

Lua-RTOS-ESP32

LCD Module Reference



LoBo 01/2017

Content

TFT Module.....	3
Function List.....	3
Constants.....	4
Functions.....	6
tft.init().....	6
tft.clear().....	6
tft.off().....	7
tft.on().....	7
tft.invert().....	7
tft.setorient().....	8
tft.setclipwin().....	8
tft.resetclipwin().....	9
tft.setrot().....	9
tft.settransp().....	10
tft.setwrap().....	10
tft.setfixed().....	10
tft.setcolor().....	11
tft.setfont().....	12
tft.getfontsize().....	12
tft.getfontheight().....	13
tft.getscreensize().....	13
tft.putpixel().....	14
tft.line().....	14
tft.rect().....	15
tft.circle().....	15

tft.triangle().....	16
tft.write().....	17
tft.stringpos().....	18
tft.hsb2rgb().....	19
tft.image().....	19
tft.bmpimage().....	20
tft.jpgimage().....	21
tft.compilefont(fontfile_name).....	22
Touch panel based on XCP2046 controller support.....	23
tft.gettouch().....	23
tft.getrawtouch().....	24
tft.setcal(calx, caly).....	24

TFT Module

Function List

tft.init	Initialize the display
tft.clear	Clear the screen
tft.write	Write strings and/or numbers to display
tft.on	Turn display on
tft.off	Turn display off
tft.setFont	Set the font used for write function
tft.getscreenSize	Get current screen size
tft.getFontsize	Get current font size in pixels
tft.getFontheight	Get current font height in pixels
tft.fixedwidth	Set fixed width or proportional character printing
tft.setrot	Set text rotation (angle)
tft.setorient	Set display orientation, default PORTRAIT
tft.setwrap	Set line wrap for tft.write() function
tft.setcolor	Set foreground and background colors
tft.settransp	Set transparency for character printing
tft.setfixed	Force fixed width printing of proportional fonts
tft.setclipwin	Set the coordinates of the clipping window
tft.resetclipwin	Reset clipping window to full screen
tft.invert	Set inverted/normal colors
tft.putpixel	Puts pixel on screen
tft.line	Draw line
tft.rect	Draw rectangle
tft.triangle	Draw triangle
tft.circle	Draw circle
tft.image	Show image from file
tft.jpgimage()	Show image from jpeg file
tft.bmpimage()	Show image from bmp file
tft.hsb2rgb	Converts HSB color values to 16-bit RGB value

Constants

tft.PORTRAIT	Default orientation
tft.PORTRAIT_FLIP	Orientation flipped portrait
tft.LANDSCAPE	Orientation landscape
tft.LANDSCAPE_FLIP	Orientation flipped landscape
tft.CENTER	Center text (write function) or jpeg image
tft.RIGHT	Right align text (write function) or jpeg image
tft.BOTTOM	Bottom align text (write function) or jpeg image
tft.LASTX	Continue writing at last X position (write function)
tft.LASTY	Continue writing at last Y position (write function)
tft.FONT_DEFAULT	Default font, DejaVu 12 proportional font
tft.FONT_DEJAVU18	
tft.FONT_DEJAVU24	
tft.FONT_UBUNTU16	
tft.FONT_COMIC24	
tft.FONT_TOONEY32	
tft.FONT_MINYA24	
tft.FONT_7SEG	7 segment vector font (digits, '-', ':', 'deg' only)
tft.ST7735	ST7735 based display, type #0
tft.ST7735B	ST7735 based display, type #1
tft.ST7735G	ST7735 based display, type #2
tft.ILI9341	ILI9341 based display
tft.BLACK	Colors
tft.NAVY	
tft.DARKGREEN	
tft.DARKCYAN	
tft.MAROON	
tft.PURPLE	
tft.OLIVE	
tft.LIGHTGREY	
tft.DARKGREY	
tft.BLUE	
tft.GREEN	
tft.CYNAN	
tft.RED	
tft.MAGENTA	
tft.YELLOW	
tft.WHITE	
tft.ORANGE	
tft.GREENYELLOW	
tft.PINK	

The module supports operations with TFT SPI display modules.

Various display modules based on [ST7735](#) and [ILI9341](#) controllers, using 4-wire SPI interface are supported.

SPI speed can be set to up to 40 MHz.

