WIKIPEDIA

Sinclair BASIC

Sinclair BASIC is a dialect of the programming language BASIC used in the 8-bit home computers from Sinclair Research and Timex Sinclair. The Sinclair BASIC interpreter was made by Nine Tiles Networks Ltd.[1]

Sinclair BASIC

Paradigm	Imperative
Designed by	John Grant, Steve Vickers
Developer	Nine Tiles Networks, Sinclair Research
First appeared	1979
Platform	ZX80, ZX81, ZX Spectrum
License	Proprietary

Contents

History

Sinclair BASIC was originally developed in 1979 for the ZX80 by Nine Tiles. The programmers were John Grant, the owner of Nine Tiles, and Steve Vickers.

It was initially an incomplete implementation of the 1978 American National Standards Institute (ANSI) minimal BASIC standard with integer arithmetic only, termed the 4K BASIC (for its ROM size) for the ZX80. It evolved through the floating-point 8K BASIC for the ZX81 and TS1000 (which was also available as an upgrade for the ZX80[2]), and became an almost complete version in the 16 KB ROM ZX Spectrum. It is present in all ZX Spectrum compatibles.

As of 2015, interpreters exist for modern operating systems, and older systems, that allow Sinclair Basic to be used easily.

Syntax

New BASIC programmers might start with a simple program, perhaps using the language's PRINT statement to display a message on the screen; a well-known and often-replicated example is Kernighan and Ritchie's Hello world program:

```
10 PRINT "Hello, World!"
```

Keywords

On the 16K/48K ZX Spectrum, there are 88 keywords in Sinclair BASIC, denoting commands (of which there are 50), functions and logical operators (31), and other keywords (16, including 9 which are also commands or functions):

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
ABS	number	EXTENDED MODE then G	Function	Returns the absolute value of number[3]
ACS	number	EXTENDED MODE then SYMBOL SHIFT + W	Function	Returns the arccosine of number[4]
AND		SYMBOL SHIFT + Y	Logical operator	Returns true if both conditions on either side of the AND keyword are true, else returns false[note 3][5]
ASN	number	EXTENDED MODE then SYMBOL SHIFT + Q	Function	Returns the arcsine of number[6]
AT	line, column;	SYMBOL SHIFT + I	Other	Used in a PRINT statement to print at the line and column specified;[7] for example, PRINT AT 5,10;"*" puts a star in column 10 of line 5.
ATN	number	EXTENDED MODE then SYMBOL SHIFT + E	Function	Returns the arctangent of number[4]
ATTR	(line, column)	EXTENDED MODE then SYMBOL SHIFT + L	Function	Returns a byte containing information on the colours of the text cell on the screen, corresponding to the specified line and column;note that, unlike most Sinclair BASIC keywords, the parentheses are required; the first three bits indicate the ink (foreground)colour, the fourth, fifth and sixth bits the paper (background) colour, the seventh bit whether the colours are bright or not, and the eight, whether they are flashing[8]
BEEP	duration, pitch	EXTENDED MODE then SYMBOL SHIFT + Z	Command	Produces sound from the computer's speaker; duration is in seconds, pitch is in semitones above (positive value) or below (negative value) middle C[9]
BIN	number	EXTENDED MODE then B	Other	Indicates number is in binary notation[10]
BORDER	number	B	Command	Sets the outer border of the screen to the colour specified by the

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
				number[note 4][11]
BRIGHT	number	EXTENDED MODE then SYMBOL SHIFT + B	Command/other	Makes all following colours brighter if number is 1, or its normal shade if 0[note 5][12]
CAT	number	EXTENDED MODE then SYMBOL SHIFT + 9	Command	Displays contents of ZX Microdrive specified by number[note 6][13]
CHR$	number	EXTENDED MODE then W	Function	Returns the character corresponding to the decimal number in the computer's character set[14]
CIRCLE	x, y, r	EXTENDED MODE then SYMBOL SHIFT + H	Command	Draws a circle with its centre at coordinates (x,y) and radius r[15]
CLEAR	address	X	Command	Clears the screen,[16] all variables and the GO SUB stack,[17] and optionally sets the maximum RAM address to be used by BASIC[18]
CLOSE #	number	EXTENDED MODE then SYMBOL SHIFT + 5	Command	Closes the specified stream number for access[note 6]
CLS		V	Command	Clears all text and graphics from the screen[16]
CODE	string	EXTENDED MODE then I	Function/other	Returns the number corresponding to the first character in string in the computer's character set;[14] also used to save arbitrary chunks memory to tape, disk, etc. and load them back in — see LOAD, SAVE and VERIFY for details
CONTINUE		C	Command	Restarts a program after it has stopped due to an error or the user pressing the CAPS SHIFT + SPACE or BREAK keys[19]
COPY		Z	Command	Sends the currently displayed screen to the printer[20]
COS	number	EXTENDED MODE then W	Function	Returns the cosine of number[21]
DATA	comma-separated values	EXTENDED MODE then D	Command/other	Provides numbers and/or strings to use with the READ

