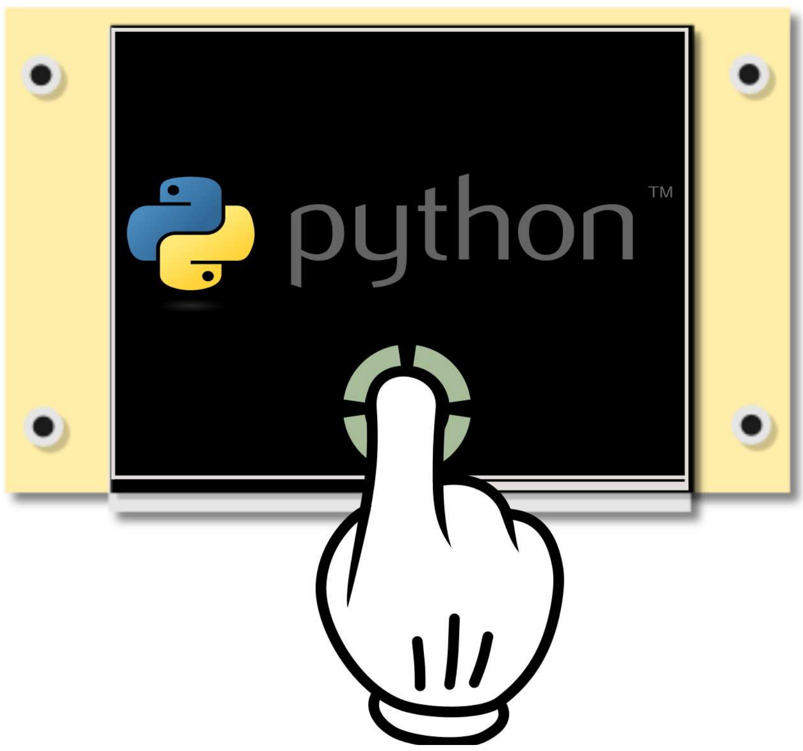
ezLCD Python Module 1.02



Generated by Doxygen 1.8.4

Sat Jul 20 2013 13:53:46

Contents

Chapter 1

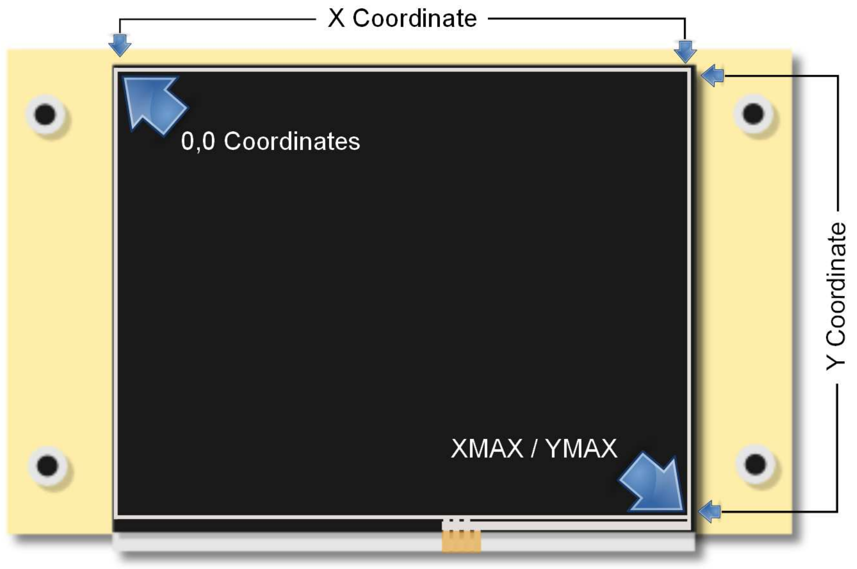
Installing the Module

install info here

requires pySerial `http://pyserial.sourceforge.net/`

Chapter 2

Introduction To The Coordinates System



The ezLCD uses a X Y coordinates system to specify the location for all graphics commands .

One thing to note is that the displays X Y start at 0, so even though you have a display that is 480x272 pixels wide XMAX is 479 and YMAX is 271.

X direction is horizontal across the display starting at the left 0 and ending at the max width of the display.

Y direction is vertical starting at the top 0 and ending at the bottom of the display.

XMAX and YMAX Values for the ezLCD 3xx Line

arLCD 319 239

ezLCD-301 399 239

ezLCD-302

ezLCD-303 319 239

ezLCD-313 319 239

ezLCD-304 479 271

Chapter 3

Introduction To The Hardware

The ezLCD modules contains a GPU an related circutry to drive a LCD display, USB interface

Internal 4mb MSD flash drive for storage of fonts, bitmaps and macros.

Display can be controlled through USB CDC Serial or TTL 3.3v Serial .

Once power is applied to the display it starts up and executes startup.ezm, it will look in /EZUSER/MACROS and if not found will look in /EZSYS/USERS .

What this file does in set all defaults for the Display and communcations port.

Including some default widget fonts and themes.

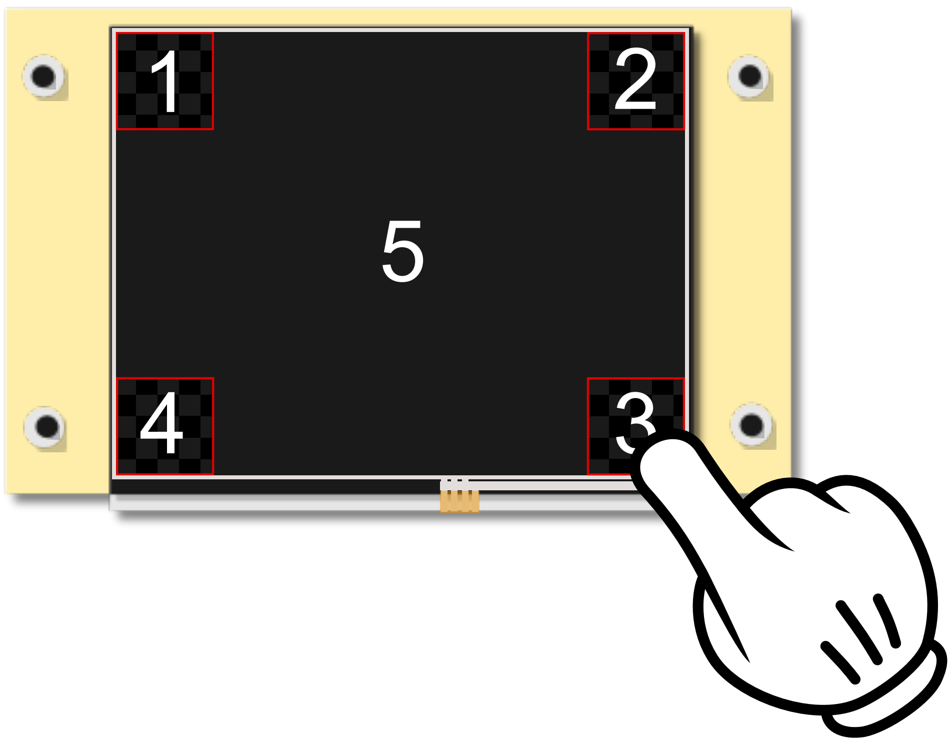
Its best to have a minimal one in the /EZUSER/MACROS directory with only the relevent settings in it .

Sample minimal startup.ezm.

```
'minimal startup.ezm

'Turn off verbose echo of commands
verbose off

'Set command port to USB CDC
cmd cdc

'set some fonts for widgets
fontw 0 0
fontw 1 0
fontw 2 0
fontw 3 serif24
fontw 4 serif24
fontw 5 serif24
fontw 6 serif24
fontw 7 serif24

'Set some themes for widgets
theme 0   1   2 0 0 0  3  3   1 0 0
theme 1 155 152 3 3 3  24  4   5 0 1
theme 2   5  20 3 3 3  4  4   5 0 2
theme 3   9   3 0 0 0  8  8   9 0 3
theme 4   7   3 0 0 0  6  6   6 6 4
theme 5 126 118 3 3 3 35 35  36 0 5
theme 6 111 106 3 3 3 12 12 101 0 6
theme 7  58  48 3 3 3 14 14  54 0 7

color white
print "Python CDC Mode 115200 Baud  "
'print device model
print 65
print "  "
'print firmware version
print 66
```

The ezLCD by default will load startup.ezm but you can have startup1.ezm through startup5.ezm

So if you press the touch screen at power up in any of the areas show below you can execute the other startup macros.


1
2
5
4
3

Chapter 4

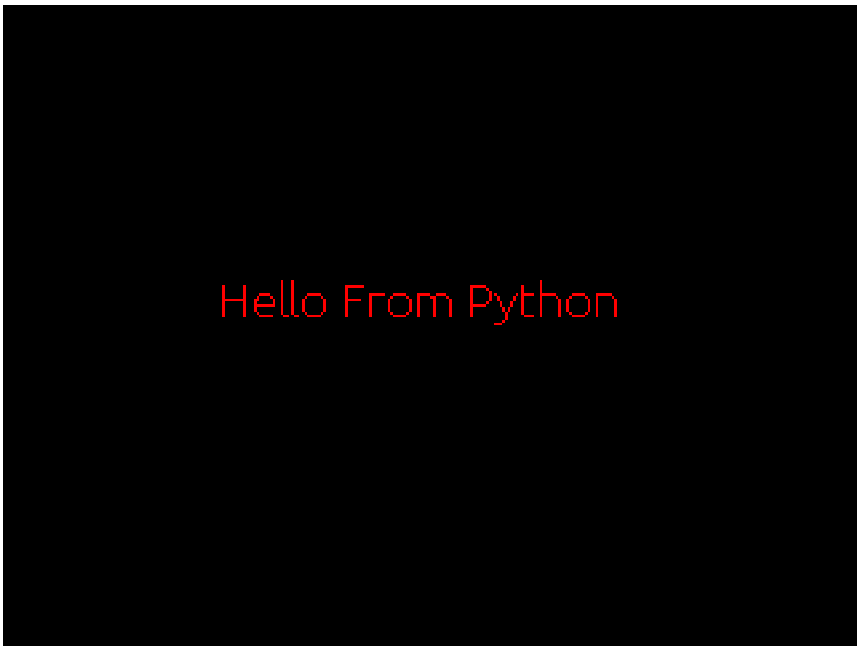
Introduction To The Software

Commands are sent to the ezLCD though the serial interface, Commands are text based and end with a carrage return **cr**.

