BIRZEIT UNIVERSITY

Electrical and Computer Engineering

Computer Design Lab – ENCS4110

LCD

Alphanumeric LCD display

This component is specifically manufactured to be used with microcontrollers, which means that it cannot be activated by standard IC circuits. It is used for displaying different messages on a miniature liquid crystal display.

The model described here is for its low price and great capabilities most frequently used in practice (LM016L LCD). It is based on the HD44780 microcontroller (Hitachi) and can display messages in two lines with 16 characters each.

It displays all the letters of alphabet, Greek letters, punctuation marks, mathematical symbols etc. In addition, it is possible to display symbols made up by the user. Other useful features include automatic message shift (left and right), cursor appearance, LED backlight etc.

Function Description:

- **Registers**

  The HD44780U has two 8-bit registers, an instruction register (IR) and a data register (DR). The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator RAM (CGRAM).

  The IR can only be written from the MPU. The DR temporarily stores data to be written into DDRAM or CGRAM and temporarily stores data to be read from DDRAM or CGRAM. Data written into the DR from the MPU is automatically written into DDRAM or CGRAM by an internal operation. The DR is also used for data storage when reading data from DDRAM or CGRAM. When address information is written into the IR, data is read and then stored into the DR from DDRAM or CGRAM by an internal operation.

- **Memory**

  In 16×2 LCD controller HD44780, there are three memory are available to store characters, numbers and special symbols. Which are DDRAM (data display RAM) which stores ASCII codes, CGROM (character generating ROM) which is responsible for stored standard character pattern, and CGRAM (character generating RAM) which holds custom character pattern space total 8 in 2×16 module.

- **Display Data RAM (DDRAM)**

  Display data RAM (DDRAM) stores display data represented in 8-bit character codes. Its extended capacity is 80×8 bits, or 80 characters. The area in display data RAM (DDRAM) that is not used for display can be used as general data RAM.

- **Character Generator ROM (CGROM)**

  The character generator ROM that is responsible for stored standard character pattern generates 5×8 dot or 5×10 dot character patterns from 8-bit character codes. It can generate 208 5×8 dot character patterns and 32 5×10 dot character patterns.

- **Character Generator RAM (CGRAM)**

  The character generating RAM which holds custom character pattern has only 8 memory location available to store user defined characters with address 0x00 - 0x07 , which is shown in the figure 3.1.

Figure 3.1: Character codes

LCD Display

Along one side of a small printed board there are pins used for connecting to the microcontroller. There are in total of 14 pins marked with numbers (16 if the backlight is built in). Their function is described in the table below:

Pin	Symbol	I/O	Description
1	VSS	—	Ground
2	VCC	—	+5V power supply
3	VEE	—	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 4/8-bit data bus
12	DB5	I/O	The 4/8-bit data bus
13	DB6	I/O	The 4/8-bit data bus
14	DB7	I/O	The 4/8-bit data bus

Table 3-1: Pin Descriptions for LCD

LCD Screen Modes

D0-D7 is the data bus and is used to pass commands and characters to the LCD. Data can be transferred to and from the display either as a single 8-bit byte or two 4-bit nibbles.

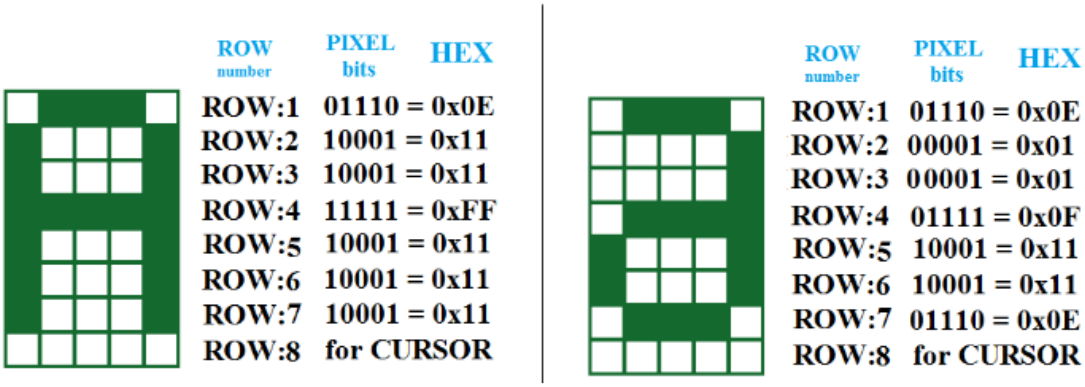
In the second case only the upper four data lines (D4-D7) are used. This 4-bit mode is beneficial when using a microcontroller with few input/output pins available.

