C Programming in AVR Studio using WinAVR

Sepehr Naimi (BIHE)

Introduction

This tutorial will teach you how to write, compile, and trace C programs in AVR Studio. For more information, you can see AVR Studio's Help and see the WinAVR site.

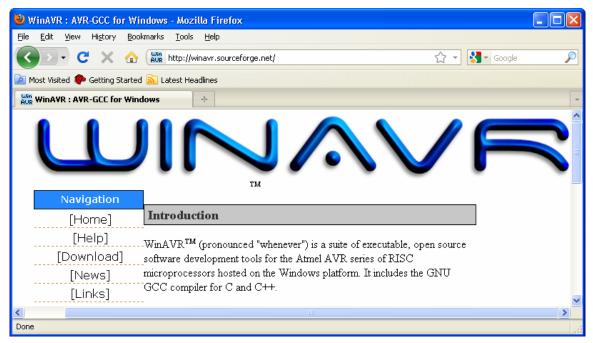


Figure 1: WinAVR website

Downloading and Installing AVR Studio

1. First download *AVR Studio*. You can download the last version of the AVR Studio IDE from the following web address: http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=2725

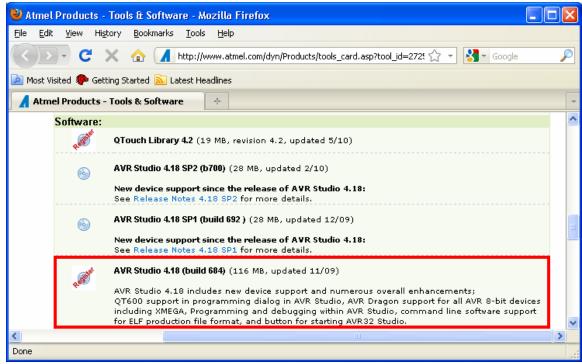


Figure 2: Downloading the Last Version of AVR Studio

2. Execute the downloaded setup file. The installation of the software is straight forward, and it can be done by pressing the Next and Install buttons a few times. Now, the AVR Studio IDE is ready to be used.

Downloading and installing WinAVR

3. Download the *WinAVR* software from the following site: http://sourceforge.net/projects/winavr/files/

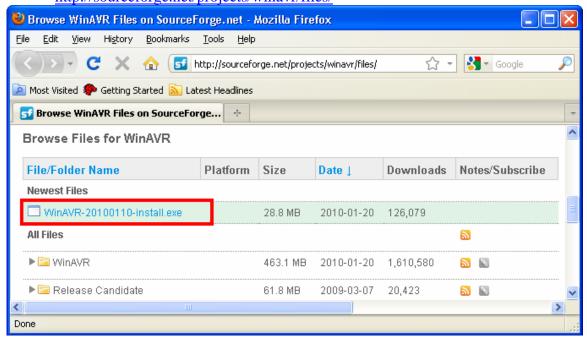


Figure 3: Downloading the Last Version of WinAVR

4. Install the *WinAVR* software by executing the downloaded file.

Opening AVR Studio

5. Open the *AVR Studio* from the start menu.

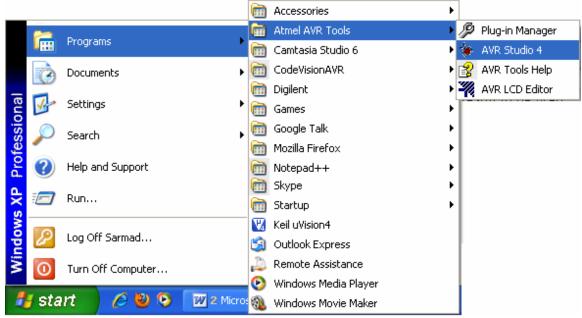


Figure 4: Running AVR Studio

Creating a new Project

6. Click on the *New Project* button.



Figure 5: Welcome window

7. Do the followings:

- In the left list, choose **AVR GCC**.
- Choose a name for your project (e.g. firstProgram) by typing the name in the text field below the *Project name*.
- You can change the location where the files of the project will be saved by clicking on the ... button, if you want.
- Press the *Next* button.

