/\*

\* File: main.c

\* Author: TrudieC

\*

\* Created on March 8, 2019, 12:39 AM

\*/

/\*March 10 12:10am keeps looping on main, but on the second time the stepper gets funky and the signals to the port are very slow, st the stepper itself isnt even moving

also the servos are twitching

March 10 8:14am -- right servo won't even move anymore?? the \_\_delay\_ms function seems to be delaying the program more than it should. running the servos seems to cause the steppers to stop working properly.

\*/

/\*

#include <xc.h>

#include <stdio.h>

#include <stdbool.h>

#include "configBits.h"

#include "lcd.h"

#include "RTC.h"

#include "I2C.h"

#include <stdio.h>

#include <math.h>

#define MINTHR 8000

#define RESOLUTION 488

#define InternalOsc\_8MHz 8000000

#define InternalOsc\_4MHz 4000000

#define InternalOsc\_2MHz 2000000

#define InternalOsc\_1MHz 1000000

#define InternalOsc\_500KHz 500000

#define InternalOsc\_250KHz 250000

#define InternalOsc\_125KHz 125000

#define InternalOsc\_31KHz 31000

#define Timer2Prescale\_1 1

#define Timer2Prescale\_4 4

#define Timer2Prescale\_16 16

//define constants

#define speed1 10 // Speed Range 10 to 1 10 = lowest , 1 = highest

#define speed2 6 // Speed Range 10 to 1 10 = lowest , 1 = highest

//1 is when it stops turning, 4 is when lights stop flickering, 5 10 is ok, 20 25 30 is finnicky and slow

#define steps1 14// how much step it will take (249 ~= 1 full rotation)

#define steps2 110 // how much step it will take (249 ~= 1 full rotation)

#define steps3 156 // how much step it will take (249 ~= 1 full rotation)

#define clockwise 0 // clockwise direction macro

#define anti\_clockwise 1 // anti clockwise direction macro

int act\_cnt = 0;

void system\_init (void); // This function will initialise the ports.

void full\_drive (char direction,int stepper\_no); // This function will drive the motor in full drive mode

void stepper(void);

void servo(void);

void PWM\_End();

//void \_\_delay\_ms(unsigned int val);

//void delay(unsigned int val);

\*/

#include "actuators.h"

void actuators\_main(int stack){

int act\_cnt = 0;

system\_init();

initLCD();

lcd\_clear();

printf("cnt is %d",act\_cnt);

\_\_delay\_ms(200);

while (1){

if (act\_cnt==0){

// lcd\_clear();

// printf("Stepper start");

stepper(stack);

stepper2(clockwise);

// \_\_delay\_ms(2000);

// lcd\_clear();

// printf("Stepper done");

servo();

// lcd\_clear();

// printf("Servo done");

// system\_init();

// PWM\_End();

// \_\_delay\_ms(200);

stepper2(anti\_clockwise);

act\_cnt++;

}

// printf("fk");

if(act\_cnt){ //this will prevent the main to loop over itself, because the counter doesnt do anything .-.

break;

}

// delay(2000);

// stepper();

}

}

void system\_init (void){

TRISB = 0x00; // PORT B as output port

//PORTB = 0x0F;

LATB = 0x0F;

TRISA = 0x00;

LATA = 0x0F;

TRISCbits.TRISC1 = 0; /\* Set CCP2 pin as output for PWM out \*/

// CCP2CON = 0x0C;

TRISCbits.TRISC2 = 0; /\* Set CCP1 pin as output for PWM out \*/

// CCP1CON = 0x0C; /\* Set PWM mode \*/

LATD = 0x00;

TRISD = 0x00; //set data direction as output

// Set all A/D ports to digital (pg. 222)

ADCON1 = 0b00001111;

TRISC = 0x00;

LATC = 0x00;

TRISE = 0x00;

LATE = 0x00;

