California State University

EGCP 450

Lab 6

Professor

Zaid Frayeh

Question1

Main.c :

```
#include <stdint.h>
#include "msp432p401r.h"
#include "SSEG.h"
#include "SysTick.h"

void main() {
    SSEG_Init();
    SysTick_Init();


    while(1)
    {

        printf("\n Press the left button to go down \n Press right button to
go up \n press both to input a number\n ");
        WaitForInterrupt();


}
}
```

SEGG.c

```
#include <stdint.h>
#include "SysTick.h"
#include "msp432p401r.h"
#include "SSEG.h"

// Global Variables
char out_num;
int out;
int digit1,digit2,digit3,digit4;

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
 */
```

```c
void SSEG_Init() {
        P4->SEL1 &= ~0x07;      /* configure P4 */
        P4->SEL0 &= ~0x07;
        P4->DIR |= 0x07;



        P5->SEL1 &= ~0xc0;      /* configure P5  */
        P5->SEL0 &= ~0xc0;
        P5->DIR &= ~0xc0;
        P5->IES &= ~0xc0;
        P5->IFG &= ~0xc0;
        P5IE |=0xc0;


        P6->SEL1 &= ~0xf0;      /* configure P6  */
        P6->SEL0 &= ~0xf0;
        P6->DIR |= 0xf0;


        NVIC ->IP[9] = (NVIC -> IP[9]&0x00ffffff)|0x40000000;
        NVIC->ISER[1]= 0x00000080;

        EnableInterrupts();
        return;
}
int SER;
/*
 * 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:
                out = 0x01;
               // out =0x04;
                break;
            case 1:
                out = 0xcf;
               // out =0x01;
                break;
            case 2:
                out = 0x92;
                break;
            case 3:
                out = 0x06;
                break;
            case 4:
                out = 0x4c;
                break;
            case 5:
                out = 0xa4;
                break;
            case 6:
                out = 0xa0;
                break;
            case 7:
                out = 0x8f;
                break;
            case 8:
                out = 0x80;
                break;
            case 9:
                out = 0x8c;
                break;


            }

    for (int i = 0; i < 8; i++)
        {
        P4OUT |= 0x01;//SRCLR |= BIT0; // Set SRCLR high again

                        P4OUT &= ~0x01;//SRCLR &= ~BIT0; // Set SRCLR low to
clear shift register

            if ((out >> i) & (0x01))
            {
                P4OUT |= 0x04; //SER |= BIT2; // Set SER high if bit is 1
            }
            else
            {
                P4OUT &= ~0x04;//SER &= ~BIT2; // Set SER low if bit is 0
            }

        }

                P4OUT |= 0x02;//RCLK |= BIT1; // Pulse RCLK to shift in data
                SysTick_Wait(25);
                P4OUT &= ~0x02;//RCLK &= ~BIT1;

            return;
}
void SSEG_Shift_Out(int counted){


        counted = counted%100000;
        digit1 = (counted%10000)/1000;
        digit2 = (counted%1000)/100;
        digit3 = (counted%100)/10;
        digit4 = (counted%10);

}

void SSEG_Disp_Num(int digit, int number){

        P4OUT = 0x00;                           // Turns off LEDs
        P6OUT = digit;                              // Selects digit
        SSEG_Out(number);    // Turns on number in selected digit

}
/*
 * Port 5 ISR
 * Uses P5IV to solve critical section/race
 */
int count =0;
void PORT5_IRQHandler() {





    uint32_t x;
    uint32_t input;
                              SSEG_Shift_Out(count);

                              SSEG_Disp_Num(0x80,digit4);
                              SysTick_Wait10ms(1);
                              SSEG_Disp_Num(0x40,digit3);
                              SysTick_Wait10ms(1);
                              SSEG_Disp_Num(0x20,digit2);
                              SysTick_Wait10ms(1);
                              SSEG_Disp_Num(0x10,digit1);
                              SysTick_Wait10ms(1);



                                           x = (P5IN/32);



                                         if(x == 5){
                                             if(count == 9999){
                                                   count = 0;
                                               }
                                               else{
                                               count++;
                                               }
                                             SysTick_Wait10ms(50);
                                           }
                                           else if(x == 3){
                                               if(count == 0){
                                                   count = 9999;
                                               }
                                               else{
                                                   count--;
                                               }
                                              SysTick_Wait10ms(50);
                                           }
                                           else if(x==7)
                                           {
                                               printf("\nPlease Enter Number
Between 0 – 9999: \n");

                                               scanf("%d",&input);
                                               if (input<0 || input >9999 )
                                               {
                                               printf("\invalid input please
try again \n");

                                               count = count;
                                               }
                                               else
                                             count=input;
                                           }


        }
```

SEGG.h

```
#ifndef __SSEG_H__
#define __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);
void PORT5_IRQHandler();
void SSEG_Shift_Out(int counted);
void SSEG_Disp_Num(int digit, int number);

#endif
```

SysTick.c

```
#include <stdint.h>
#include "msp432p401r.h"

// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(void){
      SysTick->CTRL = 0;                    // disable SysTick during setup
      SysTick->LOAD = 0x00FFFFFF;           // maximum reload value
      SysTick->VAL = 0;                    // any write to current clears it
      SysTick->CTRL = 0x00000005;           // enable SysTick with no
interrupts
}
// 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)
  }
}
```

SysTick.h

```
#ifndef __SYSTICK_H__
#define __SYSTICK_H__

// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(void);

// 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);

#endif
```

Question2

I was asked if I needed a physical shift register?

The answer is no, I don’t need a physical shift register IC I could use a software based register and implement this design. This becomes up to the designer where he can discuss the advantages and disadvantages of using a physical one
Such as less ICs , performance, resources etc.

Implementing a shift register would require defining and initializing pins on the microcontroller
And having a storage system to store the current values for the segment
Having a shift module which would shift the new values to the storage
A specific clock speed is also required
Output the data and update the segments
Repeat process continuously

Sources :

I used the sources from the HTML document provided.