```c
// Paul Valenzuela && Alfredo Leyva
// Project 2
// 11/8/19
// Drag Race

#include <stdint.h> // C99 data types
#include "tm4c123gh6pm.h"

#define SENSOR                    (*((volatile unsigned long *)0x40004030))      // Port A Pins 2 and 3
#define LEFTSIGNAL                (*((volatile unsigned long *)0x4000503C))      // Port B Pins 0-3
#define RIGHTSIGNAL               (*((volatile unsigned long *)0x40024078))      // Port E Pins 1-4

/*
    PA3   Left  Signal
    PA2   Right Signal
    PB3   Left Yellow Countdown 1
    PB2   Left Yellow Countdown 2
    PB1   Left Green
    PB0   Left Red
    PE4   Right Yellow Countdown 1
    PE3   Right Yellow Countdown 2
    PE2   Right Green
    PE1   Right Red
    PF0   Reset Button
*/

// State Machine Structure
struct State {
  unsigned long Out;                        // Output
  unsigned long Time;                       // Delay Time
  unsigned long Next[4];};                  // List of Next States

typedef const struct State STyp;
// Defining States
#define Initial              0
#define WaitForSetup         1
#define Countdown1           2
#define Countdown2           3
#define Go                   4
#define FalseStartRight      5
#define FalseStartLeft       6
#define FalseStartBoth       7
#define WinRight             8
#define WinLeft              9
#define WinBoth              10

// State Machine
STyp FSM[11]={
 {0xFF, 100,{1,1,1,1}},      // Initial State
 {0x00, 100,{1,1,1,2}},      // Wait For Setup
 {0x88, 100,{7,6,5,3}},      // Countdown 1
 {0x44, 100,{7,6,5,4}},      // Countdown 2
 {0x22, 1,{10,9,8,4}},       // Go
 {0x01, 100,{1,1,1,1}},      // False Start Right
 {0x10, 100,{1,1,1,1}},      // False Start Left
 {0x11, 100,{1,1,1,1}},      // False Start Both
 {0x02, 100,{1,1,1,1}},      // Win Right
 {0x20, 100,{1,1,1,1}},      // Win Left
 {0x22, 100,{1,1,1,1}},      // Win Both
};

unsigned long CS;                                  // Index of the Current State
unsigned long Input;                               // Input That Decides Next State

// Function Prototypes (from startup.s)
void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
void WaitForInterrupt(void);  // Go to low power mode while waiting for the next interrupt

// Implemented Function Prototypes
void SysTick_Init(unsigned int);                 // Initialize SysTick timer for 0.1s delay

void ResetSensor_Init(void);                       // Initialize Reset Button With Interrupt Priority 1
void SwitchSensor_Init(void);                      // Initialize Sensor Buttons With Interrupt Priority 2

void PortB_LED_Init(void);                         // Initialize Port B (Left  LED) Configuration
void PortE_LED_Init(void);                         // Initialize Port E (Right LED) Configuration

void GPIOPortA_Handler(void);                      // Handle Port A Interrupts
void GPIOPortF_Handler(void);                      // Handle Port F Interrupts
void SysTick_Handler(void);                        // Handle SysTick generated interrupts


// Main Method
int main(void){
  EnableInterrupts();
  PortB_LED_Init();                                  // Initialize GPIO Port B for LEDs
  PortE_LED_Init();                                  // Initialize GPIO Port E for LEDs
  SwitchSensor_Init();                               // Initialize GPIO Port A for Sensors
  ResetSensor_Init();                                // Initialize GPIO Port F for Reset
  SysTick_Init(100);                                 // Initialize the clock

  CS = Initial;                                      // Set First State as Initial State
  LEFTSIGNAL = (FSM[CS].Out & 0x00F0) >> 4;          // Left  Signal Grabs Output Bits
  RIGHTSIGNAL  = (FSM[CS].Out & 0x000F) << 1;        // Right Signal Grabs Output Bits

  while(1){
    WaitForInterrupt();                              // Wait For Interrupts To Occur
  }
}

// Initialize Port B LEDs
void PortB_LED_Init() {
  uint32_t volatile delay;          // Declaring variable that will be used to setup PORT B

  SYSCTL_RCGC2_R |= 0x00000002;     // Turn on B clock
  delay = SYSCTL_RCGC2_R;           // Delay 3-5 bus cycles

  GPIO_PORTB_AMSEL_R = 0x00;        // Disable analog function
  GPIO_PORTB_PCTL_R = 0x00000000;   // GPIO clear bit PCTL
  GPIO_PORTB_DIR_R |= 0x0F;         // PB3, PB2, PB1, PB0 output
  GPIO_PORTB_AFSEL_R = 0x00;        // No alternate function

