Lcd Library

The mikroPascal PRO for PIC provides a library for communication with Lcds (with HD44780 compliant controllers) through the 4-bit interface. An example of Lcd connections is g schematic at the bottom of this page.

For creating a set of custom Lcd characters use Lcd Custom Character Tool.

Library Dependency Tree

External dependencies of Lcd Library

The following variables must be defined in all projects using Lcd Library :	Description :	Example :
`var LCD_RS : sbit; sfr; external;`	Register Select line.	`var LCD_RS : sbit at RB4_bit;`
`var LCD_EN : sbit; sfr; external;`	Enable line.	`var LCD_EN : sbit at RB5_bit;`
`var LCD_D7 : sbit; sfr; external;`	Data 7 line.	`var LCD_D7 : sbit at RB3_bit;`
`var LCD_D6 : sbit; sfr; external;`	Data 6 line.	`var LCD_D6 : sbit at RB2_bit;`
`var LCD_D5 : sbit; sfr; external;`	Data 5 line.	`var LCD_D5 : sbit at RB1_bit;`
`var LCD_D4 : sbit; sfr; external;`	Data 4 line.	`var LCD_D4 : sbit at RB0_bit;`
`var LCD_RS_Direction : sbit; sfr; external;`	Register Select direction pin.	`var LCD_RS_Direction : sbit at TRISB4_bit;`
`var LCD_EN_Direction : sbit; sfr; external;`	Enable direction pin.	`var LCD_EN_Direction : sbit at TRISB5_bit;`
`var LCD_D7_Direction : sbit; sfr; external;`	Data 7 direction pin.	`var LCD_D7_Direction : sbit at TRISB3_bit;`
`var LCD_D6_Direction : sbit; sfr; external;`	Data 6 direction pin.	`var LCD_D6_Direction : sbit at TRISB2_bit;`
`var LCD_D5_Direction : sbit; sfr; external;`	Data 5 direction pin.	`var LCD_D5_Direction : sbit at TRISB1_bit;`
`var LCD_D4_Direction : sbit; sfr; external;`	Data 4 direction pin.	`var LCD_D4_Direction : sbit at TRISB0_bit;`

Library Routines

- Lcd_Init
- Lcd_Out
- Lcd_Out_Cp
- Lcd_Chr
- Lcd_Chr_Cp
- Lcd_Cmd

Lcd_Init

Prototype	`procedure Lcd_Init();`
Returns	Nothing.
Description	Initializes Lcd module.
Requires	Global variables: - `LCD_D7`: Data bit 7 - `LCD_D6`: Data bit 6 - `LCD_D5`: Data bit 5 - `LCD_D4`: Data bit 4 - `LCD_RS`: Register Select (data/instruction) signal pin - `LCD_EN`: Enable signal pin - `LCD_D7_Direction`: Direction of the Data 7 pin - `LCD_D6_Direction`: Direction of the Data 6 pin - `LCD_D5_Direction`: Direction of the Data 5 pin - `LCD_D4_Direction`: Direction of the Data 4 pin - `LCD_RS_Direction`: Direction of the Register Select pin - `LCD_EN_Direction`: Direction of the Enable signal pin must be defined before using this function.

Example	`// Lcd module connections`
	`var LCD_RS : sbit at RB4_bit;`
	`var LCD_EN : sbit at RB5_bit;`
	`var LCD_D4 : sbit at RB3_bit;`
	`var LCD_D5 : sbit at RB2_bit;`
	`var LCD_D6 : sbit at RB1_bit;`
	`var LCD_D7 : sbit at RB0_bit;`
	
	`var LCD_RS_Direction : sbit at TRISB4_bit;`
	`var LCD_EN_Direction : sbit at TRISB5_bit;`
	`var LCD_D4_Direction : sbit at TRISB3_bit;`
	`var LCD_D5_Direction : sbit at TRISB2_bit;`
	`var LCD_D6_Direction : sbit at TRISB1_bit;`
	`var LCD_D7_Direction : sbit at TRISB0_bit;`
	`// End Lcd module connections`
	
	`...`
	`Lcd_Init();`

Lcd_Out

Prototype	`procedure Lcd_Out(row: byte; column: byte; var text: string);`
Returns	Nothing.
Description	Prints text on Lcd starting from specified position. Both string variables and literals can be passed as a text. Parameters : ▪ `row:` starting position row number ▪ `column:` starting position column number ▪ `text:` text to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	`// Write text "Hello!" on Lcd starting from row 1, column 3:` `Lcd_Out(1, 3, "Hello!");`

Lcd_Out_CP

Prototype	`procedure Lcd_Out_CP(var text: string);`
Returns	Nothing.
Description	Prints text on Lcd at current cursor position. Both string variables and literals can be passed as a text. Parameters : ▪ `text:` text to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	`// Write text "Here!" at current cursor position:` `Lcd_Out_CP("Here!");`

Lcd_Chr

Prototype	`procedure Lcd_Chr(row: byte; column: byte; out_char: byte);`
Returns	Nothing.
Description	Prints character on Lcd at specified position. Both variables and literals can be passed as a character. Parameters : ▪ `row:` writing position row number ▪ `column:` writing position column number ▪ `out_char:` character to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	`// Write character "i" at row 2, column 3:` `Lcd_Chr(2, 3, 'i');`

