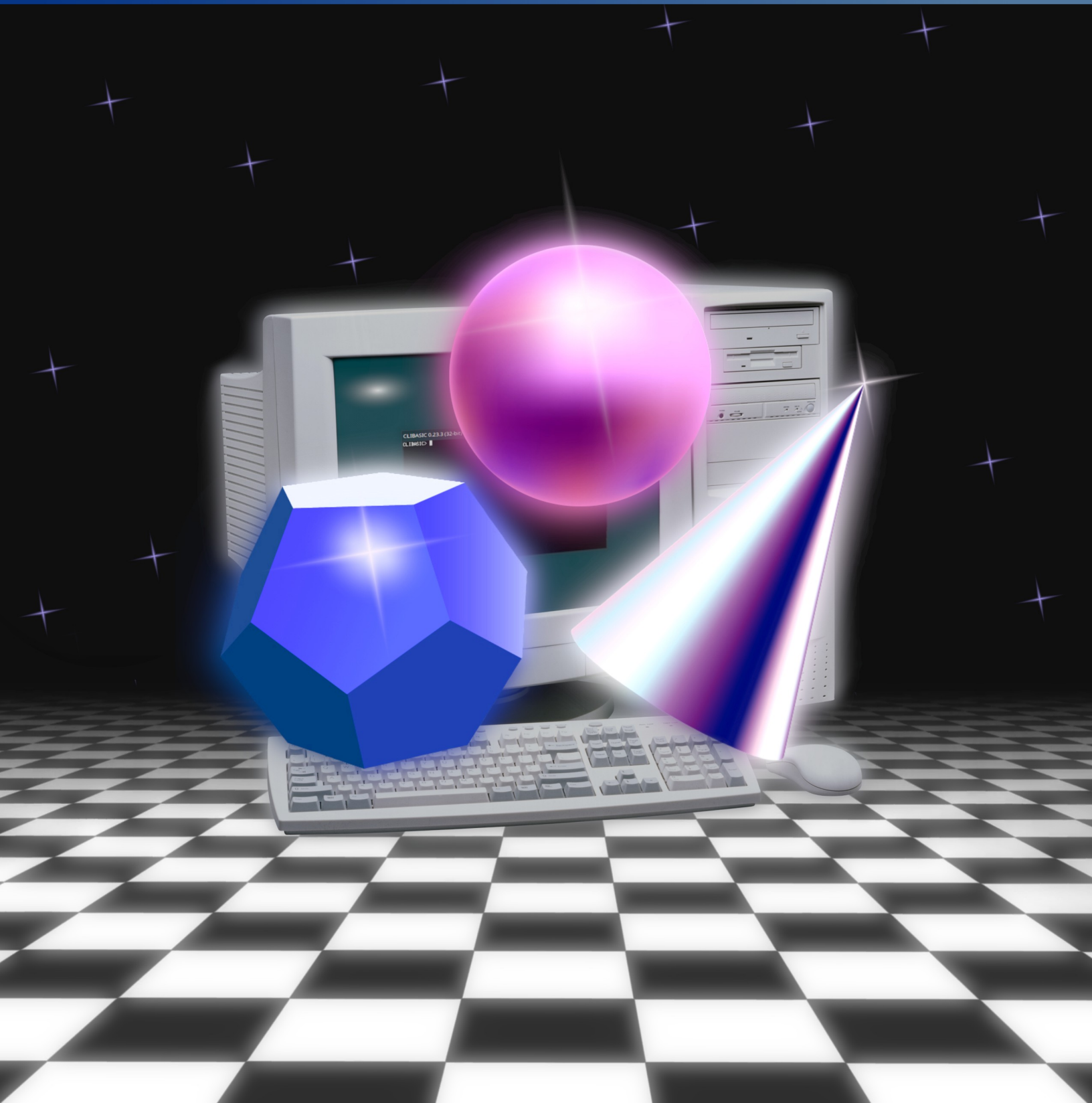


CLIBASIC

Language Manual



Contents:

Manual Symbols	<hr/>	1
Syntax	<hr/>	2
Behavior	<hr/>	3
Commands	<hr/>	4
Functions	<hr/>	9
Logic Commands	<hr/>	16
Symbols	<hr/>	17
Comparing	<hr/>	18
About	<hr/>	19

Manual symbols:

{ } = choose from

[] = optional

| = choice separator

/ = continuous choice separator

... = continue

Syntax:

- Commands are separated by newlines or colons.
- Command, function, and variable names are not case sensitive.
- Variable names are allowed to contain A-Z, 0-9, #, \$, %, !, ?, @, and _.
- Variables must have at least one non numeric valid character to be a valid name.
- Variables can be addressed as an array by appending a [, the element index, and a] to the end of the variable name.
- Function names are allowed to contain the same characters as variable names.
- Functions can be called by adding (, a comma separated list of arguments, and a) to the end of a function name.
- Arguments are separated by commas.
- Adding a comma with nothing after it will count as a argument but the command/function will be aware that the argument is empty.
- Strings must have one " to begin and another " to end.
- Strings can only be added together.
- Numbers can include . And 0-9.
- Logic compares must have blocks of a value, a =, >, <, >=, <=, ==, !=, or <>, and another value. These blocks must be separated by a &, or |.
- A whole number greater or equal to 0 may be prefixed before a command to indicate a line number.

Behavior:

- All file I/O functions and commands set an internal variable which can be queried using the `_FILEERROR()` function.
- Command, function, variable, and label names are case insensitive and are internally converted into upper-case before interpretation.
- For performance purposes, strings are not bounds-checked to be larger than `CB_BUF_SIZE` and counters/pointers are not checked to prevent `int/int32_t` overflow.
- On Windows, due to lack of a better solution, the `EXEC` command and `EXEC()` and `EXEC$()` functions are passed through `system()` and will be interpreted by `CMD` which can be insecure and/or buggy.
- The way `PATH$` is handled by extension commands and functions, depends on the operating system and/or implementation of `dlfcn`. On GNU/Linux, if a path is not specified, `dlopen()` will search in `/usr/lib`.
- Subfunctions are treated like programs and must use `_ARG$()` and `_ARGC()` to retrieve arguments.

Commands:

BELL [COUNT% [, DELAY%]]	Rings the terminal bell COUNT% times or 1 time if COUNT% is not provided with a delay of DELAY% milliseconds or 750 milliseconds if DELAY% is not provided, between the rings.
CALL FILENAME\$ [, {ARG%/ARG\$}...]	Runs FILENAME\$ in the current session and passes ARG%/ARG\$ to it.
CALLA ARRAY\$	Runs ARRAY\$[0] in the current session and passes the remaining elements of ARRAY\$ to it.
CALLSUB SUBFUNC [, {ARG%/ARG\$}...]	Calls the subfunction SUBFUNC.
{CHDIR CD} DIR\$	Changes the current directory to DIR\$.
CLS [COLOR%] [, LINE%]	Clears the screen or specified line with an optional color. If LINE% is not provided then the current line is used.
COLOR {FGC% [, BGC%] [FGC%], BGC%}	Sets the foreground color to FGC% and the background color to BGC%.
DEFRAG	Defragments the internal variable list for faster variable access.
{DEFRAGLABELS DEFRAGLBLs}	Defragments the internal label list for faster label jumping.
DEFRAGSUBS	Defragments the internal subfunction list for faster subfunction calling.
DEL {VAR\$ VAR%}...	Deletes the variable VAR\$ or VAR%. Multiple variables can be specified.
{DELLABEL DELLBL} LABEL	Deletes the label LABEL. Multiple labels can be specified.
DELSUB SUBFUNC	Deletes the subfunction SUBFUNC. Multiple subfunctions can be specified.
DIM VAR, MAX% [, {INIT\$ INIT%}]	Makes an array with the max index being MAX% and the initial value for each element being INIT\$/INIT%. If INIT\$/INIT% is not specified, ""/0 is used.

