ESET 369 LAB 4 REPORT

Student name	Kyle Rex and Brink Van Eeden
Group number	38
Lab Section	501
Lab session date	February 24, 2025
Lab instructor	Xin Zhuang

INTRODUCTION

This lab focuses on configuring the MSP430FR5944 Launchpad board to generate specific outputs based on programmed inputs. The objective of System A is to develop a periodic signal at a target frequency using Timer A1 and measure the output with an oscilloscope. System B extends the functionality by integrating an 8-button keypad matrix to trigger different functions, including sound generation through a buzzer and RGB LED control. By implementing software-based polling and interrupt-driven techniques, the lab demonstrates the effective use of microcontroller peripherals for precise signal and input handling. The results validate the functionality of both systems through observed output signals and responses to user inputs.

SYSTEM A

The purpose of System A is to correctly configure pin P6.0 of the MSP430FR5944 Launchpad board to output a periodic signal using a C/C++ code sequence. The frequency of the signal is required to be between 745 to 755 Hz and must be measured using an oscilloscope and a voltage probe. The first step in addressing this problem is making the physical connections between the oscilloscope and the MSP430FR5994 microcontroller. This can be done by connecting two specific pins, P6.0 and GND, of the launchpad board to the oscilloscope voltage probe's positive and negative connections respectively, using male-to-female jumper wires. This setup is depicted in Figure 1.

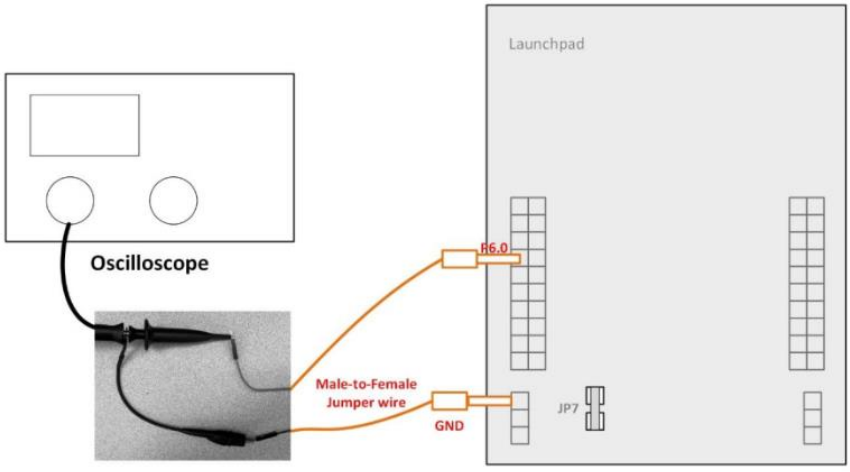

Figure 1: Physical Connection Configuration

Using the code sequences labeled 7.1 (Polling Method), 7.2 (Polling Method), and 8.2 (Interrupt Method), found in: *Learning Embedded Systems with MSP430 FRAM microcontrollers* by B. Hur, as references the code sequence used for this system was created. The code sequence configures and uses Timer A1 on an MSP430 microcontroller to output a periodic signal at pin P6.0 at a frequency of 750 Hz. The program begins by disabling the watchdog timer to prevent unintended resets and unlocking the GPIO pins. It then sets P6.0 as an output and initializes Timer A1 in "up mode" using the SMCLK clock source. The timer's capture/compare register (TA1CCR0) is set to 667 (1 MHz / 2*(750 Hz) = 667), which determines the timer period and achieves the desired toggling frequency of 750 Hz. For calculating TA1CCR0 values for different desired frequencies the equation seen in Equation 1 can be used.

$$TA1CCR0 = \left(\frac{1\ MHz\ Clock}{2*(freqency\ in\ Hz)}\right) \qquad \textbf{Equation 1}$$

In the while(1) loop, the program continuously checks whether the capture/compare interrupt flag (CCIFG) is set. When the flag is triggered (indicating that the timer has counted to 667), the program toggles P6.0 and then clears the flag to restart the process. This approach mimics an interrupt-based system but relies on software polling rather than actual hardware interrupts. With this code sequence complete running it resulted in the oscilloscope output screen seen in Figure 2 which displays the signal being around 752 Hz.

