ECE-3226/CSCI-3451 - Lab #5: LEDs, Switches, and Delay Loops

Contents:

Overview

Purpose: This lab introduces you to our hardware development platform, the STK500. You will learn create for and download programs to this platform. This platform includes a few peripherals that you will learn to use, including LEDs and switches, as well as the built-in peripherals on the ATmega32 like the timers. You will also learn how to use the JTAG mkII debugger, which enables you to step through the code while it is running on the STK500 platform (i.e. allows you to use and as you do in the simulator, but while the program is running on the hardware platform).

In particular, you will learn:

- how to connect the JTAG mkII to the STK500 and the computer and use it in AVR Studio
- how to download the program to the STK500
- how to configure and use the ports (e.g. Port B) either as inputs or outputs
- how to use LEDs and switches
- about switch/button bouncing, and how to correct it using switch debouncing
- how to create a simple delay loop
- some new assembly instructions:
    - o set register (SER) instruction
    - o set and clear bit in I/O register (SBI and CBI) instructions
    - o test and skip if bit in I/O register is set or clear (SBIS and SBIC) instructions
    - o input from and output to I/O register (IN and OUT) instructions

Pre-Lab:

Part2a , Part2d , Part2f , Part3a , Part3e

Lab Assignment

- **Part 1: The STK500 Starter Kit**

  First, let's introduce the STK500 Starter Kit. You will receive a box at the beginning of class that contains everything you need for using the STK500 hardware platform. Start by pulling the STK500 board, power supply, and serial cable out of the box. The STK500 board will be in a silver metallic bag/pouch (which is designed to protect the board from static electricity). Carefully pull the STK500 board out of the bag, then put the bag on the table and **set the board on the bag** (to continue to protect it from static electricity while you are using it).

  Notice that the board effectively has three parts. On the right side of the board are two serial ports and the port for the DC power supply, while the left side of the board has push buttons and LEDs. The middle of the board (the white area) contains the processor and the JTAG connector (discussed below). Hook up the power supply to this port. When you plug in the power supply, some LEDs may light up on the board. If not, switch the "POWER" switch, which is in the upper-right hand corner of the board (right about the DC power port). Once the board is powered, at least two LEDs will light up on the board. The first is the "POWER" LED, which is directly above that switch -- it is a red LED. The second is the "STATUS" LED, which is also on the right side of the board, but much lower, near the "PROGRAM" switch. The STATUS LED will light up as green or red, depending upon the status of the processor. It will be green when the processor is executing a program (or instruction) and red when it is not.

  Let's use the JTAG mkII in-circuit debugger (i.e. the little blue plastic box that is included in your kit). First connect the JTAG mkII to the computer using the USB cable. Then connect the JTAG mkII to the STK500 board by connecting one of the two 10-pin sockets of the JTAG mkII to the JTAG adapter on the STK500 board (which is directly above the processor on the board). You'll want to connect the 10-pin socket to the pins labelled "SCKT3100A3". Have the instructor verify your connection, so you know it's connected properly.

  To use the JTAG mkII to download and debug your code while it's running on the board, we need to appropriately select the platform in AVR Studio. When you originally created the project, you selected "AVR Simulator" as the platform (and "ATmega32" as the device). We now need to change the platform to "JTAGICE mkII". To do this, click on AVR Simulator on the upper right corner, a new window will appear, select the"JTAGICE mkII" .

  You are now ready to begin running programs on the STK500!

- **Part 2: Using LEDs and delay loops**

  a. Write a program to configure all bits of PORTB as outputs and control the LED lights on the STK500 board to display 0xC3. Keep in mind, the LEDs on the STK500 board are active low.

b. Use the provided connectors to connect PORTB to the LEDs of STK500.

c. Download and test it on the board.

d. Write a program that uses all bits of PORTB as outputs and **rotates** (to the left), lighting one LED at a time starting with LED0. In other words, LED0 will be lit in the first loop iteration, LED1 in the 2nd iteration, LED2 in the 3rd, and so on. After the last LED is lit, the loop will start over again.

e. Simulate it in AVR Studio. Download and test it on the board.
To download and run on the STK500:
Debug → Start Debugging and Break → Continue

Question 2_1: What is happening on the board? Why?

In the past, when we've encountered a programming problem with an AVR assembly program, we've been able to debug it by stepping through the code in the AVR Studio simulator until we find the problem. But now we've encountered a different problem; we verified the code in the simulator, but the code seems to be producing the incorrect results on the board. What now?

Start debugging. You should be able to step through the code while it is running on the STK500 board. Now, you should see much different results on the STK500 board.

f. Modify the program of **part-2d** and add a delay loop such that it rotates the lighted LED once every half a second (i.e. it will take the program 4 seconds to rotate through all 8 LEDs once).

g. Download the program and observe the execution of the program.

Question 2_2: Does the program work as expected? State your observations.

- **Part 3: Using Switches and delay loops**

a. Write a program to allow you to light any LED by pressing its corresponding switch (i.e. pressing SW1 will turn on LED1, pressing SW5 will turn on LED5, etc.). Configure all the bits of PORTB as outputs and all the bits of PORTD as inputs, assume PORTB bits are connected to LEDs and PORTD bits are connected to switches. Make sure that the pull-up resistors are activated for every bit of PORTD. Essentially the program will be reading the status of the switches on PORTD and output it on PORTB.

b. Use the provided connectors to connect PORTB to the LEDs and PORTD to the switches of STK500.

c. Download, and test your program on the board. Record your test results.

d. State your observations.

e. Write a program that increments by one upon pressing a push button, counting 0 through 255, and displays the result as a binary value on the 8 LEDs (i.e. 8 LEDs will be an 8-bit unsigned binary display). Use one of the bits of the PORTD as the input connected to one of the push buttons, and PORTB bits as outputs connected to the LEDs.

NOTE: There are two issues to consider:

I. The program needs to only increment the counter **once** each time the button is pressed. In other words, it needs to monitor the button and only increment the count when the button goes from being depressed (not pressed) to being pressed. (only at high to low transition of the button). Therefore after detecting a press on the push button, the program should check for the release of the push button.

II. When you press the button, the contacts as the switch closes initially bounce a few times before they settle closed (i.e. just like when you drop an object and it bounces a few times before it completely settles on the ground). This phenomenon is referred to as *switch bounce*. In order to solve the switch bounce problem, a common method, often referred to as *switch debouncing* is to add a delay after the initial button press (i.e. after seeing a button depressed to pressed transition) and before testing the state of the switch again. Since the button will bounce a few times after the initial button press, by waiting until that bouncing is done, we can avoid registering multiple button presses from a single press. One of the drawbacks in doing delay loops in this fashion is that the user cannot press the button any faster than the length of the delay loop (e.g. if the delay loop is 0.1 seconds, the user can't press the button any faster than 10 times a second). Consequently, it is desirable to select a delay time such that bouncing is eliminated while not delaying any longer than necessary.

f. Download the program and test it on the board.

Question 3_1: Experiment with different delay times. What is an approximate minimum delay time that eliminates debouncing (without unnecessarily extending the delay time)?

Last modified: 8/24/2017