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
				command[22] and allows saving the contents of an array to tape when used with the SAVE command[23] (as `SAVE filename DATA array name()`
`DEF FN`	`name(variable) = operation`	EXTENDED MODE then 1	Command	Defines a custom function that can be used with the FN command;function definitions must be of the form `f(x)=operations`, for example `f(x)=x*2` and the function name may not consist of more than one letter, plus a $-symbol if the function returns a string[24]
`DIM`	`variable(dimensions)`	D	Command	Declares an array with the specified `dimensions`, which may be multi-dimensional (for example, `DIM a(10,10)`; if used with strings, the last dimension indicates the length of each of the strings (thus, `DIM a$(2,5)` is an array of two strings each of five characters long, and `DIM b$(5)` is one string of five characters)[25]
`DRAW`	`x, y [, r]`	W	Command	Draws a line in the current `INK` colour to coordinates (x,y) from the coordinates used by the previous `PLOT` or `DRAW` command; if the optional `r` is supplied, it indicates the radius of the circle segment to be drawn, in radians[26]
`ERASE`	`drive;"filename"`	EXTENDED MODE then SYMBOL SHIFT + 7	Command	Deletes the specified file from a ZX Microdrive[note 6][27]
`EXP`	`number`	EXTENDED MODE then X	Function	Returns *e* to the power `number`[28]
`FLASH`	`number`	EXTENDED MODE then SYMBOL SHIFT + V	Command/other	Makes all following text alternate its foreground (INK) and background (PAPER) colours[note 5]
`FN`	`function(value)`	EXTENDED MODE then SYMBOL SHIFT + 2	Function	Calls the `function` defined earlier in the program using `DEF FN`[24]

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
`FORMAT`	`drive;"name"`	EXTENDED MODE then SYMBOL SHIFT + 0	Command	Formats the cartridge in the indicated Microdrive and assigns it the identifier name[note 6][29]
`FOR`	`variable = start TO end`	F	Command	Starts a `FOR`-`NEXT` loop;[30] the variable name may only be one character long[31]
`GO SUB`	`number`	H	Command	Makes the program jump to the BASIC line specified by `number`; when the program encounters the command `RETURN`, it will jump back to the statement after the `GO SUB`[32]
`GO TO`	`number`	G	Command	Makes the program jump to the BASIC line specified by `number`
`IF`	`condition THEN`	U	Command	Evaluates the `condition`, and if true, executes the statement that follows the keyword `THEN` that must come after the condition,[33] for example `IF a=1 THEN LET b=2`[note 7]
`IN`	`address`	EXTENDED MODE then SYMBOL SHIFT + I	Function	Returns a byte read from the hardware input/output port corresponding to the `address`[34]
`INK`	`number`	EXTENDED MODE then SYMBOL SHIFT + X	Command/other	Sets the foreground colour for text and graphics[note 4][note 5][35]
`INKEY$`		EXTENDED MODE then SYMBOL SHIFT + Z	Function	Returns a string representing the key being pressed on the keyboard at the moment the function is called, or an empty string if none is,[36] but does not wait for a keypress
`INPUT`	`[prompt,] variable`	I	Command	Halts program execution until the user types in something on the keyboard and presses the `Enter` key, then stores the entered value in the specified `variable`; if the optional `prompt` is supplied, this will be shown on the screen[37]

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
`INT`	`number`	EXTENDED MODE then R	Function	Returns the integer value of `number`, rounding down to the nearest whole number[3] (thus, `INT -1.1` returns −2, not −1)
`INVERSE`	`number`	EXTENDED MODE then SYMBOL SHIFT + M	Command/other	Reverses the colours on all following text if `number` is 1, so that it uses the current ink colour for the background and the current paper colour for the text, or sets them back to normal if `number` is 0[note 5][38]
`LEN`	`string`	EXTENDED MODE then K	Function	Returns the number of characters (bytes) in `string`[39]
`LET`	`variable=value`	L	Command	Assigns `value` to the named variable[40]
`LINE`		EXTENDED MODE then SYMBOL SHIFT + 3	Other	▪ When used in an `INPUT` statement before a string variable, will not put quotation marks (`""`) around its prompt,[7] for example `INPUT "Name: "; LINE n$` ▪ When used in a `SAVE` statement so that when the BASIC program being saved is loaded again, it starts automatically at the line number indicated[41]
`LIST`	`[number]`	K	Command	Outputs the current BASIC program to the screen; if the optional `number` is provided, it omits all lines with a lower number[42]
`LLIST`	`[number]`	EXTENDED MODE then SYMBOL SHIFT + V	Command	As `LIST` except the listing is output to the printer[20]
`LN`	`number`	EXTENDED MODE then SYMBOL SHIFT + Z	Function	Returns the natural logarithm of `number`[43]
`LOAD`	`"[filename]" [CODE [address[, length]]| DATA variable()]`	J	Command	Loads a program or data into RAM from tape, ZX Microdrive, disk, etc., deleting any existing BASIC program and variables;[37] if an

