ezLCD Python Module 1.02

Generated by Doxygen 1.8.4

Sat Jul 20 2013 13:53:46

Contents

Chapter 1

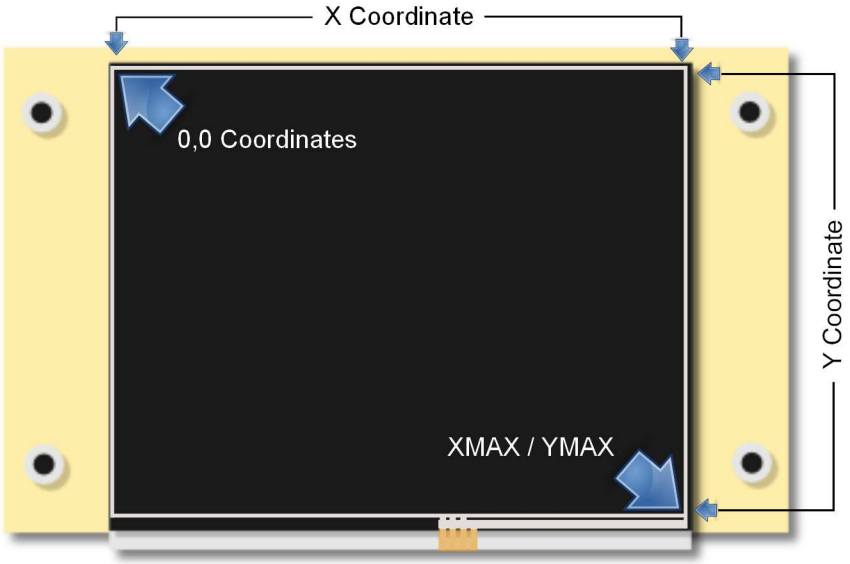
Installing the Module

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.

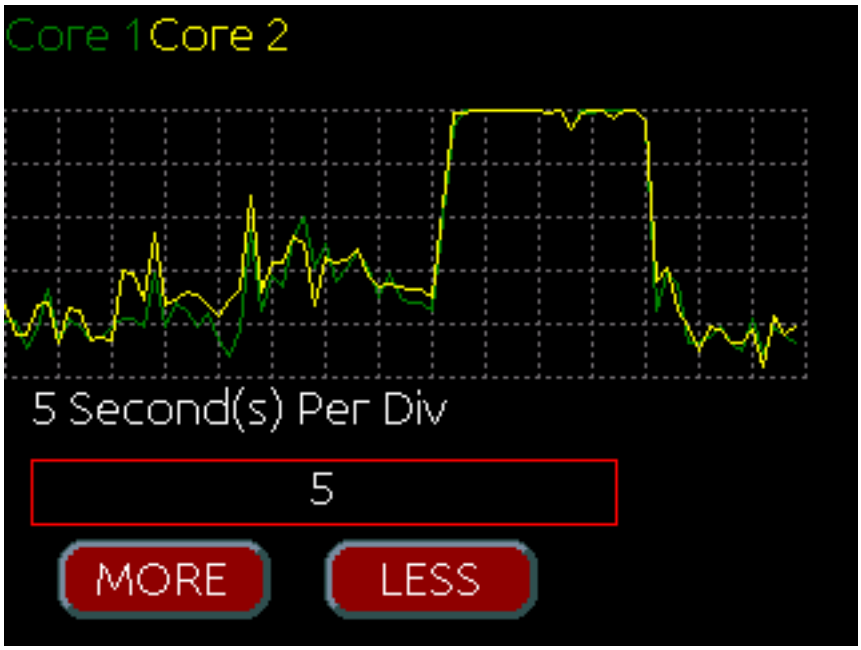
Its best to have a minimal one in the /EZUSER/MACROS directory with only the relevent settings in it .

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.


