Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

CENG 347 Lab 5

Introduction:
In this lab, we explore serial communication and LCD interfacing using a microcontroller. The primary objective is to establish a system that transmits data between the development board and an LCD display using USART communication. By creating a custom LCD driver, we will be able to display both static and dynamic messages sent from a serial terminal (such as Putty).

The lab is divided into two main parts:

1. Digital Clock with Timer0: We will configure Timer0 in Normal mode to create a basic digital clock, displaying the current time with second-level precision on the LCD.
2. Serial Communication and LCD Display: We will set up USART0 communication for 8-bit, asynchronous, no-parity, and one-stop-bit data transfer. This will allow us to send messages from the computer terminal to the LCD, displaying characters, strings, and scrolling text.

Throughout the lab, we will implement various display functionalities, including:

- Printing single characters and strings.
- Displaying user-entered strings.
- Scrolling text across the LCD.
- Clearing and overwriting messages.

By the end of the lab, we will have a fully functional menu-driven system capable of displaying and interacting with messages sent over serial communication.

Procedure:

Part 1: Timer

1. **Configure Timer0:**
    - Set Timer0 to Normal mode with a 1024 prescaler.
      Enable overflow interrupts and global interrupts.

2. **Time Tracking:**
    - Use hours, minutes, and seconds variables.
    - Increment the overflow counter and update the time every second.
      Handle rollover at 60 seconds, 60 minutes, and 24 hours.

3. **Display Time:**
    - Show the time on the LCD in HH:MM:SS format.

Part 2: Serial Setup

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

1. **Wiring:**
    - Connect the **LCD 1602** to the ATMega2560:
        1. **PORTB**: Enable and Reset.
        2. **PORTA (Upper Nibble)**: 4-bit data lines.

2. **LCD Initialization:**
    - Write the `LCD_init()` function to configure the display.
    - Create helper functions for writing, clearing, and scrolling text.

3. **USART0 Configuration:**
    - Set up serial communication (9600 baud, 8-bit, no parity, 1 stop bit).
    - Use interrupts to receive and display data.

4. **Menu System:**
    - Display a menu with four options:
        1. **Single Character:** Display one character.
        2. **String:** Display a preset message.
        3. **User-Entered String:** Display a string from the terminal.
        4. **Scrolling Text:** Scroll a string across the LCD.