Back light can be powered directly from 3.3V or with PWM pin (via MOSFET).

Connecting ESP32 to display module (defaults):

ESP32	Pin		Display
MOSI	GPIO23	->	SDI (MOSI)
MISO	GPIO25	->	SDO (MISO)
CLK	GPIO19	->	SCK
CS	GPIO22	->	CS
DC	DPIO21	->	DC
RESET, not used, pullup (4.7K) to power supply			

Different pins can be configured with `tft.config()` function.

Functions

tft.init()

Description

Initialize the tft display and clear the screen.

You must initialize the SPI interface first if not using Xadow display.

Syntax

```
res = tft.init(type [,orient])
```

Parameters

type: display type, 0, 1, 2 (probably 1 will work best) for [ST7735](#)
3 for [ILI9341](#)
You can use defined constants:
ST7735, ST7735B, ST7735G, ILI9341

orient: [optional](#), display orientation (default: PORTRAIT)

Returns

res: 0 on success, error code on error

Examples

```
>res = tft.init(tft.ILI9341,tft.LANDSCAPE)
```

tft.clear()

Description

Clear screen to default or specified color.

Syntax

```
tft.clear([color])
```

Parameters

color [optional](#); fill the screen with color (default: BLACK)

Returns

nil

Examples

```
> tft.clear(tft.BLUE)  
> tft.clear()
```

tft.off()

Description

Turns the display off, preserve power. Back light has to be turned off separately.

Syntax

```
tft.off()
```

Parameters

nil

Returns

nil

Examples

```
> tft.off()
```

tft.on()

Description

Turns the display on.

Syntax

```
tft.on()
```

Parameters

nil

Returns

nil

Examples

```
> tft.on()
```

tft.invert()

Description

Set inverted/normal colors.

Syntax

```
tft.invert(inv)
```

Parameters

inv 0: inverted colors off; 1: inverted colors on

Returns

nil

Examples

```
> tft.invert(0)
```

tft.setorient()

Description

Set display orientation.

Syntax

```
tft.setorient(orient)
```

Parameters

orient one of display orientation constants
PORTRAIT, PORTRAIT_FLIP, LANDSCAPE, LANDSCAPE_FLIP

Returns

nil

Examples

```
> tft.orient(tft.LANDSCAPE)  
> tft.orient(tft.PORTRAIT_FLIP)
```

tft.setclipwin()

Description

Sets the clipping area coordinates. All writing to screen is clipped to that area.
Starting x & y in all functions will be adjusted to the clipping area.
This setting has no effect on tft.image function.

Syntax

```
tft.setclipwin(x1, y1, x2, y2)
```

Parameters

x1,y1 upper left point of the clipping area
x1,y1 bottom right point of the clipping area

Returns

nil

Examples

```
> tft.setclipwin(20,20,220,200)
```

tft.resetclipwin()

Description

Resets the clipping are coordinates to default full screen.

Syntax

```
tft.resetclipwin()
```

Parameters

nil

Returns

nil

Examples

```
> tft.resetclipwin()
```

tft.setrot()

Description

Set text rotation (angle) for tft.write() function. Has no effect on FONT_7SEG.

Syntax

```
tft.setrot(rot)
```

Parameters

rot rotation angle (0~360)

Returns

nil

Examples

```
> tft.rot(90)
> tft.write(50,50,"Rotated text")
```

tft.settransp()

Description

Set transparency when writing the text. If transparency is on, only text foreground color is shown.

Syntax

```
tft.settransp(transp)
```

Parameters

transp 0: transparency off; 1: transparency on

Returns

nil

Examples

```
> tft.settransp(1)
```

tft.setwrap()

Description

Set line wrapping writing the text. If wrapping is on, text will wrap to new line, otherwise it will be clipped.

Syntax

```
tft.setwrap(wrap)
```

Parameters

wrap 0: line wrap off; 1: line wrap on

Returns

nil

Examples

```
> tft.setwrap(1)
```

tft.setfixed()

Description

Forces fixed width print of the proportional font.

Syntax

```
tft.setfixed(force)
```

Parameters

force 0: force fixed width off; 1: force fixed width on

Returns

nil

Examples

```
> tft.setfixed(1)
```

tft.setcolor()

Description

Set the color used when writing characters or drawing on display.

