

ITDB02_Graph16 - Arduino library support for ITDB02 LCD Board

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 especially for the 3.2" TFT LCD Screen Module: ITDB02-3.2 by ITEad studio. This library has been designed to use 16bit mode, and it should work with the 2.4" Module in 16bit mode as well, although I do not have one, so this is untested.

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: | 2.0 | Aug 15 2010 | • initial release |
| | 2.1 | Sep 30 2010 | • Added Arduino Mega compatibility • Fixed a bug with CENTER and RIGHT in LANDSCAPE mode • Fixed a bug in printNumI and printNumF when the number to be printed was 0 |
| | 2.2 | Oct 14 2010 | • Added support for ITDB02-3.2WC • Added drawBitmap() with its associated tool |
| | 2.3 | Nov 24 2010 | • Added Arduino Mega2560 compatibility • Added support for rotating text and bitmaps. |
| | 2.4 | Jan 18 2011 | • Fixed an error in the requirements |
| | 2.5 | Jan 30 2011 | • Added loadBitmap() • Optimized drawBitmap() when not using rotation |
| | 2.6 | Mar 4 2011 | • Fixed a bug in printNumF when the number to be printed was (-)0.something |
| | 3.0 | Mar 19 2011 | • General optimization |
| | 3.01 | Mar 20 2011 | • Reduced memory footprint slightly |
| | 4.0 | Mar 27 2011 | • Remade the font-system to make it more flexible |
| | 4.1 | Apr 19 2011 | • Remade the tinyFAT integration. Moved loadBitmap() to the ITDB02_tinyFAT library |

(*) Initial release is v2.0 to keep it in sync with the 8bit library.

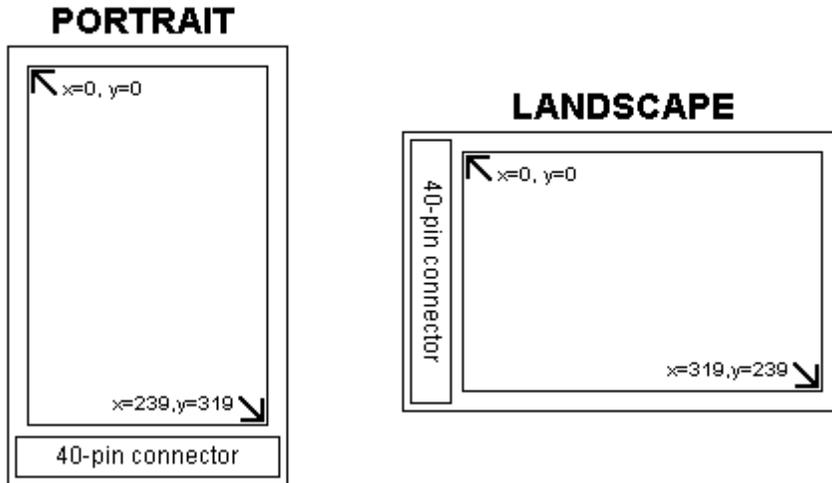
IMPORTANT:

If you are upgrading from a version below v4.0 you have to delete the old library before unpacking v4.0+

INTEGRATION WITH tinyFAT:

tinyFAT integration has been moved to a separate library. Please use the [ITDB02_tinyFAT16](#) library to enable integration.

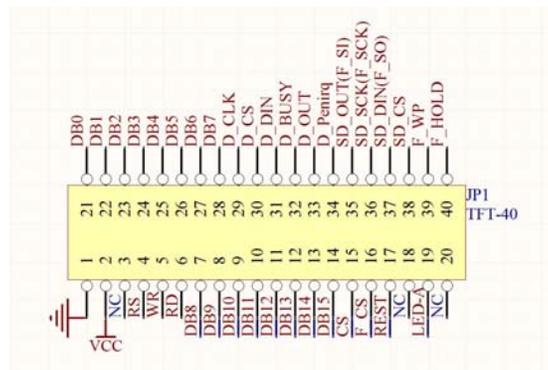
DISPLAY ORIENTATION:



Requirements:

The library require the following connections:

| Signal | ITDB02 pin | Arduino pin* | Arduino Mega pin |
|--------|------------|--------------|------------------|
| DB0 | 21 | D8 | D37 |
| DB1 | 22 | D9 | D36 |
| DB2 | 23 | D10 | D35 |
| DB3 | 24 | D11 | D34 |
| DB4 | 25 | D12 | D33 |
| DB5 | 26 | D13 | D32 |
| DB6 | 27 | A0 (D14) | D31 |
| DB7 | 28 | A1 (D15) | D30 |
| DB8 | 7 | D0 | D22 |
| DB9 | 8 | D1 | D23 |
| DB10 | 9 | D2 | D24 |
| DB11 | 10 | D3 | D25 |
| DB12 | 11 | D4 | D26 |
| DB13 | 12 | D5 | D27 |
| DB14 | 13 | D6 | D28 |
| DB15 | 14 | D7 | D29 |



ITDB02 pinout

* All boards with pinout like the Arduino Duemilanove / Arduino UNO

Defined Literals:

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

| Orientation |
|-----------------------------|
| For use with InitLCD() |
| PORTRAIT: 0 LANDSCAPE: 1 |

| Aspect Ratio |
|---------------------------------|
| For use with ITDB02() |
| ASPECT_4x3: 0 ASPECT_16x9: 1 |

Included Fonts:

| SmallFont |
|---|
|  <pre>!"#\$%&'()*+,-./ 0123456789:;<=>? @ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_ `abcdefghijklmno pqrstuvwxyz{ }~</pre> |
| Character size: 8x12 pixels Number of characters: 95 |

| BigFont |
|--|
|  <pre>!"#\$%&'()*+,-./ 0123456789:;<=>? @ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_ `abcdefghijklmno pqrstuvwxyz{ }~</pre> |
| Character size: 16x16 pixels Number of characters: 95 |

| SevenSegNumFont |
|--|
|  <pre>0123456789</pre> |
| Character size: 32x50 pixels Number of characters: 10 |

Functions:

| ITDB02(RS, WR, CS, RST[, Aspect]); | |
|---|--|
| The main class of the interface. | |
| Parameters: | RS: Arduino pin for Register Select WR: Arduino pin for Write CS: Arduino pin for Chip Select RST: Arduino pin for Reset Aspect: <optional> ASPECT_4x3 for ITDB02-2.4 and ITDB02-3.2 (both 240x320 pixels) (default) ASPECT_16x9 for ITDB02-3.2WC (240x400 pixels) |
| Usage: | ITDB02 myGLCD(19,18,17,16); // Start an instance of the ITDB02 class |

| InitLCD([orientation]); | |
|---|--|
| Initialize the LCD and set display orientation. | |
| Parameters: | Orientation: <optional> PORTRAIT (default) LANDSCAPE |
| Usage: | myGLCD.initLCD(); // Initialize the display |
| Notes: | This will reset color to white with black background. Font size will be reset to FONT_SMALL. |

| clrScr(); | |
|--|--------------------------------------|
| Clear the screen. The background-color will be set to black. | |
| Parameters: | None |
| Usage: | myGLCD.clrScr(); // Clear the screen |

| fillScr(r, g, b); | |
|---|---|
| Fill the screen with a specified color. | |
| Parameters: | r: Red component of an RGB value (0-255) g: Green component of an RGB value (0-255) b: Blue component of an RGB value (0-255) |
| Usage: | myGLCD.fillScr(255,127,0); // Fill the screen with orange |

| setColor(r, g, b); | |
|---|---|
| Set the color to use for all draw*, fill* and print commands. | |
| Parameters: | r: Red component of an RGB value (0-255) g: Green component of an RGB value (0-255) b: Blue component of an RGB value (0-255) |
| Usage: | myGLCD.setColor(0,255,255); // Set the color to cyan |

| setBackColor(r, g, b); | |
|---|---|
| Set the background color to use for all print commands. | |
| Parameters: | r: Red component of an RGB value (0-255) g: Green component of an RGB value (0-255) b: Blue component of an RGB value (0-255) |
| Usage: | myGLCD.setBackColor(255,255,255); // Set the background color to white |

| drawPixel(x, y); | |
|-------------------------|---|
| Draw a single pixel. | |
| Parameters: | x: x-coordinate of the pixel (0-239) y: y-coordinate of the pixel (0-319) |
| Usage: | myGLCD.drawPixel(119,159); // Draw a single pixel at the center of the screen |

| drawLine(x1, y1, x2, y2); | |
|----------------------------------|--|
| Draw a line between two points. | |
| Parameters: | x1: x-coordinate of the start-point (0-239) y1: y-coordinate of the start-point (0-319) x2: x-coordinate of the end-point (0-239) y2: y-coordinate of the end-point (0-319) |
| Usage: | myGLCD.drawLine(0,0,239,319); // 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 (0-239)
 y1: y-coordinate of the start-corner (0-319)
 x2: x-coordinate of the end-corner (0-239)
 y2: y-coordinate of the end-corner (0-319)
Usage: myGLCD.drawRect(119,159,239,319); // 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 (0-239)
 y1: y-coordinate of the start-corner (0-319)
 x2: x-coordinate of the end-corner (0-239)
 y2: y-coordinate of the end-corner (0-319)
Usage: myGLCD.drawRoundRect(0,0,119,159); // Draw a rounded rectangle in the upper left corner of the screen

fillRect(x1, y1, x2, y2);

