Alarik Damrow, Oliver Pi

Professor Gevik Sardarbegians

CECS 346 Sec1 (7147)

8 December 2022

Final Project Report: Smart Car

This project was difficult to build do to the many components and needs of the car. I went through two boards during the creation of the lab due to poor building of my battery connection to the board that I did not use working capacitors. In my design port B will make the motors work to rotate forward or reverse. Port F is the inputs given by the user of the car and Port A is set to interrupt when the sensor in front of the car goes off due to detecting something in front of it.

GPIO Port and Pins Table

|  |  |
| --- | --- |
| Description | Pin |
| Forward Button | PF0 |
| Reverse Button | PF4 |
| IR Sensor | PA2 |
| 1 Of 4 Output for Motor 1 | PB0 |
| 2 Of 4 Output for Motor 1 | PB1 |
| 3 Of 4 Output for Motor 1 | PB2 |
| 4 Of 4 Output for Motor 1 | PB3 |
| 1 Of 4 Output for Motor 2 | PB4 |
| 2 Of 4 Output for Motor 2 | PB5 |
| 3 Of 4 Output for Motor 2 | PB6 |
| 4 Of 4 Output for Motor 2 | PB7 |

Input Description of Ports and Pins

|  |  |
| --- | --- |
| Pin | Purpose |
| PF0 | This is the forward button that will set the car to move forward. |
| PF4 | This is the reverse button that will set the car to move back |
| PA2 | This is the sensor that will detect if something was in front of the car and the car has to stop. |

Output Description of Ports and Pins

|  |  |
| --- | --- |
| Pin | Purpose |
| PB0 | This is one of four output pins for the right stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB1 | This is one of four output pins for the right stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB2 | This is one of four output pins for the right stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB3 | This is one of four output pins for the right stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB4 | This is one of four output pins for the left stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB5 | This is one of four output pins for the left stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB6 | This is one of four output pins for the left stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |
| PB7 | This is one of four output pins for the left stepper motor. Depending on the current state in the FSM this pin will be active or not. When active it will send current and voltage to a part of the stepper motor to move forward or back depending on input given. |