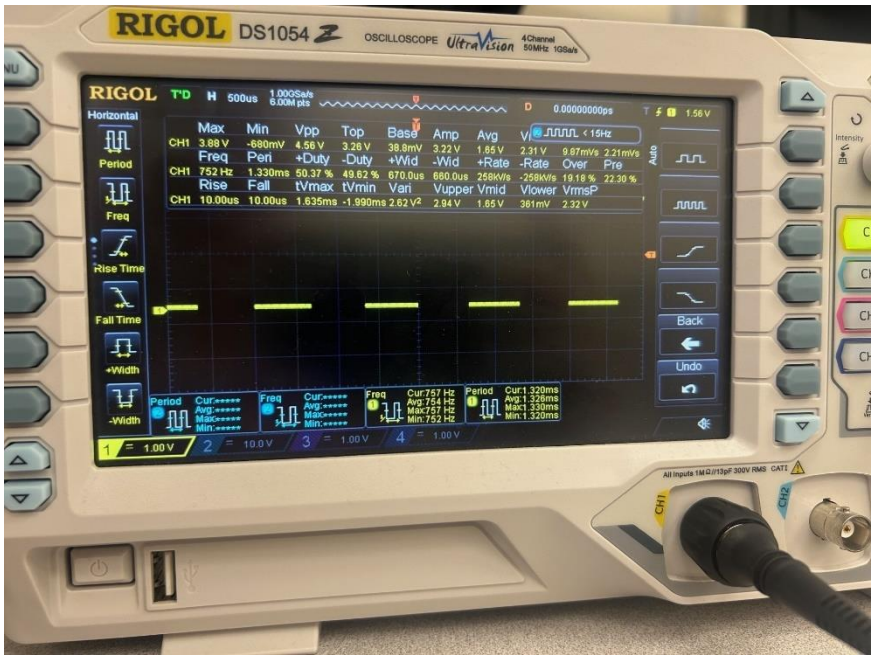

Figure 2: Physical Connection Configuration

SYSTEM B

The purpose of System B is to correctly configure the MSP430FR5944 Launchpad board to perform a variety of functions based on 8 different button inputs on the BH board using a C/C++

code sequence. When the buttons of the keypad matrix, that can be seen in Figure 3, are pressed, the proper function should perform. When the buttons are released, that function should end.

