Namal University

Mianwali

Department of Electrical Engineering

EE-252L: Introduction to Embedded Systems

Lab Manual: 10

AVR LCD Programming

Students Name	**Riaz Ud Din**
Roll Number	**NIM-BSEE-2021-36**
Submission Date	**6 / 20 /2023**
Marks Obtained	

Instructors: Dr. Hamza Zad Gul

Objectives

In this lab, the student will learn about logic programming and I/O port interfacing to configure the programmer and Microchip studio to program atmega328p.

Course Learning Outcomes

CLO1: Practice the correct use of programming constructs of assembly language
CLO2: Construct systems by interfacing AVR peripherals
CLO3: Perform the assigned task individually/as a team effectively
CLO4: Report the outcomes of task performed effectively in oral and written form

Software

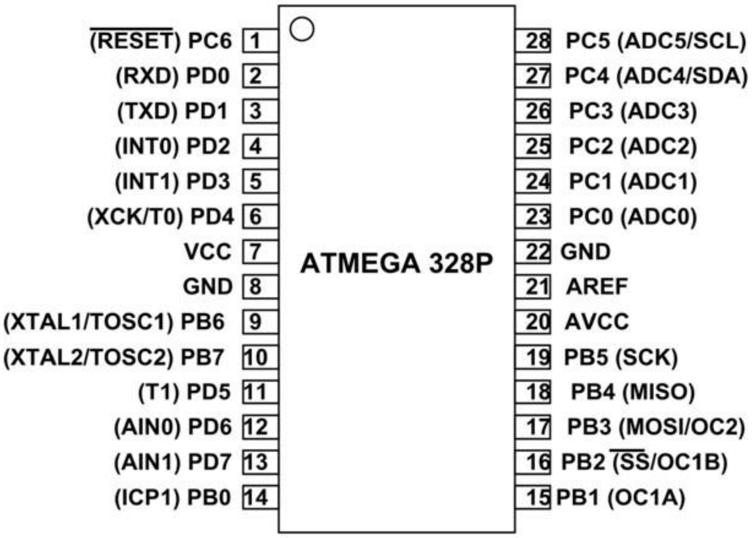
- Microchip studio

Hardware

- Atmega 328p
- Atmega328p USBasp programmer Board
- Breadboard
- Connecting wires
- LEDs
- Resistors
- Capacitors
- Crystal oscillator
- Push buttons
- Oscilloscope
- 16x2 LCD

Instructions

- You must submit the lab report complete within given deadline.
- Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom.
- Multiple such incidents will result in disciplinary action being taken.



Introduction:

How 16x2 LCD works?

The displays available on the market are based on the HD44780 standard. This standard means that the display supports almost all characters of the ASCII character table. Therefore, it is possible to display the most important characters. The display controller, which is integrated in the display, can generate these characters, and send them to the matrix. In addition to the already 208 known characters, it is also possible to draw any other characters and signs. The display controller handles most of the operations, so the code for the microcontroller is very short.

Wiring Diagram

In addition to the microcontroller, a 16 or 10 MHz clock is needed, which can be generated via a quartz oscillator with two capacitors. In addition to the display, a 10K potentiometer is needed to adjust the contrast of the LCD. The display can be controlled with 8 or 4 data lines. In this example, the display is controlled with 4-bit, as it can save 4 data lines. In addition, the display requires two more data lines (EN & RS), which are responsible for the activation and control of the display. The R / W pin is not needed and therefore connected to ground. The last two pins are used for the backlight.

Pin	Function
VSS	GND (ground)
VDD	5V (positive pole)
V0	Contrast setting
RS	switching (0 = command, 1 = data)
RW	read / write (connected to GND)
E	Enable (activates display control)
D0 - D7	data bits (4-bit = D0-D3, 8-bit = D0-D7)
A	Anode (positive pole of LED)
K	cathode (negative pole of LED)

Coding

To display data on LCD we need to follow following steps.

1. Initialize the lcd
2. Send control commands for specific operation (details are given in table below).
3. Send Characters or string to lcd.

