EE333 Introduction to Microcontrollers

Lab#4 Microcontroller Timing

Allan A. Douglas, Associate Professor

**Lab Overview**

In this lab, students will learn how to use microcontroller timers and interrupts to controller operation of software.

**Required Materials**

|  |  |
| --- | --- |
| Breadboard | Function Generator |
| Arduino UNO (or Spark Fun RedBoard equivalent) | Oscilloscope |
| ATmega328P Microcontroller Datasheet | Arduino ISE Software |

**Required Work**

The areas of **bold** text below are items required in your Lab Report.

Part 1: Interrupt Speed Test

In this part of the lab, you will connect the function generator to an interrupt input on the Arduino and write a sketch to respond to external interrupts.

Connect the function generator output to the oscilloscope on channel one. Connect the ground leads together. Set the function generator so that it generates a square wave from 0-5V at 1Hz.

Now, connect the ground on the Arduino to the common ground between the oscilloscope and the function generation. Then, connect the function generator output to the INT0 pin on the Arduino.

Create a new Arduino sketch.

* Define two global variables called **count** and **prev\_count**.
* Make sure the variables are declared as a volatile unsigned long.
* Initialize both variables to zero.

Setup()

* Enable interrupts.
* Use the attachInterrupt() function to assign INT0 to your Interrupt Service Routine (ISR).

ISR()

* Write your ISR function below the loop function.
* Each time the ISR is called, it should increment the variable **count**.

Loop()

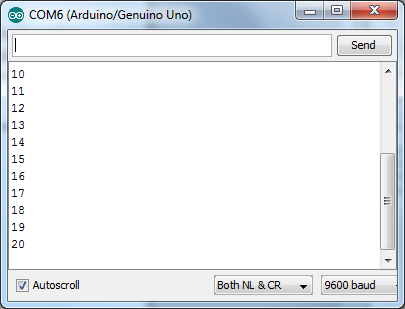
* Check to see if **count** > **prev\_count**. If this is true, print **count** to the Serial Monitor and assign **prev\_count** = **count**.
* Make sure your serial baud rate is set to 9600.

**Include a copy of your sketch in your Lab Report.**

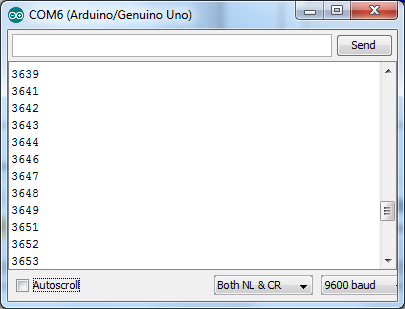
**Explain the function that enables interrupts and the attachInterrupts() functions.**

**Why is it necessary that count and prev\_count be defined as volatile variables?**

Observe the rate of change in the value of count. Your Serial Monitor should look like the following figure:



Now, gradually increase the function generator frequency. At some frequency, you will see that the stream of count values will begin to skip values in the Serial Monitor.



**Include pictures of the Serial Monitor with and without skipping in your report.**

**Include an oscilloscope picture that shows the function generator signal and the Arduino Tx signal before and after skipping occurs.**

**Why is the count skipping values? Please explain this considering the code, interrupt rate and baud rate.**

Change the serial baud rate in your sketch and in the Serial Monitor to 115,200.

Observe the Serial Monitor. It should not be skipping values any longer. You should be able to increase the function generator frequency.

**How far can you increase the frequency? Explain why you can achieve higher interrupt rates now without skipping.**

**Include a copy of your modified sketch in Lab Report.**

Part 2: Measuring Pulse Duration

In this part of the lab, you will measure the duration of a function generator pulse using the micros() function and interrupts. Then you will repeat the process with a timer.

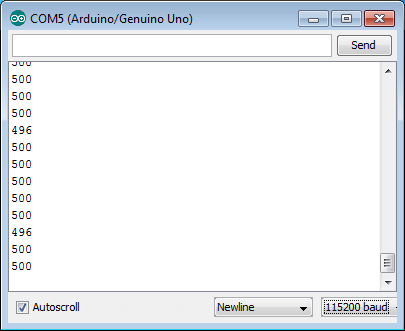
Load the pulse\_timer sketch provided with the lab.

**Explain the pulse\_timer sketch including the operation of the pulse(), setup(), and loop() functions. Make sure to discuss the operation of the attachInterrupt(0, pulse, CHANGE) instruction.**

**Which Timer is used for the micros() and millis() functions?**

Set the function generator frequency to 1kHz and set the duty cycle to 50%. You should see the value of 500 printed to the serial port repeatedly. This is a measurement of the pulse high time in microseconds on the interrupt input pin.

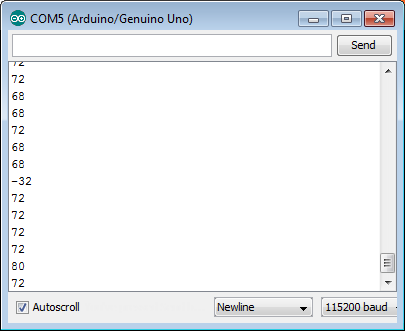
**Measure the high time of the function generator signal and compare it to the value measured by the microcontroller. Include the oscilloscope waveform in your Lab Report.**



Now, adjust the function generator frequency to 10kHz.

**Measure the high time in (microseconds ) of the function generator signal and compare it to the value measured by the microcontroller. Include the oscilloscope waveform in your Lab Report.**

Increase the duty cycle (pulse high time) to approximately 70%. Observe the Serial Monitor output. Occasionally you should see a negative number reported to the serial port.



When you see this effect, measure the high time of the function generator signal and compare it to the value measured by the microcontroller. Include the oscilloscope waveform in your Lab Report. Explain why the calculated high time is reported as a negative value occasionally. In your explanation, make sure to refer to the rising and falling edges of the function generator signal.

**Write a conclusion in your Lab Report that summarizes what you learned in the lab.**