1. CECS346-04 - Microprocessors and Controllers I

Project 3 – Final Project – Drag Race + Car

Sean Masterson

i.

|  |  |
| --- | --- |
| **Description** | **Pin** |
| Reset | PA3 |
| Red left LED | PB0 |
| Green left LED | PB1 |
| Yellow 2 left LED | PB2 |
| Yellow 1 left LED | PB3 |
| Red right LED | PB4 |
| Green right LED | PB5 |
| Yellow 2 right LED | PB6 |
| Yellow 1 right LED | PB7 |
| Switch - Left lane car | PF0 |
| IR sensor - Right lane car | PF4 |

ii.

|  |  |
| --- | --- |
| **Description** | **Pin** |
| IR sensor | PA3 |
| Left stepper motor IN1 | PB0 |
| Left stepper motor IN2 | PB1 |
| Left stepper motor IN3 | PB2 |
| Left stepper motor IN4 | PB3 |
| Right stepper motor IN1 | PB4 |
| Right stepper motor IN2 | PB5 |
| Right stepper motor IN3 | PB6 |
| Right stepper motor IN4 | PB7 |
| Stage - On-board button 2 | PF0 |
| Race - On-board button 1 | PF4 |

c.

i. The input maps to PF0 and PF4 representing the left (switch) and right (IR sensor) lane car respectively. Pin PA3 is used for reset.

ii. The input maps to move, forward, and backwards in that order. Move is active when either PF0 (on-board button 2) and/or PF4 (on-board button 1) is pressed (falling edge interrupt) and inactive when PA3 (IR sensor) is level. Forward is active when PF4 is pressed (falling edge event). Backwards is active when PF0 is pressed (falling edge event).

d.

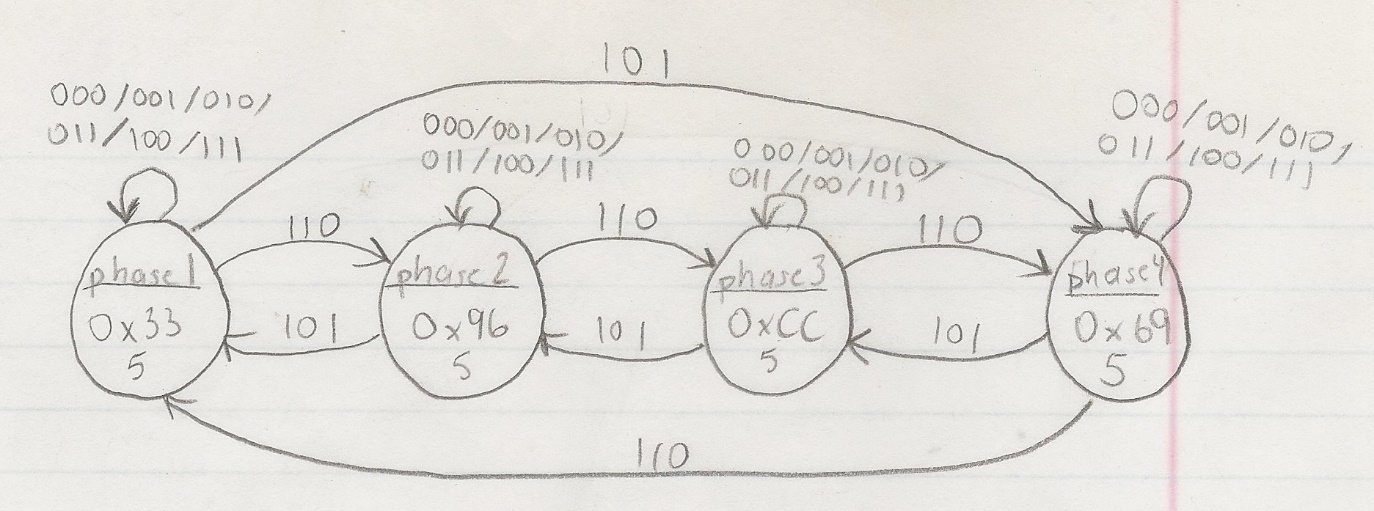
i. The output maps to pins PB7, PB6, PB5, PB4, PB3, PB2, PB1, and PB0 representing the Y1R, Y2R, GR, RR, Y1L, Y2L, GL, and RL LEDs respectively. The time is used for setting the delay between states.

ii. The output maps to pins PB7, PB6, PB5, PB4, PB3, PB2, PB1, and PB0 representing the right stepper motor IN4, IN3, IN2, IN1 and the left stepper motor IN4, IN3, IN2, and IN1 respectively. The time is used for setting the delay between states.

e.