Displaying Standard Character on LCD

Out of these three memory locations, DDRAM and CGROM are used to generate regular standard characters (ASCII characters). By using these three memory locations, a user can generate different character fonts and symbols on LCD display. A character font describes the shape and style of the character. Each shape of a character is designed by taking the number of pixels in mind. For example, in 16x2 LCD there are 16 segments available per single line. Each segment contains pixels in 5x7 or 5x10 matrix forms.

For example, in 16×2 LCD there are 16 segments available per single line. Each segment contains pixels in 5×8 or 5×10 matrix forms.

For example, a character in both uppercase ‘A’ and lowercase ‘a’ is designed by energizing corresponding pixels as shown below.



All these eight hexadecimal codes (referred as character pattern) of each character are stored in character generator ROM (CGROM) area.

The Display Data RAM (DDRAM) stores the ASCII code of a character which is sent by the microcontroller. Now the LCD controller (HD44780) maps the corresponding ASCII Code in DDRAM with CGROM address to bring the hexadecimal codes (character pattern) of that particular character. By using those hexadecimal codes the 5x7 matrix segment will light according to that character pattern to display corresponding character on it as shown in figure 3.2.

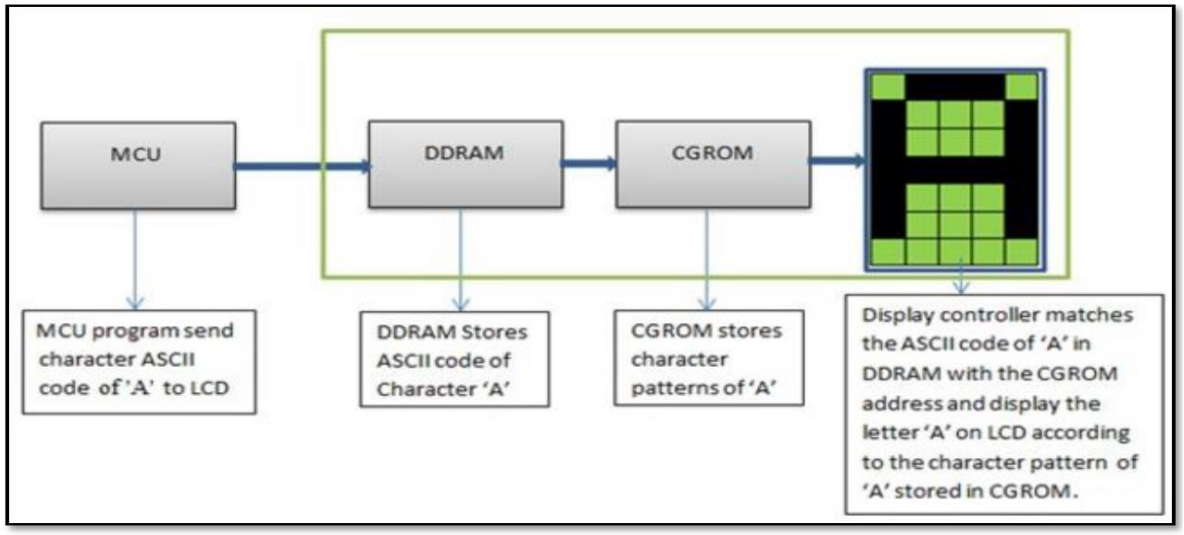


Figure 3.2 : block diagram shows character generation on LCD.

