**Name(s) \_\_Dannah Gersh\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ECE/CoE 1188: Cyber-Physical Systems Laboratory**

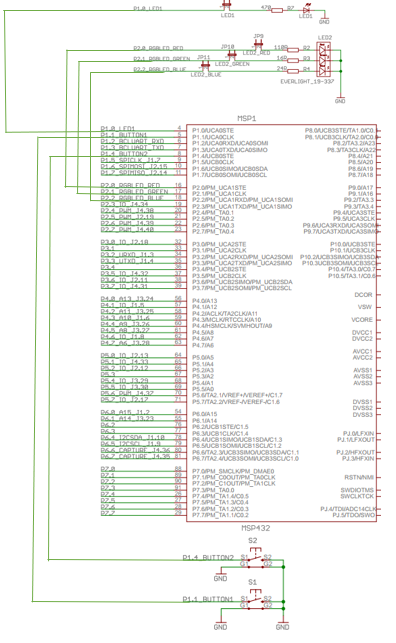
***Lab #1 Worksheet (70 points)***

Fill out all requested items, export to a PDF document and then upload this document to courseweb. Upload separate source files for each of the three parts

# **Part A – Project preparation**

1. 15 pts **Labeled Schematic**

**Include MSP432, push buttons, LEDs, port/pin numbers, etc.**



1. 5 pts **Are the push button switches active-high or active-low? How do you know? Is another component (not listed) necessary in order for this switch to operate properly?**

active low because the signal sent when the button is pressed will be a connection to ground

1. 5 pts **What is the purpose of the jumpers that were included?**

allows the user to disconnect the LEDs and connect their own device to that port if it is needed.

1. 5 pts **Create a table that lists each of these registers for Port #2 and the full 32-bit hexadecimal address of each register. In your table, list the value that each of these registers should be initialized to.**

|  |  |  |
| --- | --- | --- |
| Register | 32 bit address | Initial Value |
| Port 2 Input (P2IN) | 0x4000\_4c01h |  |
| Port 2 Output (P2OUT) | 0x4000\_4c03h |  |
| Port 2 Direction (P2DIR) | 0x4000\_4c05h | 0 |
| Port 2 Resistor Enable (P2REN) | 0x4000\_4c07h | 1 |
| Port 2 Drive Strength (P2DS) | 0x4000\_4c09h | Default |
| Port 2 Select 0 (P2SEL0) | 0x4000\_4c0bh | 0 |
| Port 2 Select 1 (P2SEL1) | 0x4000\_4c0dh | 0 |
| Port 2 Compliment Selection (P2SELC) | 0x4000\_4c17h |  |
| Port 2 Interrupt Edge Select (P2IES) | 0x4000\_4c19h |  |
| Port 2 Interrupt Enable (P2IE) | 0x4000\_4c1bh |  |
| Port 2 Interrupt Flag (P1IFG) | 0x4000\_4c1dh |  |
| Port 2 Interrupt Vector (P1IV) | 0x4000\_4c1eh |  |

# **Part B – PSeudo Code**

1. 15 pts **Write pseudo code for this program. You may use any syntax you wish, but the algorithm should be clear.**

//set up button 1 as an input

P1SEL0.1 <= 0

P1SEL2.1 <= 0

P1DIR.1 <= 0

P1REN.1 <= 1

//set up button 1 as an input

P1SEL0.4 <= 0

P1SEL2.4 <= 0

P1DIR.4 <= 0

P1REN.4 <= 1

//set up RGBLED\_RED as output

P2SEL0.0 <= 0

P2 SEL1.0 <= 0

P2DIR.0 <= 1

P2REN.0 <= 1

//set up RGBLED\_GREEN as output

P2SEL0.0 <= 0

P2 SEL1.0 <= 0

P2DIR.0 <= 1

P2REN.0 <= 1

//set up RGBLED\_BLUE as output

P2SEL0.0 <= 0

P2 SEL1.0 <= 0

P2DIR.0 <= 1

P2REN.0 <= 1

//loop get get input from buttons and set output to LED accordingly

While(1){

if (p1IN.4 == 1 && p1IN.1 == 1){

//lock in standby

P2OUT.0 = 0;

P2OUT.1 = 0;

P2OUT.2 = 1;

}

Else if ((p1IN.4 == 1 && p1IN.1 == 0) || (p1IN.4 == 0 && p1IN.1 == 1)){

//incorrect button combo

P2OUT.0 = 1;

