Lcd Library

The mikroC PRO for PIC provides a library for communication with Lcds (with HD44780 compliant controllers) through the 4-bit interface. An example of Lcd con

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 :
`extern sfr sbit LCD_RS;`	Register Select line.
`extern sfr sbit LCD_EN;`	Enable line.
`extern sfr sbit LCD_D7;`	Data 7 line.
`extern sfr sbit LCD_D6;`	Data 6 line.
`extern sfr sbit LCD_D5;`	Data 5 line.
`extern sfr sbit LCD_D4;`	Data 4 line.
`extern sfr sbit LCD_RS_Direction;`	Register Select direction pin.
`extern sfr sbit LCD_EN_Direction;`	Enable direction pin.
`extern sfr sbit LCD_D7_Direction;`	Data 7 direction pin.
`extern sfr sbit LCD_D6_Direction;`	Data 6 direction pin.
`extern sfr sbit LCD_D5_Direction;`	Data 5 direction pin.
`extern sfr sbit LCD_D4_Direction;`	Data 4 direction pin.

Library Routines

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

Lcd_Init

Prototype	`void 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 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;`

```
...

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 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	`void Lcd_Out_Cp(char *text);`
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	`void Lcd_Chr(char row, char column, char out_char);`
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	`void Lcd_Chr_Cp(char out_char);`
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	`void Lcd_Cmd(char out_char);`
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

```
// 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();
    }
  }
}
```