//1st code

#define *F\_CPU* 8000000UL // Define the CPU frequency as 8MHz

#include <avr/io.h>

#include <avr/interrupt.h>

#include <string.h>

#include <util/delay.h>

// Define the baud rate

#define BAUD 9600

#define MYUBRR *F\_CPU*/16/BAUD-1

// Define the data port and control signals

#define lcd\_data PORTA // Data port (changed to PORTA)

#define lcd\_ddr DDRA // Data direction register for the data port

#define rs PB0 // Register select pin (changed to PORTB)

#define rw PB1 // Read/Write pin (changed to PORTB)

#define en PB2 // Enable pin (changed to PORTB)

void lcd\_init();

void cmd(unsigned char a);

void dat(unsigned char b);

void show(const char\* s); // Updated to const char\*

void tx(unsigned char send);

void tx\_str(const char\* s); // Updated to const char\*

unsigned char rx();

char namegps[7], name1gps[7] = "GNRMC,", gpsdat[63];

char msggps, checkgps;

int h;

unsigned char f;

ISR(USART\_RXC\_vect) {

if (UCSRA & (1<<FE)) {

// Frame error detected

UDR; // Read the UDR to clear the error

return;

}

msggps = rx();

if(msggps == '$') {

cli(); // Disable interrupts

for(f = 0; f <= 5; f++) {

namegps[f] = rx();

}

checkgps = *strcmp*(namegps, name1gps);

if(checkgps == 0) {

for(f = 0; f <= 62; f++) {

gpsdat[f] = rx();

}

cmd(0x84);

for(h = 13; h < 15; h++) {

dat(gpsdat[h]);

}

dat('.');

for(h = 15; h < 16; h++) {

dat(gpsdat[h]);

}

for(h = 17; h < 19; h++) {

dat(gpsdat[h]);

}

dat(223);

dat(' ');

dat('N');

cmd(0xc4);

for(h = 26; h < 28; h++) {

dat(gpsdat[h]);

}

dat('.');

for(h = 28; h < 30; h++) {

dat(gpsdat[h]);

}

for(h = 31; h < 33; h++) {

dat(gpsdat[h]);

}

dat(223);

dat(' ');

dat('E');

sei(); // Enable interrupts

}

}

}

void uart\_init(unsigned int ubrr) {

UBRRH = (unsigned char)(ubrr>>8);

UBRRL = (unsigned char)ubrr;

UCSRB = (1 << RXEN) | (1 << TXEN) | (1 << RXCIE); // Enable receiver, transmitter, and RX interrupt

UCSRC = (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1); // 8-bit data

}

void LCD\_Clear() {

cmd(0x01); // clear display

cmd(0x80); // cursor at home position

}

int main(void) {

unsigned int ubrr = MYUBRR;

uart\_init(ubrr);

lcd\_ddr = 0xFF; // Set LCD data port as output (Data port is now PORTA)

DDRB = 0x07; // Set control pins as output (Control signals on PORTB)

lcd\_init();

cmd(0x80);

show(" TEAM-01 ");

*\_delay\_ms*(2000);

LCD\_Clear();

cmd(0x80);

show("\*\*GPS Tracker\*\*");

*\_delay\_ms*(2000);

LCD\_Clear();

lcd\_init();

cmd(0x80);

show("LON: ");

cmd(0xc0);

show("LAT: ");

sei(); // Enable global interrupts

while(1) {

// Main loop does nothing, all work is done in ISR

}

}

void tx(unsigned char send) {

while(!(UCSRA & (1 << UDRE))); // Wait for empty transmit buffer

UDR = send; // Send data

}

void tx\_str(const char\* s) { // Updated to const char\*

while(\*s) {

tx(\*s++);

}

}

unsigned char rx() {

while(!(UCSRA & (1 << RXC))); // Wait for data to be received

return UDR; // Get and return received data from buffer

}

void lcd\_init() {

cmd(0x38); // 8-bit mode, 2 lines, 5x7 font

cmd(0x0e); // Display on, cursor on

cmd(0x01); // Clear display

cmd(0x06); // Entry mode set, increment cursor

cmd(0x0c); // Display on, cursor off

cmd(0x80); // Set cursor position to the beginning of the first line

}

void cmd(unsigned char a) {

lcd\_data = a;

PORTB &= ~(1 << rs); // RS = 0 for command

PORTB &= ~(1 << rw); // RW = 0 for write

PORTB |= (1 << en); // Enable pulse

*\_delay\_ms*(1);

