Design Assignment 1

CpE 301 **Due: Monday, Feb. 6th, 2017 by 11:59pm** **Spring 2017 (Dr. Harris)**

Introduction

The purpose of this design assignment is to gain more practice programming – in both AVR assembly and C. This document will guide you in completing the following tasks:

1. Demonstrate your program from DA0 in hardware.
2. Write, simulate, and demonstrate in hardware an assembly program that performs specified functions.

The remainder of the document will give you additional guidance for each of the steps.

1. Demonstrate your program from DA0 in hardware

For this portion of the assignment, you will be using the Evil Mad Scientist board provided in your kits.

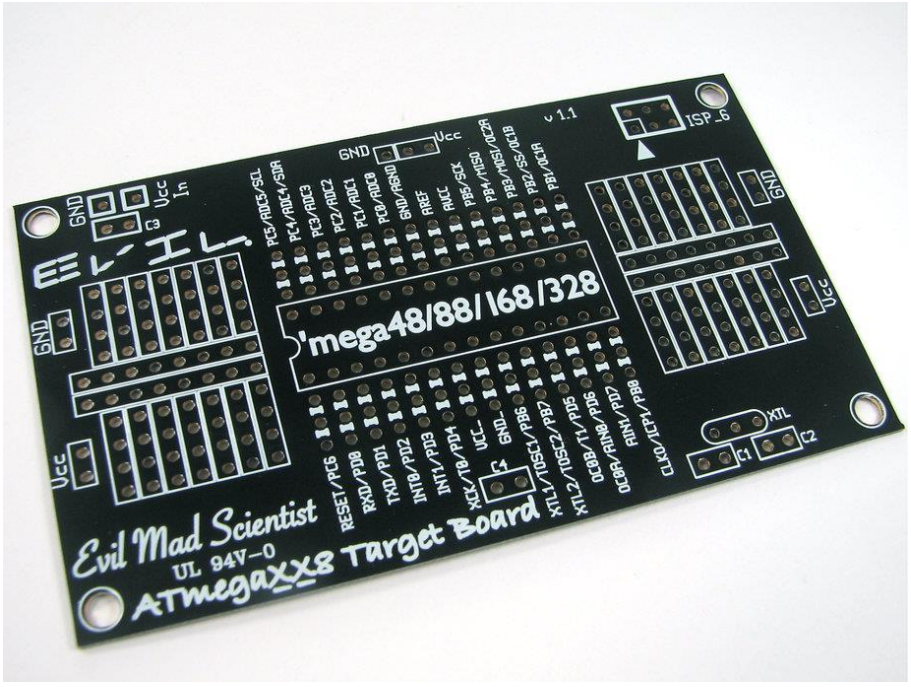

Figure 1. Evil Mad Scientist board (http://www.evilmadscientist.com/2008/tiny-portable-avr-projects-business-card-breakout-boards/)

Your board has a socket where you will place the Atmega328P part. Insert your Atmega 328p chip into the socket if it is not already inserted. Make sure that the groove at the top of the chip matches when you place it in the socket. The pin to the left of the groove is physical pin 1 on the package.

