Embedded Systems Lab

CPE 325-02

Button Pressing on the MSP430

By: David Thornton

Lab Date: September 3, 2020

Lab Due: September 15, 2020

Demonstration Due: September 15, 2020

Introduction

This lab reviews basic C programming skills and explores using the MSP430 buttons as inputs and LEDs as outputs.

Theory

Topic 1: Debouncing

- “[Debouncing] limits the rate at which a function gets invoked.” - Source
- Note that the source is talking about JavaScript, but the concept is the same.

Topic 2: Software delay

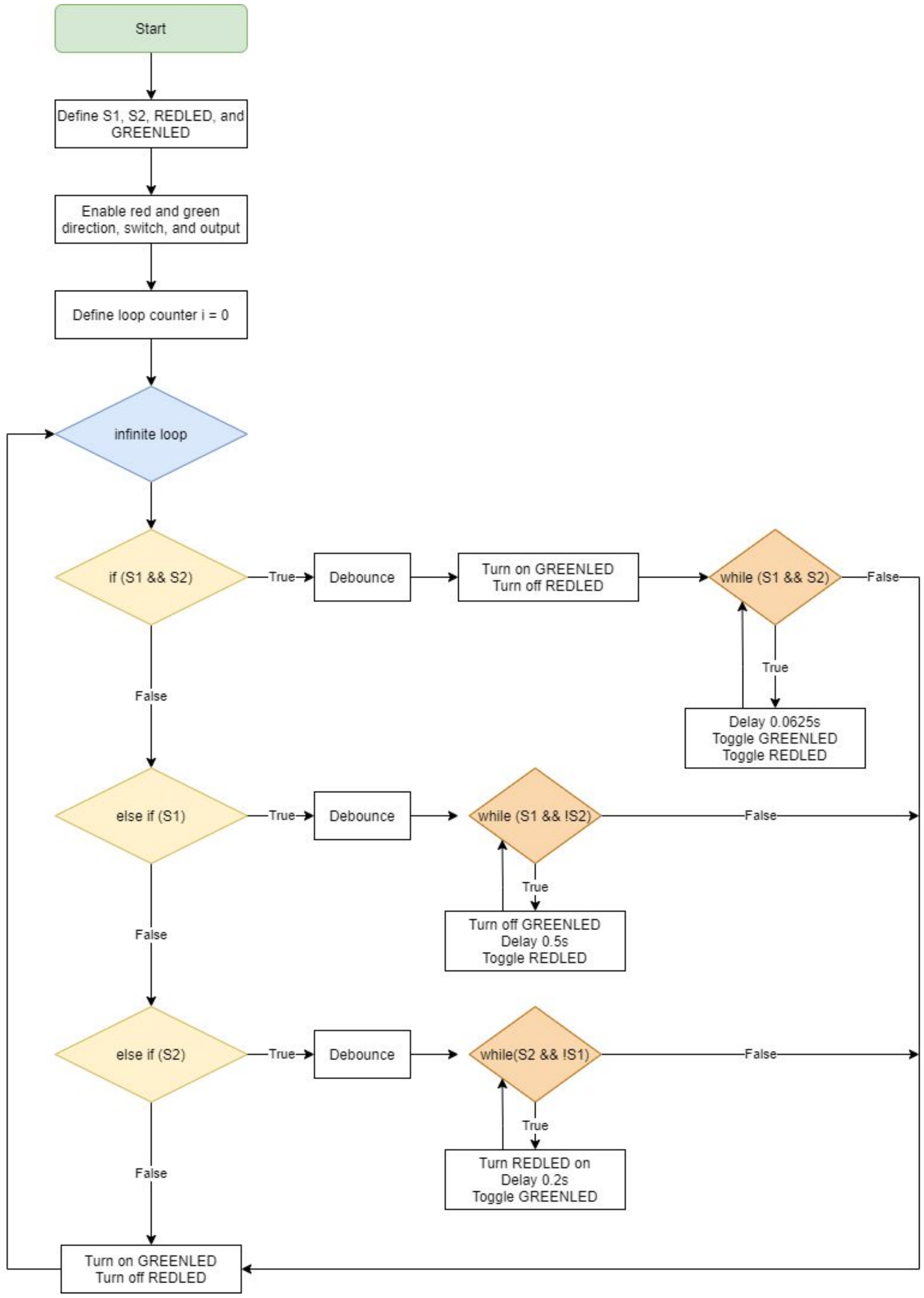
- “A software delay is easier to implement and may be sufficient if it's just a very short delay which is not significantly interrupting any other task in the main sequential code processing path.” - Source

