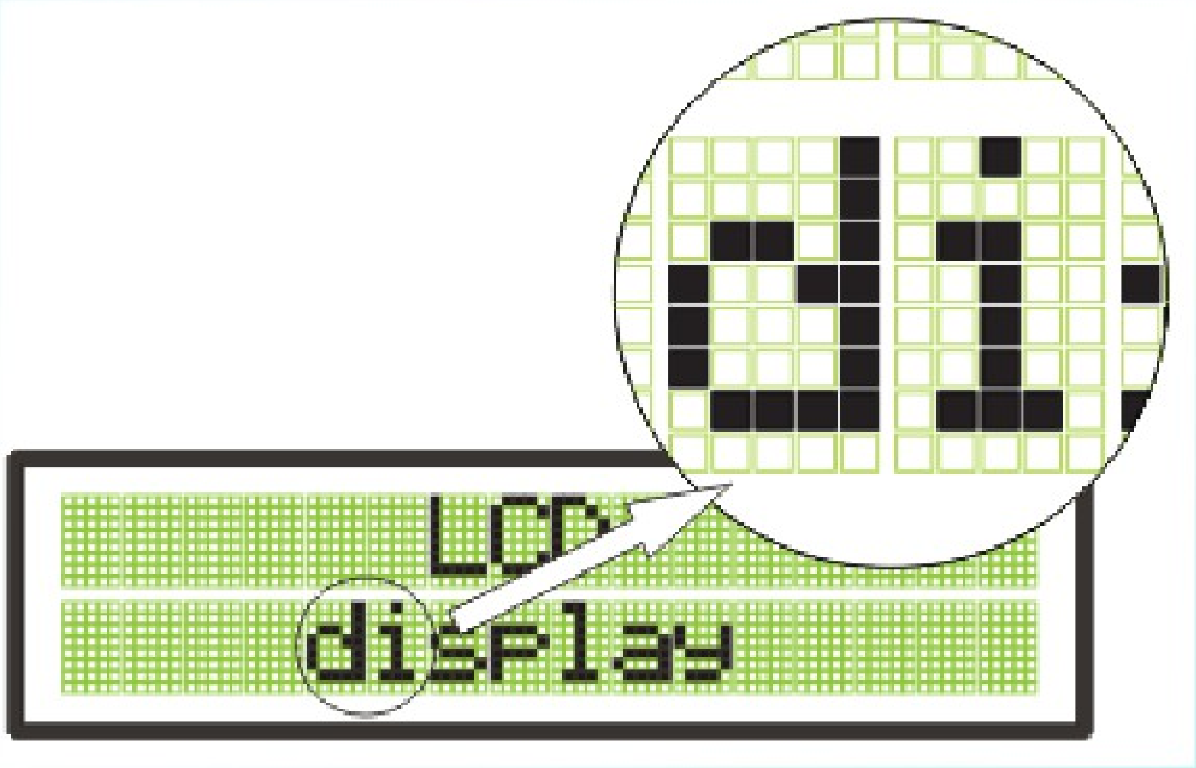
Lec 5: Liquid Crystal Display (LCD)

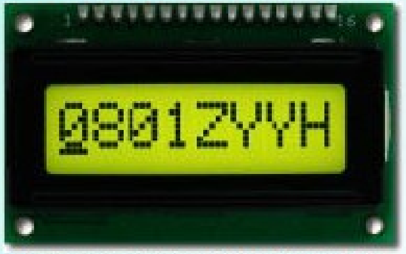
LCD Screen

- **LCD screen consists of lines (rows) and characters (columns).**
- **Every character consists of 5x8 or 5x11 dot matrix.**
- **LCD modules is specified by No. of lines and No. of characters.**
- **LCD modules may be character(0-9,a-z, A-Z) type or Graphic type.**

