Introduction to Mechatronics (ME/AE 6705)

Lab Assignment 3

Writing your first program

Objective

The main objective of this lab is to write your first C program for the MSP432 in CCS using the Driver Library. This lab provides an introduction to programming the microcontroller in C, using the driver library, and some basics of C such as loops and conditional statements.

Deliverables and Grading

To get credit for this lab assignment you must:

1. Submit a typed report as a PDF file to Canvas, answering the questions at the end of the lab along with an appropriate discussion (4 pages max, can be shorter). This is due at the beginning of class on the due date specified on Canvas. (40 points)

2. Demonstrate proper operation of your code to TAs or instructor during office hours or demo hours. You must write your own code for this lab – no lab groups are allowed. (40 points)

3. Submit the commented final version of your code on Canvas (single zip file containing your main.c code and the header file. You should only submit a single file.) (Pass/Fail)

Setup

This lab requires Code Composer Studio and the MSP432 LaunchPad. The lab uses only the onboard electronics of the LaunchPad such as switches (buttons) and LEDs. The MSP432 LaunchPad has two user programmable switches (buttons) and two programmable LEDs. The switches S1 and S2 are connected to GPIOs P1.1 and P1.4 respectively. The red LED (LED1) is connected to pin P1.0. The multicolored LED (LED2) requires 3 pins for its operation. LED2 is connected to GPIOs P2.0, P2.1, P2.2 for the red, green and blue color respectively.

Note: To create a new CCS project for the MSP432P401R target device, and include the DriverLib library in the project, follow the procedure described in Lab Assignment 2. The driverlib folder is available on Canvas. If you are using the MSP432P4111 for this lab, still choose the MSP432P401R target device in CCS.

Problem Statement

Write a program that switches on LED1 by pressing and holding S1, and off by releasing S1. Similarly, LED2 should switch on by pressing and holding S2, and off by releasing S2, i.e. they are on only if the buttons are pressed. LED2 should change color each time S2 is pressed, i.e. if LED2 is red when S2 is pressed first, it should be, for example, green the next time S2 is pressed and blue for the time after that. This sequence may repeat itself.

Background

Driver Library:

This program should be written with the help of the functions defined in the driver library. The driver library is a specialized library created by TI for the MSP series of microcontrollers. It introduces a higher level of abstraction for better readability of your source code. This programming method is an alternative to the typical direct register programming used for many MCU's (we will cover driver library vs direct register access in lecture). The Driver Library User Guide can be found on Canvas. This guide provides all the different modules of the microcontroller supported by the driver library, their associated functions and the inputs and outputs of these functions.

The '`MAP_`' is used at the beginning of each function given in the driver library. This prefix improves speed and performance. The driver library header file 'driverlib.h' has to be included in the project. This header file provides links/access to all the other headers defined in the driver library such as GPIO, Timers, ADC etc. In this lab, the GPIO functions are to be used. All driverlib functions can be accessed simply by including 'driverlib.h' as mentioned earlier. Reference the user guide to determine the functions you would like to use and their syntax.

For example, to configure the switch S1, the following function is used:

```
MAP_GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1,GPIO_PIN1)
```

This function (with above arguments) tells the microcontroller to use pin 0 of port P1 as an input with a pull up resistor.

To assist you with this initial program, a source code file with some example GPIO function calls are included with the lab assignment as an additional reference.

Pull-Up vs Pull-Down Resistors:

Switches are typically configured using pull up resistors as shown in the Figure 1. The pins connected to the switch are pulled high when the switch is not pressed. This can be visualized as the circuit shown in the Figure 1.

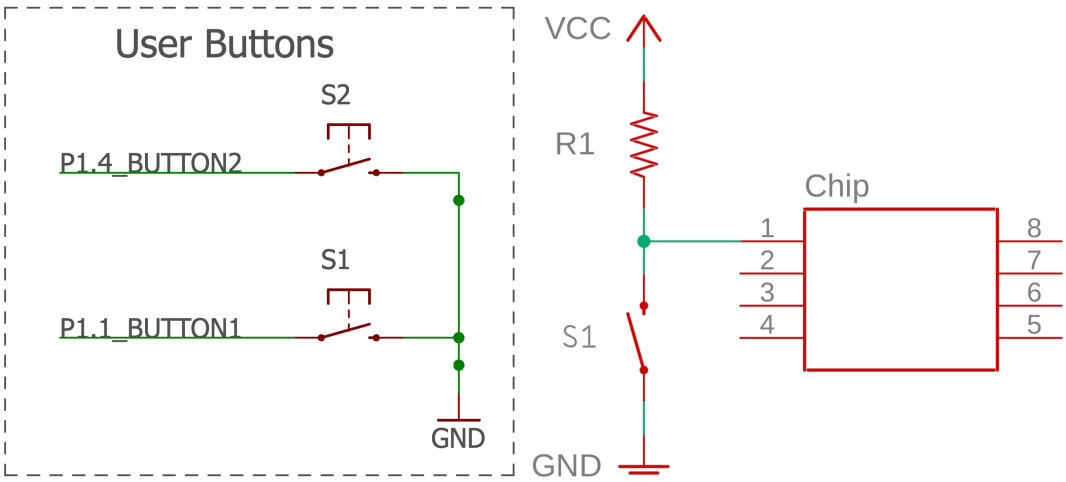

Figure 1: Left: Schematic of S1 and S2 switches on MSP432 (Image courtesy of Texas Instruments), Right: A pull-up resistor, connected to a normally open pushbutton.

When the switch is pressed, the input value read at the input pin is low. Hence, for a normally open switch the pin is at high voltage when the switch is not pressed. Similarly, the pull-down resistor circuit pulls the pin down to the ground in the normal condition. Note: the MSP432 has internal pull-up and pull-down resistors, you do not need to build you own circuit. To make use of these internal resistors use the `MAP_GPIO_setAsInputPinWithPullUpResistor()` and `MAP_GPIO_setAsInputPinWithPullDownResistor()` function calls.

You can choose to configure S1 and S2 as either pull-up or pull-down. This will simply determine whether pins P1.1 and P1.4 are high or low when the button is not pressed. You will need to configure your software program according to the choice you make.

To turn the LEDs on or off, simply set the associated output pin high or low.

Watchdog Timer:

At the beginning of your code, make sure to call the function `MAP_WDT_A_holdTimer()` to hold the watchdog timer. See the example code provided with this lab for an example. We will discuss the watchdog timer in an upcoming lecture.

Requirements

The following features must be present in your program to receive credit:

1. You must create and use at least one additional function (in addition to the main function).

2. You must create your own header file (.h), and use this header file for all #define statements and function prototypes.

3. You must use at least one macro (#define statement).

Questions

1. How should the code be modified if the requirement was to switch off the LED after a small arbitrary delay? (Hint: No timers necessary)

2. Suppose you were to write a program that toggles LED1 whenever S1 is pressed. With proper operation of the program you would expect to see the light turn on when you press and release S1, and then turn off when you press and release S1 again. Suppose you write this program as follows:

```
while(1){

      // Read button
      usiButton1 = MAP_GPIO_getInputPinValue ( GPIO_PORT_P1, GPIO_PIN1 );

      //If button is pressed...
      if ( usiButton1 == GPIO_INPUT_PIN_LOW ) {
            MAP_GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0) ;
      }
}
```

What is one (or more) potential problem(s) with this code? (Hint: See Lecture 2, slide 9). What would be a potential software modification that would fix this problem?