}

//void PWM\_End(){

// TRISCbits.TRISC2 = 0; /\* Set CCP1 pin as output for PWM out \*/

// CCP1CON = 0x00; /\* Set PWM mode \*/

// TRISCbits.TRISC1 = 0; /\* Set CCP1 pin as output for PWM out \*/

// CCP2CON = 0x00;

// OSCCON = 01111000; //OSCCON 0b00110010

//}

//

//int setPeriodTo(unsigned long FPWM){

// int clockSelectBits, TimerPrescaleBits;

// int TimerPrescaleValue;

// float period;

// unsigned long FOSC, \_resolution = RESOLUTION;

//

// if (FPWM < MINTHR) {TimerPrescaleBits = 2; TimerPrescaleValue = Timer2Prescale\_16;}

// else {TimerPrescaleBits = 0; TimerPrescaleValue = Timer2Prescale\_1;}

//

// if (FPWM > \_resolution) {clockSelectBits = 7; FOSC = InternalOsc\_8MHz;}

// else if (FPWM > (\_resolution >>= 1)) {clockSelectBits = 6; FOSC = InternalOsc\_4MHz;}

// else if (FPWM > (\_resolution >>= 1)) {clockSelectBits = 5; FOSC = InternalOsc\_2MHz;}

// else if (FPWM > (\_resolution >>= 1)) {clockSelectBits = 4; FOSC = InternalOsc\_1MHz;}

// else if (FPWM > (\_resolution >>= 1)) {clockSelectBits = 3; FOSC = InternalOsc\_500KHz;}

// else if (FPWM > (\_resolution >>= 1)) {clockSelectBits = 2; FOSC = InternalOsc\_250KHz;}

// else if (FPWM > (\_resolution >>= 1)) {clockSelectBits = 1; FOSC = InternalOsc\_125KHz;}

// else {clockSelectBits = 0; FOSC = InternalOsc\_31KHz;}

//

// period = ((float)FOSC / (4.0 \* (float)TimerPrescaleValue \* (float)FPWM)) - 1.0;

// period = round(period);

//

// OSCCON = ((clockSelectBits & 0x07) << 4) | 0x02;

// PR2 = (int)period;

// T2CON = TimerPrescaleBits;

// TMR2 = 0;

// T2CONbits.TMR2ON = 1; /\* Turn ON Timer2 \*/

// return (int)period;

//}

//

//void SetDutyCycleTo(float Duty\_cycle, int Period, int serv){

// int PWM10BitValue;

//

// PWM10BitValue = 4.0 \* ((float)Period + 1.0) \* (Duty\_cycle/100.0);

// if (serv==1){

// CCPR1L = (PWM10BitValue >> 2);

// CCP1CON = ((PWM10BitValue & 0x03) << 4) | 0x0C;

// }

// if (serv==2){

// CCPR2L = (PWM10BitValue >> 2);

// CCP2CON = ((PWM10BitValue & 0x03) << 4) | 0x0C;

// }

//// LATAbits.LATA0 = PORTCbits.RC2;

//}

//

//void rotateneg(){

// int Period;

// system\_init();

// Period = setPeriodTo(50); /\* 50Hz PWM frequency \*/

//

// SetDutyCycleTo(3.0, Period,1);

// SetDutyCycleTo(12.0, Period,2);

//// return;

//}

//void rotatepos(){

// int Period;

// system\_init();

// Period = setPeriodTo(50); /\* 50Hz PWM frequency \*/

//

// SetDutyCycleTo(8.0, Period,1); //neutral

// SetDutyCycleTo(7.0, Period,2); //neutral

//}

//

//void servo() {

// int count = 0;

//// while (1){

// if (count<1){

// rotatepos();

// \_\_delay\_ms(100);

// rotateneg();

// \_\_delay\_ms(100);

// rotatepos();

// count++;

// }

// PWM\_End();

//// }

// return;

//}

//

//void servo(void) {

// int count\_servo = 0;

// while (1){

// if (count\_servo<1){

// rotateneg();

// \_\_delay\_ms(200);

// rotatepos();

// \_\_delay\_ms(200);

// rotateneg();

// \_\_delay\_ms(200);

