Austin Bumbalough
CPE 325-08
Lab 07
10/16/2019

Lab 7 Solution

Part 1 + Bonus

In lab 7 part 1, I wrote a C program that used the MSP430 Timer B peripheral to drive an LED with a PWM signal. Two hardware switches on the board were configured to increase or decrease the brightness of the LED by configuring the PWM signal duty cycle. For the bonus problem, holding down both buttons uses the watchdog timer in interval mode to blink the LED at its current brightness level at 0.25 Hz.

Source Code

```
int brightness[5] = {50, 150, 350, 650, 975};

int main(void) {
    // Setup Ports and Registers
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer
    P1DIR &= ~(BIT1+BIT0); // Set P1.0 and P1.1 to input
    P2DIR |= BIT2; // Set P2.2 to output
    P2SEL |= BIT2; // Select special function for P2.2 (TB1 Output)
    TBCTL = MC_0; // Stop timer B while configuring
    TBCTL |= TBSSEL_2; // Timer B Clock Source: SMCLK
    TBCCTL1 |= OUTMOD_7; // Set Timer B1 to toggle reset/set output mode
    TBCCR1 = brightness[2]; // Set Timer B1 compare value
    TBCCR0 = 1000;
    TBCTL |= MC_1; // Start Timer B in up mode
    P1IES |= BIT1+BIT0; // Falling edge triggers interrupt
    P1IFG &= ~(BIT1+BIT0); // Clear any pending interrupts
    P1IE |= BIT1+BIT0; // Enable interrupts on P1.0 and P1.1
    _EINT(); // Set GIE bit in SR
```

```
    LPM0; // Put system in low power mode 3

    return 0;
}

#pragma vector = PORT1_VECTOR
__interrupt void P1ISR(void) {
    static unsigned int level = 2;
    static unsigned int bothPressed = 0;
    for (int i=10000;i>0;i--); // 100 ms delay allows both buttons to be press
    switch (P1IFG & (BIT1+BIT0)) {
        case BIT0: // SW1 pressed, brightness increases
            for (int i=10000;i>0;i--);
            if (SW1 == 0) {
                if (level < 4) level++;
            }
            P1IFG &= ~BIT0; // Clear P1.0 interrupt flag
            break;
        case BIT1: // SW2 pressed, brightness decreases
            for (int i=10000;i>0;i--);
            if (SW2 == 0) {
                if (level > 0) level--;
            }
            P1IFG &= ~BIT1; // Clear P1.1 interrupt flag
            break;
        case (BIT1+BIT0):
            if !(bothPressed) {
                for (int i=10000;i>0;i--);
                if (SW1+SW2 == 0) {
                    bothPressed = 1; // Set flag
                    WDTCTL ^= (WDT_ADLY_1000 ^ (WDTPW | WDTHOLD)); // Toggle r
                    P1IES &= ~(BIT1+BIT0); // Change interrupt trigger to risi
                }
            } else if (bothPressed) {
                for (int i=10000;i>0;i--);
                if (SW1+SW2 ==  BIT1+BIT0) {
                    bothPressed = 0; // Clear flag
                    WDTCTL ^= (WDT_ADLY_1000 ^ (WDTPW | WDTHOLD)); // Toggle r
                    P1IES |= BIT1+BIT0; // Restore interrupt trigger to fallir
                }
            }
            P1IFG &= ~(BIT1+BIT0); // Clear interrupts
            break;
    }
```

```
    TBCCR1 = brightness[level];
}

#pragma vector=WDT_VECTOR
__interrupt void wdt_isr(void) {
    TBCTL^= BIT4; // Toggling bit 4 switches timer B between stop mode and up
}
```

Part 2

In part 2 of lab 7, I used Timer B to generate a 1 KHz square wave. The generated signal was used to drive a piezo buzzer connected to P3.5. The buzzer and LED1 were pulsed at 0.5 Hz.

Source Code

```
int main(void) {
    // Setup Ports and Registers
    WDTCTL = WDT_ADLY_1000;   // set watchdog timer to interval mode 1000 ms
    P2DIR |= BIT2; // Set P2.2 (LED1) to output
    P2OUT |= BIT2; // Initialize LED1 to on
    IE1 |= WDTIE; // Enable WDT interrupts
    P3DIR |= BIT5; // Set P3.5 (Buzzer) to output
    P3SEL |= BIT5; // Select special function for P2.2 (TB4 Output)
    TBCTL = MC_0; // Stop timer B while configuring
    TBCTL |= TBSSEL_2; // Timer B Clock Source: SMCLK
    TBCCTL4 |= OUTMOD_4; // Set Timer B4 to toggle output mode
    TBCCR0 = 523; // Sets buzzer frequency to 1 KHz
    TBCTL |= MC_1; // Start Timer B in up mode
    _EINT(); // Set GIE bit in SR
    LPM0; // Put system in low power mode 0

    return 0;
}

#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void) {
    P2OUT ^= BIT2; // Toggle LED1
```

```
    P3SEL ^= BIT5; // Toggle function of P3.5
}
```