Lab 4: C Programming in Atmel Studio for the AVR Microcontroller

CompEng 3151: Digital Engineering Lab II

Last revised: July 16, 2019 (BJZ, RJS)

Description/Overview

Up until now, you have written programs for the AVR microcontroller in the Assembly language. Assembly is a low-level language that interacts closely with the hardware. All programs written in assembly are specialized to a set of hardware and thus they cannot generally be ported to other architectures without radical rewrites of the code. The other key to assembly programs is that you write every instruction as is (one mnemonic, one instruction to the processor), and there is no optimization of your code afterward.

Programming microcontrollers in the C language offers flexibility and portability when it comes to writing code. C programming often is less time consuming and much easier to write. In C (and other high-level languages) each instruction/line of code will result in maybe dozens of instructions to the processor. C code is also optimized in a two-step process by first compiling and then linking it with libraries and other object-files.

In this lab, you will get an introduction to programming the AVR Simon Board in C using Atmel Studio. After this lab, we will stick to programming in C as there are libraries and other functions that will make our interactions with peripherals in future labs much easier.

Objectives

Students will:

- Learn how to create an embedded C project for an AVR-based microcontroller
- Create a C program that uses pushbutton inputs to control LED outputs (accomplishes the same objective as in Lab 1)
- Compare the size and performance of the C program code with equivalent programs in Assembly.

Materials Required

- Atmel Studio 7 Installed on PC (ECE CLC Computers will have this)
- AVR Simon Board with a USB cable

Preparation

In order to be successful with this lab, students should review Chapter 7 in the Mazidi textbook.

Procedure

1) *Atmel Studio Tutorial*
    a. In order to become familiar with the basics of Atmel Studio 7 and coding in C, you should complete the C Programming in Atmel Studio 7 – Step by Step Tutorial.

2) *Pushbutton Control of LEDs*
    a. Up until now, we have statically programmed the LEDs to turn on and off based on switching PORT D on and off depending on what part of the code it is in. For this last part of the lab, you should choose four of the push buttons on the board and light up their respective LEDs when their button is pushed. The LEDs should remain lit for approx. 1 second before turning off.

3) 3-bit Multiplication
    a. In Lab 2, you implemented a 3-bit multiplier circuit on the AVR Simon Board. You are now going to implement this using C code. Remember to use an algorithm of successive additions to complete the multiplication. One issue you are going to encounter is manipulating the input data so that it gets into a form that can be multiplied and then manipulating it to display on the LEDs.
    b. One item of caution when using Atmel Studio to write C programs – the compiler will optimize out any variable or functions that are not used or that it finds redundant. For example, if you create a variable and want to watch its value while stepping through the program, you need to make sure that the variable is used to do something (output, arithmetic, etc.) or it will not be traceable. A method to turn off optimization is documented at the end of this lab.

Deliverables

For this lab, the three different programs you wrote for this lab will be submitted. Each of these programs should be thoroughly documented so that someone else who has not completed this activity can follow your code. Make sure each of your programs has an appropriate comment block at the top of the file (see MST Computer Science Department's Coding Standards). Also in your submission, you should provide detailed answers to the questions below.

Questions/Observations

1. You'll notice in the C program, there is a pre-defined function for delay. This function is based off the F_CPU definition at the top of your program that represents the frequency of the CPU on the AVR board. Why does the frequency of the CPU have to be known in order to implement this function?
2. Without a pre-define delay function, what would be another way you could create a delay using C code?

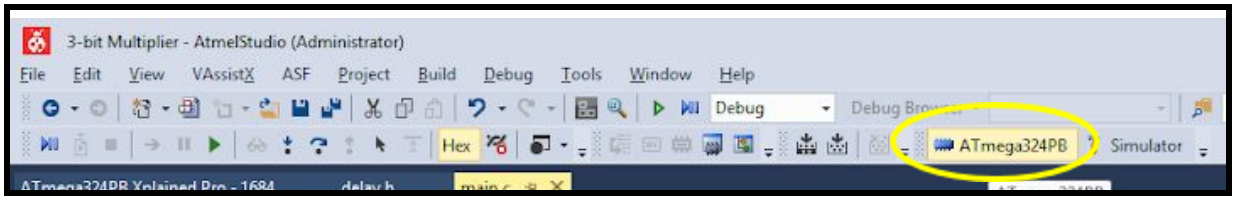
3. In comparing your assembly code from Labs 2 & 3 to the C code you created for this lab, which one did you find easier to put together? Why? Which programs (assembly or C) are larger? Why?

References

- Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi. 2010. *AVR Microcontroller and Embedded Systems: Using Assembly and C (1st ed.)*. Prentice Hall Press, Upper Saddle River, NJ, USA.

Optimization Settings

Click on the device in the Device and Debugger toolbar at the top.



Go to the Toolchain tab on the left and change the Optimization level.