**Prompt:**

System B

Write a program in C/C++ to create associated functions by keypad inputs as explained below.

Button 1 pressed: Play C note tone for about 0.5 ~ 1s. Then, stop playing the tone.

Button 2 pressed: Play D note tone for about 0.5~1s. Then, stop playing the tone.

Button 3 pressed: Play E note tone for about 0.5 ~ 1s. Then, stop playing the tone.

Button A pressed: Play F note tone for about 0.5~ 1s. Then, stop playing the tone.

Button 4 pressed: Turn on the RGB LED with a cyan color for about 1s. Then, turn it off.

Button 5 pressed: Turn on the RGB LED with a magenta color for about 1s. Then, turn it off.

Button 6 pressed: Turn on the RGB LED with a yellow color for about 1s. Then, turn it off.

(Note) If a button is not pressed, do not play a tone

For the buttons "1", "2", "3", and "A", you need to play a relevant tone once when the corresponding keypad button is pressed as described in Figure 3. Make sure to program to silence the buzzer after playing the tone once.

For the buttons "4", "5", and "6", you need to turn on the RED LED with Cyan, Magenta, or Yellow colors as described in Figure 3. Make sure to program to turn off the LED after turning on for about 1 second.

I need you to get the buttons for the LEDs working. Buttons 4, 5, and 6.

It is working but it is putting out red when 4 is pressed, green when 5 is pressed, and blue when 6 is pressed. It should be cyan, magenta, yellow for each button respectively.

**Here is what I have done so far (It has Buttons 1, 2, 3, and A working):**

#include <msp430.h>

void play\_tone(unsigned int ccr\_value);

int main(void) {

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

PM5CTL0 &= ~LOCKLPM5; // Unlock GPIO

// Timer Setup

TA1CTL = TASSEL\_2 | MC\_1 | TACLR; // Use SMCLK, Up mode, Clear Timer

TA1CCTL0 = CCIE; // Enable Timer Interrupt

\_\_enable\_interrupt(); // Enable Global Interrupts

// Buzzer Output

P6DIR |= BIT0; // Set P6.0 as output for buzzer

// Keypad Setup

P7DIR |= 0x03; // Set P7.0, P7.1 as output (Rows)

P3DIR &= ~0x0F; // Set P3.0 - P3.3 as input (Columns)

P3REN |= 0x0F; // Enable pull-up resistors

P3OUT |= 0x0F; // Set pull-up mode

while (1) {

P7OUT = 0x02; // Activate Row 1

if ((P3IN & 0x01) == 0) { // Button 1 pressed

play\_tone(956); // C note

}

else if ((P3IN & 0x02) == 0) { // Button 2 pressed

play\_tone(851); // D note

}

else if ((P3IN & 0x04) == 0) { // Button 3 pressed

play\_tone(758); // E note

}

else if ((P3IN & 0x08) == 0) { // Button A pressed

play\_tone(716); // F note

}

}

return 0;

}

void play\_tone(unsigned int ccr\_value) {

TA1CCR0 = ccr\_value; // Set Timer frequency

\_\_delay\_cycles(500000); // Play for ~0.5s

TA1CCR0 = 0; // Stop tone

}

// Timer Interrupt Service Routine

#pragma vector = TIMER1\_A0\_VECTOR

\_\_interrupt void Timer1\_A0\_ISR(void) {

P6OUT ^= BIT0; // Toggle Buzzer output

}

**Example Code using buttons that may help:**

#include <msp430.h>

void delay(){

    \_delay\_cycles(1000);

}

int main(void)

{

    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    PM5CTL0 &= ~LOCKLPM5;       // Unlock GPIO

    // Configure LED pins as outputs (Active Low)

    P5DIR |= 0x0B;  // P5.3 (LED6), P5.1 (LED5), P5.0 (LED4) output

    P8DIR |= 0x07;  // P8.2 (LED3), P8.1 (LED2), P8.0 (LED1) output

    P5OUT |= 0x0B;  // Turn off all LEDs (Active Low)

    P8OUT |= 0x07;