So if you send **cls** ending with a **cr** the device will clear the screen and return a **cr** when the command is complete,

some widgets take a bit of time (in the millsecond range) to complete so after sending a command allways wait for a **cr** to comeback before sending another command.

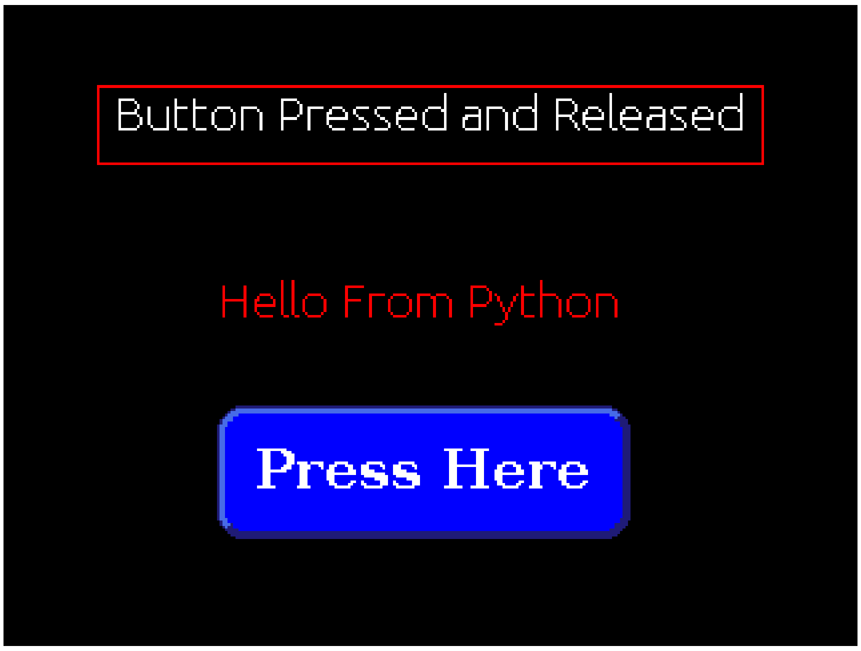
Minimal example will open the ezLCD port clear the screen and print 'Hello From Python' in red



```
# Minimal ezLCD Python demo
#

import platform
import sys


sys.path.append("C:\Users\codeman\Documents\GitHub\ezLCD3xxPython\module")
from ezLCD3xx import *

LCD = ezLCD(None)
comPort =  LCD.findezLCD()

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD(comPort[0][0])
    print comPort[0][1]
    print comPort[0][2]
#Mac
```

```
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose('off')
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# CLear screen
LCD.cls()
# Set draw color to red
LCD.color(BLUE)
# Print string at coordinates x=80 and y=100
LCD.printString("Hello From Python",80,100)

LCD.closeSerial()

```

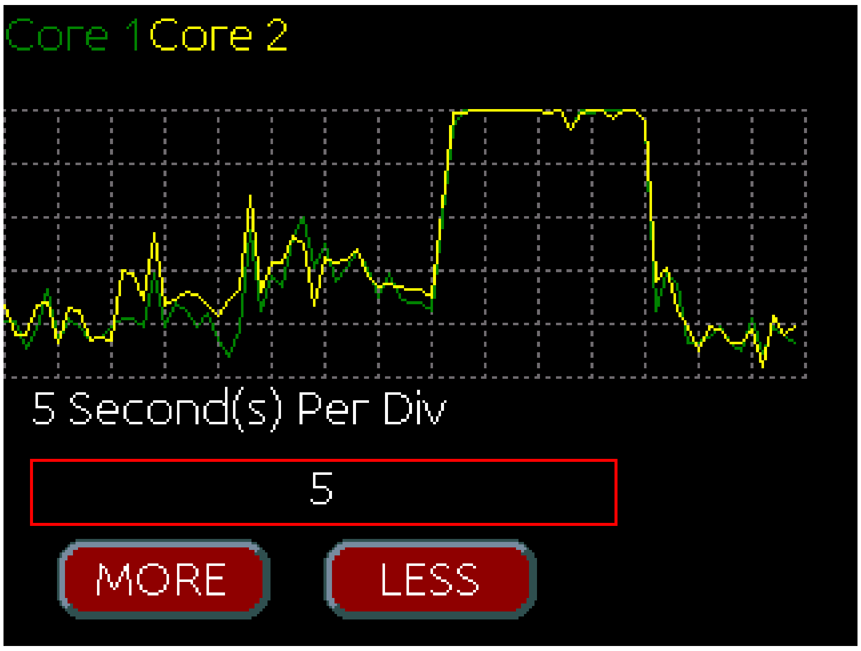
Button example will display a button widget then poll for button presses and update screen



```
# Button ezLCD Python demo
#

import platform
import sys
sys.path.append('module')
from ezLCD3xx import *

LCD = ezLCD(None)
comPort =  LCD.findezLCD()

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD(comPort[0][0])
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose(OFF)
# Turn off button press info from ezLCD
LCD.wquiet(ON)
```

```
# CLear screen
LCD.cls()
# Set draw color to red
LCD.color(RED)
# Set widget font 0
LCD.fontw(0,'1')
# Set wodget font 1
LCD.fontw(1,'0')
# Set theme #1
LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
# Print string at coordinates x=80 and y=100
LCD.printString("Hello From Python",80,100)
# Draw button widget with a ID of 1
LCD.button( 1,  80, 150, 155, 50, 1, 0, 10, 6, 3,'Press Here')
# Draw a staticText box
LCD.staticText(2, 35, 30, 250, 30, 8, 1, 1,'Press Button')
# Clear widget stack
LCD.wstack(CLEAR)
while True:
    # check widget stack this will return widget updates (button press ect.) last in first out order
    (ID, Info, Data) = LCD.wstack(FIFO)
#   print ID, Info, Data
    # check if ID = 1 widget 1 and info = pressed
    if ID == 1 and Info == 4:
        # clear the stack just to be safe
#       LCD.wstack(CLEAR)
        # change draw color to yellow
        LCD.color(YELLOW)
        # change change string 1 for text on static text ID 2
        LCD.string(1,'Button Pressed')
        # redraw static text box ID 2 3=redraw
        LCD.wstate(2, 3)
    # check if ID = 1 widget 1 and info = pressed and released
    if ID == 1 and Info == 1:
        # clear the stack just to be safe
#       LCD.wstack(CLEAR)
        # change draw color to yellow
        LCD.color(YELLOW)
        # change change string 1 for text on static text ID 2
        LCD.string(1,'Button Pressed and Released')
        # redraw static text box ID 2 3=redraw
        LCD.wstate(2, 3)


```

Load example will display the cpu load as a graph



```
#!/usr/bin/env python
# Python Serial library for ezLCD3xx
# http://www.ezlcd.com/
#
# You need the pySerial Library by Chris Liechti
# http://pyserial.wiki.sourceforge.net/pySerial
#
```

```


# END SerLCD Class Definition --------------------------------------

# Start Test Program -----------------------------------------------
import commands
import os
import re
import time as timer
import sys
import platform
import time
import psutil

sys.path.append('module')
from ezLCD3xx import *

def drawGrid():
    LCD.lineType(2)
    LCD.xy(0,30)
    LCD.color(BLACK)
    LCD.box(300,110,1)
    LCD.xy(0,0)
    LCD.color(GREEN)
    LCD.printString('Core 1')
    LCD.color(YELLOW)
    LCD.printString('  Core 2')
    LCD.color(155)
    LCD.color(LIME)
    LCD.font('1')
    LCD.font('0')
    LCD.color(151)
    for y in range(6):
        LCD.xy(0,(y*20)+39)
        LCD.line(300,(y*20)+39)
    for x in range(16):
        LCD.xy(x*20,39)
        LCD.line(x*20,139)
    LCD.xy(300,39)
    LCD.line(300,139)
    LCD.lineType(0)

def drawTime(res):
    LCD.xy(10,140)
    LCD.color(BLACK)
    LCD.box(300,30, FILLED)
    LCD.color(WHITE)
    Time=str(res)+' Second(s) Per Div'
    LCD.printString(Time)

    LCD.string(5, str(res))
    LCD.wstate(7,REDRAW)

LCD = ezLCD(None)
comPort =  LCD.findezLCD()

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD(comPort[0][0])
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')
# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

LCD.ping()
LCD.verbose('OFF')
LCD.wquiet(ON)
LCD.cls()
LCD.fontw(0,'1')
LCD.fontw(1,'0')
LCD.fontw(2,'serif24')
LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
LCD.backlight(100, 5, 10)
LCD.cls()
LCD.font('0')
LCD.fonto(0)
info = ' '
LCD.string( 1, '%')
LCD.color(WHITE)
LCD.cfgio(8,'analog')
```

```
print LCD.xmax()
print LCD.ymax()
print LCD.string(65)
print LCD.string(66)


LCD.button( 5, 20, 200, 80, 30 , 1, 0, 10, 1, 2, 'MORE')
LCD.button( 6, 120, 200, 80, 30 , 1, 0, 10, 1, 3, 'LESS')
LCD.staticText(7, 10, 170, 220, 25, 8, 1, 5, 'test')
drawGrid()
x=0
y1=239
y2=239
lx=0
ly1=239
ly2=239
res=5
drawTime(res)
LCD.wstack(CLEAR)
while True:

    oldinfo = info
    cores=psutil.cpu_percent(interval=1, percpu=True)
    y1 = 139 - cores[0]
    y2 = 139 - cores[1]
    if x!=0:
        LCD.color(GREEN)
        LCD.xy(lx,ly1)
        LCD.line(x, y1)
        LCD.color(YELLOW)
        LCD.xy(lx,ly2)
        LCD.line(x, y2)
    ly1 = y1
    ly2 = y2
    lx = x
    x += 20/res

    if x >= 300:
        x=0
        y1=239
        y2=239
        lx =0
        ly1 =239
        ly2 =239
        drawGrid()
    (ID, info, data) = LCD.wstack(LIFO)
    LCD.wstack(CLEAR)
    if ID == 5 and info==1:
        res +=1
        drawTime(res)
    if ID == 6 and info==1:
        if res > 1:
            res -=1
            drawTime(res)
LCD.closeSerial()
# End Test Program --------------------------------------
```

