

# Zxb

**Note:** This section does not explain the [ZX BASIC language](#), but the usage of its compiler: ZXB.

## Introduction

ZXB is the main SDK executable. It can act both as a compiler or as a translator:

- When used as a *compiler* (this is the default behavior) it will convert a `.BAS` text file to a binary `.BIN` or `.TZX` file you can later run on your Spectrum or in a ZX Spectrum emulator.
- If invoked as a *translator* it will convert a `.BAS` file to assembler (`.ASM` source file). You can alter edit this assembler text file (for example to perform some low-level modifications or just to see how the compiler does it work!).

## Using ZXB

ZXB is invoked from the command line as `zxb.py` if you used the *Multiplatform* (.zip) distribution or `zxb` if you installed the .MSI package.

Using ZXB is quite straightforward. You will need to type in a BASIC source in a text file. If you don't have any, create a file named *helloworld.bas* using your favorite text editor, and type in the following:

```
10 PRINT "HELLO WORLD"
```

Save this text file as `helloworld.bas`. Now let's compile it:

```
zxb.py helloworld.bas
```

**NOTE:** In Windows you should omit the trailing `.py` extension. So type `zxb helloworld.bas` instead.

If everything went ok, no message should appear on the screen. Now, if you examine your directory, you will see a `helloworld.bin` binary file. This file contains the bytes of your program.

Another supported output formats are `.TAP` and `.TZX`. These formats are supported by many emulators, and can also be converted into sound (WAV, MP3, VOC) to be later loaded on a real ZX Spectrum. TZX and TAP files can also contain a BASIC loader which will load the binary code and execute it. Let's use all of this together:

```
zxb.py helloworld.bas --tzx --BASIC --autorun
```

This will create a .tzx file. Open it with your preferred emulator, and type *LOAD ""*. You will see a BASIC loader program executing and loading your code. The machine code is finally executed using *RANDOMIZE USR 32768*.

**NOTE:** 32768 (8000h) is the default ORG for your program.

You can change the default origin using the -S command line parameter.

## Command Line Options

ZXB provides several (and useful) command line options. To see them, just type ***zxb.py -h***, which outputs:

```
usage: zxb [-h] [-d] [-O OPTIMIZE] [-o OUTPUT_FILE] [-T] [-t] [-B] [-a] [-A]
          [-S ORG] [-e STDERR] [--array-base ARRAY_BASE]
          [--string-base STRING_BASE] [-Z] [-H HEAP_SIZE] [--debug-memory]
          [--debug-array] [--strict-bool] [--enable-break] [-E] [--explicit]
          [-D DEFINES] [-M MEMORY_MAP] [-i] [-I INCLUDE_PATH] [--strict]
          [--version]
          PROGRAM
```

positional arguments:

PROGRAM                   BASIC program **file**

optional arguments:

```
-h, --help                   show this help message and exit
-d, --debug                 Enable verbosity/debugging output. Additional -d
                             increase verbosity/debug level

-O OPTIMIZE, --optimize OPTIMIZE
                             Sets optimization level. 0 = None (default level is 2)
-o OUTPUT_FILE, --output OUTPUT_FILE
                             Sets output file. Default is input filename with .bin
                             extension
-T, --tzx                   Sets output format to tzx (default is .bin)
-t, --tap                   Sets output format to tap (default is .bin)
-B, --BASIC                 Creates a BASIC loader which loads the rest of the
                             CODE. Requires -T or -t
-a, --autorun               Sets the program to be run once loaded
-A, --asm                   Sets output format to asm
-S ORG, --org ORG           Start of machine code. By default 32768
-e STDERR, --errmsg STDERR
                             Error messages file (standard error console by
                             default)

--array-base ARRAY_BASE
                             Default lower index for arrays (0 by default)
--string-base STRING_BASE
                             Default lower index for strings (0 by default)
-Z, --sinclair              Enable by default some more original ZX Spectrum
                             Sinclair BASIC features: ATTR, SCREEN$, POINT
-H HEAP_SIZE, --heap-size HEAP_SIZE
                             Sets heap size in bytes (default 4768 bytes)
--debug-memory              Enables out-of-memory debug
--debug-array               Enables array boundary checking
--strict-bool               Enforce boolean values to be 0 or 1
--enable-break              Enables program execution BREAK detection
-E, --emit-backend          Emits backend code instead of ASM or binary
--explicit                  Requires all variables and functions to be declared
                             before used
-D DEFINES, --define DEFINES
                             Defines de given macro. Eg. -D MYDEBUG or -D
                             NAME=Value
-M MEMORY_MAP, --mmap MEMORY_MAP
                             Generate label memory map
-i, --ignore-case           Ignore case. Makes variable names are case insensitive
-I INCLUDE_PATH, --include-path INCLUDE_PATH
                             Add colon separated list of directories to add to
                             include path. e.g. -I dir1:dir2
--strict                    Enables strict mode. Force explicit type declaration
--version                   show program's version number and exit
```

Some options (-h, --version) are quite obvious. Let's focus on the rest:

- **-d** or **-debug**

This will show lots of (useless) debug information for the compiler developer (e.g. to me). Usually, you won't need this at all.

