#include <lpc17xx.h>

#include "lcd.h"

#include <stdio.h>

#define TRIG\_PIN (1 << 10)

#define ECHO\_PIN (1 << 11)

//unsigned char team[] = {"TEAM 4 RTOS PROJ\"PICK & PLACE\""};

//unsigned char msg1[] = {"OBJECT NOT DETECCTED"};

//unsigned char msg2[] = {"OBJECT DETECTED"};

unsigned char msg1[] = {"NOT IN RANGE"};

unsigned char msg2[] = {"VEHICLE DETECT"};

unsigned long int temp1 = 0, temp2 = 0;

unsigned char buffer[50];

float distance;

void delay\_us(unsigned int us);

float measureDistance(void);

void displayOnLCD(char \*message);

int i;

//---------Servo functions begin------------

void delay\_uus(uint32\_t us) {

uint32\_t i, j;

for (i = 0; i < us; i++) {

for (j = 0; j < 10; j++) {

\_\_NOP(); // No operation, acts as a delay

}

}

}

void configurePins(void) {

LPC\_SC->PCONP |= (1 << 15); // Power up GPIO

LPC\_GPIO0->FIODIR |= (1 << 8); // Set P0.8 as output for servoA

LPC\_GPIO0->FIODIR |= (1 << 9); // Set P0.9 as output for servoB

}

void setServoPosition(uint32\_t pulseWidth, uint32\_t pin) {

if (pin == 0) {

LPC\_GPIO0->FIOSET = (1 << 8); // Set P0.8 HIGH

} else if (pin == 1) {

LPC\_GPIO0->FIOSET = (1 << 9); // Set P0.9 HIGH

}

delay\_us(pulseWidth); // High for the pulse width

if (pin == 0) {

LPC\_GPIO0->FIOCLR = (1 << 8); // Set P0.8 LOW

} else if (pin == 1) {

LPC\_GPIO0->FIOCLR = (1 << 9); // Set P0.9 LOW

}

delay\_uus(20000 - pulseWidth); // Complete the 20ms period

}

// Function for rotating servo from 0 to 90 (for servoA)

void smoothRotateServoA(uint32\_t startPulse, uint32\_t endPulse, uint32\_t step, uint32\_t delayBetweenSteps) {

uint32\_t pulseWidth;

// Rotate from startPulse to endPulse

for (pulseWidth = startPulse; pulseWidth <= endPulse; pulseWidth += step) {

setServoPosition(pulseWidth, 0); // ServoA is on P0.8

delay\_uus(delayBetweenSteps \* 1000); // Delay between steps

}

}

// Function for rotating servo from 0 to 140 (for servoB)

void smoothRotateServoB(uint32\_t startPulse, uint32\_t endPulse, uint32\_t step, uint32\_t delayBetweenSteps) {

uint32\_t pulseWidth;

// Rotate from startPulse to endPulse

for (pulseWidth = startPulse; pulseWidth <= endPulse; pulseWidth += step) {

setServoPosition(pulseWidth, 1); // ServoB is on P2.1

delay\_uus(delayBetweenSteps \* 1000); // Delay between steps

}

}

// Function for rotating servo from 90 to 0 (for servoA)

void smoothRotateServoA\_Backward(uint32\_t startPulse, uint32\_t endPulse, uint32\_t step, uint32\_t delayBetweenSteps) {

uint32\_t pulseWidth;

// Rotate back from endPulse to startPulse

for (pulseWidth = endPulse; pulseWidth >= startPulse; pulseWidth -= step) {

setServoPosition(pulseWidth, 0); // ServoA is on P2.0

delay\_uus(delayBetweenSteps \* 1000); // Delay between steps

}

}

// Function for rotating servo from 140 to 0 (for servoB)

void smoothRotateServoB\_Backward(uint32\_t startPulse, uint32\_t endPulse, uint32\_t step, uint32\_t delayBetweenSteps) {

uint32\_t pulseWidth;

// Rotate back from endPulse to startPulse

for (pulseWidth = endPulse; pulseWidth >= startPulse; pulseWidth -= step) {

setServoPosition(pulseWidth, 1); // ServoB is on P2.1

delay\_uus(delayBetweenSteps \* 1000); // Delay between steps

}

}

//-----------servo functions end----------

int main(void)

{

int j;

//init\_pins();

