Lab 4 – Data Logger with Timer Interrupts

Overview

Using interrupts allows for microprocessors and microcontrollers to handle events as they happen without wasting time always 'looking' for them (polling). In this lab we will be revising lab 3 to use interrupts for the timer.

Prelab

No Prelab this week, but it suggested you read: pages 16-18, 68-69, 110-116, and 198-202 of the AT90USB datasheet and in your own words describe what each of the following registers or bits does: **TIMSK0** and **OCIE0A**.

Procedure

1. Start with your code from lab 3. In this lab we will be converting our code to interrupt driven timing. Interrupts are a useful tool for handling things that occur at random (as far as the processor is concerned) times. Prior to this lab you have been using polling to look at the status of various functions in the microcontroller. Using interrupt will create cleaner code that gives more functionality to the design.

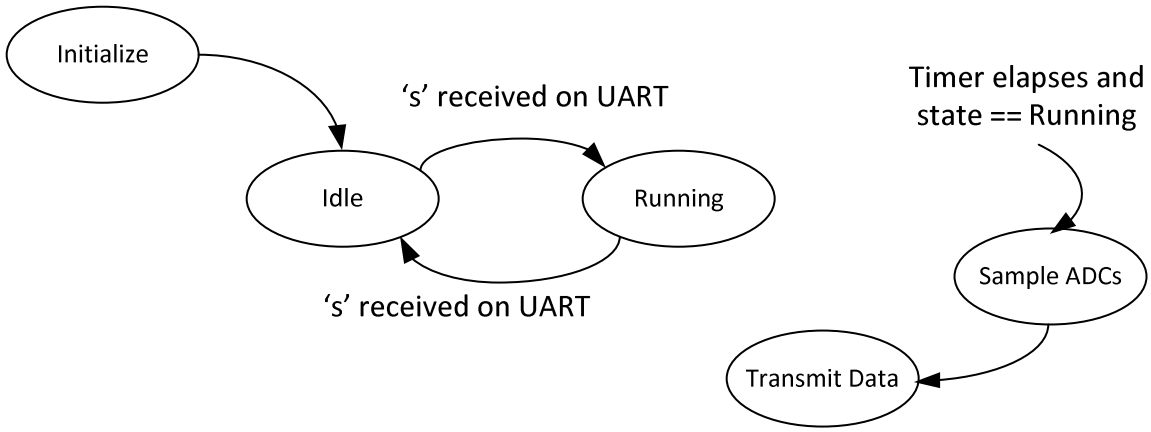

Figure 1: New Interrupt State Machine

2. When an interrupt is enabled, it will cause an Interrupt Service Routine (ISR) to occur. An ISR is like a function call that can happen at any time. You will need to replace the function of your 'checkTIMER()' with an ISR. We did an example of this in class, so look at your class notes.

3. Usually, it is important to keep ISRs as short as possible since it is possible for many ISRs to happen in quick succession. Since you are only enabling the single Timer ISR, this is not as important of a risk. To point you in the correct direction, you can move the majority of your state machine into the timer ISR and check for serial data in the main loop. This usually can work well, but it can also lead to trouble if your code is doing something that is time sensitive.

4. The first step is to implement the Timer interrupt and ensure that it does what you expect. After you know it is working demonstrate it to your TA.

5. Now that the timer interrupt is functional, integrate your lab 3 code into the interrupt and demonstrate it to your TA.

   Hints:

   - Any variables that are manipulated in both the main function AND the timer interrupt should be declared a global AND as 'volatile'. Setting them to be volatile tells the compiler that they may change values at random times (when an interrupt happens). If you don't do this, they will not work.
   - Be careful, you need to still sample only twice per second. So be careful that when the interrupt happens you correctly decide if 'this time' is the time to sample. Also be careful that whatever you do with in the interrupt is fast enough that it i completed before the next interrupt occurs!

Demonstrate and Submit code

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.

Study Questions

1. In terms of programming and in your own words, define the following two terms; Event and Interrupt.
2. Read (or reread) pages 16-18, 68-69, 110-116, and 198-202 of the AT90USB datasheet and in your own words describe what each of the following registers or bits does:
   a. **TIMSK0**
   b. **OCIE0A**
3. Using Google, look up the 'Interrupt Vectors' for AVR GCC. What are the exact names of the vectors for UART1 for data received, data sent, and USART ready for more data? Also, what is the exact name of the interrupt vector for TimerCounter0 Compare Match A? A good place to start is http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
4. What is the general syntax for defining an interrupt service routine?

Lab Summary

Task	Completed?
Prelab	
Timer interrupt works	3pts.
Final code Works	3pts.
Study Questions	4pts.