ECE 2049 – Laboratory Report

Embedded Computing in Engineering Design

Worcester Polytechnic Institute

C-Term 2021

Section: C03

Laboratory 2

Music Box!

Submitted by:

Jonathan R. Lopez

ECE Mailbox #178

2/24/2021

Professor: Yarkin Doroz

Co-Instructor: Fatemeh (Saba) Ganji

Introduction:

The purpose of lab 2 is to create a music box using our MSP430F5529, our keypad, our buzzer and our SHRP128 LCD screen. The goal of this lab is to play music using the buzzer. In this lab we will be using the buzzer, the LEDs, the keypad and LCD screen to play music.

Materials:

TI MSP430F5529 Launchpad based lab board.
Jumper wires
Buzzer
Breadboard
Sharp128 LCD Screen
0-9, * and # Keypad
Micro-USB to USB-A cable
Computer with Code Composer Studio ver. 10.2

Discussion and Results:

Getting started:

With the MSP430F5529 already wired up from Lab 0 and Lab 1, we got right into launching Code Composer Studio (CCS).

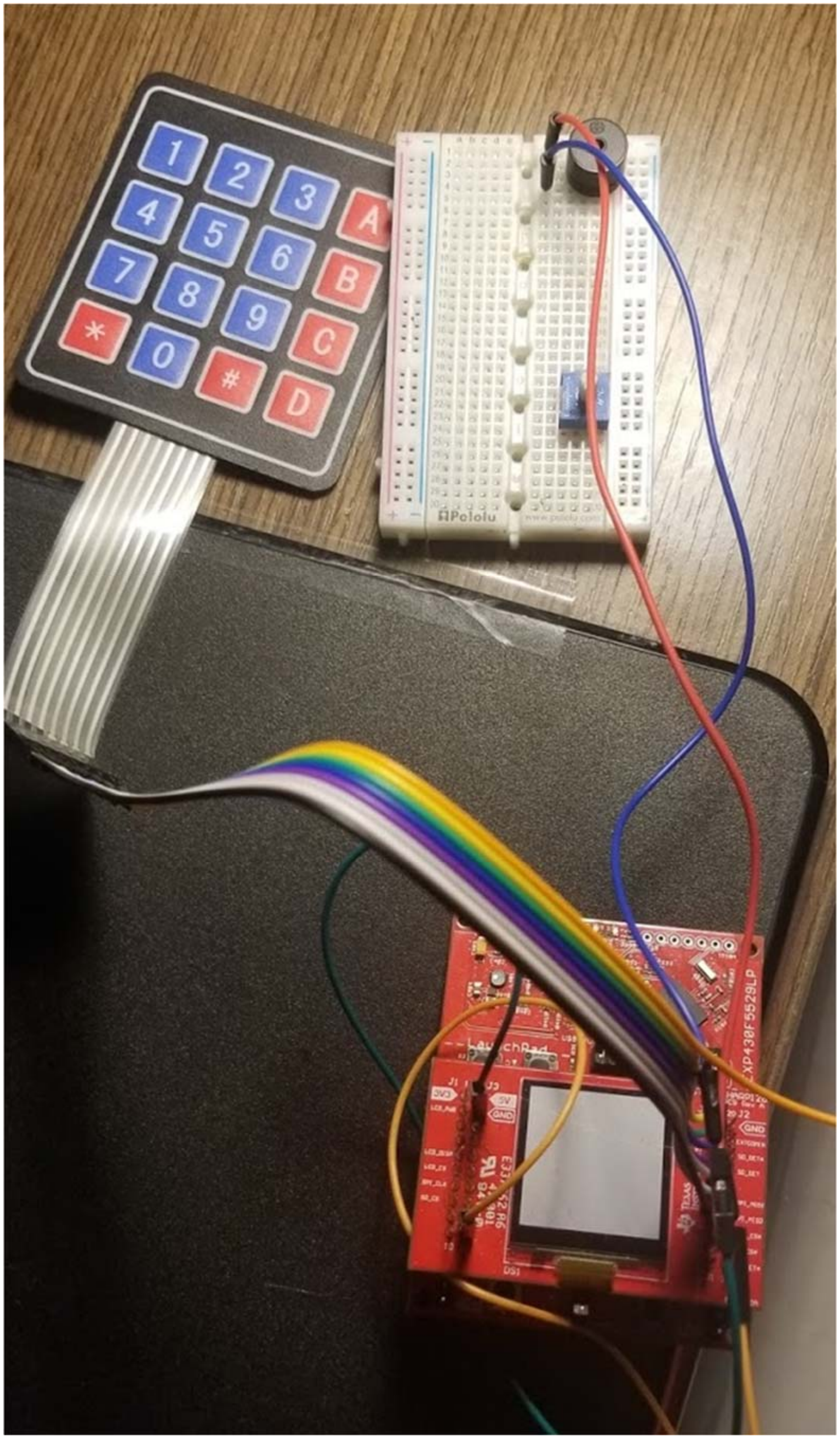
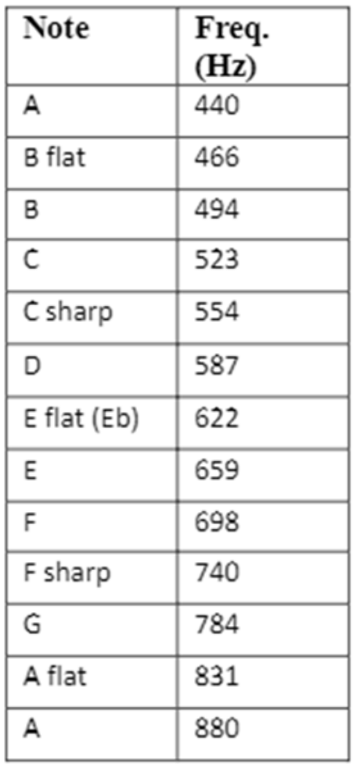