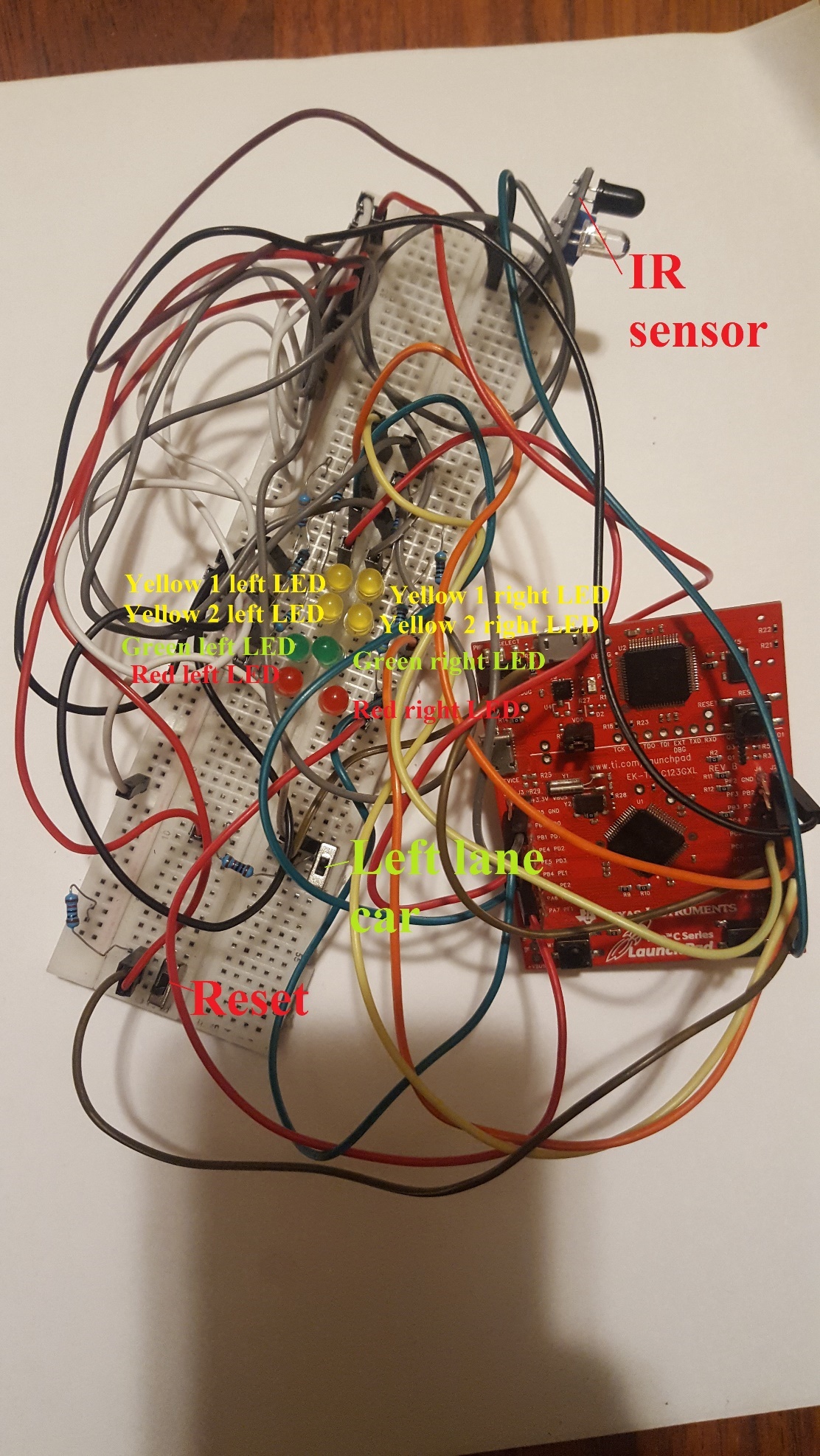
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **{move, forward, backward}** | **000** | **001** | **010** | **011** | **100** | **101** | **110** | **111** |
| **phase1 (0x33, 5)** | phase1 | phase1 | phase1 | phase1 | phase1 | phase4 | phase2 | phase1 |
| **phase2 (0x96, 5)** | phase2 | phase2 | phase2 | phase2 | phase2 | phase1 | phase3 | phase2 |
| **phase3 (0xCC, 5)** | phase3 | phase3 | phase3 | phase3 | phase3 | phase2 | phase4 | phase3 |
| **phase4 (0x69, 5)** | phase4 | phase4 | phase4 | phase4 | phase4 | phase3 | phase1 | phase4 |

f.

