

# Lua for RePhone (Xadow GSM+BLE)

## LCD Module Reference



# Content

LCD Module.....	3
Function List.....	3
Constants.....	4
Functions.....	6
lcd.init().....	6
lcd.clear().....	6
lcd.off().....	6
lcd.on().....	7
lcd.invert().....	7
lcd.setorient().....	7
lcd.setclipwin().....	8
lcd.resetclipwin().....	8
lcd.setrot().....	9
lcd.settransp().....	9
lcd.setwrap().....	9
lcd.setfixed().....	10
lcd.setcolor().....	10
lcd.setfont().....	11
lcd.getfontsize().....	11
lcd.getfontheight().....	12
lcd.getscreensize().....	12
lcd.putpixel().....	12
lcd.line().....	13
lcd.rect().....	13
lcd.circle().....	14
lcd.triangle().....	14
lcd.write().....	15
lcd.hsb2rgb().....	15
lcd.image().....	16
lcd.bmpimage().....	16

lcd.jpgimage().....	17
lcd.compilefont(fontfile_name).....	18
lcd.on touch(cb_func).....	18

# LCD Module

## Function List

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

## Constants

lcd.PORTRAIT	Default orientation
lcd.PORTRAIT_FLIP	Orientation flipped portrait
lcd.LANDSCAPE	Orientation landscape
lcd.LANDSCAPE_FLIP	Orientation flipped landscape
lcd.CENTER	Center text (write function) or jpeg image
lcd.RIGHT	Right align text (write function) or jpeg image
Lcd.BOTTOM	Bottom align jpeg image
lcd.LASTX	Continue writing at last X position (write function)
lcd.LASTY	Continue writing at last Y position (write function)
lcd.FONT_DEFAULT	Default font, DejaVu 12 proportional font
lcd.FONT_7SEG	7 segment vector font (digits, '-', ':', ':', 'deg' only
lcd.ST7735	ST7735 based display, type #0
lcd.ST7735B	ST7735 based display, type #1
lcd.ST7735G	ST7735 based display, type #2
lcd.ILI9341	ILI9341 based display
lcd.XADOW_V0	XADOW Touch Display ver. 0
lcd.XADOW_V1	XADOW Touch Display ver. 0
lcd.BLACK	<a href="#">Colors</a>
lcd.NAVY	
lcd.DARKGREEN	
lcd.DARKCYAN	
lcd.MAROON	
lcd.PURPLE	
lcd.OLIVE	
lcd.LIGHTGREY	
lcd.DARKGREY	
lcd.BLUE	
lcd.GREEN	
lcd.CYNAN	
lcd.RED	
lcd.MAGENTA	
lcd.YELLOW	
lcd.WHITE	
lcd.ORANGE	
lcd.GREENYELLOW	
lcd.PINK	

The module supports operations with TFT SPI display modules.

Xadow 1.54" Touchscreen ver. 1.0&1.1 are supported.

No SPI configuration is necessary.

Touch screen support is implemented.

Various display modules based on ST7735 and ILI9341 controllers, using 4-wire SPI interface are also supported.

Touch screen is not yet implemented (will come soon).

**SPI interface must be setup before using the module.**

**CS and DC pins must be declared.**

SPI speed can be set to up to 10 MHz.

SPI pins operate at 2.8V and can drive the display board (if powered with 2.8V) directly. Display can be powered from RePhone's 2.8V output, **bypass** any voltage regulator if present on display module in that configuration.

Use **level shifters** if display module is powered with 3.3V / 5V.

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

**Connecting RePhone to display module:**

RePhone	Pin		Display
MOSI	GPIO28	->	SDI (MOSI)
MISO	GPIO29	->	SDO (MISO), <b>not used</b>
CLK	GPIO27	->	SCK
CS	any	->	CS
DC	any	->	DC
			RESET, not used, pullup (4.7K) to power supply

## Functions

### lcd.init()

#### Description

Initialize the tft display and clear the screen.