Application:
Part 0: LCD.c Given code from the lab document
Code: (Zack)

```
// LCD.c

// Data definitions and member functions for working LCD driver
// LCD KS0066U device

// Last Modified by: Randy C. Hoover

// Date: rev1 - 03/02/2020

// Hardware configuration: Assumes ATMega2560 operating at 16MHz
// Pin details below.

/*****************************************************************
*

The four data lines as well as the two control lines may be
implemented on any available I/O pin of any port.  These are
the connections used for this program:

| ATMega2560
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
 1  | | LCD |
|------------|-----------------|-----|
|          PA7|---------------->| D7  |
|          PA6|---------------->| D6  |
|          PA5|---------------->| D5  |
|          PA4|---------------->| D4  |
|             |                 | D3  |
|             |                 | D2  |
|             |                 | D1  |
|             |                 | D0  |
|          PB1|---------------->| E   |
|          GND|---------------->| RW  |
|          PB0|---------------->| RS  |
*************************************************************************
/

#include <avr/io.h>
#include <util/delay.h>

// Helpful LCD control defines
#define LCD_Reset             0b00110000        // reset the LCD to put in
4-bit mode
#define LCD_4bit_enable       0b00100000        // 4-bit data - can't set
the line display or fonts until this is set
#define LCD_4bit_mode         0b00101000        // 2-line display, 5x8
font
#define LCD_4bit_displayOFF   0b00001000        // set display off
#define LCD_4bit_displayON    0b00001100        // set display on - no
blink
#define LCD_4bit_displayON_B1 0b00001101        // set display on - with
blink
#define LCD_4bit_displayCLEAR 0b00000001        // replace all chars with
"space"
#define LCD_4bit_entryMODE    0b00000110        // set cursor to
write/read from left -> right
#define LCD_4bit_cursorSET    0b10000000        // set cursor position

// For two line mode
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
#define LineOneStart 0x00
#define LineTwoStart 0x40                       // must set DDRAM address
in LCD controller for line two

// Pin definitions for PORTB control lines
#define LCD_EnablePin 1
#define LCD_RegisterSelectPin 0

// Prototypes
void LCD_init(void);
void LCD_E_RS_init(void);
void LCD_write_4bits(uint8_t);
void LCD_EnablePulse(void);
void LCD_write_instruction(uint8_t);
void LCD_write_char(char);

// Important notes in sequence from page 26 in the KS0066U datasheet -
initialize the LCD in 4-bit two line mode
// LCD is initially set to 8-bit mode - we need to reset the LCD
controller to 4-bit mode before we can set anything else
void LCD_init(void)
{
    // Wait for power up - more than 30ms for vdd to rise to 4.5V
    _delay_ms(100);

    // Note that we need to reset the controller to enable 4-bit mode
    LCD_E_RS_init(); // Set the E and RS pins active low for each LCD
reset

    // Reset and wait for activation
    LCD_write_4bits(LCD_Reset);
    _delay_ms(10);

    // Now we can set the LCD to 4-bit mode
    LCD_write_4bits(LCD_4bit_enable);
    _delay_us(80); // delay must be > 39us

    ///////////////// system reset is complete - set up LCD modes
////////////////////
```

```
    // At this point we are operating in 4-bit mode
    // (which means we have to send the high-nibble and low-nibble
separate)
    // and can now set the line numbers and font size
    // Notice: we use the "LCD_write_4bits()" when in 8-bit mode and the
LCD_instruction() (this just
    // makes use of two calls to the LCD_write_4bits() function)
    // once we're in 4-bit mode. The set of instructions are found in
Table 7 of the datasheet.
    LCD_write_instruction(LCD_4bit_mode);
    _delay_us(80); // delay must be > 39us

    // From page 26 (and Table 7) in the datasheet we need to:
    // display - off, display - clear, and entry mode - set
    LCD_write_instruction(LCD_4bit_displayOFF);
    _delay_us(80); // delay must be > 39us

    LCD_write_instruction(LCD_4bit_displayCLEAR);
    _delay_ms(80); // delay must be > 1.53ms

    LCD_write_instruction(LCD_4bit_entryMODE);
    _delay_us(80); // delay must be > 39us

    // The LCD should now be initialized to operate in 4-bit mode, 2
lines, 5 x 8 dot fontsize
    // Need to turn the display back on for use
    LCD_write_instruction(LCD_4bit_displayON);
    _delay_us(80); // delay must be > 39us
}

void LCD_E_RS_init(void)
{
    // Set up the E and RS lines to active low for the reset function
    PORTB &= ~(1 << LCD_EnablePin);
    PORTB &= ~(1 << LCD_RegisterSelectPin);
}

// Send a byte of Data to the LCD module
void LCD_write_4bits(uint8_t Data)
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
{
    // We are only interested in sending the data to the upper 4 bits of
PORTA
    PORTA &= 0b00001111; // Ensure the upper nybble of PORTA is cleared
    PORTA |= Data;       // Write the data to the data lines on PORTA

    // The data is now sitting on the upper nybble of PORTA - need to
pulse enable to send it
    LCD_EnablePulse();   // Pulse the enable to write/read the data
}

// Write an instruction in 4-bit mode - need to send the upper nybble
first and then the lower nybbles
void LCD_write_instruction(uint8_t Instruction)
{
    // ensure RS is low
    // PORTB &= ~(1 << LCD_RegisterSelectPin);
    LCD_E_RS_init(); // Set the E and RS pins active low for each LCD
reset

    LCD_write_4bits(Instruction);       // write the high nybble first
    LCD_write_4bits(Instruction << 4); // write the low nybble
}

// Pulse the Enable pin on the LCD controller to write/read the data lines
- should be at least 230ns pulse width
void LCD_EnablePulse(void)
{
    // Set the enable bit low -> high -> low
    // PORTB &= ~(1 << LCD_EnablePin); // Set enable low
    // _delay_us(1);                   // wait to ensure the pin is low
    PORTB |= (1 << LCD_EnablePin);  // Set enable high
    _delay_us(1);                   // wait to ensure the pin is high
    PORTB &= ~(1 << LCD_EnablePin); // Set enable low
    _delay_us(1);                   // wait to ensure the pin is low
}

// write a character to the display
void LCD_write_char(char Data)
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
{
    // Set up the E and RS lines for data writing
    PORTB |= (1 << LCD_RegisterSelectPin); // Ensure RS pin is set high
    PORTB &= ~(1 << LCD_EnablePin);      // Ensure the enable pin is low
    LCD_write_4bits(Data);                // write the upper nybble
    LCD_write_4bits(Data << 4);          // write the lower nybble
    _delay_us(80);                       // need to wait > 43us
}

// write a string
void LCD_sendString(const char* str)
{
    while (*str)
    {
        LCD_write_char(*str++);
    }
}
```

