CPE 325: Intro to Embedded Computer System

Lab03

LEDs and Switches

Submitted by: Esther Shore

Date of Experiment: June 7, 2023

Report Deadline: June 13, 2023

Demonstration Deadline: June 19, 2023

**Introduction**

This lab introduces input and output registers and controlling the built in LEDs and switches on the MSP430. The topics of debouncing and software delays are also examined.

**Theory Topics**

1. Debouncing

Debouncing is a method to remedy switch bounce or noise from resulting in unexpected behavior. It is necessary to confirm a valid switch press and prevent accidental or unintended changes. A debounce can be implemented by adding a delay after the condition and then rechecking the condition after the delay. Then, while the condition is satisfied and remains stable, an action can be taken when the switch is held.

1. Software delay

A software delay can be used to delay the execution of commands, and it can be implemented through multiple methods. One way to create a delay is using the \_\_delay\_cycles function, which takes the number of clock cycles to delay by as a parameter. A delay of 1000 clock cycles is equivalent to approximately 1ms. Another way to create a delay is using a for loop. Each iteration of a for loop is 10 clock cycles and thus, 100 iterations of a for loop is representative of about 1ms delay.

**Program 1**

***Program Description:***

This program takes switch inputs and manipulates the LEDs accordingly. To take switch inputs properly, debouncing was implemented to ensure expected behavior and to confirm the input. If-else statements were used to detect the different input states and while loops were used to detect if the switches were pressed and held. When S1 is held, LED1 blinks at 2Hz while LED2 is OFF. When S2 is held, LED2 blinks at 5Hz and LED1 is ON. Otherwise, LED1 is OFF and LED2 is ON.

For each of the blink frequencies, the calculations for the delays are shown below.

Calculations:

The period is equivalent to the inverse of the frequency.

2Hz: T=1/f => T = ½ = 0.5s = 500ms = 250ms ON and 250ms OFF

5Hz: T=1/f => T = ⅕ = 0.2s = 200ms = 100ms ON and 100ms OFF

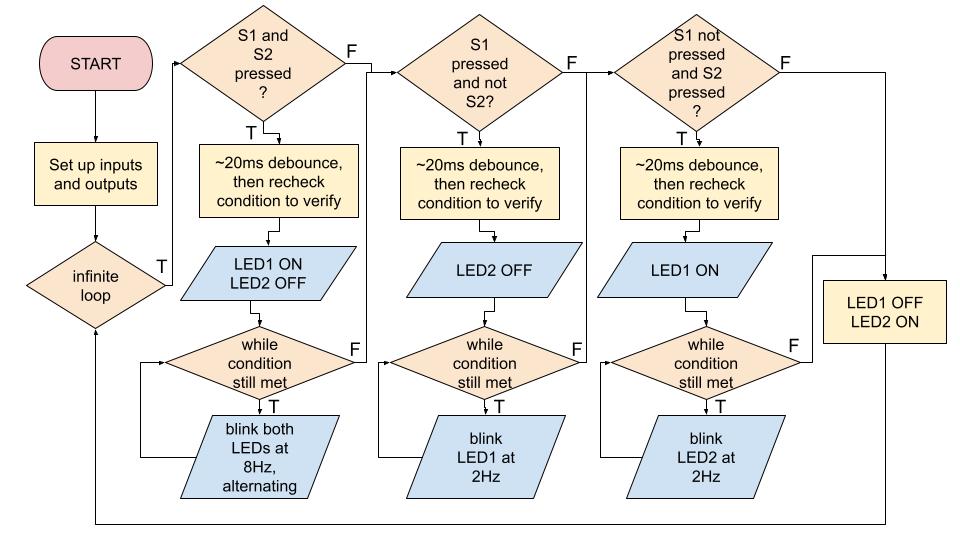
8Hz: T=1/f => T = ⅛ = 0.125s = 125ms = 62.5ms ON and 62.5ms OFF

Given a specific period that represents the time it takes for the LED to complete one blink cycle, we divide it by two to get the amount of time the LED would be ON and OFF.

