//all four us sensors

/\* March 11 2:25am : This program runs 4 ultrasonic sensors round robin. tested with 4 but RB4 has no signal. The trigger for RB4 (RB2) is blinking as expected.

\* (tested with 3 and confirmed in multisensor.c. Reminder: run with KPD disconnected.\*/

//things work for at least 3 mins lol

#include <xc.h>

#include <stdio.h>

#include <stdbool.h>

#include "configBits.h"

#include "RTC.h"

#include "I2C.h"

#include "lcd.h"

#define us\_delay 150

int dist\_final[4]; // distance array from sensors

int a[4];

int b;

void echo();

volatile int sens;

void print\_echo();

const char keys[] = "123A456B789C\*0#D";

unsigned char temp;

short int temp\_int, tick;

volatile bool key\_was\_pressed = false;

int state = 0;

int disp = 0;

int i = 0;

int pressed = 0;

//I2C

unsigned char counter = 0; // Increments each time a byte is sent

unsigned char keypress; // Stores the data corresponding to the last key press

unsigned char data; // Holds the data to be sent/received

int dists[2] = {0,-1};

bool send = true; //PIC is sending

volatile long avg\_dist;

typedef struct Poles{

int tires\_deployed; //check var

int tires\_final;//check var

int dist\_from\_start;//check var

} Poles;

struct Poles Pole[10]; //declare pole array

//void \_\_interrupt() ISR(){

// if(RBIF == 1){ //Makes sure that it is PORTB On-Change Interrupt

// if (sens){

// RBIE = 0; //Disable On-Change Interrupt

// echo();

// RBIF = 0; //Clear PORTB On-Change Interrupt flag

// RBIE = 1; //Enable PORTB On-Change Interrupt

// }

// if (~sens){

// key\_was\_pressed = true;

// INT1IF = 0; // Clear interrupt flag bit to signify it's been handled

// }

// }

//}

void \_\_interrupt() ISR(){

if(RBIF == 1){ //Makes sure that it is PORTB On-Change Interrupt

if (sens){

RBIE = 0; //Disable On-Change Interrupt

echo();

RBIF = 0; //Clear PORTB On-Change Interrupt flag

RBIE = 1; //Enable PORTB On-Change Interrupt

}

if (~sens){

key\_was\_pressed = true;

INT1IF = 0; // Clear interrupt flag bit to signify it's been handled

}

}

}

void echo(){

if (b==1){

if (RB7 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB7 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[0] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

}

}

if (b==2){

if (RB6 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB6 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[1] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

}

}

if (b==3){

if (RB5 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB5 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[2] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

}

}

if (b==4){

if (RB4 == 1){ //If ECHO is HIGH

TMR1ON = 1;

} //Start Timer

if (RB4 == 0){ //If ECHO is LOW

TMR1ON = 0; //Stop Timer

a[3] = (TMR1L | (TMR1H<<8))/58.82; //Calculate Distance

}

}

}

void trig(){

for(b=1; b<5; b++){

TMR1H = 0; //Sets the Initial Value of Timer

TMR1L = 0; //Sets the Initial Value of Timer

if(b==1){

RBIE = 0;

RD1 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RD1 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

if(b==2){

RBIE = 0;

RD0 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RD0 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

if(b==3){

RBIE = 0;

RB3 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RB3 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

if(b==4){

RBIE = 0;

RB2 = 1; //TRIGGER HIGH

\_\_delay\_us(10); //10uS Delay

RB2 = 0; //TRIGGER LOW

// \_\_delay\_ms(100); //Waiting for ECHO

RBIE = 1;

\_\_delay\_ms(us\_delay);

}

}

}

void moving\_avg(){

if(a[0]>=2 && a[0]<=400){ //Check whether the result is valid or not

dist\_final[0] = a[0

}

if(!(a[0]>=2 && a[0]<=400)){

printf("Distance 1 = X");

}

if(a[1]>=2 && a[1]<=400){ //Check whether the result is valid or not

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

printf("Distance 2 = %d",a[1]);

}

if(!(a[1]>=2 && a[1]<=400)){

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

printf("Distance 2 = X");

}

if(a[2]>=2 && a[2]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Distance 3= %d",a[2]);

}

if(!(a[2]>=2 && a[2]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Distance 3 = X");

}

if(a[3]>=2 && a[3]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Distance 4= %d",a[3]);

}

if(!(a[3]>=2 && a[3]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Distance 4 = X");