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
				empty string ("") is provided, this loads the first program found, else it will search the tape for the program named in the string; if the optional `CODE` is provided, will load the program into memory at the address it had when it was saved, or at the specified `address` (`length` is intended as a safety, to try and load the right program in case there are multiple on the tape with the right name but of different lengths);[44] if the optional `DATA variable()` is provided, will load the data from the tape into the array named `variable()`[23]
`LPRINT`	`text`	EXTENDED MODE then SYMBOL SHIFT + C	Command	As `PRINT` except output is sent to the printer[20]
`MERGE`	`"[filename]"`	EXTENDED MODE then SYMBOL SHIFT + T	Command	As `LOAD`, except it does not delete the current program and variables; if a line number exists in both, that of the newly loaded program overwrites the existing one[45]
`MOVE`	`stream1 TO stream2`	EXTENDED MODE then SYMBOL SHIFT + 6	Command	Moves data from one stream (keyboard, screen, file, printer, network, etc.) to another[note 6][46]
`NEW`		A	Command	Erases the current BASIC program and all variables[37]
`NEXT`	`variable`	N	Command	Closes a `FOR-NEXT` loop; the variable must match that of the corresponding `FOR` command[47] — “empty” `NEXT`s to refer to the immediately preceding `FOR` in the program are not allowed
`NOT`	`condition`	SYMBOL SHIFT + S	Logical operator	Returns true if the `condition` is false, else returns false[note 3][5]

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
OPEN #	stream	EXTENDED MODE then SYMBOL SHIFT + 4	Command	Opens a stream for reading from and/or writing to[note 6][48]
OR		SYMBOL SHIFT + Y	Logical operator	Returns true if either of the conditions on either side of the OR keyword are true, else returns false[note 3][5]
OUT	address, value	EXTENDED MODE then SYMBOL SHIFT + O	Command	Sends the value (a byte) to the hardware [Memory-mapped I/O|input/output port] corresponding to the address[34]
OVER	number	EXTENDED MODE then SYMBOL SHIFT + N	Command/other	Will make following text overprint with an XOR operation what is already on the screen if number is 1, instead of erasing it, or erase it if number is 0[note 5][38]
PAPER	number	EXTENDED MODE then SYMBOL SHIFT + C	Command/other	Sets the background colour for text and graphics[note 4][note 5][35]
PAUSE	delay	M	Command	Halts program execution for the specified delay, in $\frac{1}{50}$ of a second in Europe or $\frac{1}{60}$ in North America[49] (thus, PAUSE 50 halts for one second in Europe)
PEEK	address	EXTENDED MODE then O	Function	Returns a byte representing the contents of the memory location pointed to by address[50]
PI		EXTENDED MODE then M	Function	Returns the value of pi[43]
PLOT	x, y	Q	Command	Draws a pixel in the current INK colour on the screen at the coordinates (x,y)[51]
POINT	(x,y)	EXTENDED MODE then SYMBOL SHIFT + 8	Function	Returns 1 if the pixel pointed at graphical coordinates (x,y) is currently in the ink (foreground) colour, else returns 0[15]
POKE	address, value	O	Command	Sets the contents of address in RAM to value[52]

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
PRINT	[AT x,y;] text	P	Command	Prints text (which must be a string or a number) to the screen;[40] if used with AT, will print at the specified text coordinates, else in the first column of the line after that used by the last PRINT statement[7]
RANDOMIZE	[number]	T	Command	Initializes the random number generator; if used without a number (or with 0), it does this based on the computer's internal clock, else it uses the number supplied, which must be in the range [1,65535][53]
READ	variable	EXTENDED MODE then A	Command	Takes a value from a DATA statement and stores it in the named variable: the first time READ is used, it gets the first value after the first DATA, the second time it gets the next one, and so on[22]
REM	text	E	Command	Begins a comment in the source code, meaning that everything after the REM statement is ignored, until the end of the line[37] — note this includes everything after a colon, which normally begins a new segment: `10 REM Nothing to see here : PRINT "Unprintable"` will not produce any output, for example
RESTORE	[number]	EXTENDED MODE then S	Command	Resets where READ commands look for values in DATA statements: if used without a number, the next READ will use the first DATA in the program, with a number it will use the first DATA on or after the line whose number is indicated[54]
RETURN		Y	Command	Returns execution to the first statement following the last GO SUB command that

