Compile guide for C programs using GCC.

Daniel Giovanni Martínez Sandoval
dagmtzs@gmail.com

September 2024

Contents

1 Introduction

For easier recognition, filenames as well as file paths will be colored in green and links to webpages will be colored in blue.

2 Requirements

This guide is made for laptops running Windows 10 (and later). Minimum requirements have not been defined but almost any modern system (probably from 2015 and on) in good conditions should work just fine. For the hardware, the following is needed:

- ...
- ...
- ...

3 Installing software tools

To set up a basic development environment, I would suggest having the following software tools:

- A text editor, preferably, one intended for software development in C language.
- A compiler, and its related tools and libraries.
- A programmer.

My personal preference in a Windows machine is *VS Code* for the text editor, and the tools that I would recommend are the *AVR 8-bit GNU Toolchain* and *AVRDUDE* as the programmer.

3.1 VisualStudio Code

Download it from this link and follow the installer instructions.

3.2 Environment Variables

To be able to use the compiler and programmer you just installed, you need to update you Environment Variables in the following manner:

1. Click the Windows menu (or press the Windows key in your keyboard) and type *variables*.
2. One of the results should be *Edit the system environment variables*, click it.
3. In the new window, titled *System Properties*, click the button *Environment Variables*, almost at the bottom of the window.
4. In the next window, you should see two fields, one for *User variables for ...* and one for *System variables*. Under *User variables for ...*, look for the variable called **Path**.
   (a) If you don't have a **Path** variable, click *New...*
      i. In the *New User Variable* window, under *Variable name*, write **Path**
      ii. Under *Variable value*, write C:\Program Files\avr8-gnu-toolchain\bin.
      iii. Click *OK*.
   (b) If you already have a **Path** variable, select it and click *Edit...*.
      i. In the new window, click *New*.
      ii. In the active space, write C:\Program Files\avr8-gnu-toolchain\bin.
5. Once the AVR Toolchain folder has been added to the **Path** variable, repeat the step 4b to add the AVRDUDE folder so your **Path** has these two directories added:
   - C:\Program Files\avr8-gnu-toolchain\bin
   - C:\Program Files\avrdude\

3.3 Additional configuration

In order to make the development workflow a smoother experience, I'd recommend following the instructions from this article to add some very useful functionalities to VS Code.

4 Testing the setup

Once the environment is set up, you can follow these steps to create, compile and upload a program to your microcontroller:

1. Create a folder to store your AVR projects, and inside it, create a folder for this example program.
2. Open VS Code and open the folder you just created. (File -> Open Folder).
3. In the VS Code file explorer, create a new file, name it: `main.c`.
4. From my GitHub page, open my HelloWorld example following this link.
5. Copy my code and paste it into your newly created file in VS Code.
6. Save the file.
7. Connect the USBasp to your microcontroller (and to your PC if it is not connected already).

8. In the VS Code file explorer, right-click the folder you are working in and select the option *Open in Integrated Terminal*.
9. In the Integrated Terminal, execute the following commands:
    - `avr-gcc main.c -mmcu=atmega328p -o main.elf`
    - `avr-objcopy -O ihex main.elf main.hex`
    - `avrdude -p atmega328p -c usbasp -U flash:w:main.hex`

The result should be your microcontroller programmed with my code.

4.1 Troubleshooting

Here are some things that can make this process fail:

- Your system might not recognize the commands you write immediately, you can try restarting programs, logging out and back in to your Widows user account or if that doesn't work, you might restart your computer.
- Typos might prevent your commands from working, or even might cause damage to your microcontroller, double check the commands you write.
- One of the most common problems are wires and connections. When connecting each signal manually:
    - Take special care in labeling or color-coding your long wires so you don't mix them up.
    - Make sure you identify correctly the pins of your microcontroller.
    - Check the values of your crystal oscillator and capacitors.
    - Check specially the power connections to the programmer, and make sure they are close to the microcontroller, i.e., try to minimize the length of the power lines to the microcontroller power pins.
    - Sometimes the uploading speed might be too high for certain setups, so you can append to the `avrdude` command the option `-B125kHz`, or even `-B32kHz`.

5 Useful links

- AVRDUDE User Manual
- AVR Libc Home Page
- avr-gcc Toolchain Wiki
- AVRDUDE Home Page
- GCC Online Documentation