# Wanna Be Startin' Somethin'

April 5th, 2016

**Please complete this exercise before the first lab!**

If you already have advanced knowledge of the C programming language, you may skip this exercise. In this (pre-)exercise you will set up a toolchain for compiling C applications for your PC. Moreover, in a small example we will show important language features which you have to understand before you should proceed to microcontroller programming.

Here are the main benefits of working with a native compiler (for your PC):

- It is a good starting point to learn C.
- You can easily test parts of the code, testing and debugging on the microcontroller is challenging.
- You can write entire libraries and port them to the AVR, e.g. an implementation of a list or an interpolation routine.

## Task 0.1 : C/C++ GNU Toolchain

If you are using Linux, you probably have a GNU C Compiler installed and thus you can skip this task. The following instruction is for the Windows programmers among you only.

First, you have to install a C toolchain (including compiler and libraries) to be able to program in C. In this lecture we use MinGW. MinGW, a contraction of "Minimalist GNU for Windows", is a minimalist development environment for native Microsoft Windows applications.

Open *http://sourceforge.net/projects/mingw/files/* and click on *Installer*, *mingw-get-inst*, and *mingw-get-inst-** (select latest version). Download and execute the file *mingw-get-inst-*.exe*. Follow the installation, use default settings, don't use an installation path with whitespaces! As Compiler suite use the C Compiler (if you want you can also install C++). Finish the installation.

It is necessary to add the folder of MinGW's bin directory to the *Path* system variable of Windows. Go to *System Properties->Advanced->Environment Variables...*. Add *MINGW_ROOT\bin* (e.g. *C:\MinGW\bin*) to the Path system variable. Use ; as delimitter.

## Task 0.2 : Eclipse

(You can skip this section, if you have already installed Eclipse with CDT plugin.)

From the Eclipse homepage (*http://www.eclipse.org/downloads/*) download *Eclipse IDE for C/C++ Developers* for your operating system. Note that Eclipse requires an installed Java JRE. Afterwards you can extract (unzip) the downloaded file into a folder of your choice. Eclipse can be started by calling the file *eclipse*, which can be found in the main directory of the program.

## Task 0.3 : !!!Hello World!!!

Now let's finally start coding. Start Eclipse and select a workspace directory in which you want to store your projects. Press *File->new->Project...* and select *C/C++->C Project*. Use the template *Executable*

*Hello World ANSI C Project* and select MinGW GCC or Linux GCC as toolchain (if this is not possible, something went wrong during the installation). Open the source file (ending *.c ) that is located in the folder *src*. By pressing Ctrl + b the source code is compiled. If you make changes in the file do not forget to save it before you compile it. To run the program press the green play button in the toolbar and select Local *C/C++ application*. On the bottom the output of the execution is shown (!!!Hello World!!!).

Now, copy the content of the file *sheet0.c* into the existing source file (completely delete the existing code). Try to understand the program! You may also like to play around with it.

## Task 0.4 : A Bit of Masking and Shifting

Bit operations are performed in oder to access various internal components and peripherals of the microcontroller. To get familiar with the use of them, solve the following tasks by writing to and reading from a variabe `uint8_t var` with one C statement. You should do this by hand, afterwards you can check you solution by writing a small test program in C.

- Set bit 3
- Set bits 4 and 6 (with / without bit shifting)
- Clear bit 2
- Clear bits 2 and 7
- Toggle (invert) bit 3
- Set bit 2 and clear bits 5 and 7 at the same time
- Swap bits 3-5 and bits 0-2

For each operation, ensure that the remaining bits of `var` are not changed!

Note that the least significant bit (LSB) is bit 0 and most significant bit (MSB) is bit 7. Use the following bit operators:

| | | | | | |
|---|---|---|---|---|---|
| `&` | bitwise AND | `\|` | bitwise OR | `^` | bitwise XOR |
| `<<` | left shift | `>>` | right shift | `~` | one's complement |

## Task 0.5 : More Fun with Bits

Write a function that reverses the bit order of an 8 bit (use `uint8_t`) input! Given the input's bit representation `input`$= (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$ the output becomes `output`$= (b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7)$.