Chapter 5

Introduction To Themes

Themes will specify the colors used on widgets (buttons, sliders ect)

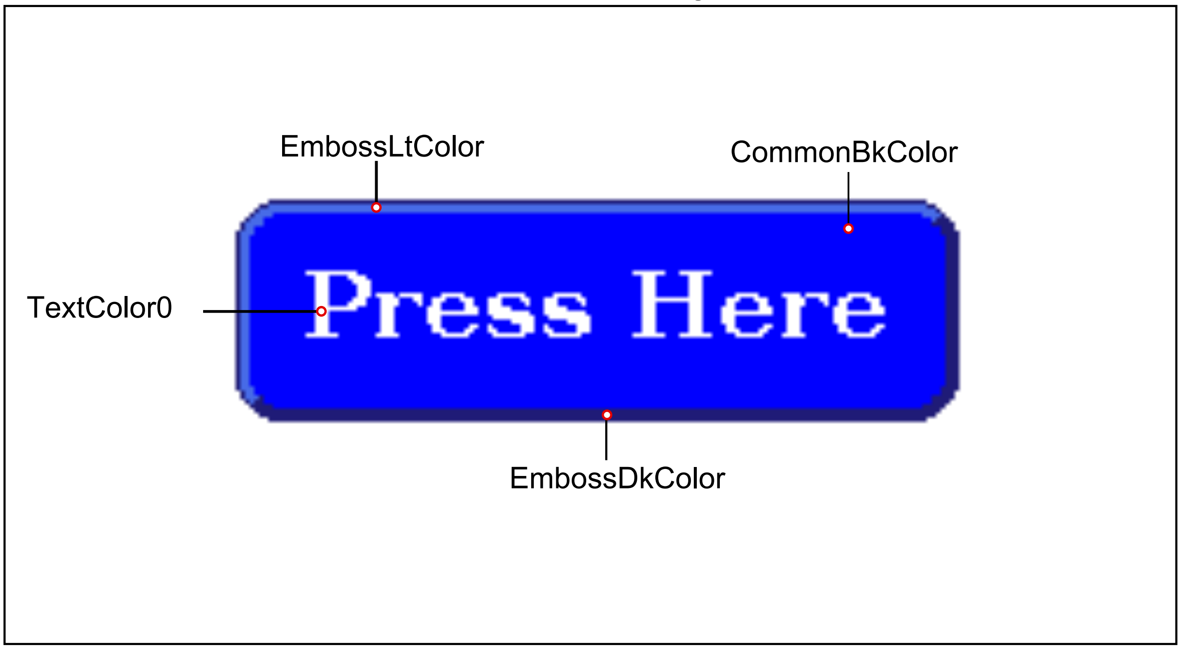
You can have 16 themes numbered 0-15.

```
LCD.theme( 1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1 )
```

Theme Component Description
Theme ID number.
EmbossDkColor Dark emboss color used for 3-D effect of Objects.
EmbossLtColor Light emboss color used for 3-D effect of Objects.
TextColor0
TextColor1
TextColorDisabled Text color used for Objects that are disabled.
Color0
Color1
ColorDisabled Color used to render Objects that are disabled.
CommonBkColor A common background color of Objects.
Font number defined with the fontw command.