Displaying Custom Characters on LCD display

To create custom characters on LCD, the display controller (HD44780) make use of CGRAM area to store hexadecimal codes (character pattern) which are designed by user. In addition to CGRAM area, DDRAM area is also used to store the CGRAM address of a particular character, which is sent by microcontroller in hexadecimal format.

Lab Work

Lab Work 1

You are going to use these keywords when you search for parts in Proteus:

Part	Keyword
Microcontroller	LPC2138
LCD	LM016L

- Write and simulate a program that displays your name on an LCD

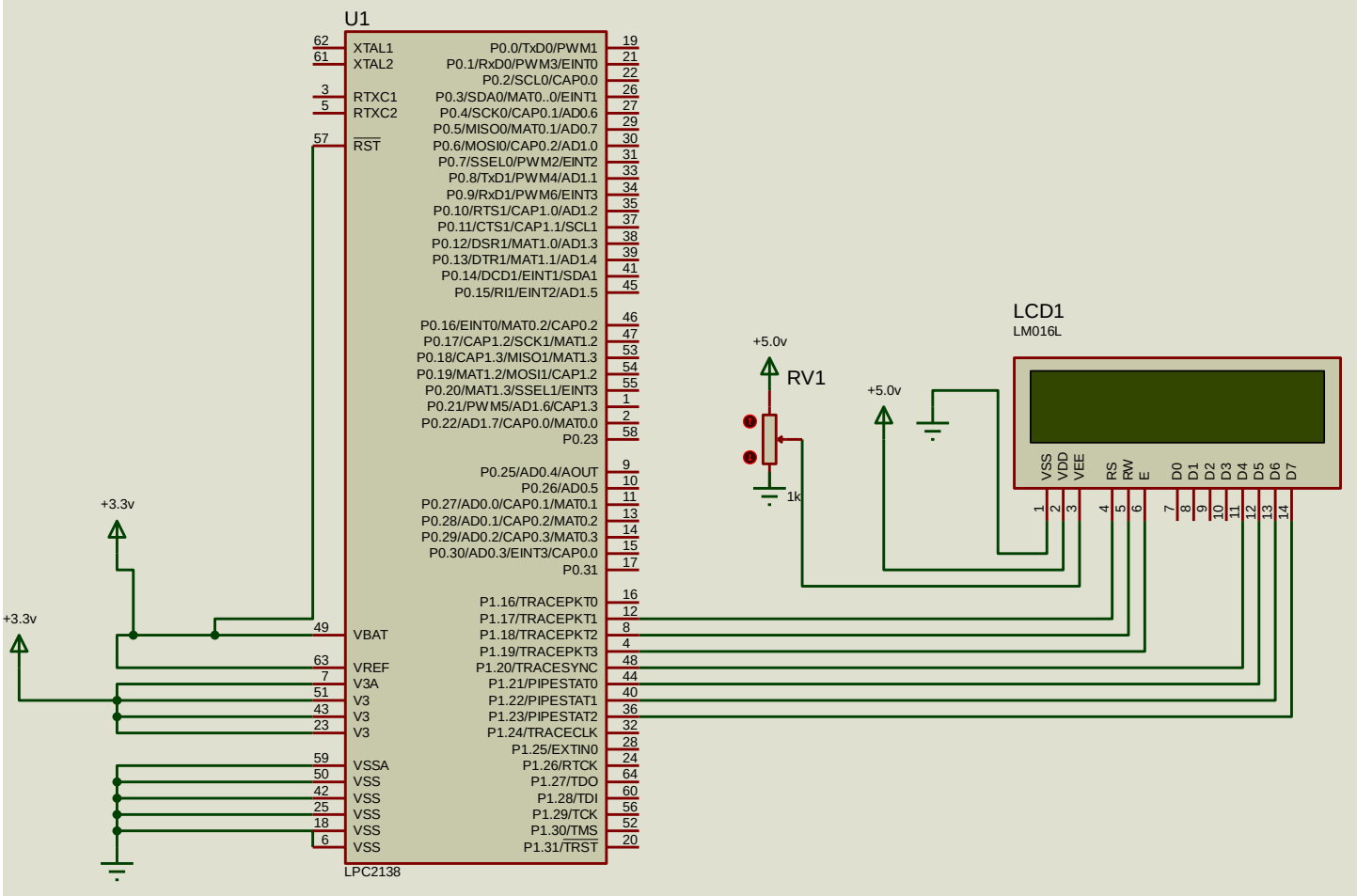
Keil: (Lab work 1)

```
#include <lpc213x.h>
#define RS 0x00020000 /* RS   - P1.17                          */
#define RW 0X00040000 /* R/W - P1.18                          */
#define EN 0X00080000 /* E    - P1.19                         */
#define CLR 0X00FE0000
#define FIRST_ROW 0x80
#define SECOND_ROW 0xC0
#define LCD_CLEAR 0x01


void Delay(unsigned int times)
{
    int i, j;
    for (j = 0; j < times; j++)
        for (i = 0; i < 300; i++)
            ;
}

void LCD_Command(char command)
{
    int Temp;
    IO1CLR = CLR;
    IO1SET = EN;
    IO1CLR = RS;
    IO1CLR = RW;
    Temp = (command & 0xF0) << 16;
    IO1SET = IO1SET | Temp;
    Delay(2);
    IO1CLR = EN;
}
```

```
void LCD_Command1(char command1)
{
    int Temp;
    IO1CLR = CLR; /* Clearing the port pins               */
    IO1SET = EN;  /* Enable pin high                      */
    IO1CLR = RS;  /* RS=0 for command register            */
    IO1CLR = RW;  /* R/W=0 for write                      */
    /* Taking the first nibble of command   */
    Temp = (command1 & 0xF0) << 16;
    IO1SET = IO1SET | Temp; /* Writing it to data line */
    Delay(2);
    IO1CLR = EN; /* Enable pin low to give H-L pulse     */

    /* same as above for the second nibble */
    IO1CLR = CLR;
    IO1SET = EN;
    IO1CLR = RS;
    IO1CLR = RW;
    Temp = (command1 & 0x0F) << 20;
    IO1SET = IO1SET | Temp;
    Delay(2);
    IO1CLR = EN;
}

void LCD_Data(char data)
{
    int Temp;
    IO1CLR = CLR;               /* Clearing the port pins               */
