

Lab Report

|  |  |
| --- | --- |
| Name: Alexis Steven Garcia | Date: 10/1/2018 |
| Course: EGCP-450 | Lab #: 3 |

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 second lab by learning how to implement external hardware with code developed from the previous lab. In other words, we will implement our assembly code while relying on external hardware. The goal of this lab is to not use peripherals on the board, but instead to depend our own provided LED, resistors, and pushbuttons that we construct into a circuit. Also, we should consider how control registers are altered when not using the built-in peripherals. Finally, we will understand how to use Multisim for drawing and analyzing electrical circuits.

# 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 demonstrated how GPIO pins can interface with external hardware and how it is comparable with using the onboard peripherals. The lab was taking advantage of what we learned from the second lab and used that code as the foundation; therefore, we just had to modify certain aspects of the code to enable the use of external hardware. Unlike the second lab which involved toggling the LED connected to P1.0 off and on while the button connected to P1.4 was pressed and having the LED emit light otherwise, this lab instructed to do the opposite using external hardware. In other words, the code should function where the LED toggles on and off indefinitely until the pushbutton is pressed. When the button is pressed the LED should turn and stay on until the button is no longer pressed.

This project utilized a pushbutton connected to P5.0 as an input and LED connected to P4.0 as an output, so it was essential to code the control registers properly. This involved clearing the select control registers for both ports to set them up as GPIO. Then it was necessary to modify P4DIR to having a logic ‘1’ connected to pin 0 and a logic ‘0’ connected to pin 0 of P5DIR since this would allow for the LED to be considered an output while the pushbutton is regarded as an input. Also, P4REN had the corresponding pin for the pushbutton set to a logic ‘0’ since we no longer had to enable a pull-down internal resistor. Finally, the last control registers to change was P4OUT and P5OUT, which required a logic ‘1’ to be set for P4.0 and a logic ‘0’ to be set for P5.0. This would allow for the led to begin on and the button to function using positive logic.

As for the rest of the code, the only thing to modify was the delay and when the LED should toggle; therefore, when the pushbutton was not pressed, the led should toggle on and off depending on the status of the led and emit light when pressed. As for the delay, the code took advantage of the loop created in the last lab, so it was modified to run in a loop that executed approximately 187,500 bus clock cycles in 62.5ms.

The final objective of this lab was to gather measurements. First from the design created on Multisim and then from our built circuit. Regarding the Multisim part, the design was given, and we were expected to use the logic analyzer to obtain a waveform. As for obtaining measurements from the circuit, our goal was to verify that voltage is low when the button is not pressed and high when the button is pressed.

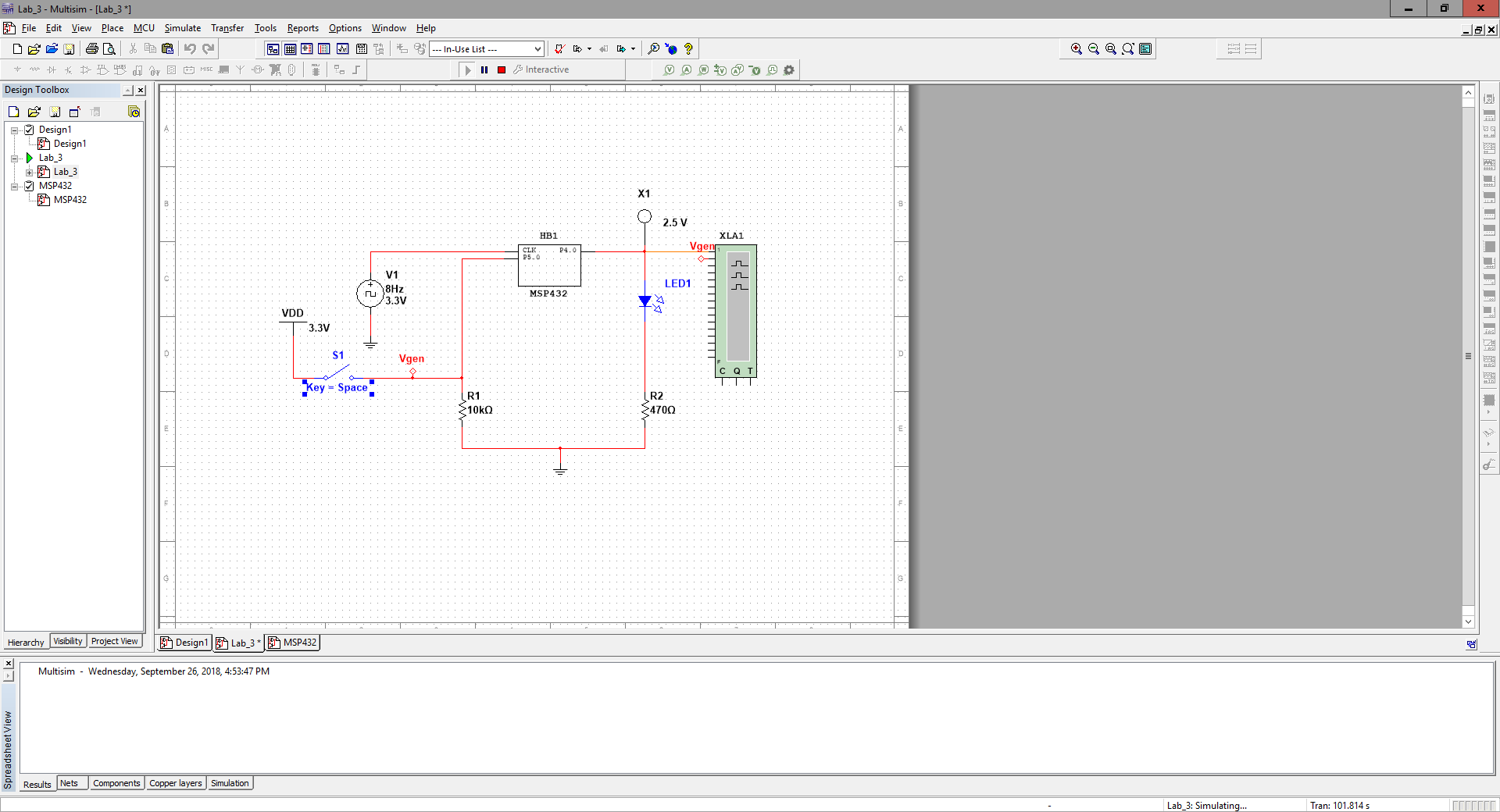
# Questions/Deliverables:

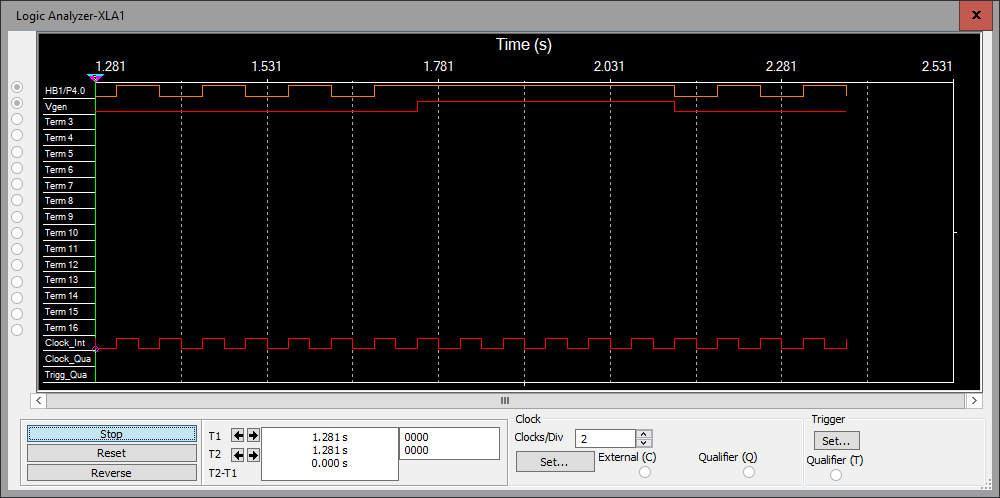
* **WARNINGS**
  + **You are responsible for any of CpE’s equipment (i.e., boards, computers, DMM, etc.). If damaged, you will have to replace it at your own expense. So…**
  + **NEVER INSERT/REMOVE WIRES/CHIPS WHEN THE POWER IS ON.**
  + **DO NOT USE THE AMMETER. If done improperly, it will damage the board permanently.**
* **NOTE: The Multisim values might be different than the actual measurements. Some of the parts (e.g., LED) do not have the same parameters.**

1. Debug your combined hardware/software system on the actual MSP432 board.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row | Parameter | Value | Units | Conditions |
| 1 | Resistance of the 470 resistor, R2 | 466 |  | With power off and disconnected from circuit  (measured with ohmmeter) |
| 2 | MSP432 Output, *VP4.0*  (Average) | 1.6 | V | Powered, but with switch not pressed (measured with voltmeter). LED blinking, so estimate an average. |
| 3 | *Vk-* LED k-  (Average) | 0.6 | V | Powered, but with switch not pressed (measured with voltmeter).  LED blinking, so estimate an average. |
| 4 | LED a+, *Va+* | 1.6 | V | Powered, but with switch not pressed (measured with voltmeter) |
| 5 | LED voltage | 1 | V | Calculated as Va+ - Vk- |
| 6 | LED current | 1.287 | mA | Calculated as Vk-/R2 |
| 7 | MSP432 Output, *VP4.0* | 3.1 | V | Powered and with switch pressed (measured with voltmeter). |
| 8 | *Vk-* LED k- | 1.4 | V | Powered and with switch pressed (measured with voltmeter) |
| 9 | LED a+, *Va+* | 3.1 | V | Powered and with switch pressed (measured with voltmeter) |
| 10 | LED voltage | 1.7 | V | Calculated as Va+ - Vk- |
| 11 | LED current | 3.65 | mA | Calculated as Vk-/R2 |

1. Also, include a screenshot of your Multisim schematic and waveform here.





# 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

; Alexis Steven Garcia

.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

.global Delay\_Func

.global External\_Init

.thumbfunc main

main: .asmfunc

BL External\_Init

loop

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

MOV R7, #1000 ; load high enough values for

MOV R8, #31 ; delay loop can reach 300,000

BL Delay\_Func ; cycles giving .1 sec delay

BL Board\_Input ; read if switches were pressed

CMP R0, #0x01 ; R0 == 0x01?

