CPE 325: Intro to Embedded Computer System

**Lab07**

**MSP430 Timers, Watchdog, A and B Timers**

**Submitted by**: Nolan Anderson

**Date of Experiment**: 10/21/2020

**Report Deadline**: 10/21/2020

# Introduction

This lab essentially covers everything you need to know about using timers In the MSP430 to perform different operations. Specifically in this lab, we are using the timers for changing the status of LEDs. In question one, we use the watchdog timer to specify 6 seconds of brightness change and we can stop the watchdog timer with a switch which will stop the change in brightness. We can use another switch to turn the watchdog timer back on. We also use Timer A and B in question one and that is used to change the overall brightness of the LED by turning it off and on very quickly. In question 2, we can also use the watchdog timer to poll and wait for a switch to be pressed in order to use less energy. That way we can poll every once and a while and when a switch is pressed it “wakes up” without having to use a while loop.

# Theory

**Watchdog Timer**:

|  |
| --- |
| The primary function of the WDT is to perform a controlled-system restart after a software problem occurs. You can turn it off it is not needed, however. You can use the watchdog timer in the interval timer mode and use its interrupt service routing for blinking an LED. |

**Timers**:

|  |
| --- |
| In the MSP430, there are several timers including the Watchdog Timer, Real Time Clock, Timer A, B, and D. Timers are useful in performing periodic tasks. Timers are helpful in that you can perform other things while you are waiting and cuts down on code complexity and saves energy. |

**Continuous Mode, Up mode and Up / Down mode of operation:**

|  |
| --- |
| * **Continuous mode**: The counter counts up to the max value and the goes back down. * **Up mode**: The timer counts up to the value for channel 0 and when it gets there the interrupt for channel 0 is called. These may correspond to output pin control or ISR vectoring. * **Up/Down Mode**: The timer counts up to the value in the channel 0 register, and then counts back down to 0. Interrupts are called when the value is reached on the up count and the down count. |

# Results & Observation

**1. For Q1, how do you implement PWM? What part of your code changes the brightness? How**

**do you handle SW press?**

|  |
| --- |
| For the PWM, I essentially used a counter to check for when the PWM reaches 3 seconds, and when it does, the PWM subtracts down to 0 and when it reaches 0 seconds again it going back up to 3 seconds just to start all over again. The part of the code that actually changes the brightness is in the timer ISR interrupts. One switches it off and one switches it on. For the switch press, since we are using the watchdog timer as the interval, I am simply stopping the watchdog timer when the switch is pressed and when the other switch is pressed it starts the watchdog timer again. |

**2. For Q2, what is the blinking frequency of LED2?**

|  |
| --- |
| 5ms on, 5ms off. Period = 1ms, total frequency is 1HZ. |

## Results Screenshots/Pictures:

**Question 1 brightness:**

As you can see, they are at different brightness levels at different times.



## Observations:

Timers in the MSP430 can be confusing to use but they are incredibly useful in making you code simpler and more efficient.

# Conclusion

Timers are very useful in embedded systems. They allow you to basically make your code more efficient by not using loop and essentially waiting for something to happen. You can also use them in different ways to perform different operations. While it is more of the “long way round” method, it does work and makes things very efficient.

Video link:

<https://drive.google.com/file/d/19v80cNgtL0QQSZvks-sAMXw-0odtZwdZ/view?usp=sharing>

Folder link:

https://drive.google.com/drive/folders/1\_Y3ABMDhCUxc9phtQ8JKHCM8LDOK7k7J?usp=sharing

