Program Code

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

<u>Part 1: 7-Segment Display Driver</u>

Main.c

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

void main()
{

    SSEG_Init();   //initialize SSEG_int here
    SysTick_Init();   //initialize SysTick
    P4OUT = 0x14;  //initial value on board is zero

    while (1)
    {

    }
}
```

✓

SSEG.c

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

//*************Global Variables***************************
char out_num[16] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
uint8_t i  = 0x10;
//testing with 15 as starting point works both ways up and down wrap

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() {
//------------------------------------------
        P4SEL0 = 0x00;  //GPIOs P4
        P4SEL1 = 0x00;  //GPIOs P4
        P4DIR  = 0xFF;  //
   //******Port6_IRQhandler*********
        P6SEL0 &= ~0x03;  //GPIOs P6
        P6SEL1 &= ~0x03;  //GPIOs P6
        P6DIR  &= ~0x03;  //make P6.0 - P6.1 inputs
        P6IES  &= ~0x03;  //high low edge
        P6IFG  &= ~0x03;  //flags
        P6IE   |=  0x03;  //Interrupt enabled
        NVIC->IP[10]=(NVIC->IP[10]&0xFFFFFF1F)
         |0x00000040; //enable to pins for input
        NVIC->ISER[1] = 0x00000100;

}
```

```
void SSEG_Out(uint8_t num) {
/* A=1,F=2,B=3,G=4,C=5,D=6,E=7,H=6 FOR MAPPING*/
    switch(num){

    case 0:          //afbgchde  mapping
        P4OUT = 0x14;//00010100 -> number 0
        break;
    case 1:
        P4OUT = 0xD7;//10010111
        break;
    case 2:
        P4OUT = 0x4C;//01001100
        break;
    case 3:
        P4OUT = 0x45;//01000101
        break;
    case 4:
        P4OUT = 0x87;//10000111
        break;
    case 5:
        P4OUT = 0x25;//00100101
        break;
    case 6:
        P4OUT = 0xA4;//10100100
        break;
    case 7:
        P4OUT = 0x57;//01010111
        break;
    case 8:
        P4OUT = 0x04;//00000100
        break;
    case 9:
        P4OUT = 0x07;//00000111
        break;
    case 10:
        P4OUT = 0x02;//00000110
        break;
    case 11:
        P4OUT = 0xA0;//10100100
        break;
    case 12:
        P4OUT = 0x38;//00111100
        break;
    case 13:
        P4OUT = 0xC0;//11000100
        break;
    case 14:
        P4OUT = 0x28;//00101100
        break;
    case 15:
        P4OUT = 0x2A;//00101110
        break;
    }
    return;
}
```

