LCD5110_Graph

Arduino and chipKit library for Nokia 5110 compatible LCDs

Manual

Introduction:

This library has been made to make it easy to use the Nokia 5110 LCD module as a graphics display on an Arduino or a chipKit.

Basic functionality of this library are based on the demo-code provided by ITead studio.

You can always find the latest version of the library at http://www.RinkyDinkElectronics.com/
For version information, please refer to www.RinkyDinkElectronics.com/

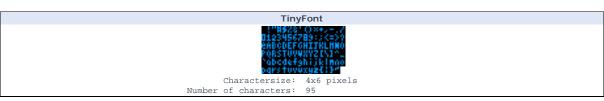
This library is licensed under a ${\tt CC}$ BY-NC-SA 3.0 (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported) License.

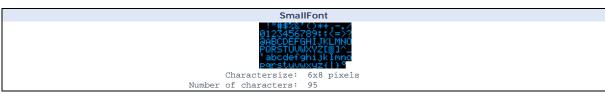
For more information see: http://creativecommons.org/licenses/by-nc-sa/3.0/

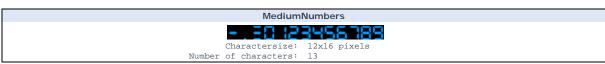
Defined Literals:

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

Included Fonts:









Functions:

LCD5110(SCK, MOSI, DC, RST, CS);

The main class constructor.

SCK: Pin for Clock signal MOSI: Pin for Data transfer

Pin for Register Select (Data/Command) DC:

Pin for Reset cs: Pin for Chip Select

LCD5110 myGLCD(8, 9, 10, 11, 12); // Start an instance of the LCD5110 class

InitLCD([contrast]);

Initialize the LCD.

contrast: <optional> Parameters:

Specify a value to use for contrast (0-127) Default is 70

myGLCD.initLCD(); // Initialize the display 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)

myGLCD.setContrast(70); // Sets the contrast to the default value of 70

enableSleep();

Put the display in Sleep Mode.

Parameters: None

Usage: ${\tt myGLCD.enableSleep();}$ // Put the display into Sleep Mode Notes: update() will not work while the display is in Sleep Mode

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:

Notes:

Usage:

The display will automatically be updated with the contents of the buffer when Sleep Mode is

disabled.

Exiting Sleep Mode will not turn on the backlight as this is a hardware function.

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: ${\tt myGLCD.update();}$ // Copy the screen buffer to the screen

Remember to call update() after you have updated the screen buffer. Notes:

Calling update() while the display is in Sleep Mode will not have any effect

clrScr();

Clear the screen buffer.

Jsage: myGLCD.clrScr(); // Clear the screen buffer

fillScr();

Fill the screen buffer.

arameters: None

Usage: myGLCD.fillScr(); // Fill the screen buffer

invert(mode);

Set inversion of the display on or off.

mode: true - Invert the display false - Normal display Parameters:

myGLCD.invert(true); // Set display inversion on Usage

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

myGLCD.setPixel(0, 0); // Turn on the upper left pixel (in the screen buffer)

cIrPixel(x, y);

Turn off the specified pixel in the screen buffer.

Parameters:

x: x-coordinate of the pixel y: y-coordinate of the pixel

myGLCD.clrPixel(0, 0); // Turn off the upper left pixel (in the screen buffer) Usage

invPixel(x, y);

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

x: x-coordinate of the pixel y: y-coordinate of the pixel

myGLCD.invPixel(0, 0); // Invert the upper left pixel (in the screen buffer)

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-coordinate of the upper, left corner of the first character y-coordinate of the upper, left corner of the first character

myGLCD.print("Hello World",CENTER,0); // Print "Hello World" centered at the top of the screen (in Usage

the screen buffer)

Notes 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 in the screen buffer.