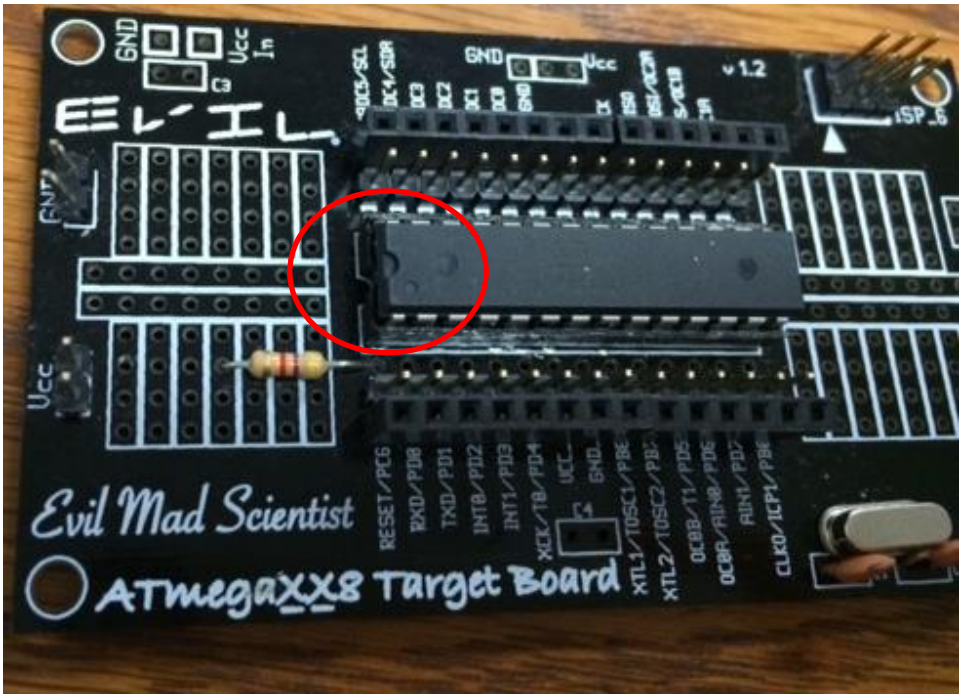

Figure 2. Groove of socket and chip match

Now you will need to supply power to the board. Insert the red wire from the battery pack into the VCC pin and the black wire from the battery pack into the GND pin. Leave one of the batteries (the middle one) out of the battery pack while you are building your circuit.

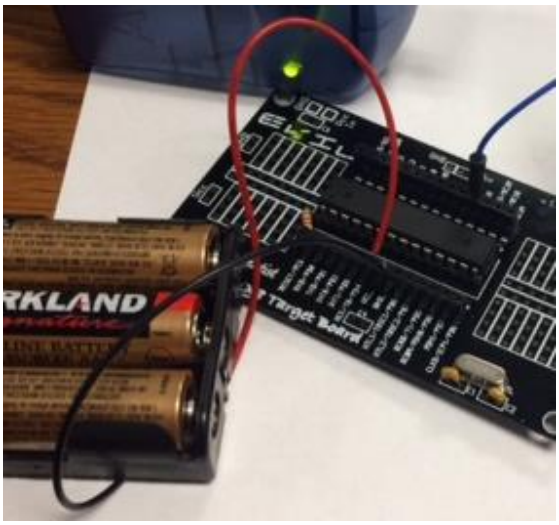

Figure 3. Insert power from battery

Now you will build the LED/resistor network shown in Figure 4 on the breadboard.

Figure 4. LED/resistor network

Connect the 330 Ohm resistor to the LED. They should be the only resistors in your kits, but you can google "resistor codes" to tell from the bands what the value of a resistor is. You will feed current (and voltage) through the resistor to the LED.

Be sure to remember that the anode is the longer pin on the LED while the cathode is the shorter pin. For the LED to turn on, the anode (i.e., the positive polarity terminal) should be connected to the higher voltage, and the cathode (shorter pin) to the lower voltage (in this case ground, GND). The end of the resistor not connected to the LED should connect to PB4 on the Evil Mad Scientist board. Figure 5 shows the connections of the breadboard. Make sure you insert the pins of the resistor into different columns.

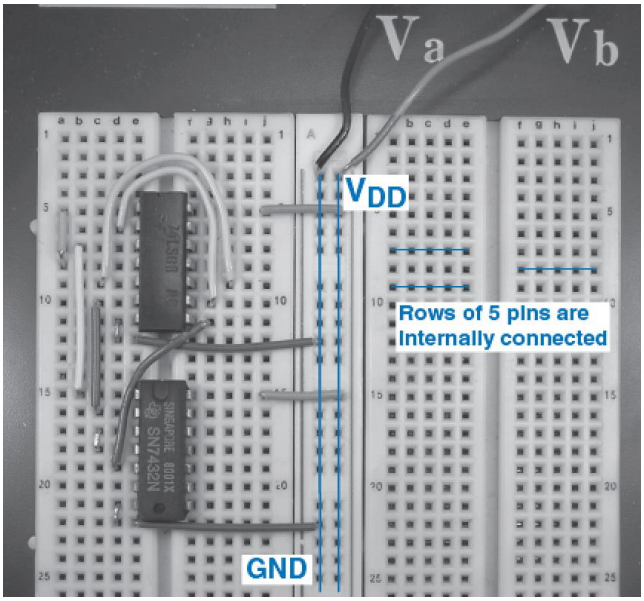

Figure 5. Breadboard layout

You may want to test your LED/resistor network by connecting the end of the resistor (that will eventually be connected to PB4) to VCC instead. If the LED doesn't light up, you get a chance to troubleshoot (bad LED, bad resistor (unlikely), wired something incorrectly, wires aren't pushed into board, etc.)