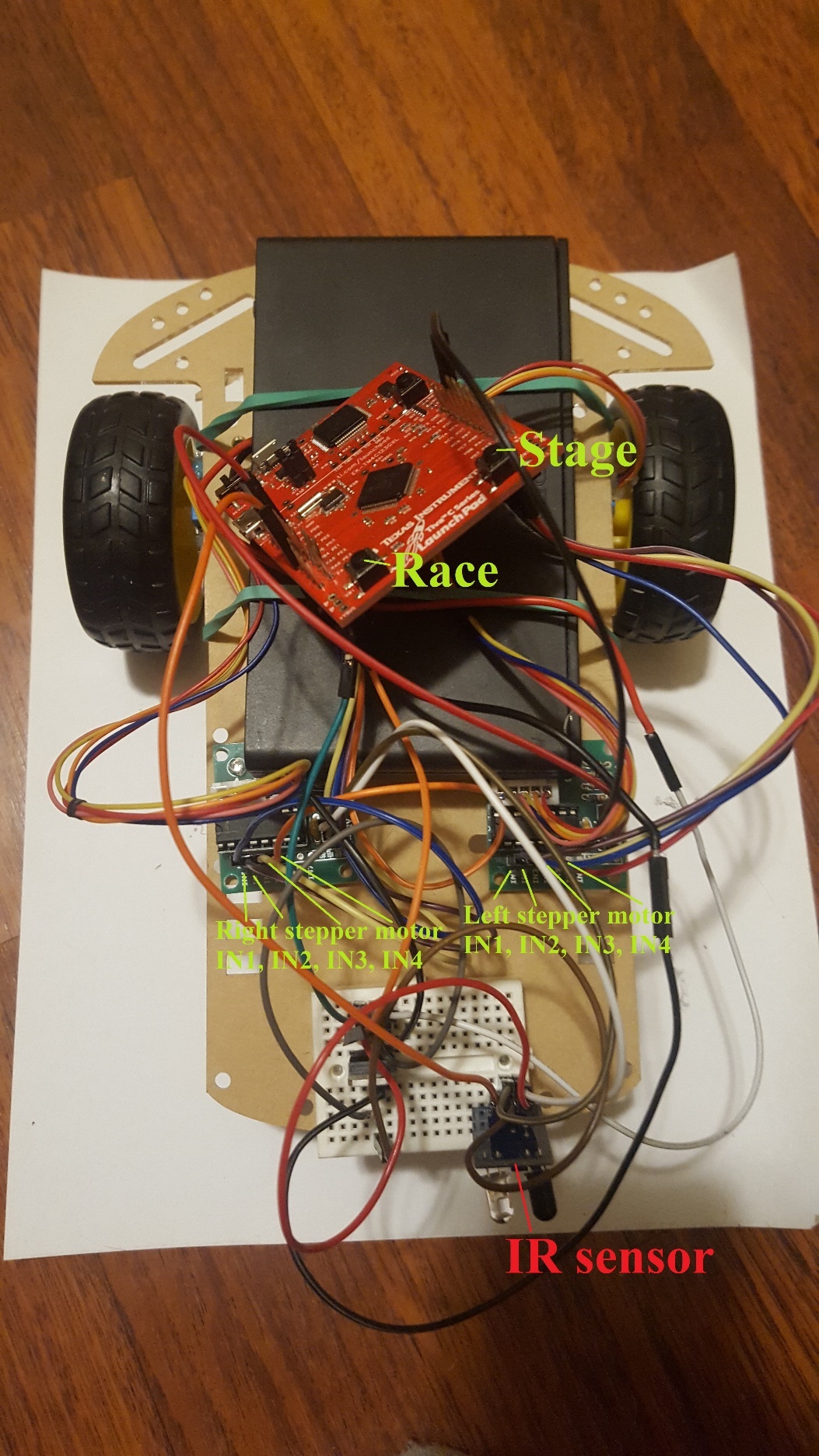


g.

i.

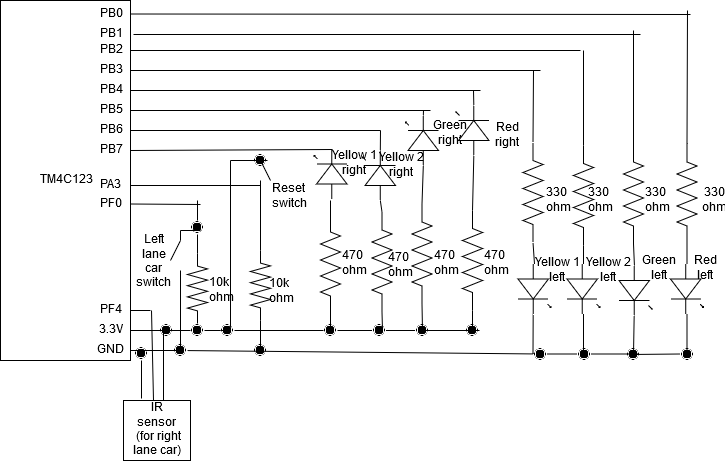


ii.