Core 1Core 2
5 Second(s) Per Div
5
MORE
LESS

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


```
1 # Minimal ezLCD Python demo
2 #
3
4 import platform
5 import sys
6
7
8 sys.path.append("C:\Users\codeman\Documents\GitHub\ezLCD3xxPython\module")
9 from ezLCD3xx import *
10
11 #check what OS we are on
12 #Windows
13 if platform.system() == 'Windows':
14     LCD = ezLCD('com6')
15 #Mac
16 elif platform.system() == 'Dawrwin':
17     LCD = ezLCD('/dev/tty.usbsomething')
18 # Bail out if comport error
19 if LCD.openSerial()==False:
20     print 'Error Opening Port'
```

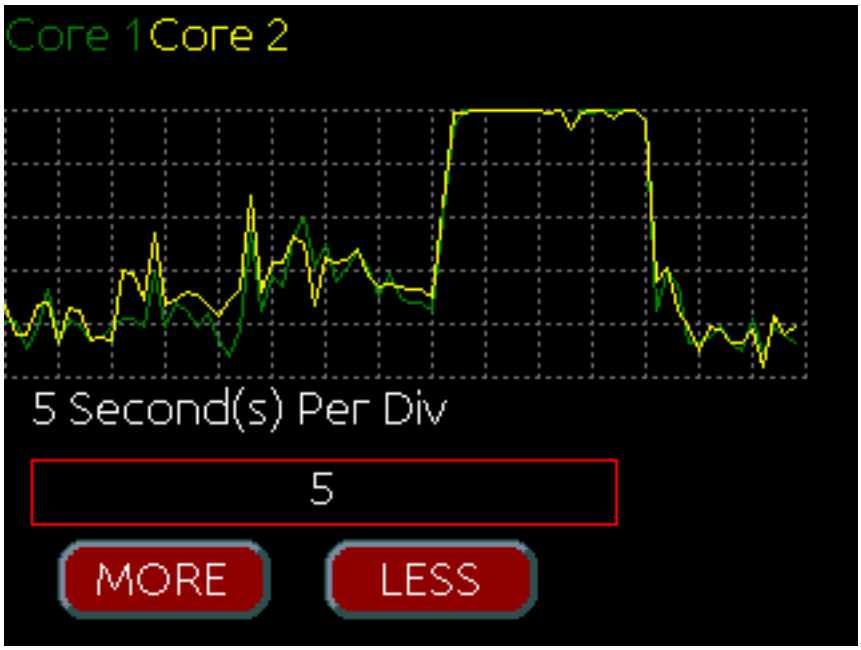
```
21     raise SystemExit
22
23 # Turn verbose off
24 LCD.verbose('off')
25 # Turn off button press info from ezLCD
26 LCD.wquiet(ON)
27 # CLear screen
28 LCD.cls()
29 # Set draw color to red
30 LCD.color(RED)
31 # Print string at coordinates x=80 and y=100
32 LCD.printString("Hello From Python",80,100)
33
```