State Table of FSM

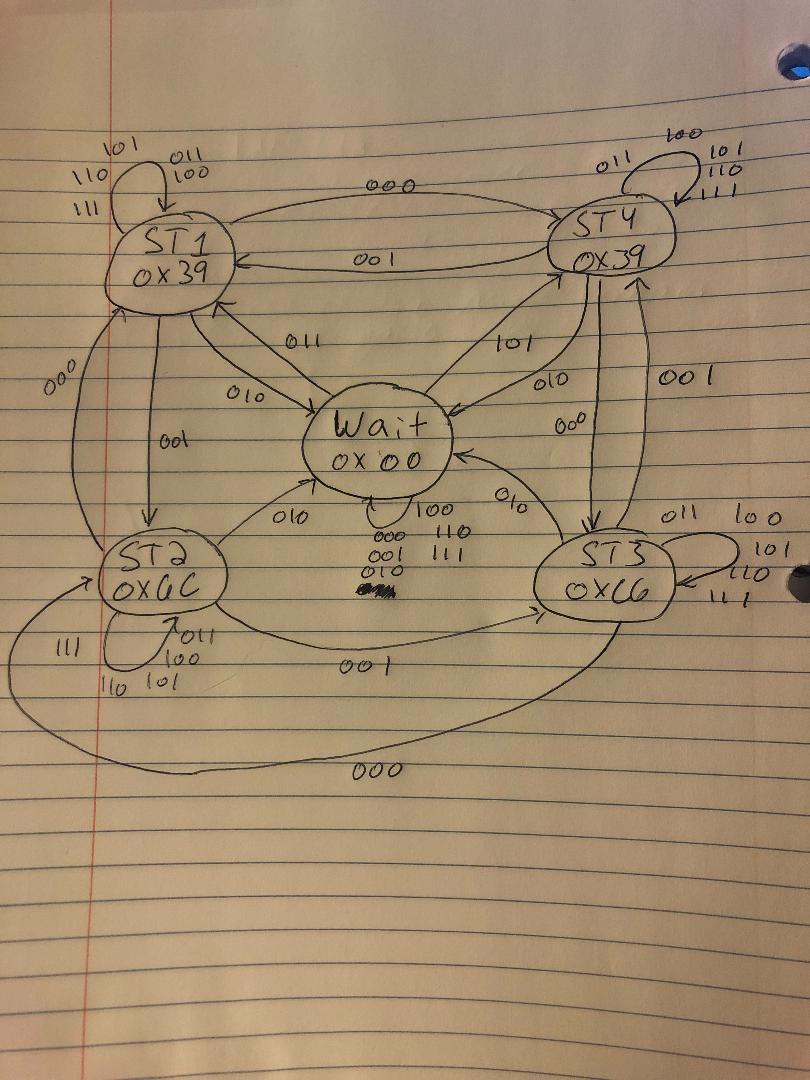
States and What is Done

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Name | Output | PB0-3 | PB4-7 |
| 0 | Wait | 0x00 | No Stage | No Stage |
| 1 | St1 | 0x39 | Stage 1 of motor | Stage 3 of motor |
| 2 | St2 | 0x6C | Stage 2 of motor | Stage 4 of motor |
| 3 | St3 | 0xC6 | Stage 4 of motor | Stage 2 of motor |
| 4 | St4 | 0x93 | Stage 3 of motor | Stage 1 of motor |