SystemInit();

SystemCoreClockUpdate();

LPC\_PINCON->PINSEL4 = 0x00000000;

LPC\_GPIO0->FIODIR = 0x00000001;

LPC\_GPIO0->FIOCLR = 0x00000001;

lcd\_init();

temp1 = 0x80;

lcd\_com();

delay\_lcd(65000);

lcd\_puts(msg1);

temp1 = 0xC0;

lcd\_com();

delay\_lcd(65000);

clr\_disp();

//sprintf(buffer,team);

temp1 = 0x81;

lcd\_com();

delay\_lcd(80);

lcd\_puts(buffer);

for(j=0;j<1000;j++)

delay\_lcd(100000);

delay\_us(100);

clr\_disp();

while (1)

{

clr\_disp();

distance = measureDistance();

/\*sprintf(buffer, "%.2f cm", distance);

temp1 = 0x81;

lcd\_com();

delay\_lcd(800);

delay\_lcd(100000);

lcd\_puts(buffer);

delay\_lcd(100000);

delay\_lcd(100000);

delay\_lcd(100000);

delay\_lcd(100000);\*/

sprintf(buffer, "%.2f cm", distance);

temp1 = 0x81;

lcd\_com();

delay\_lcd(80);

lcd\_puts(buffer);

for(j=0;j<10;j++)

delay\_lcd(100000);

delay\_us(100);

clr\_disp();

if (distance < 5)

{

clr\_disp();

displayOnLCD(msg2);

//rotating the servos

configurePins();

// Smoothly rotate servoA from 0° (1 ms) to 90° (1.5 ms)

smoothRotateServoA(1000, 2500, 10, 20); // ServoA from 0° to 90°

// Smoothly rotate servoB from 0° (1 ms) to 180° (2 ms)

//smoothRotateServoB(1000, 4000, 10, 20); // ServoB from 0° to 180°

// Smoothly rotate servoA from 90° (1.5 ms) to 0° (1 ms)

//smoothRotateServoA\_Backward(1000, 2500, 10, 20); // ServoA from 90° to 0°

// Smoothly rotate servoB from 180° (2 ms) to 0° (1 ms)

//smoothRotateServoB\_Backward(1000, 4000, 10, 20); // ServoB from 180° to 0°

delay\_lcd(100000);

delay\_lcd(100000);

delay\_lcd(100000);

clr\_disp();

}

else

{

//smoothRotateServoA\_Backward(1000, 2500, 10, 20);

sprintf(buffer, msg1);

temp1 = 0x81;

lcd\_com();

delay\_lcd(80);

lcd\_puts(buffer);

for(j=0;j<12;j++)

delay\_lcd(100000);

delay\_us(100);

sprintf(buffer, "%.2f cm", distance);

temp1 = 0xc1;

lcd\_com();

delay\_lcd(80);

lcd\_puts(buffer);

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

delay\_lcd(100000);

delay\_us(100);

clr\_disp();

temp1 = 0x81;

lcd\_com();

clr\_disp();

}

}

}

// Function to measure distance using the ultrasonic sensor

float measureDistance()

{

uint32\_t i;

float distance;

// Set TRIG\_PIN as output and ECHO\_PIN as input

LPC\_GPIO0->FIODIR |= TRIG\_PIN;

LPC\_GPIO0->FIODIR &= ~ECHO\_PIN;

// Ensure TRIG\_PIN is low initially

LPC\_GPIO0->FIOCLR |= TRIG\_PIN;

delay\_us(2);

// Generate 10us pulse on TRIG\_PIN to trigger the sensor

LPC\_GPIO0->FIOSET |= TRIG\_PIN;

delay\_us(10);

LPC\_GPIO0->FIOCLR |= TRIG\_PIN;

// Wait for ECHO\_PIN to go high

while (!(LPC\_GPIO0->FIOPIN & ECHO\_PIN));

// Measure the width of the pulse

i = 0;

while (LPC\_GPIO0->FIOPIN & ECHO\_PIN)

{

i++;

delay\_us(1);

}

// Calculate distance using the pulse width

distance = (i \* 0.017); // Speed of sound is 340 m/s

return distance;

}

void delay\_us(unsigned int us)

{

us = us \* 22; // This is roughly equivalent to 1 us delay

while (us--);

}

//lcd initialization

void lcd\_init()

