LAB #7 - Interrupts

You will be using your lab 4/5 and lab 6 code for this lab. Feel free to start with clean lab 6 code, if you so desire.

Overview

In this lab, we will be taking what you have done for lab 4/5, and replace the polling code with interrupts. This includes removing any action used to poll the timer, as well as moving the actions that happen as a result of the polling to the interrupts. You will also be turning the USART into an interrupt based receiver.

Procedure

1. Replace your timer polling from lab 4/5 with ISRs to handle the timer expirations. Refer to the data sheet for which interrupt you will need, p. 68. Also, lookup the TIMSK0 register for enabling interrupts, p. 115.

2. Look at the interrupt vector in the data sheet, and determine the name of the interrupt that tells us a receive has occurred. The actual name is TIMER0_COMPA_vect.

3. Write an ISR with the above interrupt, i.e. `ISR(TIMER0_COMPA_vect) { ... }`. It will need to do multiple things:
   1. On the interrupt occurring, there will be data in the USART Data Register, UDR1. This data must be read into a new buffer.
   2. Based on the value you encounter, you will need to change the current state (as in, the state machine state).
   3. This change of state must occur globally, and is based on the state machine which you implemented for Lab 4/5. In addition, global variables that are updated inside the Interrupt Service Routine, must be declared as volatile, i.e. `volatile unsigned char state;` This tells the compiler to not do any optimizations on this variable.

4. Lastly, change your binary clock so that it keeps time using interrupts, rather than polling.

Before leaving the lab, get checked off for credit by a Lab TA to receive credit.