

Lab Report

|  |  |
| --- | --- |
| Name: Alexis Steven Garcia | Date: December 12, 2018 |
| Course: EGCP-450 | Lab #: 8 |

Grading Criteria:

|  |  |  |
| --- | --- | --- |
| **Section** | **Earned Points** | **Possible Points** |
| Program Code: |  | 34 |
| Part 1 Demo: |  | 33 |
| Part 2 Demo: |  | 33 |
| Total: | 0 | 100 |

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

Professor Comments:

# 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**.

## Part 1: 7-Segment Display Driver

**Main.c:**

**#include** <stdint.h>

**#include** "msp432p401r.h"

**#include** "SysTick.h"

**#include** "SSEG.h"

**#include** "PushButtons.h"

**void** **main**() {

**SSEG\_Init**();

**ButtonInit**();

**while**(1);

}

**SSEG.h:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Public Functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* SSEG\_Init Function

\* Initialize 7-segment display

\* Inputs: none

\* Outputs: none

\*/

**void** **SSEG\_Init**();

/\*

\* SSEG\_Out Function

\* Output a number to a single digit of the 7-segment display

\* Inputs: a number between 0 and 15

\* Outputs: none

\*/

**void** **SSEG\_Out**(**uint8\_t** num);

**SSEG.c:**

**#include** <stdint.h>

**#include** "SysTick.h"

**#include** "msp432p401r.h"

**#include** "SSEG.h"

// Global Variables

**char** out\_num;

**void** **DisableInterrupts**(); // Disable interrupts

**void** **EnableInterrupts**(); // Enable interrupts

**long** **StartCritical** (); // previous I bit, disable interrupts

**void** **EndCritical**(**long** sr); // restore I bit to previous value

**void** **WaitForInterrupt**(); // low power mode

/\*

\* SSEG\_Init Function

\* Initialize 7-segment display

\* Inputs: none

\* Outputs: none

\*/

**void** **SSEG\_Init**() {

**SysTick\_Init**(1000);

P4SEL0 = 0x00;

P4SEL1 = 0x00;

P4DIR = 0xFF;

P4OUT = 0x3F;

}

/\*

\* SSEG\_Out Function

\* Output a number to a single digit of the 7-segment display

\* Inputs: a number between 0 and 15

\* Outputs: none

\*/

**void** **SSEG\_Out**(**uint8\_t** num) {

**switch**(num){

**case** 0:

P4OUT = 0x3F;

**break**;

**case** 1:

P4OUT = 0x06;

**break**;

**case** 2:

P4OUT = 0x5B;

**break**;

**case** 3:

P4OUT = 0x4F;

**break**;

**case** 4:

P4OUT = 0x66;

**break**;

**case** 5:

P4OUT = 0x6D;

**break**;

**case** 6:

P4OUT = 0x7D;

**break**;

**case** 7:

P4OUT = 0x07;

**break**;

**case** 8:

P4OUT = 0x7F;

**break**;

**case** 9:

P4OUT = 0x67;

**break**;

**case** 10:

P4OUT = 0x77;

**break**;

**case** 11:

P4OUT = 0x7C;

**break**;

**case** 12:

P4OUT = 0x39;

**break**;

**case** 13:

P4OUT = 0x5E;

**break**;

**case** 14:

P4OUT = 0x79;

**break**;

**case** 15:

P4OUT = 0x71;

**break**;

}

}

**PushButtons.h:**

/\*

\* PushButtons.h

\*

\* Created on: Dec 12, 2018

\* Author: Alexis

\*/

**#ifndef** PUSHBUTTONS\_H\_

**#define** **PUSHBUTTONS\_H\_**

**#define** **SENSOR** (\*((**volatile** uint8\_t \*)0x40004C40))

**void** **ButtonInit**();

**unsigned** **long** **GetButton**();

/\*

\* Port 5 ISR

\* Uses P5IV to solve critical section/race

\*/

**unsigned** **long** **PORT5\_IRQHandler**(**unsigned** **long** num);

**#endif** /\* PUSHBUTTONS\_H\_ \*/

**PushButtons.c:**

/\*

\* PushButtons.c

\*

\* Created on: Dec 12, 2018

\* Author: Alexis

\*/

**#include** <stdint.h>

**#include** "SysTick.h"

**#include** "msp432p401r.h"

**#include** "SSEG.h"

**#include** "PushButtons.h"

**unsigned** **long** Temp;

**unsigned** **long** Input;

**void** **ButtonInit**(){

Temp = 0;