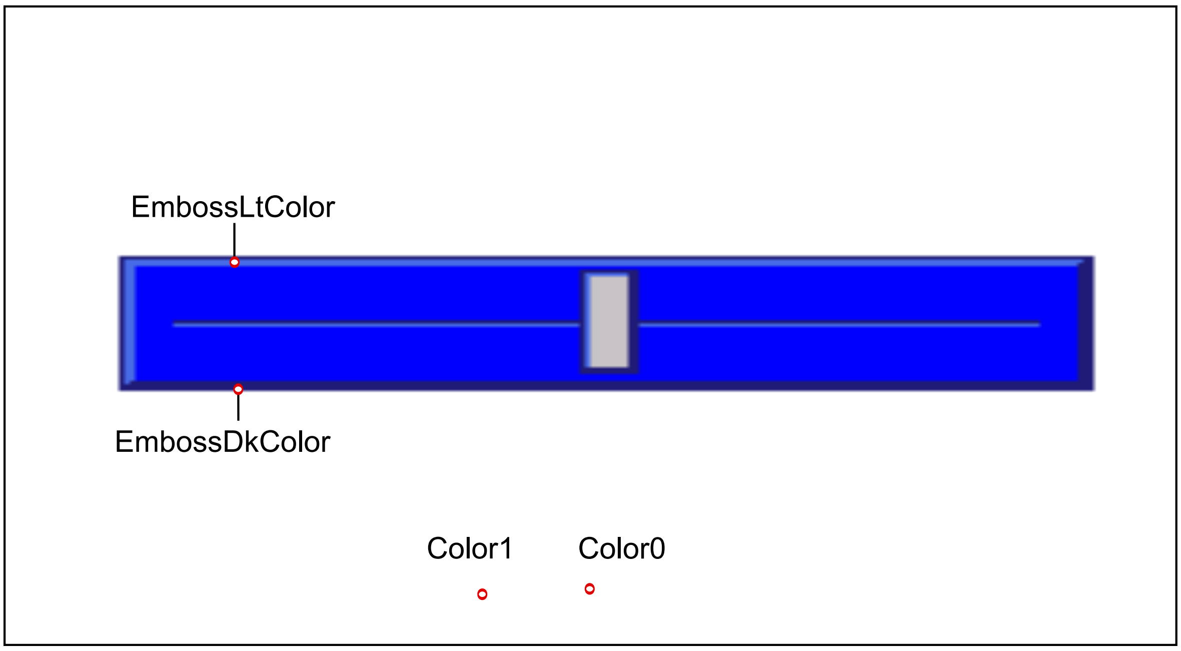
5.1 Themes on Buttons

Theme Button Layout



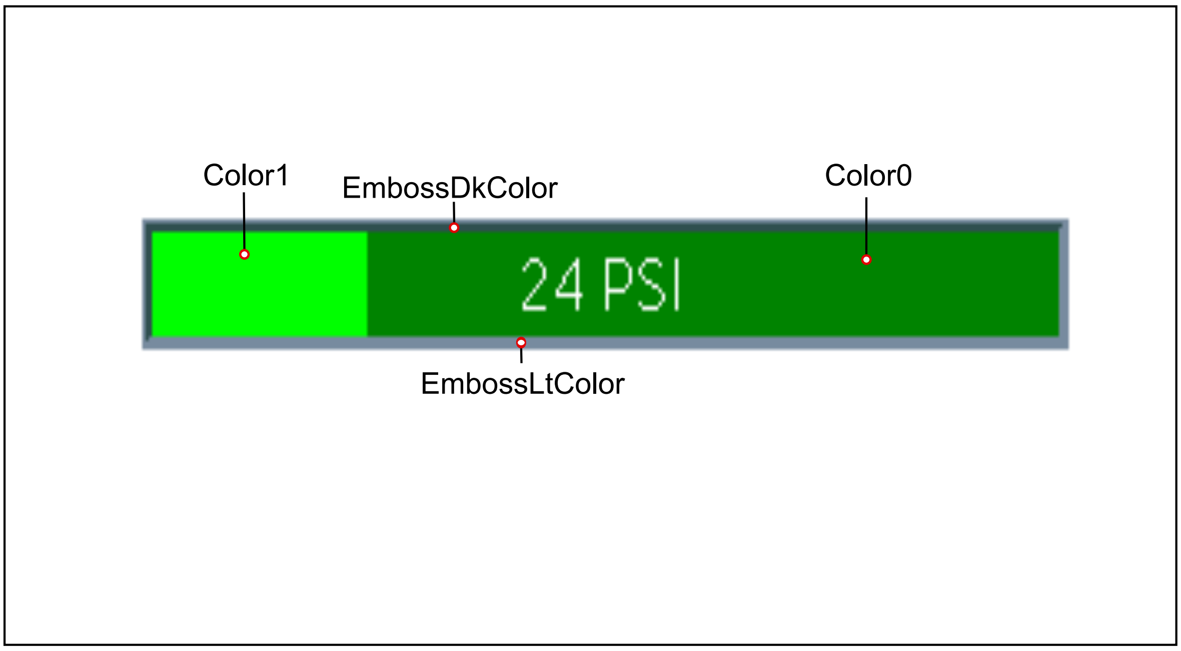
5.2 Themes on Sliders

Theme Slider Layout



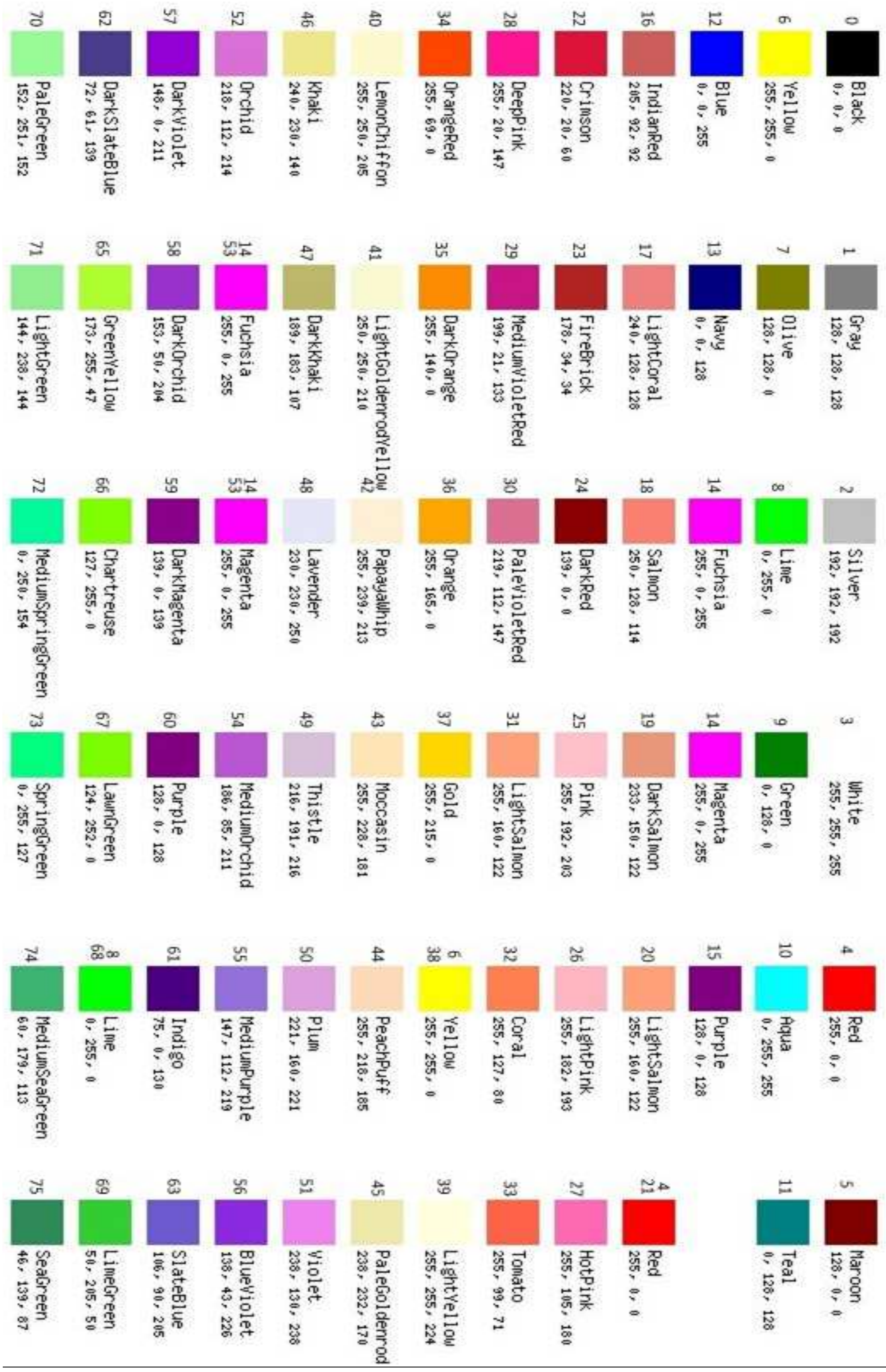
5.3 Themes on Progress Bar

Theme Progress Bar Layout

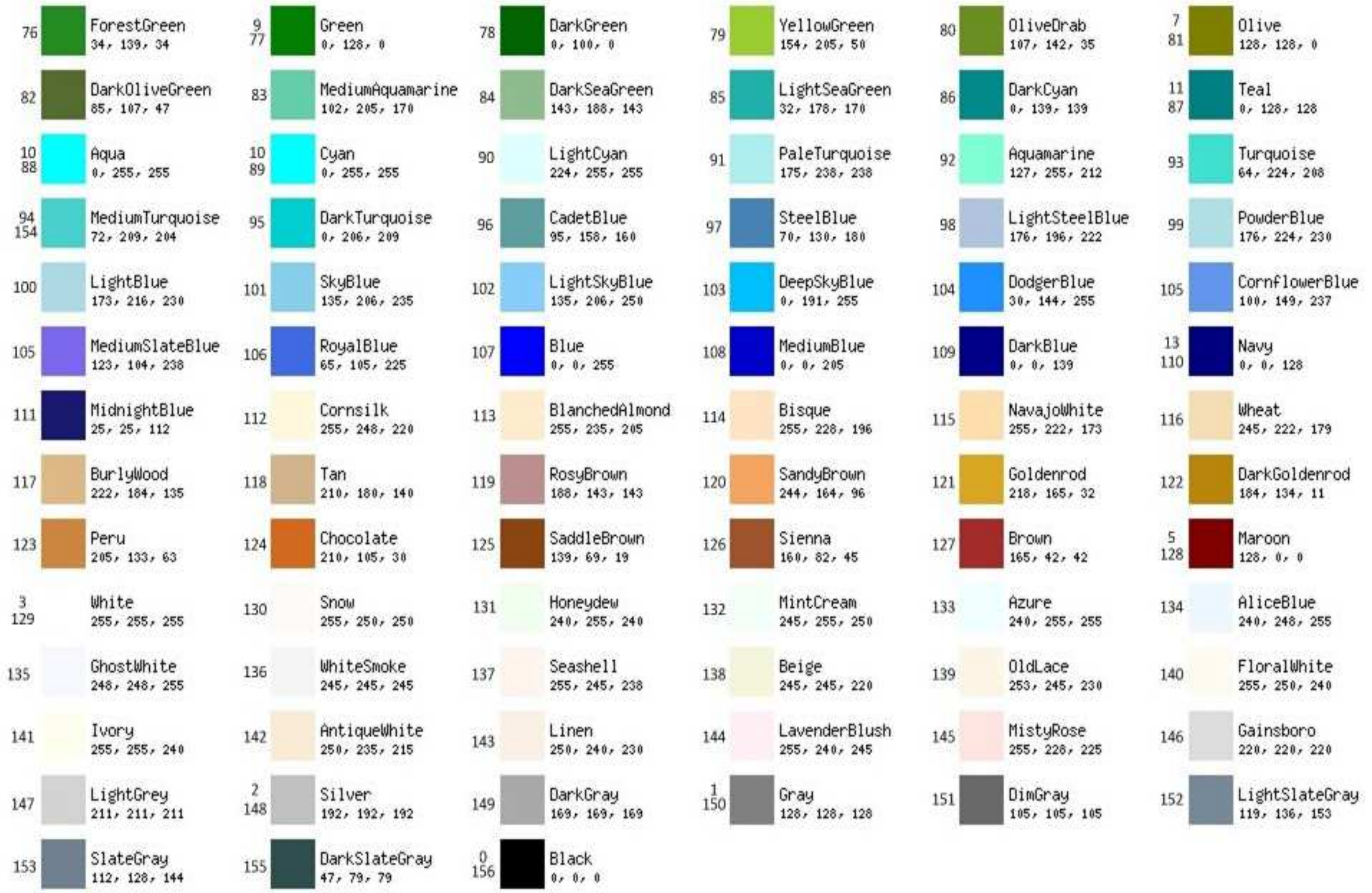


Chapter 6

Color Table



0 Black 0, 0, 0	1 Gray 128, 128, 128	2 Silver 192, 192, 192	3 White 255, 255, 255	4 Red 255, 0, 0	5 Maroon 128, 0, 0
6 Yellow 255, 255, 0	7 Olive 128, 128, 0	8 Lime 0, 255, 0	9 Green 0, 128, 0	10 Aqua 0, 255, 255	11 Teal 0, 128, 128
12 Blue 0, 0, 255	13 Navy 0, 0, 128	14 Fuchsia 255, 0, 255	14 Magenta 255, 0, 255	15 Purple 128, 0, 128	
16 IndianRed 205, 92, 92	17 LightCoral 240, 128, 128	18 Salmon 250, 128, 114	19 DarkSalmon 233, 150, 122	20 LightSalmon 255, 160, 122	4 21 Red 255, 0, 0
22 Crimson 220, 20, 60	23 FireBrick 178, 34, 34	24 DarkRed 139, 0, 0	25 Pink 255, 192, 203	26 LightPink 255, 182, 193	27 HotPink 255, 105, 180
28 DeepPink 255, 20, 147	29 MediumVioletRed 199, 21, 133	30 PaleVioletRed 219, 112, 147	31 LightSalmon 255, 160, 122	32 Coral 255, 127, 80	33 Tomato 255, 99, 71
34 OrangeRed 255, 69, 0	35 DarkOrange 255, 140, 0	36 Orange 255, 165, 0	37 Gold 255, 215, 0	6 38 Yellow 255, 255, 0	39 LightYellow 255, 255, 224
40 LemonChiffon 255, 250, 205	41 LightGoldenrodYellow 250, 250, 210	42 PapayaWhip 255, 239, 213	43 Moccasin 255, 228, 181	44 PeachPuff 255, 218, 185	45 PaleGoldenrod 238, 232, 170
46 Khaki 240, 230, 140	47 DarkKhaki 189, 183, 107	48 Lavender 230, 230, 250	49 Thistle 216, 191, 216	50 Plum 221, 160, 221	51 Violet 238, 130, 238
52 Orchid 218, 112, 214	14 53 Fuchsia 255, 0, 255	14 53 Magenta 255, 0, 255	54 MediumOrchid 186, 85, 211	55 MediumPurple 147, 112, 219	56 BlueViolet 138, 43, 226
57 DarkViolet 148, 0, 211	58 DarkOrchid 153, 50, 204	59 DarkMagenta 139, 0, 139	60 Purple 128, 0, 128	61 Indigo 75, 0, 130	63 SlateBlue 106, 90, 205
62 DarkSlateBlue 72, 61, 139	65 GreenYellow 173, 255, 47	66 Chartreuse 127, 255, 0	67 LawnGreen 124, 252, 0	8 68 Lime 0, 255, 0	69 LimeGreen 50, 205, 50
70 PaleGreen 152, 251, 152	71 LightGreen 144, 238, 144	72 MediumSpringGreen 0, 250, 154	73 SpringGreen 0, 255, 127	74 MediumSeaGreen 60, 179, 113	75 SeaGreen 46, 139, 87



