CS 3468
Lab 6

Jared Wallace

Objectives

1. Understand I/O control in assembly
2. Understand the behavior of interrupts in assembly

Tasks

Review

The RESET interrupt is the entrance to a sensor application. When the reset button is pressed (or the power is turned on), the RESET pin (PIN 20) of the ATmega128 processor triggers the RESET interrupt. The interrupt will initialize the sensor before giving control of the sensor to the application.

Part 1: The Reset Interrupt

Unlike in the last lab, by writing in assembly we gain complete, low level control over the interrupt process. What was handled for us last time, the initialization of the sensor application, can now be described in more detail. The process is as follows:

1. Clear status register and disable interrupts
2. Configure the power mechanism of the processor
3. Configure the data memory
4. Configure the I/O pins
5. Configure the I/O components
6. Enable interrupts and go the main function of the application

In this task, you will need to accomplish each step of the initialization process. After you have completed that, you will need to implement the same routine you made for the previous interrupt lab, which is the counter that tracks the quantity of times the reset button is pressed. As before, you will need to light the LED's in sequence to indicate the current count.

Fill out the table below to indicate the mapping between the counter and the LED's:

LED 0	**LED 1**	**LED 2**	**Count**
			0
			1
			2
			3
			4
			5
			6
			7

Review

UART is a type of serial communication between a sensor and another piece of equipment. In our case, the sensor will send information to the computer through UART communication. Accordingly, the sensor is the sender, and the computer will be the receiver. I will show you how to receive data from the sensor via a handy dandy Java program. The physical connection between your board and the computer will be via USB.

Part 2: UART communication in sensors by way of assembly

In this task, first make an application routine to send the internal count to the computer. Then, make an interrupt routine such that once a count is sent, the interrupt will be invoked to send another byte (0xF0) to the computer. *Make sure that the sending of the interrupt does not raise an interrupt to itself.* This will lead to a very cool yet ultimately fruitless infinite loop.

For ten bonus points, design an experiment to measure the transmission error rate of the serial communication.

Lab Report

1. Please demonstrate your program to the lab instructor and let him check your code at the end of the current lab project.

2. Your project report is due at the beginning of the next lab.

3. Grading criteria

   - Demonstration, 15 percent
   - Code, 15 percent
   - Report, 70 percent

Report instructions

Format:

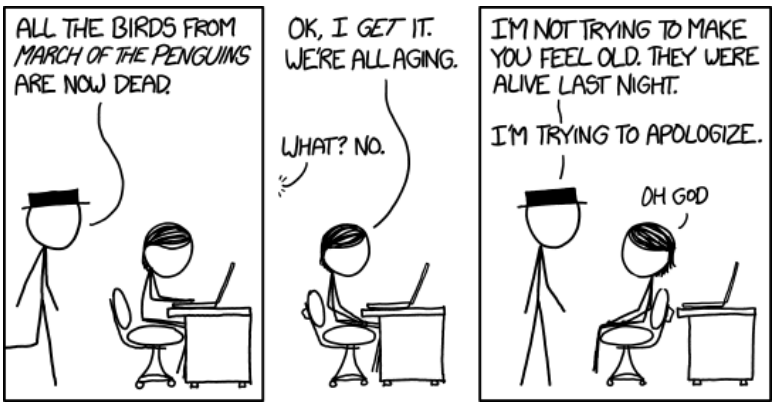
1. Include your name and ID in the first page
2. Font size of at least 10pt
3. Single spaced
4. Maximum of 5 pages (I will take points off for exceeding this without any good reason)
5. Please submit as PDF online, and turn in a hard copy

Content:

1. Introduction (10 percent of your grade) Please summarize the task of this lab and what you have learned in the lab
2. Reset interrupt (20 percent of your grade) Please describe how your code implements the initialization of the sensor app. Please describe how your code toggles the LED's.
3. UART (40 percent of your grade) Please describe in detail how your code transmits the count and handles the TX interrupt. Show me the result of using the following command (during the running of your program, of course)

   ```
   java -jar serial.jar raw
   ```

   Explain the possibility of a transmission error (how it could occur, in other words)



You ARE getting older, though