Code: (Alexis)

```
#include <avr/io.h>
#include <util/delay.h>

#define LCD_Reset             0b00110000
#define LCD_4bit_enable       0b00100000
#define LCD_4bit_mode         0b00101000
#define LCD_4bit_displayOFF   0b00001000
#define LCD_4bit_displayON    0b00001100
#define LCD_4bit_displayON_B1 0b00001101
#define LCD_4bit_displayCLEAR 0b00000001
#define LCD_4bit_entryMODE    0b00000110
#define LCD_4bit_cursorSET    0b10000000

#define LineOneStart 0x00
#define LineTwoStart 0x40

#define LCD_EnablePin 1
#define LCD_RegisterSelectPin 0

void LCD_init(void);
void LCD_E_RS_init(void);
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
void LCD_write_4bits(uint8_t);
void LCD_EnablePulse(void);
void LCD_write_instruction(uint8_t);
void LCD_write_char(char);
void LCD_sendString(const char*);

void LCD_init(void)
{
	_delay_ms(100);
	LCD_E_RS_init();
	LCD_write_4bits(LCD_Reset);
	_delay_ms(10);
	LCD_write_4bits(LCD_4bit_enable);
	_delay_us(80);
	LCD_write_instruction(LCD_4bit_mode);
	_delay_us(80);
	LCD_write_instruction(LCD_4bit_displayOFF);
	_delay_us(80);
	LCD_write_instruction(LCD_4bit_displayCLEAR);
	_delay_ms(80);
	LCD_write_instruction(LCD_4bit_entryMODE);
	_delay_us(80);
	LCD_write_instruction(LCD_4bit_displayON);
	_delay_us(80);
}

void LCD_E_RS_init(void)
{
	PORTB &= ~(1 << LCD_EnablePin);
	PORTB &= ~(1 << LCD_RegisterSelectPin);
}

void LCD_write_4bits(uint8_t Data)
{
	PORTA &= 0b00001111;
	PORTA |= Data;
	LCD_EnablePulse();
}

void LCD_write_instruction(uint8_t Instruction)
{
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
	LCD_E_RS_init();
	LCD_write_4bits(Instruction);
	LCD_write_4bits(Instruction << 4);
}

void LCD_EnablePulse(void)
{
	PORTB |= (1 << LCD_EnablePin);
	_delay_us(1);
	PORTB &= ~(1 << LCD_EnablePin);
	_delay_us(1);
}

void LCD_write_char(char Data)
{
	PORTB |= (1 << LCD_RegisterSelectPin);
	PORTB &= ~(1 << LCD_EnablePin);
	LCD_write_4bits(Data);
	LCD_write_4bits(Data << 4);
	_delay_us(80);
}

void LCD_sendString(const char* str)
{
	while (*str)
	{
	LCD_write_char(*str++);
	}
}
```

Part 1: Timer

Code: (Zack)

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include "LCD.c"

// Hardcoded initial time
volatile uint8_t hours = 12;
volatile uint8_t minutes = 34;
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
volatile uint8_t seconds = 45;

volatile uint16_t overflow_count = 0;  // Count Timer0 overflows

// Function to initialize Timer0 in Normal Mode
void timer0_init(void) {
    // Set Timer0 to Normal Mode
    TCCR0A = 0;  // Normal Mode
    TCCR0B = (1 << CS02) | (1 << CS00);  // Prescaler 1024
    TIMSK0 = (1 << TOIE0);  // Enable overflow interrupt
    sei();  // Enable global interrupts
}