Syntax

```
tft.setcolor(color[,bgcolor])
```

Parameters

color foreground color for text and drawing
bgcolor **optional**; background color for writing text

Returns

nil

Examples

```
> tft.setcolor(tft.YELLOW)  
> tft.setcolor(tft.ORANGE, tft.DARKGREEN)
```

tft.setFont()

Description

Set the font used when writing the text to display.

Two embeded fonts are available:

tft.FONT_DEFAULT (default, DejaVu12),

tft.FONT_7SEG (vector font, imitates 7 segment displays).



7-segment font is the vector font for which any size can be set (distance between bars and the bar width). Only characters **0,1,2,3,4,5,6,7,8,.,-,:,/** are available. Character **'/'** draws the degree sign.

Any number of fonts given by name and read from file can be used.

See example fonts for font file format.

Syntax

```
tft.setFont(font [,size, width])
```

Parameters

font one of the available fonts
size **optional**; only for FONT_7SEG, distance between bars
 (default: 12; min=6; max=40)
width **optional**; only for FONT_7SEG, bar width
 (default: 2; min=1; max=12 or size/2)

Returns

nil

Examples

```
> tft.setFont(tft.FONT_DEFAULT)
> tft.setFont(tft.FONT_7SEG, 20, 4)
> tft.setFont("@font/Ubuntu.fon")
```

tft.getfontsize()

Description

Get current font size in pixels. Useful if FONT_7SEG is used to get actual character width and height.

Syntax

```
tft.getfontsize()
```

Parameters

nil

Returns

xsize width of the font character in pixels.
For the proportional fonts, maximal char width will be returned
ysize height of the font character in pixels

Examples

```
> tft.getfontsize()  
8   12
```

tft.getfontheight()

Description

Get current font height in pixels.

Syntax

tft.getfontheight()

Parameters

nil

Returns

ysize height of the font character in pixels

Examples

```
> tft.setfont("/@font/Ubuntu.fon")  
> tft.getfontheight()  
16
```

tft.getscreenize()

Description

Get current screen size (width & height) in pixels.

Syntax

tft.getscreenize()

Parameters

nil

Returns

xsize width of the screen in pixels
ysize height of the screen in pixels

Examples

```
> tft.getscreensize()  
240    320
```

tft.putpixel()

Description

Draws pixel on display at coordinates (x,y) using foreground or given color

Syntax

```
tft.putpixel(x, y [, color])
```

Parameters

x, y coordinates of pixel
color **optional**: pixel color (default: current foreground color)

Returns

nil

Examples

```
> tft.putpixel(10,10)  
> tft.putpixel(20,40,tft.GREEN)
```

tft.line()

Description

Draws line from (x1,y1) to (x2,y2) using foreground or given color

Syntax

```
tft.line(x1, y1, x2, y2 [,color])
```

Parameters

x1,y1 coordinates of line start point
x1,y1 coordinates of line end point

color **optional**: line color (default: current foreground color)

Returns

nil

Examples

```
> tft.line(0,0,127,159)
> tft.line(20,40,80,10,tft.ORANGE)
```

tft.rect()

Description

Draws rectangle at (x,y) w pixels wide, h pixels high, with given color. If the fill color is given, fills the rectangle.

Syntax

```
tft.rect(x, y, w, h, color [,fillcolor])
```

Parameters

x, y	coordinates of the upper left corner of the rectangle
w	width of the rectangle
h	height of the rectangle
color	rectangle outline color
fillcolor	optional : rectangle fill color

Returns

nil

Examples

```
> tft.rect(10,10,100,110,tft.RED)
> tft.rect(0,0,128,160,tft.ORANGE,tft.YELLOW)
```

tft.circle()

Description

Draws circle with center at (x,y) and radius r, with given color. If the fill color is given, fills the circle.

Syntax

```
tft.circle(x, y, r, color [,fillcolor])
```


Parameters

x, y	coordinates circle center
r	radius of the circle
color	circle outline color
fillcolor	optional: circle fill color

Returns

nil

Examples

```
> tft.circle(64,80,20,tft.RED)
> tft.circle(50,60,30,tft.ORANGE,tft.YELLOW)
```

tft.triangle()

Description

Draws triangle between three given points, with given color. If the fill color is given, fills the triangle.