76	ForestGreen 34, 139, 34	9 77	Green 0, 128, 0	78	DarkGreen 0, 100, 0	79	YellowGreen 154, 205, 50	80	OliveDrab 107, 142, 35	7 81	Olive 128, 128, 0
82	DarkOliveGreen 85, 107, 47	83	MediumAquamarine 102, 205, 170	84	DarkSeaGreen 143, 188, 143	85	LightSeaGreen 32, 178, 170	86	DarkCyan 0, 139, 139	11 87	Teal 0, 128, 128
10 88	Aqua 0, 255, 255	10 89	Cyan 0, 255, 255	90	LightCyan 224, 255, 255	91	PaleTurquoise 175, 238, 238	92	Aquamarine 127, 255, 212	93	Turquoise 64, 224, 208
94 154	MediumTurquoise 72, 209, 204	95	DarkTurquoise 0, 206, 209	96	CadetBlue 95, 158, 160	97	SteelBlue 70, 130, 180	98	LightSteelBlue 176, 196, 222	99	PowderBlue 176, 224, 230
100	LightBlue 173, 216, 230	101	SkyBlue 135, 206, 235	102	LightSkyBlue 135, 206, 250	103	DeepSkyBlue 0, 191, 255	104	DodgerBlue 30, 144, 255	105	CornflowerBlue 100, 149, 237
105	MediumSlateBlue 123, 104, 238	106	RoyalBlue 65, 105, 225	107	Blue 0, 0, 255	108	MediumBlue 0, 0, 205	109	DarkBlue 0, 0, 139	13 110	Navy 0, 0, 128
111	MidnightBlue 25, 25, 112	112	Cornsilk 255, 248, 220	113	BlanchedAlmond 255, 235, 205	114	Bisque 255, 228, 196	115	NavajoWhite 255, 222, 173	116	Wheat 245, 222, 179
117	BurlyWood 222, 184, 135	118	Tan 210, 180, 140	119	RosyBrown 188, 143, 143	120	SandyBrown 244, 164, 96	121	Goldenrod 218, 165, 32	122	DarkGoldenrod 184, 134, 11
123	Peru 205, 133, 63	124	Chocolate 210, 105, 30	125	SaddleBrown 139, 69, 19	126	Sienna 160, 82, 45	127	Brown 165, 42, 42	5 128	Maroon 128, 0, 0
3 129	White 255, 255, 255	130	Snow 255, 250, 250	131	Honeydew 240, 255, 240	132	MintCream 245, 255, 250	133	Azure 240, 255, 255	134	AliceBlue 240, 248, 255
135	GhostWhite 248, 248, 255	136	WhiteSmoke 245, 245, 245	137	Seashell 255, 245, 238	138	Beige 245, 245, 220	139	OldLace 253, 245, 230	140	FloralWhite 255, 250, 240
141	Ivory 255, 255, 240	142	AntiqueWhite 250, 235, 215	143	Linen 250, 240, 230	144	LavenderBlush 255, 240, 245	145	MistyRose 255, 228, 225	146	Gainsboro 220, 220, 220
147	LightGrey 211, 211, 211	2 148	Silver 192, 192, 192	149	DarkGray 169, 169, 169	1 150	Gray 128, 128, 128	151	DimGray 105, 105, 105	152	LightSlateGray 119, 136, 153
153	SlateGray 112, 128, 144	155	DarkSlateGray 47, 79, 79	0 156	Black 0, 0, 0						

Chapter 7

Introduction To Fonts

Chapter 8

Introduction To Bitmaps

picture supports gif, jpg and bmp

Chapter 9

Introduction To Widgets

9.1 Over View of Widgets

text text

9.2 Buttons

text text

9.3 TouchZone

TouchZones work like buttons but do not display any graphics on their own

You have to supply the image.

So we can take the image below and can make 21 TouchZones one for each Emoticon

9.4 Slider

text text

9.5 ProgressBar

text text

9.6 Gauge

text text

9.7 AnalogMeter

text text

9.8 DigitalMeter

text text

9.9 StaticText

text text

9.10 GroupBox

text text

9.11 Dial

text text

9.12 Choice

text text

9.13 CheckBox

text text

9.14 Radio Buttons

text text

Chapter 10

Examples

Chapter 11

Module Index

11.1 Modules

Here is a list of all modules:

Chapter 12

Namespace Index

12.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 13

Hierarchical Index

13.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 14

Class Index

14.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 15

Module Documentation

15.1 Commands

Functions

- def module.ezLCD3xx.direct

    The direct command will send a string direct to the GPU.
- def module.ezLCD3xx.verbose

    The Verbose command will turn on or off more verbose errors.
- def module.ezLCD3xx.xmax

    The xmax command will return the max x of current display.
- def module.ezLCD3xx.ymax

    The ymax command will return the max y of current display.
- def module.ezLCD3xx.ping

    the ping command
- def module.ezLCD3xx.backlight

    The backlight command will set backlight brightness and timeout.
- def module.ezLCD3xx.wquiet

    The wquiet command disables the touch event data being sent to the console port.
- def module.ezLCD3xx.cfgio

    The cfgio command will configure io pins.
- def module.ezLCD3xx.io

    The io command use to set and clear io pins.
- def module.ezLCD3xx.play

    The play command will play a macro stored on the drive of the ezLCD.
- def module.ezLCD3xx.run

    The run command will run a macro stored on the drive of the ezLCD.
- def module.ezLCD3xx.reset

    The reset command will reset the ezLCD and run startup.ezm same as power up.
- def module.ezLCD3xx.snapshot

    The snapshot command will write a copy of the current display to the flash drive as a bmp.
- def module.ezLCD3xx.calibrate

    The calibrate command will re calibrate the touch screen.

15.1.1 Detailed Description

15.1.2 Function Documentation

15.1.2.1 def module.ezLCD3xx.backlight (*self*, *brightness*, *timeout =* `None`, *level =* `None`)

The backlight command will set backlight brightness and timeout.

Parameters

brightness	1
timeout	2
level	3

15.1.2.2 def module.ezLCD3xx.cfgio (*self, pin, function*)

The cfgio command will configure io pins.

Parameters

pin	
function	

15.1.2.3 def module.ezLCD3xx.direct (*self, string*)

The direct command will send a string direct to the GPU.

Parameters

string	string to send

15.1.2.4 def module.ezLCD3xx.io (*self, pin, level* = `None`)

The io command use to set and clear io pins.

Parameters

pin	
level	

Returns

io level

15.1.2.5 def module.ezLCD3xx.ping (*self*)

the ping command

Returns

0

15.1.2.6 def module.ezLCD3xx.play (*self, filename*)

The play command will play a macro stored on the drive of the ezLCD.

Parameters

filename	macro filename

15.1.2.7 def module.ezLCD3xx.reset (*self*)

The reset command will reset the ezLCD and run startup.ezm same as power up.

15.1.2.8 def module.ezLCD3xx.run (*self, filename*)

The run command will run a macro stored on the drive of the ezLCD.

Parameters

filename	macro filename

15.1.2.9 def module.ezLCD3xx.snapshot (*self, x, y, w, h, filename*)

The snapshot command will write a copy of the current display to the flash drive as a bmp.

Parameters

x	starting x position
y	starting y position
w	width
h	height
filename	filename.bmp Make sure you have space on the internal flash drive !

15.1.2.10 def module.ezLCD3xx.verbose (*self, state*)

The Verbose command will turn on or off more verbose errors.

Parameters

state	0=off 1=on

15.1.2.11 def module.ezLCD3xx.wquiet (*self, state*)

The wquiet command disables the touch event data being sent to the console port.

Parameters

state	0=off 1=on

15.1.2.12 def module.ezLCD3xx.xmax (*self*)

The xmax command will return the max x of current display.

Returns

x-horizontal resolution in pixels starting from 0

15.1.2.13 def module.ezLCD3xx.ymax (*self*)

The ymax command will return the max y of current display.

Returns

y-vertical resolution in pixels starting from 0

15.2 Primitve Drawing Commands

Functions

- def module.ezLCD3xx.cls

  The cls command will clear the screen to black it no color is given.
- def module.ezLCD3xx.color

  The color command see ezLCD3xx manual for colors.
- def module.ezLCD3xx.colorId

  The colorId command.
- def module.ezLCD3xx.xy

  The xy command will set or return the x y coordinates.
- def module.ezLCD3xx.plot

  The plot command will set a pixel to current color and if used x y.
- def module.ezLCD3xx.lineType

  The lineType Command will set the line type for the line command.
- def module.ezLCD3xx.lineWidth

  The lineWidth Command will set the line width for the line command.
- def module.ezLCD3xx.line

  The line command will draw a line from current xy to line(x,y)
- def module.ezLCD3xx.box

  The box command will draw a box starting from the current xy in width and height with option for filled.
- def module.ezLCD3xx.circle

  The circle command will draw a circle in the current xy with radius and optional filled.
- def module.ezLCD3xx.pie

  The pie command will draw a pie slice at current xy.
- def module.ezLCD3xx.arc

  The arc command will draw a arc i the current xy optional filled.
- def module.ezLCD3xx.clipArea

  The cliparea command allows you to designate a rectangular/box area that you can draw in.
- def module.ezLCD3xx.clipEnable

  The clipenable command enables or disables cliparea.