}

}

void print\_echo(){

lcd\_clear();

moving\_avg();

if(a[0]>=2 && a[0]<=400){ //Check whether the result is valid or not

printf("Distance 1 = %d",a[0]);

}

if(!(a[0]>=2 && a[0]<=400)){

printf("Distance 1 = X");

}

if(a[1]>=2 && a[1]<=400){ //Check whether the result is valid or not

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

printf("Distance 2 = %d",a[1]);

}

if(!(a[1]>=2 && a[1]<=400)){

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

printf("Distance 2 = X");

}

if(a[2]>=2 && a[2]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Distance 3= %d",a[2]);

}

if(!(a[2]>=2 && a[2]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("Distance 3 = X");

}

if(a[3]>=2 && a[3]<=400){ //Check whether the result is valid or not

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Distance 4= %d",a[3]);

}

if(!(a[3]>=2 && a[3]<=400)){

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("Distance 4 = X");

}

}

void ultrasonic\_main(){

TRISB = 0b11110000; //RB5 6 as Input PIN (ECHO)

RBIF = 0; //Clear PORTB On-Change Interrupt Flag

RBIE = 1; //Enable PORTB On-Change Interrupt

//check to see if this works

TRISBbits.RB0 = 0;

LATBbits.LATB0 = 1; //DISABLES KPD

while(send){

trig();

RBIE = 0;

print\_echo();

RBIE = 1;

\_\_delay\_ms(1000);

if ((a[0]>=2 && a[0] <= 15) ||(a[1]>=2 && a[1] <= 15)||(a[2]>=2 && a[2] <= 15)||(a[3] >=2 && a[3]<= 15)){

//Tell Arduino to stop motors by writing 9

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Write('9'); // Write key press data which shows up on Arduino's serial monitor

I2C\_Master\_Stop();

//Turn PIC into receiver mode

send = false;

}

// \_\_delay\_ms(400);

}

while (~send){ //gets data from arduino

//Read Arduino's encoder data

I2C\_Master\_Start();

I2C\_Master\_Write(0b00010001); // 7-bit Arduino slave address + Read

avg\_dist = I2C\_Master\_Read(NACK); // Read one char only

I2C\_Master\_Stop();

if(avg\_dist){

while(1){

// if (dists[0] != 0 && dists[1] == -1){

// dists[1] = data;

// }

// if (dists[0] == 0){

// dists[0] = data;

// }

// lcd\_clear();

//

// printf("%d",avg\_dist); //puts character on LCD

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

// printf("Motor dist1 %d",dists[0]);

// lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

// printf("Motor dist2 %d",dists[1]);

\_\_delay\_ms(8000);

}

send = true; //return to PIC as sender

}

else{

send = false;

}

}

}

/\*

int number\_deploy(int avg\_dist, poles\_detected){

int tires\_t;

int tires\_detected;

// while (DC motors off){

// if (poles\_detected != 0){

if ((avg\_dist/1000)<30){

//Total number of tires tires\_t

tires\_t = 1;

}

if ((avg\_dist/1000)>30){

tires\_t = 2;

}

// }

if (poles\_detected == 0){

tires\_t = 2;

}

if ((a[0]>=2 && a[0] <= 15)){

// sensl\_1 = true; //lower sensor is high

if ((a[1]>=2 && a[1] <= 15)){

// sensl\_2 = true; //

tires\_detected = 1;

}

else{

tires\_detected = 0;

}

}

if ((tires\_detected==1)){

if((a[2]>=2 && a[2] <= 15)&&(a[3] >=2 && a[3]<= 15)){

tires\_detected++

}

}

t\_count = tires\_t - tires\_detected;

return

}\*/

void UI\_main(){

sens = 0;

// i2c\_mine();

// RD2 is the character LCD RS (Register Select pin)

// RD3 is the character LCD enable (E)

// RD4-RD7 are character LCD data lines

LATD = 0x00;

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

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

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

// Enable RB1 (keypad data available) interrupt

INT1IE = 1;

// Initialize LCD

initLCD();

// Enable interrupts

ei();

I2C\_Master\_Init(100000);

I2C\_Master\_Start();

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Stop();

// Main loop

Poles Pole[10]; //is this allowed

// I2C\_Master\_Start(); // Start condition

// I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

// I2C\_Master\_Write(temp); // Write key press data which shows up on Arduino's serial monitor

// I2C\_Master\_Stop();

unsigned long tick = 0;

if((pressed == 0)&&(sens==0)){

lcd\_clear();

printf("1 - Start");