Lab Assignment

1. Write a program in C that achieves the following tasks:
    a. Interface SW1 and SW2 as inputs.
    b. Interface LED1 and LED2 as outputs. LED1 should be OFF and LED2 should be ON at the beginning of the program.
    c. Detect pressing of SW1 and/or SW2.
        i. If SW1 is held, turn off LED2 and blink LED1 at 2Hz. (You should show your calculation for exact timing generation in your report and present it to the instructor during the demo. Please look at demo 2 for hint).
        ii. If SW2 is held, turn on LED1 and blink LED2 at 5Hz. (You should show your calculation for exact timing generation in your report and present it to the instructor during the demo).
        iii. If none of the switches are pressed, the LEDs should go back to the original state where LED1 is OFF and LED2 is ON.
2. Bonus: If SW1 and SW2 are both held, blink LED1 and LED2 alternatively at 8Hz. LEDs should go back to the original state as described in 1b above, if switches are released. State of both switches being pressed can be achieved either by pressing both the switches at once, or by pressing a switch when holding the other one. (For e.g., pressing SW1 when holding SW2 should blink both LEDs at 8Hz. Releasing any one of the switches should meet the condition as described in 1c above.)

Note: Implementation of the bonus question needs to be as an extension of original assignment instead of as a separate program.

Flow Chart



Observations

The program satisfies the assignment requirements.

Conclusion

This lab expanded my knowledge of bit operations in C, hardware and software delays, Code Composer Studio, and how to run and debug a program on the MSP-EXP430F5529LP. The most significant issues I faced during this lab were related to the extra credit, specifically making sure that (S1 && S2) is the first check. Demo link

Appendix

Appendix 1: Lab3.c

```
/*------------------------------------------------------
* File: Lab3.c
* Description: Uses S1 and S2 to blink LED's
* Input: Buttons on MSP-EXP430F5529LP
* Output: LED lights blinking
* Author: David Thornton
* Lab Section: 2
* Date: September 15, 2020
* *--------------------------------------------------*/

#include <msp430.h>
#define S1 ((P2IN & BIT1) == 0)
#define S2 ((P1IN & BIT1) == 0)
#define REDLED 0x01                 // LED1 - Mask for BIT0 (0000_0001b)
#define GREENLED 0x80                 // LED2 - Mask for BIT7 (1000_0000b)
void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;          // Stop watchdog timer
   P1DIR |= REDLED;                // Set P1.0 to output direction (0000_0001b)
   P1REN |= BIT1;                 // Enable the pull-up resistor at P1.1
   P1OUT |= BIT1;                 // Turn P1 output on
   P4DIR |= GREENLED;                // Set P4.7 to output direction (1000_0000b)
   P2REN |= BIT1;                 // Enable the pull-up resistor at P2.1
   P2OUT |= BIT1;                 // Turn P2 output on

   unsigned int i = 0;               // Loop counter
   while(1)                      // Infinite loop
   {
      if (S1 && S2)                 // If S1 and S2 are both pressed
      {
         for (i = 1000; i > 0; i--); // Debounce 10 ms
         P4OUT |= GREENLED;          // Turn on GREENLED
         P1OUT &= ~REDLED;           // Clear REDLED
         while(S1 && S2)             // Blink REDLED and GREENLED alternately at 8Hz (0.125 s)
         {
            __delay_cycles(62500);  // Delay of ~0.0625 s (0.125 s / 2)
            P4OUT ^= GREENLED;      // Toggle GREENLED
            P1OUT ^= REDLED;        // Toggle REDLED
         }
      }
      else if (S1)                  // If only S1 is pressed
      {
         for (i = 1000; i > 0; i--); // Debounce 10 ms
         while (S1 && !S2)
         {
            P4OUT &= ~GREENLED;     // Turn GREENLED off
            __delay_cycles(500000); // Delay of ~0.5 s (2Hz)
            P1OUT ^= REDLED;        // Toggle REDLED
         }
      }
      else if (S2)                  // If only S2 is pressed
      {
         for (i = 1000; i > 0; i--); // Debounce 10 ms
         while(S2 && !S1)
         {
            P1OUT |= REDLED;        // Turn REDLED on
            __delay_cycles(200000); // Delay of ~0.2 s (5Hz)
            P4OUT ^= GREENLED;      // Toggle GREENLED
         }
      }
      else
      {
      P4OUT |= GREENLED;             // GREENLED on at begining of program
      P1OUT &= ~REDLED;              // Clear REDLED
      }
   }
}
/*  The processor clock frequency is approximately 1 MHz
*   1/frequency = seconds  1/1MHz = 1 µs = 1000 ms
*   2Hz = 1/2 s = 0.5 s   = 500 ms = 500 * 1000 cycles
*   5Hz = 1/5 s = 0.2 s   = 200 ms = 200 * 1000 cycles
*   8Hz = 1/8 s = 0.125 s = 125 ms = 125 * 1000 cycles */
```