    // Configure keypad rows as output

    P7DIR |= 0x03;  // P7.0, P7.1 as output

    P7OUT |= 0x03;  // Set both rows high initially

    // Configure keypad columns as input

    P3DIR &= ~0x07;  // P3.0, P3.1, P3.2 as input

    P3REN |= 0x07;   // Enable pull-up/pull-down resistors

    P3OUT |= 0x07;   // Enable pull-up resistors

    while (1)

    {

        P7OUT |= 0x03;   // Set both rows high

        P7OUT &= ~0x01;  // Activate row 1

        delay();

        if ((P3IN & 0x01) == 0) { P8OUT &= ~0x01; } else { P8OUT |= 0x01; } // Button 1 -> LED 1

        if ((P3IN & 0x02) == 0) { P8OUT &= ~0x02; } else { P8OUT |= 0x02; } // Button 2 -> LED 2

        if ((P3IN & 0x04) == 0) { P8OUT &= ~0x04; } else { P8OUT |= 0x04; } // Button 3 -> LED 3

        P7OUT |= 0x03;   // Reset rows

        P7OUT &= ~0x02;  // Activate row 2

        delay();

        if ((P3IN & 0x01) == 0) { P5OUT &= ~0x01; } else { P5OUT |= 0x01; } // Button 4 -> LED 4

        if ((P3IN & 0x02) == 0) { P5OUT &= ~0x02; } else { P5OUT |= 0x02; } // Button 5 -> LED 5

        if ((P3IN & 0x04) == 0) { P5OUT &= ~0x08; } else { P5OUT |= 0x08; } // Button 6 -> LED 6

    }

    return 0;

}

**Example Code using RGB LED that may help:**

#include <msp430.h>

void delay\_ms(unsigned int ms) {

    while (ms--) {

        \_\_delay\_cycles(1000);  // Assuming 1 MHz clock

    }

}

int main(void) {

    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    PM5CTL0 &= ~LOCKLPM5;       // Enable GPIO

    // Configure RGB LED pins as outputs (Active Low)

    P6DIR |= BIT0 | BIT1 | BIT2;

    P6OUT |= BIT0 | BIT1 | BIT2; // Turn off LEDs initially

    // Configure S2 button (P5.5) as input with pull-up resistor

    P5DIR &= ~BIT5;

    P5REN |= BIT5;

    P5OUT |= BIT5;

    while (1) {

        if ((P5IN & BIT5) == 0) {  // Button pressed (active low)

            P6OUT &= ~BIT0;  // Red ON

            P6OUT |= BIT1 | BIT2;

            delay\_ms(500);

            P6OUT &= ~BIT1;  // Green ON

            P6OUT |= BIT0 | BIT2;

            delay\_ms(500);

            P6OUT &= ~BIT2;  // Blue ON

            P6OUT |= BIT0 | BIT1;

            delay\_ms(500);

            P6OUT &= ~(BIT1 | BIT2);  // Cyan ON (Green + Blue)

            P6OUT |= BIT0;

            delay\_ms(500);

            P6OUT &= ~(BIT0 | BIT2);  // Magenta ON (Red + Blue)

            P6OUT |= BIT1;

            delay\_ms(500);

            P6OUT &= ~(BIT0 | BIT1);  // Yellow ON (Red + Green)

            P6OUT |= BIT2;

            delay\_ms(500);

        } else {

            P6OUT |= BIT0 | BIT1 | BIT2; // Turn off LED when button is released

        }

    }

}

**Connections for this lab:**

MSP430FR5994 Launchpad

P3.0 Controls buttons 1, 4, 7, \* (lateral)

P3.1 Controls buttons 2, 5, 8, 0 (lateral)

P3.2 Controls buttons 3, 6, 9, # (lateral)

P3.3 Controls buttons A, B, C, D (lateral)

P7.0 Controls buttons 1, 2, 3, A (longitudinal)

P7.1 Controls buttons 4, 5, 6, B (longitudinal)

Tips for buttons:

The port direction for P3.0 to P3.2 need to be configured as input

The port direction for P7.0 to P7.1 need to be configured as output

P3 controls the columns (input)

P7 controls the rows (OUTOUT)

P8.0 Controls red color of the RGB LED

P8.1 Controls green color of the RGB LED

P8.2 Controls blue color of the RGB LED

Hint: This is active low

P6.0 Controls the buzzer that makes the sound and should be sent the signals.