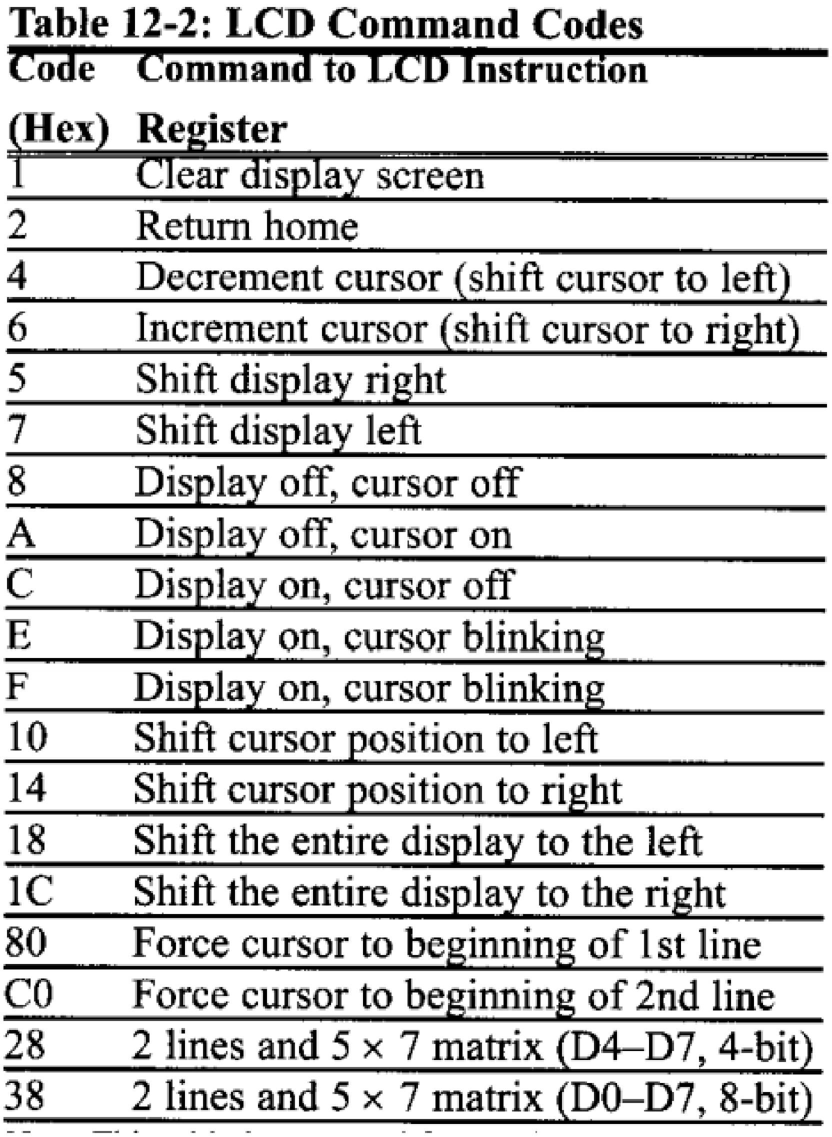


Table 12-2: LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
28	2 lines and 5 × 7 matrix (D4–D7, 4-bit)
38	2 lines and 5 × 7 matrix (D0–D7, 8-bit)