BEQ sw1pressed ; if so, switch is pressed

B nopressed ; if so, switch isn't pressed

sw1pressed

ORR R6, R6, #0x01 ; turn led on or keep it on

B continue

nopressed

CMP R6, #0x01 ; check if led is on or off

BEQ led\_on

B led\_off

led\_on

AND R6, R6, #0x00 ; led is on so turn it off

B continue

led\_off

ORR R6, R6, #0x01 ; led is off so turn 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 since some are carried

; over from previous labs

.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

P4IN .field 0x40004C21,32 ; Port 4 Input

P4OUT .field 0x40004C23,32 ; Port 4 Output

P4DIR .field 0x40004C25,32 ; Port 4 Direction

P4REN .field 0x40004C27,32 ; Port 4 Resistor Enable

P4SEL0 .field 0x40004C29,32 ; Port 4 Select 0

P4SEL1 .field 0x40004C0B,32 ; Port 4 Select 1

P5IN .field 0x40004C40,32 ; Port 1 Input

P5OUT .field 0x40004C42,32 ; Port 1 Output

P5DIR .field 0x40004C44,32 ; Port 1 Direction

P5REN .field 0x40004C46,32 ; Port 1 Resistor Enable

P5SEL0 .field 0x40004C4A,32 ; Port 1 Select 0

P5SEL1 .field 0x40004C4C,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

LED1 .equ 0x01

EMPTY .equ 0x00

.global Switch\_Init

.global Switch\_Input

.global Board\_Init

.global Board\_Input

.global Delay\_Func

.global External\_Init

;------------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, #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)

BX LR

.endasmfunc

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

; Read and return the status of the switches.

; Input: none

; Output: 0x01 if only Switch 1 is pressed

; Modifies: R1, R0

Board\_Input: .asmfunc

LDR R1, P5IN

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

AND R0, R0, #0x01 ; get just input pins P5.0

BX LR

.endasmfunc

;------------Delay\_Func------------

; Sets the delay for .1 seconds

; Input: none

; Output: none

; Modifies: R7, R8

Delay\_Func: .asmfunc

delay

NOP

SUB R7, #1

CMP R7, #0

BEQ delay\_2

B delay

delay\_2

MOV R7, #1000

SUB R8, #1

CMP R8, #0

BEQ done

B delay

done

BX LR

.endasmfunc

;------------External\_Init------------

; Initialize GPIO Port 5 for positive logic switches on P5.0 and

; P4.0 as an input

; Input: none

; Output: none

; Modifies: R0, R1

External\_Init: .asmfunc

LDR R1, P4SEL0

LDRB R0, [R1]

AND R0, R0, #EMPTY ; configure P4 as GPIO

STRB R0, [R1]

LDR R1, P4SEL1

LDRB R0, [R1]

AND R0, R0, #EMPTY ; configure P4 as GPIO

STRB R0, [R1]

LDR R1, P4DIR

LDRB R0, [R1]

ORR R0, R0, #LED1 ; set up P4.0 as output

STRB R0, [R1]

LDR R1, P4REN

LDRB R0, [R1]

AND R0, R0, #EMPTY ; no pull resistors needed

STRB R0, [R1]

LDR R1, P4OUT

LDRB R0, [R1]

BIC R0, #0xFF

ORR R0, R0, #LED1 ; set up P4OUT to only use what we need

STRB R0, [R1]

LDR R4, P4OUT ; R4 = &P4OUT (pointer)

LDR R1, P5SEL0

LDRB R0, [R1]

AND R0, R0, #EMPTY ; configure P5 as GPIO

STRB R0, [R1]

LDR R1, P5SEL1

LDRB R0, [R1]

AND R0, R0, #EMPTY ; configure P5 as GPIO

STRB R0, [R1]

LDR R1, P5DIR

LDRB R0, [R1]

ORR R0, R0, #EMPTY ; set up P5.0 as input

STRB R0, [R1]

LDR R1, P5REN

LDRB R0, [R1]

AND R0, R0, #EMPTY ; no pull resistors needed

STRB R0, [R1]

LDR R1, P5OUT

LDRB R0, [R1]

BIC R0, #0xFF

AND R0, R0, #EMPTY ; set up P5OUT to only use what we need

STRB R0, [R1]

LDR R5, P5OUT ; R5 = &P4OUT (pointer)

BX LR

.endasmfunc

.end