15.2.1 Detailed Description

15.2.2 Function Documentation

15.2.2.1 def module.ezLCD3xx.arc (*self*, *radius*, *start*, *end*, *fill* = 0)

The arc command will draw a arc i the current xy optional filled.

Parameters

radius	radius of arc
start	start angle
end	end angle
fill	1=filled arc 0=outline only ∗optional defaults to outline

15.2.2.2 def module.ezLCD3xx.box (*self*, *width*, *height*, *fill* = 0)

The box command will draw a box starting from the current xy in width and height with option for filled.

Parameters

width	width of box in pixels
height	height of box in pixels
fill	1=filled box 0=outline only ∗optional defaults to outline

15.2.2.3 def module.ezLCD3xx.circle (*self*, *radius*, *fill* = `0`)

The circle command will draw a circle in the current xy with radius and optional filled.

Parameters

radius	radius of circle
fill	1=filled circle 0=outline only ∗optional defaults to outline

15.2.2.4 def module.ezLCD3xx.clipArea (*self*, *left*, *top*, *right*, *bottom*)

The cliparea command allows you to designate a rectangular/box area that you can draw in.

Any surrounding area will be protected and no changes can be made to it

Parameters

left	
top	
right	
bottom	

15.2.2.5 def module.ezLCD3xx.clipEnable (*self*, *enable*)

The clipenable command enables or disables cliparea.

Parameters

enable	0=off 1=on

15.2.2.6 def module.ezLCD3xx.cls (*self*, *Color* = `None`)

The cls command will clear the screen to black it no color is given.

Parameters

Color	color to clear screen to

15.2.2.7 def module.ezLCD3xx.color (*self*, *color* = `None`)

The color command see ezLCD3xx manual for colors.

Parameters

color	number

Returns

color as a tuple

15.2.2.8 def module.ezLCD3xx.colorId (*self*, *ID*, *R* = `None`, *G* = `None`, *B* = `None`)

The colorId command.

Parameters

ID	color ID number
R	Red Value
G	Green Value
B	Blue Value

Returns

color as a tuple if r g b is None

15.2.2.9 def module.ezLCD3xx.line (*self, x, y*)

The line command will draw a line from current xy to line(x,y)

Parameters

x	
y	

15.2.2.10 def module.ezLCD3xx.lineType (*self, option*)

The lineType Command will set the line type for the line command.

Parameters

option	0 = solid, 1= dotted (1 pixel spacing between dots), 2 = dashed (2 pixel spacing between dashes)

15.2.2.11 def module.ezLCD3xx.lineWidth (*self, width*)

The lineWidth Command will set the line width for the line command.

Parameters

width	thin line (width = 1) or a thick line (width =3). Only [width] = 1 or 3 are available.

15.2.2.12 def module.ezLCD3xx.pie (*self, radius, start, end*)

The pie command will draw a pie slice at current xy.

Parameters

radius	radius of pie
start	start angle
end	end angle

15.2.2.13 def module.ezLCD3xx.plot (*self, x =* `None`*, y =* `None`)

The plot command will set a pixel to current color and if used x y.

Parameters

x	optional
y	optional

15.2.2.14 def module.ezLCD3xx.xy (*self*, *x* = None, *y* = None)

The xy command will set or return the x y coordinates.

Parameters

x	x position
y	y position

Returns

x y if x and y not supplied

```
# Set x y to 100 100
LCD.xy(100,100)
# Get Current x y
(x,y)=LCD.xy()
```

15.3 Widgets

Functions

- def module.ezLCD3xx.ameter

  The ameter widget.
- def module.ezLCD3xx.ameter_color

  The ameter_color command.
- def module.ezLCD3xx.dmeter

  The dmeter widget.
- def module.ezLCD3xx.button

  The button widget.
- def module.ezLCD3xx.choice

  The choice widget allows you to print a string and display buttons for the user to choose a response.
- def module.ezLCD3xx.groupBox

  The groupBox widget.
- def module.ezLCD3xx.radioButton

  The radioButton widget.
- def module.ezLCD3xx.staticText

  The staticText widget.
- def module.ezLCD3xx.slider

  The slider widget.
- def module.ezLCD3xx.progressBar

  The progressBar widget.
- def module.ezLCD3xx.gauge

  The gauge widget.
- def module.ezLCD3xx.touchZone

  The touchZone command.
- def module.ezLCD3xx.dial

  The dial widget.
- def module.ezLCD3xx.theme

  The theme command sets the colors for widgets.
- def module.ezLCD3xx.fontw

  The fontW command will set the font for widget.
- def module.ezLCD3xx.string

  The string command will set or return a internal string.
- def module.ezLCD3xx.wstack

  The wstack command will return the stack of widgets pressed 32 levels.
- def module.ezLCD3xx.wvalue

  The wvalue command will set or return a value to or from a widget.
- def module.ezLCD3xx.wstate

  The wstate command.

15.3.1 Detailed Description

15.3.2 Function Documentation

15.3.2.1 def module.ezLCD3xx.ameter (*self*, *ID*, *x*, *y*, *width*, *height*, *options*, *value*, *minV*, *maxV*, *theme*, *stringID*, *meterType* = `0`, *text* = `None`)

The ameter widget.

Parameters

ID	
x	
y	
width	
height	
options	
value	
minV	
maxV	
theme	
stringID	
meterType	

15.3.2.2 def module.ezLCD3xx.ameter_color (*self*, *ID*, *color1*, *color2*, *color3*, *color4*, *color5*, *color6*)

The ameter_color command.

Parameters

ID	
color1	
color2	
color3	
color4	
color5	
color6	

15.3.2.3 def module.ezLCD3xx.button (*self*, *ID*, *x*, *y*, *width*, *height*, *options*, *align*, *radius*, *theme*, *stringID*, *text* = `None`)

The button widget.

Parameters

ID	
x	
y	
width	
height	
options	
align	
radius	
theme	
stringID	
text	optional text for button

15.3.2.4 def module.ezLCD3xx.choice (*self*, *string*, *theme*, *string1* = `None`, *string2* = `None`, *string3* = `None`)

The choice widget allows you to print a string and display buttons for the user to choose a response.

Parameters

string	the text about the buttons
theme	the theme ID
string1	string for left button ∗optional defaults to YES
string2	string for center button ∗optional defaults to NO
string3	string for right button ∗optional defaults to CANCEL

Returns

1=left button
0=center button
-1=right button

15.3.2.5 def module.ezLCD3xx.dial (*self, ID, x, y, radius, option, resolution, value, maxx, theme*)

The dial widget.

Parameters

ID	
x	
y	
radius	
option	
resolution	
value	
maxx	
theme	

15.3.2.6 def module.ezLCD3xx.dmeter (*self, ID, x, y, width, height, options, value, digits, dp, theme*)

The dmeter widget.

Parameters

ID	
x	
y	
width	
height	
options	
value	
digits	
dp	
theme	

15.3.2.7 def module.ezLCD3xx.fontw (*self, fontnumber, name*)

The fontW command will set the font for widget.

Parameters

fontnumber	number of the font
name	filename of font '0' and '1' are internal fonts

15.3.2.8 def module.ezLCD3xx.gauge (*self*, *ID*, *x*, *y*, *width*, *height*, *options*, *initial*, *mmin*, *mmax*, *theme*, *stringID* = `None`, *text* = `None`)

The gauge widget.

Parameters

ID	
x	
y	
width	
height	
options	
value	
mmax	
theme	
stringID	
text	

15.3.2.9 def module.ezLCD3xx.groupBox (*self*, *ID*, *x*, *y*, *width*, *height*, *options*, *theme*, *stringID*)

The groupBox widget.

Parameters

ID	
x	
y	
width	
height	
options	
theme	
stringID	

15.3.2.10 def module.ezLCD3xx.progressBar (*self*, *ID*, *x*, *y*, *width*, *height*, *options*, *value*, *mmax*, *theme*, *stringID*, *text* = `None`)

The progressBar widget.

Parameters

ID	
x	
y	
width	
height	

options	
value	
mmax	
theme	
stringID	
text	

15.3.2.11 def module.ezLCD3xx.radioButton (*self, ID, x, y, width, height, options, theme, stringID*)

The radioButton widget.

Parameters

ID	
x	
y	
width	
height	
options	Options: 1=draw , 2=disabled, 3=checked, 4=first, 5=first and checked.
theme	
stringID	

15.3.2.12 def module.ezLCD3xx.slider (*self, ID, x, y, width, height, options, rrange, resolution, value, theme*)

The slider widget.

Parameters

ID	
x	
y	
width	
height	
options	
rrange	
resolution	
value	
theme	

15.3.2.13 def module.ezLCD3xx.staticText (*self, ID, x, y, width, height, options, theme, stringID, text =* `None`)

The staticText widget.

Parameters

ID	
x	
y	
width	
height	
options	Options: 1=left, 2=disabled , 3=right , 4=center, 5=left framed, 6=disabled framed, 7=right framed, 8=center framed , 9=redraw text.

theme	theme
stringID	stringID number
text	text to display ∗optional

15.3.2.14 def module.ezLCD3xx.string (*self, stringID, string* = `None`)

The string command will set or return a internal string.

Parameters

stringID	number of string to set or return
string	string to set optional internal strings are used for text on buttons and other widgets Strings are defined as 128 characters. There are 64 strings (0 to 63). String 61-63 are used by the CHOICE command. String 64 is temp location. String 65 is the product string String 66 is the firmware string string cmd = 16

15.3.2.15 def module.ezLCD3xx.theme (*self, ID, EmbossDkColor, EmbossLtColor, TextColor0, TextColor1, TextColorDisabled, Color0, Color1, ColorDisabled, CommonBkColor, Fontw*)

The theme command sets the colors for widgets.