Figure 1: MSP430F5529 already wired up and ready for Lab 2

<u>Approach:</u>



Note	Freq. (Hz)
A	440
B flat	466
B	494
C	523
C sharp	554
D	587
E flat (Eb)	622
E	659
F	698
F sharp	740
G	784
A flat	831
A	880

Figure 1: Note PWM values for the Buzzer

You will need to do some math to convert these frequencies to number of ACLK tics. Discuss your conversion of frequency in Hz to Timer B CCR0 settings in your report

The buzzer buzzes due to a PWM (Pulse Width Modulated) signal is applied to that pin. Instead of getting rid of the BuzzerOn() function I created a new function called BuzzerPWM() in peripherals.c

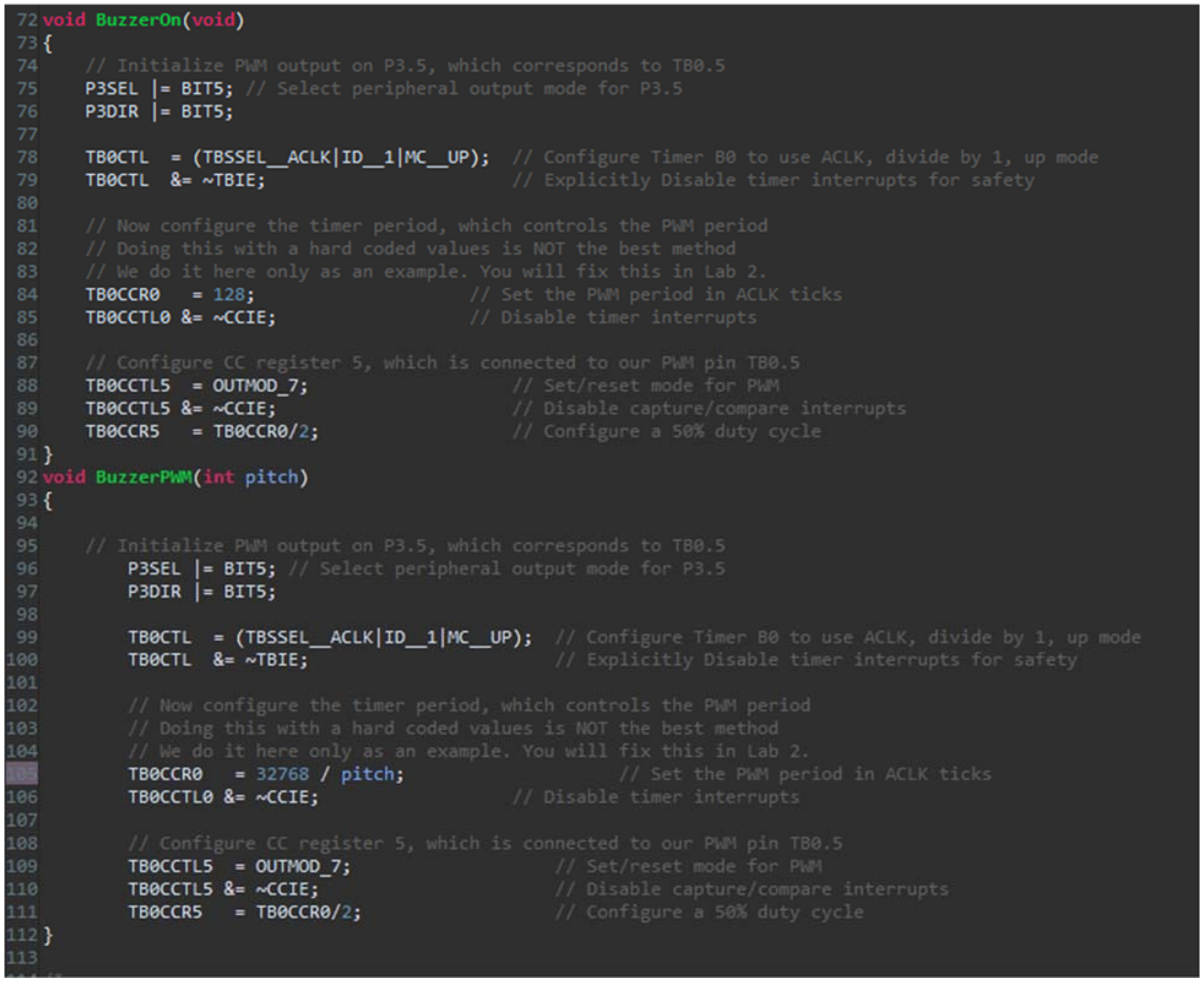
From Lab 1 Simon #8 Bonus:

Altering the function to accept a PWM value wasn't particularly difficult. It now needs an integer input for the pitch you want. In the function body, I replaced the 128 with the frequency of the ACLK which is 32,768Hz and divided it by the integer parameter. In this case it's named pitch. This is for the PWM period in ACLK ticks. This is how you vary the pitch by adjusting the ACLK signal period.

To not clog main.c I put all the notes in a header file notes.h and my songs are in songs.h. This is to keep things organized.

The first song I chose for this lab is "Midnight City" by M83.
The second song I chose for this lab is "Frame of Mind" by Tristam and Braken.

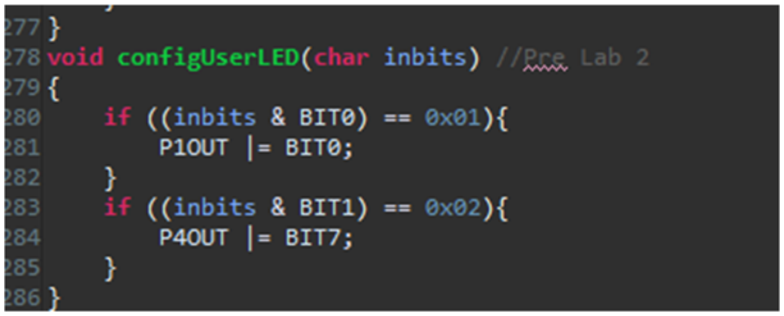
Both are over 28 notes and have over 8 different pitches. They can be found in songs.h



```c
void BuzzerOn(void)
{
    // Initialize PWM output on P3.5, which corresponds to TB0.5
    P3SEL |= BIT5; // Select peripheral output mode for P3.5
    P3DIR |= BIT5;

    TB0CTL  = (TBSSEL__ACLK|ID__1|MC__UP);  // Configure Timer B0 to use ACLK, divide by 1, up mode
    TB0CTL  &= ~TBIE;                       // Explicitly Disable timer interrupts for safety

    // Now configure the timer period, which controls the PWM period
    // Doing this with a hard coded values is NOT the best method
    // We do it here only as an example. You will fix this in Lab 2.
    TB0CCR0   = 128;                    // Set the PWM period in ACLK ticks
    TB0CCTL0 &= ~CCIE;                  // Disable timer interrupts

    // Configure CC register 5, which is connected to our PWM pin TB0.5
    TB0CCTL5  = OUTMOD_7;                   // Set/reset mode for PWM
    TB0CCTL5 &= ~CCIE;                      // Disable capture/compare interrupts
    TB0CCR5   = TB0CCR0/2;                  // Configure a 50% duty cycle
}
void BuzzerPWM(int pitch)
{

    // Initialize PWM output on P3.5, which corresponds to TB0.5
        P3SEL |= BIT5; // Select peripheral output mode for P3.5
        P3DIR |= BIT5;

        TB0CTL  = (TBSSEL__ACLK|ID__1|MC__UP);  // Configure Timer B0 to use ACLK, divide by 1, up mode
        TB0CTL  &= ~TBIE;                       // Explicitly Disable timer interrupts for safety

        // Now configure the timer period, which controls the PWM period
        // Doing this with a hard coded values is NOT the best method
        // We do it here only as an example. You will fix this in Lab 2.
        TB0CCR0   = 32768 / pitch;                  // Set the PWM period in ACLK ticks
        TB0CCTL0 &= ~CCIE;                  // Disable timer interrupts

        // Configure CC register 5, which is connected to our PWM pin TB0.5
        TB0CCTL5  = OUTMOD_7;                   // Set/reset mode for PWM
        TB0CCTL5 &= ~CCIE;                      // Disable capture/compare interrupts
        TB0CCR5   = TB0CCR0/2;                  // Configure a 50% duty cycle
}

