

Lab Report

|  |  |
| --- | --- |
| Name 1: Alexis Steven Garcia | Date: September 10, 2018 |
| Course: EGCP-450 | Lab #: 1 |

Grading Criteria:

|  |  |  |
| --- | --- | --- |
| **Section** | **Earned Points** | **Possible Points** |
| Problem/Objective: |  | 10 |
| Background: |  | 15 |
| Questions/Deliverables: |  | 15 |
| Program Code: |  | 30 |
| Demo: |  | 30 |
| Total: | 0 | 100 |

**PLEASE UPLOAD YOUR REPORT IN TITANIUM. NO PAPER REPORTS.**

Professor Comments:

# Problem/Objective

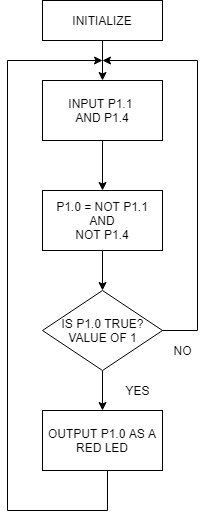
State the problem statement and/or objective of the lab. This must be a complete paragraph (i.e., at least 5 sentences).

The objective of this lab was to gain a better understanding of how software development is incorporated on a microcontroller to implement digital logic. In this lab, we were given five different projects, all accomplishing something different. Our assignment was to narrow our choices for the project that best matches the problem given. In this case, the project we sought after should have outputted a red led when no buttons were pressed. This digital logic that would accomplish this should have followed the following scenario: F = (not A) and (not B). Of course, the variables would be substituted for their appropriate input and output ports.

# Background

Briefly describe what you did in the lab including technical detail. It must be at least two **complete** paragraphs to receive full credit.

# Questions/Deliverables

1. Provide the flowchart of the system. If you want, you can hand write your work, take a picture, and paste the image here. One app that I would suggest to easily do this is “CamScanner”.
2. Provide the pseudo code for the algorithm

for (;;){

Get Input\_P1.1

Get Input\_P1.4

Output\_P1.0 = NOT Input\_P1.1 AND NOT P1.4

if (OUTPUT\_P1.0 is True)

Output\_P1.0 as red LED

else

do nothing

}

1. List 3 debugging tools available to you in CCS.

Three useful debugging tools available to us in CCS are breakpoints, viewing registers, and the memory and watch windows. Breakpoints are useful when we desire to pause a program at a specified location to perform some form of debugging. The viewing register tool allows us to do exactly what the name implies, view the registers. This debugging tool can help us in making sure we don’t overwrite something by mistake. The memory and watch windows aid us in several ways including keeping track of variable values and addresses.

# Program Code

Copy your code here. Please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your code as text, not as a screenshot/image**.

; InputOutput.s

; Runs on MSP432

; Test the GPIO initialization functions by setting the LED

; color according to the status of the switches.

; Daniel Valvano

; June 20, 2015

; This example accompanies the book

; "Embedded Systems: Introduction to the MSP432 Microcontroller",

; ISBN: 978-1512185676, Jonathan Valvano, copyright (c) 2015

; Section 4.2 Program 4.1

;

;Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu

; You may use, edit, run or distribute this file

; as long as the above copyright notice remains

;THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED

;OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF

;MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.

;VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,

;OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

;For more information about my classes, my research, and my books, see

;http://users.ece.utexas.edu/~valvano/

; built-in LED1 connected to P1.0

; negative logic built-in Button 1 connected to P1.1

; negative logic built-in Button 2 connected to P1.4

; built-in red LED connected to P2.0

; built-in green LED connected to P2.1

; built-in blue LED connected to P2.2

.thumb

**.text**

**.align** 2

P1IN .field 0x40004C00,32 ; Port 1 Input

P2IN .field 0x40004C01,32 ; Port 2 Input

P2OUT .field 0x40004C03,32 ; Port 2 Output

P1OUT .field 0x40004C02,32 ; Port 1 Output

P1DIR .field 0x40004C04,32 ; Port 1 Direction

P2DIR .field 0x40004C05,32 ; Port 2 Direction

P1REN .field 0x40004C06,32 ; Port 1 Resistor Enable

P2REN .field 0x40004C07,32 ; Port 2 Resistor Enable

P1DS .field 0x40004C08,32 ; Port 1 Drive Strength

P2DS .field 0x40004C09,32 ; Port 2 Drive Strength

P1SEL0 .field 0x40004C0A,32 ; Port 1 Select 0

P2SEL0 .field 0x40004C0B,32 ; Port 2 Select 0

P1SEL1 .field 0x40004C0C,32 ; Port 1 Select 1

P2SEL1 .field 0x40004C0D,32 ; Port 2 Select 1

RED .equ 0x01

GREEN .equ 0x02

BLUE .equ 0x04

SW1 .equ 0x02 ; on the left side of the LaunchPad board

SW2 .equ 0x10 ; on the right side of the LaunchPad board

**.global** main

.thumbfunc main

**main:** .asmfunc

BL Port1\_Init ; initialize P1.1 and P1.4 and make them inputs (P1.1 and P1.4 built-in buttons)

