****

**Main**

/\* ========================================

Quinton Cline

Motor Control

\* ========================================

\*/

#include "project.h"

#include <stdio.h>

///////////////////////

//// DC Motor ///////

//////////////////////

//DC Motor Constants

#define motorReverse 0b00000100

#define motorForward 0b00001000

//DC motor Variables

int ADC=0;

int BDC=1;

int direction=0; //Stors the current diretion

int reverseDirection=0; //

void dcMotorControll()

{

//Changes the direction forward ->reverse

if (reverseDirection == 0){

direction = motorReverse;

//Writes the bytes to the

SPIM\_WriteTxData(motorReverse);

//Change the variable for direction

reverseDirection = 1;

}

//Changes reverse->forward

else{

direction = motorForward;

//Writes the bytes to

SPIM\_WriteTxData(motorForward);

//Change the variable for direction

reverseDirection = 0;

}

//Loop tell transmit is done

while(!(SPIM\_ReadTxStatus() & SPIM\_STS\_BYTE\_COMPLETE)){}

Latch\_Write(0); //rising edge to the transfer latch

Latch\_Write(1); //Reset the latch

Enable\_Write(0); //write low to enable

}

//////////////////////

//// STEPPER ////

////////////////////

//Stepper Motor wire constants

#define Blue 0b10000000

#define Pink 0b00000001

#define Yellow 0b00100000

#define Orange 0b01000000

//The stepper order

const int steps[4] = {Blue,Pink,Yellow,Orange};

//Variable for the current step

int step=0;

//This is the function that is called by the intterupt when the button is pushed

void stepperMotorControl(){

//Set enable to high

Enable\_Write(1);

//Send step to SPI to be written to the motor

SPIM\_WriteTxData(steps[step] | direction);

//Remain in while loop until the data is done being transmitted

while(!(SPIM\_ReadTxStatus() & SPIM\_STS\_BYTE\_COMPLETE)){}

//rising edge to the transfer latch

Latch\_Write(0);

Latch\_Write(1);

Enable\_Write(0); //write low to enable

}

int main(void)

{

//Variables

int capPos=0;

CyGlobalIntEnable; /\* Enable global interrupts. \*/

//Start stuff up

CapSense\_Start();

LED\_PWM\_Start();

DC\_PWM\_Start();

SERVO\_PWM\_Start();

SPIM\_Start();

DirectionInt\_Start();

Stepper\_Timer\_Start();

StepperInterrupt\_Start();

StepperInterrupt\_ClearPending();

CapSense\_InitializeAllBaselines();

SERVO\_PWM\_WriteCompare(700); // Start the servo on the left

while(1)

{

//Checks to see if it is busy

if(CapSense\_NOT\_BUSY==CapSense\_IsBusy())

{

//scan all widgets

CapSense\_ScanAllWidgets();

CapSense\_ProcessAllWidgets();

//If it is touched go into if statement

if(CapSense\_GetCentroidPos(CapSense\_LINEARSLIDER0\_WDGT\_ID) != CapSense\_SLIDER\_NO\_TOUCH )

{

//Grab the calue

capPos=CapSense\_GetCentroidPos(CapSense\_LINEARSLIDER0\_WDGT\_ID);

//Set LED PWM

LED\_PWM\_WriteCompare(capPos );

//Set DC PWM

DC\_PWM\_WriteCompare(capPos);

//Re-map servo and write servo pwm

int servoPos = ((1600\*(capPos))/(100)) + 700;

SERVO\_PWM\_WriteCompare(servoPos );

//Re-map stepper and write to stepper timer

int stepperScaled=12000-(capPos\*100);

Stepper\_Timer\_WritePeriod(stepperScaled);

}

else

{

capPos=0;

}

//Setup scan for next loop

CapSense\_ScanAllWidgets();

}

}

}

**DC Interupt**

**Header**

#include "project.h"

extern void dcMotorControll();

**Code**

//Call the function

dcMotorControll();

//Delay for a bit

CyDelay(10);

//Clear pending

DirectionInt\_ClearPending();

**Stepper Interupt**

**Header**

#include "project.h"

extern void stepperMotorControl();

extern int step;

extern int steps[];

**Code**

//Call stepper motor function

stepperMotorControl();

//Increment step

step++;

//If at the end of steps array go back to the beginning

if(step>3)

{

step=0;

}

//Clear pending

Stepper\_Timer\_ClearInterrupt(Stepper\_Timer\_INTR\_MASK\_TC);