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
				was executed[32]
RND		EXTENDED MODE then T	Function	Returns a pseudo-random number with eight significant figures in the range [0,1)[55]
RUN	`[number]`	R	Command	Starts the current BASIC program, from its first line if no `number` is specified, else from the line with that number (or the first one after, if it does not exist)[56]
SAVE	`"filename" [DATA variable() | LINE number]`	S	Command	Saves the current BASIC program to tape or other storage device, with the `filename` specified; if the optional `LINE` followed by a line number is used, then the program will start automatically at the indicated line number when it is `LOAD`ed back in; with the optional `DATA`, the command saves the contents of the array named by the variable instead of the current BASIC program[23]
SCREEN$	`[(line, column)]`	EXTENDED MODE then SYMBOL SHIFT +	Function/other	As a function, identifies the character at the specified line and column on the screen.[7] Used after the filename in a `LOAD` or `SAVE` command, indicates that the contents of the display memory should be loaded or saved; this essentially makes it a shortcut for `CODE 16384,6912`[note 8] but does not work with `VERIFY` because the contents of the display memory will be different by the time that command reads back the saved data;[57]
SGN	`number`	EXTENDED MODE then F	Function	Returns 1 if `number` is positive, 0 if it is 0, and −1 if it is negative[3]
SIN	`number`	EXTENDED MODE then Q	Function	Returns the sine of `number`[21]
STEP	`number`	SYMBOL SHIFT + D	Other	Indicates the interval used by a `FOR` statement,[31] for

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
				example FOR n=2 TO 6 STEP 2 will skip n=3 and n=5 in the loop
STOP		SYMBOL SHIFT + A	Command	Ends execution of the current program, exiting to the BASIC editor; can also be given when the computer is waiting for input using the INPUT command;[56] once the program is stopped, it can be resumed with CONTINUE
SQR	number	EXTENDED MODE then H	Function	Returns the square root of number[24]
STR$	number	EXTENDED MODE then Y	Function	Returns the character from the computer's character set corresponding to number[58]
TAB	column	EXTENDED MODE then P	Other	In a PRINT statement, makes sure that the text to be output begins in the column specified, wrapping to the next line as necessary, but never more than one line[16]
TAN	number	EXTENDED MODE then E	Function	Returns the tangent of number[59]
THEN	statement	SYMBOL SHIFT + G	Other	Follows the condition in an IF statement to indicate what should happen when the condition evaluates to true[note 7][33]
TO		SYMBOL SHIFT + F	Other	Indicates a range from the number to the left of TO to the number of the right of it, inclusive;[60] when used with FOR both numbers must be supplied, while if used to slice strings, either may be left off to indicate the start or end of the string
USR	string or address	EXTENDED MODE then L	Function	When called with a single-character string, this returns the memory address at which the glyph for the user-defined graphic character corresponding to that character is defined.[10] If called with an address, it starts machine code

Keyword	Parameters[note 1]	Entered using[note 2]	Type	Summary
				execution at that address (thus making it one of the few Sinclair BASIC functions to have a Side effect) and returns the contents of the Z80's BC register pair.[61]
VAL	string	EXTENDED MODE then J	Function	Evaluates the string as a number and returns the result;[62] this can perform calculations: VAL "1+2" returns 3, for example, and also evaluates variables and even other VAL statements: LET a=1: VAL "a+VAL ""2"""[note 9] also returns 3
VAL$	string	EXTENDED MODE then SYMBOL SHIFT + J	Function	Similar to VAL but evaluates the string as a string[3]
VERIFY	"[filename]"	EXTENDED MODE then SYMBOL SHIFT + R	Command	Reads a program from tape or other storage, much like LOAD, but instead of loading it into memory, compares it to the program that is currently in memory; this is intended to make sure the program, has been SAVEd correctly[63]

Keyword entry

In 48K models and older, the keywords are entered via Sinclair's unique keyword entry system, as indicated on the table. The most common commands need one keystroke only; for example, pressing only P at the start of a line on a Spectrum produces the full command PRINT. Less frequent commands require more complex key sequences: BEEP (for example) is keyed by pressing CAPS SHIFT plus SYMBOL SHIFT to access extended mode (later models include an EXTENDED MODE key), keeping SYMBOL SHIFT held down and pressing Z. Keywords are colour-coded on the original Spectrum keyboard to indicate which mode is required:[64]

- White: key only
- Red on the key itself: SYMBOL SHIFT plus the key
- Green above the key: EXTENDED MODE followed by the key
- Red below the key: EXTENDED MODE followed by SYMBOL SHIFT plus the key

The ZX81 8K BASIC used the shorter forms GOTO, GOSUB, CONT and RAND, whereas the Spectrum used the longer forms GO TO, GO SUB, CONTINUE and RANDOMIZE. The ZX80 4K BASIC also used these longer forms but differed by using the spelling RANDOMISE. The ZX81 8K BASIC was the only version to use FAST, SCROLL, SLOW and UNPLOT. The ZX80 4K BASIC had the exclusive function TL$(); it was equivalent to the string operator (2 TO) in later versions.

