

CLIBASIC Language Manual

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 Unix-based systems such as Linux, to mimic the Windows behavior of opening CLIBASIC in a console window, CLIBASIC will attempt to open in xterm if `GUI_CHECK` is defined.
- 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.

Commands:

CALL FILENAME\$	Opens and runs FILENAME\$ in the current session. Called programs will retain the same arguments as the parent.
{CHDIR CD} DIR\$	Changes the current directory to DIR\$.
CLS [COLOR]	Clears the screen with optional color.
COLOR {FGC [, BGC] [FGC], BGC}	Sets the foreground color to FGC and the background color to BGC.
DEL {VAR\$ VAR}	Deletes the variable VAR\$ or VAR.
DIM {VAR\$ VAR}, MAX [, {INIT\$ INIT}]	Makes an array with the max index being MAX and the initial value for each element being INIT/INIT\$.
EXEC PROGRAM\$ [, {ARG\$/ARG}...]	Runs PROGRAM\$ and passes the remaining arguments to PROGRAM\$.
{EXIT QUIT} [CODE]	Exits with CODE (or 0 if CODE is not supplied).
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.
FLUSH FILENUM	Flushes the file number FILENUM.
FSEEK FILENUM, POSITION	Moves the file cursor of file number FILENUM to POSITION.
FWRITE FILENUM, STRING\$	Writes STRING\$ to file number FILENUM.
{GOTO GO} NAME	Jumps to label with the value NAME.
{LABEL LBL} NAME	Creates a label with the value NAME.
LOCATE {X, [Y] [X], Y}	Moves the cursor to X, Y. If a negative number is provided then the cursor will be moved back by the absolute value of that amount.
{MKDIR MD} NAME\$	Makes a directory named NAME\$.
{MOVE MV RENAME REN} OLD\$, NEW\$	Moves/renames the file/directory OLD\$ to NEW\$.

<code>PRINT [{STRING\$/NUMBER} {,/;} ...]</code>	Prints text on the screen, ‘;’ means print without newline and ‘,’ means print tab.
<code>PUT [{STRING\$/NUMBER} ...]</code>	Puts STRING\$ or NUMBER on the terminal.
<code>{REMOVE RM} PATH\$</code>	Removes the file or directory PATH\$.
<code>RESETTIMER</code>	Resets the timer.
<code>RUN FILENAME\$ [, {ARG ARG\$}...]</code>	Runs FILENAME\$ in a new session and passes ARG/ARG\$ to it.
<code>{{SET LET} {VAR\$ VAR}, {STRING\$ NUMBER} {VAR\$ VAR} = {STRING\$ NUMBER}}</code>	Sets the variable VAR\$ or VAR to STRING\$ or NUMBER.
<code>SH COMMAND\$</code>	Runs COMMAND\$ in sh on Linux and Command Prompt on Windows
<code>{SRAND SRND} SEED</code>	Seeds the random number generator with SEED.
<code>WAIT SEC</code>	Waits for SEC seconds.
<code>WAITMS MSEC</code>	Waits for MSEC milliseconds.
<code>WAITUS USEC</code>	Waits for USEC microseconds.
<code>_AUTOCMDHIST</code>	Enables automatic history saving (saves to ‘.clibasic_history’ to the user’s home directory, remove this file to disable this feature).
<code>_LOADCMDHIST FILENAME\$</code>	Loads the command history from FILENAME\$.
<code>_PROMPT STRING\$</code>	Sets the prompt string to solve to STRING\$.
<code>_PROMPTTAB WIDTH</code>	Sets the prompt tab width to WIDTH.
<code>_RESETTITLE</code>	Resets the terminal title.
<code>_SAVECMDHIST FILENAME\$</code>	Saves the command history to FILENAME\$.
<code>_SETENV NAME\$, VALUE\$</code>	Sets the environment variable with the name NAME\$ to VALUE\$.
<code>_SHATTRIB {ATTRIB\$ ATTRIB}, {VALUE\$ VALUE}</code>	Sets the ‘SH’ attribute ATTRIB\$ or ATTRIB to VALUE\$ or VALUE.
<code>_TITLE STRING\$</code>	Sets the terminal title to STRING\$.

<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:

<code>ASC (STRING\$ [, POSITION])</code>	Returns the ASCII code of character <code>POSITION</code> (starting at and defaulting if not specified to zero) of <code>STRING\$</code> .
<code>BASENAME\$ (FILENAME\$)</code>	Returns the file name out of the file path provided by <code>FILENAME\$</code> .
<code>BGC ()</code>	Returns the current background color.
<code>{CHDIR CD} (DIR\$)</code>	Attempts to change the current directory to <code>DIR\$</code> 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).
<code>CHR\$ (CODE)</code>	Returns ASCII character <code>CODE</code> .
<code>CHRAT\$ (STRING\$, POSITION)</code>	Returns the character at <code>POSITION</code> of <code>STRING\$</code> .
<code>CWD\$ ()</code>	Returns the current working directory.
<code>CINT (NUMBER)</code>	Returns <code>NUMBER</code> rounded.
<code>COS (NUMBER)</code>	Returns the cosine of <code>NUMBER</code> .
<code>COSH (NUMBER)</code>	Returns the hyperbolic cosine of <code>NUMBER</code> .
<code>CURX ()</code>	Returns the X position of the cursor.
<code>CURY ()</code>	Returns the Y position of the cursor.
<code>DIRNAME\$ (FILENAME\$)</code>	Returns the directory name out of the file path provided by <code>FILENAME\$</code> .
<code>EOF (FILENUM)</code>	Returns 1 if the end of file number <code>FILENUM</code> has been reached and 0 otherwise. -1 is returned if an invalid <code>FILENUM</code> is passed.
<code>EOFD (FILENUM)</code>	Returns 1 if the file cursor for file number <code>FILENUM</code> has passed the size of the file data and 0 otherwise. -1 is returned if an invalid <code>FILENUM</code> is passed.
<code>EXEC (PROGRAM\$ [, {ARG\$/ARG}...])</code>	Runs <code>PROGRAM\$</code> , passes the remaining arguments to <code>PROGRAM\$</code> , then returns the exit code of the program or 127 if running <code>PROGRAM\$</code> failed.