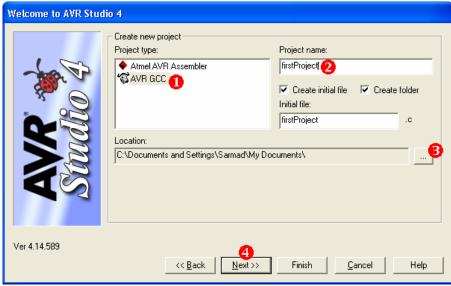


Figure 6: Name the project

8. Choose *AVR Simulator* from the left list and *ATmega32* from the right list and press *Finish*.

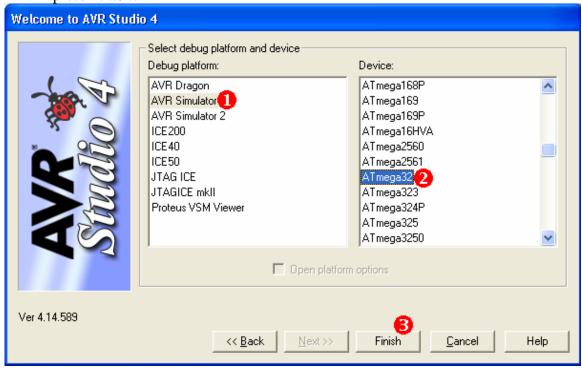


Figure 7: Choosing the microcontroller

Writing the first program

9. Type or copy the following program:

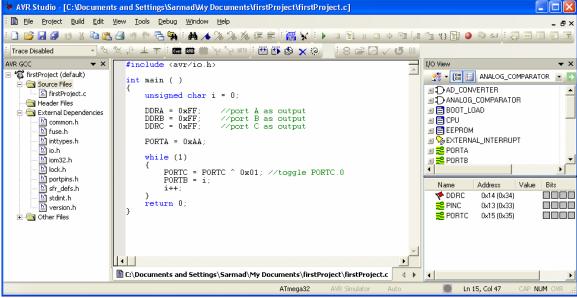


Figure 8: Writing a program

Saving

10. Save the program by pressing Ctrl + S or choosing Save from the File menu

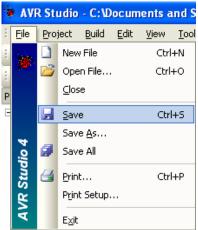


Figure 9: The File menu

Compiling

11. Press *F7* or select *Build* from the *Build* menu or click on the build icon in the toolbar.

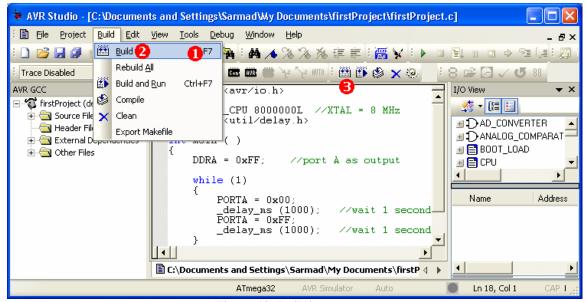


Figure 10: Building

12. See the *Build* window. The window shows if your program has syntax error or not. By looking at the window, you can see the amount of memory which is used by your program, as well.

Debugging

13. Select *Start Debugging* from the *Debug* menu. A yellow arrow appears next to the first instruction of the program and shows the next instruction which will be executed.

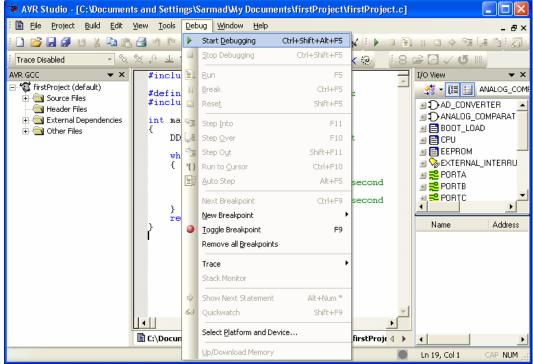


Figure 11: Debug menu

Tracing

- 14. To execute the next instruction press *F10* or select the *Step Over* from the *Debug* menu. There are also another tracing tools in the *Debug* menu, as well:
 - *Step Into*: It is executes the next instruction, like Step Over. The only difference between them is that, if the next instruction is a function call, the Step into, goes to the function; but Step Over executes the function completely and goes to the next instruction.
 - *Step Out*: If you are in a function, Step Out executes the program up to the end of the function.
 - For more information about the Tracing tools you can see the AVR Studio's help.

Watching

See Figure 12. In this part you learn to use the different tools to watch the program.