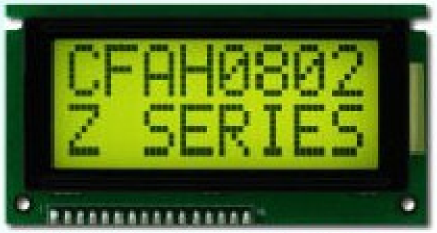


LCD Modules

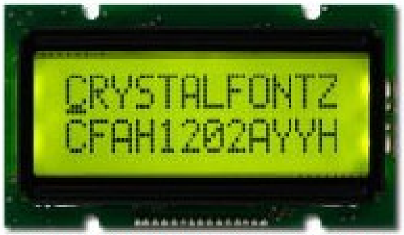
Character LCD Modules



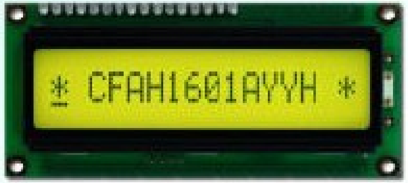
1 x 8 LCD



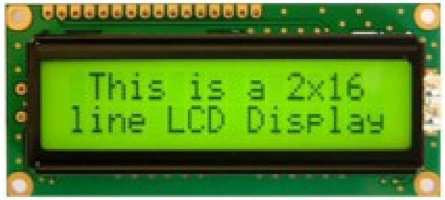
2 x 8 LCD



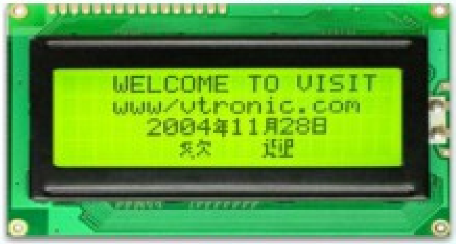
2 x 16 LCD



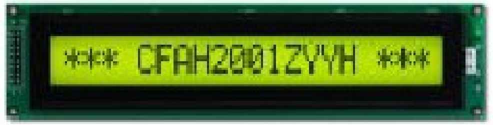
1 x 16 LCD



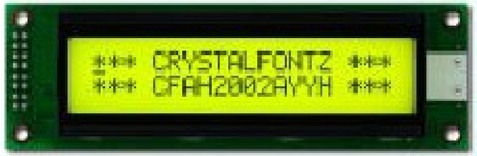
2 x 16 LCD



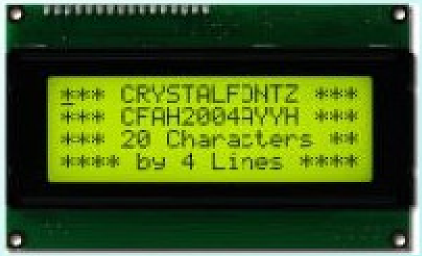
4 x 16 LCD



1 x 20 LCD



2 x 16 LCD



4 x 20 LCD

Graphic LCD Modules

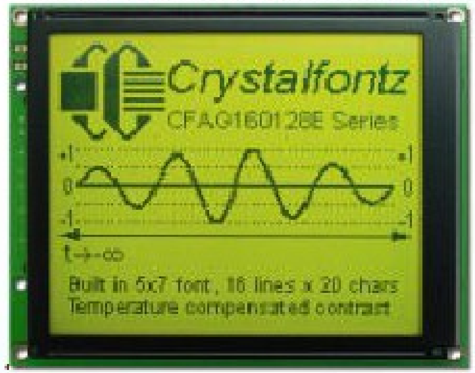
122x32 dots

128x32 dots

160x32 dots

128x64 dots

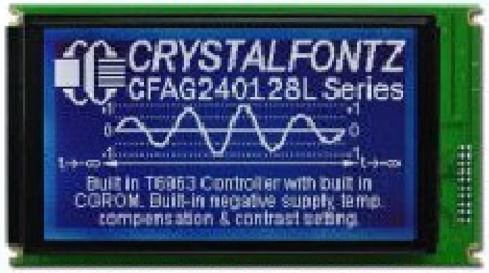
128x128 dots


Crystalfontz
CFAG160128E Series
Built in 5x7 font, 16 lines x 20 chars
Temperature compensated contrast

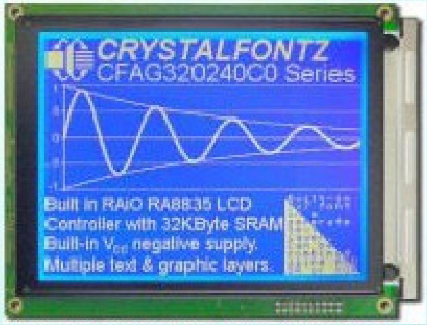
160x128 dots

CRYSTALFONTZ
CFAG160160B Series
Built in LC7981 Controller
Built in negative voltage supply
Built in Character Generator ROM

160x160 dots


CRYSTALFONTZ
CFAG240128L Series
Built in T6963 Controller with built in CGROM. Built-in negative supply, temp. compensation & contrast setting.

