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 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.ezm
verbose off
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
```

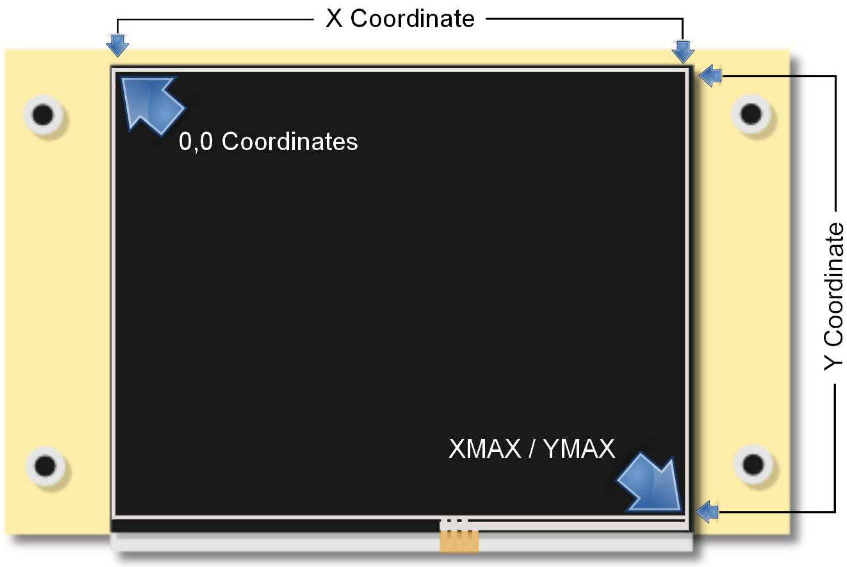
Commands are sent to the ezLCD though the serial interface, Commands are text based and end with a carrage return **cr**.

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.

Chapter 3

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.

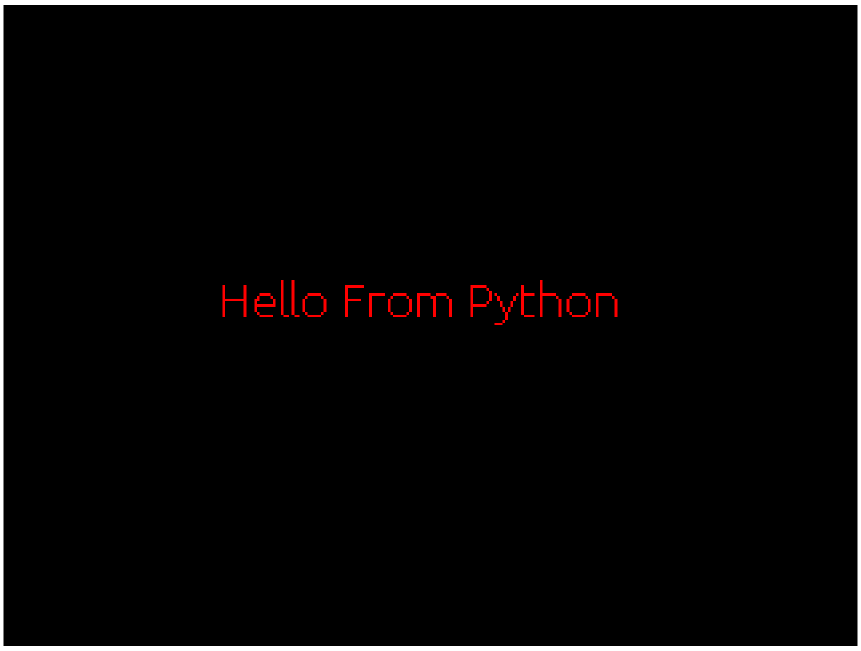
XMAX and YMAX Values for the ezLCD 3xx Line

arLCD	319	239
ezLCD-301	399	239
ezLCD-302		
ezLCD-303	319	239
ezLCD-313	319	239
ezLCD-304	479	271

Chapter 4

Introduction To The Software

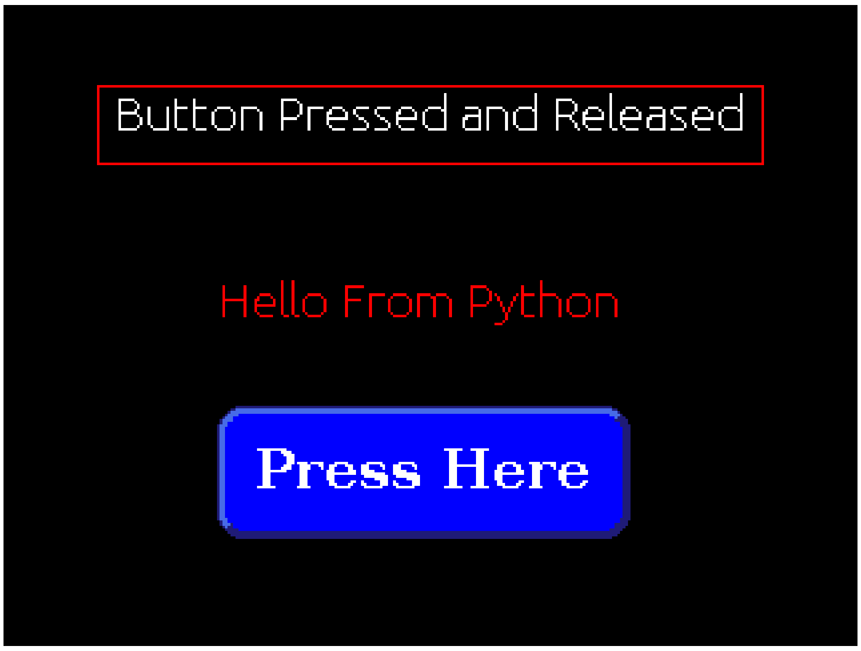
Minimal example will open the ezLCD port clear the screen and print 'Hello From Python' in red