Lcd_Chr_CP

Prototype	`procedure Lcd_Chr_CP(out_char: byte);`
Returns	Nothing.
Description	Prints character on Lcd at current cursor position. Both variables and literals can be passed as a character. Parameters : ▪ `out_char:` character to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	`// Write character "e" at current cursor position:` `Lcd_Chr_CP('e');`

Lcd_Cmd

Prototype	`procedure Lcd_Cmd(out_char: byte);`
Returns	Nothing.
Description	Sends command to Lcd. Parameters : ▪ `out_char:` command to be sent **Note :** Predefined constants can be passed to the function, see Available Lcd Commands.
Requires	The Lcd module needs to be initialized. See Lcd_Init table.
Example	`// Clear Lcd display:`

```
Lcd_Cmd(_LCD_CLEAR);
```

Available Lcd Commands

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. Display data RAM 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 display data RAM
_LCD_MOVE_CURSOR_RIGHT	Move cursor right without changing display data RAM
_LCD_TURN_ON	Turn Lcd display on
_LCD_TURN_OFF	Turn Lcd display off
_LCD_SHIFT_LEFT	Shift display left without changing display data RAM
_LCD_SHIFT_RIGHT	Shift display right without changing display data RAM

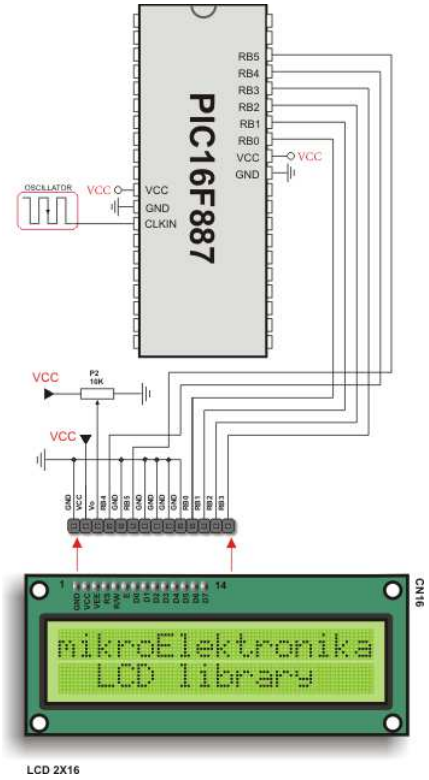
Library Example

The following code demonstrates usage of the Lcd Library routines:

Copy Code To Clipboard

```
program  Lcd_Test;

// LCD module connections
var LCD_RS : sbit at RB4_bit;
var LCD_EN : sbit at RB5_bit;
var LCD_D4 : sbit at RB0_bit;
var LCD_D5 : sbit at RB1_bit;
var LCD_D6 : sbit at RB2_bit;
var LCD_D7 : sbit at RB3_bit;

var LCD_RS_Direction : sbit at TRISB4_bit;
var LCD_EN_Direction : sbit at TRISB5_bit;
var LCD_D4_Direction : sbit at TRISB0_bit;
var LCD_D5_Direction : sbit at TRISB1_bit;
var LCD_D6_Direction : sbit at TRISB2_bit;
var LCD_D7_Direction : sbit at TRISB3_bit;
// End LCD module connections

var txt1 : array[16] of char;
    txt2 : array[9]  of char;
    txt3 : array[8]  of char;
    txt4 : array[7]  of char;
    i    : byte;                       // Loop variable

procedure Move_Delay();                // Function used for text moving
  begin
    Delay_ms(500);                     // You can change the moving speed here
  end;

  begin

    TRISB := 0;
    PORTB := 0xFF;
    TRISB := 0xFF;
    ANSEL  := 0;                        // Configure AN pins as digital I/O
    ANSELH := 0;

    txt1 := 'mikroElektronika';
    txt2 := 'EasyPIC6';
    txt3 := 'Lcd4bit';
    txt4 := 'example';

    Lcd_Init();                        // Initialize LCD
    Lcd_Cmd(_LCD_CLEAR);               // Clear display
    Lcd_Cmd(_LCD_CURSOR_OFF);          // Cursor off
    LCD_Out(1,6,txt3);                 // Write text in first row
    LCD_Out(2,6,txt4);                 // Write text in second row
    Delay_ms(2000);
    Lcd_Cmd(_LCD_CLEAR);               // Clear display

    LCD_Out(1,1,txt1);                 // Write text in first row
    Lcd_Out(2,5,txt2);                 // Write text in second row
    Delay_ms(500);

    // Moving text
    for i:=0 to 3 do                   // Move text to the right 4 times
      begin
        Lcd_Cmd(_LCD_SHIFT_RIGHT);
        Move_Delay();
      end;
```

```
  while TRUE do                       // Endless loop
    begin
      for i:=0 to 7 do                // Move text to the left 8 times
        begin
          Lcd_Cmd(_LCD_SHIFT_LEFT);
          Move_Delay();
        end;

      for i:=0 to 7 do                // Move text to the right 8 times
        begin
          Lcd_Cmd(_LCD_SHIFT_RIGHT);
          Move_Delay();
        end;

    end;
end.
```



Lcd HW connection

What do you think about this topic ? Send us feedback!

Want more exampl
Find them o