LAB 4: DIMINISHING FREQUENCY CONTROL FOR LED FLASHING

Nicholas Snyder

University of New Hampshire
ECE 649/796

Objectives:

The purpose of this lab was to continue working with timers and combine GPIO with interrupts. PWM was also used for playing the buzzing tunes to further understand the timing operations of the MSP430.

Background:

The supporting material included the current lectures on timers and newly introduced port interrupts. This lab used pieces from every previous lab, so those materials were also included.

Analysis:

This lab consisted of four tasks each with a different feature which required knowledge of different aspects of the MSP430. The first task was to display a welcome message to the user and direct them to press button S1 to continue the program. This drew upon labs one and two which delt with buttons and the LCD respectively. Task two asked to wait three seconds after pressing S1 to initiate a countdown on the LCD and flash the red LED. This could be completed with the same background as task one. Task three was the most complex as it asked to create a PWM signal to a GPIO pin that when in series with a buzzer would output a note of a specified frequency. This required a deeper knowledge of timers and interrupts. Task four was a continuation of task three as it asked to play a simple song of at least sixteen notes.

Challenges:

This was one of the harder labs for sure. I had to go to office hours, and I still haven't completed all the tasks. As it stands I have yet to complete task three but up until that point I have been successful in completing the other tasks. I have been struggling with the timer and interrupt sections most of all. I am not sure which registers I should be using and where I should be modifying them.