Parameters

ID	Theme ID
EmbossDkColor	Dark color for 3d effect
EmbossLtColor	Light color for 3d effect
TextColor0	
TextColor1	
TextColor-Disabled	
Color0	
Color1	
ColorDisabled	
CommonBk-Color	
Fontw	widget font for theme

15.3.2.16 def module.ezLCD3xx.touchZone (*self, ID, x, y, width, height, options*)

The touchZone command.

Parameters

ID	
x	
y	
width	
height	

options	

15.3.2.17 def module.ezLCD3xx.wstack (*self*, *option*)

The wstack command will return the stack of widgets pressed 32 levels.

Parameters

option	0=FIFO 1=LIFO 2=CLEAR FIFO Fist in Fist out LIFO Last in First out CLEAR Clear the stack

Returns

truple of ID, Info, Data
Button Widget Values
- ID = widgetID of widget pressed
- Info 1=Pressed and released 2=Cancel 4=Pressed
- Data button state
TouchZone Widget Vaules
- ID = widgetID of widget pressed
- Info 1=Pressed and released 2=Cancel 4=Pressed
- Data button state
Slider Widget Values
- ID = widgetID of widget pressed
- Info 1 = value incremented 2 = value decremented
- Data slider value
CheckBox Widget Vaules
- ID = widgetID of widget pressed
- Info 4 = checked 1 = unchecked
- Data state
Dial Widget Vaules
- ID = widgetID of widget pressed
- Info 1 = turned clockwise 2 = turned counter-clockwise
- Data dial value

```
# check wstack for button presses
(ID, Info, Data) = LCD.wstack(LIFO)
```

15.3.2.18 def module.ezLCD3xx.wstate (*self*, *ID*, *option*)

The wstate command.

Parameters

ID	widget ID
option	0 = delete, 1 = enable, 2 = disable, 3 = redraw

15.3.2.19 def module.ezLCD3xx.wvalue (*self*, *ID*, *value* = `None`)

The wvalue command will set or return a value to or from a widget.

Parameters

ID	
value	

15.4 Bitmaps and Fonts

Functions

- def module.ezLCD3xx.picture

  The picture command will display a bitmap in bmp, jpg, gif formats with optional coordinates.
- def module.ezLCD3xx.font

  The font command will set current font to use for printString fonts are located in the /EZSYS/FONTS and /EZUSER/-FONTS
  use the ezLCD-3xx Font Converter from earthlcd.com
  to convert truetype fonts to ezLCD format
  internal fonts will display faster than external fonts.
- def module.ezLCD3xx.fonto

  The FONTO command will change the orientation or direction the text prints.
- def module.ezLCD3xx.printString

  print string in current color and font and optional coordinates

15.4.1 Detailed Description

15.4.2 Function Documentation

15.4.2.1 def module.ezLCD3xx.font (*self*, *font*)

The font command will set current font to use for printString fonts are located in the /EZSYS/FONTS and /EZUSE-R/FONTS

use the ezLCD-3xx Font Converter from earthlcd.com

to convert truetype fonts to ezLCD format

internal fonts will display faster than external fonts.

Parameters

font	font name
	'0' and '1' are internal fonts '0' is medium and '1' is small
	`# Set font to internal medium font`
	`LCD.font('0')`
	`# Set font to LCD24`
	`LCD.font('LCD24')`

15.4.2.2 def module.ezLCD3xx.fonto (*self*, *orientation* = `None`)

The FONTO command will change the orientation or direction the text prints.

Parameters

orientation	0 90 180 270

Returns

orientation current orientation if orientation is not suppled

```
LCD.fonto(0)
LCD.color(YELLOW)
LCD.printString('Hello',100,100)
LCD.fonto(90)
LCD.color(RED)
LCD.printString('Hello',100,100)
LCD.fonto(180)
LCD.color(BLUE)
LCD.printString('Hello',100,100)
 LCD.fonto(270)
 LCD.color(GREEN)
 LCD.printString('Hello',100,100)
```

15.4.2.3 def module.ezLCD3xx.picture (*self*, *image*, *x* = None, *y* = None)

The picture command will display a bitmap in bmp, jpg, gif formats with optional coordinates.

Parameters

image	filename of image 'logo.gif'
x	x coordinates

y	y coordinates
	x y are optional and if not supplied will display image at current xy
	`# display python.gif at 10 10`
	`LCD.picture('python.gif',10,10)`
	`# display python.gif at current x y`
	`LCD.picture('python.gif')`
	image cmd = 24

15.4.2.4 def module.ezLCD3xx.printString (*self*, *string*, *x* = None, *y* = None, *orientation* = None)

print string in current color and font and optional coordinates

Parameters

string	string to print
x	x coordinates
y	y coordinates

orientation	rotate text direction
	x y are optional and if not supplied will print string at current xy
	orientation is optional but if used x y must be supplied
	∗∗ orientation will be restored to previous orientation after printing string ∗∗
	`# display string 'Hello World' at 10 10`
	`LCD.printString('Hello World',10,10)`
	`# display string 'Hello World' at current x y`
	`LCD.printString('Hello World')`
	`# diplay string 'Hello World' at 10 10 rotated 90`
	`LCD.printString('Hello World',10,10,90)`

Chapter 16

Namespace Documentation

16.1 module.ezLCD3xx Namespace Reference

Classes

- class ezLCD

Functions

- def __init__

  ezLCD object
- def **findezLCD**
- def **openSerial**
- def **closeSerial**
- def WaitForCR

  This is a internal use function.
- def direct

  The direct command will send a string direct to the GPU.
- def verbose

  The Verbose command will turn on or off more verbose errors.
- def xmax

  The xmax command will return the max x of current display.
- def ymax

  The ymax command will return the max y of current display.
- def ping

  the ping command
- def backlight

  The backlight command will set backlight brightness and timeout.
- def wquiet

  The wquiet command disables the touch event data being sent to the console port.
- def cfgio

  The cfgio command will configure io pins.
- def io

  The io command use to set and clear io pins.
- def play

  The play command will play a macro stored on the drive of the ezLCD.
- def run

The run command will run a macro stored on the drive of the ezLCD.

- def reset

  The reset command will reset the ezLCD and run startup.ezm same as power up.
- def snapshot

  The snapshot command will write a copy of the current display to the flash drive as a bmp.
- def calibrate

  The calibrate command will re calibrate the touch screen.
- def cls

  The cls command will clear the screen to black it no color is given.
- def color

  The color command see ezLCD3xx manual for colors.
- def colorId

  The colorId command.
- def xy

  The xy command will set or return the x y coordinates.
- def plot

  The plot command will set a pixel to current color and if used x y.
- def lineType

  The lineType Command will set the line type for the line command.
- def lineWidth

  The lineWidth Command will set the line width for the line command.
- def line

  The line command will draw a line from current xy to line(x,y)
- def box

  The box command will draw a box starting from the current xy in width and height with option for filled.
- def circle

  The circle command will draw a circle in the current xy with radius and optional filled.
- def pie

  The pie command will draw a pie slice at current xy.
- def arc

  The arc command will draw a arc i the current xy optional filled.
- def clipArea

  The cliparea command allows you to designate a rectangular/box area that you can draw in.
- def clipEnable

  The clipenable command enables or disables cliparea.
- def ameter

  The ameter widget.
- def ameter_color

  The ameter_color command.
- def dmeter

  The dmeter widget.
- def button

  The button widget.
- def choice

  The choice widget allows you to print a string and display buttons for the user to choose a response.
- def groupBox

  The groupBox widget.
- def radioButton

  The radioButton widget.
- def staticText

  The staticText widget.

- def slider

  The slider widget.
- def progressBar

  The progressBar widget.
- def gauge

  The gauge widget.
- def touchZone

  The touchZone command.
- def dial

  The dial widget.
- def theme

  The theme command sets the colors for widgets.
- def fontw

  The fontW command will set the font for widget.
- def string

  The string command will set or return a internal string.
- def wstack

  The wstack command will return the stack of widgets pressed 32 levels.
- def wvalue

  The wvalue command will set or return a value to or from a widget.
- def wstate

  The wstate command.
- def picture

  The picture command will display a bitmap in bmp, jpg, gif formats with optional coordinates.
- def font

  The font command will set current font to use for printString fonts are located in the /EZSYS/FONTS and /EZUSER/-FONTS
  use the ezLCD-3xx Font Converter from earthlcd.com
  to convert truetype fonts to ezLCD format
  internal fonts will display faster than external fonts.
- def fonto

  The FONTO command will change the orientation or direction the text prints.
- def printString

  print string in current color and font and optional coordinates

Variables

- int **BLACK** = 0
- int **GRAY** = 1
- int **SILVER** = 2
- int **WHITE** = 3
- int **RED** = 4
- int **MAROON** = 5
- int **YELLOW** = 6
- int **OLIVE** = 7
- int **LIME** = 8
- int **GREEN** = 9
- int **AQUA** = 10
- int **TEAL** = 11
- int **BLUE** = 12
- int **NAVY** = 13
- int **FUCHISA** = 14

- int **PURPLE** = 15
- int **FILLED** = 1
- int **ON** = 1
- int **OFF** = 0
- int **FIFO** = 0
- int **LIFO** = 1
- int **CLEAR** = 2
- int **DELETE** = 0
- int **ENABLE** = 1
- int **DISABLE** = 2
- int **REDRAW** = 3
- interface

  open serial port
- **ser**
- **sio**

16.1.1 Detailed Description

```
Python Module for earthlcd.com ezLCD 3xx line of displays
http://earthlcd.com

(c)2013 ken segler
ken@earthlcd.com
requires pySerial http://pyserial.sourceforge.net/
```

16.1.2 Function Documentation

16.1.2.1 def module.ezLCD3xx.__init__ (*self*, *interface*)

ezLCD object

16.1.2.2 def module.ezLCD3xx.WaitForCR (*self*)

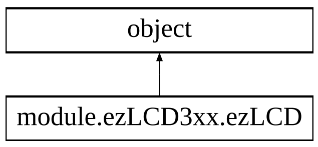
This is a internal use function.