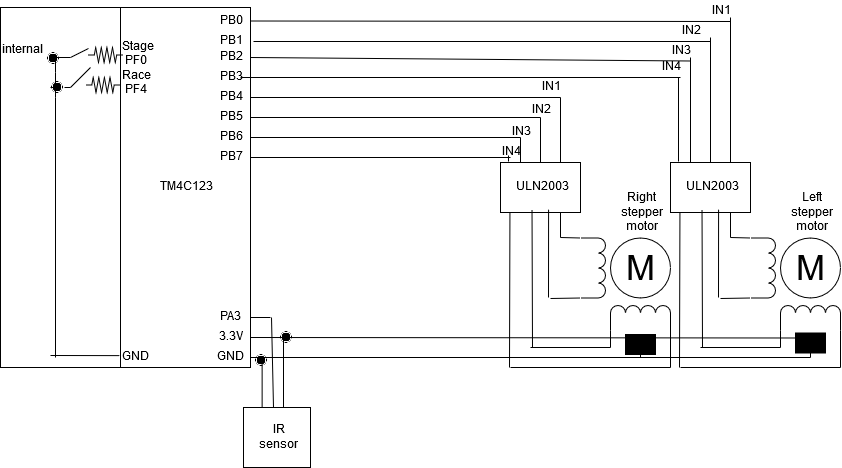


h.

i.



ii.

i.

**i.**

// Documentation

// CECS346 Project 3 Part 1 - Drag Race

// Description: Design a system based on a drag race. A 'Christmas Tree' is the light signal that drives us to

// determine when to accelerate (ie start the race), which is simulated by our LEDs. A track has

// multiple sensors to determine if a car is in the proper position ("staged"), simulated by pressing

// our "lane sensor" buttons and IR sensor, one per lane (left lane and right lane).

// Student: Sean Masterson

// Input/Output:

// PB7 - Yellow LED 1 right

// PB6 - Yellow LED 2 right

// PB5 - Green LED right

// PB4 - Red LED right

// PB3 - Yellow LED 1 left

// PB2 - Yellow LED 2 left

// PB1 - Green LED left

// PB0 - Red LED left

// PA3 - Reset input

// PF4 - IR sensor (also connected to on-board switch 1)

// PF0 - Switch on breadboard (also connected to on-board switch 2 )

// Preprocessor Directives

#include <stdint.h>

#include "tm4c123gh6pm.h"

// Function Prototypes (from startup.s)

void DisableInterrupts(); // Disable interrupts

void EnableInterrupts(); // Enable interrupts

void WaitForInterrupt(); // Go to low power mode while waiting for the next interrupt

// Function Prototypes

void EdgeCounter\_InitF(); // Initialize edge trigger interrupt for PF4+0 (SW1+2) both edges

void EdgeCounter\_InitA(); // Initialize edge trigger interrupt for PA3 level

void Switch\_InitA(void);

void Switch\_InitB(void);

void Switch\_InitF(void);

void SysTick\_Init(); // Initialize SysTick timer for 0.1s delay with interrupt enabled

void GPIOPortF\_Handler(); // Handle GPIO Port F interrupts

void GPIOPortA\_Handler(); // Handle GPIO Port A interrupts

void SysTick\_Handler(); // Handle SysTick generated interrupts

#define LIGHT\_PORTB (\*((volatile unsigned long \*)0x400053FC)) // PB01234567

#define SENSOR\_PORTA (\*((volatile unsigned long \*)0x40004020)) // PA3

#define SENSOR\_PORTF (\*((volatile unsigned long \*)0x40025044)) // PF04

struct State {

uint32\_t Out;

uint32\_t Time; // 1 ms units

uint32\_t Next[4]; // list of next states

};

typedef const struct State STyp;

#define initialize 0

#define waitForStaging 1

#define countdownY1 2

#define countdownY2 3

#define go 4

#define winLeft 5

#define winRight 6

#define winBoth 7

#define falseStartLeft 8

#define falseStartRight 9

#define falseStartBoth 10

STyp FSM [11] = {

{0xFF, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}}, // initialize

{0x00, 500, {waitForStaging, waitForStaging, waitForStaging, countdownY1}}, // waitForStaging

{0x88, 500, {falseStartBoth, falseStartLeft, falseStartRight, countdownY2}}, // countdownY1

{0x44, 500, {falseStartBoth, falseStartLeft, falseStartRight, go}}, // countdownY2

{0x22, 50, {winBoth, winLeft, winRight, go}}, // go

{0x20, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}}, // winLeft

{0x02, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}}, // winRight

{0x22, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}}, // winBoth

{0x10, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}}, // falseStartLeft

{0x01, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}}, // falseStartRight