***Explanation of Results:***

The calculations for the delays make sense because in order to blink faster, the LED must be on for less time. All LEDs respond as expected to the switch inputs. A flowchart of the program logic is displayed below.

***Program Flowchart:***



**Figure 1:** Program 1 + Bonus Flowchart

**Bonus**

***Program Description:***

This program is an extension of Program 1 that blinks both LEDs at 8Hz alternating when both switches are pressed and held. The source code is located in the Appendix and the delay calculation was expressed above. The flow of control for this code is expressed in the existing flowchart.

**Conclusion**

Overall, this lab was completed with minimal issues. The main difficulty that was found was detecting the proper switch inputs. This was resolved by confirming that the other switch was not pressed when detecting each individual switch. The lab helped gain a lot of familiarity about the MSP430 I/O and manipulating those outputs through ON, OFF, and toggle using bitwise OR, AND, and XOR operators.

***Appendix:***

**Table 1:** Program 1 + Bonus Source Code

| /\*------------------------------------------------------------------------------  \* File: Lab03\_P1.c  \* Function: Manipulates the MSP430 LEDs based on the button presses.  \* Description: The default state is LED1 OFF and LED2 ON; If S1 is held,  \* LED2 OFF and blink LED1 at 2Hz, if S2 is held, LED1 ON and  \* blink LED2 at 5Hz, if S1 and S2 are held, LED1 and LED2  \* blink alternatively at 8Hz  \* Input: S1 and S2 button presses  \* Output: Blinking LEDs  \* Author(s): Esther Shore  \* Date: June 6, 2023  \* ---------------------------------------------------------------------------\*/  #include <msp430.h>  #define S1 P2IN&BIT1  #define S2 P1IN&BIT1  void main(void) {  WDTCTL = WDTPW + WDTHOLD; // stop watchdog timer  P1DIR |= BIT0; // set P1.0 to output direction  P4DIR |= BIT7; // set P4.7 to output direction  P2DIR &= ~BIT1; // set P2.1 as input for S1 input  P2REN |= BIT1; // enable pull-up resistor at P2.1  P2OUT |= BIT1;  P1DIR &= ~BIT1; // set P1.1 as input for S2 input  P1REN |= BIT1; // enable pull-up resistor at P1.1  P1OUT |= BIT1;  unsigned int i = 0;  while(1) {  if ((S1) == 0 && (S2) == 0) {  for (i = 2000; i > 0; i--); // ~20ms debounce  if ((S1) == 0 && (S2) == 0) { // set initial states for alternating  P1OUT |= BIT0; // LED1 ON  P4OUT &= ~BIT7; // LED2 OFF  }  while ((S1) == 0 && (S2) == 0) {  \_\_delay\_cycles(62500); // delay 62.5ms to blink at 8Hz  P1OUT ^= BIT0; // toggle LED1  P4OUT ^= BIT7; // toggle LED2  }  } else if ((S1) == 0 && (S2) != 0) { // if S1 pressed  for (i = 2000; i > 0; i--); // ~20ms debounce  if ((S1) == 0 && (S2) != 0) {  P4OUT &= ~BIT7; // LED2 OFF  }  while ((S1) == 0 && (S2) != 0) {  \_\_delay\_cycles(250000); // delay 250ms to blink at 2Hz  P1OUT ^= BIT0; // toggle LED1  }  } else if ((S1) != 0 && (S2) == 0) { // if S2 is pressed  for (i = 2000; i > 0; i--);  if ((S1) != 0 && (S2) == 0) {  P1OUT |= BIT0; // LED1 ON  }  while ((S1) != 0 && (S2) == 0) {  \_\_delay\_cycles(100000); // delay 100ms to blink at 5Hz  P4OUT ^= BIT7; // toggle LED2  }  } else { // otherwise default state  P1OUT &= ~BIT0; // LED1 OFF  P4OUT |= BIT7; // LED2 ON  }  }  } |
| --- |