RPILCD

2.2 inch, 2.4 inch and 2.8 inch LCD with touch library

Manual

PREFACE:

This library is the communication of 2.2 LCD module, 2.4 LCD module and 2.8 LCD module.

This library supports 8 bit LCD module and it will work with raspberry pi.

DEFINED LITERALS:

Set the pin link to the LCD

SetPinNU(P0, P1, P2, P3, P4, P5, P6, P7, Prs, Pcs, Pwr, Prst, Pdout, Pirq, Pdin, Pclk);

Parameters:

P0: the D8 of the LCD module;

P1: the D9 of the LCD module;

P2: the D10 of the LCD module;

P3: the D11 of the LCD module;

P4: the D12 of the LCD module;

P5: the D13 of the LCD module;

P6: the D14 of the LCD module;

P7: the D15 of the LCD module;

Prs: the RS of the LCD module;

Pcs: the CS of the LCD module;

Pwr: the WR of the LCD module;

Prst: the RST of the LCD module;

Pdout: the T_DOUT of the LCD module;

Pirq: the T_IRQ of the LCD module;

Pdin: the T_DIN of the LCD module;

Pclk: the T_CLK of the LCD module;

Choose the size of the LCD

SetLCDSize(int a);

Parameters: LCD_22:2.2 inch LCD

LCD_24:2.4 inch LCD

LCD_28:2.8 inch LCD

Initialize LCD

LCDInit(void);

Parameters: NULL

Initialize the touch of the LCD

Touch_Init(void);

Parameters: NULL

Set font

setFont(unsigned short int mxsize,unsigned short int mysize,unsigned short int moffset)

parameters: mxsize: the wide of the font

mysize: the high of the font

moffset: the offset is 32

Set VGA Colors

setColor() and setBackColor();

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)

VGA Colors

Predefined colors for use with setColor() and setBackColor()

VGA_BLACK	VGA_SILVER	VGA_GRAY	VGA_WHITE
VGA_MAROON	VGA_RED	VGA_PURPLE	VGA_FUCHSIA
VGA_GREEN	VGA_LIME	VGA_OLIVE	VGA_YELLOW
VGA_NAVY	VGA_BLUE	VGA_TEAL	VGA_AQUA

INCLUDED FONTS:

SmallFont

Charactersize: 8x12 pixels
Number of characters: 95

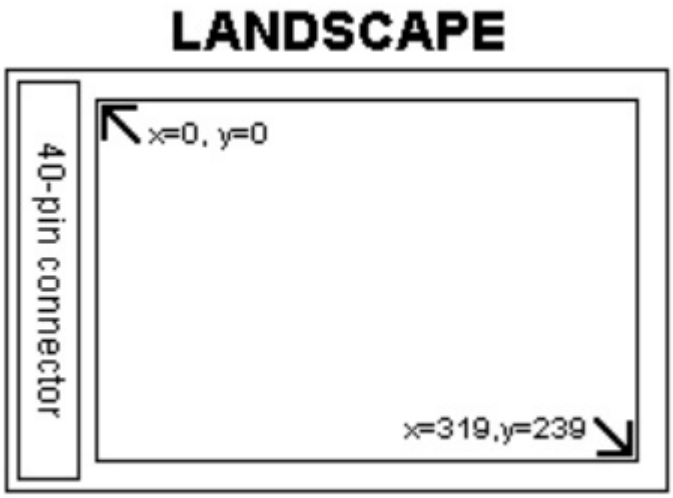
BigFont

Charactersize: 16x16 pixels
Number of characters: 95

SevenSegNumFont

Charactersize: 32x50 pixels
Number of characters: 10

Display orientarion:



FUNCTIONS:

SetPinNU(P0, P1, P2, P3, P4, P5, P6, P7, Prs, Pcs, Pwr, Prst, Pdout, Pirq, Pdin, Pclk)

Set the pin link to the LCD

Parameters:

P0: the D8 of the LCD module;

P1: the D9 of the LCD module;

P2: the D10 of the LCD module;

P3: the D11 of the LCD module;

P4: the D12 of the LCD module;

P5: the D13 of the LCD module;

P6: the D14 of the LCD module;

P7: the D15 of the LCD module;

Prs: the RS of the LCD module;

Pcs: the CS of the LCD module;

Pwr: the WR of the LCD module;

Prst: the RST of the LCD module;

Pdout: the T_DOUT of the LCD module;

Pirq: the T_IRQ of the LCD module;

Pdin: the T_DIN of the LCD module;

Pclk: the T_CLK of the LCD module;

Usage: SetPinNU(0,1,2,3,4,5,6,7,8,10,9,11,15,16,14,12)

SetLCDSize(size)

Choose the size of the LCD

Parameters: LCD_22:2.2 inch LCD

LCD_24:2.4 inch LCD

LCD_28:2.8 inch LCD

Usage: SetLCDSize(LCD_22); // choose 2.2 inch LCD to display

Lcd_Init ();

Initialize the LCD.

Parameters: NULL

Usage: Lcd_Init (); // Initialize the display

getDisplayXSize();

Get the width of the screen.

Parameters: None

Returns: Width of the screen in pixels

Usage: Xsize = getDisplayXSize(); // Get the width

getDisplayYSize();

Get the height of the screen.