// Function to update the clock time
void update_time() {
    seconds++;

    if (seconds >= 60) {
        seconds = 0;
        minutes++;

        if (minutes >= 60) {
            minutes = 0;
            hours++;

            if (hours >= 24) {
                hours = 0;  // Reset to midnight
            }
        }
    }
}

// Function to display the time on Serial Monitor
void display_time()
{
    char time_str[9];  // Buffer to store formatted time HH:MM:SS

    // Format the time string as HH:MM:SS
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
    snprintf(time_str, sizeof(time_str), "%02d:%02d:%02d", hours, minutes,
seconds);

    // Clear the LCD and move cursor to the beginning
    LCD_write_instruction(LCD_4bit_displayCLEAR);
    _delay_ms(80); // Allow time for the clear operation

    // Set cursor position to the first line
    LCD_write_instruction(LCD_4bit_cursorSET | LineOneStart);

    // Display the formatted time on the LCD
    for (int i = 0; time_str[i] != '\0'; i++) {
        LCD_write_char(time_str[i]);
    }
}

ISR(TIMER0_OVF_vect)
{
    overflow_count++;

    if (overflow_count >= 61) {  // Roughly 1 second
        overflow_count = 0;
        update_time();
        display_time();
    }
}


int main(void)
{
    DDRB = 0x23;
    DDRA = 0xF0;

    timer0_init();           // Initialize Timer0

    //char MyChar = 'C';

    // Initialize the LCD for 4-bit mode, two lines, and 5 x 8 dots
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
    // Inits found on Page 26 of datasheet and Table 7 for function set
instructions
    LCD_init();

    // Write a single character
    //LCD_write_char(MyChar);

    // line two
    //LCD_write_instruction(LCD_4bit_cursorSET | LineTwoStart);
    //_delay_us(80); // delay must be > 37us - datasheet forgets to
mention this
    //MyChar = 'Q';
    //LCD_write_char(MyChar);

    while (1)
    {
        PORTB ^= 0x20;
        _delay_ms(500);
    }
    return 1;
}
```

Code: (Alexis)

```
#include <avr/io.h>
#include <util/delay.h>

//LCD command
#define LCD_Reset             0x30
#define LCD_4bit_enable       0x20
#define LCD_4bit_mode         0x28
#define LCD_4bit_displayOFF   0x08
#define LCD_4bit_displayON    0x0C
#define LCD_4bit_displayCLEAR 0x01
#define LCD_4bit_entryMODE    0x06
#define LCD_4bit_cursorSET    0x80

#define LineOneStart 0x00

#define LCD_EnablePin         1
#define LCD_RegisterSelectPin 0
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
void LCD_init(void);
void LCD_E_RS_init(void);
void LCD_write_nibble(uint8_t data);
void LCD_write_instruction(uint8_t instruction);
void LCD_write_char(char data);
void LCD_EnablePulse(void);
char nibble_to_hex(uint8_t nibble);
void display_nibble_as_hex(void);

void LCD_init(void)
{
   _delay_ms(20);
   LCD_E_RS_init();

   LCD_write_nibble(LCD_Reset);
   _delay_ms(5);
   LCD_write_nibble(LCD_Reset);
   _delay_ms(5);
   LCD_write_nibble(LCD_Reset);
   _delay_ms(5);

   LCD_write_nibble(LCD_4bit_enable);
   _delay_ms(5);

   LCD_write_instruction(LCD_4bit_mode);
   _delay_ms(5);
   LCD_write_instruction(LCD_4bit_displayOFF);
   _delay_ms(5);
   LCD_write_instruction(LCD_4bit_displayCLEAR);
   _delay_ms(5);
   LCD_write_instruction(LCD_4bit_entryMODE);
   _delay_ms(5);
   LCD_write_instruction(LCD_4bit_displayON);
   _delay_ms(5);
}

void LCD_E_RS_init(void)
{
   DDRB |= (1 << LCD_EnablePin) | (1 << LCD_RegisterSelectPin);
   PORTB &= ~((1 << LCD_EnablePin) | (1 << LCD_RegisterSelectPin));