ZX Spectrum

ZX Spectrum+

Unique code points are assigned in the ZX80 character set, ZX81 character set and ZX Spectrum character set for each keyword or multi-character operator, i.e. <=, >=, <>, `""` (tokenized on the ZX81 only), `**` (replaced with ↑ on the Spectrum). These are expanded by referencing a token table in ROM. Thus, a keyword uses one byte of memory only, a significant saving over traditional letter-by-letter storage. This also meant that the BASIC interpreter could quickly determine any command or function by evaluating one byte, and that the keywords need not be *reserved words* like in other BASIC dialects or other programming languages, e.g., it is allowed to define a variable named `PRINT` and output its value with `PRINT PRINT`. This is also related to the syntax requirement that every line start with a command keyword, and pressing the one keypress for that command at the start of a line changes the editor from command mode to letter mode. Thus, variable assignment requires `LET` (i.e., `LET a=1` not only `a=1`). This practice is also different from other BASIC dialects. Further, it meant that unlike other BASIC dialects, the interpreter needed no parentheses to identify functions; `SIN x` was sufficient, no `SIN(x)` needed (though the latter was allowed). The 4K BASIC ROM of the ZX80 had a short list of exceptions to this: the functions `CHR$()`, `STR$()`, `TL$()`, `PEEK()`, `CODE()`, `RND()`, `USR()` and `ABS()` did not have one-byte tokens but were typed in letter-by-letter and required the parentheses. They were listed as the INTEGRAL FUNCTIONS on a label above and to the right of the keyboard.[65]

The 128K Spectrum models, the ZX Spectrum 128, +2, +3, +2A, and +2B, also stored keywords internally in one-byte code points, but used a conventional letter-by-letter BASIC input system. They also introduced two new commands:

- `PLAY`, which operated the 128k models' General Instrument AY-3-8910 music chip
- `SPECTRUM`, which switched the 128k Spectrum into a 48k Spectrum compatibility mode

The original Spanish ZX Spectrum 128 included four additional commands in Spanish,[66] one of which was undocumented. These can be translated as:

- `EDIT` (to edit a line number or invoke the full screen string editor)
- `RENUM` (to renumber the program lines)
- `DELETE` (to delete program lines)
- `WIDTH` (to set the column width of the RS232 device, but undocumented as the code was broken)

Unlike the `LEFT$()`, `MID$()` and `RIGHT$()` functions used in the ubiquitous Microsoft BASIC dialects for home computers, parts of strings in Sinclair BASIC are accessed by numeric range. For example, `a$(5 TO 10)` gives a substring starting with the 5th and ending with the 10th character of the variable `a$`. Thus, it is possible to replace the `LEFT$()` and `RIGHT$()` commands by simply omitting the left or right array position respectively; for example `a$( TO 5)` is equivalent to `LEFT$(a$,5)`. Further, `a$(5)` alone is enough to replace `MID$(a$,5,1)`.

Variable names

Variables holding numeric values may be any length, while string and array variable names must consist of only one alphabetical character. Thus, `LET a=5`, `LET Apples=5`, `LET a$="Hello"`, `DIM a(10)` and `DIM a$(10)` are all good, while `LET Apples$="Fruit"`, `DIM Apples(10)` and `DIM Apples$(10)` are not.

The long variable names allowed for numeric variables can include alphanumeric characters after the first character, so `LET a0=5` is allowed but not `LET 0a=5`. The long variable names can also include spaces, which are ignored, so `LET number of apples = 5` is the same as `LET numberofapples = 5`

Official versions

- *4K BASIC* is the original ZX80 BASIC with integer-only arithmetic, by John Grant of Nine Tiles for the ZX80, so named for residing in 4 KiB read-only memory (ROM).
- *8K BASIC* is the ZX81 BASIC (also available as an upgrade for the ZX80[2]), updated with floating-point arithmetic by Steve Vickers, so named for residing in 8 KiB ROM.
- *48 BASIC* is the BASIC for the original 16/48 kB random-access memory (RAM) ZX Spectrum (and clones), with colour and more peripherals added by Steve Vickers and John Grant. It resides in 16 KiB ROM and began to be called 48 BASIC with the introduction of the ZX Spectrum 128 at which time the 16 kB Spectrum was no longer sold and most existing ones in use had been upgraded to 48 kB[67]
- *128 BASIC* is the BASIC for the ZX Spectrum 128.[68] It offers extra commands and uses letter-by-letter input.
- *+3 BASIC* is the BASIC with disk support for the ZX Spectrum +3.[67]
- *T/S 2000 BASIC* was used on the Spectrum-compatible Timex Sinclair 2068 (TS2068) and has the following six keywords and the ordinary Sinclair BASIC ones:
  - DELETE deletes BASIC program line ranges. CAPS SHIFT + 0 with the K cursor produces the command DELETE.
  - FREE is a function that gives the amount of free RAM. PRINT FREE will show how much RAM is free.
  - `ON ERR` is an error-handling function mostly used as `ON ERR GO TO` or `ON ERR CONT`.
  - RESET can be used to reset the behaviour of `ON ERR`. It was also intended to reset peripherals.
  - `SOUND` controls the AY-3-8192 sound chip.
  - STICK is a function that gives the position of the internal joystick (Timex Sinclair 2090).
- *BASIC64* by Timex of Portugal, is a software extension[69] to allow better Basic programming with the 512×192 graphic mode available only on Timex 2000 series computers. This extension adds commands and does a complete memory remap to avoid the system overwriting the extended screen memory area. Two versions exist due to different memory maps - a version for TC2048 and a version for TS/TC2068.