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


```
1 # Button ezLCD Python demo
2 #
3
4 import platform
5 import sys
6 sys.path.append('module')
7 from ezLCD3xx import *
8
9 #check what OS we are on
10 #Windows
11 if platform.system() == 'Windows':
12     LCD = ezLCD('com4')
13 #Mac
14 elif platform.system() == 'Dawrwin':
15     LCD = ezLCD('/dev/tty.usbsomething')
16 #Linux
17 elif platform.system() == 'Linux':
18     LCD = ezLCD('/dev/ttyACM0')
19
20 # Bail out if comport error
21 if LCD.openSerial()==False:
22     print 'Error Opening Port'
23     raise SystemExit
24
25 # Turn verbose off
26 LCD.verbose('off')
27 # Turn off button press info from ezLCD
28 LCD.wquiet(ON)
29 # CLear screen
30 LCD.cls()
31 # Set draw color to red
32 LCD.color(RED)
33 # Set widget font 0
34 LCD.fontw(0,'1')
35 # Set wodget font 1
36 LCD.fontw(1,'0')
37 # Set theme #1
38 LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
```

```
39 # Print string at coordinates x=80 and y=100
40 LCD.printString("Hello From Python",80,100)
41 # Draw button widget with a ID of 1
42 LCD.button( 1,  80, 150, 155, 50, 1, 0, 10, 6, 3,'Press Here')
43 # Draw a staticText box
44 LCD.staticText(2, 35, 30, 250, 30, 8, 1, 1,'Press Button')
45 # Clear widget stack
46 LCD.wstack(CLEAR)
47
48 while True:
49     # check widget stack this will return widget updates (button press ect.) last in first out order
50     (ID, Info, Data) = LCD.wstack(LIFO)
51 #   print ID, Info, Data
52     # check if ID = 1 widget 1 and info = pressed
53     if ID == 1 and Info == 4:
54         # clear the stack just to be safe
55 #       LCD.wstack(CLEAR)
56         # change draw color to yellow
57         LCD.color(YELLOW)
58         # change change string 1 for text on static text ID 2
59         LCD.string(1,'Button Pressed')
60         # redraw static text box ID 2 3=redraw
61         LCD.wstate(2, 3)
62     # check if ID = 1 widget 1 and info = pressed and released
63     if ID == 1 and Info == 1:
64         # clear the stack just to be safe
65 #       LCD.wstack(CLEAR)
66         # change draw color to yellow
67         LCD.color(YELLOW)
68         # change change string 1 for text on static text ID 2
69         LCD.string(1,'Button Pressed and Released')
70         # redraw static text box ID 2 3=redraw
71         LCD.wstate(2, 3)
72
73
```

Load example will display the cpu load as a graph




```
1 #!/usr/bin/env python
2 # Python Serial library for ezLCD3xx
3 # http://www.ezlcd.com/
4 #
5 # You need the pySerial Library by Chris Liechti
6 # http://pyserial.wiki.sourceforge.net/pySerial
7 #
8
9
10 # END SerLCD Class Definition ------------------------------------
11
12 # Start Test Program ---------------------------------------------
13 import commands
14 import os
15 import re
16 import time as timer
```

```
17 import sys
18 import platform
19 import time
20 import psutil
21
22 sys.path.append('module')
23 from ezLCD3xx import *
24
25 def drawGrid():
26     LCD.lineType(2)
27     LCD.xy(0,30)
28     LCD.color(BLACK)
29     LCD.box(300,110,1)
30     LCD.xy(0,0)
31     LCD.color(GREEN)
32     LCD.printString('Core 1')
33     LCD.color(YELLOW)
34     LCD.printString('Core 2')
35     LCD.color(155)
36     LCD.color(LIME)
37     LCD.font('1')
38     LCD.font('0')
39     LCD.color(151)
40     for y in range(6):
41         LCD.xy(0,(y*20)+39)
42         LCD.line(300,(y*20)+39)
43     for x in range(16):
44         LCD.xy(x*20,39)
45         LCD.line(x*20,139)
46     LCD.xy(300,39)
47     LCD.line(300,139)
48     LCD.lineType(0)
49
50 def drawTime(res):
51     LCD.xy(10,140)
52     LCD.color(BLACK)
53     LCD.box(300,30, FILLED)
54     LCD.color(WHITE)
55     Time=str(res)+' Second(s) Per Div'
56     LCD.printString(Time)
57
58     LCD.string(5, str(res))
59     LCD.wstate(7,REDRAW)
60
61
62 #check what OS we are on
63 #Windows
64 if platform.system() == 'Windows':
65     LCD = ezLCD('com6')
66 #Mac
67 elif platform.system() == 'Dawrwin':
68     LCD = ezLCD('/dev/tty.usbsomething')
69 #Linux
70 elif platform.system() == 'Linux':
71     LCD = ezLCD('/dev/ttyACM0')
72 # Bail out if comport error
73 if LCD.openSerial()==False:
74     print 'Error Opening Port'
75     raise SystemExit
76
77 LCD.ping()
78 LCD.verbose('OFF')
79 LCD.wquiet(ON)
80 LCD.cls()
81 LCD.fontw(0,'1')
82 LCD.fontw(1,'0')
83 LCD.fontw(2,'serif24')
84 LCD.theme(1, 155, 152, 3, 0, 3, 24, 4, 5, 0, 1)
85 LCD.backlight(100, 5, 10)
86 LCD.cls()
87 LCD.font('0')
88 LCD.fonto(0)
89 info = ' '
90 LCD.string( 1, '%')
91 LCD.color(WHITE)
92 LCD.cfgio(8,'analog')
93 print LCD.xmax()
94 print LCD.ymax()
95 LCD.xy(100,100)
96 (x,y) = LCD.xy()
97 print int(x), int(y)
98 (r,g,b)=LCD.colorId(3)
99 print r,g,b
100 print LCD.string(65)
101 print LCD.string(66)
102 print LCD.color()
103 print LCD.io(8)
```

