#include <xc.h>

#include <stdbool.h>

#include "configBits.h"

#include "lcd.h"

#include "I2C.h"

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

void main(void) {

// RB1, RB4, RB5, RB6, RB7 as inputs (for keypad)

LATB = 0x00;

TRISB = 0b11110010;

// RD2 is the character LCD RS

// RD3 is the character LCD enable (E)

// RD4-RD7 are character LCD data lines

LATD = 0x00;

TRISD = 0x00;

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

ADCON1 = 0b00001111;

initLCD();

// Write the address of the slave device, that is, the Arduino Nano. Note

// that the Arduino Nano must be configured to be running as a slave with

// the same address given here. Note that other addresses can be used if

// desired, as long as the change is reflected on both the PIC and Arduino

// ends

I2C\_Master\_Init(100000);

I2C\_Master\_Start();

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

I2C\_Master\_Stop();

// Main loop

unsigned char mem[3]; // Initialize array to check for triple-A sequence

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

while(1) {

if(send){

while (PORTBbits.RB1 == 0){

continue;

}

//exit loop once RB1 is read high

keypress = (PORTB & 0xF0) >> 4;

while (PORTBbits.RB1 == 1){

continue;

}

//exit while loop once key is released

data = keys[keypress];

//DISABLES KPD, Allows use of on change pins on Arduino

// TRISEbits.RE0 = 0;

// LATEbits.LATE0 = 1;

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

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

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

I2C\_Master\_Stop();

// Check for a 1. If this occurs, switch

// the PIC to receiver mode. To switch back to transmitter mode,

// reset the PIC

if (data == '1'){

send = false; //PIC is no longer in send mode.

}

// mem[counter] = data;

// counter++;

// counter = (counter == 3) ? 0 : counter; //if count = 3, count is reset to 0 else it is not reset

// if((mem[0] == 'A') && (mem[1] == 'A') && (mem[2] == 'A')){

// send = false;

// }

}

else{ //if PIC is in receiver mode

// Receive data from Arduino and display it on the LCD

I2C\_Master\_Start();

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

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

I2C\_Master\_Stop();

/\* if the serial monitor enters B, PIC returns to receiver mode\*/

// if(data=='B'){

// send = true;

// }

lcd\_clear();

printf("Motor fdjahl");

if(data){ //make sure this only runs once so that the distance array is reliable

lcd\_clear();

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

dists[1] = data;

}

if (dists[0] == 0){

dists[0] = data;

}

//printf("%d",data); //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);

}

//go immediately back to send mode

//

//go back to send mode

// if (dists[1] != -1){ //distances sent to pic

// dists[0] = 0;

// dists[1] = -1;

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

// printf("Sensor dist %d",dists[0]);

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

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

// \_\_delay\_ms(8000);

// send = true; //PIC goes back to send mode

// }

}

}

}