Now connect the ISP programmer to the board, as shown in Figure 6.

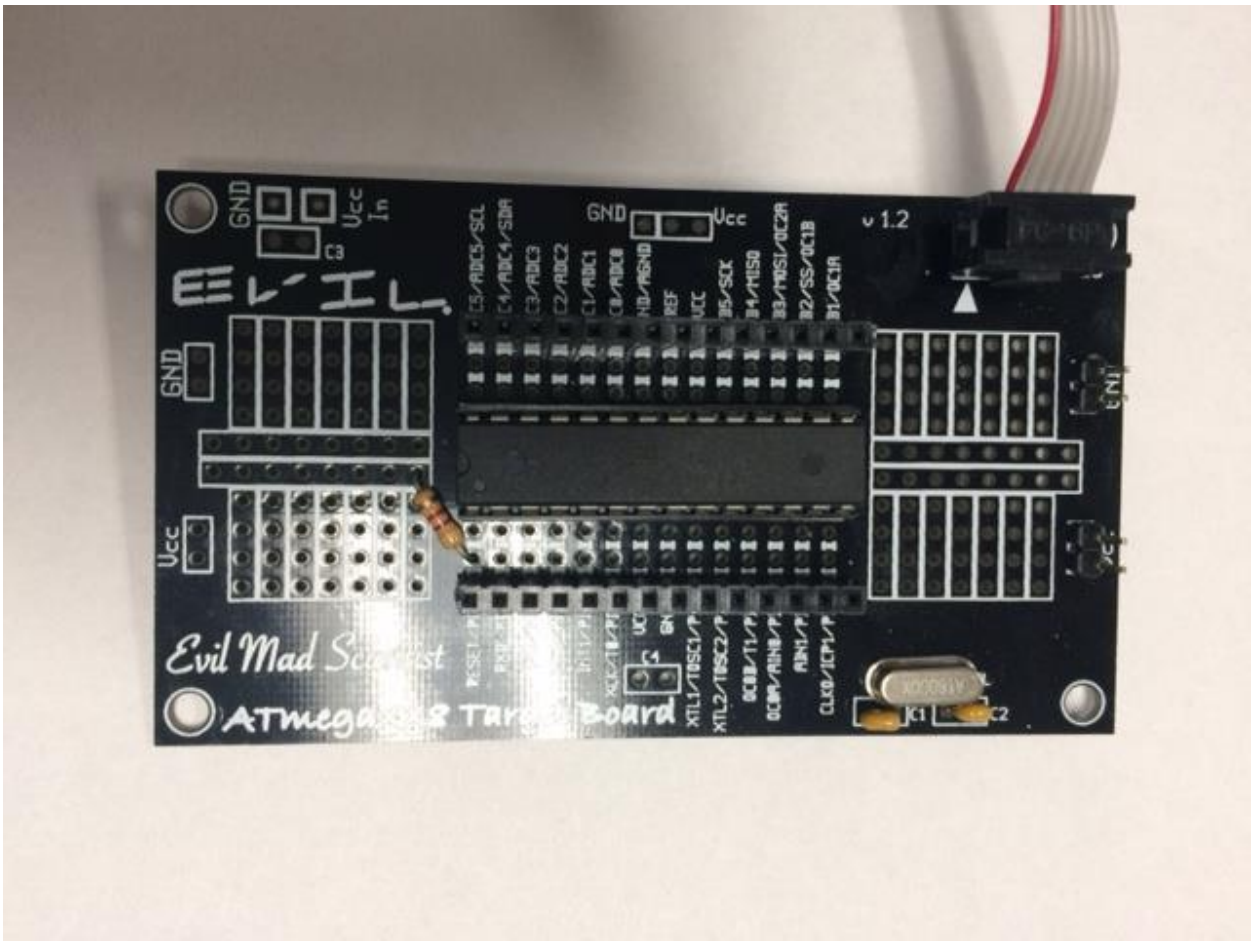

Figure 6. ISP programmer connection

You will notice a faint arrow symbol on the black header of the cable. Match that arrow to the arrow marked on the board, and also notice the location in Figure 6 of the red stripe.