EXEC\$(PROGRAM\$ [, {ARG\$/ARG}...])	Runs PROGRAM\$, passes the remaining arguments to PROGRAM\$, then returns the output of PROGRAM\$.
EXP(NUMBER)	Returns the exponent of NUMBER.
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.

<code>LCASE\$(STRING\$)</code>	Returns the lower-case version of <code>STRING\$</code> .
<code>LEN\$(STRING\$)</code>	Returns the length of <code>STRING\$</code> .
<code>LIMIT(NUMBER, {MAX MIN, [MAX] })</code>	Returns <code>NUMBER</code> trimmed to <code>MIN</code> and/or <code>MAX</code> .
<code>LINE\$(LINE, STRING\$)</code>	Returns line <code>LINE</code> (starting at zero) of <code>STRING\$</code> .
<code>LINES\$(STRING\$)</code>	Returns the line count of <code>STRING\$</code> .
<code>LOG(NUMBER)</code>	Returns the natural logarithm of <code>NUMBER</code> .
<code>LOG10(NUMBER)</code>	Returns the common logarithm if <code>NUMBER</code> .
<code>{MKDIR MD}(PATH\$)</code>	Makes a directory with the location of <code>PATH\$</code> and returns 1 if successful, otherwise returns 0.
<code>MOD(NUMBER)</code>	Returns the modulus of <code>NUMBER</code> .
<code>{MOVE MV RENAME REN}(OLD\$, NEW\$)</code>	Moves/renames the file/directory <code>OLD\$</code> to <code>NEW\$</code> and returns 1 if successful, otherwise 0 is returned.
<code>OCT\$(NUMBER)</code>	Returns the octal version of <code>NUMBER</code> .
<code>PAD\$(STRING\$/NUMBER, WIDTH [, CHAR\$])</code>	Returns <code>STRING\$/NUMBER</code> padded to <code>WIDTH</code> using <code>CHAR\$</code> . <code>CHAR\$</code> must contain one character, if <code>CHAR\$</code> is not provided then ' ' is used for <code>STRING\$</code> and '0' is used for <code>NUMBER</code> .
<code>PI()</code>	Returns Pi.
<code>{RAND RND}({MAX MIN, MAX})</code>	Returns a random number from <code>MIN</code> (0 if <code>MIN</code> is not supplied) to <code>MAX</code> .
<code>{REMOVE RM}(PATH\$)</code>	Returns 1 if the removal of <code>PATH\$</code> 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 <code>COMMAND\$</code> in the command line or shell and returns the exit status.
<code>SH\$(COMMAND\$)</code>	Runs <code>COMMAND\$</code> in the command line or shell and returns the text the command outputs.
<code>SIN(NUMBER)</code>	Returns the sine of <code>NUMBER</code> .

<code>SINH (NUMBER)</code>	Returns the hyperbolic sine of <code>NUMBER</code> .
<code>SNIP\$(STRING\$, {TO {[FROM], TO FROM [, TO]}})</code>	Returns the part of <code>STRING\$</code> defined by <code>FROM</code> through <code>TO</code> .
<code>STR\$(NUMBER)</code>	Returns <code>NUMBER</code> as a string.
<code>TAN (NUMBER)</code>	Returns the tangent of <code>NUMBER</code> .
<code>TANH (NUMBER)</code>	Returns the hyperbolic tangent of <code>NUMBER</code> .
<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>UCASE\$(STRING\$)</code>	Returns the upper-case version of <code>STRING\$</code> .
<code>VAL (STRING\$ [, TYPE])</code>	Returns the numeric value of <code>STRING\$</code> , <code>TYPE</code> is what type the number is (0 = DEC, 1 = HEX, 2 = OCT, etc/not supplied = Auto (scanf auto-detect)).
<code>WIDTH ()</code>	Returns the width of the terminal.
<code>_ARGC ()</code>	Returns how many arguments were passed to the program
<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>_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 function or command.

<code>_HOME\$()</code>	Returns the path to the user's home directory.
<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>_TXTLOCK()</code>	Returns 1 if the text lock is in effect and 0 otherwise.
<code>_VER\$()</code>	Returns the CLIBASIC version.

Logic Commands:

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/LOOP block.
LOOPWHILE CONDITION	Jumps to the beginning of a DO/LOOP 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:

?	Shortcut to PRINT.
\$	Shortcut to SH.
@	Shortcut to LABEL.
%	Shortcut to GOTO.
~	Shortcut to _TEST().
{ ' # }	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