I: Write the following code in microchip studio and implement the circuit in proteus. Also implement this circuit on hardware
Write comments in front of each line of code

```
/*
 * lab_10.cpp
 *
 * Created: 6/20/2023 1:18:42 PM
 * Author : fahim
 */

#define F_CPU 16000000UL   // AVR frequency
#include <avr/io.h>        // AVR standard I/O header
#include <util/delay.h>    // Delay header
#include <inttypes.h>      // Integer types header
#define LCD_Port PORTD     // LCD data port
#define LCD_DPin  DDRD     // LCD data direction register
#define RSPIN PD0          // Register select pin for LCD
```

```
#define ENPIN PD1         // Enable pin for LCD

// Function to send a command to the LCD
void LCD_Action(unsigned char cmnd)
{
      LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); // Send high nibble of command
      LCD_Port &= ~(1 << RSPIN);      // RS = 0 for command mode
      LCD_Port |= (1 << ENPIN);        // Enable LCD (EN = 1) for high-to-low transition
      _delay_us(1);                    // Wait for enable pulse width
      LCD_Port &= ~(1 << ENPIN);      // Disable LCD (EN = 0) for low-to-high transition
      _delay_us(200);                  // Wait for command execution time
      LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  // Send low nibble of command
      LCD_Port |= (1 << ENPIN);        // Enable LCD (EN = 1) for high-to-low transition
      _delay_us(1);                    // Wait for enable pulse width
      LCD_Port &= ~(1 << ENPIN);      // Disable LCD (EN = 0) for low-to-high transition
      _delay_ms(2);                    // Wait for command execution time
}

// Function to initialize the LCD
void LCD_Init (void)
{
      LCD_DPin = 0xFF;       // Set LCD data pins as output
      _delay_ms(15);         // Power-up delay

      LCD_Action(0x33);     // Initialization sequence for 4bit data
      LCD_Action(0x32);     // Initialization sequence for 4bit data
      LCD_Action(0x28);     // 2 lines and 5x7 matrix (4-bit mode)
      LCD_Action(0x02);     // Return home
      LCD_Action(0x0C);     // Display on, cursor off
      LCD_Action(0x06);     // Shift cursor right
      LCD_Action(0x01);     // Clear display
      _delay_ms(2);         // Wait for initialization to complete
}

// Function to clear the LCD display
void LCD_Clear()
{
      LCD_Action(0x01);   // Clear display
      _delay_ms(2);       // Wait for clear operation to complete
      LCD_Action(0x80);   // Move cursor to the beginning of the first line
}

// Function to print a string on the LCD
void LCD_Print (char *str)
{
      int i;  // Index variable
      for(i=0; str[i]!=0; i++)  // Iterate through each character in the string
      {
            LCD_Port = (LCD_Port & 0x0F) | (str[i] & 0xF0); // Send high nibble of
character
            LCD_Port |= (1<<RSPIN);                        // Set RS pin high for data
mode
            LCD_Port|= (1<<ENPIN);                         // Enable LCD (EN = 1) for
high-to-low transition
            _delay_us(1);                                  // Wait for enable pulse width
            LCD_Port &= ~(1<<ENPIN);                        // Disable LCD (EN = 0) for
low-to-high transition
                _delay_us(200);                                 // Wait for character execution
time
                LCD_Port = (LCD_Port & 0x0F) | (str[i] << 4);    // Send low nibble of
character
                LCD_Port |= (1<<ENPIN);                          // Enable LCD (EN = 1) for
high-to-low transition
                _delay_us(1);                                   // Wait for enable pulse width
                LCD_Port &= ~(1<<ENPIN);                        // Disable LCD (EN = 0) for
low-to-high transition
                _delay_ms(2);                                   // Wait for character execution
time
        }
}

// Function to print a string on the LCD at a specific position
void LCD_Printpos (char row, char pos, char *str)
{
        if (row == 0 && pos<16)
                LCD_Action((pos & 0x0F) | 0x80);  // Set cursor to the specified position on
the first line
        else if (row == 1 && pos<16)
                LCD_Action((pos & 0x0F) | 0xC0);  // Set cursor to the specified position on
the second line
        LCD_Print(str);                         // Print the string
}

int main()
{
        LCD_Init();                             // Initialize the LCD
        LCD_Print("Fahim Riaz Imran");          // Print initial message

        while (1)
        {
                LCD_Printpos(0, 0, "Fahim Riaz Imran");  // Print on the first line
                _delay_ms(500);                         // Delay for 500ms
                LCD_Printpos(1, 0, "are Friends");       // Print on the second line
                _delay_ms(500);                         // Delay for 500ms
                LCD_Clear();                            // Clear the LCD display
        }

        while (1)
                ;  // Infinite loop to keep the program running

        return 0;
}
```

Answer the following questions:

1. Explain the code written in lab task 1?

The code includes functions for initializing the LCD screen, clearing the display, and printing strings on the screen. The LCD_Init function initializes the pins connected to the LCD display and sets up the display parameters such as 2-line mode and 5x8 dot matrix. The LCD_Clear function clears the contents of the screen while LCD_Print prints a given string on it. The LCD_Printpos function prints a string at a particular position on the screen, with row and position parameters determining where exactly on the screen to print. Finally, the main program initializes the display, prints a string "Dr Hamza is best", and infinitely loops through printing "Dr Hamza is" on line 1 followed by "the best" on line 2 every half second before clearing the screen.