Chapter 17

Class Documentation

17.1 module.ezLCD3xx.ezLCD Class Reference

Inheritance diagram for module.ezLCD3xx.ezLCD:



The documentation for this class was generated from the following file:

- C:/Users/codeman/Documents/GitHub/ezLCD3xxPython/module/ezLCD3xx.py

Chapter 18

Example Documentation

18.1 ButtonAlign.py

```
# Button Align ezLCD Python demo
#

import platform
import sys
sys.path.append('..\module')
from ezLCD3xx import *

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD('com58')
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose('off')
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# CLear screen
LCD.cls()
# Set draw color to red
LCD.color(RED)
# Set widget font 0
LCD.fontw(0,'1')
# Set wodget font 1
LCD.fontw(1,'0')
# Set theme #1
LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
# Draw button widget with a ID of 1
LCD.color(WHITE)
LCD.printString('Button Alignment',90,4)
LCD.color(RED)
LCD.xy(0,0)
LCD.box(319,239)
LCD.button( 1,  30,  25, 250, 30, 1, 0, 10, 6, 1,'0 = Centered')
LCD.button( 2,  30,  60, 250, 30, 1, 1, 10, 6, 2,'1 = Right')
LCD.button( 3,  30,  95, 250, 30, 1, 2, 10, 6, 3,'2 = Left')
LCD.button( 4,  30, 130, 250, 50, 1, 3, 10, 6, 4,'3 = Bottom')
LCD.button( 5,  30, 185, 250, 50, 1, 4, 10, 6, 5,'4 = Top')
```

18.2 ButtonOptions.py

```
# Button Options ezLCD Python demo
#

```

```
import platform
import sys
sys.path.append('..\module')
from ezLCD3xx import *

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD('com4')
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose('off')
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# CLear screen
LCD.cls()
# Set draw color to red
LCD.color(RED)
# Set widget font 0
LCD.fontw(0,'1')
# Set wodget font 1
LCD.fontw(1,'0')
# Set theme #1
LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
# Draw button widget with a ID of 1
LCD.color(WHITE)
LCD.printString('Button Options',90,4)
LCD.color(RED)
LCD.xy(0,0)
LCD.box(319,239)
LCD.button( 1,  30, 25, 250, 30, 1, 0, 10, 6, 3,'1 = Draw')
LCD.button( 2,  30, 60, 250, 30, 1, 0, 10, 6, 4,'2 = Disabled')
LCD.button( 3,  30, 95, 250, 30, 1, 0, 10, 6, 5,'3 = Pressed')
LCD.button( 4,  30, 130, 250, 30, 1, 0, 10, 6, 6,'4 = Not Pressed')
LCD.button( 5,  30, 165, 250, 30, 1, 0, 10, 6, 7,'5 = Pressed Disabled')
LCD.button( 6,  30, 200, 250, 30, 1, 0, 10, 6, 7,'6 = Not Pressed Disabled')


```

18.3 ButtonRadius.py

```
# Button Radius ezLCD Python demo
#

import platform
import sys
sys.path.append('..\module')
from ezLCD3xx import *

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD('com4')
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose('off')
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# CLear screen
LCD.cls()
# Set draw color to red
```

```
LCD.color(RED)
# Set widget font 0
LCD.fontw(0,'1')
# Set wodget font 1
LCD.fontw(1,'0')
# Set theme #1
LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
# Draw button widget with a ID of 1
LCD.color(WHITE)
LCD.printString('Button Radius',100,4)
LCD.color(RED)
LCD.xy(0,0)
LCD.box(319,239)
LCD.button( 1,  30, 25, 250, 30, 1, 0, 0, 6, 3,'Radius = 0')
LCD.button( 2,  30, 60, 250, 30, 1, 0, 10, 6, 4,'Radius = 10')
LCD.button( 3,  30, 95, 250, 30, 1, 0, 15, 6, 5,'Radius = 15')
LCD.button( 4,  30, 130, 100, 100, 1, 0, 20, 6, 6,'Radius = 20')
LCD.button( 5,  180, 130, 100, 100, 1, 0, 50, 6, 7,'Radius = 50')

```

18.4 GaugeDemo.py

```
# # Gauge Python demo
# Ken Segler
#
#

import platform
import sys
import time as timer
import random

sys.path.append('..\module')
from ezLCD3xx import *

LCD = ezLCD(None)
comPort =  LCD.findezLCD()

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD(comPort[0][0])
# Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
# Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial() == False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose('off')
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# Clear screen
LCD.cls()
# Use internal medium font
LCD.fontw(1, '0')
# Set draw color to red
LCD.color(RED)
# set x y to 0
LCD.xy(0, 0)
# draw box
LCD.box(320, 240)
# set theme #1
LCD.theme(1, 155, 152, 0, 0, 0, 151, 8, 9, 0, 1)
# Set draw color to red
LCD.color(WHITE)
# Print string at coordinates x=80 and y=100
LCD.printString("Gauge Demo", 100, 10)
# LCD.printString(" Update Theme Based On Value", 30,40)
#   def gauge(self, ID, x, y, width, height, options, initial, mmin, mmax, theme, stringID = None, text =
    None ):
LCD.gauge(1, 20, 90, 280, 30, 1, 1, 1, 200, 1, 1, ' Degrees F')
value = 1
low = -1
high = -1
average = -1
while True:
    value +=1
    if value >200:
        value =0
    timer.sleep(.1)
    LCD.wvalue(1, value)




```

18.5 ProgressTheme.py

```
## Progress bar Python demo
# Ken Segler
#
# This demo will display a progress bar and change the theme based on the value of the progress bar
# Starts with a green theme then at 30 changes to yellow then to red after 60
#

import platform
import sys
import time as timer
import random

sys.path.append('..\module')
from ezLCD3xx import *

LCD = ezLCD(None)
comPort =  LCD.findezLCD()

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD(comPort[0][0])
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

# Turn verbose off
LCD.verbose(OFF)
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# Clear screen
LCD.cls()
# Use internal medium font
LCD.fontw(1,'0')
# Set text font to internal medium
LCD.font('0')
# Set draw color to red
LCD.color(RED)
# set x y to 0
LCD.xy(0,0)
# draw box
LCD.box(320,240)
# set theme #1
LCD.theme(1, 155, 152, 3, 0, 0, 9, 8, 9, 0, 1)
# Set draw color to red
LCD.color(WHITE)
# Print string at coordinates x=80 and y=100
LCD.printString("Progress Bar Demo",80,10)
LCD.printString(" Update Theme Based On Value", 30,40)
LCD.progressBar(1, 20, 150, 280, 30, 1, 1, 100, 1, 1,' PSI')
LCD.color(8)
LCD.printString('LOW', 20,125)
LCD.color(6)
LCD.printString('MEDIUM', 120,125)
LCD.color(4)
LCD.printString('HIGH', 255,125)

value = 1

while True:
    timer.sleep(.1)
    value +=1
    # update widget 1 value
    LCD.wvalue(1, value)
    if value == 30:
        # change theme when value get to 30
        LCD.theme(1, 155, 152, 0, 3, 0, 37, 6, 6, 6, 1)
        # redraw widget 1
        LCD.wstate(1, 3)
    if value == 60:
        # change theme when value get to 60
        LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
        # redraw widget 1
        LCD.wstate(1,3)
    if value==100:
        # change theme when value get to 100
        LCD.theme(1, 155, 152, 3, 0, 0, 9, 8, 9, 0, 1)
        value = 1
        # reset widget 1 to 0
        LCD.wvalue(1,value)
        # redraw widget 1
        LCD.wstate(1,3)
```

18.6 TouchZoneIM.py

```
# Button Align ezLCD Python demo
#

import platform
import sys
sys.path.append('..\module')
from ezLCD3xx import *

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD('com4')
#Mac
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
#Linux
elif platform.system() == 'Linux':
    LCD = ezLCD('/dev/ttyACM0')

# Bail out if comport error
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

tzData = ( 1, 0, 33,  2, 46, 33, 3, 92, 33, 4, 138, 33, 5, 184, 33, 6, 230, 33, 7, 276, 33,
           8, 0, 79, 9, 46, 79, 10, 92, 79, 11, 138, 79, 12, 184, 79, 13, 230, 79, 14, 276, 79,
          15, 0, 125, 16, 46, 125, 17, 92, 125, 18, 138, 125, 19, 184, 125, 20, 230, 125, 21, 276, 125)

# Turn verbose off
LCD.verbose('off')
# Turn off button press info from ezLCD
LCD.wquiet(ON)
# CLear screen
LCD.cls()
# Set draw color to red
LCD.color(RED)
# Set widget font 0
LCD.fontw(0,'1')
# Set wodget font 1
LCD.fontw(1,'0')
# Set theme #1
LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
# Draw button widget with a ID of 1
LCD.picture('im.gif')
LCD.color(RED)
LCD.xy(0,0)
LCD.box(320,240)
LCD.printString('TouchZone Demo', 80, 10)
tzX = 0
tzY = 33
for count in range(0, 63, 3):
    LCD.touchZone(tzData[count], tzData[count+1], tzData[count+2],43 ,43, ENABLE)

while True:
    (ID, Info, Data) = LCD.wstack(FIFO)
    if ID > 0 and Info ==4:
        ID -=1
        LCD.color(BLACK)
        LCD.xy(tzData[(ID*3)+1],tzData[(ID*3)+2] )
        LCD.box(43,45)
        string ='TouchZone ' + str(ID+1) +' Pressed'
        LCD.printString(string, 60, 200)
    if ID > 0 and Info ==1 or Info ==2:
        ID -=1
        LCD.color(WHITE)
        LCD.xy(tzData[(ID*3)+1],tzData[(ID*3)+2] )
        LCD.box(43,45)
        LCD.printString(string, 60, 200)


```

Index