   DDRC |= 0xF0;
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
    PORTC &= 0x0F;
}

void LCD_write_nibble(uint8_t data)
{
    PORTC = (PORTC & 0x0F) | (data & 0xF0);
    LCD_EnablePulse();
}

void LCD_write_instruction(uint8_t instruction)
{
    PORTB &= ~(1 << LCD_RegisterSelectPin);
    LCD_write_nibble(instruction);
    LCD_write_nibble(instruction << 4);
}

void LCD_write_char(char data)
{
    PORTB |= (1 << LCD_RegisterSelectPin);
    LCD_write_nibble(data);
    LCD_write_nibble(data << 4);
    _delay_ms(2);
}

void LCD_EnablePulse(void)
{
    PORTB |= (1 << LCD_EnablePin);
    _delay_us(1);
    PORTB &= ~(1 << LCD_EnablePin);
    _delay_us(1);
}

//4-bit to hex
char nibble_to_hex(uint8_t nibble)
{
    nibble &= 0x0F;
    if (nibble < 10)
        return '0' + nibble;
    return 'A' + (nibble - 10);
}

//display as a hex character
void display_nibble_as_hex(void)
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
{
   uint8_t input = (PIND >> 4) & 0x0F;
   char hex_char = nibble_to_hex(input);

   LCD_write_instruction(LCD_4bit_displayCLEAR);
   _delay_ms(2);
   LCD_write_instruction(LCD_4bit_cursorSET | LineOneStart);
   _delay_ms(2);
   LCD_write_char(hex_char);
}

int main(void)
{
   DDRB |= (1 << 7);
   DDRB |= 0x03;  //set output
   DDRC |= 0xF0;  //set PC4-PC7 outputs

   //PD4-PD7 input
   DDRD &= ~(0xF0);
   PORTD |= 0xF0;

   LCD_init();

   while (1)
   {
       display_nibble_as_hex();
       _delay_ms(500);
   }
}
```

Part 2: Serial Setup

Code: (Zack)

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include "LCD.c"

#define F_CPU 16000000UL    // 16 MHz clock
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
// Function prototypes
void USART0_init(unsigned int ubrr);
void USART0_transmit(char data);
char USART0_receive(void);
void USART0_sendString(const char* str);

int main(void)
{
    DDRB = 0x23;
    DDRA = 0xF0;

    // Initialize the LCD for 4-bit mode, two lines, and 5 x 8 dots
    // Inits found on Page 26 of datasheet and Table 7 for function set
instructions
    LCD_init();

    // Initialize the USART0
    USART0_init(MYUBRR);

    char receivedChar;

    while (1)
    {
        // Receive a character from the terminal
        receivedChar = USART0_receive();

        // Display the received character on the LCD
        LCD_write_char(receivedChar);
    }

    return 0;
}

// Initialize the USART0 for serial communication
void USART0_init(unsigned int ubrr)
{
    // Set baud rate
    UBRR0H = (unsigned char)(ubrr >> 8);
    UBRR0L = (unsigned char)ubrr;
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
    // Enable receiver and transmitter
    UCSR0B = (1 << RXEN0) | (1 << TXEN0);

    // Set frame format: 8 data bits, 1 stop bit, no parity
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

// Transmit a single character over USART0
void USART0_transmit(char data)
{
    // Wait for empty transmit buffer
    while (!(UCSR0A & (1 << UDRE0)));

    // Put data into the buffer, sends the data
    UDR0 = data;
}

// Receive a single character from USART0
char USART0_receive(void)
{
    // Wait for data to be received
    while (!(UCSR0A & (1 << RXC0)));

    // Get and return received data from the buffer
    return UDR0;
}

// Send a string over USART0
void USART0_sendString(const char* str)
{
    while (*str)
    {
        USART0_transmit(*str++);
    }
}
```