{0x11, 1000, {waitForStaging, waitForStaging, waitForStaging, waitForStaging}} // falseStartBoth

};

uint8\_t S; // index of current state

uint8\_t Input; // input obtained from switches

int main(void) {

SysTick\_Init(); // initialize SysTick timer

EdgeCounter\_InitF();

EdgeCounter\_InitA();

// Initialize GPIO on Ports A, B, F

// Port A Init

Switch\_InitA();

// Port B Init

Switch\_InitB();

// Port F Init

Switch\_InitF();

// Initial state

S = initialize;

LIGHT\_PORTB = ((~FSM[S].Out & 0xF0)) + ((FSM[S].Out & 0x0F)); // set lights port b to initial state

while(1) {

WaitForInterrupt();

}

}

// Subroutine to wait about 0.1 sec

// Inputs: None

// Outputs: None

// Notes: the Keil simulation runs slower than the real board

#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 // Count flag

#define NVIC\_ST\_CTRL\_CLK\_SRC 0x00000004 // Clock Source

#define NVIC\_ST\_CTRL\_INTEN 0x00000002 // Interrupt enable

#define NVIC\_ST\_CTRL\_ENABLE 0x00000001 // Counter mode

#define NVIC\_ST\_RELOAD\_M 0x00FFFFFF // Counter load value

void Switch\_InitA(void){ // input

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x01; // 1) activate clock for Port A

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

// 2) no need to unlock GPIO Port A

GPIO\_PORTA\_DIR\_R &= ~0x08; // 3) input on PA3

GPIO\_PORTA\_AFSEL\_R &= ~0x08; // 4) PCTL GPIO on PA3

GPIO\_PORTA\_DEN\_R |= 0x08; // 5) enable PA3

}

void Switch\_InitB(void){ // output

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x02; // 1) activate clock for Port B

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

// 2) no need to unlock GPIO Port B

GPIO\_PORTB\_DIR\_R |= 0xFF; // 3) output PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7

GPIO\_PORTB\_AFSEL\_R &= ~0xFF; // 4) PCTL GPIO on PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7

GPIO\_PORTB\_DEN\_R |= 0xFF; // 5) enable PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7

}

void Switch\_InitF(void){ // input

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x20; // 1) activate clock for Port F

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock PortF (for PF0)

GPIO\_PORTF\_CR\_R |= 0x01; // allow changes to PF0 (PF4-1 unlocked by default)

GPIO\_PORTF\_DIR\_R &= ~0x11; // 3) input on PF4, PF0

GPIO\_PORTF\_AFSEL\_R &= ~0x11; // 4) PCTL GPIO on PF4, PF0

GPIO\_PORTF\_PUR\_R |= 0x11; // 4.5) Enable pull up resistor

GPIO\_PORTF\_DEN\_R |= 0x11; // 5) enable PF4, PF0

}

// Initialize SysTick timer for 0.1s delay with interrupt enabled

void SysTick\_Init() {

NVIC\_ST\_CTRL\_R = 0;

NVIC\_ST\_RELOAD\_R = 16000000 - 1; // 1 s / (1 / 16 MHz)

NVIC\_ST\_CURRENT\_R = 0;

// priority 3

NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R & 0x00FFFFFF) | 0x60000000;

// enable SysTick with core clock and interrupts

NVIC\_ST\_CTRL\_R |= 0x07; // 0111

EnableInterrupts();

}

// Initialize edge trigger interrupt for PF4, PF0 (SW1+2) both edges

void EdgeCounter\_InitF() { // must be friendly

SYSCTL\_RCGC2\_R |= 0x20; // (a) actiate clock for port F

GPIO\_PORTF\_DIR\_R &= ~0x11; // (c) make PF4, PF0 in (built-in button)

