a signed 32-bit integer.

ON

ON*expr*GOTO*line1* [, *line2*]*

Branch to *line-N*, where *N* is INT(*expr*). *N* must be between 1 and the number of line numbers listed. If *N* is out of range, execution falls through.

ON*expr*GOSUB*line1* [, *line2*]*

Like the GOTO form but each branch is a subroutine call.

OR

*expr*OR*expr*

Bitwise OR of two integers.

PALETTE

PALETTE*args*

Set entries in the colour palette. See Chapter 3.

PEEK

`PEEK(addr)`

Returns the byte value at *addr*. PEEK is the historical byte alias for PEEK8.

`PEEK8(addr)`

Returns the byte value at *addr*. *addr* may be any byte address.

`PEEK16(addr)`, `PEEK32(addr)`, and `PEEK64(addr)`

Return 16-, 32-, and 64-bit integer values from aligned addresses. PEEK64 preserves the exact qword payload, so `HEX$(PEEK64(addr))` and `POKE64 dst, PEEK64(src)` can round-trip 64-bit pointers and bitfields.

PI

The constant 3.141593. Use `2 * PI` for a full turn, or use `TWOPI`.

PLOT

`PLOTx,y[,colour]`

Set a single pixel. See Chapter 5.

POKE

`POKEaddr,expr``

Write the low byte of *expr* at *addr*. POKE is the historical byte alias for POKE8.

`POKE8addr,expr``

Write the low byte of *expr* at *addr*. *addr* may be any byte address.

`POKE16addr,expr``

Write the low 16 bits of *expr* at *addr*. *addr* must be 2-byte aligned.

`POKE32addr,expr``

Write the low 32 bits of *expr* at *addr*. *addr* must be 4-byte aligned.

`POKE64addr,expr``

Write a 64-bit value at *addr*. *addr* must be 8-byte aligned. A direct integer-compatible literal expression such as `&H1122334455667788` is preserved for the store. Values returned by PEEK64 and MEMALLOC also keep the exact integer payload needed for address and qword round-trips.

POKEY

`POKEYargs`

Program the POKEY sound chip. See Chapter 17.

POS

`POS(expr)`

Returns the current cursor column (0-based). The argument is ignored. Use POS (0).

PRINT

```
PRINT [item[,|; item]*[;]]`
```

Print zero or more items to the terminal. The separators have these effects:

Separator	Effect
;	No padding between items.
,	Insert one TAB byte (CHR\$(9)) between items.
trailing ;	Suppress the closing newline.
trailing ,	Insert one TAB byte; do not suppress the newline.

The single character ? is an alternate spelling of PRINT. LIST prints it as PRINT.

```
PRINT "A"; "B"; "C"  
ABC
```

PSG

PSGargs

Program the PSG sound chip. See Chapter 13.

READ

```
READvar[, var]*
```

Consume successive items from the program's DATA statements and assign them to *var*. Numeric variables get numeric items; string variables get string items. RESTORE resets the read pointer.

REM

REMcomment

Everything from REM to the end of the line is a comment.

RESTORE

```
RESTORE
```

Reset the DATA read pointer to the first DATA item in the program.

RETURN

```
RETURN
```

End a GOSUB. Execution resumes at the statement after the GOSUB that called this subroutine.

RIGHT\$

```
RIGHT$(str-expr, n)
```

Returns the last n bytes of *str-expr*.

RND

RND(*expr*)

Returns a pseudo-random number in the range [0, 1). A negative argument seeds the generator; otherwise the argument is ignored.

RUN

RUN

Run the stored programme from the lowest-numbered line. Numeric line arguments are not parsed, so RUN 100 behaves like RUN and always restarts from the first stored line. RUN preserves variables and arrays; only the DATA pointer and the control stack are reset.

RUN A0T

Direct-mode form. Compile the stored programme to native IE64 code inside the machine and run the compiled code immediately. The visible program result should match RUN, but BASIC first prints:

```
Compiling to native code...
```

RUN A0T restarts from the first stored line, resets the same programme state as RUN, and discards any older compiled STOP continuation. If there is no stored programme to compile, BASIC prints ?NO CODE TO COMPILE and returns to direct mode.

RUN "*filename*" is a direct-mode form for running a saved program image. See Chapter 35.

SADD

SADD(*str-var*)

Returns the address of the byte buffer that holds the current value of *str-var*. Byte writes through this address mutate the string variable. Re-read SADD after assigning a new value to the variable, since the variable may then point at a different byte buffer.

SAP

SAP*args*

Play a SAP music file through the POKEY. See Chapter 17.

SAVE

SAVE*filename*

Save the program to a file. See Chapter 35.

SCREEN

SCREEN*args*

Select a screen mode. See Chapter 5.

SCROLL

SCROLLargs

Scroll the VGA screen. See Chapter 5.

SGN

SGN(expr)

Returns -1, 0, or +1 depending on the sign of *expr*.

SID

SIDargs

Program the SID sound chip. See Chapter 15.

SIN

SIN(expr)

Returns the sine of *expr*, in radians.

SOUND

SOUNDargs

Play a tone on a sound channel. See Chapter 11.

SQR

SQR(expr)

Returns the square root of *expr*. *expr* must not be negative.

STEP

See **FOR**.

STOP

STOP

Stop the program and save its position so that **CONT** can resume. For **RUN AOT**, this is a native continuation inside the compiled IE64 code. A **STOP** reached inside active compiled **GOSUB** nesting is not a resumable subroutine state; **CONT** resumes after the **STOP**, but a following **RETURN** reports **?RETURN WITHOUT GOSUB**.

STR\$

STR\$(expr)

Returns the decimal string representation of *expr*. A leading space is reserved for the sign: a positive number has a leading space; a negative number has a leading -.

