CPE 325: Intro to Embedded Computer System

Lab03

Digital I/O Ports, Experimenter Board Hardware, Hardware Debugging

Submitted by: Hannah Kelley

Date of Experiment: September 10, 2025

Report Deadline: September 17, 2025

Demonstration Deadline: September 17, 2025

Theory

Topic 1: Debouncing

a) When a mechanical switch is pressed or released, the internal contacts do not transition cleanly from one state to another. Instead, they physically vibrate or "bounce" for a short period of time, typically a few milliseconds. This bouncing produces multiple rapid on-off transitions instead of a single, clean signal change. These unintended transitions can cause a digital circuit to register multiple button presses from a single physical action. To mitigate this problem debouncing is used. Debouncing ensures that only one logical transition is recorded for each physical button press or release. Debouncing can be done at the hardware or software level. On the hardware level, special circuits are employed to get rid of the messy signal bouncing generates. On the software level, short delays are implemented to ensure a button was pressed or released before continuing with the expected behavior.

Topic 2: Software Delay

a) In this experiment, debouncing was implemented using a software delay. When the program detects a change in the switch state, it does not immediately register the input as valid. Instead, it introduces a short pause in execution—20 milliseconds in this case—to allow the mechanical contacts within the switch to settle. After the delay, the switch state is checked again. If the switch remains in the same state, the input is confirmed as a valid press or release; otherwise, the change is ignored. This method prevents the program from misinterpreting the rapid on-off transitions caused by bouncing as multiple button presses.

Results & Observation

Program 1:

Program Description:

The objective of this experiment was to write a C program to interface two switches (SW1 and SW2) as inputs and two LEDs (LED1 and LED2) as outputs. At program start, LED1 is OFF and LED2 is ON. The system responds to switch inputs as follows:

- SW1 held: LED2 turns OFF and LED1 blinks at 4 Hz.
- SW2 held: LED1 turns ON and LED2 blinks at 8 Hz.
- No switches pressed: LEDs return to the original state (LED1 OFF, LED2 ON).
- Bonus condition: If both switches are held, LED1 and LED2 blink simultaneously at 2 Hz. Releasing either switch restores the appropriate single-switch or idle behavior.

Report Questions:

1. **Delay loop calculations**

   The MSP430 experiment boards operate with clock cycles that are 1µs long. This means that the delay loops pause program execution for a total time equal to the number of iterations multiplied by the clock cycle duration.

   For the 20ms debouncing loops, I implemented for-loops with 20,000 iterations. This timing is verified since 20,000 iterations multiplied by the 1µs clock cycle duration yields a total delay of 20ms.

   For the 2 Hz blinking LEDs, I implemented a for-loop with 12,500 iterations. This timing is verified since 12,500 iterations multiplied by the 1-µs clock cycle duration yields a total delay of 125ms. Although a 2 Hz signal has a period of 500ms, the LEDs alternate between ON and OFF states within that cycle. This means each LED is ON for only a quarter of the full period (125 ms), then OFF for another 125ms, while the opposite LED is lit. As a result, the alternating pattern requires a 125ms delay rather than the full 250ms.

   The 4 Hz and 8 Hz blinking LEDs work similarly. For the 4 Hz for-loop, I used 6,250 iterations, since 6,250 iterations multiplied by the 1-µs clock cycle duration yields a total delay of 6.25 ms. For the 8 Hz for-loop, I used 3,125 iterations, which gives a total delay of 3.125 ms.

Program Flowchart:

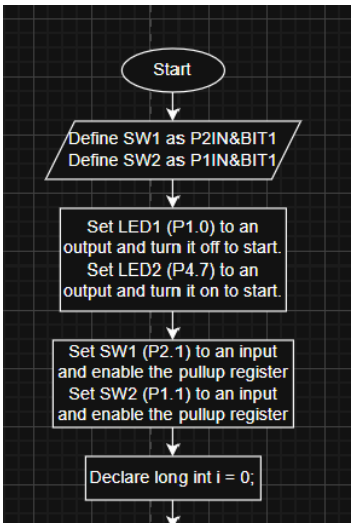

Figure 01: Program 1 Flowchart Part 1

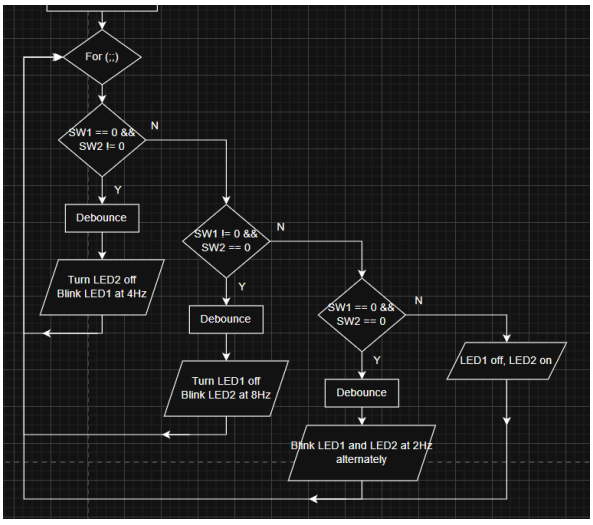

Figure 02: Program 1 Flowchart Part 2

Appendix

Table 01: Program 1 Source Code

```
/*-------------------------------------------------------------------------
 * File:        Lab03_P1.c
 * Description:
 *
 * Board:       5529
 * Input:       switch presses
 * Output:      blinking LEDs of various speeds
 * Author:      Hannah Kelley
 * Date:        September 10, 2025
 *-------------------------------------------------------------------------*/
#include <msp430.h>
#include <stdio.h>

#define SW1 P2IN&BIT1
#define SW2 P1IN&BIT1

int main(void) {
    // stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;
    // set I/O ports
    P1DIR |= BIT0;  // LED1 = output
    P4DIR |= BIT7;  // LED2 = output
    P2DIR &= ~BIT1; // switch1 = input
    P2REN |= BIT1;  // enable pull-up register at P2.1
    P2OUT |= BIT1;  // switch1 = output
    P1DIR &= ~BIT1; // switch2 = input
    P1REN |= BIT1;  // enable pull-up register at P1.1
    P1OUT |= BIT1;  // switch2 = output
    // starting LED positions LED1 = OFF LED2 = ON
    P1OUT &=~ BIT0;  // LED1 = OFF
    P4OUT |= BIT7;   // LED2 = ON

    // event loop
    long int i = 0;
    for (;;) { // infinite loop
        if (((SW1) == 0) && ((SW2) != 0)) { // only SW1 pressed
            // LED1 blinks at 4Hz, LED2 off
            for (i = 2000; i > 0; i--); // 20 ms delay, debouncing
            if (((SW1) == 0) && ((SW2) != 0)) {
                P4OUT &= ~BIT7; // LED2 off
            }
            while (((SW1) == 0) && ((SW2) != 0)) {
                for (i = 6250; i > 0; i--); // 0.125 s delay toggle
                P1OUT ^= BIT0; // LED1 toggle
            }
            P1OUT &= ~BIT0;  // LED1 = OFF
            P4OUT |= BIT7;   // LED2 = ON
        }
        else if (((SW1) != 0) && ((SW2) == 0)) {
            // LED1 on, LED2 blinks at 8Hz
            for (i = 2000; i > 0; i--); // 20 ms delay, debouncing
            if (((SW1) != 0) && ((SW2) == 0)) {
                P1OUT |= BIT0;  // LED1 = ON
```

```
            }
            while (((SW1) != 0) && ((SW2) == 0)) {
                for (i = 3125; i > 0; i--);
                P4OUT ^= BIT7; // LED2 toggle
            }
        }
        else if (((SW1) == 0) && ((SW2) == 0)) {
            // both LEDs blink ALTERNATELY at 2Hz
            for (i = 2000; i > 0; i--); // 20 ms delay, debouncing
            while (((SW1) == 0) && ((SW2) == 0)) {
                for (i = 12500; i > 0; i--);
                P1OUT ^= BIT0;
                P4OUT ^= BIT7;
            }
        }
        else {
            // LED1 off, LED2 on
            P1OUT &= ~BIT0;  // LED1 = OFF
            P4OUT |= BIT7;   // LED2 = ON
        }
    }
    return 0; // end of program
}
```