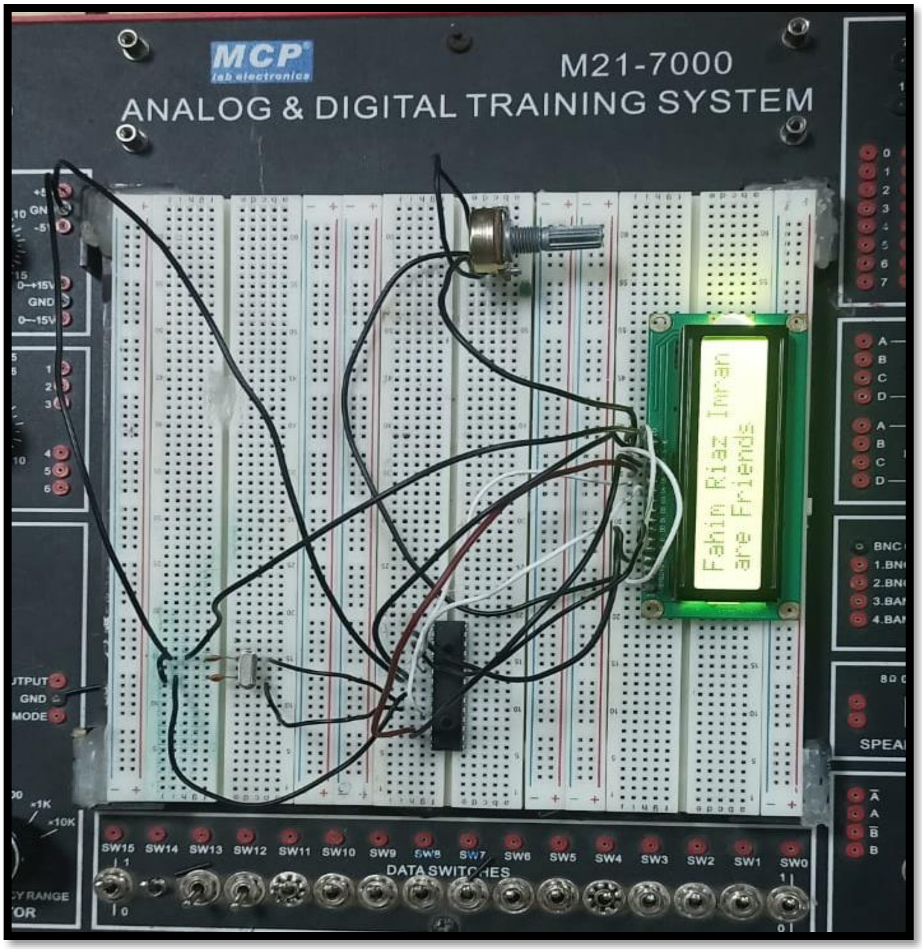
2. Why do we have different wait time in LCD_Print function?

The different wait times in the LCD_Print function allow for proper processing and display of each character on the LCD. The short delay (200 microseconds) after sending the higher nibble ensures the LCD controller can handle the data. The longer delay (2 milliseconds) after sending the lower nibble allows the character to be displayed before proceeding to the next one. These delays prevent issues like data overflow and ensure a clear and readable output on the LCD.

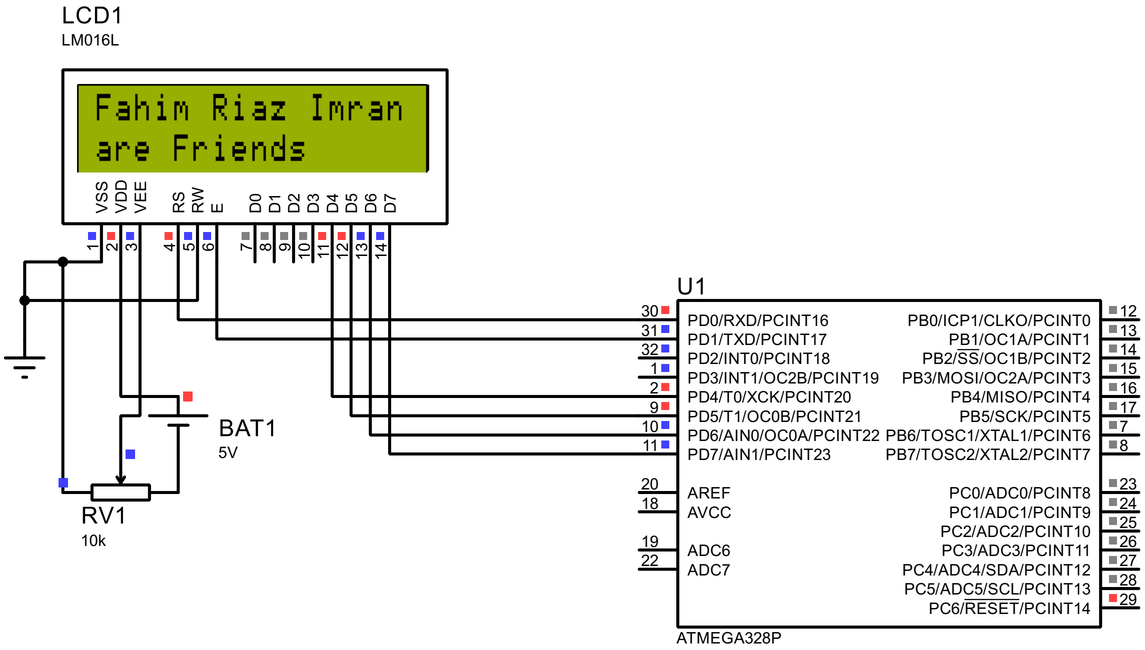
3. What is the purpose of inttypes.h header file?