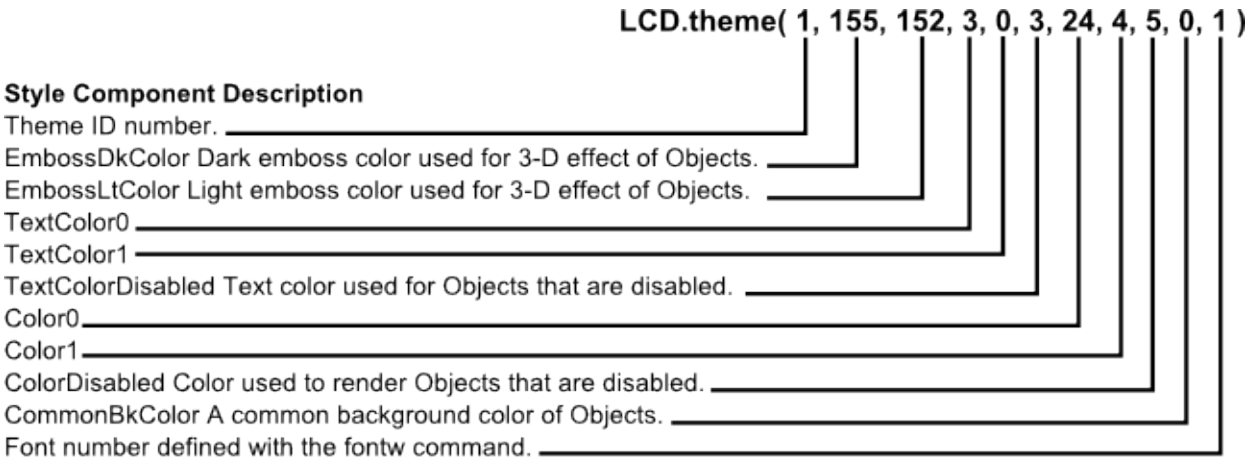
```
104
105
106 LCD.button( 5, 20, 200, 80, 30 , 1, 0, 10, 1, 2, 'MORE')
107 LCD.button( 6, 120, 200, 80, 30 , 1, 0, 10, 1, 3, 'LESS')
108 LCD.staticText(7, 10, 170, 220, 25, 8, 1, 5, 'test')
109 drawGrid()
110 x=0
111 y1=239
112 y2=239
113 lx=0
114 ly1=239
115 ly2=239
116 res=5
117 drawTime(res)
118 LCD.wstack(CLEAR)
119 while True:
120
121     oldinfo = info
122     cores=psutil.cpu_percent(interval=1, percpu=True)
123     y1 = 139 - cores[0]
124     y2 = 139 - cores[1]
125     if x!=0:
126         LCD.color(GREEN)
127         LCD.xy(lx,ly1)
128         LCD.line(x, y1)
129         LCD.color(YELLOW)
130         LCD.xy(lx,ly2)
131         LCD.line(x, y2)
132     ly1 = y1
133     ly2 = y2
134     lx = x
135     x += 20/res
136
137     if x >= 300:
138         x=0
139         y1=239
140         y2=239
141         lx =0
142         ly1 =239
143         ly2 =239
144         drawGrid()
145     (ID, info, data) = LCD.wstack(LIFO)
146     LCD.wstack(CLEAR)
147     if ID == 5 and info==1:
148         res +=1
149         drawTime(res)
150     if ID == 6 and info==1:
151         if res > 1:
152             res -=1
153             drawTime(res)
154 LCD.closeSerial()
155 # End Test Program -------------------------------------
```

Chapter 5

Introduction To Themes

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

The command looks bad but is easy to understand .