PORTB &= ~(1 << en);

*\_delay\_ms*(1);

}

void dat(unsigned char b) {

lcd\_data = b;

PORTB |= (1 << rs); // RS = 1 for data

PORTB &= ~(1 << rw); // RW = 0 for write

PORTB |= (1 << en); // Enable pulse

*\_delay\_ms*(1);

PORTB &= ~(1 << en);

*\_delay\_ms*(1);

}

void show(const char\* s) { // Updated to const char\*

while(\*s) {

dat(\*s++);

}

}

//Second Code

#define *F\_CPU* 8000000UL // Define the CPU frequency as 8MHz

#include <avr/io.h>

#include <avr/interrupt.h>

#include <string.h>

#include <util/delay.h>

// Define the baud rate

#define BAUD 9600

#define MYUBRR *F\_CPU*/16/BAUD-1

// Define the data port and control signals for the LCD

#define lcd\_data PORTA // Data port (changed to PORTA)

#define lcd\_ddr DDRA // Data direction register for the data port

#define rs PB0 // Register select pin (changed to PORTB)

#define rw PB1 // Read/Write pin (changed to PORTB)

#define en PB2 // Enable pin (changed to PORTB)

void lcd\_init();

void cmd(unsigned char a);

void dat(unsigned char b);

void show(const char\* s); // Updated to const char\*

void uart\_init(unsigned int ubrr);

unsigned char rx();

void check\_range();

int main(void) {

unsigned int ubrr = MYUBRR;

uart\_init(ubrr);

lcd\_ddr = 0xFF; // Set LCD data port as output (Data port is now PORTA)

DDRB = 0x07; // Set control pins as output (Control signals on PORTB)

lcd\_init();

cmd(0x80);

show(" Team-01");

*\_delay\_ms*(1000);

LCD\_Clear();

cmd(0x80);

show(" \*\*GPS Tracker\*\*");

*\_delay\_ms*(1000);

sei(); // Enable global interrupts

while(1) {

check\_range(); // Check if any data is received and update the LCD

*\_delay\_ms*(500); // Delay for 5 seconds

}

}

void uart\_init(unsigned int ubrr) {

UBRRH = (unsigned char)(ubrr>>8);

UBRRL = (unsigned char)ubrr;

UCSRB = (1 << RXEN) | (1 << TXEN); // Enable receiver and transmitter

UCSRC = (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1); // 8-bit data

}

unsigned char rx() {

if (UCSRA & (1 << RXC)) {

return UDR; // Get and return received data from buffer

} else {

return 0; // No data received

}

}

void check\_range() {

unsigned char received\_data = rx();

if (received\_data != 0) {

LCD\_Clear();

cmd(0x80);

show("Lost item is");

cmd(0xC0);

show("within range");

*\_delay\_ms*(500);

LCD\_Clear();

cmd(0x80);

show("Lat:15.3676044");

cmd(0xC0);

show("Lon:75.1270270");

} else {

LCD\_Clear();

cmd(0x80);

show("Lost item is");

cmd(0xC0);

show("not in range");

}

}

void lcd\_init() {

cmd(0x38); // 8-bit mode, 2 lines, 5x7 font

cmd(0x0e); // Display on, cursor on

cmd(0x01); // Clear display

cmd(0x06); // Entry mode set, increment cursor

cmd(0x0c); // Display on, cursor off

cmd(0x80); // Set cursor position to the beginning of the first line

}

void cmd(unsigned char a) {

lcd\_data = a;

PORTB &= ~(1 << rs); // RS = 0 for command

PORTB &= ~(1 << rw); // RW = 0 for write

PORTB |= (1 << en); // Enable pulse

*\_delay\_ms*(1);

PORTB &= ~(1 << en);

*\_delay\_ms*(1);

}

void dat(unsigned char b) {

lcd\_data = b;

PORTB |= (1 << rs); // RS = 1 for data

PORTB &= ~(1 << rw); // RW = 0 for write

PORTB |= (1 << en); // Enable pulse

*\_delay\_ms*(1);

PORTB &= ~(1 << en);

*\_delay\_ms*(1);

}

void show(const char\* s) { // Updated to const char\*

while(\*s) {

dat(\*s++);

}

}

void LCD\_Clear() {

cmd(0x01); /\* clear display \*/

cmd(0x80); /\* cursor at home position \*/

}