Now you are ready to test your program from DA0 in hardware. The ISP programmer will indicate that the Atmega328p chip has power when the green LED is on (see Figure 7, left image).

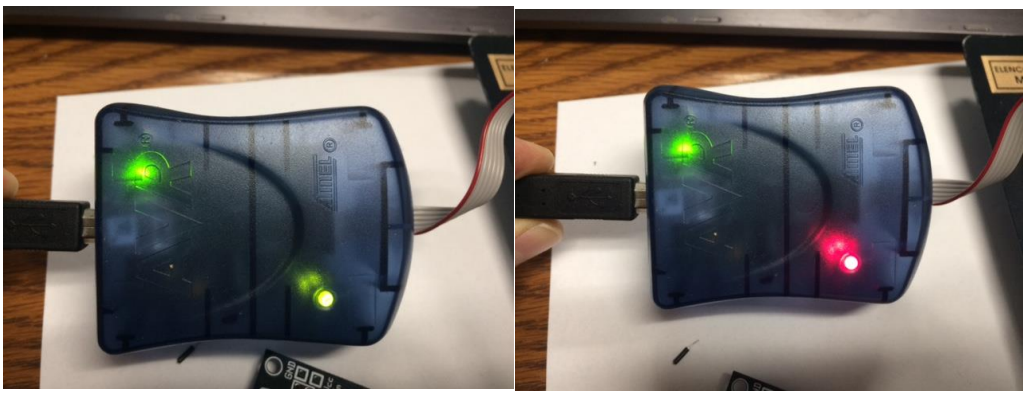

Figure 7. ISP programmer indicating that power is on (left) or off (right)

Make sure that the chip has power. Now download the program by clicking on Tools → Device Programming.

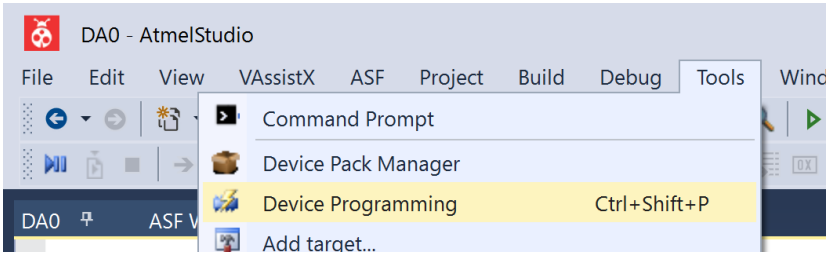

Figure 8. Device Programming

Then Select AVRISP mkII as the tool and click Apply.

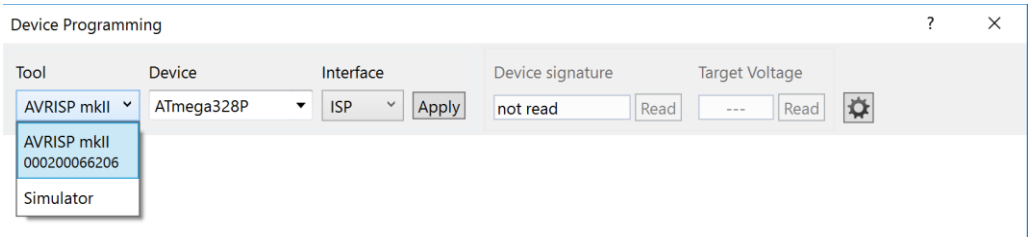

Figure 9. AVRISP mkII Tool

In this order, click on the read buttons under (1) Target Voltage and then (2) Device signature. The window should now look similar to Figure 10.