Chapter 6

Color Table


0 Black 0, 0, 0
1 Gray 128, 128, 128
2 Silver 192, 192, 192
3
6 Yellow 255, 255, 0
7 Olive 128, 128, 0
8 Lime 0, 255, 0
9
12 Blue 0, 0, 255
13 Navy 0, 0, 128
14 Fuchsia 255, 0, 255
14
16 IndianRed 205, 92, 92
17 LightCoral 240, 128, 128
18 Salmon 250, 128, 114
19
22 Crimson 220, 20, 60
23 FireBrick 178, 34, 34
24 DarkRed 139, 0, 0
25
28 DeepPink 255, 20, 147
29 MediumVioletRed 199, 21, 133
30 PaleVioletRed 219, 112, 147
31
34 OrangeRed 255, 69, 0
35 DarkOrange 255, 140, 0
36 Orange 255, 165, 0
37
40 LemonChiffon 255, 250, 205
41 LightGoldenrodYellow 250, 250, 210
42 PapayaWhip 255, 239, 213
43
46 Khaki 240, 230, 140
47 DarkKhaki 189, 183, 107
48 Lavender 230, 230, 250
49
52 Orchid 218, 112, 214
14 53 Fuchsia 255, 0, 255
14 53 Magenta 255, 0, 255
54
57 DarkViolet 148, 0, 211
58 DarkOrchid 153, 50, 204
59 DarkMagenta 139, 0, 139
60
62 DarkSlateBlue 72, 61, 139
65 GreenYellow 173, 255, 47
66 Chartreuse 127, 255, 0
67
70 PaleGreen 152, 251, 152
71 LightGreen 144, 238, 144
72 MediumSpringGreen 0, 250, 154
73
76 ForestGreen 34, 139, 34
9 77 Green 0, 128, 0
78 DarkGreen 0, 100, 0
79
82 DarkOliveGreen 85, 107, 47
83 MediumAquamarine 102, 205, 170
84 DarkSeaGreen 143, 188, 143
85
10 88 Aqua 0, 255, 255
10 89 Cyan 0, 255, 255
90 LightCyan 224, 255, 255
91
94 154 MediumTurquoise 72, 209, 204
95 DarkTurquoise 0, 206, 209
CadetBlue 95, 158, 160
97
100 LightBlue
101 SkyBlue
102 LightSkyBlue
103

Chapter 7

Introduction To Fonts

Chapter 8

Introduction To Bitmaps

picture supports gif, jpg and bmp

Chapter 9

Introduction To Widgets

Buttons

TouchZone

Slider

ProgressBar

AnalogMeter

DigitalMeter

StaticText

GroupBox

Dial

Choice

CheckBox

Radio

Chapter 10

Module Index

10.1 Modules

Here is a list of all modules:

Chapter 11

Namespace Index

11.1 Namespace List

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

Chapter 12

Hierarchical Index

12.1 Class Hierarchy

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

Chapter 13

Class Index

13.1 Class List

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

Chapter 14

Module Documentation

14.1 Commands

Functions

- 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.

14.1.1 Detailed Description

14.1.2 Function Documentation

14.1.2.1 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

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

The cfgio command will configure io pins.

Parameters

pin	
function	

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

The io command use to set and clear io pins.

Parameters

pin	
level	

Returns

io level

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

the ping command

Returns

0

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

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

Parameters

filename	macro filename

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

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

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

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

Parameters

filename	macro filename

14.1.2.8 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 !

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

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

Parameters

state	0=off 1=on

14.1.2.10 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

14.1.2.11 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

14.1.2.12 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

14.2 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.