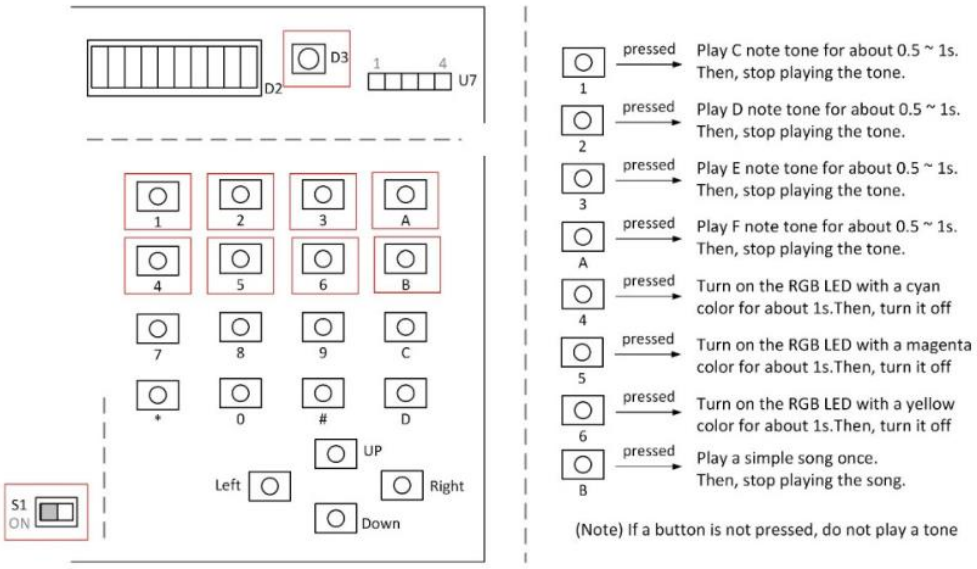

Figure 3: Button Function Configuration

When each button is pressed an associated function should follow. When button 1 is pressed it should play C5 note (523.251 Hz – 956 TA1CCR0 value) tone for about 0.5 to 1 seconds using the buzzer. Then, stop playing the tone. When button 2 is pressed it should play D5 note (587.330 Hz – 851 TA1CCR0 value) tone for about 0.5 to 1 seconds using the buzzer. Then, stop playing the tone. When button 3 is pressed it should play E5 note (659.255 Hz – 758 TA1CCR0 value) tone for about 0.5 to 1 seconds using the buzzer. Then, stop playing the tone. When button A is pressed it should play F5 note (698.457 Hz – 716 TA1CCR0 value) tone for about 0.5 to 1 seconds using the buzzer. Then, stop playing the tone. When button 4 is pressed it should turn on the RGB LED with a cyan (green + blue) color for about 1 second. Then, turn it off. When button 5 is pressed it should turn on the RGB LED with a magenta (red + blue) color for about 1 second. Then, turn it off. When button 6 is pressed it should turn on the RGB LED with a yellow (green + red) color for about 1 second. Then, turn it off. When button B is pressed it should play 4 measures of a simple song once using the buzzer. Then, stop playing the song. If any of these buttons are not pressed nothing should happen, which includes playing any sort of tone.

The first step in addressing this problem is making the physical board connections between the BH board and the MSP430FR5994 microcontroller. This can be done by connecting ten specific pins together using male-to-female jumper wires, checking that all the DIP switches on the BH board are in the correct configuration, and ensuring the microcontroller itself is properly seated on the BH board. The P3 pins control the button columns and the P7 pins control the button rows while the P8 pins control the RGB LED and the P6 pin controls the buzzer. Pin P3.0 is a lateral connection used for buttons 1, 4, 7, * and its port direction must be configured as an input. Pin P3.1 is a lateral connection used for buttons 2, 5, 8, 0 and its port direction must be configured as an input. Pin P3.2 is a lateral connection used for buttons 3, 6, 9, # and its port direction must be

configured as an input. Pin P3.3 is a lateral connection used for buttons A, B, C, D and its port direction must be configured as an input. Pin P7.0 is a horizontal connection used for buttons 1, 2, 3, A and its port direction must be configured as an output. Pin P7.1 is a horizontal connection used for buttons 4, 5, 6, B and its port direction must be configured as an output. With these configurations the eight buttons relevant for this system (Buttons 1, 2, 3, A, 4, 5, 6, and B) are all set up. Pin P8.0 controls the color red of the RGB LED, pin P8.1 controls the color green of the RGB LED, and pin P8.2 controls the color blue of the RGB LED. These are all active low meaning the LED color is on when the input is 0. Pin P6.0 controls the buzzer and must be provided with a periodic signal from the microcontroller to function. This signal is modified by changing the TA1CCR0 value. Different TA1CCR0 values result in different frequencies and each frequency corresponds to a unique musical note that the buzzer will play. This setup follows the configuration that is required for this system that could previously be seen in Figure 3.

Getting the 3 colors (Cyan, magenta, and yellow) required for this lab involved having two of the three original colors (Red, green, and blue) on together at the same time. To get the color cyan the colors green and blue were on together. To get the color magenta the colors red and blue were on together. To get the color yellow the colors green and red were on together. This combination of colors is represented in Table 1.

Table 1: Color Combinations (0 = LED on, 1 = LED off)

	P6.0 (Red)	P6.1 (Green)	P6.2 (Blue)
Cyan	1	0	0
Magenta	0	1	0
Yellow	0	0	1

Using code sequences from previous labs as references the code sequence used for this system was created. One part of the system was coded and tested at a time to ensure that there were no issues with the code or the hardware. The code sequence created meets the prompt's requirements by associating keypad inputs with specific tones and LED color changes. It utilizes Timer_A1 to generate PWM signals for the buzzer and configures GPIO pins for both input (keypad) and output (LEDs and buzzer). The program begins by disabling the watchdog timer and unlocking LPM5 mode to ensure proper GPIO functionality. It then sets up Timer_A1 using SMCLK (1MHz clock), running in Up Mode, and enables interrupts to allow precise timing control for the buzzer. The GPIO configuration designates P6.0 as the buzzer output, P8.0-P8.2 as LED outputs (initialized OFF using active-low logic), and P7.0-P7.1 as row outputs for the keypad matrix. The columns (P3.0-P3.3) are set as inputs with pull-up resistors to detect keypresses when pulled low.

