

Lab Report

|  |  |
| --- | --- |
| Name: Alexis Steven Garcia | Date: September 19, 2018 |
| Course: EGCP-450 | Lab #: 2 |

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 build upon the first lab by learning simple programming structures. In other words, we will implement assembly code such a mask that can be used for bitwise operations, if-then statements that will branch to different functions depending on the condition and looping that can allow our program to do a multiple of things several times. Also, we will be exposed to clock profiling that will aid us in creating a time delay function. The focus for the time delay function is to make it long enough so that the LED is visibly seen being toggled. With a delay as small as 100ms, it is enough to see the led being turned on and off when the button is pressed.

# Background

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

This lab proved to be a little more problematic to code at first due to the misinterpretation of how I/O ports and control registers functioned; however, once it was explained, it was easy to manipulate the code for this project. This lab was adding onto what we learned from the first lab and used that code as the foundation. This lab combined the operation of the first lab with a few topics we learned. Unlike the first lab though which involved turning off the led connected to P1.0 while the buttons connected to P1.1, P1.4, or both were pressed, we only used the led and the button connected to P1.4. The objective of this lab was to maintain the led on when no button was pressed and toggle the led on and off while the button was pressed.

This project used a pushbutton as an input and led as an output, so the first order of business was to code the control registers properly. This involved clearing P1SEL0 and P1SEL1 to select the port as GPIO. Then it was obligatory to change P1DIR to having a logic ‘1’ connected to pin 0 and a logic ‘0’ connected to pin 4 since this would allow for the led to be considered an output while the pushbutton is an input. Also, P1REN had the corresponding pin for the pushbutton set to a logic ‘1’ to enable an internal resistor. Finally, the last control register to change was P1OUT, which required a logic ‘1’ to be set for both the button and led. This would allow for the led to begin on and the button to function properly.

As for the main code, the first thing to do was code to keep the led on when no button was pressed. As for when the pushbutton was pressed, the led should toggle on and off depending on the status of the led. Nothing to complicated; however, the toggle required a delay in between so that it can be apparent that the led is being affected. The delay involved a loop that executed 300,000 bus clock cycle in 100ms. Even with a small delay, the delay was enough to see the led being turned off and on.

# Questions/Deliverables:

1. What is a “Reset Vector” and why do you need it?

A “Reset Vector” is a pointer that points to the address of the first instruction it should execute when a reset happens. It is needed because it allows for the program counter to be loaded with the address pointing to the start of the startup code.

1. In this lab, you were asked to create a delay based on the number of cycles. Use this to calculate the estimated time delay in seconds. Show your work to receive full credit. 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”.

According to the lab instructions, the MSP432 default bus clock is 3MHz ± 0.5%; therefore, there are 3,000 bus clock cycles in 1ms. Our goal is to get 100ms.

# 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**. Also, provide your flowchart here. Again, 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”.

**Main.asm**

; SwitchTestMain.asm

; Runs on MSP432

; Make P1.0 an output and make P1.4 an input (enable pull-up for P1.4)

; The system starts with the LED ON (make P1.0 =1)

; Delay for about 100 ms (using a loop(s))

; If the switch is pressed (P1.4 is 0), then toggle the LED once, else turn the LED ON

; Repeat steps 3 and 4 over and over

; Alexis Steven Garcia

; September 12, 2018

; 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

; positive logic switch connected to P1.5

; 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

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

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

SWEXT .equ 0x20 ; external switch

RED .equ 0x01

LED1 .equ 0x01

**.global** main

**.global** Switch\_Init

**.global** Switch\_Input

**.global** Board\_Init

**.global** Board\_Input

.thumbfunc main

**main:** .asmfunc

LDR R1, P1SEL0

LDRB R0, [R1]

BIC R0, R0, #LED1 ; configure built-in LED1 as GPIO

STRB R0, [R1]

LDR R1, P1SEL1

LDRB R0, [R1]

BIC R0, R0, #LED1 ; configure built-in LED1 as GPIO

STRB R0, [R1]

LDR R1, P1DIR

LDRB R0, [R1]

ORR R0, R0, #LED1 ; set up P1.0 as output

STRB R0, [R1]

LDR R1, P1REN

LDRB R0, [R1]

ORR R0, R0, #0x12 ; enable pull resistors

STRB R0, [R1]

LDR R1, P1OUT

LDRB R0, [R1]

BIC R0, #0xFF

ORR R0, R0, #0x13 ; set up P1OUT to only use what we need

STRB R0, [R1]