Appendix:

```c
#include <msp430.h>
#include "driverlib.h"
#include "myGpioLab3.h"
#include "myClocks.h"
#include "myLcdLab3.h"

#define redLED BIT0
#define greenLED BIT7
#define GREEN_ON 0x80
```

```
#define GREEN_OFF 0x7F
#define RED_ON 0x01
#define RED_OFF 0xFE
#define BUTTON11 BIT1
#define BUTTON12 0x06
#define BUZZER BIT3

void config_clocks(void);
void welcome_message(void);
void countdown(void);
void play_music(void);
void play_note(char);

/**
 * main.c
 */
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer
    PM5CTL0 &= ~LOCKLPM5; // Enable the GPIO pins

    initGPIO();     // Required for the LCD
    initClocks();   // Required for the LCD
    myLCD_init();   // Required for the LCD

    P1DIR = redLED; // Direct pin as output
    P1REN = BUTTON11; // pull-up button 1
    P1OUT &= ~redLED; // Turn LED Off
    P1SEL0 |= BUZZER;

    P9DIR = greenLED;
    P9OUT &= ~greenLED;

    welcome_message();
    countdown();
    config_clocks();
    play_music();

    return 0;
}

//*******************************
#pragma vector = TIMER0_A0_VECTOR
__interrupt void myISR()
{
    // clears the flag (CCIFG in TA0CCTL0)
    TA0CCTL0 &= ~CCIFG;

    P1OUT ^= BUZZER | redLED;

    P9OUT ^= greenLED;
}

void config_clocks()
{
    TA0CCTL0 |= CCIE; // Enable Channel 0 CCIE bit
    TA0CCTL0 &= ~CCIFG; // Clear Channel 0 CCIFG bit
    TA0CCR0 = 0xffff; // 1 second period
    TA0CCR2 = TA0CCR0 / 2; // 50% duty
    TA0CCTL2 = OUTMOD_7; //

    // Timer_A: ACLK, divide by 1, up mode, clear TAR (leaves TAIE=0)
    TA0CTL = TASSEL_1 | ID_0 | MC_1 | TACLR;
    // Enable the global interrupt bit (call an intrinsic function)
    __enable_interrupt();
}

void welcome_message()
{
    while((P1IN & BIT1) != 0)
    {
        // while button is not pressed

        myLCD_showChar('P', 1);
        myLCD_showChar('R', 2);
        myLCD_showChar('E', 3);
        myLCD_showChar('S', 4);
        myLCD_showChar('S', 5);
        myLCD_showChar(' ', 6);

        __delay_cycles(5000000);

        myLCD_showChar('S', 1);
        myLCD_showChar('1', 2);
        myLCD_showChar(' ', 3);
        myLCD_showChar(' ', 4);
        myLCD_showChar(' ', 5);
        myLCD_showChar(' ', 6);

        __delay_cycles(5000000);
    }
}

void countdown()
{
    // clear LCD
    myLCD_showChar(' ', 1);
    myLCD_showChar(' ', 2);
    myLCD_showChar(' ', 3);
    myLCD_showChar(' ', 4);
    myLCD_showChar(' ', 5);
    myLCD_showChar(' ', 6);

    char c;
    int i;
    for (i = 3; i > 0; i--)
    {
        c = i + '0';

        myLCD_showChar(c, 1);
        P1OUT |= BUZZER | redLED;
        __delay_cycles(3000000);

        myLCD_showChar(' ', 1);
        P1OUT &= ~BUZZER & ~redLED;
        __delay_cycles(3000000);
    }

    myLCD_showChar(' ', 1);
}

void play_music()
{
    // music plays
            // musical note is a PWM signal at 50% duty and at some period

            /**
```

```
             * half     2       16384
             *
             * quarter  4       8192
             *
             * 0    a        440     74
             * 1    b flat   466     70
             * 3    b        494     66
             * 4    c        523     63
             * 5    c sharp  554     59
             * 6    d        587     56
             * 7    e flat   622     53
             * 8    e        659     50
             * 9    f        698     47
             * 10   f sharp  740     44
             * 11   g        784     42
             * 12   a flat   831     39
             * 13   a        880     37
             */

    // EDC, EDC, FED, FED
    play_note('e');
    play_note('d');
    play_note('c');

    __delay_cycles(2000000);

    play_note('e');
    play_note('d');
    play_note('c');

    __delay_cycles(2000000);

    play_note('f');
    play_note('e');
    play_note('d');

    __delay_cycles(2000000);

    play_note('f');
    play_note('e');
    play_note('d');
}

void play_note(char note)
{
    myLCD_showChar(' ', 1);
    myLCD_showChar(' ', 2);

    switch (note)
    {
        case 'a':
            TA1CCR0 = 74;
            myLCD_showChar('A', 1);
            break;
        case 'x':
            TA1CCR0 = 70;
            myLCD_showChar('A', 1);
            myLCD_showChar('b', 2);
            break;
        case 'b':
            TA1CCR0 = 66;
            myLCD_showChar('B', 1);
            break;
        case 'c':
            TA1CCR0 = 63;
            myLCD_showChar('C', 1);
            break;
        case 'z':
            TA1CCR0 = 59;
            myLCD_showChar('C', 1);
            myLCD_showChar('#', 2);
            break;
        case 'd':
            TA1CCR0 = 56;
            myLCD_showChar('D', 1);
            break;
        case 'v':
            TA1CCR0 = 53;
            myLCD_showChar('E', 1);
            myLCD_showChar('b', 2);
            break;
        case 'e':
            TA1CCR0 = 50;
            myLCD_showChar('E', 1);
            break;
        case 'f':
            TA1CCR0 = 47;
            myLCD_showChar('F', 1);
            break;
        case 'w':
            TA1CCR0 = 44;
            myLCD_showChar('F', 1);
            myLCD_showChar('#', 2);
            break;
        case 'g':
            TA1CCR0 = 42;
            myLCD_showChar('G', 1);
            break;
        case 'y':
            TA1CCR0 = 39;
            myLCD_showChar('A', 1);
            myLCD_showChar('b', 2);
            break;
        case 'p':
            TA1CCR0 = 37;
            myLCD_showChar('A', 1);
            break;
        default:
            break;
    }

    TA1CCR2 = TA1CCR0 / 2;
    __delay_cycles(2000000);
}
```