// }count\_servo++;

// else{

// break;

// }

// }

// lcd\_clear();

// printf("Servo loop");

//// \_\_delay\_ms(2000);

// PWM\_End();

// LATCbits.LATC2 = 0;

// return;

//}

void servoRotate0() //0 Degree

{

unsigned int i;

// if (servo==1){

for(i=0;i<40;i++)

{

RC1 = 1;

RC2 = 1;

\_\_delay\_us(1400);

RC1 = 0;

\_\_delay\_us(1150);

RC2 = 0;

\_\_delay\_us(18600);

}

// }

// if (servo==2){

// for(i=0;i<40;i++)

// {

// RC2 = 1;

// \_\_delay\_us(449); //goes down to go lower

// RC2 = 0;

// \_\_delay\_us(27300); //goes up to go lower

// }

// }

}

void servoRotate180() //180 Degree

{

unsigned int i;

// if (servo==1){

for(i=0;i<40;i++)

{

RC1 = 1;

RC2 = 1;

\_\_delay\_us(450); //goes down to go lower

RC2 =0;

\_\_delay\_us(2100);

RC1 = 0;

\_\_delay\_us(13550);

}

// }

// if (servo==2){

// for(i=0;i<40;i++)

// {

// RC2 = 1;

// \_\_delay\_us(2200);

// RC2 = 0;

// \_\_delay\_us(17800);

// }

// }

}

void servo()

{

// TRISC = 0; // PORTB as Ouput Port

{

// servoRotate0(1); //0 Degree

servoRotate0(); //0 Degree

// \_\_delay\_ms(2000);

// servoRotate90(); //90 Degree

// servoRotate180(1); //180 Degree

servoRotate180(); //180 Degree

\_\_delay\_ms(500);

servoRotate0(); //0 Degree

}

}

void stepper(int stack){

int count\_stepper = 0;

// while (1){

if (count\_stepper<1){

lcd\_clear();

// printf("step1");

// lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

// printf("Step speed %d",speed);

if (stack == 1){

for(int i=0;i<steps1;i++){

// full\_drive(clockwise, 2); //ramp stepper , fwd

full\_drive(clockwise, 1); //right stepper forwards A's

// full\_drive(anti\_clockwise, 3); //left stepper forwards

}

}

if (stack == 2){

for(int i=0;i<steps1;i++){

full\_drive(clockwise, 3); //ramp stepper , fwd

// full\_drive(clockwise, 1); //right stepper forwards A's

// full\_drive(anti\_clockwise, 3); //left stepper forwards

}

}

// printf("cnt is %d", count\_stepper);

count\_stepper++;

}

// if ((count\_stepper>=1)&&(count\_stepper<3)){

// lcd\_clear();

// printf("step2");

// lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

// for(int i=0;i<steps2;i++){

// full\_drive(clockwise, 2); //forwards

// }

// printf("cnt2 is %d", count\_stepper);

// count\_stepper++;

// }

else{

// lcd\_clear();

//// printf("steps done");

// lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

// printf("cnt3 is %d", count\_stepper);

// break;

return;

}

// }

}

void stepper2(char direction){

int count\_stepper = 0;

// while (1){

if (count\_stepper<1){

lcd\_clear();

printf("step1");

lcd\_set\_ddram\_addr(LCD\_LINE2\_ADDR);

printf("Step speed %d",speed2);

for(int i=0;i<110;i++){

full\_drive(direction, 2);

// full\_drive(anti\_clockwise, 2);

}

printf("cnt is %d", count\_stepper);

count\_stepper++;

}

// }

}

// while (1){

// if (count\_stepper<2){

// for(int i=0;i<steps;i++){

// full\_drive(anti\_clockwise);

// count\_stepper++;

// }

// }

// else{

// break;

// }

// }

//return;

//}

void full\_drive (char direction, int stepper\_no){

if (stepper\_no == 1){

if (direction == anti\_clockwise){

LATA = 0b00000011;

\_\_delay\_ms(speed1);

LATA = 0b00000110;