Draw a filled rectangle between two points.

Parameters: x1: x-coordinate of the start-corner (0-239)
 y1: y-coordinate of the start-corner (0-319)
 x2: x-coordinate of the end-corner (0-239)
 y2: y-coordinate of the end-corner (0-319)
Usage: myGLCD.fillRect(119,0,239,159); // Draw a filled rectangle in the upper right corner of the screen

fillRoundRect(x1, y1, x2, y2);

Draw a filled 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 (0-239)
 y1: y-coordinate of the start-corner (0-319)
 x2: x-coordinate of the end-corner (0-239)
 y2: y-coordinate of the end-corner (0-319)
Usage: myGLCD.fillRoundRect(0,159,119,319); // Draw a filled, rounded rectangle in the lower 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 (0-239)
 y: y-coordinate of the center of the circle (0-319)
 radius: radius of the circle in pixels
Usage: myGLCD.drawCircle(119,159,20); // Draw a circle in the middle of the screen with a radius of 20 pixels

fillCircle(x, y, radius);

Draw a filled circle with a specified radius.

Parameters: x: x-coordinate of the center of the circle (0-239)
 y: y-coordinate of the center of the circle (0-319)
 radius: radius of the circle in pixels
Usage: myGLCD.fillCircle(119,159,10); // Draw a filled circle in the middle of the screen with a radius of 10 pixels

print(st, x, y[, deg]);

Print a string at the specified coordinates. An optional background color can be specified. Default background is black. You can use the literals LEFT, CENTER and RIGHT as the x-coordinate to align the string on the screen.

Changed in v2.3

Parameters: st: the string to print
 x: x-coordinate of the upper, left corner of the first character (0-239)
 y: y-coordinate of the upper, left corner of the first character (0-319)
 deg: <optional>
 Degrees to rotate text (0-359). Text will be rotated around the upper left corner.
Usage: myGLCD.print("Hello, World!",CENTER,0); // Print "Hello, World!" centered at the top of the screen
Notes: CENTER and RIGHT will not calculate the coordinates correctly when rotating text.

printNumI(num, x, y);

Print an integer number at the specified coordinates. An optional background color can be specified. Default background is black. 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 (0-239)
y: y-coordinate of the upper, left corner of the first digit/sign (0-319)
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. An optional background color can be specified. Default background is black.

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.

| Fractional digits | Approx range |
|-------------------|---------------|
| 1 | +/- 200000000 |
| 2 | +/- 20000000 |
| 3 | +/- 2000000 |
| 4 | +/- 200000 |
| 5 | +/- 20000 |

setFont(fontname);

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

Added in v4.0

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

drawBitmap(x, y, sx, sy, data[, scale]);

Draw a bitmap on the screen.

Added in v2.2

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
scale: <optional>
Scaling factor. Each pixel in the bitmap will be drawn as <scale>x<scale> pixels on screen.
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 565" or "ImageConverter565.exe" in the Tools-folder to convert pictures into compatible arrays. The online-tool can be found on my website.
Requires that you #include <avr/pgmspace.h>

drawBitmap(x, y, sx, sy, data, deg, rox, roy);

Draw a bitmap on the screen with rotation.

Added in v2.3

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
deg: Degrees to rotate bitmap (0-359)
rox: x-coordinate of the pixel to use as rotational center relative to bitmaps upper left corner
roy: y-coordinate of the pixel to use as rotational center relative to bitmaps upper left corner
Usage: myGLCD.drawBitmap(50, 50, 32, 32, bitmap, 45, 16, 16); // Draw a bitmap rotated 45 degrees around its center
Notes: You can use the online-tool "ImageConverter 565" or "ImageConverter565.exe" in the Tools-folder to convert pictures into compatible arrays. The online-tool can be found on my website.
Requires that you #include <avr/pgmspace.h>