GPIO\_PORTF\_AFSEL\_R &= ~0x11; // disable alt funct on PF4, PF0

GPIO\_PORTF\_DEN\_R |= 0x11; // enable digital I/O on PF4, 0

GPIO\_PORTF\_PCTL\_R &= ~0x000F000F; // configure PF4, 0 as GPIO

GPIO\_PORTF\_AMSEL\_R &= ~0x11; // disable analog functionality on PF4, 0

GPIO\_PORTF\_PUR\_R |= 0x11; // enable weak pull-up on PF4, 0

GPIO\_PORTF\_IS\_R &= ~0x11; // (d) PF4, 0 is edge-sensitive

GPIO\_PORTF\_IBE\_R |= 0x11; // PF4, 0 is both edges

//GPIO\_PORTF\_IEV\_R |= 0x11; // PF4, 0 rising edge event

GPIO\_PORTF\_ICR\_R = 0x11; // (e) clear flag0, 4

GPIO\_PORTF\_IM\_R |= 0x11; // (f) arm interrupt on PF0, 4

NVIC\_PRI7\_R = (NVIC\_PRI7\_R & 0xFF00FFFF) | 0x00400000; // (g) priority 2

NVIC\_EN0\_R = 0x40000000; // (h) enable interrupt 30 in NVIC

EnableInterrupts(); // (i) Clears the I bit

}

// Initialize level trigger interrupt for PA3

void EdgeCounter\_InitA() { // must be friendly

SYSCTL\_RCGC2\_R |= 0x01; // (a) actiate clock for port A

GPIO\_PORTA\_DIR\_R &= ~0x08; // (c) make PA3 input

GPIO\_PORTA\_AFSEL\_R &= ~0x08; // disable alt funct on PA3

GPIO\_PORTA\_DEN\_R |= 0x08; // enable digital I/O on PA3

GPIO\_PORTA\_PCTL\_R &= ~0x0000F000; // configure PF3 as GPIO

GPIO\_PORTA\_AMSEL\_R &= ~0x08; // disable analog functionality on PA3

GPIO\_PORTA\_IS\_R |= 0x08; // (d) PA3 is level-sensitive

GPIO\_PORTA\_IBE\_R &= ~0x08; // PA3 is not both edges

GPIO\_PORTA\_IEV\_R |= 0x08; // PA3 level high

GPIO\_PORTA\_ICR\_R = 0x08; // (e) clear flag3

GPIO\_PORTA\_IM\_R |= 0x08; // (f) arm interrupt on PA3

NVIC\_PRI0\_R = (NVIC\_PRI0\_R & 0xFFFFFF00) | 0x00000020; // (g) priority 1

NVIC\_EN0\_R = 0x00000001; // (h) enable interrupt 0 in NVIC

EnableInterrupts(); // (i) Clears the I bit

}

// Handle GPIO Port F interrupts. When Port F interrupt triggers, go to next state

void GPIOPortF\_Handler() {

GPIO\_PORTF\_ICR\_R = 0x11; // Clear flag for port F pins 0 and 4.

Input = ((~SENSOR\_PORTF & 0x10 ) >> 3) + (~SENSOR\_PORTF & 0x01); // read sensors and shift bits

}

// Handle GPIO Port A interrupts. When Port A interrupt triggers, reset

void GPIOPortA\_Handler() {

GPIO\_PORTA\_ICR\_R = 0x08; // Clear flag for port A pin 3.

S = initialize;

LIGHT\_PORTB = 0x0F; // set lights port b. Note pins 7-4 are negative logic and 3-0 positive.

SysTick\_Init(); // Reset timer value to that of initial state

}

// Handle SysTick generated interrupts. When timer interrupt triggers, do what's necessary

void SysTick\_Handler() {

S = FSM[S].Next[Input];

LIGHT\_PORTB = ((~FSM[S].Out & 0xF0)) + ((FSM[S].Out & 0x0F)); // set lights port b

// Reload timer with new value

NVIC\_ST\_CTRL\_R = 0; // stop timer

NVIC\_ST\_RELOAD\_R = (16000 \* FSM[S].Time) - 1; // Wait FSM[S].Time milliseconds (16000 ticks = 1 ms)

NVIC\_ST\_CURRENT\_R = 0; // delete count value

NVIC\_ST\_CTRL\_R |= 0x07; // 0111, start timer

}

**ii.**

// Documentation

// CECS346 Project 3 Part 2 - Car

// Description: Connect two stepper motors to TM4C.

// Student: Sean Masterson

// Input/Output:

// PA3 - IR sensor

// PB7 - Right stepper motor IN4

// PB6 - Right stepper motor IN3

// PB5 - Right stepper motor IN2

// PB4 - Right stepper motor IN1

// PB3 - Left stepper motor IN4

// PB2 - Left stepper motor IN3

// PB1 - Left stepper motor IN2

// PB0 - Left stepper motor IN1

// PF4 - On-board button 1

// PF0 - On-board button 2

// Preprocessor Directives

#include <stdint.h>

#include "tm4c123gh6pm.h"

// Function Prototypes (from startup.s)

void DisableInterrupts(); // Disable interrupts

void EnableInterrupts(); // Enable interrupts

void WaitForInterrupt(); // Go to low power mode while waiting for the next interrupt