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

num: the value to print (-2,147,483,648 to 2,147,483,647) INTEGERS ONLY x-coordinate of the upper, left corner of the first digit/sign 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 ' ' $(\mbox{\rm space})\,.$

myGLCD.print(num,CENTER,0); // Print the value of "num" centered at the top of the screen (in the Jsage

screen buffer)

printNumF(num, dec, x, y[, divider[, length[, filler]]]);

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-coordinate of the upper, left corner of the first digit/sign y-coordinate of the upper, left corner of the first digit/sign divider: <Optional>

Single character to use as decimal point. Default is '.'

<optional> length:

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)

myGLCD.print(num, 3, CENTER,0); // Print the value of "num" with 3 fractional digits top centered Usage

(in the screen buffer)

Supported range depends on the number of fractional digits used. Approx range is +/- $2*(10^{9-ec})$ Votes

invertText(mode);

Select if text printed with print(), printNumI() and printNumF() should be inverted.

Parameters: mode: true - Invert the text false - Normal text

myGLCD.invertText(true); // Turn on inverted printing Usage

Notes: SetFont() will turn off inverted printing

setFont(fontname):

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

fontname: Name of the array containing the font you wish to use Parameters myGLCD.setFont(SmallFont); // Select the font called SmallFont Usage:

Notes: You must declare the font-array as an external or include it in your sketch.

```
drawLine(x1, y1, x2, y2);
Draw a line between two points in the screen buffer
                   x1: x-coordinate of the start-point
y1: y-coordinate of the start-point
Parameters:
                   x2: x-coordinate of the end-point y2: y-coordinate of the end-point
                    myGLCD.drawLine(0,0,83,47); // Draw a line from the upper left to the lower right corner
```

```
clrLine(x1, y1, x2, y2);
Clear a line between two points in the screen buffer.
                    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
                    myGLCD.clrLine(0,0,83,47); // Clear 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.
                 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
                 myGLCD.drawRect(42,24,83,47); // Draw a rectangle in the lower right corner of the screen
```

```
clrRect(x1, y1, x2, y2);
Clear a rectangle between two points in the screen buffer.
                    x1: x-coordinate of the start-corner
y1: y-coordinate of the start-corner
Parameters:
                    x2: x-coordinate of the end-corner
y2: y-coordinate of the end-corner
Usage
                    myGLCD.clrRect(42,24,83,47); // Clear 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.
                  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
                  myGLCD.drawRoundRect(0,0,41,23); // Draw a rounded rectangle in the upper left corner of the screen
Usage:
```

```
clrRoundRect(x1, y1, x2, y2);
Clear 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/cleared.
                 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
```

```
drawCircle(x, y, radius);
Draw a circle with a specified radius in the screen buffer.
                              x-coordinate of the center of the circle y-coordinate of the center of the circle
Parameters
                    radius: radius of the circle in pixels
```

 ${\tt myGLCD.clrRoundRect(0,0,41,23);} \ // \ {\tt Clear} \ a \ {\tt rounded} \ {\tt rectangle} \ {\tt in} \ {\tt the} \ {\tt upper} \ {\tt left} \ {\tt corner} \ {\tt of} \ {\tt the} \ {\tt screen}$

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

```
clrCircle(x, y, radius);
Clear a circle with a specified radius in the screen buffer.
                     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
Parameters:
                     myGLCD.clrCircle(41,23,20); // Clear a circle in the middle of the screen with a radius of 20 pixels
```

Jsage

Usage:

Usage

drawBitmap (x, y, data, sx, sy);

Draw a bitmap in the screen buffer.

Parameters:

 $x\hbox{-coordinate}$ of the upper, left corner of the bitmap y-coordinate of the upper, left corner of the bitmap

array containing the bitmap-data width of the bitmap in pixels data: sx: sy: height of the bitmap in pixels

 ${\tt myGLCD.drawBitmap(0, 0, bitmap, 32, 32); // Draw a 32x32 pixel bitmap in the upper left corner}$ Notes:

myGCD.Grawbitmap(0, 0, Bitmap, 32, 32); // Braw a 32x32 pixel Bitmap in the upper left corner
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.
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.