```

Figure 2: BuzzerPWM and BuzzerOn functions

Pre-Lab:



```c
}
void configUserLED(char inbits) //Pre Lab 2
{
    if ((inbits & BIT0) == 0x01){
        P1OUT |= BIT0;
    }
    if ((inbits & BIT1) == 0x02){
        P4OUT |= BIT7;
    }
}
```

Figure 3: Prelab

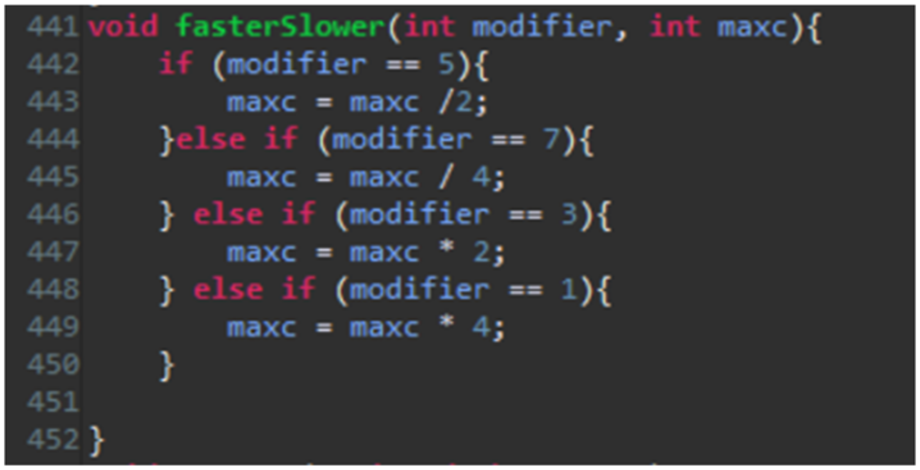
In the final version, the countdown must be measured by Timer A2 and NOT implemented using software delays. Explain the difference between event (or interrupt) driven code and polling. Is your final code strictly event driven or will you use a mix of interrupts and polling? Explain in your report.

In the final version of my project, I had no swDelays at all. It wasn't all event driven though. I did poll the keypad several times during the program. The difference between the two is that with polling you might miss an event. Meanwhile with an interrupt you will never miss it. You are trying to poll for it so you might miss it if it comes in too soon.

How will you control the duration of your notes? Will you do this within the buzzer function or within the main loop? Remember you will need to be checking buttons during the notes. Explain your choice in your report.

With the button checking, I have them being checked only during the playing stage and put them at the beginning of the main function. Therefore, they do not block the code for the buzzer to go off. I will start a timer at the beginning of each song. Then I will compare it to the duration of that note, the song duration and the delta of the time to get it to play for that amount. The duration of each note in songs.h are in 1/16th notes.

I made it so you have to press the same button twice to get to the second level of being fast or slow. If you press 2 it gets fast if you press 2 again it gets faster. If you press 3 it gets slower. If you press 3 again it gets even slower. To do this I adjusted the value for maxcount.



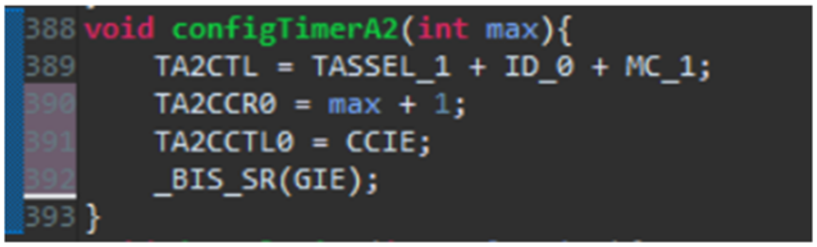
```
void fasterSlower(int modifier, int maxc){
    if (modifier == 5){
        maxc = maxc /2;
    }else if (modifier == 7){
        maxc = maxc / 4;
    } else if (modifier == 3){
        maxc = maxc * 2;
    } else if (modifier == 1){
        maxc = maxc * 4;
    }

}
```

Figure 4: fastSlower Function

Explain in you report why software delay would then no longer work and why you must implement note duration using the timer interrupts.

swDelays will no longer work because swDelays are blocking code. This means nothing else can go on in the program. Only that delay. In this case you have to use event driven programming such as interrupts for the buzzer/note duration.