- **-O** or **--optimize**

The default optimization level is 1. Setting this to a value greater than 1 will enable the compiler code optimizations (e.g. Peephole optimizer). Setting this to 0 will produce slower

code, but could be useful for debugging purposes (both for the compiler or the BASIC program). A value of 3 will enable **aggressive** optimizations not fully tested yet! So, beware!

- **-o or --output**

Sets the output file name. By default it will be the same as the input file, but with the extension changed as appropriated (.BIN, .TAP, .ASM, .TZX).

- **-T or --tzx**

Outputs the binary result in [TZX Format](#). This format can be converted to sound (.WAV or .MP3).

- **-t or --tap**

Outputs the binary result in [TAP Format](#).

- **-B or --BASIC**

This is a very useful option. It will prepend a ZX spectrum BASIC loader that will load the rest of your binary compiled program. This option requires the **--tap** or **--tzx** to be specified. This way you can type `LOAD ""` to load your program.

- **-a or --autorun**

Makes your program to run automatically. If specified with the `-B` or `--basic` option, your program will autorun if loaded with `LOAD ""`. If `--basic` is no used this option is ignored.

- **-A or --asm**

This will create the .asm (assembler) file. Useful to see / edit the assembler code. You could later assemble it using [ZXbasm](#) included assembler.

- **-S or --org**

This will change the default machine code ORiGin. By default your code will start at memory position 32768 (8000h). But you can change this with this parameter.

- **-e or --stderr**

This specifies an output file name for error msgs. This is useful if you want to capture compilation error messages (for example, to call ZX BASIC compiler from within an [IDE](#)).


- **--array-base**

Unlike original Sinclair BASIC, array indexes starts from 0, not from 1 (see [DIM](#)). You can change this behavior. For example setting `--array-base=1` will make array indexes start from 1 (like in Sinclair BASIC). This option (array-base=1) is active when `--sinclair` compatibility flag is specified.

- **--string-base**

Unlike original Sinclair BASIC, string indexes starts from 0, not from 1. That is, in ZX BASIC, `a$(1)` is the 2nd letter in `a$` string variable, and `a$(0)` the 1st. You can change this behavior, setting which position in the string refers to the 1st letter. For example, setting `--string-base=1` will make strings start from 1 (like in Sinclair BASIC). This option (string-base=1) is active when `--sinclair` compatibility flag is specified.

- **-Z or --sinclair**

Tries to make ZX BASIC as much compatible as possible with Sinclair BASIC (ej. Arrays and Strings start at position 1). Also includes some external functions like `POINT`, `ATTR` and `SCREEN$`, not available by default (they are in external libraries).  [v: latest](#) ▼

- **-H or --heap-size**

Set the size of the heap. Default heap size is above 4K (4768 bytes exactly). The heap is a memory zone used to store and manipulate strings (and other dynamic size objects where available). If you don't make use of strings, you can get back part of the heap memory. You might also need more heap space, so set it with this flag. The heap zone comes at the end of your program, and its size is fixed (won't change during program execution).

- **--debug-memory**

During your program execution, using strings might fail due to lack of memory, but your program won't report it and will continue executing (except the strings not fitting into the heap will be converted to `NULL` string or `""`). The same applies to other dynamic objects. So enabling this flag, will make your program to stop reporting a ROM *Out of memory* error. This will add a little overhead to your program execution, but it's useful to detect *Out of Memory* errors.

- **--debug-array**

As above, using wrong subscript (out of range) in arrays won't trigger an error. Setting this flag will raise ROM error Subscript out of Range. This flag will add a little overhead to your program execution, but it's useful to detect Out of Range errors.

- **--strict-bool**

By default, ZX BASIC will treat boolean values as 0 = False, Any other value = True. Some programmers expect TRUE = 1 always. Using this option will enforce boolean results to be always 0 or 1. Using this option might add a little overhead to your program. Using `--sinclair` option will also enable this feature.

- **--enable-break**

Unlike Sinclair BASIC, Your program, being converted to machine code, won't be affected by BREAK. If you enable this flag, you will be able to stop your program pressing BREAK (Shift + SPACE). The ROM Break message will also report the line in which the program was interrupted. This option will add some overhead to your program.

- **--explicit**

Requires all variables to be declared with DIM before being used. This is something similar to Sinclair BASIC, in which when you tried to read a variable not previously set, a "Variable not Found" error was triggered. This option is really useful and you should enable it for large programs.

- **--strict**

Requires all variables (and parameters and functions!) to have an explicit type declared (e.g. `UInteger`). Otherwise, forgetting a type will cause an error and the program won't compile. This is very useful to avoid forgetting type declarations. When the type is explicitly declared the compiler can make better assumptions and further error checking and optimizations.

This is all you need to know to use the compiler. Proceed to the [ZX BASIC](#) page for a language reference.