Lab 9 – Interrupt Driven Output

Matthew Romleski  
Jayson Treml

For this lab the goal was to toggle an LED built into the XMEGA-A1U board in response to an interrupt provided by a button on Port C pin 3. As outlined in the lab assignment, the program needed to have the following aspects:

1. Utilize the stack pointer (due to the auto push & pop that is done when an interrupt occurs & finishes).
2. Have PQ3 set for output (this is the built-in port for the LED).
3. Have PC3 set for input on rising edge w/ INT0 configured and set to high priority.
4. Setup the interrupt table with the proper vector for the ISR.
5. Enable interrupts globally.
6. Loop forever through a dummy loop which just feeds data to PORTQ\_OUT.

To solve this, the program just needed to be coded in accordance to the requirements listed. Steps 1, 2, and the beginning of 3 were completed as has been done in previous labs. To configure INT0, set priority (3 part b), and enable interrupts (5), the necessary commands could be found by reviewing the slides for the lecture on interrupts. Alternatively, they could have been found in the datasheet for our processor provided on the official ATMEL website.

When setting up the interrupt vector (4), it was written essentially the same way as a subroutine except the final line (which is usually RET) was replaced with RETI. Then, the vector is fed to the interrupt table using .org and the interrupt that this vector should be associated with (in this case, PORTC\_INT0\_vect).

In conclusion, we learned how to write interrupt vectors and implement them into the interrupt table so that when an interrupt is triggered (when the button is pressed), the system moves from the task it’s currently working on (the loop) to deal with the interrupt.

For convenient viewing, you can find the program coded during the lab appended to this report.

Here is the resulting program which was coded during the lab period:   
