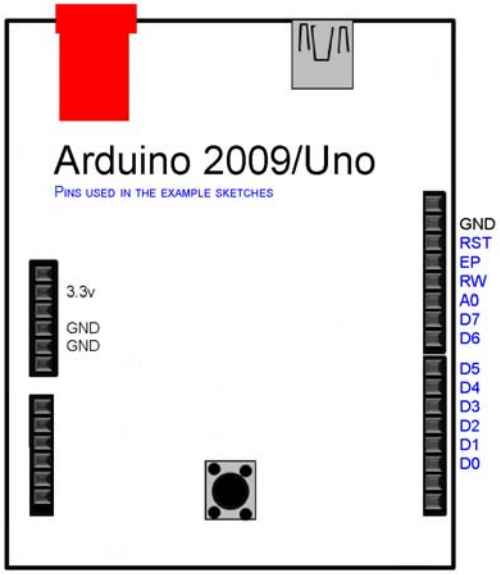
MGLCD - Arduino library support for Monochrome Graphics LCDs

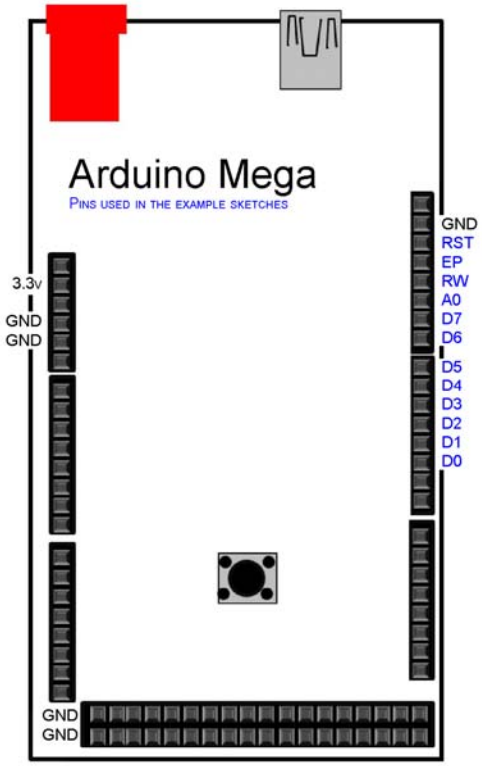
Basic functionality of this library are based on the demo-code provided by ElecFreaks. You can find the latest version of the library at http://www.henningkarlsen.com/electronics

This library has been made to make it easy to use Monochrome Graphics LCDs with 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

Version:	1.0	01 Oct 2011	• initial release





Defined Literals:

Alignment

For use with print(), printNumI() and printNumF()

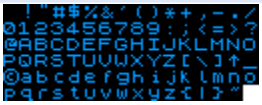
```
   LEFT:  0
  RIGHT:  9999
 CENTER:  9998
```

Included Fonts:

SmallFont

```
Charactersize:  6x8 pixels
Number of characters:  95
```

WideFont



```
Charactersize:  8x8 pixels
Number of characters:  95
```

MediumNumbers

```
Charactersize:  12x16 pixels
Number of characters:  13
```

BigNumbers

```
Charactersize:  14x24 pixels
Number of characters:  13
```

Functions:

MGLCD(D0, D1, D2, D3, D4, D5, D6, D7, A0, RW, EP, RST);

Class constructor.

```
Parameters:     D0-D7:  Arduino pins for Data bus
                A0:     Arduino pin for Register Select (Data/Command)
                RW:     Arduino pin for Read/Write
                EP:     Arduino pin for Data Latching
                RST:    Arduino pin for Reset
Usage:          MGLCD myGLCD(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13); // Start an instance of the MGLCD class
```

initLCD();

Initialize the LCD.

```
Parameters:     None
Usage:          myGLCD.initLCD(); // Initialize the display
Notes:          This will reset and clear the display.
```

rotateDisplay(value);

Select if the output on the display should be rotated 180 degrees.

```
Parameters:     value: true  - Rotate output 180 degrees
                       false – Do not rotate output
Usage:          myGLCD.rotateDisplay(true); // Rotate output to the display
Notes:          rotateDisplay() must be called before calling initLCD() to have any effect.
```

clrScr();

Clear the screen.

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

fillScr();

Fill the screen.

```
Parameters:     None
Usage:          myGLCD.fillScr(); // Fill 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
```

setPixel(x, y);

Turn on the specified pixel.

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

clrPixel(x, y);

Turn off the specified pixel.

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

invPixel(x, y);

Invert the state of the specified pixel.

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

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`

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`

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`

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 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>
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.

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(0,0,127,63); // Draw a line from the upper left to the lower right corner
```

drawRect(x1, y1, x2, y2);

Draw a rectangle between two points.

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(64,32,127,63); // 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.
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(0,0,63,31); // Draw a rounded rectangle in the upper left corner of the screen
```

drawCircle(x, y, radius);

Draw a circle with a specified radius.

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(63,31,20); // Draw a circle in the middle of the screen with a radius of 20 pixels
```