Lab Assignment Checks: Interrupts (Q3) and Keypad (Q1)

Embedded Systems Lab

September 22, 2025

Chapter 1

Introduction

On Thursday's lab session, the following tasks will be checked:

1. **Q3 (Interrupts)** from Lab Assignment 1.
2. **Q1 (Keypad)** from Lab Assignment 2 (for any key except key '1' that was shown in class).

This report provides the required codes, explanations, and schematics.

Chapter 2

Q3: Interrupts (Lab Assignment 1)

2.1 Objective

Use a pushbutton connected to P2.3 to trigger an interrupt. On each button press, toggle the LED on P1.0.

2.2 Code

```c
#include <msp430.h>

#define LED     BIT0     // P1.0 -> LED
#define BUTTON  BIT3     // P2.3 -> Pushbutton

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer
    PM5CTL0 &= ~LOCKLPM5;       // enable GPIO

    P1DIR |= LED;               // set LED as output
    P1OUT &= ~LED;              // LED off initially

    P2DIR &= ~BUTTON;           // P2.3 input
    P2REN |= BUTTON;            // enable resistor
    P2OUT |= BUTTON;            // pull-up

    P2IE  |= BUTTON;            // enable interrupt
    P2IES |= BUTTON;            // falling edge
    P2IFG &= ~BUTTON;           // clear flag

    __bis_SR_register(GIE);     // enable global interrupts

    while(1);                   // idle, wait for ISR
}

// ISR for Port 2
#pragma vector=PORT2_VECTOR
__interrupt void PORT2_ISR(void)
```

```
{
    if (P2IFG & BUTTON) {
        P1OUT ^= LED;           // toggle LED
        P2IFG &= ~BUTTON;       // clear flag
    }
}
```

2.3 Explanation

- The button is pulled up internally, so pressing pulls P2.3 low.
- Interrupt is triggered on falling edge.
- ISR toggles LED state on each press.

Chapter 3

Q1: Keypad (Lab Assignment 2)

3.1 Objective

Scan a 4x4 matrix keypad. Configure rows as outputs and columns as inputs with pull-ups. Detect a key press and toggle LED on P1.0. Requirement: **demonstrate any key except key '1'** (here we use key D4).

3.2 Code

```c
#include <msp430.h>

void keypad_init();

void keypad_init()
{
    // LED
    P1DIR |= BIT0;

    // Rows: P1.3, P1.6, P1.7, P2.4
    P1DIR |= (BIT3 | BIT6 | BIT7);
    P2DIR |= BIT4;

    // Cols: P2.5, P2.6, P3.1, P3.2
    P2DIR &= ~(BIT5 | BIT6);
    P3DIR &= ~(BIT1 | BIT2);

    // Pull-ups
    P2REN |= (BIT5 | BIT6);
    P3REN |= (BIT1 | BIT2);
    P2OUT |= (BIT5 | BIT6);
    P3OUT |= (BIT1 | BIT2);
}

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    keypad_init();

    while (1)
    {
        // Drive Row1 low, others high
        P1OUT &= ~BIT3;
        P2OUT |= BIT4;
        P1OUT |= (BIT6 | BIT7);

        // Check if D4 (mapped to P3.2) is pressed
        if (!(P3IN & BIT2))
        {
            P1OUT ^= BIT0;          // toggle LED
            __delay_cycles(100000);
            keypad_init();          // reinitialize
        }
    }
}
```

3.3 Explanation

- Each row is driven low in turn.
- Columns are read. If a column input is low, the key at (row,col) is pressed.
- In this example, key D4 corresponds to Row1 + Col4 (P1.3 + P3.2).
- On press, LED toggles.

3.4 Keypad Schematic

The following schematic shows the 4x4 keypad layout (rows vs columns):

Row1 (P1.3)	1	2	3	A
Row2 (P1.6)	4	5	6	B
Row3 (P1.7)	7	8	9	C
Row4 (P2.4)	*	0	#	**D**

Col1 (P2.5) Col2 (P2.6) Col3 (P3.1) Col4 (P3.2)

The figure shows the keypad as a 4x4 matrix with rows connected to outputs and columns connected to inputs. In this example, key **D4** (Row4 + Col4) is used.

Chapter 4

Conclusion

- Q3 demonstrated the use of GPIO interrupts: button press toggled LED via ISR.
- Q1 demonstrated scanning of a 4x4 keypad: detecting a pressed key and toggling LED.
- Schematic and code illustrate how rows and columns map to MSP430 pins.