

Lcd Library

mikroC PRO for PIC Libraries > Hardware Libraries >



Lcd Library

The mikroC PRO for PIC provides a library for communication with Lcds (with HD44780 compliant cc given on the 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 :	Examp
<code>extern sfr sbit LCD_RS;</code>	Register Select line.	<code>sbit LCI</code>
<code>extern sfr sbit LCD_EN;</code>	Enable line.	<code>sbit LCI</code>
<code>extern sfr sbit LCD_D7;</code>	Data 7 line.	<code>sbit LCI</code>
<code>extern sfr sbit LCD_D6;</code>	Data 6 line.	<code>sbit LCI</code>
<code>extern sfr sbit LCD_D5;</code>	Data 5 line.	<code>sbit LCI</code>
<code>extern sfr sbit LCD_D4;</code>	Data 4 line.	<code>sbit LCI</code>
<code>extern sfr sbit LCD_RS_Direction;</code>	Register Select direction pin.	<code>sbit LCI</code> <code>TRISB4_</code>
<code>extern sfr sbit LCD_EN_Direction;</code>	Enable direction pin.	<code>sbit LCI</code> <code>TRISB5_</code>
<code>extern sfr sbit LCD_D7_Direction;</code>	Data 7 direction pin.	<code>sbit LCI</code> <code>TRISB3_</code>
<code>extern sfr sbit LCD_D6_Direction;</code>	Data 6 direction pin.	<code>sbit LCI</code> <code>TRISB2_</code>
<code>extern sfr sbit LCD_D5_Direction;</code>	Data 5 direction pin.	<code>sbit LCI</code> <code>TRISB1_</code>
<code>extern sfr sbit LCD_D4_Direction;</code>	Data 4 direction pin.	<code>sbit LCI</code> <code>TRISB0_</code>

Library Routines

- [Lcd_Init](#)
- [Lcd_Out](#)
- [Lcd_Out_Cp](#)
- [Lcd_Chr](#)
- [Lcd_Chr_Cp](#)
- [Lcd_Cmd](#)

Lcd_Init

Prototype	<code>void Lcd_Init();</code>
Returns	Nothing.
Description	Initializes Lcd module.
Requires	<p>Global variables:</p> <ul style="list-style-type: none"> ▪ <code>LCD_D7</code>: Data bit 7 ▪ <code>LCD_D6</code>: Data bit 6 ▪ <code>LCD_D5</code>: Data bit 5 ▪ <code>LCD_D4</code>: Data bit 4 ▪ <code>LCD_RS</code>: Register Select (data/instruction) signal pin ▪ <code>LCD_EN</code>: Enable signal pin ▪ <code>LCD_D7_Direction</code>: Direction of the Data 7 pin ▪ <code>LCD_D6_Direction</code>: Direction of the Data 6 pin ▪ <code>LCD_D5_Direction</code>: Direction of the Data 5 pin ▪ <code>LCD_D4_Direction</code>: Direction of the Data 4 pin ▪ <code>LCD_RS_Direction</code>: Direction of the Register Select pin ▪ <code>LCD_EN_Direction</code>: Direction of the Enable signal pin <p>must be defined before using this function.</p>
Example	<pre>// 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;</pre>

```

sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
...

Lcd_Init();

```

Lcd_Out

Prototype	void Lcd_Out(char row, char column, char *text);
Returns	Nothing.
Description	Prints text on Lcd starting from specified position. Both string variables and literals can be used. Parameters : <ul style="list-style-type: none"> ▪ 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	<i>// Write text "Hello!" on Lcd starting from row 1, column 3:</i> Lcd_Out(1, 3, "Hello!");

Lcd_Out_Cp

Prototype	void Lcd_Out_Cp(char *text);
Returns	Nothing.
Description	Prints text on Lcd at current cursor position. Both string variables and literals can be used. Parameters : <ul style="list-style-type: none"> ▪ text: text to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	<i>// Write text "Here!" at current cursor position:</i> Lcd_Out_Cp("Here!");

Lcd_Chr


Prototype	void Lcd_Chr(char row, char column, char out_char);
------------------	--

Returns	Nothing.
Description	<p>Prints character on Lcd at specified position. Both variables and literals can be pa</p> <p>Parameters :</p> <ul style="list-style-type: none"> ▪ <code>row</code>: writing position row number ▪ <code>column</code>: writing position column number ▪ <code>out_char</code>: character to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	<pre>// Write character "i" at row 2, column 3: Lcd_Chr(2, 3, 'i');</pre>