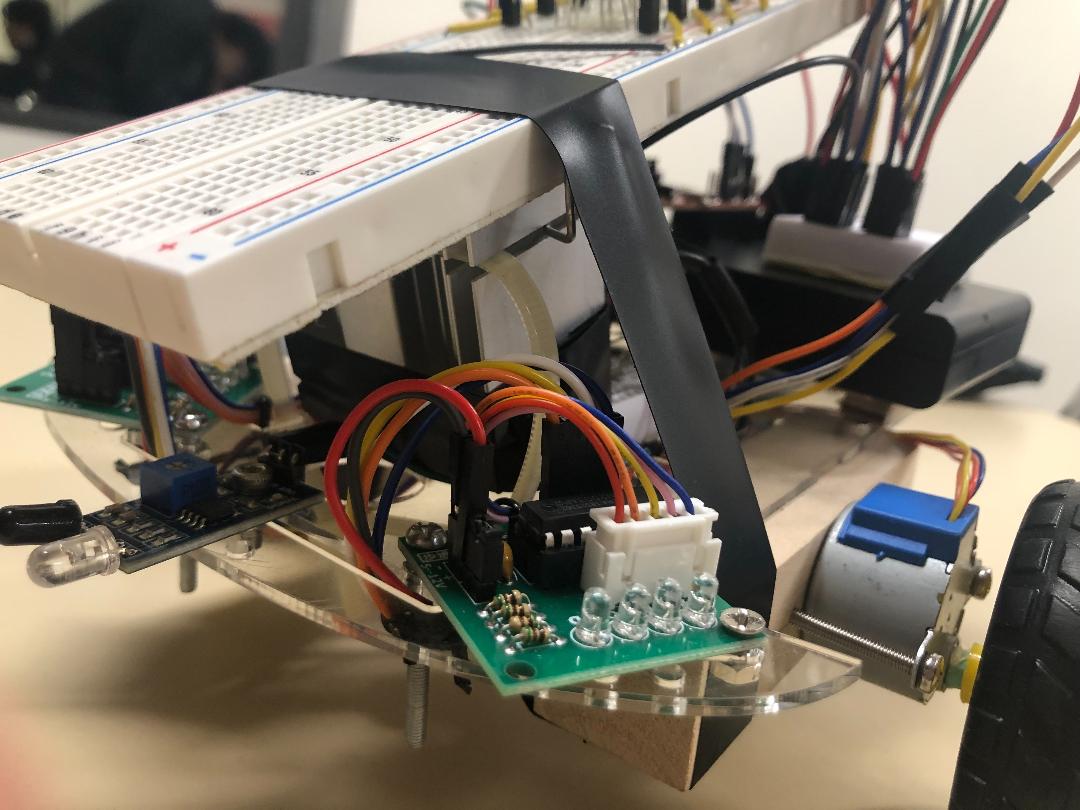
State and the inputs

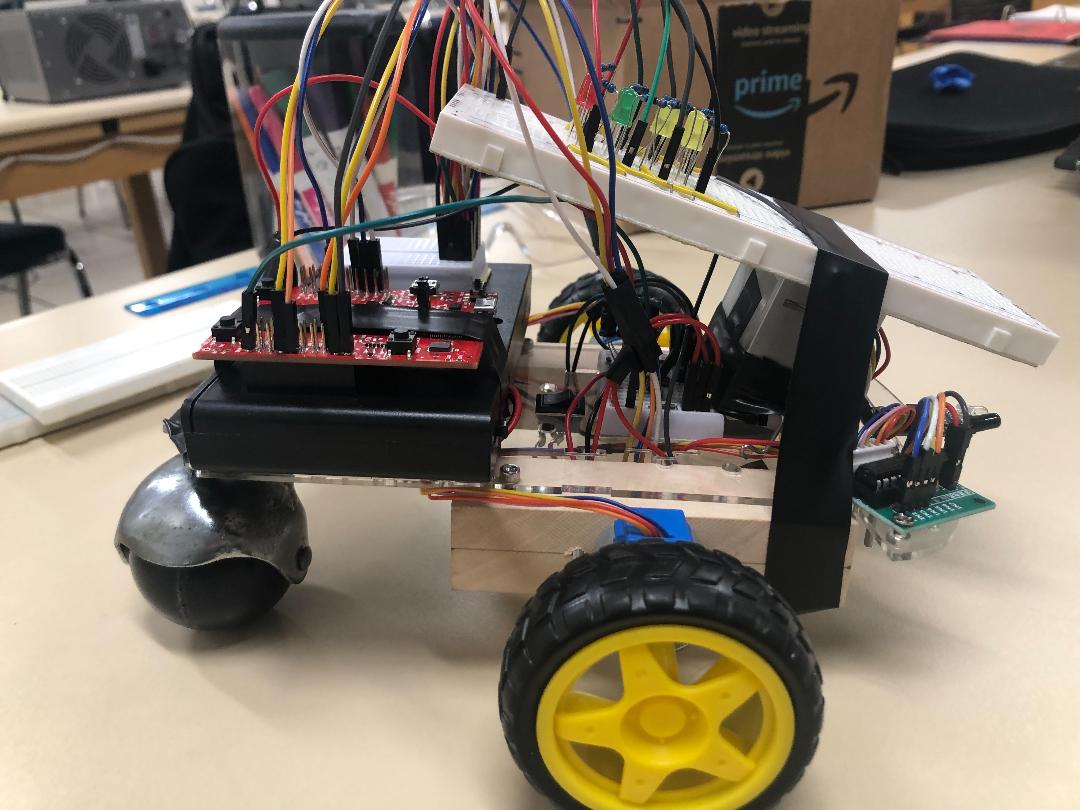
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Number | Name | Input = 0 | Input = 1 | Input = 2 | Input = 3 | Input = 4 | Input = 5 | Input = 6 | Input = 7 |
| 0 | Wait | Wait | Wait | Wait | St1 | Wait | St4 | Wait | Wait |
| 1 | St1 | St4 | St2 | Wait | St1 | St1 | St1 | St1 | St1 |
| 2 | St2 | St1 | St3 | Wait | St2 | St2 | St2 | St2 | St2 |
| 3 | St3 | St2 | St4 | Wait | St3 | St3 | St3 | St3 | St3 |
| 4 | St4 | St3 | St1 | Wait | St4 | St4 | St4 | St4 | St4 |