P2OUT.1 = 0;

P2OUT.2 = 0;

}

Else if (p1IN.0 == 1 && p1IN.1 == 0) {

//Correct button combo

P2OUT.0 = 0;

P2OUT.1 = 1;

P2OUT.2 = 0;

}

}

So…

//to set up both buttons

Assign 0x00 to addr 0x4000\_4C0A

Assign 0x00 to addr 0x4000\_4C0C

Assign 0x00 to addr 0x4000\_4C04

Assign 0x12 to addr 0x4000\_4C06

//to set up the three LEDs

Assign 0x00 to addr 0x4000\_4C0B //sel0

Assign 0x00 to addr 0x4000\_4C0D //sel1

Assign 0x07 to addr 0x4000\_4C07 //ren

Assign 0x07 to addr 0x4000\_4C05 //dir

Startloop

Load data from 0x4000\_4c000 //p1in

Get bit 4 and 1

Check if they are both 0

If both 0:

Assign 0x02 to addr 0x4000\_4c03 //p2out, display green LED

Skip to end of loop

If both 1:

Assign 0x04 to addr 0x4000\_4c03 //p2out, display blue LED

Skip to end of loop

If different values:

Assign 0x01 to addr 0x4000\_4c03 //p2out, display red LED

Endloop -> branch to start loop

# **Part D – WRITING YOUR ASSEMBLY LANGUAGE PROGRAM**

1. 25 pts **Digtal Lock Source Code**

.thumb

**.text**

**.align** 2

**.global** main

.thumbfunc main

**main:** .asmfunc

; This line is a comment. Your program goes here

; Add LOTS of Comments!

; foo MOV R0, #0x01 ; This is a sample line of code with a label ‘foo’

;------ init buttons

;store address for p1sel0 in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c0A

**add** r0, r2, r3

;store value for p1sel0 in register 1

**MOV** R1, #0x00

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;store address for p1sel1 in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c0C

**add** r0, r2, r3

;store value for p1sel1 in register 1

**MOV** R1, #0x00

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;store address for p1dir in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c04

**add** r0, r2, r3

;store value for p1dir in register 1

**MOV** R1, #0x00

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;store address for p1ren in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c06

**add** r0, r2, r3

;store value for p1ren in register 1

**MOV** R1, #0x12

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;--------------------------------------------------------------

;-------- init LEDs

;store address for p2sel0 in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c0b

**add** r0, r2, r3

;store value for p2sel0 in register 1

**MOV** R1, #0x00

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;store address for p2sel1 in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c0d

**add** r0, r2, r3

;store value for p2sel1 in register 1

**MOV** R1, #0x00

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;store address for p2dir in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c05

**add** r0, r2, r3

;store value for p2dir in register 1

**MOV** R1, #0x07

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;store address for p2ren in register 0

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c07

**add** r0, r2, r3

;store value for p2ren in register 1

**MOV** R1, #0x07

;store value into memory location stored in r0 with an offset of 0

STRB R1, [R0, #0]

;-----------------------------------------------------------------

loopStart

;check port1 input register

;load from io register

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c00

**add** r0, r2, r3

**mov** r12, #0

ldrb r12, [r0,#0]

; we want to look at bits 1 and 4, so and result with 0x12

; the buttons are active low, so bits 1 and 4 will be 0 if they are pressed

**and** r4, r12, #0x12

**cmp** r4, #0x00

; branch if not equal to 00

bne not11

; if both buttons output 00 we have a correct button combo, display green

; address of P2OUT for LEDs

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c03

**add** r0, r2, r3

; value to store is 0x02

**mov** r1, #0x02

strb r1, [r0, #0]

b loopEnd

not11

**cmp** r4, #0x12

; branch if not equal to 12

bne not00

;if both buttons output 11 we are in standby, display blue

; address of P2OUT for LEDs

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c03

**add** r0, r2, r3

; value to store is 0x04

**mov** r1, #0x04

strb r1, [r0, #0]

b loopEnd

not00

; button combo is 01 or 10 bc its not 00 or 11, this is incorrect, display red

; address of P2OUT for LEDs

**mov** r2, #0x4000

lsl r3, r2, #0x10

**mov** r2, #0x4c03

**add** r0, r2, r3

; value to store is 0x01

**mov** r1, #0x01

strb r1, [r0, #0]

loopEnd **B** loopStart

.end