Lab Time______________________ Name____________________________

Lab 4. Initializing and Using GPIO Ports in ARM Assembly

Preparation

You will need a LaunchPad and a laptop/computer with Keil uVision5 installed. Download the Lab4 starter project from Canvas.

Starter project Lab4

Purpose

The purpose of this lab is for you to become familiar with using the GPIO ports on the Tiva board. You will write code that initializes the Port F pins connected to the LEDs and pushbuttons on the board.

Introduction

The Tiva Launchpad uses memory mapping to interface with I/O devices. This means that the actual physical devices such as pushbuttons and LEDs are accessed as if they are normal memory locations. The GPIO ports the pushbuttons and LEDs on the board are connected to must first be initialized before they can be used. Attempting to access GPIO pins that have not been correctly initialized can result in a Hard Fault which will effectively cause the board to stop executing your program. You must be very careful when writing to entire ports that you only touch the pins that you are directly using. For this reason, **you should not use the MOV instruction during this lab.**

Procedure

1. Complete the initialization function "PortF_Init" located in file "PortF.s" to initialize both of the pushbutton pins as inputs and the led pins as outputs. The figure on page 144 of your textbook can be used to determine what is connected to each pin. This figure can also be found in the appendix section of the online version of the textbook. **Do not use the "MOV" instruction in any of the code you write.** All of the LEDs should turn on if the initialization function is correct. Demonstrate this to the TA.

Signature [illegible] Date 3/3/20

[illegible] 3/3/2020

2. Now you will create eight simple subroutines in "PortF.s" with the following functionality: (1-3) Toggle each LED, (4) Turn on the red LED, (5) Turn on the blue LED, (6) Turn off all of the LEDs, (7) Return the state of the pushbuttons, and lastly (8) a delay function that is 250ms to 500ms. You may reuse your delay subroutine from earlier labs or write a new one. **Your LED and pushbutton subroutines should use Port F's data register (GPIO_PORTF_DATA_R) and should not use the MOV instruction. Use the same approach as in Part 1 to target individual pins. Make sure you export each subroutine at the top of "PortF.s".**

   Demonstrate to the TA that your subroutines are correct by calling them in the main loop. Use the delay function to make sure you can observe the LEDs changing. Your LED toggle subroutines should be called multiple times to prove that the LEDs are toggling. Use the debugger to show that your pushbutton subroutine is returning the correct button state.

Signature [illegible] Date 3/3/2020

[illegible] 3/3/2020

Lab Time______________________ Name____________________________

3. Use the subroutines from the previous part to write a program with the following behavior depending on which buttons are being held down:

   No buttons held down: Toggle the green LED, others off

   SW1 only: Blue LED on, others off

   SW2 only: Red LED on, others off

   Both: Toggle green. Toggle red and blue each time the green is toggled on

   The behavior for the last one should look something like "all off → all on → green off, red and blue on → green on, red and blue off → start over at all off". Your program should check the button state before each transition so that it can exit the toggling pattern early rather than having to wait for the entire routine to complete before it can stop.

   It will likely be helpful if you first create a flowchart of your design showing the logical flow of what you intend to do. Once you are satisfied then you can move on to coding the design. Demonstrate your program to the TA.

Signature Cole Farrell Date 3/3

Cole Farrell