Syntax

```
tft.triangle(x1, y1, x2, y2, x3, y3, color [,fillcolor])
```

Parameters

x1, y1, x2, y2, x3, y3	coordinates of the 3 triangle points
color	triangle outline color
fillcolor	optional: triangle fill color

Returns

nil

Examples

```
> tft.triangle(50,20,80,100,20,100,tft.RED)
> tft.triangle(50,20,80,100,20,100,tft.RED, tft.WHITE)
```

tft.write()

Description

Write strings and/or numbers to display. Rotation of the displayed text can be set with tft.setrot() function.

Two special characters are allowed in strings:

- `\r` CR (0x0D), clears the display to EOL
- `\n` LF (0x0A), continues to the new line, x=0

Syntax

```
tft.write(x, y, data1, [data2, ... datan])
```

Parameters

- x:** x position (column; 0~screen width-1)
Special values can be entered:
tft.CENTER, centers the text; tft.RIGHT, right justifies the text
tft.LASTX, continues from last X position
- y:** y position (row; 0~screen height-1)
Special values can be entered:
tft.CENTER, centers the text; tft.BOTTOM, bottom justifies the text
tft.LASTY, continues from last Y position
- data1:** number or string to write to the display
If simple number is given, integer is printed. The number can be given as a table containing number (float) and number of decimal places.
- data2:** optional
- datan:** optional

Returns

nil

Examples

```
>tft.setcolor(tft.YELLOW)
>tft.write(0,0,"Hi, ESP32-Lua")
>t=2.3456
>tft.write(8,16,"Temp=", {t,2})
```

tft.stringpos()

Description

Returns actual coordinates at which the string would be written and string width and height. Usefull if using CENTER, RIGHT, BOTTOM, LAST_X, LAST_Y coordinates.

Calculated using current font properties.

Syntax

`x, y, w, h = tft.stringpos(x, y, str)`

Parameters

- x: x position (column; 0~screen width-1)
Special values can be entered:
tft.CENTER, centers the text; tft.RIGHT, right justifies the text
tft.LASTX, continues from last X position
- y: y positoin (row; 0~screen height-1)
Special values can be entered:
tft.CENTER, centers the text; tft.BOTTOM, bottom justifies the text
tft.LASTY, continues from last Y position
- str: string which would be written to the display

Returns

- x: actual X position
- y: actual Y position
- w: width of the string in pixels
- h: height os string in pixels

Examples

```
>tft.setFont(tft.FONT_DEFAULT)
>tft.stringpos(tft-CENTER, tft.CENTER, "Hi, ESP32-Lua")
109      116      99      11
>tft.setFont(tft.FONT_COMIC24)
>tft.stringpos(tft-CENTER, tft.CENTER, "Hi, ESP32-Lua")
72       113      174      25
>tft.stringpos(tft-RIGHT, tft.BOTTOM, "Hi, ESP32-Lua")
144      213      174      25
```

tft.hsb2rgb()

Description

Converts HSB (hue, saturation, brightness) color values to 16-bit RGB value.

Syntax

```
Color = tft.hsb2rgb(hue, sat, bri)
```

Parameters

hue	float, hue value (0.0 ~ 359.9999)
sat	float, saturation value (0.0 ~ 1.0)
bri	brightness value (0.0 ~ 1.0)

Returns

color	16-bit RGB color value
-------	------------------------

Examples

```
> tft.circle(50,60,30,tft.ORANGE,tft.hsb2rgb(90.0,1.0,0.5))
```

tft.image()

Description

Shows the image from file. The image file must be in raw 16bit format.

Any image can be converted with *ImageConverter565.exe* which can be found in on GitHub repository.

Be careful to give the right image width and height.

Syntax

```
tft.image(x, y, xsize, ysize, filename)
```

Parameters

x:	x position of the image upper left corner
y:	y position of the image upper left corner
xsize:	image xsize (width)
ysize;	image ysize (height)
filename:	name of the row image file

Returns

nil

Examples

```
>tft.rot(tft.PORTRAIT)
>tft.clear()
>tft.image(0,0,128,96,"newyear_128x96.img")
>tft.rot(tft.LANDSCAPE)
>tft.image(0,0,160,123,"nature_160x123.img")
```

tft.bmpimage()

Description

Shows the image from file. The image file must be in bmp.