// Function Prototypes

void EdgeCounter\_InitA(); // Initialize edge trigger interrupt for PA3 level

void EdgeCounter\_InitF(); // Initialize edge trigger interrupt for PF4+0 (SW1+2) both falling edge

void Switch\_InitA(void);

void Switch\_InitB(void);

void Switch\_InitF(void);

void SysTick\_Init(); // Initialize SysTick timer for 1 ms delay with interrupt enabled

void GPIOPortA\_Handler(); // Handle GPIO Port A interrupts

void GPIOPortF\_Handler(); // Handle GPIO Port F interrupts

void SysTick\_Handler(); // Handle SysTick generated interrupts

#define SENSOR\_PORTA (\*((volatile unsigned long \*)0x40004020)) // PA3

#define PORTB (\*((volatile unsigned long \*)0x400053FC)) // PB01234567

#define SENSOR\_PORTF (\*((volatile unsigned long \*)0x40025044)) // PF04

struct State {

uint32\_t Out; // output

uint32\_t Time; // 1 ms units

uint32\_t Next[8]; // list of next states

};

typedef const struct State STyp;

#define phase1 0

#define phase2 1

#define phase3 2

#define phase4 3

// Input == move, cw, ccw

STyp FSM [4] = {

// Since the left stepper is facing the opposite direction as the right stepper motor, they need to turn in opposite directions.

{0x33, 5, {phase1, phase1, phase1, phase1, phase1, phase4, phase2, phase1}}, // phase1

{0x96, 5, {phase2, phase2, phase2, phase2, phase2, phase1, phase3, phase2}}, // phase2

{0xCC, 5, {phase3, phase3, phase3, phase3, phase3, phase2, phase4, phase3}}, // phase3

{0x69, 5, {phase4, phase4, phase4, phase4, phase4, phase3, phase1, phase4}}, // phase4

};

uint8\_t S; // index of current state

uint8\_t Input; // input obtained from switches

uint32\_t stepsRemaining; // steps remaining

int main(void) {

SysTick\_Init(); // initialize SysTick timer

EdgeCounter\_InitA();

EdgeCounter\_InitF();

// Initialize GPIO on Ports A, B, F

// Port A Init

Switch\_InitA();

// Port B Init

Switch\_InitB();

// Port F Init

Switch\_InitF();

// Initial state

S = phase1;

while(1) {

WaitForInterrupt();

}

}

// Subroutine to wait about 0.1 sec

// Inputs: None

// Outputs: None

// Notes: the Keil simulation runs slower than the real board

#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 // Count flag

#define NVIC\_ST\_CTRL\_CLK\_SRC 0x00000004 // Clock Source

#define NVIC\_ST\_CTRL\_INTEN 0x00000002 // Interrupt enable

#define NVIC\_ST\_CTRL\_ENABLE 0x00000001 // Counter mode

#define NVIC\_ST\_RELOAD\_M 0x00FFFFFF // Counter load value

void Switch\_InitA(void){ // input

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x01; // 1) activate clock for Port A

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

// 2) no need to unlock GPIO Port A

GPIO\_PORTA\_DIR\_R &= ~0x08; // 3) input on PA3

GPIO\_PORTA\_AFSEL\_R &= ~0x08; // 4) PCTL GPIO on PA3

GPIO\_PORTA\_DEN\_R |= 0x08; // 5) enable PA3

}

void Switch\_InitB(void){ // output

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x02; // 1) activate clock for Port B

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

// 2) no need to unlock GPIO Port B

GPIO\_PORTB\_DIR\_R |= 0xFF; // 3) output PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7

GPIO\_PORTB\_AFSEL\_R &= ~0xFF; // 4) PCTL GPIO on PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7

GPIO\_PORTB\_DEN\_R |= 0xFF; // 5) enable PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7

}

void Switch\_InitF(void){ // input

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x20; // 1) activate clock for Port F

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock PortF (for PF0)

GPIO\_PORTF\_CR\_R |= 0x01; // allow changes to PF0 (PF4-1 unlocked by default)

GPIO\_PORTF\_DIR\_R &= ~0x11; // 3) input on PF4, PF0

GPIO\_PORTF\_AFSEL\_R &= ~0x11; // 4) PCTL GPIO on PF4, PF0

GPIO\_PORTF\_PUR\_R |= 0x11; // 4.5) Enable pull up resistor

GPIO\_PORTF\_DEN\_R |= 0x11; // 5) enable PF4, PF0

}

// Initialize SysTick timer for 1 ms delay with interrupt enabled

void SysTick\_Init() {

NVIC\_ST\_CTRL\_R = 0;

NVIC\_ST\_RELOAD\_R = 16000 - 1; // 1 s / (1 / 16 MHz)

NVIC\_ST\_CURRENT\_R = 0;

// priority 3

NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R & 0x00FFFFFF) | 0x60000000;

// enable SysTick with core clock and interrupts

NVIC\_ST\_CTRL\_R |= 0x07; // 0111

EnableInterrupts();

