#include <avr/io.h> //standard AVR header

#include <avr/interrupt.h>

#include <stdio.h>

#include <stdlib.h>

#define F\_CPU 16000000UL // THE CPU FREQUENCY

#include <util/delay.h> //delay header

#define LCD\_DPRT PORTD //LCD DATA PORT

#define LCD\_DDDR DDRD //LCD DATA DDR

#define LCD\_DPIN PIND //LCD DATA PIN

#define LCD\_CPRT PORTB //LCD COMMANDS PORT

#define LCD\_CDDR DDRB //LCD COMMANDS DDR

#define LCD\_CPIN PINB //LCD COMMANDS PIN

#define LCD\_RS 2 //LCD RS

#define LCD\_EN 0 //LCD EN

unsigned char dir = 1; //Global Variables

unsigned int OCR;

char angChar [5];

unsigned int dist;

char distChar [4];

void lcdPutValue(unsigned char val){

LCD\_DPRT &= 0x0F;

LCD\_DPRT |= (val&0xF0); //send cmnd to data port

LCD\_CPRT |= (1<<LCD\_EN); //EN = 1 for H-to-L pulse

\_delay\_us(1); //wait to make enable wide

LCD\_CPRT &= ~ (1<<LCD\_EN); //EN = 0 for H-to-L pulse

\_delay\_us(100); //wait to make enable wide

LCD\_DPRT &= 0x0F;

LCD\_DPRT |= val<<4; //send cmnd to data port

LCD\_CPRT |= (1<<LCD\_EN); //EN = 1 for H-to-L pulse

\_delay\_us(1); //wait to make enable wide

LCD\_CPRT &= ~ (1<<LCD\_EN); //EN = 0 for H-to-L pulse

\_delay\_us(100); //wait to make enable wide

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void lcdCommand( unsigned char cmnd ){

LCD\_CPRT &= ~ (1<<LCD\_RS); //RS = 0 for command

lcdPutValue(cmnd);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void lcdData( unsigned char data )

{

LCD\_CPRT |= (1<<LCD\_RS); //RS = 1 for data

lcdPutValue(data);

}

void lcdClear()

{

lcdCommand(0x01);

\_delay\_us(1700);

}

void lcdShowCursor()

{

lcdCommand(0x0E);

\_delay\_us(50);

}

void lcdHideCursor()

{

lcdCommand(0x0C);

\_delay\_us(50);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void lcdInit() {

LCD\_DDDR |= 0xF0;

LCD\_CDDR |= (1<<LCD\_RS)|(1<<LCD\_EN);

LCD\_CPRT &=~(1<<LCD\_EN); //LCD\_EN = 0

\_delay\_us(2000); //wait for init.

lcdCommand(0x33); //send $33 for init.

lcdCommand(0x32); //send $32 for init

lcdCommand(0x28); //init. LCD 2 line,5\*7 matrix

lcdCommand(0x0e); //display on, cursor on

lcdCommand(0x06); //shift cursor right

lcdClear();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void lcdGotoxy(unsigned char x, unsigned char y)

{

unsigned char firstCharAdr[]={0x80,0xC0,0x94,0xD4};//Table 12-4

lcdCommand(firstCharAdr[y-1] + x - 1);

\_delay\_us(100);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void lcdPrint( char \* str )

{

unsigned char i = 0;

while(str[i] != 0) //while it is not end of string

{

lcdData(str[i]);

i++ ;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ultrasonicInit() {

DDRB |= (1<<3); //Ultrasonic Trigger = PB.5

PORTB &= ~(1<<3); //Start with low signal for trigger pin

DDRB &= ~(1<<4); //Ultrasonic Echo = PB.6

PINB |= (1<<4); //Pullup echo pin

TCCR2A = 0x42;

OCR2A = 79;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned int echoUltrasonic() {

TCCR2B = 0x01;

unsigned int echoDuration = 0;

while(((PINB & (1<<4)) == 0) & (echoDuration < 2000)) { //Timeout at 2000 us

\_delay\_us(1);

echoDuration++;

}

TCCR2B = 0x00;

return echoDuration;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main(void) {

ultrasonicInit();

lcdInit();

lcdHideCursor();

DDRB |= (1<<1); //Motor = PB.1

TCCR1A = 0x00;

TCCR1B = 0x09;

sei();

TIMSK1 = (1<<OCIE1A);

OCR1A = 7000;

while(1) {

dist = echoUltrasonic();

OCR = OCR1A;

sprintf(angChar, "%u", OCR);

itoa(0.034 \* dist / 2, distChar, 10);

lcdClear();

lcdGotoxy(1,1);

lcdPrint("OCR1A = ");

lcdGotoxy(9, 1);

lcdPrint(angChar);

lcdGotoxy(1,2);

lcdPrint("Distance = ");

lcdGotoxy(12, 2);

lcdPrint(distChar);

lcdShowCursor();

}

return 0;

}

ISR(TIMER1\_COMPA\_vect) { //Control Servo via timer1.

PORTB ^= (1<<1);

if (dir == 1) { //Incrementing Case.

OCR1A += 10;

if (OCR1A >= 39000) { //Reverse direction at max (180 degrees)

dir = 0;

}

}

else { //Decrementing Case.

OCR1A -= 10;

if (OCR1A <= 7000) { //Reverse direction at min (0 degrees)

dir = 1;

}

}

}