SWAP

*SWAP**var, var*

Exchange the values of two variables. Both variables must be of the same type (both numeric or both string).

TAB

TAB(*expr*)

Used only inside a PRINT list. Prints spaces so that the cursor sits at column *expr* (counting from 0). Has no effect if the cursor has already passed *expr*.

TAN

TAN(*expr*)

Returns the tangent of *expr*, in radians.

TED

*TED**args*

Program the TED chip (video and audio). See Chapter 6 and Chapter 16.

TEXTURE

*TEXTURE**args*

Bind a texture for Voodoo rasterisation. See Chapter 9.

THEN

See **IF**.

TO

See **FOR**.

TRIANGLE

*TRIANGLE**args*

Rasterise a Voodoo triangle. Use **TRICOLOR** in the argument list to give one colour for each vertex. See Chapter 9.

TROFF

TROFF

Turn off line-by-line tracing.

NOTE: **TROFF** shares stored token byte \$97 with **DEF**. When the token appears before **FN**, BASIC treats it as a function definition. Appendix A records the token map.

TRON

TRON

Turn on line-by-line tracing. While tracing is on, each line number is printed before its statements run.

TRANSPILE

TRANSPILE "*filename*"

Direct-mode command. Convert the stored BASIC programme to IE64 assembly text and write the matching assembly source file. It does not assemble the text and it does not write a .ie64 image.

The assembly text is the same self-contained source that COMPILE writes beside its image. ASSEMBLE "*filename*" can assemble it later into the same kind of flat IE64 image. See Chapter 35.

TWOPI

The constant 6.283185 (two times pi).

TYPE

TYPE "*filename*"

Direct-mode command. Print a text file from the disk volume to the screen. The name must be quoted. The file must be text; if it contains binary control bytes, BASIC prints ?NOT A TEXT FILE and leaves the screen untouched by the file contents.

TYPE is not a stored-programme statement and cannot be compiled. It is useful at the prompt for reading saved listings, notes, and generated source text. See Chapter 35.

UCASE\$

UCASE\$(*str-expr*)

Returns *str-expr* with every lowercase letter replaced by its uppercase equivalent. Non-letter bytes are unchanged.

ULA

ULAargs

Program the ULA display. See Chapter 8.

UNTIL

See LOOP.

USR

USR(*addr*)

Call IE64 machine code at *addr*. The routine returns an integer value through the result register; USR returns that value as a number. See Chapter 25 for the calling convention.

VAL

VAL(*str-expr*)

Parse *str-expr* as a number. Stops at the first character that is not part of a numeric literal.

```
PRINT VAL ("3.14XYZ")
3.140000
```

VARPTR

VARPTR(*var*)

Returns the address of the storage slot that holds the value of *var*. Numeric scalar variables expose a 16-byte public cell: tag at +0, reserved at +4, and payload at +8. Tag 1 is FP64 and tag 2 is I64. Direct writes through VARPTR must keep the tag and payload consistent.

VERTEX

VERTEX*args*

Submit a Voodoo vertex. See Chapter 9.

VOODOO

VOODOO*args*

Program the Voodoo rasteriser. See Chapter 9.

VSYNC

VSYNC

Wait for vertical retrace. See Chapter 3.

WAIT

WAIT*addr, mask[, xor]*

Read the 32-bit value at *addr* repeatedly until $((\text{value} \text{ EOR } \text{xor}) \text{ AND } \text{mask})$ is non-zero. *xor* defaults to 0. WAIT returns after approximately one million polls if the condition is never met.

```
REM WAIT UNTIL A COOKED KEY BYTE IS QUEUED
WAIT &H000F072C,1
PRINT PEEK32(&H000F0728)
0
```

The polled byte is 0 until a key has been typed; press a key before the example runs to see the cooked code instead.

WEND

WEND

Alternate spelling of UNTIL when terminating a WHILE loop. LIST prints this form as UNTIL.

WHILE

WHILE*condition*

Begin a loop that runs while *condition* is non-zero. The loop body runs until a matching LOOP UNTIL or LOOP WHILE. See LOOP.

XOR

There is no XOR keyword in this BASIC. The exclusive-OR operator is EOR.

ZBUFFER

ZBUFFERargs

Enable or configure the Voodoo Z-buffer. See Chapter 9.

2.4 Operator symbols

Every operator is described at its alphabetical position above. The operator symbols are summarised here for quick reference. See Chapter 1 §1.8 for the precedence table.

Symbol	Keyword name
+	(add)
-	(subtract or unary minus)
*	(multiply)
/	(divide)
^	(power)
<<	LSHIFT
>>	RSHIFT
<	LT
<=	(LT followed by =)
=	EQUAL
<>	(LT followed by >)
>	GT
>=	(GT followed by =)
?	PRINT (alternate spelling)

2.5 Cross-reference index

The hardware-related keywords above point at the chapter that documents the underlying device in full. The list below collects those references:

Keyword	Chapter
SCREEN, CLS, PLOT, LINE, CIRCLE, BOX, LOCATE	5
PALETTE, VSYNC	3
COLOR	5
COPPER, BLIT	4
SCROLL	5

Keyword	Chapter
TED	6 (video), 16 (audio)
ANTIC, GTIA	7
ULA	8
VOODOO, ZBUFFER, VERTEX, TRIANGLE, TEXTURE	9
SOUND, ENVELOPE, GATE	11
SOUND PLAY, SOUND STOP	23
PSG	13
SID	15
POKEY, SAP	17
AHX	18
SOUND MOD, MOD STATUS	19
HOST	36
COSTART, COSTOP, COWAIT	32
CALL, USR	25
LOAD, SAVE, BLOAD, RUN ".ie", RUN AOT, COMPILE, TRANSPILE, ASSEMBLE, TYPE	35