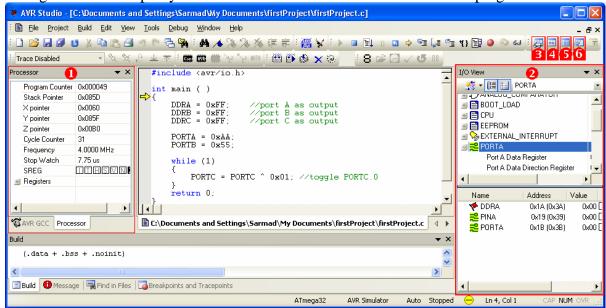


Figure 12: AVR Studio environment

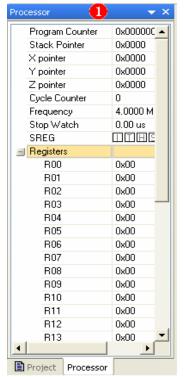


Figure 13: Processor

- **Processor:** It shows the contents of the registers which are related to the CPU: general purpose registers (R0 to R31), PC, SP (Stack Pointer), status Register (SREG), X, Y, and Z registers. See Figure 13. Cycle Counter counts the number of machine cycles which have been passed and the Stop Watch represents how much time has elapsed. You can use the parameters to measure the execution time of your program. You can reset them as well, by right clicking on them and choosing reset. (Figure 15)
- can see the value of the different I/O registers. See Figure 14; in the upper box, the related I/O registers are grouped. Click on *PORTC* and see the values of *PORTC* and *DDRC*.

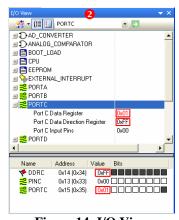


Figure 14: I/O View

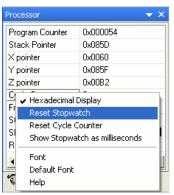


Figure 15: Reset Stopwatch

In Figure 12, you saw some icons in the toolbar which are numbered as 3, 4, and 5 (Figure 16). The use of them is discussed in this page.



Figure 16: Some watching tools

• Watch: Click on the tool which is numbered as 3 in Figure 16. The watch window appears (Figure 17); in this window you can see the value of different variables at the correct time. Double click under the *Name* title and type *i* and then press *Enter*; the value of the *i* will be displayed, and if you continue tracing by pressing the F10 button (Step Over) the changes of *i* will be displayed.

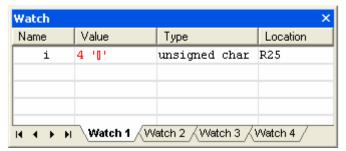


Figure 17: Watch

• **Register:** Click on the tool which is numbered as 4 in Figure 12, the *Register* window will be displayed. This window shows the contents of all of the general purpose registers, at the current time. You can close the window by clicking on the X which is displayed on the top right corner of the window.

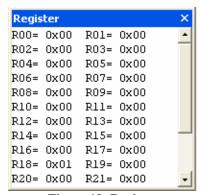


Figure 18: Register

- Memory: Click on the tool which is numbered as 5 in Figure 16. The Memory window appears (Figure 19); in this window you can see the contents of different locations of memory, at the correct time. In the window:
- 1) The gray column shows the address of the first location in each row. For example, in the picture, location \$60 contains 1, \$61 contains 2 and so on.

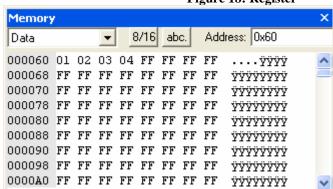


Figure 19: Memory

- 2) We can choose which of the memories to be displayed using the comboBox on the top left corner of the window:
- > Data: SRAM memory
- **EEPROM:** EEPROM
- > I/O: I/O registers
- > **Program:** Flash Memory
- > **Register:** general purpose registers

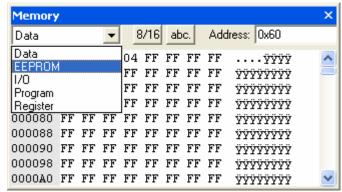


Figure 20: Memory

- **Disassembler:** This window shows the contents of the flash memory. In the window:
 - 1) The black texts display our program.
 - 2) Below each of the instructions of our program, its assembly equivalent is mentioned.
 - 3) The gray numbers at the middle of the lines mention the machine code of each instruction. For example, according to Figure 21, the machine equivalent of *OUT 0x17,R24* is *BB87*.
 - 4) The last column describes what the assembly instruction does. For example as you see in Figure 21, *OUT* is *Out to I/O location*, or *JMP* is *Jump*.
 - 5) The gray numbers at the beginning of each line, mention at which location of flash memory each of the instructions are located. For example, in Figure 21, "OUT 0x14,R24" is located in address 004C.
 - 6) The yellow arrow points to the next instruction which will be executed.

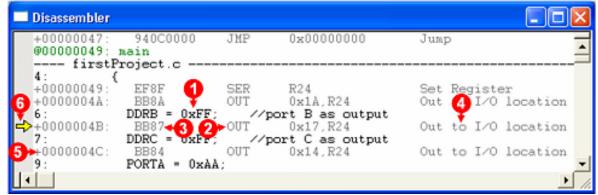


Figure 21: Disassembler