Writing Data

Code: (Zack)

```
#include <avr/io.h>
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <string.h>
#include "LCD.c"

#define F_CPU 16000000UL    // 16 MHz clock
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

// Function prototypes
void USART0_init(unsigned int ubrr);
void USART0_transmit(char data);
char USART0_receive(void);
void USART0_sendString(const char* str);
void display_menu();
void scroll_text(const char* str);
void display_menu_final();
void display_menu_term();
void handle_option(char option);
void setLineOne();


int main(void)
{
    DDRB = 0x23;
    DDRA = 0xF0;

    // Initialize the LCD for 4-bit mode, two lines, and 5 x 8 dots
    // Inits found on Page 26 of datasheet and Table 7 for function set
instructions
    LCD_init();

    // Initialize the USART0
    USART0_init(MYUBRR);


   // Redisplay the menu after processing
   _delay_ms(1000);
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
    display_menu();
    display_menu_term();

    while (1)
    {
        char buffer[16];          // Buffer to store the user's input
        uint8_t index = 0;        // Index to track the buffer position
        char receivedChar;

        // Clear the buffer
        memset(buffer, 0, sizeof(buffer));

        // Read characters until the user presses Enter
        while (1)
        {
            receivedChar = USART0_receive();

            // Break the loop if Enter is pressed
            if (receivedChar == '\n' || receivedChar == '\r')
            {
             USART0_transmit('\n');
                buffer[index] = '\0';  // Null-terminate the string
                break;
            }

            // Store the character in the buffer
            if (index < sizeof(buffer) - 1)
            {
                buffer[index++] = receivedChar;
                USART0_transmit(receivedChar);  // Echo back the character
            }
        }

        // Handle the option stored in the buffer
        if (index > 0)
        {
            handle_option(buffer[0]);  // Send the first character to
handle_option
        }
```

```
        // Display the menu again
        USART0_transmit('\n');
        USART0_sendString("==========End of Option=========" );
        USART0_transmit('\n');
        USART0_transmit('\n');
        display_menu_term();
        display_menu_final();
        USART0_receive();

    }
     return 0;
}

// Initialize the USART0 for serial communication
void USART0_init(unsigned int ubrr)
{
    // Set baud rate
    UBRR0H = (unsigned char)(ubrr >> 8);
    UBRR0L = (unsigned char)ubrr;

    // Enable receiver and transmitter
    UCSR0B = (1 << RXEN0) | (1 << TXEN0);

    // Set frame format: 8 data bits, 1 stop bit, no parity
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

// Transmit a single character over USART0
void USART0_transmit(char data)
{
    // Wait for empty transmit buffer
    while (!(UCSR0A & (1 << UDRE0)));

    // Put data into the buffer, sends the data
    UDR0 = data;
}

// Receive a single character from USART0
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
char USART0_receive(void)
{
    // Wait for data to be received
    while (!(UCSR0A & (1 << RXC0)));

    // Get and return received data from the buffer
    return UDR0;
}

// Send a string over USART0
void USART0_sendString(const char* str)
{
    while (*str)
    {
        USART0_transmit(*str++);
    }
}

void display_menu_term()
{
    USART0_sendString("Weclome Options Are:");
    USART0_transmit('\n');
    USART0_sendString("1) single character");
    USART0_transmit('\n');
    USART0_sendString("2) string of characters");
    USART0_transmit('\n');
    USART0_sendString("3) string of characters the user enters");
    USART0_transmit('\n');
    USART0_sendString("4) scrolling text");
    USART0_transmit('\n');
    USART0_sendString("Enter Option: ");
}

// Display the menu on the LCD

void setLineOne()
{
    LCD_write_instruction(LCD_4bit_displayCLEAR);
    _delay_ms(80);
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
    LCD_write_instruction(LCD_4bit_cursorSET | LineOneStart);
    _delay_ms(80);
}


void display_menu()
{
    setLineOne();

    // Welcome Message
    LCD_sendString("Welcome to Lab05");

    LCD_write_instruction(LCD_4bit_cursorSET | LineTwoStart);
    _delay_ms(80);

    LCD_sendString("Options:");
    _delay_ms(3000);

    //Option 1 message
    setLineOne();

    scroll_text("1) single character");

    _delay_ms(1000);

    //Option 2 message
    setLineOne();

    scroll_text("2) string of characters");

    _delay_ms(1000);

    //Option 3 message
    setLineOne();

    scroll_text("3) string of characters the user enters");

    _delay_ms(1000);
```

```
     setLineOne();

     scroll_text("4) scrolling text");

     _delay_ms(1000);

     display_menu_final();
}

void display_menu_final()
{
    //Option Final
    setLineOne();

    LCD_sendString("Enter Option: ");

    _delay_ms(1000);

    LCD_write_instruction(LCD_4bit_cursorSET | LineTwoStart);
    _delay_ms(80);

    LCD_sendString("1,2,3,4 - Enter");
    _delay_ms(3000);
}

// Handle user's menu option

void handle_option(char option)
{
    setLineOne();

    if(option == '1')
    {
        USART0_receive();
        USART0_sendString("Enter Character:" );
        char inputchar = USART0_receive();
        USART0_transmit(inputchar);

        LCD_write_char(inputchar);
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
        LCD_write_instruction(LCD_4bit_cursorSET | LineTwoStart);
        _delay_ms(80);
        LCD_write_char(inputchar);
        _delay_ms(80);
        USART0_transmit('\n');
    }
    else if(option == '2')
    {
        LCD_sendString("Hello!");
    }
    else if(option == '3')
    {
        USART0_receive();
        USART0_sendString("Enter String:" );
        char userStr[16];  // Buffer for user string
        int i = 0;

        // Receive and store string until newline or max length
        while (i < 15)
        {
            char ch = USART0_receive();
            USART0_transmit(ch);
            if (ch == '\n' || ch == '\r') break;  // Stop on newline
            userStr[i++] = ch;
        }
        userStr[i] = '\0';  // Null-terminate the string

        // Display the user-entered string
        LCD_sendString(userStr);
    }
    else if (option == '4')
    {
        scroll_text("Hellow World, I am Computer!");
    }
    else
    {
        LCD_sendString("Invalid Option");
    }
    _delay_ms(2000);  // Delay before returning to menu
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
}

// Scrolls the text across the LCD
void scroll_text(const char* str)
{

    int len = strlen(str);

    // If the string is shorter than 16 chars, no need to scroll
    if (len <= 16)
    {
        LCD_write_instruction(0x01);        // Clear the display
        _delay_ms(30);
        LCD_sendString(str);
        _delay_ms(2000);
        return;
    }

    for (int i = 0; i < len - 15; i++)
    {
        LCD_write_instruction(0x01);        // Clear the display
        _delay_ms(20);

        // Print the next 16 characters
        for (int j = 0; j < 16; j++)
        {
            LCD_write_char(str[i + j]);
        }

        _delay_ms(500);  // Scrolling speed
    }
}
```