inttypes.h header file is used in computer programming to define integer data types with specific widths and formats. This means that it helps programmers ensure that their code works properly on different systems, regardless of the underlying architecture or operating system. For example, if a programmer needs an integer variable to be exactly 32 bits wide, they can use the int32_t type defined in inttypes.h instead of relying on platform-specific types like "int" or "long". By using these standardized types, programmers can write more portable and robust code that will work correctly across a variety of platforms and devices. So basically, the purpose of the inttypes.h header file is to provide a set of standard definitions for integer data types that are consistent across different systems and architectures.

Hardware:


MCP lab electronics
M21-7000
ANALOG & DIGITAL TRAINING SYSTEM
Fahim Riaz Imran
are Friends
DATA SWITCHES
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0

Software


LCD1
LM016L
Fahim Riaz Imran
are Friends
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
BAT1
5V
RV1
10k
U1
30
31
32
1
2
9
10
11
20
18
19
22
PD0/RXD/PCINT16
PD1/TXD/PCINT17
PD2/INT0/PCINT18
PD3/INT1/OC2B/PCINT19
PD4/T0/XCK/PCINT20
PD5/T1/OC0B/PCINT21
PD6/AIN0/OC0A/PCINT22
PD7/AIN1/PCINT23
AREF
AVCC
ADC6
ADC7
PB0/ICP1/CLKO/PCINT0
PB1/OC1A/PCINT1
PB2/SS/OC1B/PCINT2
PB3/MOSI/OC2A/PCINT3
PB4/MISO/PCINT4
PB5/SCK/PCINT5
PB6/TOSC1/XTAL1/PCINT6
PB7/TOSC2/XTAL2/PCINT7
PC0/ADC0/PCINT8
PC1/ADC1/PCINT9
PC2/ADC2/PCINT10
PC3/ADC3/PCINT11
PC4/ADC4/SDA/PCINT12
PC5/ADC5/SCL/PCINT13
PC6/RESET/PCINT14
12
13
14
15
16
17
7
8
23
24
25
26
27
28
29
ATMEGA328P

Introduction to Embedded System Lab Rubrics

- **Method of Evaluation** Viva Conducted during lab and lab reports submitted by students

Assessment tool/ weightage/ (CLO, PLO)	**Excellent (10 - 9)**	**Good (8 – 7)**	**Satisfactory (6 – 4)**	**Unsatisfactory (3 – 1)**	**Poor 0**	**Marks Obtained**
Programming (CLO1, PLO5)	Correct Code. Easy to understand with proper comments	Correct Code but without proper indentation or comments	Slightly incorrect code with proper comments	Incorrect code with improper format and no comments	Code not submitted	
Circuit Design (CLO2: PLO3)	Circuit is simulated/implemented correctly without any errors	Circuit is simulated but implemented with minor errors	Circuit is simulated & implementation both have errors	Circuit is simulated & implemented however some components are missing/incorrect value	Circuit is simulated/implemented does not work	
Individual/ Teamwork (CLO3:PLO9)	The student/s worked effectively throughout lab to perform the assigned tasks	The student/s performed all the assigned lab tasks however one member took lead	The student/s completed all tasks however failed to work effectively	The student/s attempted all the tasks however the one member did most of the work	The student/s did not work together/at all	
Lab Report (CLO4:PLO10)	The student was able to effectively answer all questions regarding performed tasks and report provides all information without mistakes	The student was able to effectively answer all questions regarding performed tasks however the report has minor mistakes	The student was able to answer most questions regarding performed tasks and information in report is not communicated effectively	The student was able to answer some questions regarding performed tasks and report is confusing and misleading	The student was not able to answer questions regarding performed tasks and report information is incorrect/irrelevant	
Total						