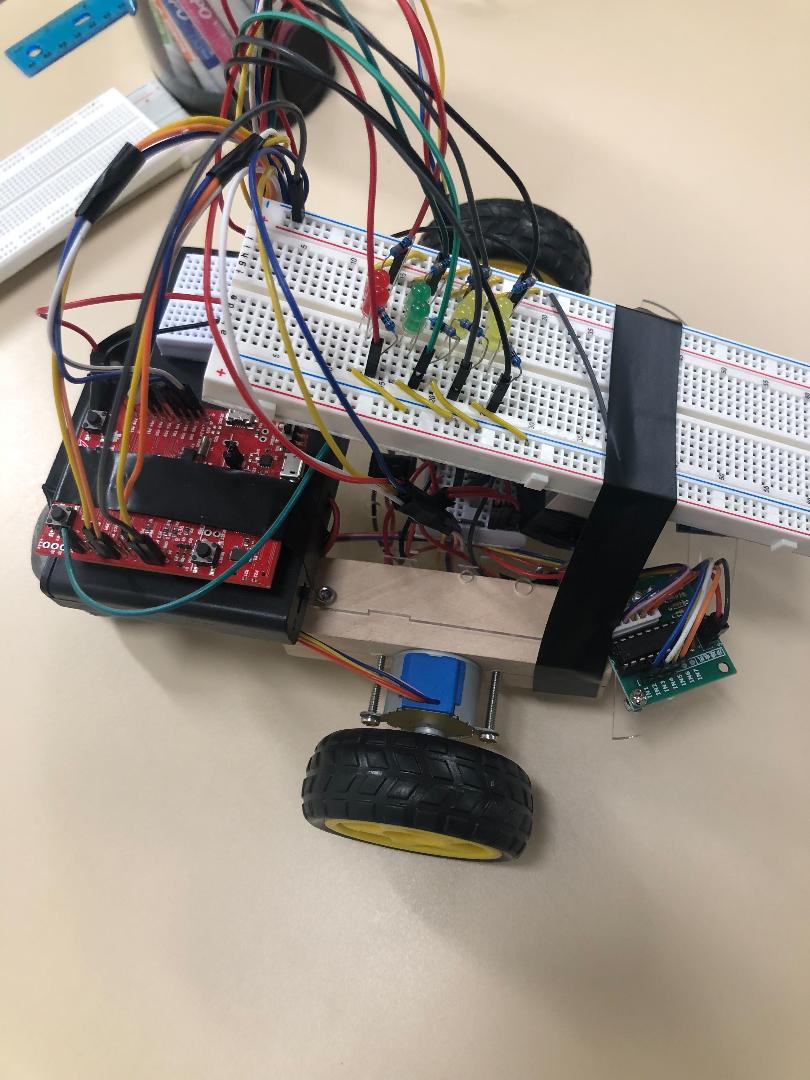
FSM Diagram



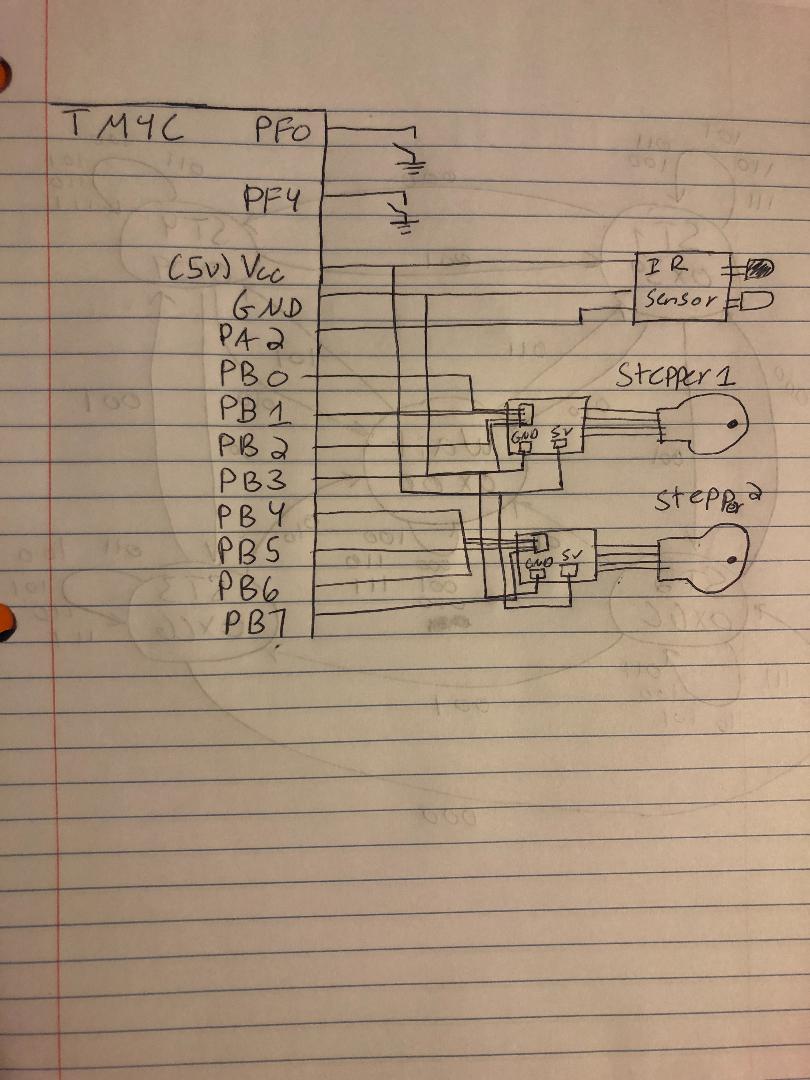
Photographs of Hardware







Hardware Diagram



Source Code

#include "tm4c123gh6pm.h"

#include <stdint.h>

//Port B

#define GPIO\_PORTB\_OUT (\*((volatile unsigned long \*)0x400053FC)) //bits 7-0

#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))

//Port A

#define GPIO\_PORTA\_IN (\*((volatile unsigned long \*)0x400043FC)) //bits 3-2

#define GPIO\_PORTA\_DATA\_R (\*((volatile unsigned long \*)0x400043FC))

#define GPIO\_PORTA\_DIR\_R (\*((volatile unsigned long \*)0x40004400))

#define GPIO\_PORTA\_AFSEL\_R (\*((volatile unsigned long \*)0x40004420))

#define GPIO\_PORTA\_PUR\_R (\*((volatile unsigned long \*)0x40004510))

#define GPIO\_PORTA\_DEN\_R (\*((volatile unsigned long \*)0x4000451C))

#define GPIO\_PORTA\_LOCK\_R (\*((volatile unsigned long \*)0x40004520))

#define GPIO\_PORTA\_CR\_R (\*((volatile unsigned long \*)0x40004524))