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

printf("2 - Summary");

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("3 - Date&Time ");

}

while(sens==0){

// if(pressed == 0){

// lcd\_clear();

// printf("1 - Start");

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

// printf("2 - Summary");

// lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

// printf("3 - Date&Time ");

// }

if (send){

if(key\_was\_pressed){

pressed = 1;

key\_was\_pressed = false; // Clear the flag

unsigned char keypress = (PORTB & 0xF0) >> 4; //right shift

temp = keys[keypress];

temp\_int = (temp-'0');

if (temp\_int == 0){

state = 0;

lcd\_clear();

printf("1 - Start");

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

printf("2 - Summary");

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("3 - Date&Time ");

}

if ((temp\_int == 1)&&(temp!= '\*')&&(temp!= '#')){

state = 0;

lcd\_clear();

printf("Machine In Use");

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu ");

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

//lcd\_clear();

//Start I2C, writes 1 to the Arduino's serial monitor

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b00010000); // 7-bit Arduino slave address + write

I2C\_Master\_Write(temp); // Write key press data which shows up on Arduino's serial monitor

I2C\_Master\_Stop();

sens = 1;

}

if ((temp\_int == 2)||(state == 2)){

if ((temp\_int == 2)){ //this loop is never traversed..

state = 2;

disp = 0;

lcd\_clear();

printf("Op Time:");

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

printf("2min5sec");

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu #>");

}

if ((temp == '\*') && (disp != 0)){

disp = disp - 1;

}

if ((temp == '#')&& (disp <12)){

disp++;

}

if (disp == 0){

lcd\_clear();

printf("Op Time:");

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

printf("2min5sec");

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu #>");

//\_\_delay\_ms(1500);

}

if (disp == 1){

lcd\_clear();

printf("No. Tires: 3");

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

printf("No. Poles: 2");

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("<\* 0-Menu #>");

//\_\_delay\_ms(1500);

}

if (disp > 1 && disp <11){

i = disp - 2;

lcd\_clear();

printf("P%d Dep: %d", i+1 , Pole[i].tires\_deployed);

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

printf("P%d Fin: %d",i+1 , Pole[i].tires\_final);

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("P%d Dist: %d",i+1 , Pole[i].dist\_from\_start);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("<\* 0-Menu #>");

}

if (disp == 11){

i = disp - 2;

lcd\_clear();

printf("P%d Dep: %d", i+1 , Pole[i].tires\_deployed);

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

printf("P%d Fin: %d",i+1 , Pole[i].tires\_final);

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("P%d Dist: %d",i+1 , Pole[i].dist\_from\_start);

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf("<\* 0-Menu ");

}

}

if (temp\_int == 3){

state = 0;

tick = 0;

while(~key\_was\_pressed){

if(tick % 1000 == 0){

lcd\_clear();

printf("DATE & TIME");

I2C\_Master\_Init(100000);

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b11010000); // 7 bit RTC address + Write

I2C\_Master\_Write(0x00); // Set memory pointer to seconds

I2C\_Master\_Stop(); // Stop condition

// Read current time

I2C\_Master\_Start(); // Start condition

I2C\_Master\_Write(0b11010001); // 7 bit RTC address + Read

for(unsigned char i = 0; i < 6; i++){

time[i] = I2C\_Master\_Read(ACK); // Read with ACK to continue reading

}

time[6] = I2C\_Master\_Read(NACK); // Final Read with NACK

I2C\_Master\_Stop(); // Stop condition

// Print received data on LCD

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

printf("%02x/%02x/%02x", time[6],time[5],time[4]); // Print date in YY/MM/DD

lcd\_set\_ddram\_addr(LCD\_LINE3\_ADDR);

printf("%02x:%02x:%02x", time[2],time[1],time[0]); // HH:MM:SS

lcd\_set\_ddram\_addr(LCD\_LINE4\_ADDR);

printf(" 0-Menu ");

}

//\_\_delay\_ms(1000);

tick++;

//lcd\_clear();

}

//lcd\_clear();

}

}

}

}

}

void main(){

TRISD = 0x00; // LCD Pins as Output

GIE = 1; //Global Interrupt Enable

ADCON1=0x0F; // Set all A/D ports to digital (pg. 222)

initLCD();

T1CON = 0x10; //Initialize Timer Module

// if (sens==0){

UI\_main();

// }

if (sens){

ultrasonic\_main();

}

}