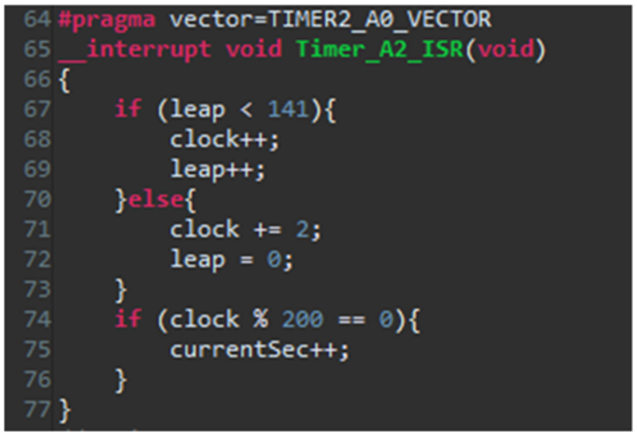
Explain in your report how you setup Timer A2



```c
void configTimerA2(int max){
    TA2CTL = TASSEL_1 + ID_0 + MC_1;
    TA2CCR0 = max + 1;
    TA2CCTL0 = CCIE;
    _BIS_SR(GIE);
}
```

Figure 5: TimerA2 config

In main, I set max to be 163 +1 = 164 which is about 1/200 of a second. I didn't want it to be that large where error could accumulate that large. It also had to be within a .005 resolution. Taking a look at Figure 6. I calculated the leap for the timer to be within the .005 resolution



```c
#pragma vector=TIMER2_A0_VECTOR
__interrupt void Timer_A2_ISR(void)
{
    if (leap < 141){
        clock++;
        leap++;
    }else{
        clock += 2;
        leap = 0;
    }
    if (clock % 200 == 0){
        currentSec++;
    }
}
```

Figure 6: #pragma statement

Summary and Conclusion:

In conclusion, this lab had the purpose of getting students more familiar with the interrupts, timers, the C programming language and the environment of Code Composer Studio as well as get the hang of using the builder, debugger and the variable window within Eclipse/CCS. This concludes this laboratory experiment with the Music Box! on the MSP430F5529 using the buzzer, keypad, and Code Composer Studio.

<u>Appendices</u>

ASCII table:
https://canvas.wpi.edu/courses/23392/files/3432901?wrap=1

Lab 2 Handout:
https://canvas.wpi.edu/courses/23392/files/3416791?wrap=1

Demo Project:
https://canvas.wpi.edu/courses/23392/files/3416786?wrap=1

CCS download:
https://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html

CCS tutorial:
https://www.youtube.com/watch?v=2W0ZHO0vzJE

MSP430 user guide:
https://canvas.wpi.edu/courses/23392/files/3432903?wrap=1

C Review pdf
https://canvas.wpi.edu/courses/23392/files/3439859?module_item_id=542026

ECE 2049 Lab BOM:
https://canvas.wpi.edu/courses/23392/files/3411701?module_item_id=527923

ECE 2049 Course Information:
https://www.wpi.edu/academics/calendar-courses/course-descriptions/17856/electrical-computer-engineering#ECE-2049

ECE2049 C-2021 Lab 2
Sign-off Sheet

***Report due*: 03/03/21**

Student 1: ____________________ ECE mailbox: ______

Student 2: ____________________ ECE mailbox: ______

YOU ARE RESPONSIBLE FOR ***ALL*** THE REQUIREMENTS LISTED IN THE REQUIREMENTS SECTION OF THIS ASSIGNMENT!

		Student 1 Student 2
Power Up and Reset to Welcome mode (Display welcome message)	5	
Timer A2 properly configured with resolution set to 5 ms or less	10	
Countdown to start of song w/ Launchpad LEDs flashing using Timer A2	5	
Buzzer routines with pitch and duration (min of 8 pitches)	10	
Playback of song at least 28 notes long using Timer A2 to measure note duration	20	
Displaying Multiple Songs	10	
Pressing 1: Pause/Play functionality	10	
Pressing 2/3: Playing songs Faster/Slower (at least 5 speed levels)	15	
Pressing 4: finish playing a song and return to main screen to choose songs	5	
Pressing # as a reset at anytime	5	
Using LEDS while playing/pausing a song: Green/Red	5	
Report (answering ***all*** questions from the requirements section)	50	
Total points	150	

BOTH partners MUST be present for ALL sign-offs!!

TA's signature: ________________________ Date: ____________