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

#### Syntax

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

#### Parameters

type: display type, **0**, **1**, **2** (probably 1 will work best) for [ST7735](#)  
**3** for [ILI9341](#)  
**8** for [Xadow Ver 1.0](#)  
**9** for [Xadow Ver 1.1](#)  
You can use defined constants:  
ST7735, ST7735B, ST7735G, ILI9341, XADOW\_V0, XADOW\_V1  
orient: [optional](#), display orientation (default: PORTRAIT)

#### Returns

res: 0 on success, error code on error

#### Examples

```
-- setup SPI 10 MHz clock
>spi.setup({mode=0, cs=2, dc=1, speed=10000})
>res = lcd.init(lcd.ILI9341,lcd.LANDSCAPE)
```

### lcd.clear()

#### Description

Clear screen to default or specified color.

#### Syntax

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

#### Parameters

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

#### Returns

nil

#### Examples

```
> lcd.clear(lcd.BLUE)
> lcd.clear()
```

### lcd.off()

#### Description

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

#### Syntax

```
lcd.off()
```

**Parameters**

nil

**Returns**

nil

**Examples**

```
> lcd.off()
```

## lcd.on()

**Description**

Turns the display on.

**Syntax**

```
lcd.on()
```

**Parameters**

nil

**Returns**

nil

**Examples**

```
> lcd.on()
```

## lcd.invert()

**Description**

Set inverted/normal colors.

**Syntax**

```
lcd.invert(inv)
```

**Parameters**

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

**Returns**

nil

**Examples**

```
> lcd.invert(0)
```

## lcd.setorient()

**Description**

Set display orientation.



**Syntax**

```
lcd.setorient(orient)
```

**Parameters**

orient one of display orientation constants  
PORTRAIT, PORTRAIT\_FLIP, LANDSCAPE, LANDSCAPE\_FLIP

**Returns**

nil

**Examples**

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

## lcd.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 lcd.image function.

**Syntax**

```
lcd.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**

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

## lcd.resetclipwin()

**Description**

Resets the clipping are coordinates to default full screen.

**Syntax**

```
lcd.resetclipwin()
```

**Parameters**

nil

**Returns**

nil

**Examples**

```
> lcd.resetclipwin()
```

## lcd.setrot()

### Description

Set text rotation (angle) for lcd.write() function. Has no effect on FONT\_7SEG.

### Syntax

```
lcd.setrot(rot)
```

### Parameters

rot      rotation angle (0~360)

### Returns

nil

### Examples

```
> lcd.rot(90)
> lcd.write("Rotated text")
```

## lcd.settransp()

### Description

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

### Syntax

```
lcd.settransp(transp)
```

### Parameters

transp    0: transparency off; 1: transparency on

### Returns

nil

### Examples

```
> lcd.settransp(1)
```

## lcd.setwrap()

### Description

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

### Syntax

```
lcd.setwrap(wrap)
```

### Parameters

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

**Returns**

nil

**Examples**

```
> lcd.setwrap(1)
```

## lcd.setfixed()

**Description**

Forces fixed width print of the proportional font.

**Syntax**

```
lcd.setwrap(force)
```

**Parameters**

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

**Returns**

nil

**Examples**

```
> lcd.setfixed(1)
```

## lcd.setcolor()

**Description**

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

**Syntax**

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

**Parameters**

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

**Returns**

nil

**Examples**

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

## lcd.setFont()

### Description

Set the font used when writing the text to display.

Two embeded fonts are available:

FONT\_DEFAULT (default, DejaVu12),

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

```
lcd.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

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

## lcd.getFontsize()

### Description

Get current font size in pixels. Useful if FONT\_7SEG is used to get actual character width and height.