ENDSUB	Ends a subcfunfunction declaration.
EXEC PROGRAM\$ [, {ARG\$/ARG%}...]	Executes PROGRAM\$ and passes the remaining arguments to PROGRAM\$.
EXECA ARRAY\$	Executes ARRAY\$ where element 0 is the program and the rest of the elements are passed as arguments.
{EXIT QUIT} [CODE%] {EXIT QUIT} [{DATA\$/DATA%}]	Default: Exits with CODE% (or 0 if CODE% is not supplied). In subfunction: Returns DATA\$/DATA% after being called by CALLFUNC().
EXTENSIONS	Prints a list of loaded extentions.
FCLOSE FILENUM%	Closes file number FILENUM%. If -1 is passed then all open files will be closed.
FILES [DIR\$]	Lists the files and directories in the directory specified by DIR\$ or the current directory if DIR\$ is not provided.
FILL ARRAY [, {VAL\$/VAL%}]	Fills ARRAY with VAL\$/VAL%. VAL\$/VAL% is re-evaluated for each element. If VAL\$/VAL% is not specified, ""/0 is used.
FLUSH FILENUM%	Flushes the file number FILENUM%.
FSEEK FILENUM%, POSITION%	Moves the file cursor of file number FILENUM% to POSITION%.
FUNC NAME	Begins a subfunction declaration that will return a number.
FUNC\$ NAME	Begins a subfunction declaration that will return a string.
FWRITE FILENUM%, STRING\$	Writes STRING\$ to file number FILENUM%.
GOSUB NAME	Jumps to a label named NAME and saves the current position.
{GOTO GO} NAME	Jumps to a label named NAME.
{LABEL LBL} NAME	Creates a label named NAME.
LOADEXT PATH\$...	Loads the shared library specified by PATH\$ as an extension.

LOCATE {X%, [Y%] [X%], Y%}	Moves the cursor to X%, Y%.
{MKDIR MD} NAME\$	Makes a directory named NAME\$.
{MOVE MV RENAME REN} OLD\$, NEW\$	Moves/renames the file/directory OLD\$ to NEW\$.
PRINT [{STRING\$/NUMBER%} {,/;} ...]	Prints text on the screen, ‘;’ means print without newline and ‘,’ means print tab.
PUT [{STRING\$/NUMBER%} ...]	Puts STRING\$ or NUMBER on the terminal.
REDIM ARRAY, MAX%	Resizes ARRAY so the maximum index is MAX%.
{REMOVE RM} PATH\$	Removes the file or directory PATH\$.
RESETTIMER	Resets the timer.
RETURN	Returns to a position saved by GOSUB.
RLOCATE {X%, [Y%] [X%], Y%}	Moves the cursor by X%, Y%.
RUN FILENAME\$ [, {ARG\$/ARG%}...]	Runs FILENAME\$ in a new session and passes ARG\$/ARG% to it.
RUNA ARRAY	Runs ARRAY[0] in a new session and passes the rest of the elements of ARRAY to it.
{SET LET} VAR, {VAL\$ VAL%} VAR = {VAL\$ VAL%}	Sets the variable VAR to VAL\$ or VAL%.
SH COMMAND\$	Runs COMMAND\$ in sh on Linux and Command Prompt on Windows
{SRAND SRND} SEED%	Seeds the random number generator with SEED%.
SUB NAME	Begins a subfunction declaration.
SWAP ARRAY1, ARRAY2	Swaps two arrays.
UNLOADEXT NAME\$/NUM%	Unloads an extension using its name or the value returned by LOADEXT().
WAIT SEC%	Waits for SEC% seconds.
WAITMS MSEC%	Waits for MSEC% milliseconds.
WAITUS USEC%	Waits for USEC% microseconds.