Other versions, extensions, derivatives and successors

Interpreters for the ZX Spectrum family

Several ZX Spectrum interpreters exist.[70]

- *Beta BASIC* by Dr. Andy Wright, was originally a BASIC extension, but became a full interpreter.
- *YS MegaBasic* by Mike Leaman.[70]
- *ZebraOS* by Zebra Systems in New York, a cartridge version of T/S 2000 BASIC that used the 512×192 screen mode.
- *Sea Change ROM (https://web.archive.org/web/20211213170025/https://web.archive.org/web/20150901085346/http://www.wearmouth.demon.co.uk/)* by Steve Vickers and Ian Logan, modified by Geoff Wearmouth, a replacement ROM with an enhanced Sinclair BASIC.[71]
- *Gosh Wonderful* by Geoff Wearmouth, a replacement ROM that fixes bugs and adds a tokenizer, stream lister, delete and renumber commands.[70][72]
- *OpenSE BASIC* (formerly SE BASIC) by Andrew Owen, a replacement ROM with bug fixes and many enhancements including ULAplus[73] support, published as open source in 2011[74][75]

Compilers for the ZX Spectrum family

Several ZX Spectrum compilers exist.[70]

- *HiSoft COLT Compiler* (a.k.a. HiSoft COLT Integer Compiler)[76]
- *HiSoft BASIC* (a.k.a. HiSoft BASIC Compiler), an integer and floating-point capable compiler[77]
- *Laser Compiler*[78]
- *Softek 'IS' Integer Compiler*[79] (successor to Softek Integer Compiler[80])
- *Softek 'FP' Full Compiler*[81]
- *ZIP Compiler*[82]

Derivatives and successors for other computers

- *SuperBASIC*, a much more advanced BASIC dialect introduced with the Sinclair QL personal computer, with some similarities to the earlier Sinclair BASICs
- *SAM Basic*, the BASIC on the SAM Coupé, generally considered a ZX Spectrum clone
- *ROMU6* by Cesar and Juan Hernandez - MSX[70]
- *Spectrum 48* by Whitby Computers - Commodore 64[70]
- *Sparky eSinclair BASIC* by Richard Kelsh, an operating system loosely based on ZX Spectrum BASIC - Zilog eZ80[83]
- *Sinbas* by Pavel Napravnik - DOS[70]
- *Basic*[84] (and CheckBasic[85]) by Philip Kendall - Unix
- *BINSIC*[86] by Adrian McMenamin, a reimplementation in Groovy closely modelled on ZX81 BASIC - Java

- *BASin*[87] by Paul Dunn, a complete Sinclair BASIC integrated development environment (IDE) based on a ZX Spectrum emulator[70] - Windows
- *SpecBAS*[88] (a.k.a. SpecOS) by Paul Dunn, an integrated development environment (IDE) providing an enhanced superset of Sinclair BASIC - Windows, Linux, Pandora, and Raspberry Pi
- *ZX-Basicus*[89] by Juan-Antonio Fernández-Madrigal, a synthesizer, analyzer, optimizer, interpreter and debugger of Sinclair BASIC 48K for PCs, freely downloadable for Linux and Windows.

See also

- List of computer system emulators § Sinclair ZX80
- List of computer system emulators § Sinclair ZX81
- List of computer system emulators § Sinclair ZX Spectrum and clones

Notes

1. Optional parameters are enclosed in [square brackets]
2. These assume the computer is in K (keyword) mode, which it normally is at the start of a line when entering BASIC. On the Spectrum 16K and 48K, Extended Mode is entered by pressing CAPS SHIFT and SYMBOL SHIFT simultaneously rather than the EXTENDED MODE key that is present on the Spectrum+ and later models.
3. “False” in Sinclair BASIC equates to 0 (zero), everything else equates to “true”. Functions that return true-or-false values thus actually return 0 for false and 1 for true, while `AND` usually returns the first of the conditions supplied for true, or 1 if no numerical values were given. For example, `6 AND 7` returns 6, while `NOT 6=7` returns 1.
4. The available numbers for colours are:

1	2	3	4	5	6	7	0
blue	**red**	**magenta**	**green**	**cyan**	**yellow**		**black**

   In all colour-related commands, the number 8 may be used to indicate “transparent” while in `INK` and `PAPER` may also be set to 9 for “contrast” — that is, to put a dark colour on a light background or vice versa automatically.