✓

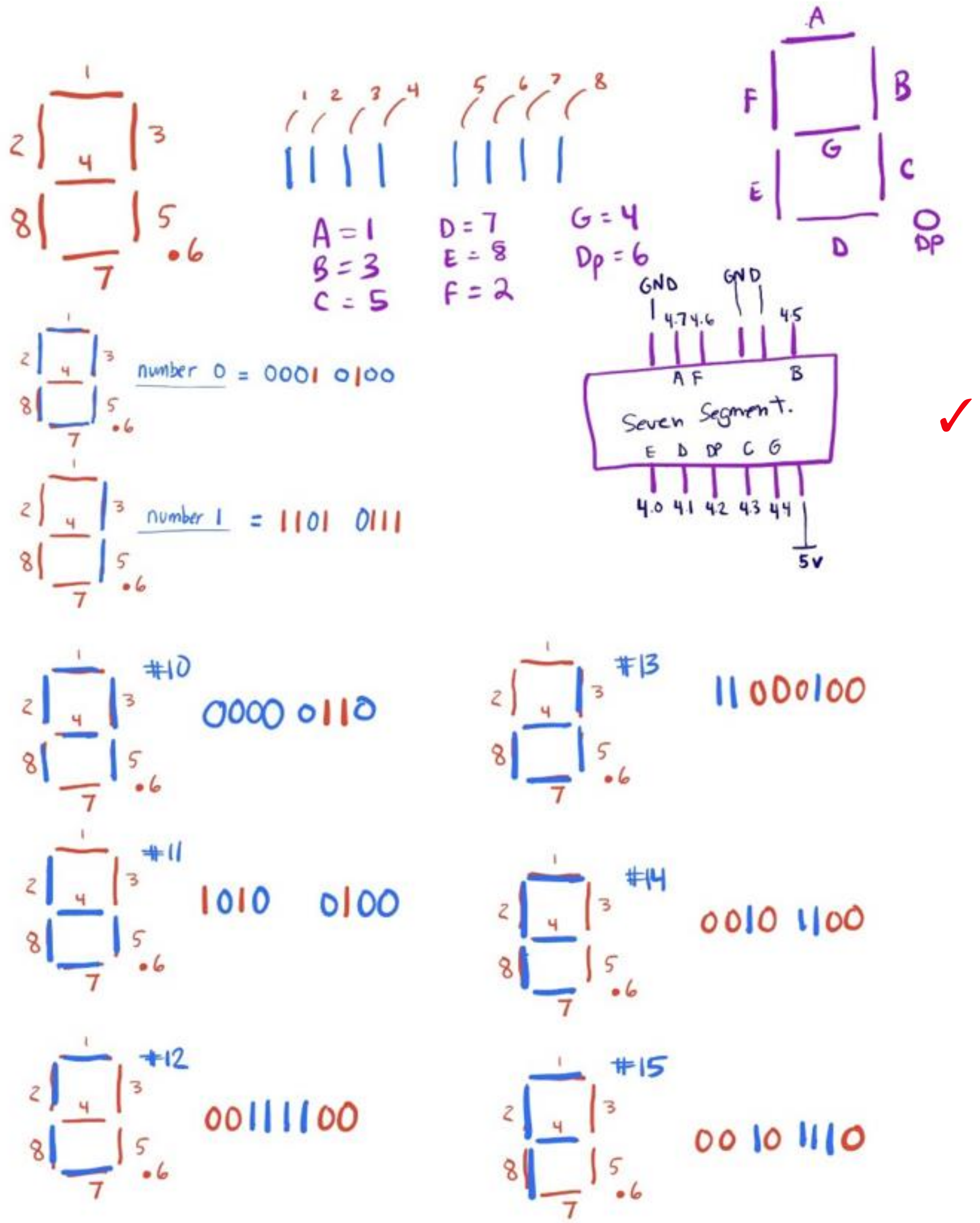
```
/*
 *
 * Port 5 ISR
 * Uses P5IV to solve critical section/race
 */
void PORT6_IRQHandler(){
    uint8_t status;
    status = P6IV;

    if(status==0x02)
        {                 //poll p6.0 2*(n+1) = 2
        i++;              //increment through SSEG_Out using array
        if (i >= 0x10){   // if i is 16, then set it equal to zero
        i = 0x00;         // sseg_out greater than 16 wrap set zero
        }
    }
    else
    {
        i--;             //decrement
        if (i >= 0x10){   // if i is 16, then set it equal to 0x0F
        i = 0x0F;         // sseg_out greater than 15 wrap
        }
    }

    SSEG_Out(out_num[i]);//print out cases on segment seven
    SysTick_Wait(60000); //de_bounce 20 ms 60000/3000000 = 0.018 or 20ms
}
```

✓

SSEG.h

```c
/*************** Public Functions ***************/
/*
 * SSEG_Init Function
 * Initialize 7-segment display
 * Inputs: none
 * Outputs: none
 */
void SSEG_Init();

/*
 * SSEG_Out Function
 * Output a 4-digit number to the display
 * Inputs: none
 * Outputs: none
 */
void SSEG_Out(uint8_t num);

/*
 * SSEG_Shift_Out Function
 * Shifts data out serially
 * Inputs: 8-bit data
 * Outputs: none
 */
void SSEG_Shift_Out(char *data);

/*
 * SSEG_Disp_Num Function
 * Separate the input number into 4 single digit
 * Inputs: num between 0 and 9999
 * Outputs: none
 */
void SSEG_Disp_Num(int digit);

/*
 * SSEG_Off Function
 * Turns off all 7-seg digits
 * Inputs: none
 * Outputs: none
 */
void SSEG_Off();
```

Below is the mapping I used based on how I connected MSP432 P4.0 - > P4.7 pins.



