James Hicks
Michael Meyer
EENG 260 (Lab Session 3) - Lab Report #2
October 17 2023

LAB 2

GOAL

This lab was aimed at being able to turn on an onboard LED on the TM4C board. Additionally the lab requires that we use an onboard switch such that when we press the onboard switch (sw1) the LED turns on and when we let go the LED turns back off.

BACKGROUND AND THEORY

In order to complete this lab it is necessary to know how to configure GPIO and how to read and write from GPIO data.

The first step in configuring and initializing GPIO is to write to the RCGCGPIO register which is detailed on page 340 of the datasheet. You need to set high bits in this register that correspond to the port you want to send the clock signal. This is essentially enabling the port as without the clock being sent to the GPIO port it cannot be used. In this lab we want to use port F so all we need to do is set the 5th bit high.

Secondly you need to set the directions for the GPIO pins that you are using. It's important to note that writing a one to a bit in the GPIODIR register for a port means that you are setting the pin that corresponds to that bit to be an output. It's important to note here that a 0 corresponds to an input and the GPIODIR register is all zeros by default. This means that anywhere you want to have a pin act as an input you don't actually have to write to that ports GPIODIR register as it defaults to input anyway. In this case we want to set PF3 to be an output so we need to set the third bit in the GPIODIR register of port f to be a one.

Next we need to enable a pull up resistor for PF4 (portf pin4). You need to enable a pull up resistor on this pin because this pin is connected to the switch. When we enable a pull up resistor we are able to provide well defined voltage levels at the input so that the board can tell easily when the button is pressed and when it is not pressed. To enable the pull up resistor for PF4 we need to set the fourth bit high in the GPIOPUR register (data sheet 667).

Lastly we need to enable digital for the desired pins. Namely PF3 and PF4. We want the switch to read that it is either pressed or not pressed and we want the LED to either be on or off so digital is the way to go. We want to set both the third and fourth bits high here so we'll write 0x18 to the register.

The above are the steps to initializing most GPIO pins. Some pins are locked and require that you go to the GPIO lock register, put in the unlock code. Enable commit for the port you want to write to and after that you are able to enable pull up resistors and set drive strengths etc. These steps are critical to understand to be able to use GPIO on the TM4C board.

The critical part of this lab is understanding how to read and write data to and from the GPIODATA register for the port. We need to read the value off of PF4 to know whether the switch is pushed down and we need to write data to the LED to be able to turn it on and off accordingly. In order to do this we need to use offsets. The offset you use when reading or writing to the GPIODATA register corresponds to the bits you will actually read from or write to. This process is detailed on page 654 of the data sheet. However the corresponding offset for reading from pin 4 is 0x040 and the offset to write to pin 3 is 0x020.

Putting all of this together with some simple loops to continually poll the switch to determine whether the switch is pressed or not we can complete this lab.

CODE

Below is the initialization code for lab 2

```
;ENABLE CLOCK TO PORT F (340 data sheet)
    LDR R0, SYSCTL_RCGCGPIO_R ;load RCGCGPIO_R into R0
    LDR R1, [R0]              ;load what's at RCGCGPIO_R into R1
    ORR R1, #0x20             ;set the 5th bit high
    STR R1, [R0]              ;stores modded bits into RCGCGPIO_R
    NOP                       ;time for the clock to finish
    NOP

;UNLOCK GPIO PORT F
    LDR R0, GPIO_PORTF_LOCK_R ;load LOCK_R address into R0
    LDR R1, GPIO_UNLOCK_CODE  ;unlock code for GPIO_CR (684 datasheet)
    STR R1, [R0]              ;unlock GPIO_CR

;ENABLE COMMIT FOR PORT F
    LDR R0, GPIO_PORTF_CR_R   ;load CR_R address into R0
    LDR R1, [R0]              ;load what's in CR_R into R1
    ORR R1, #0x10             ;set the 4th bit high
    STR R1, [R0]              ;enable commit for PF4 (bit 4)

;SET DIRECTION (663 data sheet)
    LDR R0, GPIO_PORTF_DIR_R  ;load DIR_R address into R0
    LDR R1, [R0]              ;loads what's at DIR_R into R1
    ORR R1, #0x08             ;set the 3rd bit high
    STR R1, [R0]              ;sets the direction of PF3 to output

;ENABLE PULL UP RESISTOR FOR PF4
    LDR R0, GPIO_PORTF_PUR_R  ;load PUR_R address into R0
    LDR R1, [R0]              ;load what's at PUR_R into R1
    ORR R1, #0x10             ;set 4th bit high
    STR R1, [R0]              ;enables pull up resistor for PF4

;SET DIGITAL ENABLE (682 data sheet)
    LDR R0, GPIO_PORTF_DEN_R  ;load DEN_R address into R0
    LDR R1, [R0]              ;load what's at DEN_r into R1
    ORR R1, #0x18             ;sets the 3rd and 4th bit high
    STR R1, [R0]              ;stores modified DEN_R bits into DEN_R

    AND R1, #0                ;clear R1
```

CONCLUSION

The aim of this lab was to get familiar with the process of initializing, reading from and writing to GPIO on the TM4C123 board. When attempting to get this lab to work I ran into issues with the SYSCTL register. This register is integral to the function of the board and one must be careful when writing to that register. Additionally, for practice and for safety's sake I ended up using the unlock code and enabling

commit for portf. Even though (I think) I technically don't have to be able to use switch 1. A point of difficulty was learning how to calculate which offset I needed to use in order to write to the GPIODATA port. It took some getting used to.