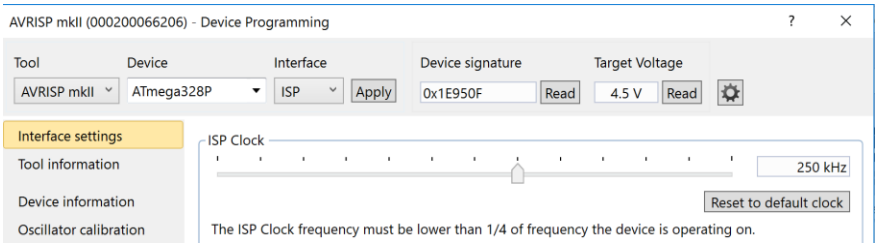

Figure 10. Read voltage and device

Now click on Memories on the left-hand side of the window and Program (see Figure 11).

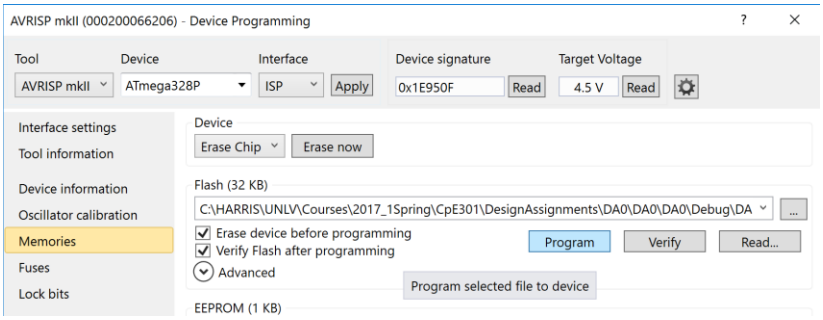

Figure 11. Program chip

The program is now loaded onto the chip and running. If you don't see the LED light up, debug your hardware and/or your program.

Once everything is working, experiment with modifying your code to make the LED light up or turn off.

2. Write, simulate, and demonstrate in hardware an assembly program that does the following:

a. Stores 25 numbers starting at memory address RAMEND/2 location. For the numbers to store, use the lower 8 bits of the memory address RAM_MIDDLE = RAMEND/2. Increment from the RAM_MIDDLE location to get the subsequent 24 numbers. Use the X/Y/Z registers as pointers to fill up 25 numbers starting from memory address RAM_MIDDLE.
b. Uses X/Y/Z register to parse through the 25 numbers and add all numbers divisible by 7 and place the 16-bit result in R20:21.
c. UsesX/Y/Z register to parse through the 25 numbers and add all numbers divisible by 3 and place the 16-bit result in R23:24. Parsing of the numbers for task b and c has to be done simultaneously.
d. Sets register R7.4 if the sum from part (a) is greater than 8-bits. Also asserts PB.4 (that is connected to an LED in hardware) in this case.
e. Sets register R7.3 if the sum from part (b) is greater than 8-bits. Also asserts PB.3 (that will be connected to an LED in hardware) in this case.

Also do the following:

- Determine the execution time @ 16MHz/#cycles of your algorithm using the simulation.
- Run your program in hardware and confirm the results.

What to Turn In

The following must be submitted via WebCampus by the due date/time to receive credit for the assignment. Messy, difficult to understand, or disorganized work will receive no credit.

Total points available: 100

0. **Time:** Indicate the amount of time this assignment took in hours. This will not affect your grade (unless omitted) but will help gauge the workload for this and future semesters. **[-5 points if omitted]**
1. **A pdf or word document** that contains (**in this order**):
   - **a.** A 1-paragraph description of your design. **[10 pts]**
   - **b.** The assembly code. The AVR assembly code must have been built (i.e., compiled/assembled) and working. The assembly code should be well-documented with a comment for each instruction. **[30 pts]**
   - **c.** A flow chart of your assembly code. **[10 pts]**
   - **d.** A schematic of your hardware. **[5 pts]**
   - **e.** Screenshots of Atmel Studio during debugging at the beginning and end of Steps 2.a-e of your assembly program. Be sure to show the necessary register and memory values. **[30 pts]**
   - **f.** A link to a ~1 minute Youtube video showing your program working in hardware. **[10 pts]**
   - **g.** The execution time of your program – show your work and put a box around your final answer. **[5 pts]**