```
// CECS346 Project1: Traffic light controller with FSM
// Student Name:Brandon Jamjampour,Daniel Banuelos, Anastacia Estrella
// Lab description:Implementing a FSM Traffic light controller using two Ports
which were port B and the on board Leds PortF green and red
//for the ped to walk or stop, and used port B to control 6 leds for south
direction of cars and west direction of cars and
//3 sensors to detect a car on south a car on west or a ped depending on the inputs
the traffic light controller would allow
//cars to go and share or let the ped walk across safely, also implementing a flash
element to tell the ped that they are
//running out of time to cross

// Hardware Design
// 1) Port B will be used to control 6 LEDs: (PB5) Red west, (PB4) Yellow West,
(PB3) Green West,
//(PB2) Red South, (PB1) Yellow South, (PB0) Green South
//Port F will be Pedistrain Light for the onboard leds
//(PF3) Green led for ped to walk (PB1) Red to not let ped and flash to hurry up
// 2) Port E will be used for the 3 switches/sensors: (PE2) South, West (PE1), Ped
Sensor (PE0)

#include <stdint.h>   // for data type alias


// Registers for switches
// Complete the following register definitions
// Registers for sensor port E sensor
#define SENSOR                                                   (*((volatile
unsigned long *)0x4002401C))  //portE bit addresses PE2 - PE0
#define GPIO_PORTE_DATA_R       (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))

//// Registers for LEDs traffic light port B
#define T_LIGHT                  (*((volatile unsigned long *)0x400050FC)) // bit
addresses for the four LEDs on PB5 - PB0
#define GPIO_PORTB_DIR_R       (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_AFSEL_R     (*((volatile unsigned long *)0x40005420))
#define GPIO_PORTB_DEN_R       (*((volatile unsigned long *)0x4000551C))
#define GPIO_PORTB_AMSEL_R     (*((volatile unsigned long *)0x40005528))
#define GPIO_PORTB_PCTL_R      (*((volatile unsigned long *)0x4000552C))
#define SYSCTL_RCGC2_R         (*((volatile unsigned long *)0x400FE108))

// Register for Pedistrain on port F
#define P_LIGHT                (*((volatile unsigned long *)0x40025028))
#define GPIO_PORTF_DIR_R       (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R     (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_DEN_R       (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_AMSEL_R     (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R      (*((volatile unsigned long *)0x4002552C))


// Constants definitions SysTick
#define     NVIC_ST_CTRL_R                           (*((volatile unsigned long
*)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
```

```
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))
#define NVIC_ST_CTRL_COUNT      0x00010000
#define NVIC_ST_CTRL_CLK_SRC    0x00000004
#define     NVIC_ST_CTRL_ENABLE     0x00000001
#define NVIC_ST_RELOAD_M           0x00FFFFFF
#define wait_quarter_sec           4000000

#define SYSCTL_RCGC2_GPIOB     0x00000002   // port B Clock Gating Control
#define SYSCTL_RCGC2_GPIOE     0x00000010   // port E Clock Gating Control
#define SYSCTL_RCGC2_GPIOF     0x00000020             //port F clock gating control

void T_Light_Init(void);
void P_Light_Init(void);
void Sensor_Init(void);
void SysTick_init(void);//always put initilzation into main
void SysTick_Wait(unsigned long delay);
void Wait_HalfSecond(unsigned long delay);
// FSM state data structure
struct State {
  uint8_t Out;
      uint8_t OutPed;
  uint32_t Time;
  uint32_t Next[8];
};

typedef const struct State STyp;

// Constants definitions
enum my_states {GoS,WaitS,GoW,WaitW,GoP,WaitPon1,WaitPoff1,WaitPon2,WaitPoff2};//