Lcd_Chr_Cp

Prototype	<code>void Lcd_Chr_Cp(char out_char);</code>
Returns	Nothing.
Description	<p>Prints character on Lcd at current cursor position. Both variables and literals can</p> <p>Parameters :</p> <ul style="list-style-type: none"> ▪ <code>out_char</code>: character to be written
Requires	The Lcd module needs to be initialized. See Lcd_Init routine.
Example	<pre>// Write character "e" at current cursor position: Lcd_Chr_Cp('e');</pre>

Lcd_Cmd

Prototype	<code>void Lcd_Cmd(char out_char);</code>
Returns	Nothing.
Description	<p>Sends command to Lcd.</p> <p>Parameters :</p> <ul style="list-style-type: none"> ▪ <code>out_char</code>: command to be sent <div>  Note : Predefined constants can be passed to the function, see Available I </div>

Requires	The Lcd module needs to be initialized. See Lcd_Init table.
Example	<pre>// Clear Lcd display: Lcd_Cmd(_LCD_CLEAR);</pre>

Available Lcd Commands

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

Library Example

The following code demonstrates usage of the Lcd Library routines:



Copy Code To Clipboard

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

sbit LCD_RS_Direction at TRISB4_bit;
```

```

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

char txt1[] = "mikroElektronika";
char txt2[] = "EasyPIC6";
char txt3[] = "Lcd4bit";
char txt4[] = "example";

char i;                                // Loop variable

void Move_Delay() {                    // Function used for text moving
    Delay_ms(500);                      // You can change the moving speed here
}

void main(){
    ANSEL = 0;                          // Configure AN pins as digital I/O
    ANSELH = 0;
    C1ON_bit = 0;                       // Disable comparators
    C2ON_bit = 0;

    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(2000);

    // Moving text
    for(i=0; i<4; i++) {                 // Move text to the right 4 times
        Lcd_Cmd(_LCD_SHIFT_RIGHT);
        Move_Delay();
    }

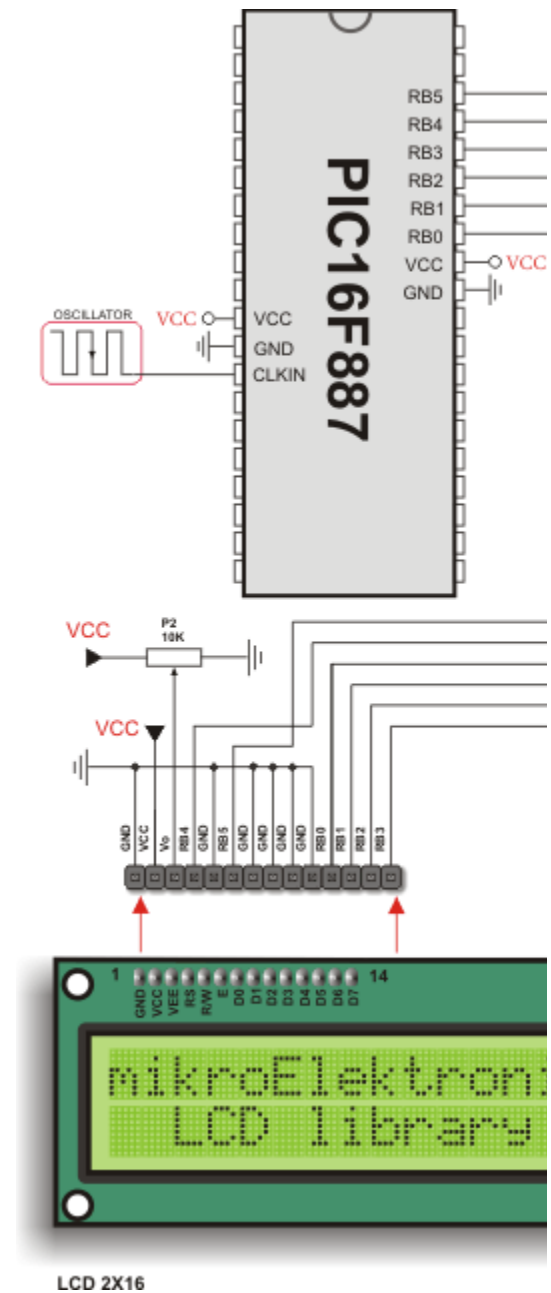
    while(1) {                           // Endless loop
        for(i=0; i<8; i++) {             // Move text to the left 7 times
            Lcd_Cmd(_LCD_SHIFT_LEFT);
            Move_Delay();
        }
    }
}

```

```

for(i=0; i<8; i++) {           // Move text to the right 7 times
    Lcd_Cmd(_LCD_SHIFT_RIGHT);
    Move_Delay();
}
}
}

```



Lcd HW connection

Copyright (c) 2002-2010 mikroElektronika. All rights reserved.
What do you think about this topic ? [Send us](#)