<code>_AUTOCMDHIST</code>	Enables automatic history saving (saves to <code>‘.clibasic_history’</code> to the user’s home directory, remove this file to disable this feature).
<code>_HIDECUR</code>	Hides the cursor.
<code>_LIMITCMDHIST LIMIT%</code>	Limits the command history to <code>LIMIT%</code> entries.
<code>_LOADCMDHIST FILENAME\$</code>	Loads the command history from <code>FILENAME\$</code> .
<code>_PROMPT STRING\$</code>	Sets the prompt string to solve to <code>STRING\$</code> .
<code>_PROMPTTAB WIDTH%</code>	Sets the prompt tab width to <code>WIDTH%</code> .
<code>_RESETTITLE</code>	Resets the terminal title.
<code>_SAVECMDHIST FILENAME\$</code>	Saves the command history to <code>FILENAME\$</code> .
<code>_SETENV NAME\$, VALUE\$</code>	Sets the environment variable with the name <code>NAME\$</code> to <code>VALUE\$</code> .
<code>_SHATTRIB {ATTRIB\$ ATTRIB%}, {VALUE\$ VALUE%}</code>	Sets the ‘SH’ attribute <code>ATTRIB\$</code> or <code>ATTRIB%</code> to <code>VALUE\$</code> or <code>VALUE%</code> .
<code>_SHOWCUR</code>	Shows the cursor.
<code>_TITLE STRING\$</code>	Sets the terminal title to <code>STRING\$</code> .

<pre>_TXTATTRIB {ATTRIB\$ ATTRIB%}, {VALUE\$, VALUE%}</pre>	<p>Sets the text attribute ATTRIB\$ or ATTRIB% to VALUE\$ or VALUE%.</p> <p>Available attributes:</p> <ul style="list-style-type: none"> 0/"RESET" 1/"BOLD" 2/"ITALIC" 3/"UNDERLINE" 4/"DBL_UNDERLINE"/"DOUBLE_UNDERLINE" 5/"SQG_UNDERLINE"/"SQUIGGLY_UNDERLINE" 6/"STRIKETROUGH" 7/"OVERLINE" 8/"DIM" 9/"BLINK" 10/"HIDDEN" 11/"REVERSE" 12/"UNDERLINE_COLOR" 13/"FGC" 14/"BGC" 15/"TRUECOLOR"/"TRUE_COLOR"/"24BITCOLOR"/"24 BIT_COLOR"
<pre>_TXTLOCK</pre>	<p>Stops the keyboard from echoing on the terminal.</p>
<pre>_TXTUNLOCK</pre>	<p>Undoes the effect of _TXTLOCK.</p>
<pre>_UNSETENV NAME\$</pre>	<p>Unsets the environment variable named NAME\$.</p>

Functions:

ABS (NUM%)	Returns the absolute value of NUM%.
AND (VAL1%, VAL2%)	Returns the result of a bitwise and on VAL1% using VAL2%. Both values are read as a 64-bit unsigned integer.
ASC (STRING\$ [, POSITION%])	Returns the ASCII code of character POSITION% (starting at and defaulting if not specified to zero) of STRING\$.
BASENAME\$ (FILENAME\$)	Returns the file name out of the file path provided by FILENAME\$.
BGC ()	Returns the current background color.
CALLFUNC (SUBFUNC)	Calls and returns the value returned by the subfunction SUBFUNC.
{CHDIR CD} (DIR\$)	Attempts to change the current directory to DIR\$ and returns 0 on success and an error code on failure (the error code is taken directly from the C variable 'errno' set by the C 'chdir()' function).
CHR\$ (CODE%)	Returns ASCII character CODE%.
CHRAT\$ (STRING\$, POSITION%)	Returns the character at POSITION% of STRING\$.
CWD\$ ()	Returns the current working directory.
CINT (NUMBER%)	Returns NUMBER% rounded.
COS (NUMBER%)	Returns the cosine of NUMBER%.
COSH (NUMBER%)	Returns the hyperbolic cosine of NUMBER%.
CURX ()	Returns the X position of the cursor.
CURY ()	Returns the Y position of the cursor.