define all states goN and assings it 0 1 two 3 increments by one each time

// Output pins are:3(white), 2(red), 1(yellow), 0(green)
// Input pins are: 1:sw2, 0:sw1
STyp FSM[9]={//this is the fsm that allows the states and 8 possible tranistions
      {0x21,0x02,8,{GoS,WaitS,WaitS,WaitS,GoS,WaitS,WaitS,WaitS}},
      {0x22,0x02,4,{GoW,GoP,GoW,GoW,GoP,GoP,GoW,GoP}},
      {0x0C,0x02,8,{GoW, WaitW, GoW,WaitW, WaitW,WaitW,WaitW,WaitW}},
      {0x14,0x02,4,{GoS,GoP,GoS,GoP,GoS,GoP,GoS,GoS}},     //changed
  {0x24,0x08,8,{GoP,GoP,WaitPon1,WaitPon1,WaitPon1,WaitPon1,WaitPon1,WaitPon1}},
      {0x24,0x08,1,
{WaitPoff1,WaitPoff1,WaitPoff1,WaitPoff1,WaitPoff1,WaitPoff1,WaitPoff1,WaitPoff1}},
      {0x24,0x02,1,
{WaitPon2,WaitPon2,WaitPon2,WaitPon2,WaitPon2,WaitPon2,WaitPon2,WaitPon2}},
      {0x24,0x08,1,
{WaitPoff2,WaitPoff2,WaitPoff2,WaitPoff2,WaitPoff2,WaitPoff2,WaitPoff2,WaitPoff2}},
      {0x24,0x02,1,{GoS,GoP,GoW,GoW,GoS,GoS,GoW,GoW}},//changed
      };

int main(void){
uint32_t S;  // index to the current state
uint32_t Input;

      T_Light_Init();
      P_Light_Init();
      Sensor_Init();
      SysTick_init();
      S = GoS;    // FSM start with green
  while(1){
            T_LIGHT = FSM[S].Out;//set the output to port B to the traffic light
depends on state
            P_LIGHT = FSM[S].OutPed;//set the output to port F on board depending
on the state
            Wait_HalfSecond(FSM[S].Time);
            Input = SENSOR;
            S = FSM[S].Next[Input];
            // Put your FSM engine here
  }
}
void SysTick_init(void){
      NVIC_ST_CTRL_R = 0;
      NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M;
      NVIC_ST_CURRENT_R = 0;

      NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE + NVIC_ST_CTRL_CLK_SRC;
}
void SysTick_Wait(unsigned long delay){
      NVIC_ST_RELOAD_R = delay - 1;
      NVIC_ST_CURRENT_R = 0;
      while((NVIC_ST_CTRL_R & NVIC_ST_CTRL_COUNT) == 0){}
}

void Wait_HalfSecond(unsigned long delay){
      unsigned long i;
      for(i = 0; i < delay; i++){
            SysTick_Wait(wait_quarter_sec);//create 0.25second delay
            }
}

void Sensor_Init(void){
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOE;      // Activate Port E clocks
      while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOE)!=SYSCTL_RCGC2_GPIOE){} // wait for
clock to be active
//
      GPIO_PORTE_AMSEL_R &= ~0x07; // Disable analog function on PE2-0
  GPIO_PORTE_PCTL_R &= ~0x00000FFF; // Enable regular GPIO
  GPIO_PORTE_DIR_R &= ~0x07;    // Inputs on PE2-0
  GPIO_PORTE_AFSEL_R &= ~0x07;  // Regular function on PE2-0
  GPIO_PORTE_DEN_R |= 0x07;    // Enable digital signals on PE2-0
}

void T_Light_Init(void){
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB;       // Activate Port B clocks
      while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOB)!=SYSCTL_RCGC2_GPIOB){} // wait for
clock to be active
//
  GPIO_PORTB_AMSEL_R &= ~0x3F; // Disable analog function on PB5-0
  GPIO_PORTB_PCTL_R &= ~0x00FFFFFF; // Enable regular GPIO
  GPIO_PORTB_DIR_R  |= 0x3F;   // Outputs on PB5-0
  GPIO_PORTB_AFSEL_R &= ~0x3F; // Regular function on PB5-0
  GPIO_PORTB_DEN_R  |= 0x3F;  // Enable digital on PB5-0
}

void P_Light_Init(void){
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF;       // Activate Port F clocks
      while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOF)!=SYSCTL_RCGC2_GPIOF){} // wait for
clock to be active
//
  GPIO_PORTF_AMSEL_R &= ~0x0A; // Disable analog function on PF3 and PF1
  GPIO_PORTF_PCTL_R &= ~0x0000F0F0; // Enable regular GPIO
  GPIO_PORTF_DIR_R  |= 0x0A;   // Outputs on PF3 PF1
  GPIO_PORTF_AFSEL_R &= ~0x0A; // Regular function on PF3 PF1
  GPIO_PORTF_DEN_R  |= 0x0A;  // Enable digital on PF3 PF1
}
```