{

unsigned char arr1[4]={0x30,0x30,0x30,0x20};

unsigned char arr2[4]={0x28,0x0C,0x06,0x80};

unsigned int i;

/\* Ports initialized as GPIO \*/

LPC\_PINCON->PINSEL3 &= 0xFFFF00FF; //P1.20 to P1.23

LPC\_PINCON->PINSEL7 &= 0XFFF3FFFF; //P3.25

LPC\_PINCON->PINSEL7 &= 0xFFCFFFFF; //P3.26

LPC\_PINCON->PINSEL9 &= 0xFCFFFFFF; //P4.28

/\* Setting the directions as output \*/

LPC\_GPIO1->FIODIR |= DT\_CTRL; // data lines - P1.20 to P1.23

LPC\_GPIO3->FIODIR |= RS\_CTRL; // RS - P3.25

LPC\_GPIO3->FIODIR |= RW\_CTRL; // RW - P3.26

LPC\_GPIO4->FIODIR |= EN\_CTRL; // P4.28

clear\_ports();

delay\_lcd(3200);

for (i=0;i<4;i++)

{

temp2=arr1[i];

wr\_cn();

delay\_lcd(30000);

}

for (i=0;i<4;i++)

{

temp1=arr2[i];

lcd\_com();

delay\_lcd(800);

}

return;

}

void lcd\_com(void)

{

temp2= temp1 & 0xf0;

temp2 = temp2 << 16; //data lines from 20 to 23

wr\_cn();

temp2 = temp1 & 0x0f;

temp2 = temp2 << 20;

//temp2 = temp2 << 4;

wr\_cn();

delay\_lcd(20000);

return;

}

// command nibble o/p routine

void wr\_cn(void) //write command reg

{

clear\_ports();

LPC\_GPIO1->FIOPIN = temp2; // Assign the value to the data lines

LPC\_GPIO3->FIOCLR = RW\_CTRL; // clear bit RW

LPC\_GPIO3->FIOCLR = RS\_CTRL; // clear bit RW

LPC\_GPIO4->FIOSET = EN\_CTRL; // EN=1

delay\_lcd(25);

LPC\_GPIO4->FIOCLR = EN\_CTRL; // EN =0

return;

}

// data o/p routine which also outputs high nibble first

// and lower nibble next

void lcd\_data(void)

{

temp2 = temp1 & 0xf0;

temp2 = temp2<< 16;

wr\_dn();

temp2= temp1 & 0x0f;

temp2= temp2 << 20;

//temp2= temp2 << 4;

wr\_dn();

delay\_lcd(1000);

return;

}

// data nibble o/p routine

void wr\_dn(void)

{

clear\_ports();

LPC\_GPIO1->FIOPIN = temp2; // Assign the value to the data lines

LPC\_GPIO3->FIOSET = RS\_CTRL; // set bit RS

LPC\_GPIO3->FIOCLR = RW\_CTRL; // clear bit RW

LPC\_GPIO4->FIOSET = EN\_CTRL; // EN=1

delay\_lcd(25);

LPC\_GPIO4->FIOCLR = EN\_CTRL; // EN =0

return;

}

void delay\_lcd(unsigned int r1)

{

unsigned int r;

for(r=0;r<r1;r++);

return;

}

void clr\_disp(void)

{

temp1 = 0x01;

lcd\_com();

delay\_lcd(10000);

return;

}

void clear\_ports(void)

{

/\* Clearing the lines at power on \*/

LPC\_GPIO1->FIOCLR = DT\_CTRL; //Clearing data lines

LPC\_GPIO3->FIOCLR = RS\_CTRL; //Clearing RS line

LPC\_GPIO3->FIOCLR = RW\_CTRL; //Clearing RW line

LPC\_GPIO4->FIOCLR = EN\_CTRL; //Clearing Enable line

return;

}

void displayOnLCD(char \*message)

{

temp1 = 0xC1; // Set cursor to the second row

lcd\_com();

delay\_lcd(800);

lcd\_puts(msg2);

}

void lcd\_puts(unsigned char \*buf1)

{

unsigned int i=0;

while(buf1[i]!='\0')

{

temp1 = buf1[i];

lcd\_data();

i++;

if(i==16)

{

temp1 = 0xC1;

lcd\_com();

}

}

return;

}