### Syntax

```
lcd.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

```
> lcd.getfontsize()  
8    12
```

## lcd.getfontheight()

### Description

Get current font height in pixels.

### Syntax

```
lcd.getfontheight()
```

### Parameters

nil

### Returns

ysize    height of the font character in pixels

### Examples

```
> lcd.setfont(lcd.FONT_BIG)  
> lcd.getfontsize()  
16
```

## lcd.getscreensize()

### Description

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

### Syntax

```
lcd.getscreensize()
```

### Parameters

nil

### Returns

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

### Examples

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

## lcd.putpixel()

### Description

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

### Syntax

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

### Parameters

x, y	coordinates of pixel
color	<b>optional</b> : pixel color (default: current foreground color)

### Returns

nil

### Examples

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

## lcd.line()

### Description

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

### Syntax

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

### Parameters

x1,y1	coordinates of line start point
x1,y1	coordinates of line end point
color	<b>optional</b> : line color (default: current foreground color)

### Returns

nil

### Examples

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

## lcd.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

```
lcd.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	<b>optional</b> : rectangle fill color

### Returns

nil

### Examples

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

## lcd.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

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

### Parameters

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

### Returns

nil

### Examples

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

## lcd.triangle()

### Description

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

### Syntax

```
lcd.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	<b>optional:</b> triangle fill color

### Returns

nil

### Examples

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

## lcd.write()

### Description

Write strings and/or numbers to display. Rotation of the displayed text can be set with `lcd.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

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

### Parameters

x: x position (column; 0~screen width-1)  
Special values can be entered:  
    `lcd.CENTER`, centers the text; `lcd.RIGHT`, right justifies the text  
    `lcd.LASTX`, continues from last X position  
y: y position (row; 0~screen height-1)  
Special values can be entered:  
    `lcd.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

```
>lcd.setcolor(lcd.YELLOW)
>lcd.write(0,0,"RePhone")
>t=2.3456
>lcd.write(8,16,"Temp=", {t,2})
```

## lcd.hsb2rgb()

### Description

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

### Syntax

```
Color = lcd.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

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



## lcd.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

```
lcd.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

```
>lcd.rot(lcd.PORTRAIT)
>lcd.clear()
>lcd.image(0,0,128,96,"rephone_128x96.img")
>lcd.rot(lcd.LANDSCAPE)
>lcd.image(0,0,160,128,"rephone_160x128.img")
```

## lcd.bmpimage()

### Description

Shows the image from file. The image file must be in bmp.  
If image dimensions are greater then screen size, the image is cropped.  
Only RGB 24-bit BMP images can be displayed

### Syntax

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

### Parameters

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

### Returns

nil

### Examples

```
>lcd.rot(lcd.PORTRAIT)
>lcd.clear()
>lcd.image(0,0,"rephone.bmp")
```

## lcd.jpgimage()

### Description

Shows the image from file. The image file must be in jpeg.  
If image dimensions are greater then 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.
Colorspace:	YCbCr three components only. Grayscale image is not supported.
Sampling factor:	4:4:4, 4:2:2 or 4:2:0.

### Syntax

```
lcd.jpgimage(x, y, maxscale, filename)
```

### Parameters

x:	x position of the image upper left corner lcd.CENTER, lcd.RIGHT can be used to align image on screen
y:	y position of the image upper left corner lcd.CENTER, lcd.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

```
>lcd.rot(lcd.PORTRAIT)
>lcd.clear()
>lcd.image(0,0,0,"rephone.jpg")
>lcd.rot(lcd.LANDSCAPE)
>lcd.image(0,0,3,"rephone_big.jpeg")
```

## lcd.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 `lcd.setfont()` function.  
It is recommended that all font files are placed in some subdirectory.

### Syntax

```
lcd.compilefont( font_filename)
```

### Parameters

font\_filename:            font source file name

### Returns

nil

### Examples

```
>lcd.compilefont("Ubuntu.c")
```

## lcd.ontouch(cb\_func)

### Description

Register Lua callback function to be executed on touchscreen event.  
For now only available for Xadow displays.

### Syntax

```
lcd.ontouch(cb_func)
```

### Parameters

cb\_func:            callback function to be executed on touch event  
                    Prototype:   **cb\_func(event)**  
                    Event: 1: tap  
                              4: long tap  
                              5: double click

### Returns

nil

### Examples

```
>function cb_touch(event) print("Touch: "..event) end  
>lcd.ontouch(cb_touch)
```