    IO1SET = EN;                /* Enable pin high                      */
    IO1SET = RS;                /* RS=1 for data register               */
    IO1CLR = RW;                /* R/W=0 for write                      */
    Temp = (data & 0xF0) << 16; /* Taking the first nibble of command   */
    IO1SET = IO1SET | Temp;     /* Writing it to data line              */
    Delay(2);
    IO1CLR = EN; /* Enable pin low to give H-L pulse     */

    IO1CLR = CLR;               /* Clearing the port pins               */
    IO1SET = EN;                /* Enable pin high                      */
    IO1SET = RS;                /* RS=1 for data register               */
    IO1CLR = RW;                /* R/W=0 for write                      */
    Temp = (data & 0x0F) << 20; /* Taking the second nibble of command  */
    IO1SET = IO1SET | Temp;     /* Writing it to data line              */
    Delay(2);
    IO1CLR = EN; /* Enable pin low to give H-L pulse     */
}

void LCD_String(unsigned char *dat)
{
    /* Check for termination character */
    while (*dat != '\0')
    {
        /* Display the character on LCD */
        LCD_Data(*dat);
        /* Increment the pointer */
        dat++;
    }
}
```

```
void LCD_Init(void)
{
    Delay(15);
    LCD_Command(0x30);
    Delay(10);
    LCD_Command(0x30);
    Delay(5);
    LCD_Command(0x30);
    LCD_Command(0x20);
    LCD_Command1(0x28);
    LCD_Command1(0x01); /* Clear display                    */
    LCD_Command1(0x06); /* Auto increment                   */
    LCD_Command1(0x0C); /* Cursor off                       */
}

int main()
{
    unsigned char name[] = "ENCS4110";
    IO1DIR = 0x00FE0000; /* LCD pins set as o/p??????????    */
    LCD_Init();          /* Initialise LCD      */
      while(1) {
                LCD_Command1(FIRST_ROW);
                LCD_String(name);
                Delay(50);
                    LCD_Command1(SECOND_ROW);
                LCD_String("Lab");
        Delay(50);

       //LCD_Command1(LCD_CLEAR); /* Clear screen                    */
           //Delay(80);
      }

}
```

