LCD5110_Basic

Arduino and chipKit library for Nokia 5110 compatible LCDs

Manual

PREFACE:

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

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

You can always find the latest version of the library at http://electronics.henningkarlsen.com/

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://electronics.henningkarlsen.com/contact.php.

For version information, please refer to **version.txt**.

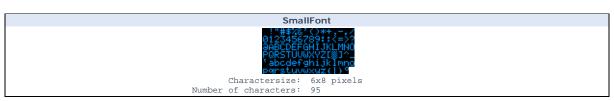
This library is licensed under a **CC BY-NC-SA 3.0** (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported) License.

For more information see: http://creativecommons.org/licenses/by-nc-sa/3.0/

Defined Literals:

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

Included Fonts:







Functions:

```
Initialize the LCD.

Parameters: contrast: <optional>
Specify a value to use for contrast (0-127)
Default is 70

Usage: myGLCD.initLCD(); // Initialize the display
Notes: This will reset and clear the display.
```

setContrast(contrast);		
Set the contrast of the LCD.		
Parameters:	contrast: Specify a value to use for contrast (0-127)	
Usage:	myGLCD.setContrast(70); // Sets the contrast to the default value of 70	

	enableSleep();		
Put the display in Sleep Mode.			
Parameters:	None		
Usage:	myGLCD.enableSleep(); // Put the display into Sleep Mode		
Notes:	Entering Sleep Mode will not turn off the backlight as this is a hardware function.		

	disableSleep();	
Re-enable the display after it has been put in Sleep Mode.		
Parameters:	None	
Usage:	${ t myGLCD.disableSleep();}$ // Wake the display after putting it into Sleep Mode	
Notes:	The display will automatically be cleared when Sleep Mode is disabled. Exiting Sleep Mode will not turn on the backlight as this is a hardware function.	

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

```
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
```

```
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.

The string can be either a char array or a String object
```

```
printNuml(num, x, y[, length[, filler]]);
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) 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
    length: <optional>
        minimum number of digits/characters (including sign) to display
    filler: <optional>
        filler character to use to get the minimum length. The character will be inserted in front of the number, but after the sign. Default is ' ' (space).

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[, divider[, length[, filler]]]);

```
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
y: y-coordinate of the upper, left corner of the first digit/sign
                   divider:
                                 <Optional>
                                 Single character to use as decimal point. Default is '.
                    length:
                                 <optional>
                                 minimum number of digits/characters (including sign) to display
                    filler:
                                 <optional>
                                filler character to use to get the minimum length. The character will be inserted in front of the number, but after the sign. Default is ' ^{\prime} (space).
                   myGLCD.print(num, 3, CENTER,0); // Print the value of "num" with 3 fractional digits top centered
Usage
                    Supported range depends on the number of fractional digits used.
Notes:
                    Approx range is +/- 2*(10^(9-dec))
                    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.
```

```
Select font to use with print(), printNuml() 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.
```

```
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
data: array containing the bitmap-data
sx: width of the bitmap in pixels
sy: height of the bitmap in pixels
sy: height of the bitmap in pixels
Usage: myGLCD.drawBitmap(0, 0, bitmap, 32, 32); // 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> when using an Arduino other than Arduino Due.
```