}

// Initialize level trigger interrupt for PA3

void EdgeCounter\_InitA() { // must be friendly

SYSCTL\_RCGC2\_R |= 0x01; // (a) actiate clock for port A

GPIO\_PORTA\_DIR\_R &= ~0x08; // (c) make PA3 input

GPIO\_PORTA\_AFSEL\_R &= ~0x08; // disable alt funct on PA3

GPIO\_PORTA\_DEN\_R |= 0x08; // enable digital I/O on PA3

GPIO\_PORTA\_PCTL\_R &= ~0x0000F000; // configure PF3 as GPIO

GPIO\_PORTA\_AMSEL\_R &= ~0x08; // disable analog functionality on PA3

GPIO\_PORTA\_IS\_R |= 0x08; // (d) PA3 is level-sensitive

GPIO\_PORTA\_IBE\_R &= ~0x08; // PA3 is not both edges

GPIO\_PORTA\_IEV\_R &= ~0x08; // PA3 level low

GPIO\_PORTA\_ICR\_R = 0x08; // (e) clear flag3

GPIO\_PORTA\_IM\_R |= 0x08; // (f) arm interrupt on PA3

NVIC\_PRI0\_R = (NVIC\_PRI0\_R & 0xFFFFFF00) | 0x00000020; // (g) priority 1

NVIC\_EN0\_R = 0x00000001; // (h) enable interrupt 0 in NVIC

EnableInterrupts(); // (i) Clears the I bit

}

// Initialize edge trigger interrupt for PF4, PF0 (SW1+2) both edges

void EdgeCounter\_InitF() { // must be friendly

SYSCTL\_RCGC2\_R |= 0x20; // (a) actiate clock for port F

GPIO\_PORTF\_DIR\_R &= ~0x11; // (c) make PF4, PF0 in (built-in button)

GPIO\_PORTF\_AFSEL\_R &= ~0x11; // disable alt funct on PF4, PF0

GPIO\_PORTF\_DEN\_R |= 0x11; // enable digital I/O on PF4, 0

GPIO\_PORTF\_PCTL\_R &= ~0x000F000F; // configure PF4, 0 as GPIO

GPIO\_PORTF\_AMSEL\_R &= ~0x11; // disable analog functionality on PF4, 0

GPIO\_PORTF\_PUR\_R |= 0x11; // enable weak pull-up on PF4, 0

GPIO\_PORTF\_IS\_R &= ~0x11; // (d) PF4, 0 is edge-sensitive

GPIO\_PORTF\_IBE\_R &= ~0x11; // PF4, 0 is NOT both edges

GPIO\_PORTF\_IEV\_R &= ~0x11; // PF4, 0 falling edge event

GPIO\_PORTF\_ICR\_R = 0x11; // (e) clear flag0, 4

GPIO\_PORTF\_IM\_R |= 0x11; // (f) arm interrupt on PF0, 4

NVIC\_PRI7\_R = (NVIC\_PRI7\_R & 0xFF00FFFF) | 0x00400000; // (g) priority 2

NVIC\_EN0\_R = 0x40000000; // (h) enable interrupt 30 in NVIC

EnableInterrupts(); // (i) Clears the I bit

}

// Handle GPIO Port A interrupts. When Port A interrupt triggers, reset

void GPIOPortA\_Handler() {

GPIO\_PORTA\_ICR\_R = 0x08; // Clear flag for port A pin 3.

Input &= ~0x04; // move disable

}

// Handle GPIO Port F interrupts. When Port F interrupt triggers, go to next state

void GPIOPortF\_Handler() {

GPIO\_PORTF\_ICR\_R = 0x11; // Clear flag for port F pins 0 and 4.

Input |= 0x04; // move enable

if ((~SENSOR\_PORTF & 0x01) == 0x01) { // PF0 (SW2) pressed

Input &= ~0x01;

Input |= 0x02;

stepsRemaining = 8 \* 64; // 64:1 gear ratio

}

else if (((~SENSOR\_PORTF & 0x10 ) >> 3) == 0x02) { // PF4 (SW1) pressed

Input &= ~0x02;

Input |= 0x01;

stepsRemaining = 143 \* 64; // 64:1 gear ratio

}

else {

Input &= ~0x03;

stepsRemaining = 0;

}

}

// Handle SysTick generated interrupts. When timer interrupt triggers, do what's necessary

void SysTick\_Handler() {

if (stepsRemaining > 0) {

S = FSM[S].Next[Input];

PORTB = FSM[S].Out; // set stepper motors on port B

--stepsRemaining;

}

// Reload timer with new value

NVIC\_ST\_CTRL\_R = 0; // stop timer

NVIC\_ST\_RELOAD\_R = (16000 \* FSM[S].Time) - 1; // Wait FSM[S].Time milliseconds (16000 ticks = 1 ms)

NVIC\_ST\_CURRENT\_R = 0; // delete count value

NVIC\_ST\_CTRL\_R |= 0x07; // 0111, start timer

}