DATE (ATTRIB\$ ATTRIB%)	Returns the date attribute ATTRIB\$ or ATTRIB%. Available attributes: 0/"SEC"/"SECOND" 1/"MIN"/"MINUTE" 2/"HR"/"HOUR" 3/"DAY" 4/"MON"/"MONTH" 5/"YEAR" 6/"WDAY"/"WEEKDAY" 7/"YDAY"/"YEARDAY" 8/"DST"/"DAYLIGHT"/"DAYLIGHTSAVING"
DIRNAME\$ (FILENAME\$)	Returns the directory name out of the file path provided by FILENAME\$.
EOF (FILENUM%)	Returns 1 if the end of file number FILENUM% has been reached and 0 otherwise. -1 is returned if an invalid FILENUM% is passed.
EOFD (FILENUM%)	Returns 1 if the file cursor for file number FILENUM% has passed the size of the file data and 0 otherwise. -1 is returned if an invalid FILENUM% is passed.
EXEC (PROGRAM\$ [, {ARG\$/ARG%} ...])	Executes PROGRAM\$, passes the remaining arguments to PROGRAM\$, then returns the exit code of the program or 127 if running PROGRAM\$ failed.
EXEC\$ (PROGRAM\$ [, {ARG\$/ARG%} ...])	Executes PROGRAM\$, passes the remaining arguments to PROGRAM\$, then returns the output of PROGRAM\$.
EXECA (ARRAY\$)	Executes ARRAY\$ where element 0 is the program and the rest of the elements are passed as arguments, then returns the exit code of the program or 127 if running PROGRAM\$ failed.
EXECA\$ (ARRAY\$)	Executes ARRAY\$ where element 0 is the program and the rest of the elements are passed as arguments, then returns the output of PROGRAM\$.
EXP (NUMBER%)	Returns the exponent of NUMBER%.
EXTENSIONS\$ ()	Returns a list of loaded extensions.

EXTLOADED (NAME\$/NUM%)	Returns 1 if an extension with the name NAME\$ or a number matching NUM% is loaded, 0 otherwise.
EXTNAME\$ (NUM%)	Returns the name of an extension using a number provided by LOADEXT(), or nothing on failure.
FCLOSE (FILENUM%)	Closes FILENUM% and returns 1 if successful, otherwise returns 0. If -1 is passed then all open files will be closed.
FILES\$ ([DIR\$])	Returns a list of files and directories in the directory specified by DIR\$ or the current directory if DIR\$ is not provided.
FLUSH (FILENUM%)	Returns 1 if flushing file number FILENUM% was successful and 0 otherwise. -1 is returned if an invalid file number is passed.
FOPEN (FILE\$, MODE\$)	Opens file FILE\$ with mode MODE\$ and returns a file number if successful, otherwise -1 is returned.
FREAD (FILENUM%)	Returns the next character of file number FILENUM% as a number or -1 if unsuccessful or the end of the file was reached.
FREAD\$ (FILENUM%)	Returns the next character of file number FILENUM%. An empty
FSIZE (FILENUM%)	Returns the position of the last character in FILENUM% or -1 if the file number does not exist.
FWRITE (FILENUM%, STRING\$)	Writes STRING\$ to file number FILENUM% and returns 1 if successful, otherwise returns 0.
FGC ()	Returns the current foreground color.
HEIGHT ()	Returns the height of the terminal.
HEX\$ (NUMBER%)	Returns the hexadecimal version of NUMBER%.
INKEY\$ ()	Returns a character from the terminal.
INPUT\$ (PROMPT\$)	Returns a string after prompting for PROMPT\$ (asks “?: ” if PROMPT\$ is not supplied).
INT (NUMBER%)	Returns NUMBER% rounded down.

