//March 17th 3:28pm: This code is tested and working correctly with calibrated servos.

#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

void PWM\_Init(){

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

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

// TRISAbits.TRISA0 = 0;

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

CCP2CON = 0x0C;

}

void PWM\_End(){

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

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

// TRISAbits.TRISA0 = 0; /\* Set CCP1 pin as output for PWM out \*/

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

CCP2CON = 0x00;

}

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;

PWM\_Init();

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

SetDutyCycleTo(2.5688 , Period,1);

SetDutyCycleTo(12.0, Period,2);

// return;

}

void rotatepos(){

int Period;

PWM\_Init();

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

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

SetDutyCycleTo(2.5688, Period,2); //neutral

}

void main() {

int count = 0;

while (1){

// if (count<1){

// rotatepos();

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

rotateneg();

\_\_delay\_ms(80);

rotatepos();

\_\_delay\_ms(80);

rotateneg();

// count++;

// }

}

return;

}