Lab 6: Timers on the MSP430*

adrian valenzuela
CJ Ganier

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

In this lab, we will cover various timing options for the MSP430.

In this lab, we will cover the timing options for the MSP430. The first part explains the clocking system of the processor and the options it allows. The second part will cover the timers and timer interrupts available on the MSP. Each of these sections is strongly related to real time programming, but that topic will be dealt with separately in another lab. In general, a real-time application is one which responds to its inputs as fast as they happen. The microprocessor is generally expected to take less time to handle an interrupt or stimulus than the time before the next event will happen.

The timer system is broken into three primary parts on the MSP430: Timer A, Timer B, and the Watchdog timer. Timer B is larger and more versatile than Timer A. The User's Guide[1] and data sheet will explain the differences, but the way that the control registers configure each timer is largely the same. The watchdog timer will be covered in a module[2] of its own. The timers are closely tied with real time applications because they govern the occurrence of the periodic functions of the processor.

Exercise 1

Timer A

Refer to **Chapter 11: Timer_A** of the User's Guide[3] to get a detailed description of all the options in Timer A. Basically, setting up the timers requires that you define a source for the timer and to specify a direction for the count. It may also be helpful to clear the timer register before you begin to guarantee and accurate count from the first instance. Set up Timer A in Continuous Mode and sourced from SMCLK. Set TACCR0 and TACCR1 to have two different values. Output TA0 and TA1 from header J8 of the board so that you may directly observe the output of Timer A. You may need to remove the jumpers in order to have access to these signals.

Using two different channels of the Oscilloscope try to recreate parts of **Figure 11-13. Output Example- Timer in Continuous Mode**. On Channel 1 show **Output Mode 4: Toggle** and on Channel 2 show **Output Mode 6: Toggle/Set**. Vary the TACCTLx in order to get as close to the orginal figure as possible. Take a screenshot of the scope and include it in your lab report.

Try this again using Timer B. This time on Channel 1 show **Output Mode 4: Toggle** and on Channel 2 show **Output Mode 2: Toggle/Reset**. Take a screenshot and submit it. What are the differences between Timer A and Timer B? What was the frequency your signals? What is the relationship between TACCTLx and the frequency?

*Version 1.6: Aug 21, 2005 7:03 pm -0500
†http://creativecommons.org/licenses/by/1.0
[1]http://cnx.rice.edu/content/m12396/latest/usersguide.pdf
[2]"Watchdog Timer" <http://cnx.org/content/m11998/latest/>
[3]http://cnx.rice.edu/content/m12396/latest/usersguide.pdf

Exercise 2

Timer

Set up the timers to fire interrupts. Using the `seg_count()` function from lab 5[4], use the timers to write a counter. The seven-segment display should display the number of seconds that have elapsed since the timer was started. Button_1 should start/stop the timer. Once the seven-segment gets to its maximum value of 15 it should roll over. There should be no for-loops in your program, and should be entirely interrupt driven. You may use Timer A or Timer B. It is possible to have each Capture Control Register to fire an interrupt once it reaches its max value. How was the timer set up to calculate one second?

Exercise 3

Duty Cycle

We have discussed earlier that the Duty Cycle related to the width of a pulse. If we trigger an LED with a reletively high frequency square wave, it will appear to be on constantly eventhough it is actually switching on and off quickly. As we vary the duty cycle of our trigger signal, then the LED may appear to get dimmer or brighter depending on which way we vary it.

Set up the timers to toggle an LED. Without changing the frequency of your timing pulse, change the duty cycle so that the LED appears to fade in and out. The time it takes to go from completely off to max brightness shouldn't take more than a second, then it should repeat. Once again, there should be no for-loops in your program, and you can use any combination of the timers that you wish.

Extra Credit: Once you get a single light to fade in and out, try to get all of the lights to fade asynchronously. This means that while one LED is fading out the next one should begin to fade in and so on. Good luck.

[4] "Lab 5: Interrupts" <http://cnx.org/content/m12322/latest/>