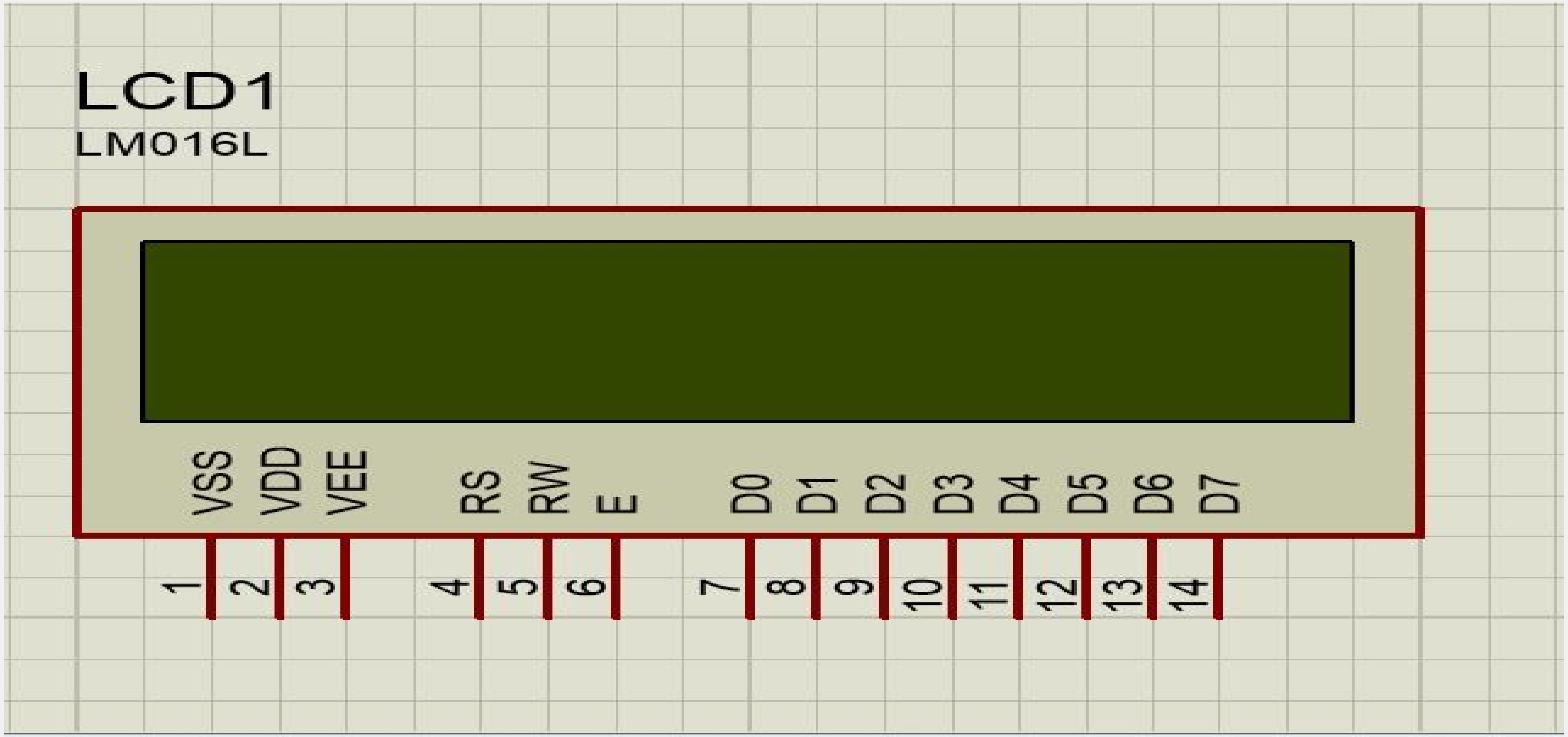
240x128 dots


CRYSTALFONTZ
CFAG320240C0 Series
Built in RAIO RA8835 LCD Controller with 32KByte SRAM
Built-in Vcc negative supply.
Multiple text & graphic layers.

320x240 dots

LCD Display(16X2) LM016L

❖ **LCD LM016L** : is a16x2 Alphanumeric LCD with two rows Of 16 characters in each row.



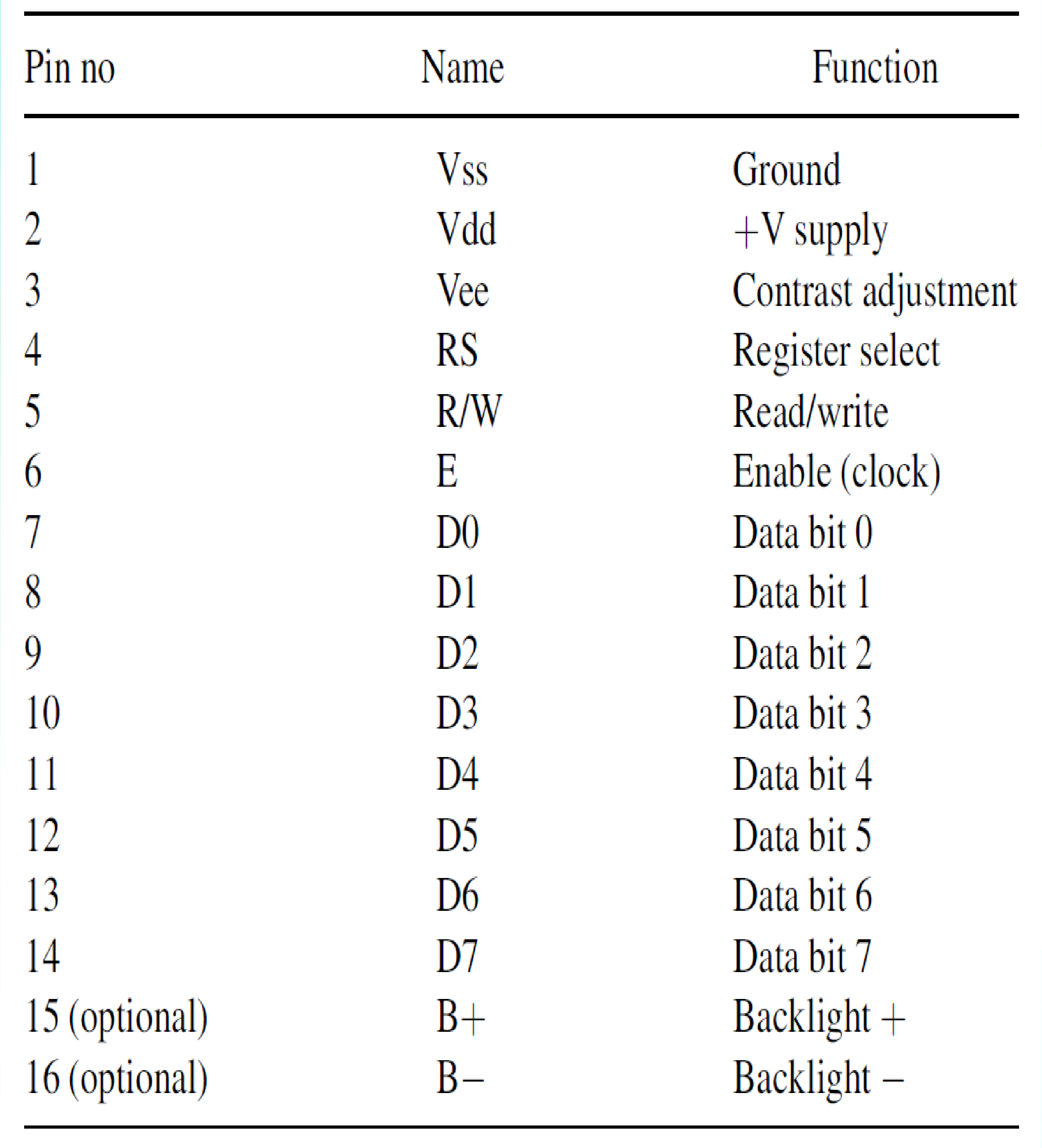
Pin Functions of the LCD Module

- **VSS** is the 0 V or ground. **VDD** pin should be connected to +5 V.
- Pin 3 is named as **VEE** and this is the contrast control pin.
  - This pin is used to adjust the contrast of the LCD and it should be connected to a variable voltage supply **(0-VDD)** through a 5 K/10 K potentiometer.
  - **This pin can be connected to ground if contrast adjustment is not needed.**
- Pin 4 is the Register Select (**RS**):
  - When this pin is **LOW**, any data sent to the display is treated as **commands**.
  - When RS is **HIGH**, data sent is treated as **character** data for the LCD display.
- Pin 5 is the read/write (**R/W**) pin.
  - This pin is pulled **LOW** in order to **write commands or character** data to the LCD (*i.e.* microcontroller to display data transfer).
  - When this pin is **HIGH**, character data or status information can be **read from** the LCD (*i.e.* display to microcontroller data transfer).
  - The R/W pin is usually connected to **ground**, as we normally want to **send commands and data** to the LCD display.