BL Port2\_Init ; initialize P2.2-P2.0 and make them outputs (P2.2-P2.0 built-in LEDs)

loop

BL Port1\_Input ; read both of the switches on Port 1

**CMP** R0, #0x10 ; R0 == 0x10?

BEQ sw1pressed ; if so, switch 1 pressed

**CMP** R0, #0x02 ; R0 == 0x02?

BEQ sw2pressed ; if so, switch 2 pressed

**CMP** R0, #0x00 ; R0 == 0x00?

BEQ bothpressed ; if so, both switches pressed

**CMP** R0, #0x12 ; R0 == 0x12?

BEQ nopressed ; if so, neither switch pressed

; if none of the above, unexpected return value

**MOV** R0, #(RED+GREEN+BLUE) ; R0 = (RED|GREEN|BLUE) (all LEDs on)

BL Port2\_Output ; turn all of the LEDs on

**B** loop

sw1pressed

**MOV** R0, #0 ; R0 = 0 (no LEDs on)

BL Port2\_Output ; turn all of the LEDs off

**B** loop

sw2pressed

**MOV** R0, #0 ; R0 = 0 (no LEDs on)

BL Port2\_Output ; turn all of the LEDs off

**B** loop

bothpressed

**MOV** R0, #0 ; R0 = 0 (no LEDs on)

BL Port2\_Output ; turn all of the LEDs off

**B** loop

nopressed

**MOV** R0, #RED ; R0 = RED (red LED on)

BL Port2\_Output ; turn the red LED on

**B** loop

.endasmfunc

;------------Port1\_Init------------

; Initialize GPIO Port 1 for negative logic switches on P1.1 and

; P1.4 as the LaunchPad is wired. Weak internal pull-up

; resistors are enabled.

; Input: none

; Output: none

; Modifies: R0, R1

**Port1\_Init:** .asmfunc

LDR R1, P1SEL0

**MOV** R0, #0x00 ; configure P1.4 and P1.1 as GPIO

STRB R0, [R1]

LDR R1, P1SEL1

**MOV** R0, #0x00 ; configure P1.4 and P1.1 as GPIO

STRB R0, [R1]

LDR R1, P1DIR

**MOV** R0, #0x00 ; make P1.4 and P1.1 inputs

STRB R0, [R1]

LDR R1, P1REN

**MOV** R0, #0x12 ; enable pull resistors on P1.4 and P1.1

STRB R0, [R1]

LDR R1, P1OUT

**MOV** R0, #0x12 ; P1.4 and P1.1 are pull-up

STRB R0, [R1]

BX LR

.endasmfunc

;------------Port1\_Input------------

; Read and return the status of the switches.

; Input: none

; Output: R0 0x10 if only Switch 1 is pressed

; R0 0x02 if only Switch 2 is pressed

; R0 0x00 if both switches are pressed

; R0 0x12 if no switches are pressed

; Modifies: R1

**Port1\_Input:** .asmfunc

LDR R1, P1IN

LDRB R0, [R1] ; read all 8 bits of Port 1

**AND** R0, R0, #0x12 ; select the input pins P1.1 and P1.4

BX LR

.endasmfunc

;------------Port2\_Init------------

; Initialize GPIO Port 2 red, green, and blue LEDs as

; the LaunchPad is wired.

; Input: none

; Output: none

; Modifies: R0, R1

**Port2\_Init:** .asmfunc

LDR R1, P2SEL0

**MOV** R0, #0x00 ; configure P2.2-P2.0 as GPIO

STRB R0, [R1]

LDR R1, P2SEL1

**MOV** R0, #0x00 ; configure P2.2-P2.0 as GPIO

STRB R0, [R1]

LDR R1, P2DS

**MOV** R0, #0x07 ; make P2.2-P2.0 high drive strength

STRB R0, [R1]

LDR R1, P2DIR

**MOV** R0, #0x07 ; make P2.2-P2.0 out

STRB R0, [R1]

LDR R1, P2OUT

**MOV** R0, #0x00 ; all LEDs off

STRB R0, [R1]

BX LR

.endasmfunc

;------------Port2\_Output------------

; Set the output state of P2.

; Input: R0 new state of P2 (only 8 least significant bits)

; Output: none

; Modifies: R1

**Port2\_Output:** .asmfunc

LDR R1, P2OUT

STRB R0, [R1] ; write to P2.7-P2.0

BX LR

.endasmfunc

.end