ME4405 - Fundamentals of Mechatronics (Spring 2020)

Lab Assignment Three

Writing your First Program

Due Thursday, February 13rd, 2020

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 report answering the questions at the end of the lab. This is due on Canvas **by 5pm** on the above due date. **(25 points)**
2. Demonstrate proper operation of your code to the TA or instructor during lab or office hours. **(25 points)**
3. Lab assignment quiz which will be taken at the beginning of lab section on Friday, February 14th. **(10 points)**
4. Submit the commented final version of your code on Canvas (with all necessary files in a zipped folder) **(Pass/Fail)**

Procedure:

This lab requires Code Composer Studio and the MSP432 device. The lab uses only the onboard electronics of the device such as switches (buttons) and LEDs. The MSP432 device has two user programmable switches (buttons) and two programmable LEDs. The switches S1 and S2 are connected to pins 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. The LED2 is connected to pins P2.0, P2.1, P2.2 for the red, green and blue color respectively.

Note: For this lab you will need to create a CCS project, include the driverlib.h header file and link the driverlib as you did for Lab 2.

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 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 S2, the function "GPIO_setAsInputPinWithPullUpResistor (GPIO_PORT_P1,GPIO_PIN4)" is used. This function (with above arguments) tells the microcontroller to use pin 4 of port P1 as an input pin with a pull up resistor. S1 and S2 correspond to P1.1 and P1.4, respectively.

NOTE: The prefix 'MAP_' was used at the beginning of each function for previous versions of the MSP driver library. For example, MAP_GPIO_setAsInputPinWithPullUpResistor (GPIO_PORT_P1,GPIO_PIN0) was used instead of the function found in the previous paragraph. The current driver library version does not require this prefix, and CCS is backward compatible (for now) and will still accept functions with or without the prefix. However, we encourage you to code your programs without the prefixes.

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.

NOTE: This code is to be used for reference only in order to get you up and running with MSP C programming. Please do not copy-paste any code when actually coding for this Lab. This can be a potential source of bugs or errors which will be difficult to debug.

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.

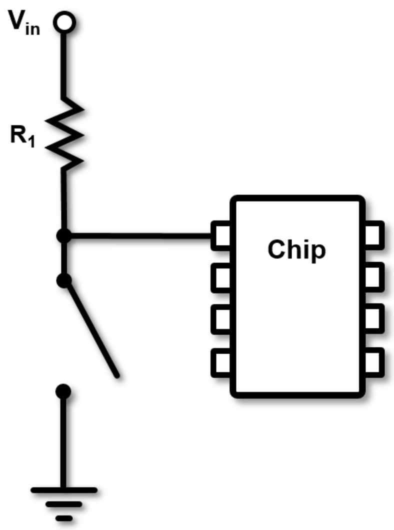

Fig. 1: Pull up resistor circuit

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 resistors; unless you choose to use the pull-down configuration, you do not need to build your own circuit. To make use of these internal resistors use the GPIO_setAsInputPinWithPullUpResistor() and GPIO_setAsInputPinWithPullDownResistor() function calls.

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 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:

Assume that P1.1 has been set up with a pull-up resistor. Please do not worry about whether variables are defined or not. Your only task for this problem is to find faults in the pseudo code.

```
while(1){

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

      //If button is pressed...
      if ( usiButton1 == GPIO_INPUT_PIN_LOW ) {

            GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0) ;

      }

}
```

What are two problems with this code? What would be some potential software modifications that would fix these problems?