  GPIO_PORTB_DEN_R |= 0x0F;         // Enable digital pins PB3-PB0
}

// Initialize Port E LEDs
void PortE_LED_Init() {
  uint32_t volatile delay;          // Declaring variable that will be used to setup PORT E

  SYSCTL_RCGC2_R |= 0x00000010;     // Turn on E clock
  delay = SYSCTL_RCGC2_R;           // Delay 3-5 bus cycles

  GPIO_PORTE_AMSEL_R = 0x00;        // Disable analog function
  GPIO_PORTE_PCTL_R = 0x00000000;   // GPIO clear bit PCTL
  GPIO_PORTE_DIR_R |= 0x1E;         // PE3, PE2, PE1, PE4 output
  GPIO_PORTE_AFSEL_R = 0x00;        // No alternate function

  GPIO_PORTE_DEN_R |= 0x1E;         // Enable digital pins PE4-PE1
}

// Initialize SysTick timer for 0.01s delay with interrupt enabled
void SysTick_Init(unsigned int amp) {
  NVIC_ST_CTRL_R &= 0x00;                                  // Disable SysTick during setup
  NVIC_ST_RELOAD_R = 16e4 * amp;                           // Maximum reload value .01 Seconds
  NVIC_ST_CURRENT_R = 0;                                   // Any write to current clears it

  NVIC_ST_CTRL_R |= 0x07;                                  // Enable SysTick with core clock
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF00FFFF) | 0x00700000;   // Sets Priority 3
}

// Initialize edge trigger interrupt for PF0 (SW2) rising edge
void ResetSensor_Init() {
  uint32_t volatile delay;          // Declaring variable that will be used to setup PORT F

  SYSCTL_RCGC2_R |= 0x00000020;     // Turn on F clock
  delay = SYSCTL_RCGC2_R;           // Delay 3-5 bus cycles

  GPIO_PORTF_LOCK_R |= 0x4C4F434B;   // Unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x01;           // Allow changes to PF0
  GPIO_PORTF_DIR_R &= ~0x01;         // PF0 Input
  GPIO_PORTF_PUR_R |= 0x01;          // Enable pullup resistors on PF0
  GPIO_PORTF_DEN_R |= 0x01;          // Digital Enable PF0

  GPIO_PORTF_IS_R &= ~0x01;          // PF0 is edge-Sensitive
  GPIO_PORTF_IBE_R &= ~0x01;         // PF0 is not both edges
  GPIO_PORTF_IEV_R |= 0x01;          // PF0 Falling edge event
  GPIO_PORTF_ICR_R |= 0x01;          // Clear flag0
  GPIO_PORTF_IM_R |= 0x01;           // arm interrupt on PF0

  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF00FFFF) | 0x00200000; // Priority 1
  NVIC_EN0_R |= 0x40000000;                           // Enable interrupt 30 NVIC
}

// Initialize edge trigger interrupt for PA3 PA2 rising edge
void SwitchSensor_Init(void)
{
  uint32_t volatile delay;            // Declaring variable that will be used to setup PORT A

  SYSCTL_RCGC2_R |= 0x00000001;       // Turn on A clock
  delay = SYSCTL_RCGC2_R;             // Delay 3-5 bus cycles

  GPIO_PORTA_DIR_R &= ~0x0C;          // PA3, PA2 Input
  GPIO_PORTA_DEN_R |= 0x0C;           // Digital Enable PA3, PA2


  GPIO_PORTA_IS_R &= ~0x0C;           // PA3, PA2 is edge-Sensitive
  GPIO_PORTA_IBE_R |= 0x0C;           // PA3, PA2 is both edges
  //GPIO_PORTA_IEV_R                  // Since IEV is undefined, Port A is both edge triggered
  GPIO_PORTA_ICR_R |= 0x0C;           // Clear flag3, flag2
  GPIO_PORTA_IM_R |= 0x0C;            // Arm interrupt on PA3, PA2

  NVIC_PRI7_R = (NVIC_PRI7_R&0xFFFFFF00) | 0x00000040; // Priority 2
  NVIC_EN0_R |= 0xFFFFFFFF;                           // enable interrupt 30 NVIC

}

// Handle SysTick generated interrupts. When timer interrupt triggers, do what's necessary then decide
the next state based on inputs.
void SysTick_Handler(void) {
    CS = FSM[CS].Next[Input];
    LEFTSIGNAL = (FSM[CS].Out & 0x00F0) >> 4;             // Left  Signal Grabs Output Bits
    RIGHTSIGNAL  = (FSM[CS].Out & 0x000F) << 1;           // Right Signal Grabs Output Bits
    SysTick_Init(FSM[CS].Time);                           // Reset Timer
}

// Handle GPIO Port F interrupts. When Port F interrupt triggers, do what's necessary then send output
signals to the circuit.
void GPIOPortF_Handler(void) {
    CS = Initial;
    LEFTSIGNAL = (FSM[CS].Out & 0x00F0) >> 4;             // Left  Signal Grabs Output Bits
    RIGHTSIGNAL  = (FSM[CS].Out & 0x000F) << 1;           // Right Signal Grabs Output Bits
    SysTick_Init(FSM[CS].Time);                           // Reset Timer
    GPIO_PORTF_ICR_R = 0x01;                              // Acknowledge flag0
}

// Handle GPIO Port A interrupts
void GPIOPortA_Handler(void) {
    Input = SENSOR >> 2;          // Grab The Inputs From PA3, PA2
    GPIO_PORTA_ICR_R = 0x0C;      // Acknowldge Flags
}

```