5. `INK`, `PAPER`, `FLASH`, `BRIGHT`, `OVER` and `INVERSE` set attributes for outputting text and graphics to the screen. They can be used either as commands, to apply to all subsequent output until set again, or within a `PRINT` statement, to apply only from that point until the end of the statement.
6. `CAT`, `ERASE`, `FORMAT` and `MOVE` were originally designed to be used with peripherals, but at the launch of ZX Spectrum, they had not been completely implemented, such that their use generated an error message (Invalid Stream). Later with the aid of the ZX Interface 1 shadow ROM, they were used for the ZX Microdrive. (The shadow ROM was paged when the BASIC interpreter detected a syntax error, which is why most ZX Microdrive commands use a `"*"`).
7. Unlike many other BASIC dialects, Sinclair Basic did not include the ELSE operator in the `IF–THEN[–ELSE]` clause. A workaround would be to use an `IF–THEN–GO TO` construct instead, bypassing the lines that would have been in an ELSE clause with the `GO TO`
8. The Spectrum’s display memory starts at address 16384 and is 6912 bytes long
9. A string inside a string must have its quotes doubled in Sinclair BASIC

References

1. Garfield, Simon (2010-02-28). "Sir Clive Sinclair: "I don't use a computer at all" " (https://web.archive.org/web/20211213170025/https://www.theguardian.com/technology/2010/feb/28/clive-sinclair-interview-simon-garfield). *The Guardian*. Retrieved 2011-05-23. "He is keen to credit [...], not least Nine Tiles, the company that made the Basic operating software."
2. "ZX80 - 8K BASIC ROM Upgrade" (https://web.archive.org/web/20211213170025/http://www.fruitcake.plus.com/Sinclair/ZX80/ROMUpgrade/ZX80_ROMUpgrade.htm). *www.fruitcake.plus.com*.
3. Vickers 1983, p. 59.
4. Vickers 1983, p. 70.
5. Vickers 1983, p. 85.
6. Vickers 1983, p. 69–70.
7. Vickers 1983, p. 101.
8. Vickers 1983, p. 116.
9. Vickers 1983, p. 135.
10. Vickers 1983, p. 93.
11. Vickers 1983, p. 113.
12. Vickers 1983, p. 110–111.
13. Cambridge Communication 1983, p. 15.
14. Vickers 1983, p. 91.
15. Vickers 1983, p. 123.
16. Vickers 1983, p. 103.
17. "World of Spectrum - Documentation - ZX Spectrum manual - Chapter 24" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/ZXBasicManual/zxmanchap24.html). *worldofspectrum.org*.
18. Vickers 1983, p. 168.
19. Vickers 1983, p. 19.
20. Vickers 1983, p. 151.
21. Vickers 1983, p. 68.
22. Vickers 1983, p. 41.
23. Vickers 1983, p. 142.
24. Vickers 1983, p. 60.
25. Vickers 1983, p. 79–81.
26. Vickers 1983, p. 11–123.
27. Cambridge Communication 1983, p. 18.
28. Vickers 1983, p. 66.
29. Cambridge Communication 1983, p. 19.
30. Vickers 1983, p. 31.
31. Vickers 1983, p. 32.
32. Vickers 1983, p. 37.
33. Vickers 1983, p. 25.
34. Vickers 1983, p. 159.
35. Vickers 1983, p. 110.
36. Vickers 1983, p. 131.
37. Vickers 1983, p. 16.
38. Vickers 1983, p. 112.
39. Vickers 1983, p. 57.

40. Vickers 1983, p. 13.
41. Vickers 1983, p. 144.
42. Vickers 1983, p. 15.
43. Vickers 1983, p. 67.
44. Vickers 1983, p. 142–143.
45. Vickers 1983, p. 147.
46. Cambridge Communication 1983, p. 39.
47. Vickers 1983, p. 31–32.
48. Cambridge Communication 1983, p. 22.
49. Vickers 1983, p. 129.
50. Vickers 1983, p. 130.
51. Vickers 1983, p. 121.
52. Vickers 1983, p. 163.
53. Vickers 1983, p. 74.
54. Vickers 1983, p. 42.
55. Vickers 1983, p. 73.
56. Vickers 1983, p. 14.
57. Vickers 1983, p. 143.
58. Vickers 1983, p. 58.
59. Vickers 1983, p. 69.
60. Vickers 1983, p. 32, 51.
61. Vickers 1983, p. 180.
62. Vickers 1983.
63. Vickers 1983, p. 141.
64. Vickers 1983, p. 7–8.
65. "Picture of ZX80" (https://web.archive.org/web/20211213170025/https://upload.wikimedia.org/wikipedia/commons/5/54/Sinclair_ZX80_%281980%29_-_Computer_History_Museum.jpg).
66. "Spectrum 128 ROM Disassembly - Spanish Spectrum 128" (https://web.archive.org/web/20211213170025/http://www.fruitcake.plus.com/Sinclair/Spectrum128/ROMDisassembly/Spectrum128ROMDisassembly3.htm). *www.fruitcake.plus.com*.
67. "World of Spectrum - Documentation - ZX Spectrum +3 - Chapter 7" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/ZXSpectrum128+3Manual/chapter7.html). *worldofspectrum.org*.
68. "World of Spectrum - Documentation - ZX Spectrum 128 Manual Page 6" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/ZXSpectrum128Manual/sp128p06.html). *worldofspectrum.org*.
69. "Timex tech info - Basic 64 for TC2048" (https://web.archive.org/web/20211213170025/http://timex.comboios.info/tmxtechb64-2048.html). *timex.comboios.info*.
70. http://www.worldofspectrum.org/sinclairbasic/
71. "Sinclair BASIC history - Sinclair Wiki" (https://web.archive.org/web/20211213170025/https://sinclair.wiki.zxnet.co.uk/wiki/Sinclair_BASIC_history). *sinclair.wiki.zxnet.co.uk*.
72. "The Incomplete Spectrum ROM Assembly" (https://web.archive.org/web/20211213170025/https://web.archive.org/web/20150901085346/http://www.wearmouth.demon.co.uk). Archived from the original (https://web.archive.org/web/20211213170025/http://www.wearmouth.demon.co.uk/) on 2015-09-01.
73. "ULAplus" (https://web.archive.org/web/20211213170025/https://sites.google.com/site/ulaplus/). *sites.google.com*.