If image dimensions are greater than screen size, the image is cropped.

Only RGB 24-bit BMP images can be displayed

Syntax

```
tft.bmpimage(x, y, filename)
```

Parameters

x:	x position of the image upper left corner tft.CENTER, tft.RIGHT can be used to align image on screen
y:	y position of the image upper left corner tft.CENTER, tft.BOTTOM can be used to align image on screen
filename:	name of the jpeg image file

Returns

nil

Examples

```
>tft.rot(tft.PORTRAIT)
>tft.clear()
>tft.image(0,0,"tiger.bmp")
```

tft.jpgimage()

Description

Shows the image from file. The image file must be in jpeg.

If image dimensions are greater than screen size, image can be automatically scaled.

Limits:

JPEG standard:	Baseline only. Progressive and Lossless JPEG format are not supported.
Image size:	Upto 65520 x 65520 pixels.
Colourspace:	YCbCr three components only. Grayscale image is not supported.
Sampling factor:	4:4:4, 4:2:2 or 4:2:0.

Syntax

tft.jpgimage(x, y, maxscale, filename)

Parameters

x:	x position of the image upper left corner tft.CENTER, tft.RIGHT can be used to align image on screen
y:	y position of the image upper left corner tft.CENTER, tft.BOTTOM can be used to align image on screen
maxscale:	0~3 scale factor; the image is automatically scaled to fit the screen if maxscale > 0 up to maxscale (1/2, 1/4, 1/8)
filename:	name of the jpeg image file

Returns

nil

Examples

```
>tft.rot(tft.PORTRAIT)
>tft.clear()
>tft.image(0,0,0,"tiger.jpg")
```

tft.compilefont(fontfile_name)

Description

Compile font source file (extension must be .c) to the binary font file (same name, extension .fon) which can be used with tft.setfont() function.
It is recommended that all font files are placed in some subdirectory.

Syntax

```
tft.compilefont( font_filename)
```

Parameters

font_filename:	font source file name
----------------	-----------------------

Returns

nil

Examples

```
>tft.compilefont("/@fonts/Ubuntu.c")
```

Touch panel based on XCP2046 controller support

Touch panels based on [XCP2046](#) controller, usually found on [ILI9341](#) based TFT boards are fully supported.

The same SPI interface is used as for tft. The controller's MOSI&MISO pins has to be connected in parallel with the LCD MOSI&MISO pins, separate TP CS pin has to be defined.

The XCP2046 IRQ pin is usually not used, but can be connected to one RePhone's eint pins in which case the EINT callback can be used to detect touch event.

Before using the touch panel, it has to be calibrated. For that purpose, Lua script [tpcalib.lua](#) is available. Once calibrated, the calibration constants are saved in system parameters and automatically loaded on boot.

The demonstration Lua program [paint.lua](#) is also available. Load it with `dofile("paint.lua")` and execute with `paint.run([orient])`. ***Orient*** is optional parameter to set the screen orientation. Default value is `tft.PORTRAIT_FLIP`.

tft.gettouch()

Description

Get the touch panel calibrated coordinates.
The coordinates are adjusted to screen orientation
Only available for **ILI9341 based displays**.

Syntax

```
stat, x, y = tft.gettouch()
```

Parameters

nil

Returns

stat: 0 in no touch detected, >0 if the the panel is touched
x: calibrated X coordinate of the touched point, nil if stat=0
y: calibrated Y coordinate of the touched point, nil if stat=0

Examples

```
>print(tft.gettouch())
```


tft.getrawtouch()

Description

Get the touch panel raw (uncalibrated) coordinates.
Only available for **ILI9341 based displays**.

Syntax

```
stat, x, y = tft.gettouch()
```

Parameters

nil

Returns

stat: 0 in no touch detected, >0 if the the panel is touched
x: raw X coordinate of the touched point, nil if stat=0
y: raw Y coordinate of the touched point, nil if stat=0

Examples

```
>print(tft.gettouch())
```

tft.setcal(calx, caly)

Description

Set the touch panel calibration constants.
Only available for **ILI9341 based displays**.

Syntax

```
tft.gettouch(calx, caly)
```

Parameters

calx calibration constant obtained from calibration program
caly calibration constant obtained from calibration program

Returns

none