LDR R4, P1OUT ; R4 = &P1OUT (pointer)

loop

LDRB R6, [R4] ; 8-bit contents of register P1OUT

**MOV** R7, #1000 ; load high enough values for

**MOV** R8, #50 ; delay loop can reach 300,000

BL delay ; cycles giving .1 sec delay

BL Board\_Input ; read if switches were pressed

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

BEQ sw1pressed ; if so, switch P1.4 pressed

**B** nopressed ; if so, switch isn't pressed

delay ; nested for loop

**NOP** ; for (i=50; i>0; i--)

**SUB** R7, #1 ; reset j

**CMP** R7, #0 ; for (j=1000; j>0; j--)

BEQ delay\_2

**B** delay

delay\_2

**MOV** R7, #1000

**SUB** R8, #1

**CMP** R8, #0

BEQ done

**B** delay

done

BX LR

sw1pressed

**CMP**  R6, #0x13 ; check if led is on or off

BEQ led\_on

**B** led\_off

led\_on

**AND** R6, R6, #0x12 ; led is on so turn it off

**B** continue

led\_off

ORR R6, R6, #0x13 ; led is off so turn it on

**B** continue

nopressed

ORR R6, R6, #0x13 ; turn led on or keep it on

**B** continue

continue

STRB R6, [R4] ; store R6 into 8-bit P1OUT

BL Switch\_Input ; status = R0 = 0x00 or 0x01

**B** loop

.endasmfunc

.end

**Switch.asm**

; Switch.asm

; Runs on MSP432

; Alexis Garcia

; September 12, 2018

; Not all functions are used

.thumb

**.text**

**.align** 2

P1IN .field 0x40004C00,32 ; Port 1 Input

P1OUT .field 0x40004C02,32 ; Port 1 Output

P1DIR .field 0x40004C04,32 ; Port 1 Direction

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

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

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

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

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

SWEXT .equ 0x20 ; external switch

**.global** Switch\_Init

**.global** Switch\_Input

**.global** Board\_Init

**.global** Board\_Input

;------------Switch\_Init------------

; Initialize GPIO Port 1 bit 5 for input. An external pull-down

; resistor is used.

; Input: none

; Output: none

; Modifies: R0, R1

**Switch\_Init:** .asmfunc

; configure P1.5 as GPIO

LDR R1, P1SEL0

LDRB R0, [R1]

BIC R0, R0, #0x20 ; configure P1.5 as GPIO

STRB R0, [R1]

LDR R1, P1SEL1

LDRB R0, [R1]

BIC R0, R0, #0x20 ; configure P1.5 as GPIO

STRB R0, [R1]

; make P1.5 in

LDR R1, P1DIR

LDRB R0, [R1]

BIC R0, R0, #0x20 ; input direction

STRB R0, [R1]

; disable pull resistor on P1.5

LDR R1, P1REN

LDRB R0, [R1]

BIC R0, R0, #0x20 ; disable pull resistor

STRB R0, [R1]

BX LR

.endasmfunc

;------------Switch\_Input------------

; Read and return the status of GPIO Port 1 bit 5.

; Input: none

; Output: R0 0x20 if P1.5 high

; R0 0x00 if P1.5 low

; Modifies: R1

**Switch\_Input:** .asmfunc

LDR R1, P1IN

LDRB R0, [R1] ; 8-bit contents of register

**AND** R0, R0, #0x20 ; get just P1.5

BX LR ; return 0x20 or 0x00

.endasmfunc

;------------Board\_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

**Board\_Init:** .asmfunc

; configure P1.4 and P1.1 as GPIO

LDR R1, P1SEL0

LDRB R0, [R1]

BIC R0, R0, #0x12 ; configure P1.4 and P1.1 as GPIO

STRB R0, [R1]

LDR R1, P1SEL1

LDRB R0, [R1]

BIC R0, R0, #0x12 ; configure P1.4 and P1.1 as GPIO

STRB R0, [R1]

; make P1.4 and P1.1 in

LDR R1, P1DIR

LDRB R0, [R1]

BIC R0, R0, #0x12 ; input direction

STRB R0, [R1]

; enable pull resistors on P1.4 and P1.1

LDR R1, P1REN

LDRB R0, [R1]

ORR R0, R0, #0x12 ; enable pull resistors

STRB R0, [R1]

; P1.4 and P1.1 are pull-up

LDR R1, P1OUT

LDRB R0, [R1]

ORR R0, R0, #0x12 ; pull-up resistors

STRB R0, [R1]

BX LR

.endasmfunc

;------------Board\_Input------------

; Read and return the status of the switches.

; Input: none

; Output: 0x10 if only Switch 1 is pressed

; 0x02 if only Switch 2 is pressed

; 0x00 if both switches are pressed

; 0x12 if no switches are pressed

; Modifies: R1

**Board\_Input:** .asmfunc

LDR R1, P1IN

LDRB R0, [R1] ; 8-bit contents of register

**AND** R0, R0, #0x12 ; get just input pins P1.4 and P1.1

BX LR

.endasmfunc

.end