Proteus: (lab work 1)



Lab Work 2

You are going to use these keywords when you search for parts in Proteus:

Part	Keyword
Microcontroller	LPC2138
LCD	LM016L

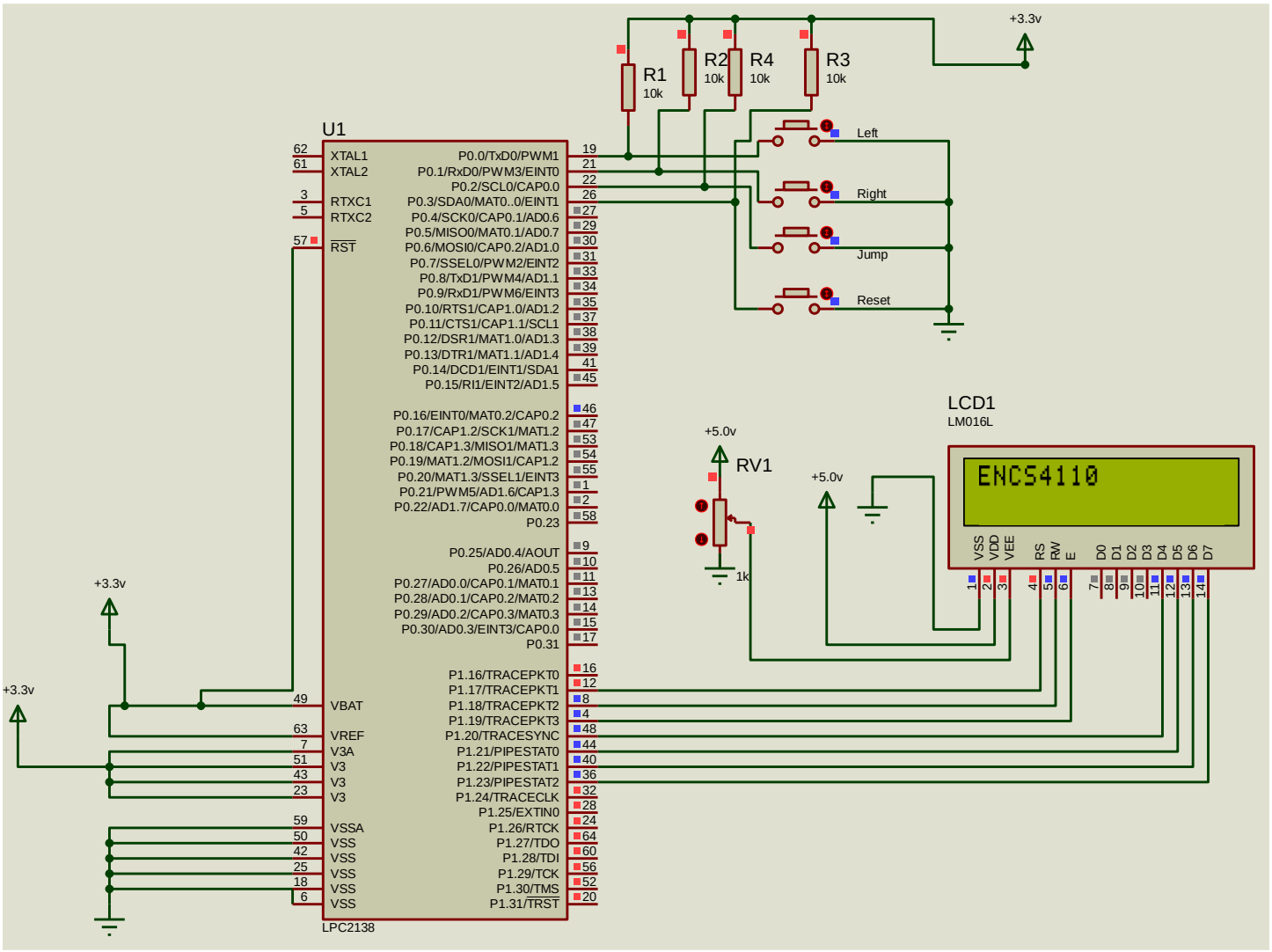
• Write a program that displays your name on LCD with movement. Your program should allow the user to control the direction of the movement (shift left, right, clear or move to the second row) using push buttons.