74. "ZX Interface 2 - SE BASIC (3rd Party ROM Cartridge)" (https://web.archive.org/web/20211213170025/http://www.fruitcake.plus.com/Sinclair/Interface2/Cartridges/Interface2_RC_New_3rdParty_SEBASIC.htm). *www.fruitcake.plus.com*.
75. "OpenSE BASIC" (https://web.archive.org/web/20211213170025/https://sourceforge.net/projects/sebasic/). *SourceForge*.
76. "World of Spectrum - HiSoft COLT Compiler" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
77. "World of Spectrum - HiSoft BASIC" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
78. "World of Spectrum - Laser Compiler" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
79. "World of Spectrum - Softek 'IS' BASIC Compiler" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
80. "World of Spectrum - Integer Compiler" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
81. "World of Spectrum - Softek 'FP' Full Compiler" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
82. "World of Spectrum - ZIP Compiler" (https://web.archive.org/web/20211213170025/https://worldofspectrum.org/software). *World of Spectrum*.
83. "Sparky eZX BASIC Project" (https://web.archive.org/web/20211213170025/http://rk-internet.com/eZXSparky/). *rk-internet.com*.
84. "Philip Kendall - Basic" (https://web.archive.org/web/20211213170025/http://www.shadowmagic.org.uk/spectrum/basic.html). *www.shadowmagic.org.uk*.
85. "Philip Kendall - CheckBasic" (https://web.archive.org/web/20211213170025/http://www.shadowmagic.org.uk/spectrum/checkbasic.html). *www.shadowmagic.org.uk*.
86. "Binsic Is Not Sinclair Instruction Code" (https://web.archive.org/web/20211213170025/https://cartesianproduct.wordpress.com/binsic-is-not-sinclair-instruction-code/). June 25, 2012.
87. "ZX Spin and BASin - ULAplus" (https://web.archive.org/web/20211213170025/https://sites.google.com/site/ulaplus/home/zx-spin-and-basin). *sites.google.com*.
88. "ZXDunny/SpecBAS" (https://web.archive.org/web/20211213170025/https://github.com/ZXDunny/SpecBAS). June 5, 2021 – via GitHub.
89. "ZX-Basicus: analyzer/synthesizer/optimizer/interpreter of Sinclair BASIC programs for the ZX Spectrum 48K" (https://web.archive.org/web/20211213170025/https://jafma.net/software/zxbasicus/). *jafma.net*.

Bibliography

- Ardley, Neil (1984). *Sinclair ZX Spectrum+ User Guide*. Dorling Kindersley in association with Sinclair Research. ISBN 0-86318-080-9.
- Vickers, Steven (1982). *Sinclair ZX Spectrum BASIC Programming*. Sinclair Research.
- Vickers, Steven (1983). *Sinclair ZX Spectrum BASIC Programming* (2 ed.). Sinclair Research.
- Cambridge Communication (1983). *Sinclair ZX Spectrum Microdrive and Interface 1 manual*. Sinclair Research.

External links

- Sinclair ZX Spectrum BASIC Programming (https://web.archive.org/web/20211213170025/http://www.worldofspectrum.org/ZXBasicManual/): The original 1982 manual by Steven Vickers (referenced above)

- Sinclair ZX81 Basic Programming (https://web.archive.org/web/20211213170025/http://www.worldofspectrum.org/ZX81BasicProgramming/): also by Vickers
- The History of Sinclair BASIC (https://web.archive.org/web/20211213170025/http://scratchpad.wikia.com/wiki/Sinclair_BASIC_History): By Andrew Owen
- Timex Computer World (https://web.archive.org/web/20211213170025/http://timex.comboios.info/tmxtechb64-2048.html): Basic 64 user manual for Timex Computer 2048
- Sinclair BASIC grammar (https://web.archive.org/web/20211213170025/http://jafma.net/software/ll1grammar/index.htm): A LL(1) grammar specification for parsing Sinclair BASIC 16/48K