14.2.1 Detailed Description

14.2.2 Function Documentation

14.2.2.1 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

14.2.2.2 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

14.2.2.3 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

14.2.2.4 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	

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

The clipenable command enables or disables cliparea.

Parameters

enable	0=off 1=on

14.2.2.6 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

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

The color command see ezLCD3xx manual for colors.

Parameters

color	number

Returns

color as a tuple

14.2.2.8 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

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

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

Parameters

x	
y	

14.2.2.10 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)

14.2.2.11 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.

14.2.2.12 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

14.2.2.13 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

14.2.2.14 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

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

14.3 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 command.
- 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 command.
- def module.ezLCD3xx.progressBar

  The progressBar command.
- def module.ezLCD3xx.touchZone

  The touchZone command.
- def module.ezLCD3xx.dial

  The dial command.
- 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.

14.3.1 Detailed Description

14.3.2 Function Documentation

14.3.2.1 def module.ezLCD3xx.ameter (*self, ID, x, y, width, height, options, value, minV, maxV, theme, stringID, meterType* = 0)

The ameter widget.

Parameters

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

14.3.2.2 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	

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

The button command.

Parameters

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

14.3.2.4 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

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

The dial command.

Parameters

ID	
x	
y	
radius	
option	
resolution	
value	
maxx	
theme	

14.3.2.6 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	

14.3.2.7 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

14.3.2.8 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	

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

The progressBar command.

Parameters

ID	
x	
y	
width	
height	
options	
value	
mmax	
theme	
stringID	

14.3.2.10 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	

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

The slider command.

Parameters

ID	
x	
y	
width	
height	
options	
rrange	
resolution	
value	
theme	

14.3.2.12 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

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

The string command will set or return a internal string.

Parameters

stringNumber	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

14.3.2.14 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

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

The touchZone command.

Parameters

ID	
x	
y	
width	
height	
options	

14.3.2.16 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

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

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

The wstate command.

Parameters

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

14.3.2.18 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	

14.4 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

14.4.1 Detailed Description

14.4.2 Function Documentation

14.4.2.1 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
	
	`1 # Set font to internal medium font`
	`2 LCD.font('0')`
	`3 # Set font to LCD24`
	`4 LCD.font('LCD24')`

14.4.2.2 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

```
1 LCD.fonto(0)
2 LCD.color(YELLOW)
3 LCD.printString('Hello',100,100)
4 LCD.fonto(90)
5 LCD.color(RED)
6 LCD.printString('Hello',100,100)
7 LCD.fonto(180)
```

```
8 LCD.color(BLUE)
9 LCD.printString('Hello',100,100)
10 LCD.fonto(270)
11 LCD.color(GREEN)
12 LCD.printString('Hello',100,100)
```

14.4.2.3 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
	
	`1 # display python.gif at 10 10`
	`2 LCD.picture('python.gif',10,10)`
	`3 # display python.gif at current x y`
	`4 LCD.picture('python.gif')`

14.4.2.4 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 **
	
	`1 # display string 'Hello World' at 10 10`
	`2 LCD.printString('Hello World',10,10)`
	`3 # display string 'Hello World' at current x y`
	`4 LCD.printString('Hello World')`
	`5 # diplay string 'Hello World' at 10 10 rotated 90`
	`6 LCD.printString('Hello World',10,10,90)`

Chapter 15

Namespace Documentation

15.1 module.ezLCD3xx Namespace Reference

Classes

- class ezLCD

Functions

- def __init__

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

  This is a internal use function.
- 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 command.
- 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 command.
- def progressBar

The progressBar command.

- def touchZone

  The touchZone command.
- def dial

  The dial command.
- 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**

15.1.1 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/
```

15.1.2 Function Documentation

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

ezLCD object

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

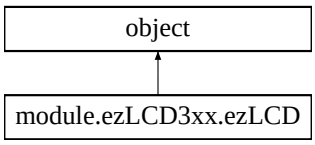
This is a internal use function.

Chapter 16

Class Documentation

16.1 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

Index