

## **LCD5110\_Basic - Arduino library support for Nokia 5110 compatible LCDs**

Copyright (C)2011 Henning Karlsen. All right reserved

Basic functionality of this library are based on the demo-code provided by ITEad studio. You can find the latest version of the library at <http://www.henningkarlsen.com/electronics>

This library has been made to make it easy to use the basic functions of the Nokia 5110 LCD module on an Arduino.

If you make any modifications or improvements to the code, I would appreciate that you share the code with me so that I might include it in the next release. I can be contacted through <http://www.henningkarlsen.com/electronics/contact.php>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.


You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA


Version:	1.0	12 Aug 2011	• initial release
	1.1	04 Sep 2011	• Added invertText();


## Defined Literals:

Alignment
For use with print(), printNumI() and printNumF()
LEFT: 0
RIGHT: 9999
CENTER: 9998

## Included Fonts:

SmallFont
 A 6x8 pixel font character set showing digits 0-9, uppercase and lowercase letters, and various symbols.
Character size: 6x8 pixels
Number of characters: 95

MediumNumbers
 A 12x16 pixel font character set showing digits 0-9, a decimal point, and a minus sign.
Character size: 12x16 pixels
Number of characters: 13

BigNumbers
 A 14x24 pixel font character set showing digits 0-9, a decimal point, and a minus sign.
Character size: 14x24 pixels
Number of characters: 13

## Functions:

LCD5110(SCK, MOSI, DC, RST, CS);	
Class constructor.	
Parameters:	SCK: Arduino pin for Clock signal MOSI: Arduino pin for Data transfer DC: Arduino pin for Register Select (Data/Command) RST: Arduino pin for Reset CS: Arduino pin for Chip Select
Usage:	LCD5110 myGLCD(8, 9, 10, 11, 12); // Start an instance of the LCD5110 class
InitLCD();	
Initialize the LCD.	
Parameters:	None
Usage:	myGLCD.initLCD(); // Initialize the display
Notes:	This will reset and clear the display.
clrScr();	
Clear the screen.	
Parameters:	None
Usage:	myGLCD.clrScr(); // Clear the screen
clrRow(row[, start_x[, end_x]]);	
Clear a part of, or a whole row.	
Parameters:	row: 8 pixel high row to clear (0-5) start_x: <optional> x-coordinate to start the clearing on (default = 0) end_x: <optional> x-coordinate to end the clearing on (default = 83)
Usage:	myGLCD.clrRow(5, 42); // Clear the right half of the lower row
invert(mode);	
Set inversion of the display on or off.	
Parameters:	mode: true - Invert the display false - Normal display
Usage:	myGLCD.invert(true); // Set display inversion on
invertText(mode);	
Select if text printed with print(), printNumI() and printNumF() should be inverted.	
Parameters:	mode: true - Invert the text false - Normal text
Usage:	myGLCD.invertText(true); // Turn on inverted printing
Notes:	SetFont() will turn off inverted printing
print(st, x, y);	
Print a string at the specified coordinates. You can use the literals LEFT, CENTER and RIGHT as the x-coordinate to align the string on the screen.	
Parameters:	st: the string to print x: x-coordinate of the upper, left corner of the first character y: y-coordinate of the upper, left corner of the first character
Usage:	myGLCD.print("Hello World",CENTER,0); // Print "Hello World" centered at the top of the screen
Notes:	The y-coordinate will be adjusted to be aligned with an 8 pixel high display row. In effect only 0, 8, 16, 24, 32 and 40 can be used as y-coordinates.
printNumI(num, x, y);	
Print an integer number at the specified coordinates. You can use the literals LEFT, CENTER and RIGHT as the x-coordinate to align the string on the screen.	
Parameters:	num: the value to print (-2,147,483,648 to 2,147,483,647) <i>INTEGERS ONLY</i> x: x-coordinate of the upper, left corner of the first digit/sign y: y-coordinate of the upper, left corner of the first digit/sign
Usage:	myGLCD.print(num,CENTER,0); // Print the value of "num" centered at the top of the screen
Notes:	The y-coordinate will be adjusted to be aligned with an 8 pixel high display row. In effect only 0, 8, 16, 24, 32 and 40 can be used as y-coordinates.

#### **printNumF(num, dec, x, y);**

Print a floating-point number at the specified coordinates.

You can use the literals LEFT, CENTER and RIGHT as the x-coordinate to align the string on the screen.

**WARNING:** Floating point numbers are not exact, and may yield strange results when compared. Use at your own discretion.

Parameters:      num: the value to print (*See note*)  
                  dec: digits in the fractional part (1-5) *0 is not supported. Use printNumI() instead.*  
                  x:    x-coordinate of the upper, left corner of the first digit/sign (0-239)  
                  y:    y-coordinate of the upper, left corner of the first digit/sign (0-319)  
Usage:            myGLCD.print(num, 3, CENTER, 0); // Print the value of "num" with 3 fractional digits top centered  
Notes:            Supported range depends on the number of fractional digits used.  
                  Approx range is +/- 2\*(10<sup>9-dec</sup>)  
                  The y-coordinate will be adjusted to be aligned with an 8 pixel high display row.  
                  In effect only 0, 8, 16, 24, 32 and 40 can be used as y-coordinates.

#### **setFont(fontname);**

Select font to use with print(), printNumI() and printNumF().

Parameters:      fontname: Name of the array containing the font you wish to use  
Usage:            myGLCD.setFont(SmallFont); // Select the font called SmallFont  
Notes:            You must declare the font-array as an external or include it in your sketch.

#### **drawBitmap(x, y, sx, sy, data[, flash]);**

Draw a bitmap on the screen.

Parameters:      x:        x-coordinate of the upper, left corner of the bitmap  
                  y:        y-coordinate of the upper, left corner of the bitmap  
                  sx:       width of the bitmap in pixels  
                  sy:       height of the bitmap in pixels  
                  data:     array containing the bitmap-data  
                  flash: <optional>  
                          true    - data-array is in flash memory (Default)  
                          false - data-array is in RAM  
Usage:            myGLCD.drawBitmap(0, 0, 32, 32, bitmap); // Draw a 32x32 pixel bitmap in the upper left corner  
Notes:            You can use the online-tool "[ImageConverter Mono](#)" to convert pictures into compatible arrays.  
                  The online-tool can be found on my website.  
                  Requires that you #include <avr/pgmspace.h>