The main loop scans the keypad by sequentially activating each row and checking for button presses in the corresponding columns. When a key is pressed, it plays the correct tone or activates the appropriate LED. If Button 1, 2, 3, or A is pressed, the code sets TA1CCR0 to the appropriate value for C (523.251 Hz), D (587.330 Hz), E (659.255 Hz), or F (698.457 Hz) respectively, playing each note for 0.25 seconds before stopping by setting TA1CCR0 = 0. If Button 4, 5, or 6 is pressed, the corresponding RGB LED turns on with cyan, magenta, or yellow color, staying active for 1

second before turning off. When Button B is pressed, the program plays a 4-measure song, called Careless Wisper, by setting TA1CCR0 values corresponding to musical note frequencies, inserting delays between notes, and ensuring the melody is 4 measures or longer, fulfilling the prompt's requirements. The songs notes can be seen in Figure 4 and its associated values can be seen in Table 2.

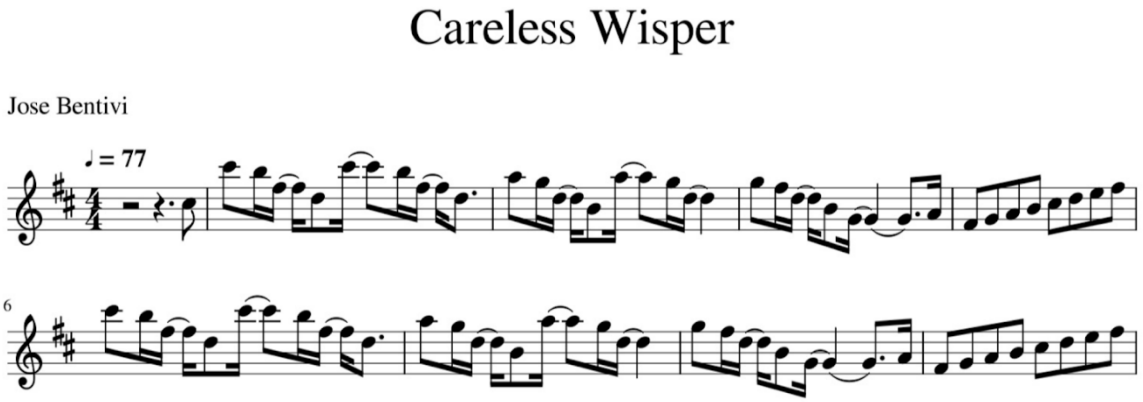

Figure 3: Careless Wisper Musical Notes

Table 2: Careless Wisper Note TA1CRR0 Values

Note Name	Note Frequency	Note CCR0 Value
C#6	1108.731	451
B5	987.767	506
A5	880	568
G5	783.991	638
F#5	739.989	676
E5	659.255	758
D5	587.330	851
C#5	554.365	902
B4	493.883	1012
A4	440	1136
G4	391.995	1276
F#4	369.994	1351

The Timer_A1 interrupt service routine (ISR) toggles P6.0 every time the timer reaches TA1CCR0, generating a square wave tone for the buzzer. The ISR ensures proper PWM signal generation, with the note's frequency controlled by TA1CCR0. After the delay period, the sound

stops by setting TA1CCR0 = 0. This method provides precise tone generation without unnecessary CPU load. Overall, the program successfully fulfills the prompt by correctly implementing tone playback for specific buttons, LED activation with the required colors, and structured song playback. The keypad scanning logic ensures that inputs are read correctly, and the use of Timer_A1-based PWM allows for accurate sound production.

CONCLUSION

The lab successfully met its objectives by demonstrating the correct configuration and operation of both systems. System A produced a stable periodic signal within the required frequency range, as verified using an oscilloscope. System B accurately mapped keypad inputs to predefined functions, generating musical tones and activating LED colors based on user interactions. The lab reinforced the importance of proper GPIO configuration, timer-based signal generation, and input scanning techniques.

REFERENCES

B. Hur, *Learning Embedded Systems with MSP430 FRAM microcontrollers*, 2nd ed., 2023.

K. Rex, *ESET 369 Lab Report 3*, 2025.

K. Rex, *ESET 369 Lab Report 2*, 2025.

K. Rex, *ESET 369 Lab Report 1*, 2025.