#define GPIO\_PORTA\_AMSEL\_R (\*((volatile unsigned long \*)0x40004528))

#define GPIO\_PORTA\_PCTL\_R (\*((volatile unsigned long \*)0x4000452C))

#define GPIO\_PORTA\_ODR\_R (\*((volatile unsigned long \*)0x4000450C))

#define GPIO\_PORTA\_IS\_R (\*((volatile unsigned long \*)0x40004404))

#define GPIO\_PORTA\_IBE\_R (\*((volatile unsigned long \*)0x40004408))

#define GPIO\_PORTA\_ICR\_R (\*((volatile unsigned long \*)0x4000441C))

#define GPIO\_PORTA\_IM\_R (\*((volatile unsigned long \*)0x40004410))

#define NVIC\_PRI0\_R (\*((volatile unsigned long \*)0xE000E400))

//Port F

#define GPIO\_PORTF\_DATA\_R (\*((volatile unsigned long \*)0x400253FC))

#define GPIO\_PORTF\_DIR\_R (\*((volatile unsigned long \*)0x40025400))

#define GPIO\_PORTF\_AFSEL\_R (\*((volatile unsigned long \*)0x40025420))

#define GPIO\_PORTF\_PUR\_R (\*((volatile unsigned long \*)0x40025510))

#define GPIO\_PORTF\_DEN\_R (\*((volatile unsigned long \*)0x4002551C))

#define GPIO\_PORTF\_LOCK\_R (\*((volatile unsigned long \*)0x40025520))