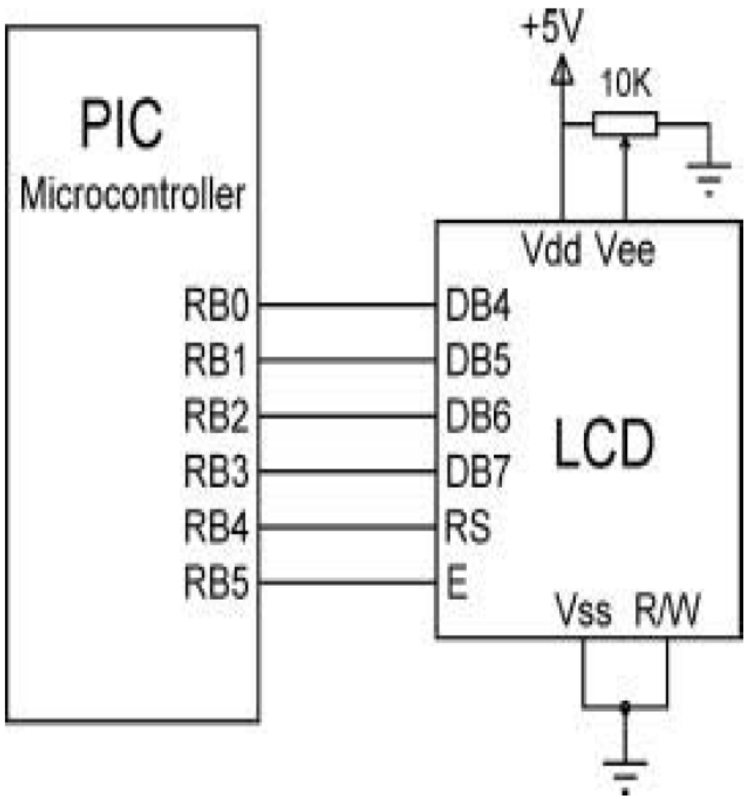
Pin Functions of the LCD Module (*cont'd*)

- Pin 6 is the Enable (**E**) pin which is used to initiate the transfer of commands or character data between the LCD module and the microcontroller.
  - When writing to the display, data is transferred only on the **HIGH to LOW transition** of this pin.
  - When reading from the display, data becomes available after the **LOW to HIGH transition** of the enable pin and this data remains valid as long as the enable pin is HIGH.
- Pins 7 to 14 are the eight data bus lines (**D0 to D7**).
  - Data can be transferred between the microcontroller and the LCD module using either an 8-bit interface or a 4-bit interface.
  - Usually, only the upper four data lines (D4 to D7) are used and the data is transferred as two 4-bit nibbles.
  - This mode has the advantage that fewer I/O lines are required to communicate the microcontroller with the LCD.

LCD pin Configuration



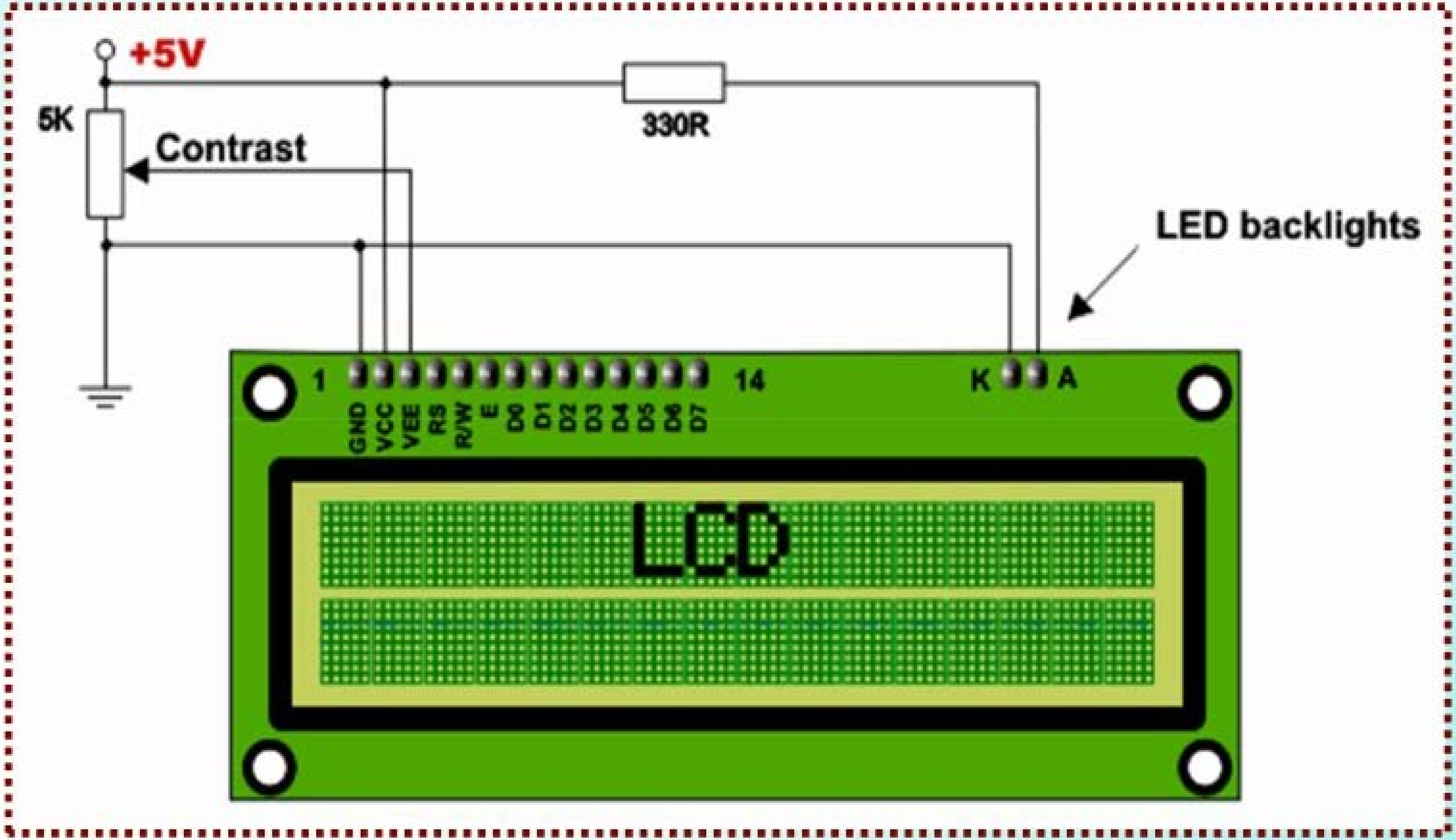
Pin no	Name	Function
1	Vss	Ground
2	Vdd	+V supply
3	Vee	Contrast adjustment
4	RS	Register select
5	R/W	Read/write
6	E	Enable (clock)
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7
15 (optional)	B+	Backlight +
16 (optional)	B−	Backlight −



