

LCD5110_Graph - 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 Nokia 5110 LCD module as a graphics display 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	01 Sep 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:

TinyFont
 A 4x6 pixel font showing characters from 0 to 9, uppercase A-Z, lowercase a-z, and various symbols.
Character size: 4x6 pixels
Number of characters: 95

SmallFont
 A 6x8 pixel font showing characters from 0 to 9, uppercase A-Z, lowercase a-z, and various symbols.
Character size: 6x8 pixels
Number of characters: 95

MediumNumbers
 A 12x16 pixel font showing characters from 0 to 9, uppercase A-Z, lowercase a-z, and various symbols.
Character size: 12x16 pixels
Number of characters: 13

BigNumbers
 A 14x24 pixel font showing characters from 0 to 9, uppercase A-Z, lowercase a-z, and various symbols.
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.

update();

Copy the screen buffer to the screen.

This is the only command, except invert(), that will make anything happen on the physical screen. All other commands only modify the screen buffer.

Parameters: None
Usage: myGLCD.update(); // Copy the screen buffer to the screen
Notes: Remember to call update() after you have updated the screen buffer.

clrScr();

Clear the screen buffer.

Parameters: None
Usage: myGLCD.clrScr(); // Clear the screen buffer

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

setPixel(x, y);

Turn on the specified pixel in the screen buffer.

Parameters: x: x-coordinate of the pixel
 y: y-coordinate of the pixel
Usage: myGLCD.setPixel(0, 0); // Turn on the upper left pixel (in the screen buffer)

clrPixel(x, y);

Turn off the specified pixel in the screen buffer.

Parameters: x: x-coordinate of the pixel
 y: y-coordinate of the pixel
Usage: myGLCD.clrPixel(0, 0); // Turn off the upper left pixel (in the screen buffer)

invPixel(x, y);

Invert the state of the specified pixel in the screen buffer.

Parameters: x: x-coordinate of the pixel
 y: y-coordinate of the pixel
Usage: myGLCD.invPixel(0, 0); // Invert the upper left pixel (in the screen buffer)

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 in the screen buffer.

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 (in the screen buffer)

printNumI(num, x, y);

Print an integer number at the specified coordinates in the screen buffer.

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) *INTEGERS ONLY*
 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 (in the screen buffer)

printNumF(num, dec, x, y);

Print a floating-point number at the specified coordinates in the screen buffer.

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 (in the screen buffer)

Notes: Supported range depends on the number of fractional digits used.
 Approx range is +/- 2*(10^(9-dec))

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 in the screen buffer.

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>
 While the bitmap data *MUST* be a multiple of 8 pixels high you do not need to display all the rows.
 Example: If the bitmap is 24 pixels high and you specify sy=20 only the upper 20 rows will be displayed.

drawLine(x1, y1, x2, y2);

Draw a line between two points in the screen buffer.

Parameters: x1: x-coordinate of the start-point
 y1: y-coordinate of the start-point
 x2: x-coordinate of the end-point
 y2: y-coordinate of the end-point

Usage: myGLCD.drawLine(0,0,83,47); // Draw a line from the upper left to the lower right corner

drawRect(x1, y1, x2, y2);

Draw a rectangle between two points in the screen buffer.

Parameters: x1: x-coordinate of the start-corner
 y1: y-coordinate of the start-corner
 x2: x-coordinate of the end-corner
 y2: y-coordinate of the end-corner

Usage: myGLCD.drawRect(42,24,83,47); // Draw a rectangle in the lower right corner of the screen

drawRoundRect(x1, y1, x2, y2);

Draw a rectangle with slightly rounded corners between two points in the screen buffer.
The minimum size is 5 pixels in both directions. If a smaller size is requested the rectangle will not be drawn.

Parameters: x1: x-coordinate of the start-corner
 y1: y-coordinate of the start-corner
 x2: x-coordinate of the end-corner
 y2: y-coordinate of the end-corner

Usage: myGLCD.drawRoundRect(0,0,41,23); // Draw a rounded rectangle in the upper left corner of the screen

drawCircle(x, y, radius);

Draw a circle with a specified radius in the screen buffer.

Parameters: x: x-coordinate of the center of the circle
 y: y-coordinate of the center of the circle
 radius: radius of the circle in pixels

Usage: myGLCD.drawCircle(41,23,20); // Draw a circle in the middle of the screen with a radius of 20 pixels