Keil: (lab work 2)

```
// Do not forget to copy the other LCD functions from the lab work 1 above.

#include <lpc213x.h>
#define RS 0x00020000 /* RS  - P1.17                              */
#define RW 0X00040000 /* R/W - P1.18                              */
#define EN 0X00080000 /* E   - P1.19                              */
#define CLR 0X00FE0000
#define FIRST_ROW 0x80
#define SECOND_ROW 0xC0
#define LCD_CLEAR 0x01

void LCD_set_cursor_pos(int row, int col)
{
    unsigned char cp;
    if (row == 0)
        cp = FIRST_ROW;
    else
        cp = SECOND_ROW;
    cp |= col;
    LCD_Command1(cp);
}

int main()
{
    unsigned char name[] = "ENCS4110";
   int col = 0, row = 0;
    IO1DIR = 0x00FE0000; /* LCD pins set as output P1.16 ..P1.19         */
    IO0DIR &= (0xFFFFFFF0); //Push button keys as inputs P0.0 ..P0.3
    LCD_Init();          /* Initialise LCD      */

 while(1) {

        if (!(IO0PIN & (1 << 0)))                // if Left key is pressed
        {
            col--;
            if (col < 0)
                col = 15;
            LCD_Command1(LCD_CLEAR);
            LCD_set_cursor_pos(row, col);
            LCD_String(name);
            Delay(100);
        }

        if (!(IO0PIN & (1 << 1)))        // Right key
        { //right
            col++;
            col %= 16;
            LCD_Command1(LCD_CLEAR);
            LCD_set_cursor_pos(row, col);
            LCD_String(name);
            Delay(100);
        }

        if (!(IO0PIN & (1 << 2)))              //jump key
        { //jump
            row = (row == 0 ? 1 : 0);
            LCD_Command1(LCD_CLEAR);
```

```
            LCD_set_cursor_pos(row, col);
            LCD_String(name);
            Delay(300);
        }

        if (!(IO0PIN & (1 << 3)))        // Reset key
        { //reset
            LCD_Command1(LCD_CLEAR);
            row = col = 0;
            LCD_set_cursor_pos(row, col);
            LCD_String(name);
            Delay(300);
        }
    }


}
```

Proteus: (Lab work 2)



Lab work 3

Write a program that displays "Hello" at the 1st raw and “World” on the 2nd raw on the LCD. (The upper word should be firstly appears from the left of the LCD then it is shifted continually to the other side. The lower word must have the opposite movement at the time .All that happen after a button press from the user)