Code: (Alexis) I combined the code for the two sections of part 2

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#define LCD_Reset 0x30
#define LCD_4bitEnable 0x20
#define LCD_4bitMode 0x28
#define LCD_4bitDisplayOff 0x08
#define LCD_4bitDisplayOn 0x0C
#define LCD_4bitDisplayClear 0x01
#define LCD_4bitEntryMode 0x06
#define LCD_4bitCursorSet 0x80

#define LineOneStart 0x00
#define LineTwoStart 0x40

#define LCD_EnablePin 1
#define LCD_RS 0

#define BAUD 9600
#define UBRR_VALUE ((F_CPU / (16UL * BAUD)) - 1)

//lcd functions
void lcdInit(void);
void lcdErsInit(void);
void lcdWriteNibble(uint8_t data);
void lcdWriteInst(uint8_t inst);
void lcdWriteChar(char data);
void lcdEnablePulse(void);
void lcdWriteString(const char *str);
char nibbleToHex(uint8_t nibble);

//usart functions
void usartInit(void);
void usartSendChar(char c);
void usartSendString(const char *str);
char usartReceiveChar(void);

//menu functions
void menu(void);
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
void optSingleChar(void);
void optFixedString(void);
void optUserString(void);
void optScrollingText(void);
void scrollText(const char *str);

//lcd init
void lcdInit(void)
{
	_delay_ms(20);
	lcdErsInit();

	lcdWriteNibble(LCD_Reset);
	_delay_ms(5);
	lcdWriteNibble(LCD_Reset);
	_delay_ms(5);
	lcdWriteNibble(LCD_Reset);
	_delay_ms(5);

	lcdWriteNibble(LCD_4bitEnable);
	_delay_ms(5);

	lcdWriteInst(LCD_4bitMode);
	_delay_ms(5);
	lcdWriteInst(LCD_4bitDisplayOff);
	_delay_ms(5);
	lcdWriteInst(LCD_4bitDisplayClear);
	_delay_ms(10);
	lcdWriteInst(LCD_4bitEntryMode);
	_delay_ms(5);
	lcdWriteInst(LCD_4bitDisplayOn);
	_delay_ms(5);
}