ISFILE (PATH\$)	Returns 1 if PATH\$ is a file, 0 if PATH\$ is a directory, or -1 if PATH\$ cannot be found.
LCASE\$ (STRING\$)	Returns the lower-case version of STRING\$.
LEN (STRING\$)	Returns the length of STRING\$.
LIMIT (NUMBER%, {MAX% MIN%, [MAX%] })	Returns NUMBER% trimmed to MIN% and/or MAX%.
LINE\$ (LINE%, STRING\$)	Returns line LINE% (starting at zero) of STRING\$.
LINES (STRING\$)	Returns the line count of STRING\$.
LOADEXT (PATH\$...)	Loads the shared library specified by PATH\$ as an extension and returns a number >= 0 if successful or -1 otherwise.
LOG (NUMBER%)	Returns the natural logarithm of NUMBER%.
LOG10 (NUMBER%)	Returns the common logarithm if NUMBER%.
{MKDIR MD} (PATH\$)	Makes a directory with the location of PATH\$ and returns 1 if successful, otherwise returns 0.
MOD (NUMBER%)	Returns the modulus of NUMBER%.
{MOVE MV RENAME REN} (OLD\$, NEW\$)	Moves/renames the file/directory OLD\$ to NEW\$ and returns 1 if successful, otherwise 0 is returned.
NOT (VAL%)	Reads VAL% as a 64-bit unsigned integer and returns the result of a bitwise not.
OCT\$ (NUMBER%)	Returns the octal version of NUMBER%.
OR (VAL1%, VAL2%)	Returns the result of a bitwise or on VAL1% using VAL2%. Both values are read as a 64-bit unsigned integer.
PAD (STRING\$/NUMBER%, WIDTH% [, CHAR\$])	Returns STRING\$/NUMBER padded to WIDTH% using CHAR\$. CHAR\$ must contain one character, if CHAR\$ is not provided then ' ' is used for STRING\$ and '0' is used for NUMBER%.
PI ()	Returns Pi.

<code>{RAND RND} ([MIN%,] MAX%)</code>	Returns a random number from MIN (0 if MIN% is not supplied) to MAX%.
<code>READEXTNAME\$ (PATH\$)</code>	Reads the library PATH\$ as an extension and returns the name if successful, or nothing on failure.
<code>{REMOVE RM} (PATH\$)</code>	Returns 1 if the removal of PATH\$ was successful, otherwise returns 0.
<code>RGB (RED%, GREEN%, BLUE%)</code>	Returns a 24-bit color code from separate red, green, and blue values.
<code>SH (COMMAND\$)</code>	Runs COMMAND\$ in the command line or shell and returns the exit status.
<code>SH\$ (COMMAND\$)</code>	Runs COMMAND\$ in the command line or shell and returns the text the command outputs.
<code>SHIFT (VAL%, AMOUNT%)</code>	Shifts VAL% left bitwise by AMOUNT%.
<code>SIN (NUMBER%)</code>	Returns the sine of NUMBER%.
<code>SINH (NUMBER%)</code>	Returns the hyperbolic sine of NUMBER%.
<code>SNIP\$ (STRING\$, {TO% {[FROM%], TO% FROM% [, TO%]} })</code>	Returns the part of STRING\$ defined by FROM% through TO%.
<code>STR\$ (NUMBER%)</code>	Returns NUMBER% as a string.
<code>TAN (NUMBER%)</code>	Returns the tangent of NUMBER%.
<code>TANH (NUMBER%)</code>	Returns the hyperbolic tangent of NUMBER%.
<code>TIME ()</code>	Returns the current time in seconds.
<code>TIMEMS ()</code>	Returns the current time in milliseconds.
<code>TIMEUS ()</code>	Returns the current time in microseconds.
<code>TIMER ()</code>	Returns the timer value in seconds.
<code>TIMERMS ()</code>	Returns the timer value in milliseconds.
<code>TIMERUS ()</code>	Returns the timer value in microseconds.
<code>TRUECOLOR ()</code>	Returns 1 if the truecolor attribute is enabled, 0 otherwise.
<code>UCASE\$ (STRING\$) "CALL</code>	Returns the upper-case version of STRING\$.