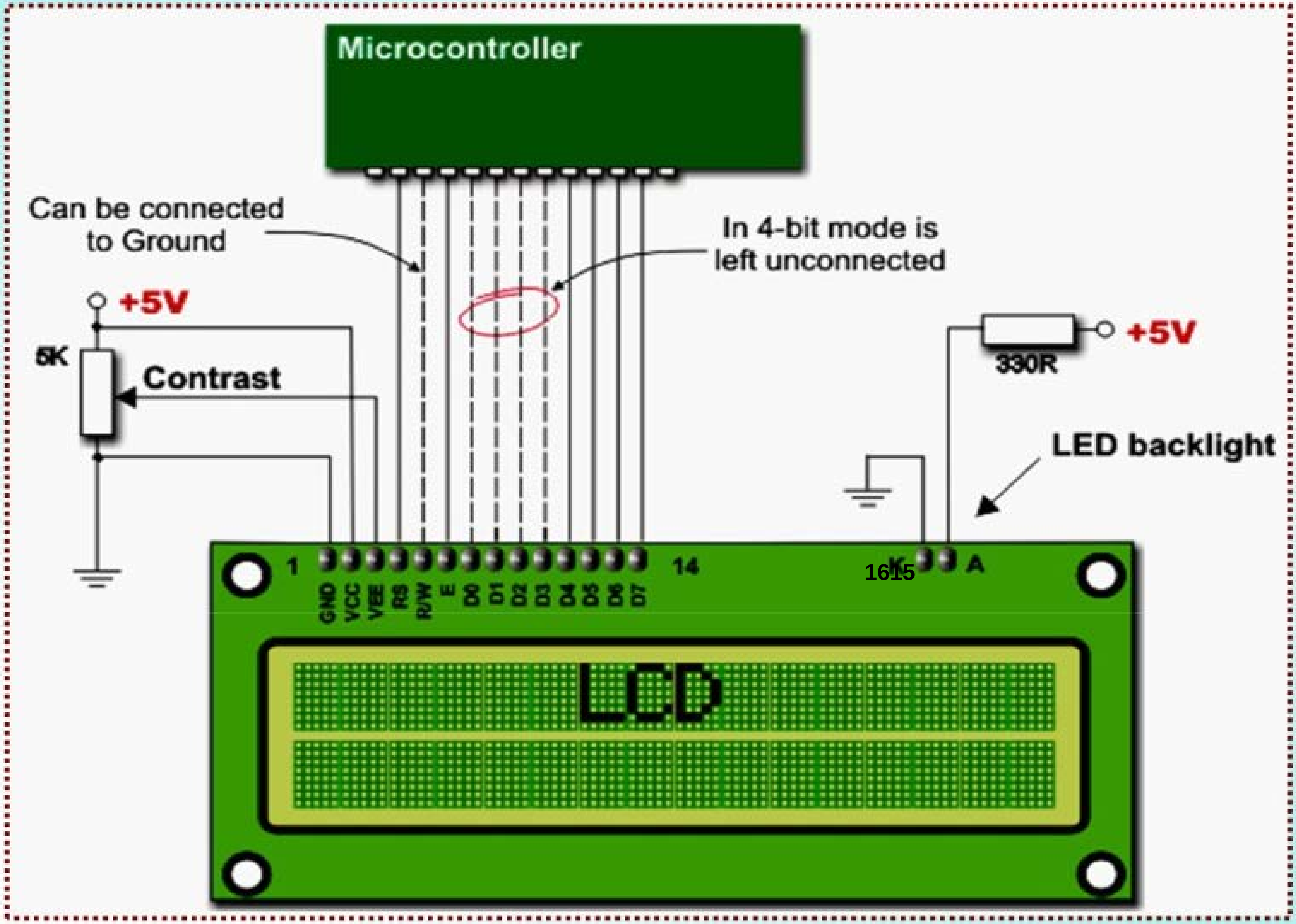
Connecting LCD to a PIC microcontoller (4-bit Interface)

Display Contrast & Led Backlight

- **Some of the LCD displays have built-in backlight (blue or green LEDs).**



LCD Connecting



MikroC LCD Functions

- The library functions are based on using 4-bit interface.
- ***LCD Library (4-bit interface):***

Lcd_Init
Lcd_Out
Lcd_Out_Cp
Lcd_Chr
Lcd_Chr_Cp
Lcd_Cmd

MikroC LCD Functions

- **Lcd_Init( )**
  - This function initializes the LCD module and **it must be called before calling the other LCD functions.**
  - The function is called with no arguments.
  - Before this function is called, **the interface between the microcontroller and the LCD must be defined** using statements of the following format:

```
//Lcd pinout settings
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;

//Pin direction
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
```

This configuration assumes that the connection between the LCD and the microcontroller is as follows:

LCD	**Microcontroller Port**
RS	RB4
EN	RB5
D7	RB3
D6	RB2
D5	RB1
D4	RB0

Example call: Lcd_Init();

MikroC LCD Functions

- **Lcd_Out( )**
  - This function displays **text/string** on the LCD starting from specified row and column positions. **This function receives a string (not number).**
  - **Example call:** Lcd_Out(1, 3, “Hello”); //Display text “Hello” at row 1, column 3

- **Lcd_Out_Cp( )**
  - This function displays **text/string** at **the current cursor position**. **This function receives a string (not number).**
  - **Example call:** Lcd_Out_Cp(“Hello”); //Display text “Hello” at current position

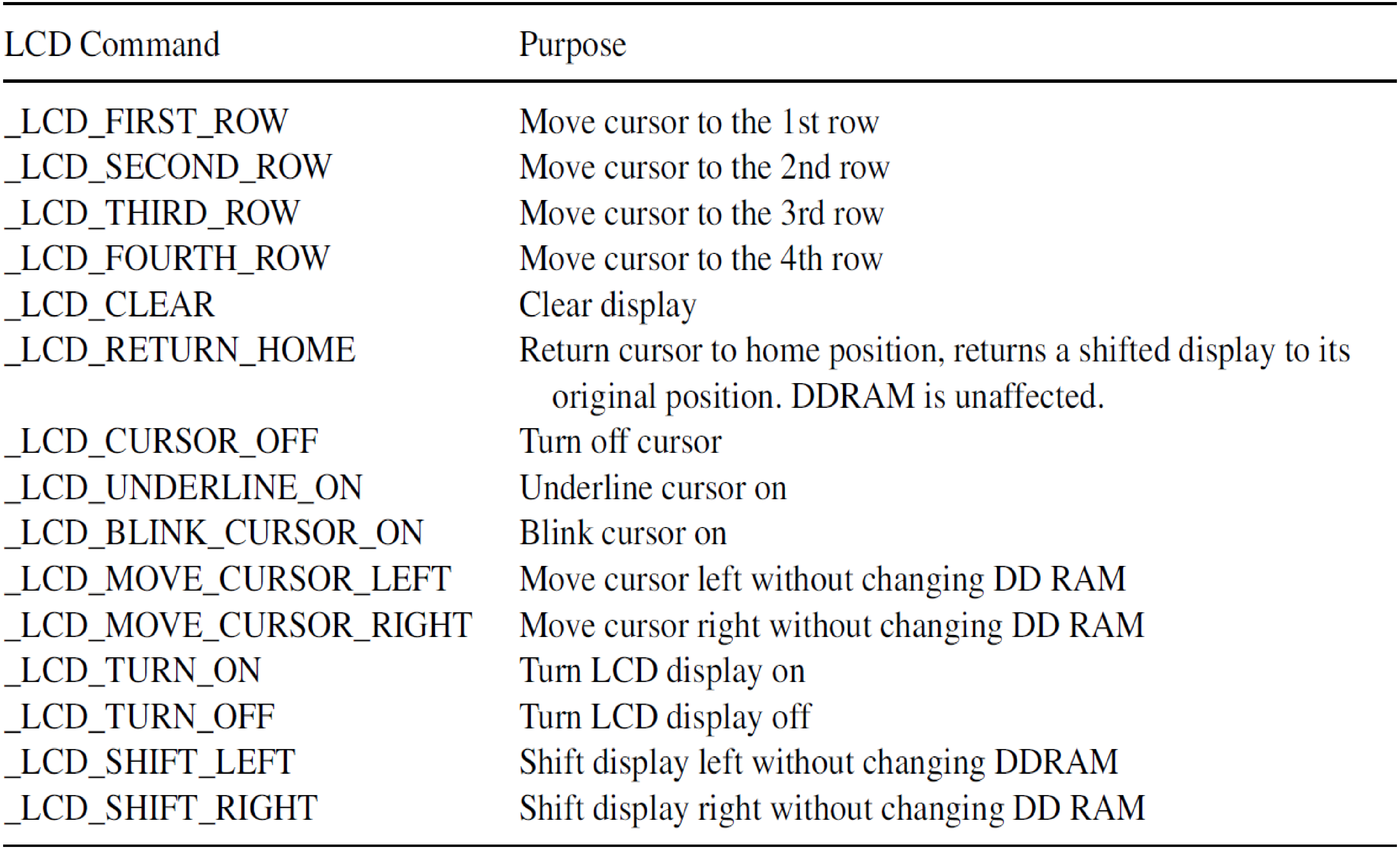
- **Lcd_Chr( )**
  - This function displays **a single character** at the specified row and column positions.
  - **Example call:** Lcd_Chr(1, 2, ‘X’); //Display character “X” at row 1, column 2

MikroC LCD Functions

- **Lcd_Chr_Cp( )**
  - This function displays **a single character** at **the current cursor position**.
  - **Example call**: Lcd_Chr_Cp(‘X’);   //Display character “X” at current position

- **Lcd_Cmd( )**
  - This function **sends a command** to the LCD. A list of the valid commands is given in the next slide.
  - **Example call:** Lcd_Cmd(_LCD_CLEAR);   //Clear display