# Appendix

## Appendix 1

|  |
| --- |
| /\*------------------------------------------------------------------------------  \* Student: Nolan Anderson  \* Program: main.c  \* Date: Oct 21, 2020  \* Input: Switches on the board  \* Output: The LED 1 increases in brightness and then decreases in brightness  \* and then restarts again.  \* Description: This code uses the watchdogtimer essentially quickly toggle the red  \* led to maximum brightness and then back down. It increases and decreases  \* for 3 seconds.  \*-----------------------------------------------------------------------------\*/  **#include** <msp430F5529.h>  **#define** SW1 ((P2IN&BIT1)== 0)  **#define** SW2 ((P1IN&BIT1)== 0)  **void** **main**(**void**)  {  WDTCTL = WDT\_MDLY\_32; // 32ms interval, default/  \_EINT(); // Enable interrupts  SFRIE1 |= WDTIE; // Enable WDT interrupt  P1DIR |= BIT0; // LED1 as output  P1OUT &= ~BIT0; // ensure LED1 off  P2DIR &= ~BIT1; // Configuring Switch 1  P2REN |= BIT1;  P2OUT |= BIT1;  P1DIR &= ~BIT1; // Configuring Switch 2  P1REN |= BIT1;  P1OUT |= BIT1;  TA0CCTL0 = CCIE; // TA0 count triggers interrupt  TA0CCR0 = 96; // Set TA0 (and maximum) count value  TA0CCTL1 = CCIE; // TA0.1 count triggers interrupt  TA0CCR1 = 96; // Set TA0.1 count value  TA0CTL = TASSEL\_1 | MC\_3; // ACLK is clock source, UP/DOWN mode  **int** i; // Counter  **for**(;;) // Infinite loop  {  **if**(SW1) // If switch 1 is pressed  {  **for**( i = 2000; i>= 0; i--); // Debounce .02ms  **if** (SW1) // If switch 1 is pressed  {  WDTCTL = WDTPW + WDTHOLD; // Stop the watchdog timer  }  }  **if**(SW2) // If switch 2 is pressed  {  **for**( i = 2000; i >= 0 ; i--); // Debounce .02ms  **if**(SW2) // If switch 2 is pressed  {  WDTCTL = WDT\_MDLY\_32; // Start up the watchdog timer again  }  }  }  }  // The following two ISRs are called by the TA0CCTL interrupts  // And they essentially quickly toggle the LED on and off very fast.  **#pragma** vector = TIMER0\_A0\_VECTOR  **\_\_interrupt** **void** **timerISR**(**void**)  {  P1OUT |= BIT0; // Turn red led on  TA0CCTL0 &= ~CCIFG; // Clear interrupt flag  }  **#pragma** vector = TIMER0\_A1\_VECTOR  **\_\_interrupt** **void** **timerISR2**(**void**)  {  P1OUT &= ~BIT0; // Turn red led off  TA0CCTL1 &= ~CCIFG; // Clear interrupt flag  }  **#pragma** vector = WDT\_VECTOR  **\_\_interrupt** **void** **wdtISR**(**void**)  {  **static** **int** counter = 96;  **static** **int** pwm = 1;  // If the counter is at the bottom.  **if**(pwm == 0 )  {  **if**(counter == 96)  {  pwm = 1;  }  counter++; // Increment the counter  TA0CCR1 = counter; // Set the interrupt equal to the timer  }  // If the counter is at the top  **if**(pwm == 1)  {  **if**(counter == 0) // If the counter is equal to 0, at bottom  {  pwm = 0; // Set the PWM to 0  }  counter--; // Decrement the counter (decreasing in brightness)  TA0CCR1 = counter; // Set the interrupt equal to the timer  }  } |

## Appendix 2

|  |
| --- |
| /\*------------------------------------------------------------------------------  \* Student: Nolan Anderson  \* Program: main.c  \* Date: Oct 21, 2020  \* Input: No input  \* Output: LED that blinks at 1hz  \* Description: This code uses the watchdog timer to interface with the Switch 1  \* ISR to blink an LED at 1hz.  \*-----------------------------------------------------------------------------\*/  **#include** <msp430.h>  **#define** GREENLED 0x80 // LED2 - Mask for BIT7 (1000\_0000b)  **void** **main**(**void**)  {  WDTCTL = WDT\_MDLY\_0\_5; // 5ms interval timer  P4DIR |= GREENLED; // Configure the P4.7 as output.  \_EINT(); // enable interrupts  SFRIE1 |= WDTIE; // Enable WDT interrupt  P2IE |= BIT1; // Enable interrupt at P1.1 for Switch 1  P2IES |= BIT1; // Enable hi->lo edge for interrupt  P2IFG &= ~BIT1; // Clear any errornous interrupt flag  \_BIS\_SR(LPM0\_bits + GIE); // Enter LPM0 with interrupt  }  // This is the switch 1 ISR  **#pragma** vector = PORT2\_VECTOR  **\_\_interrupt** **void** **PORT2\_ISR**(**void**)  {  P2IFG &= ~BIT1; // Clearing the flag  P4OUT ^= GREENLED; // Toggle the green LED.  }  // Watchdog Timer interrupt service routine  **#pragma** vector=WDT\_VECTOR  **\_\_interrupt** **void** **watchdog\_timer**(**void**)  {  **static** **int** i = 0; // Int variable for counter  i++; // Increment the counter  **if** (i == 1000) // If it is on the 1000th iteration  {  P2IFG |= BIT1; // Enable interrupt at P1.1 for Switch 1  i = 0; // Reset i value  }  } |