<code>UNLOADEXT (NAME\$/NUM%)</code>	Unloads an extension using its name or the value returned by <code>LOADEXT()</code> and returns 1 if successful or 0 otherwise.
<code>VAL (STRING\$ [, TYPE%])</code>	Returns the numeric value of <code>STRING\$</code> , <code>TYPE%</code> is what type the number is (0/not supplied = DEC, 1 = HEX, 2 = OCT, 3 = BIN).
<code>WIDTH ()</code>	Returns the width of the terminal.
<code>XOR (VAL1%, VAL2%)</code>	Returns the result of a bitwise exclusive or on <code>VAL1%</code> using <code>VAL2%</code> . Both values are read as a 64-bit unsigned integer.
<code>_ARG\$ ([N%])</code>	Returns argument <code>N%</code> or all arguments except #0 if <code>N%</code> is not provided. Argument 0 is the full/real path to the program file.
<code>_ARGC ()</code>	Returns how many arguments were passed to the program
<code>_BITS\$ ()</code>	Returns the executable bit format.
<code>_ENV\$ (STRING\$)</code>	Returns the content of the environment variable defined by <code>STRING\$</code> .
<code>_ENVSET (STRING\$)</code>	Returns 1 if the environment variable defined by <code>STRING\$</code> is set and 0 otherwise.
<code>_ERRNOSTR\$ (ERRNO%)</code>	Returns the corresponding error string for error number <code>ERRNO%</code> .
<code>_FILEERROR ()</code>	Returns the last error produced by a file I/O command or function.
<code>_HOME\$ ()</code>	Returns the path to the user's home directory.
<code>_ISATTY (N%)</code>	Checks if a standard file descriptor is a TTY. Available values for <code>N%</code> : 0 (STDIN) 1 (STDOUT) 2 (STDERR)
<code>_OS\$ ()</code>	Returns the current operating system name.
<code>_PROMPT\$ ()</code>	Returns the prompt string.

<code>_RET()</code>	Returns the exit status generated by RUN, CALL, or any EXEC or SH command or function.
<code>_STARTCMD\$()</code>	Returns the full/real path to the command used to start CLIBASIC.
<code>_TEST(CONDITION)</code>	Evaluates and returns the result of testing CONDITION.
<code>_TXTATTRIB {ATTRIB\$ ATTRIB%}</code>	Returns the text attribute ATTRIB\$/ATTRIB%. Refer to _TXTATTRIB for the list of attributes.
<code>_TXTLOCK()</code>	Returns 1 if the text lock is in effect and 0 otherwise.
<code>_VER\$()</code>	Returns the CLIBASIC version.

Logic Commands:

BREAK	Breaks out of a DO or FOR block.
CONTINUE	Skips the remaining commands in a DO or FOR block.
DO	Begins a DO block.
ELSE	Inverts an IF command.
ENDIF	Ends an IF block.
FOR VAR, INIT%, CONDITION, I%	Begins a FOR block, sets VAR to INIT% and loops while adding I% to VAR while CONDITION is true.
IF CONDITION	Begins an IF block and runs commands if CONDITION is true.
LOOP	Jumps to the beginning of a DO block.
LOOPWHILE CONDITION	Jumps to the beginning of a DO block if CONDITION is true.
NEXT	Jumps to the beginning of a FOR block.
REM	Comments out one command.
{WHILE DOWHILE} CONDITION	Begins a DO block while CONDITION is true.

Symbols:

<code>?</code>	Shortcut to <code>PRINT</code> .
<code>></code>	Shortcut to <code>PUT</code> .
<code>\$</code>	Shortcut to <code>SH</code> .
<code>@</code>	Shortcut to <code>LABEL</code> .
<code>%</code>	Shortcut to <code>GOTO</code> .
<code>~</code>	Shortcut to <code>_TEST()</code> .
<code>{ ' # }</code>	Comment until the end of the line.

Comparing:

=	Equal to
<>	Not equal to
>	Greater than
<	Less than
{>= =>}	Greater than or equal to
{<= =<}	Less than or equal to
&	And
	Or

About :

Cover image: PQCraft

- **Cover image geometric shapes original image:** pikisuperstar/Freepik
- **Cover image creation program:** GIMP

Manual content: PQCraft

- **Manual content creation program:** LibreOffice