Part 2: 4-Digit 7-Segment Device Driver

Main.c

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

void input_num(int inputNum)//function to prompt user for input
{

    inputNum = -1; //initialize at -1 or 0 doesn't matter
    printf("Please enter between 0 - 9999: ");//ask
    scanf("%d", &inputNum);//user inputs gathered here
    if (inputNum >= 0 && inputNum <= 9999)//condition
    {
        SSEG_Disp_Num(inputNum);//send inputs to SSEG_Disp_Num
    }
    else//if original parameters not met, ---> execute else
    {
        printf("No negative or values over 9999 allowed \n");
        //condition
    }
}

void main()
{
    SSEG_Init();   //initialize SSEG_init
    SysTick_Init();//initialize SysTick
    SSEG_Off();    //initialize with LEDs off

    while (1)
    {
        input_num(1);//execute first
    }
}
```

✓

SSEG.c

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

/*********************************************
 * LED.B-[  U  ]-VCC
 * LED.C-[     ]-LED.A
 * LED.D-[     ]-Pin 14 - SER   Pin 5.2 &0x04
 * LED.E-[     ]-GND OE
 * LED.F-[     ]-Pin 12 - RCLK  Pin 5.1 &0x02
 * LED.G-[     ]-Pin 11 - SRCLK Pin 5.0 &0x01
 * LED.H-[     ]-VCC SRCLR
 * GND  -[     ]-None (QH')*/
//*************Global Variables*************
char digits[10] = {
              //  -gfedcbah
        0x81, //0 -10000001    Seven segment
        0xF3, //1 -11110011         _a_
        0x48, //2 -01001001    f - [   ] -b
        0x61, //3 -01100001        [_g_]
        0x33, //4 -00110011    e - [   ] - c
        0x25, //5 -00100101        [_d_](.)-h
        0x07, //6 -00000111
        0xF1, //7 -11110001
        0x01, //8 -00000001
        0x31  //9 -00110001
        };

int count = 0;//count is starting at 0
uint32_t wait_10ms = 30000; //10 ms a second delay.
/*
 * SSEG_Init
 * Initialize 7-segment display
 * Inputs: none
 * Outputs: none
 */
void SSEG_Init()
{
//******Port4 Inits*********
    P4SEL0 &= ~0x3F; // Using pins P4.0 to P4.3
    P4SEL1 &= ~0x3F; // 1,2,8,10 = 21 or 0x15
//******Port5 Inits*********
    P5SEL0 &= ~0x07; // Using pins P5.0 to P5.2
    P5SEL1 &= ~0x07; //
    P5DIR  |=  0x07; //
}
```