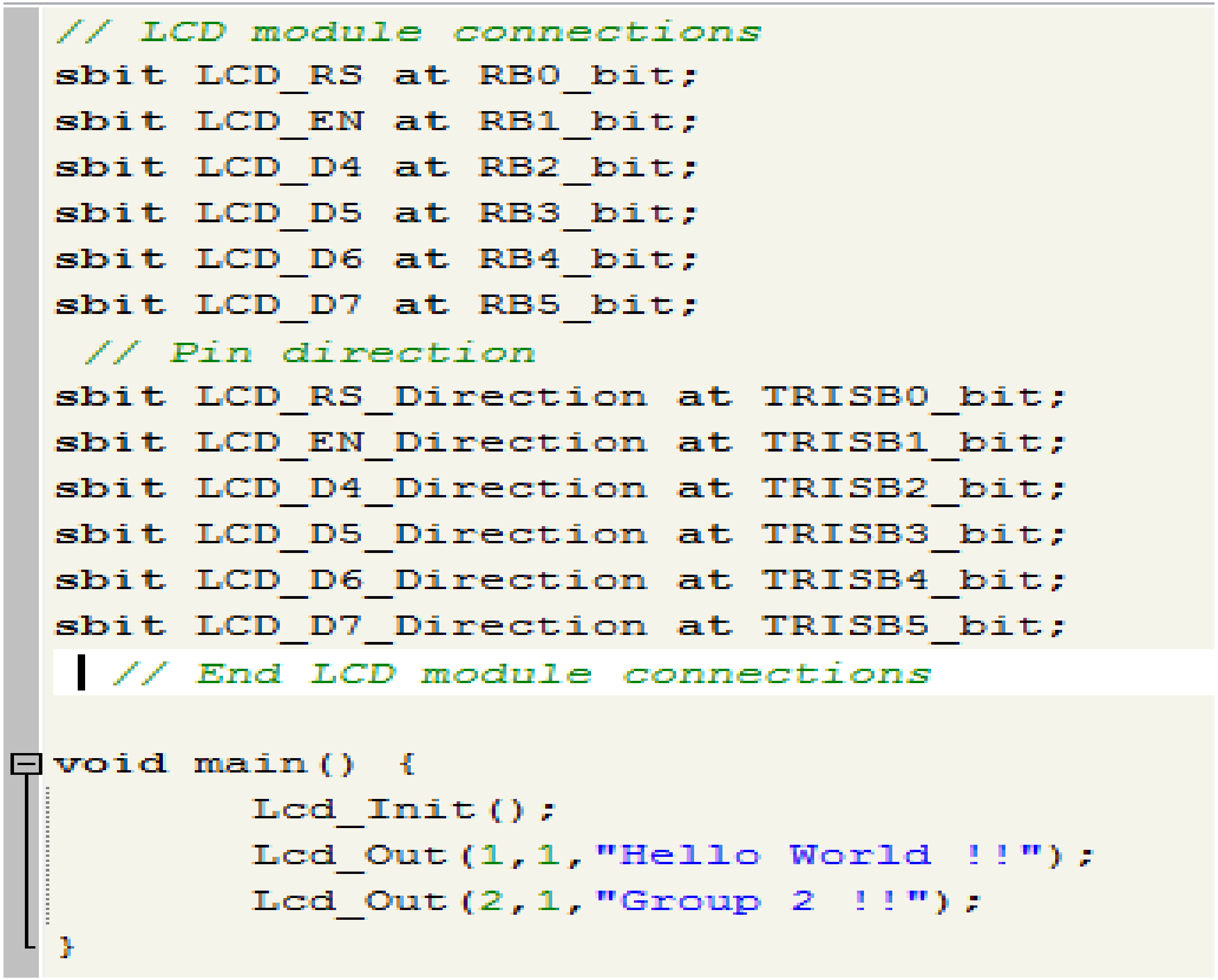
MikroC LCD Functions



LCD Command	Purpose
_LCD_FIRST_ROW	Move cursor to the 1st row
_LCD_SECOND_ROW	Move cursor to the 2nd row
_LCD_THIRD_ROW	Move cursor to the 3rd row
_LCD_FOURTH_ROW	Move cursor to the 4th row
_LCD_CLEAR	Clear display
_LCD_RETURN_HOME	Return cursor to home position, returns a shifted display to its original position. DDRAM is unaffected.
_LCD_CURSOR_OFF	Turn off cursor
_LCD_UNDERLINE_ON	Underline cursor on
_LCD_BLINK_CURSOR_ON	Blink cursor on
_LCD_MOVE_CURSOR_LEFT	Move cursor left without changing DD RAM
_LCD_MOVE_CURSOR_RIGHT	Move cursor right without changing DD RAM
_LCD_TURN_ON	Turn LCD display on
_LCD_TURN_OFF	Turn LCD display off
_LCD_SHIFT_LEFT	Shift display left without changing DDRAM
_LCD_SHIFT_RIGHT	Shift display right without changing DD RAM

Display Data RAM (DDRAM): The data displayed currently by the LCD is stored in the DDRAM.

Code in MikroC

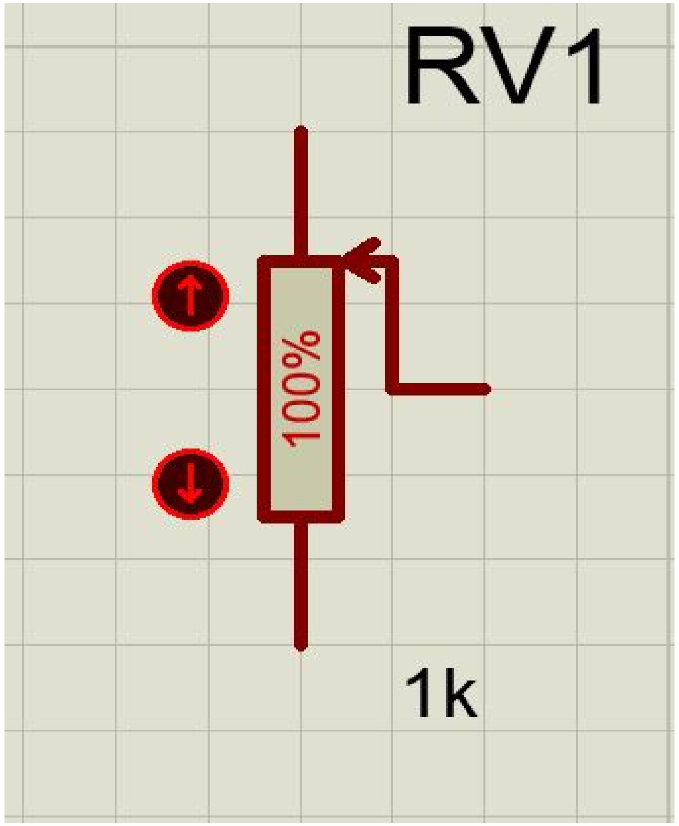


```c
// LCD module connections
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB2_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D7 at RB5_bit;
 // Pin direction
sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit;
  // End LCD module connections

void main() {
       Lcd_Init();
       Lcd_Out(1,1,"Hello World !!");
       Lcd_Out(2,1,"Group 2 !!");
}
```