Input = 0;

P5->SEL0 = 0x00; //make P5.1-P5.0 GPIO Inputs

P5->SEL1 = 0x00;

P5->DIR = 0x00;

}

**unsigned** **long** **GetButton**(){

Input = (SENSOR&0x30)>>4;

**return** Input;

}

**unsigned** **long** **PORT5\_IRQHandler**(**unsigned** **long** num){

**int** value = num;

**if**(value > 15){

value = 0;

}

**else** **if** (value < 0){

value = 15;

}

**return** value;

}

**SysTick.h:**

// Initialize SysTick with busy wait running at bus clock.

**void** **SysTick\_Init**();

// Time delay using busy wait.

// The delay parameter is in units of the core clock. (units of 333 nsec for 3 MHz clock)

**void** **SysTick\_Wait**(**uint32\_t** delay);

// Time delay using busy wait.

// This assumes 3 MHz system clock.

**void** **SysTick\_Wait10ms**(**uint32\_t** delay);

**void** **SysTick\_Handler**();

**SysTick.c:**

**#include** <stdint.h>

**#include** "msp432p401r.h"

**#include** "SSEG.h"

**#include** "PushButtons.h"

**void** **DisableInterrupts**(**void**); // Disable interrupts

**void** **EnableInterrupts**(**void**); // Enable interrupts

**long** **StartCritical** (**void**); // previous I bit, disable interrupts

**void** **EndCritical**(**long** sr); // restore I bit to previous value

**void** **WaitForInterrupt**(**void**); // low power mode

**volatile** **uint32\_t** Counts;

**volatile** **uint32\_t** Button;

**void** **SysTick\_Init**(**unsigned** **long** period) {

**long** sr = **StartCritical**();

Counts = 0;

SysTick->CTRL = 0; // disable SysTick during setup

SysTick->LOAD = period - 1; // maximum reload value

SysTick->VAL = 0; // any write to current clears it

SCB->SHP[3] = (SCB->SHP[3]&0x00FFFFFF)|0x40000000; // priority 2

SysTick->CTRL = 0x00000007; // enable SysTick with no interrupts

**EndCritical**(sr);

}

// Time delay using busy wait.

// The delay parameter is in units of the core clock. (units of 333 nsec for 3 MHz clock)

**void** **SysTick\_Wait**(**uint32\_t** delay){

// method #1: set Reload Value Register, clear Current Value Register, poll COUNTFLAG in Control and Status Register

**if**(delay <= 1){

// without this step:

// if delay == 0, this function will wait 0x00FFFFFF cycles

// if delay == 1, this function will never return (because COUNTFLAG is set on 1->0 transition)

**return**; // do nothing; at least 1 cycle has already passed anyway

}

SysTick->LOAD = (delay - 1);// count down to zero

SysTick->VAL = 0; // any write to CVR clears it and COUNTFLAG in CSR

**while**((SysTick->CTRL&0x00010000) == 0){};

// method #2: repeatedly evaluate elapsed time

/\* volatile uint32\_t elapsedTime;

uint32\_t startTime = SysTick->VAL;

do{

elapsedTime = (startTime-SysTick->VAL)&0x00FFFFFF;

}

while(elapsedTime <= delay);\*/

}

// Time delay using busy wait.

// This assumes 3 MHz system clock.

**void** **SysTick\_Wait10ms**(**uint32\_t** delay){

**uint32\_t** i;

**for**(i=0; i<delay; i++){

**SysTick\_Wait**(30000); // wait 10ms (assumes 3 MHz clock)

}

}

**void** **SysTick\_Handler**() {

Button = **GetButton**();

**if**(Button == 1){

Counts++;

}

**else** **if**(Button == 2){

Counts--;

}

// Counts++;

// Counts--;

Counts = **PORT5\_IRQHandler**(Counts);

**SSEG\_Out**(Counts);

**SysTick\_Wait**(500000);

}

## Part 2: 4-Digit 7-Segment Device Driver