//lcd e and rs init
void lcdErsInit(void)
{
	DDRB |= (1 << LCD_EnablePin) | (1 << LCD_RS);
	PORTB &= ~((1 << LCD_EnablePin) | (1 << LCD_RS));

	DDRC |= 0xF0;
```

```
	PORTC &= 0x0F;
}

//write a nibble
void lcdWriteNibble(uint8_t data)
{
	PORTC = (PORTC & 0x0F) | (data & 0xF0);
	lcdEnablePulse();
}

//write an instruction
void lcdWriteInst(uint8_t inst)
{
	PORTB &= ~(1 << LCD_RS);
	lcdWriteNibble(inst);
	lcdWriteNibble(inst << 4);
}

//write a character
void lcdWriteChar(char data)
{
	PORTB |= (1 << LCD_RS);
	lcdWriteNibble(data);
	lcdWriteNibble(data << 4);
	_delay_ms(2);
}

//lcd enable pulse
void lcdEnablePulse(void)
{
	PORTB |= (1 << LCD_EnablePin);
	_delay_us(1);
	PORTB &= ~(1 << LCD_EnablePin);
	_delay_us(1);
}

//write a string
void lcdWriteString(const char *str)
{
	while (*str)
	{
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
	lcdWriteChar(*str++);
	}
}

//convert nibble to hex
char nibbleToHex(uint8_t nibble)
{
	nibble &= 0x0F;
	if (nibble < 10)
	return '0' + nibble;
	return 'A' + (nibble - 10);
}

//usart init
void usartInit(void)
{
	UBRR0H = (uint8_t)(UBRR_VALUE >> 8);
	UBRR0L = (uint8_t)(UBRR_VALUE);
	UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
	UCSR0B = (1 << RXEN0) | (1 << TXEN0);
}

//send char via usart
void usartSendChar(char c)
{
	while (!(UCSR0A & (1 << UDRE0))) ;
	UDR0 = c;
}

//send string via usart
void usartSendString(const char *str)
{
	while (*str)
	{
	usartSendChar(*str++);
	}
}

//receive char via usart
char usartReceiveChar(void)
{
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
	while (!(UCSR0A & (1 << RXC0))) ;
	return UDR0;
}

//menu
void menu(void)
{
	usartSendString("\r\nMenu:\r\n");
	usartSendString("1. Single Character\r\n");
	usartSendString("2. Fixed String\r\n");
	usartSendString("3. User String\r\n");
	usartSendString("4. Scrolling Text\r\n");
	usartSendString("Enter option (1-4): ");
}

//main
int main(void)
{
	DDRB |= 0x03;
	DDRC |= 0xF0;

	lcdInit();
	usartInit();

	usartSendString("\r\n--- LCD Serial Interface ---\r\n");

	while (1)
	{
	menu();
	char choice = usartReceiveChar();
	usartSendChar(choice);
	switch (choice)
	{
		case '1':
			optSingleChar();
			break;
		case '2':
			optFixedString();
			break;
		case '3':
			optUserString();
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 5
03/26/25

```
                  break;
            case '4':
                  optScrollingText();
                  break;
            default:
                  usartSendString("\r\nInvalid option.\r\n");
                  break;
      }
      usartSendString("\r\nPress any key to continue...");
      usartReceiveChar();
      }

      return 0;
}
```

Results:

Part 1: Timer

The code ran successfully and as expected. This was verified by the TA visually.

Part 2: Serial Setup

The code ran successfully and as expected. This was verified by the TA visually.

Conclusion

In this lab, we successfully interfaced an LCD display with the arduino using a 4-bit data transmission mode. We implemented a digital clock using Timer0 in Normal mode and established serial communication via USART0 to send and display messages from a terminal. The menu-driven system allowed users to print single characters, display static and user-input strings, and implement scrolling text. Our code functioned as expected, handling character input, line wrapping, and screen clearing. This lab reinforced our understanding of LCD initialization, serial communication, and timer functionality, providing hands-on experience in embedded systems programming. We were able to successfully complete it and had the TA verify the code visually.