POT-HG VARIABLE RESISTOR in Proteus

POT-HG is the only active variable resistor that allows you to change the resistance during simulation run-time



EX1: Writing Text on the LCD

- Write a microcontroller PIC16F877A C program to write on a LCD display the “Hello” text in the first row and starting from the first character and the “Engineers” text in the second row and starting from the first character.

```
 sbit LCD_RS at RD2_bit;
 sbit LCD_EN at RD3_bit;
 sbit LCD_D4 at RD4_bit;
 sbit LCD_D5 at RD5_bit;
 sbit LCD_D6 at RD6_bit;
 sbit LCD_D7 at RD7_bit;

 sbit LCD_RS_Direction at TRISD2_bit;
 sbit LCD_EN_Direction at TRISD3_bit;
 sbit LCD_D4_Direction at TRISD4_bit;
 sbit LCD_D5_Direction at TRISD5_bit;
 sbit LCD_D6_Direction at TRISD6_bit;
 sbit LCD_D7_Direction at TRISD7_bit;

 void main( ) {
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);

  Lcd_Out(1,1,“Hello");
Lcd_Out(2,1,"Engineers");
 }
```

EX2: Moving Text

- Write a microcontroller PIC16F877A C program to move a text “Hello” on a LCD display from left to right with delay 50 ms.

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void main( ) {
  int i ;
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);
   while(1) {
     for(i=1; i<=16 ; i++)   {
        Lcd_Out(1,i,“Hello");
     delay_ms(50);
        Lcd_Cmd(_LCD_CLEAR);
     }
    }
}
```

EX3: Display a Counter

- Write a microcontroller PIC16F877A C program to display a counter from 1 to 10 on a LCD display with delay 250 ms.
- **IntToStr(Signed_integer_number,Destination_string)**: converts input **signed integer** number [-32768 .. 32767] to a string.
  - The output string has fixed width of 7 characters including null character at the end (string termination).
  - The output string is **right justified** and the remaining positions on the left (if any) are filled with **blanks**.
  - **Destination_string** should be at least **7** characters in length.
  - Ex. : intToStr(12,str) → “$_{\text{bbbb}}$12”

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void main( ) {
  int  i;
  char  str[7];   // For IntToStr(number, string) function, destination string should be at least 7 characters in length
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);
  while(1) {
       for(i=1; i<=10 ; i++)  {
           Lcd_out(1,1,"i=");
           IntToStr(i,str);
           Lcd_Out_CP(str);
           delay_ms(250);
       }
   }
}
```

EX4: Displaying a Seconds Counter

- Write a microcontroller PIC16F877A C program to make a seconds counter (0-59) with LCD display
- Assume (1:256) Prescaler is assigned to TMR0 and f = 4 MHz.

The initial value of TMR0 =0 (T0 =0)

- overflow Time of TMR0 = (256-T0) x prescale x 4 x clock_period

  $= 256 \times 256 \times 4 \times 0.25*10^{-6}$ sec = 65536 μSec

$$1 \text{ overflow} \longrightarrow 65536\ \mu\text{Sec}$$

$$X \text{ overflows} \longrightarrow 1 \text{ Sec}$$

$$X = \frac{1}{65536 * 10^{-6}} = 15.2587 = 16$$

No. of required overflows to get time equal to 1 sec = 16 overflows.

```
int overflows_no = 0;
int seconds = 0;

void interrupt( ){
    if(INTCON.TMR0IF == 1){
      overflows_no++;
      INTCON.TMR0IF = 0;
      TMR0 = 0;
    }
    if (overflows_no == 16){ //  1 sec delay
        seconds = (seconds+1) % 60;
        overflows_no = 0;
    }
}

void main( ){
     char str[7];
     OPTION_REG = 0x07;
     INTCON.GIE = 1;
     INTCON.T0IE = 1;
     TMR0 = 0;      // initial value (T0) of timer TMR0

     Lcd_Init();
     Lcd_Cmd(_LCD_CURSOR_OFF);
     while(1) {
        IntToStr(seconds,str);
         Lcd_Out (1,1,str);
     }
}
```

Another Solution

```
int overflows_no = 0 ;
int seconds = -1 ; // when interrupt( ) function is called for the first time, the seconds variable is increased and become 0

void interrupt( ){
      char str[7];
      if(INTCON.TMR0IF == 1){
         overflows_no++;
         INTCON.TMR0IF = 0;
         TMR0 = 0;
      }
      if (overflows_no == 16){ //  1 sec delay
           seconds = (seconds+1) % 60;
           overflows_no = 0;

           Lcd_Cmd(_LCD_CURSOR_OFF);
           IntToStr(seconds,str);
            Lcd_Out (1,1,str);
      }
}

void main( ){
       OPTION_REG = 0x07;
       INTCON.GIE = 1;
       INTCON.T0IE = 1;
       TMR0 = 0;        // initial value (T0) of timer TMR0

       Lcd_Init();
}
```

EX5: Blinking Text

- Write a microcontroller PIC16F877A C program to blink a text “Hi Engineers” on a LCD display with delay 250 ms.

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void main() {
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);


    while(1) {
           Lcd_out(1,1,"Hi Engineers");
         delay_ms(250);
           Lcd_Cmd(_LCD_CLEAR);
             delay_ms(250);
     }
}
```