#define GPIO\_PORTF\_CR\_R (\*((volatile unsigned long \*)0x40025524))

#define GPIO\_PORTF\_AMSEL\_R (\*((volatile unsigned long \*)0x40025528))

#define GPIO\_PORTF\_PCTL\_R (\*((volatile unsigned long \*)0x4002552C))

#define GPIO\_LOCK\_KEY 0x4C4F434B

//SysTick Interrupts

#define NVIC\_SYS\_PRI3\_R (\*((volatile unsigned long \*)0xE000ED20))

#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

#define NVIC\_EN0\_R (\*((volatile unsigned long \*)0xE000E100))

#define SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

#define IRSENSOR (\*((volatile unsigned long \*)0x40004010))

#define RESET (\*((volatile unsigned long \*)0x40025004))

#define Motor (\*((volatile unsigned long \*)0x400053FC))

#define SYSCTL\_RCGC2\_GPIOB 0x00000002 //port B Clock Gating Control

#define SYSCTL\_RCGC2\_GPIOA 0x00000001 //port A Clock Gating Control

#define PD0 (\*((volatile unsigned long \*)0x40007004))

#define PD1 (\*((volatile unsigned long \*)0x40007008))

#define PD2 (\*((volatile unsigned long \*)0x40007010))

#define Stepper (\*((volatile unsigned long \*)0x4002401C))

#define PF0 (\*((volatile unsigned long \*)0x40025004))

#define PF4 (\*((volatile unsigned long \*)0x40025008))

void DisableInterrupts(void); // Disable interrupts

void EnableInterrupts(void); // Enable interrupts

long StartCritical (void); // previous I bit, disable interrupts

void EndCritical(long sr); // restore I bit to previous value

void WaitForInterrupt(void); // low power mode

void Ports\_Init(void);

void SysTick\_Init(void);

void SysTick\_Handler (void);

void GPIOPortA\_Handler(void);

void Delay(void);

void SysTick\_Wait10ms(int delay);

struct State {

unsigned long Out;

unsigned long Time;

unsigned long Next[8];};

typedef const struct State STyp;

#define Wait 0

#define St1 1

#define St2 2

#define St3 3

#define St4 4

STyp FSM[5]={

{0x00,0x80000,{Wait,Wait,Wait,St1,Wait,St4,Wait,Wait}},

{0x39,0x80000,{St4,St2,Wait,St1,St1,St1,St1,St1}},

{0x6C,0x80000,{St1,St3,Wait,St2,St2,St2,St2,St2}},

{0xC6,0x80000,{St2,St4,Wait,St3,St3,St3,St3,St3}},

{0x93,0x80000,{St3,St1,Wait,St4,St4,St4,St4,St4}},

};

unsigned long i; //state index for iteration

unsigned long Input;

unsigned long In;

unsigned long Rep;

void EdgeCounter\_Init(void){

DisableInterrupts();

//Port A

GPIO\_PORTA\_IS\_R &= ~0x0C; // (d) edge-sensitive

GPIO\_PORTA\_IBE\_R &= 0x0C;

GPIO\_PORTA\_ICR\_R = 0x0C; // (e) clear flag

GPIO\_PORTA\_IM\_R |= 0x0C; // (f) arm interrupt on PA2-3

NVIC\_PRI1\_R = (NVIC\_PRI7\_R&0xFF00FFFF)|0x00400000; // (g) priority 2

NVIC\_EN0\_R |= 0x00000011;

EnableInterrupts();

}

int main(void){

Ports\_Init();

EdgeCounter\_Init();

SysTick\_Init();

i = Wait;

Rep = 0;

//GPIO\_PORTF\_DATA\_R = 0x00;

while(1){

Motor = FSM[i].Out;

Delay();

//In = 3;

//In = Stepper;

//In = (4\*PD2)+(2\*PD1)+PD0;

if (GPIO\_PORTF\_DATA\_R == 0x01)

{

In = 3;

Rep = (2000)\*4.5;

}

else if (GPIO\_PORTF\_DATA\_R == 0x10)

{

In = 5;