\_\_delay\_ms(speed1);

LATA = 0b00001100;

\_\_delay\_ms(speed1);

LATA = 0b00001001;

\_\_delay\_ms(speed1);

LATA = 0b00000011;

\_\_delay\_ms(speed1);

}

if (direction == clockwise){

LATA = 0b00001001;

\_\_delay\_ms(speed1);

LATA = 0b00001100;

\_\_delay\_ms(speed1);

LATA = 0b00000110;

\_\_delay\_ms(speed1);

LATA = 0b00000011;

\_\_delay\_ms(speed1);

LATA = 0b00001001;

\_\_delay\_ms(speed1);

}

}

if (stepper\_no == 2){ //RC1,5, 6, 7

if (direction == anti\_clockwise){

LATCbits.LATC0 = 1;

LATEbits.LATE0 = 1;

LATEbits.LATE1 = 0;

LATEbits.LATE2 = 0;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 0;

LATEbits.LATE0 = 1;

LATEbits.LATE1 = 1;

LATEbits.LATE2 = 0;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 0;

LATEbits.LATE0 = 0;

LATEbits.LATE1 = 1;

LATEbits.LATE2 = 1;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 1;

LATEbits.LATE0 = 0;

LATEbits.LATE1 = 0;

LATEbits.LATE2 = 1;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 1;

LATEbits.LATE0 = 1;

LATEbits.LATE1 = 0;

LATEbits.LATE2 = 0;

\_\_delay\_ms(speed2);

}

if (direction == clockwise){

LATCbits.LATC0 = 1;

LATEbits.LATE0 = 0;

LATEbits.LATE1 = 0;

LATEbits.LATE2 = 1;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 0;

LATEbits.LATE0 = 0;

LATEbits.LATE1 = 1;

LATEbits.LATE2 = 1;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 0;

LATEbits.LATE0 = 1;

LATEbits.LATE1 = 1;

LATEbits.LATE2 = 0;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 1;

LATEbits.LATE0 = 1;

LATEbits.LATE1 = 0;

LATEbits.LATE2 = 0;

\_\_delay\_ms(speed2);

LATCbits.LATC0 = 1;

LATEbits.LATE0 = 0;

LATEbits.LATE1 = 0;

LATEbits.LATE2 = 1;

\_\_delay\_ms(speed2);

}

}

if (stepper\_no == 3){

printf("what");

if (direction == clockwise){ //RC1,5,6,7

LATCbits.LATC5 = 1;

LATCbits.LATC6 = 0;

LATCbits.LATC7 = 0;

LATAbits.LATA5 = 1;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 1;

LATCbits.LATC6 = 1;

LATCbits.LATC7 = 0;

LATAbits.LATA5 = 0;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 0;

LATCbits.LATC6 = 1;

LATCbits.LATC7 = 1;

LATAbits.LATA5 = 0;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 0;

LATCbits.LATC6 = 0;

LATCbits.LATC7 = 1;

LATAbits.LATA5 = 1;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 1;

LATCbits.LATC6 = 0;

LATCbits.LATC7 = 0;

LATAbits.LATA5 = 1;

\_\_delay\_ms(speed1);

}

if (direction == anti\_clockwise){

LATCbits.LATC5 = 1;

LATCbits.LATC6 = 1;

LATCbits.LATC7 = 0;

LATAbits.LATA5 = 0;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 0;

LATCbits.LATC6 = 1;

LATCbits.LATC7 = 1;

LATAbits.LATA5 = 0;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 0;

LATCbits.LATC6 = 0;

LATCbits.LATC7 = 1;

LATAbits.LATA5 = 1;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 1;

LATCbits.LATC6 = 0;

LATCbits.LATC7 = 0;

LATAbits.LATA5 = 1;

\_\_delay\_ms(speed1);

LATCbits.LATC5 = 1;

LATCbits.LATC6 = 1;

LATCbits.LATC7 = 0;

LATAbits.LATA5 = 0;

\_\_delay\_ms(speed1);

}

}

}