FASTBASIC v4.3

for Atari 8-Bit Computers

by Daniel Serpell

Cross Compiler Version

FastBasic Cross Compiler

This is the FastBasic cross compiler. It takes a basic sources and compiles to an assembly file for CC65.

Requisites

To use, you need CC65 (preferable the 2.16 version or newer) installed, the scripts assume that it's available, in the path (on Linux / macOS) or in the "C:\cc65" folder (on Windows).

Download CC65 from http://cc65.github.io/cc65/getting-started.html

Linux & macOS installation

Install CC65 and place it into the PATH. Extract the FastBasic compiler on any folder.

Windows installation

Install CC65 to the "C:\cc65" path and then extract the FastBasic compiler, it is recommended to place it in the"C:\cc65\fb" folder.

If you need a different path for CC65 you will need to alter the "fb.bat" and "fb-int.bat" files to specify the correct path at the top.

Basic Usage

For simple compilation of BAS files to XEX (Atari DOS executable), use the included "fb" and "fb-int" scripts.

- On Linux:

  ```
  /path/to/fb myprog.bas
  ```

  or:

  ```
  /path/to/fb-int myprog.bas
  ```

- On Windows:

  ```
  C:\cc65\fb\fb myprog.bas
  ```

  or:

```
C:\cc65\fb\fb-int myprog.bas
```

There are two compilers, one for the full version “fastbasic-fp”, used with the “fb” script, and another for the integer only version “fastbasic-int”, used with the “fb-int” script. The advantage of the integer only version is that it produces smaller executables.

The script generates three files from the basic source:

- XEX file, standard Atari 8-bit executable.
- ASM file, the assembly source.
- LBL file, a list of labels, useful for debugging. This file includes a label for each line number in the basic source.

The compilation is a two step process:

- The included compiler reads the basic source and produces an assembly file:

  ```
  fastbasic-fp myprog.bas myprog.asm
  ```

- The CL65 tool is used to assemble and link with the runtime library.

  ```
  cl65 -t atari -C /path/to/fastbasic.cfg myprog.asm -o myprog.xex /
      path/to/fastbasic-fp.lib
  ```

Advanced Usage

Passing options to the compiler

The compiler scripts `fb` and `fb-int` allows passing options to the compiler, allowed options are:

- **-v**
  Shows the compiler version.
- **-n**
  Disable all the optimizations, the produced code will be the same as the native IDE. This is useful to debug problems with the optimizations passes, should not be used normally.
- **-prof**
  Helps profiling the compiler generated code. Outputs statistics of the most used tokens and token pairs.

- **-d**
  Enable parser debug options. This is only useful to debug parser, as it writes the full parsing tree with all the tried constructs.
- **-h**
  Shows available compiler options.
- **-S**:*address*
  Sets the start address of the compiled binary. The default value is `$2000`, set in the configuration file `fastbasic.cfg`. You can specify a different address to allow for a bigger DOS, or to have more memory available if you don't use a DOS. The address can be specified as decimal or hexadecimal with `0x` at front.
- **-X**:*cl65-option*
  Passes the given option to the CL65 linker. See the CC65 documentation for valid options, some useful options are listed bellow:
  - **-X:–asm-include-dir -X**:*path*
    Adds a path to search for assembly included files, used in your custom ASM sources.
  - **-X:–asm-define-sym**=*symbol*
    Define an assembly symbol, used in custom ASM sources.

Linking other assembly files

The compiler support linking to external assembly modules, you can pass them to the "fb" command line:

```
fb myprog.bas myasm.asm
```

This will compile "myprog.bas" to "myprog.asm" and then assemble the two files together using cl65. You can pass multiple ".asm" (or ".o") files to the command line, but only one basic file.

From the FastBasic source, you can reference any symbol via "@name", for example:

```
' FastBasic code
'
? "External USR sample:"
? USR( @Add_10, 25 )
```

The ASM file must export the `ADD_10` (always uppercase) symbol, for example:

```
; Assembly module
  .export ADD_10
```

```

.proc ADD_10
  pla   ; Parameters are pased over the stack, in reverse order
  tax
  pla
  clc
  adc #10
  bcc no
  inx
no:
  ; Return value in A/X registers
  rts
```

You can also export ZP symbols, to import them use “@@name”.