Lab 4 - State machines and timers

The files you will be using for this lab can be downloaded [here](#).

**Before you begin:**

1. What do the following register and bit names stand for in reference to the AT90USB TIMER0 (chapter 13 of [this datasheet](#))? For each, write a sentence or so in your own words describing each:
   1. TCCR0B
   2. OCR0A
   3. TIFR0
   4. OCF0A
   5. CS02:0

2. Explain in words what the bit OCF0A in TIFR0 does. How could this be used to perform some operation at a specific frequency?

---

**Procedure**

1. In wunder.c, there are 3 function stubs and a #define that need to be filled out.

   - **unsigned char initialize_TIMER0()**
     
     This function should configure the TIMER0 on the Wunderboard to in a Clear Timer on Compare match (CTC) mode with a base clock frequency of clk\_/1024 (see #define CK1024).

   - **unsigned char check_TIMER0()**
     
     This function should immediately clear the OCF0A flag if the timer has elapsed and return a 1 so that it is ready for the timer to elapse again. If the timer has not elapsed, it should immediately return a 0.

     **HINT:** Look at the OCF0A bit in the TIFR0 register to see if you can write a new value. Don't forget you have to manually reset this flag every time the timer elapses.

   - **unsigned char set_TIMER0(unsigned char clock, unsigned char count)**
     
     This function takes two values, clock and count. The value of count should be copied into OCR0A and the value of clock should be used to set CS02:0. The TCNT0 variable should also be reset to 0 so that the new timer rate starts from 0. This function should return a 1 if it fails and a 0 if it does not.
2. Once you have written each of the functions, be sure to test them and show that you can blink LEDs at a certain speed. You will need to show the TA that you can change the speed of blinking based on the input values of the set_TIMER0() function.

3. In a previous lab you implemented serial transmission. To start the data logging you need to be able to receive a byte from the PC and receive a different byte to stop. You must create a state machine in your program using an enum. The image below shows the state machine you need to implement. To start sampling, use the character 's' to stop sampling you must also use 's'. See usart.h for the functions available to receive data.

4. Next combine all of the parts together to make a program that reads from the accelerometer and transmits the values to the PC. These values should be transmitted exactly 2 times a second. Once the values have been stored for about a minute, you should cut and paste the data in your terminal program into a file and graph it using excel or similar.

---

**Check-off**

Show your code and program to your TA. They will watch your system run for 30 seconds and count the number of samples in 30 seconds (it should be 60 ±1). They will come back and look at your graphed output of the samples you took in their presence. Be sure you bring a printed copy of your code to give to your TA.