Parameters: None

Returns: Height of the screen in pixels

Usage: Ysize = getDisplayYSize(); // Get the height

clrScr();

Clear the screen. The background-color will be set to black.

Parameters: None

Usage: clrScr(); // Clear the screen to black.

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: fillScr(255,127,0); // Fill the screen with orange

fillScr(color);

Fill the screen with a specified pre-calculated RGB565 color.

Parameters: color: RGB565 color value

Usage: fillScr(0x0000); // Fill the screen with black

setColorRGB(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: setColorRGB(0,255,255); // Set the color to cyan.

setColor(color);

Parameters:

color: RGB565 color value

Usage: setColor(0x0000); // Set the color to black

getColor();

Get the currently selected color.

Parameters: None

Returns: Currently selected color as a RGB565 value (word)

Usage:Color = getColor(); // Get the current color

setBackColorRGB(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: setBackColorRGB(255,255,255); // Set the background color to white

setBackColor(color);

Set the specified pre-calculated RGB565 background color to use for all print commands.

Parameters:

color: RGB565 color value

Usage: setBackColor(0xffff); // Set the background color to white

getBackColor();

Get the currently selected background color.

Parameters: None

Returns: Currently selected background color as a RGB565 value (word)

Usage: BackColor = getBackColor(); // Get the current background color

drawPixel(x, y);

Draw a single pixel.

Parameters:

x: x-coordinate of the pixel

y: y-coordinate of the pixel

Usage: drawPixel(119,159); // Draw a single pixel

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: drawLine(0,0,239,319); // Draw a diagonal line

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: drawRect(119,159,239,319); // Draw a rectangle

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: drawRoundRect(0,0,119,159); // Draw a rounded rectangle

fillRect(x1, y1, x2, y2);

Draw a filled 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: fillRect(119,0,239,159); // Draw a filled rectangle

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

y1: y-coordinate of the start-corner

x2: x-coordinate of the end-corner

y2: y-coordinate of the end-corner

Usage: fillRoundRect(0,159,119,319); // Draw a filled, rounded rectangle

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: drawCircle(119,159,20); // Draw a circle 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

y: y-coordinate of the center of the circle

radius: radius of the circle in pixels

Usage: fillCircle(119,159,10); // Draw a filled circle 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.

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

deg: <optional>Degrees to rotate text (0-359). Text will be rotated around the upper left corner.

Usage: print(“Hello, World!”,CENTER,0); // Print “Hello, World!”

printNumI(num, x, y[, length[, filler]]);

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

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: printNumI(num,CENTER,0,0); // Print the value of “num”

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

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

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 ' ' (space).

Usage: printNumF(num, 3, CENTER,0,0); // Print the value of “num” with 3 fractional digits

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

setFont(x,y,offset);

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

Parameters:

X: The x seize of the Font;

Y: The Y size of the Font;

Offset: We set it to 32;

Usage: setFont(16,16,32); // Select the font called BigFont

getFont();

Get the currently selected font.

Parameters: None

Returns: Currently selected font

Usage: CurrentFont = getFont(); // Get the current font

getFontXsize();

Get the width of the currently selected font.

Parameters: None

Returns: Width of the currently selected font in pixels

Usage: Xsize = getFontXsize (); // Get font width

getFontYsize();

Get the height of the currently selected font.

Parameters: None

Returns: Height of the currently selected font in pixels

Usage: Ysize = getFontYsize (); // Get font height

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

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

scale: <optional>Scaling factor. Each pixel in the bitmap will be drawn as <scale>x<scale> pixels on screen.

Usage: drawBitmap(0, 0, 32, 32, bitmap); // Draw a 32x32 pixel bitmap

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

Draw a bitmap on the screen with rotation.

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: drawBitmap(50, 50, 32, 32, bitmap, 45, 16, 16); // Draw a bitmap rotated 45 degrees around its center

For Touch:

Touch_Init ();

Initialize the touch screen.

Parameters:NULL

Returns: Nothing

Usage: Touch_Init ();// Initialize the touch screen

Touch_DataAvailable();

Check to see if new data from the touch screen is waiting.

Parameters: None

Returns:NULL

Boolean: true means data is waiting, otherwise false

Usage: check = Touch_DataAvailable() // See if data is waiting

Touch_Read();

Read waiting data from the touch screen. This function should be called if dataAvailable() is true. Use Touch_GetX() and Touch_GetY() to get the coordinates.

Parameters: None

Returns: Touch read data;

Usage: Touch_Read(); // Read data from touch screen

Notes: After calling read(), raw data from the touch screen is available in the variables TP_X and TP_Y. Do not use these if you do not know how to handle the raw data. Use Touch_GetX() and Touch_GetY() instead.

Touch_GetX();

Get the x-coordinate of the last position read from the touch screen.

Parameters: None

Returns: Integer

Usage: x = Touch_GetX(); // Get the x-coordinate

Touch_GetY();

Get the y-coordinate of the last position read from the touch screen.

Parameters: None

Returns: Integer

Usage: y = Touch_GetY(); // Get the y-coordinate

Touch_SetPrecision(precision);

Set the precision of the touch screen.

Parameters: precision: PREC_LOW, PREC_MEDIUM, PREC_HI, PREC_EXTREME

Returns: Nothing

Usage: Touch_SetPrecision(PREC_MEDIUM); // Set precision to medium