Rep = 800;

}

i = FSM[i].Next[In];

if (In == 3)

{

while(In == 3)

{

while (Rep != 0)

{

Motor = FSM[i].Out;

Delay();

i = FSM[i].Next[1];

Rep --;

}

In = 2;

i = FSM[i].Next[In];

Motor = FSM[i].Out;

Delay();

}

}

else if (In == 5)

{

while(In == 5)

{

while (Rep != 0)

{

Motor = FSM[i].Out;

Delay();

i = FSM[i].Next[0];

Rep --;

}

In = 2;

i = FSM[i].Next[In];

Motor = FSM[i].Out;

Delay();

}

}

//WaitForInterrupt();

}

}

void Ports\_Init(void){

volatile unsigned long delay;

SYSCTL\_RCGC2\_R = 0x3B; // 1) Ports A B D

delay = SYSCTL\_RCGC2\_R; // 2) no need to unlock

//Port A

GPIO\_PORTA\_DATA\_R = 0x400043FC;

GPIO\_PORTA\_CR\_R = 0x1C;

GPIO\_PORTA\_AMSEL\_R &= 0x00; // 3) disable analog function on PA3-2

GPIO\_PORTA\_PCTL\_R &= 0x00000000; // 4) enable regular GPIO

GPIO\_PORTA\_DIR\_R &= 0x00; // 5) inputs on PA3-2

GPIO\_PORTA\_AFSEL\_R &= 0x00; // 6) regular function on PA3-2

GPIO\_PORTA\_DEN\_R |= 0x0C; // 7) enable digital on PA3-2

GPIO\_PORTA\_ODR\_R |= 0x38;

//Port F

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // allow changes to PF4-0

GPIO\_PORTF\_AMSEL\_R = 0x00; // 3) disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // 4) PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // 5) PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // 6) disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // 7) enable digital I/O on PF4-0

//Port B

GPIO\_PORTB\_AMSEL\_R &= ~0xFF;

GPIO\_PORTB\_PCTL\_R &= ~0x00FFFFFF; // 4) enable regular GPIO

GPIO\_PORTB\_DIR\_R |= 0xFF; // 5) outputs on PB5-0

GPIO\_PORTB\_AFSEL\_R &= ~0xFF; // 6) regular function on PB5-0

GPIO\_PORTB\_DEN\_R |= 0xFF; // 7) enable digital on PB5-0

}

void SysTick\_Handler()

{

In = 2;

i = FSM[i].Next[In];

Motor = FSM[i].Out;

Delay();

}

void GPIOPortA\_Handler()

{

In = 2;

i = FSM[i].Next[In];

Motor = FSM[i].Out;

Delay();

}

void Delay(){

volatile uint32\_t time;\

time = 727240\*200/(91\*200\*3);

while(time){

time--;

}

}

void SysTick\_Init(void){

NVIC\_ST\_CTRL\_R = 0; // disable SysTick during setup

NVIC\_ST\_RELOAD\_R = NVIC\_ST\_RELOAD\_M; // maximum reload value

NVIC\_ST\_CURRENT\_R = 0; // any write to current clears it

NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R&0x00FFFFFF) | 0x60000000; //priority 3

NVIC\_ST\_CTRL\_R = NVIC\_ST\_CTRL\_ENABLE+NVIC\_ST\_CTRL\_CLK\_SRC;

}

// Time delay using busy wait.

// The delay parameter is in units of the core clock. (units of 20 nsec for 50 MHz clock)

void SysTick\_Wait(unsigned long delay){

volatile unsigned long elapsedTime;

unsigned long startTime = NVIC\_ST\_CURRENT\_R;

do{

elapsedTime = (startTime-NVIC\_ST\_CURRENT\_R)&0x00FFFFFF;

}

while(elapsedTime <= delay);

}

// Time delay using busy wait.

// This assumes 50 MHz system clock.

void SysTick\_Wait10ms(int delay){

unsigned long i;

for(i=0; i<delay; i++){

SysTick\_Wait(500000); // wait 10ms (assumes 50 MHz clock)

}

}