✓

```
/*
 * 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)
{

    uint8_t hex = digits[num];//hex variable
    int i; //index
    int bits[8] = { 0 };//array size 8                          ✓

    for (i = 0; i < 8; i++)//hex to binary
    {
        bits[i] = hex & 1; //hex to binary
        hex = hex >> 1;    //hex to binary
    }
    P5OUT &= ~0x02;//Read mode RCLK Clock_pin p5.1 (HIGH)
    for (i = 7; i > 0; i--)//decrement 8 bits to load
    {
        if (bits[i] == 0)  //load bit by bit
        {
            P5OUT &= ~0x04;//Latch_Pin set HIGH p5.2
        }
        else //wait until = 0, then do ->else
        {
            P5OUT |= 0x04; //Latch_Pin set LOW p5.2
        }
        //****SRCLK pulse here P5.0****
        P5OUT |= 0x01;  //LOW pulse  SER_data pin p5.0
        P5OUT &= ~0x01; //HIGH pulse SER_data pin p5.0
        //*****************************
    }
    P5OUT |= 0x02;//Write mode RCLK Clock_pin p5.1 (LOW)
    SSEG_Off();   //turn off p4 set to high, in between cycles

    if (count == 0) //if count 0 turn on pin 4.0
    {
        P4DIR |= 0x01;        //turn on p4.0 LOW
        SysTick_Wait(wait_10ms);        //Wait 10ms
    }
    else if (count == 1)//if count 1 turn on pin 4.1
    {
        P4DIR |= 0x02;        //turn on p4.1 LOW
        SysTick_Wait(wait_10ms);        //Wait 10ms
    }
    else if (count == 2)//if count 2 turn on pin 4.3
    {
        P4DIR |= 0x08;        //turn on p4.3 LOW
        SysTick_Wait(wait_10ms);        //Wait 10ms
    }
    else if (count == 3)//if count 3 turn on pin 4.4
    {
        P4DIR |= 0x10;        //turn on p4.4 LOW
        SysTick_Wait(wait_10ms);        //Wait 10ms
    }
}
/*
 * SSEG_Shift_Out Function
 * Shifts data out serially
 * Inputs: 8-bit data
 * Outputs: none                                         ✓
 */
void SSEG_Shift_Out(char *data)
{

    while (1)
    { //Infinite loop
        while (count < 4) //takes in 1 digit at a time from user input
        {
            SSEG_Out(data[count]);
            //send data from user input, to SSEG_Out 1 at a time
            //to be output serially/ parallel (AKA 1x1 to SSEG_Out)
            count++;
            //increment count
        }
        count = 0;            //reset count to zero
    }

}
/*
 * SSEG_Disp_Num Function
 * Separate the input number into 4 single digit
 * Inputs: num between 0 and 9999
 * Outputs: none
 */
void SSEG_Disp_Num(int digit)
{
    int cnt = 0; //initialize at zero
    char splitDigits[4];//array size 4
    while (cnt < 4)//count based on 4 inputs
    {
        splitDigits[3 - cnt] = digit % 10;
        //Separate
        digit = digit / 10;
        //separate
        cnt++;//until 4
    }
    SSEG_Shift_Out(splitDigits);
    //Separated inputs go to SSEG_Shift_Out();
}
```

```
/*
 * SSEG_Off Function
 * Turns off all 7-seg digits
 * Inputs: none
 * Outputs: none
 */
void SSEG_Off()
{   //Set all pins D1,D2,D3,D4 'HIGH' to turn
    //off seven segments 4 displays
    P4DIR &= ~0x01;       //turn off p4.0 HIGH
    P4DIR &= ~0x02;       //turn off p4.2 HIGH
    P4DIR &= ~0x08;       //turn off p4.3 HIGH
    P4DIR &= ~0x10;       //turn off p4.4 HIGH
    return;
}
```

✓

SSEG.h

```c
/**************** Public Functions ****************/
/*
 * SSEG_Init Function
 * Initialize 7-segment display
 * Inputs: none
 * Outputs: none
 */
void SSEG_Init();

/*
 * SSEG_Out Function
 * Output a 4-digit number to the display
 * Inputs: none
 * Outputs: none
 */
void SSEG_Out(uint8_t num);

/*
 * SSEG_Shift_Out Function
 * Shifts data out serially
 * Inputs: 8-bit data
 * Outputs: none
 */
void SSEG_Shift_Out(char *data);

/*
 * SSEG_Disp_Num Function
 * Separate the input number into 4 single digit
 * Inputs: num between 0 and 9999
 * Outputs: none
 */
void SSEG_Disp_Num(int digit);

/*
 * SSEG_Off Function
 * Turns off all 7-seg digits
 * Inputs: none
 * Outputs: none
 */
void SSEG_Off();
```