```
# Minimal ezLCD Python demo
#

import platform
import sys


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

#check what OS we are on
#Windows
if platform.system() == 'Windows':
    LCD = ezLCD('com6')
#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
```

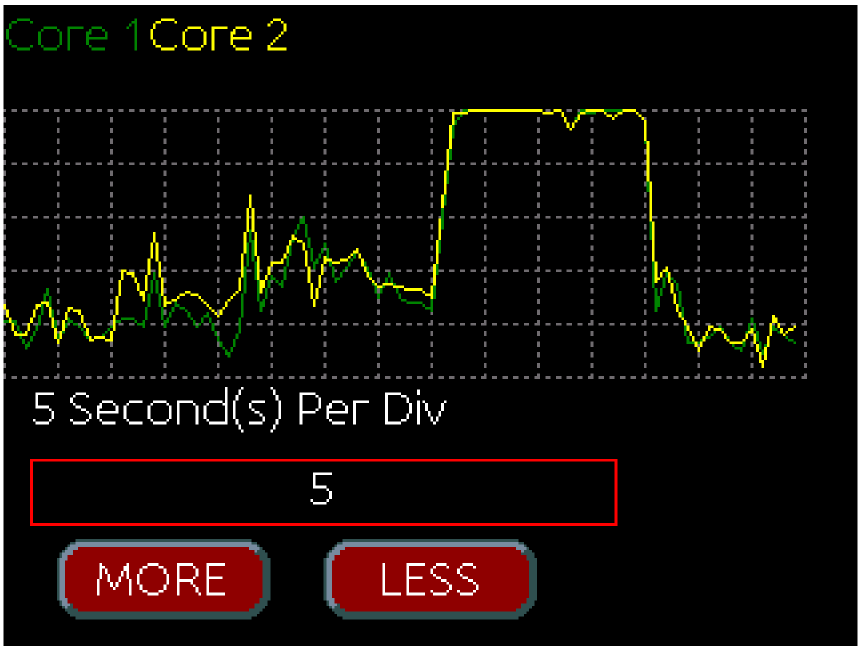
```
LCD.color(RED)
# Print string at coordinates x=80 and y=100
LCD.printString("Hello From Python",80,100)

```

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("C:\Users\segler\Documents\GitHub\ezLCD3xxPython\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')
# 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(LIFO)
    # 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("C:\Users\codeman\Documents\GitHub\ezLCD3xxPython\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)


if platform.system() == 'Windows':
    LCD = ezLCD('com6')
elif platform.system() == 'Dawrwin':
    LCD = ezLCD('/dev/tty.usbsomething')
if LCD.openSerial()==False:
    print 'Error Opening Port'
    raise SystemExit

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()
LCD.xy(100,100)
(x,y) = LCD.xy()
print int(x), int(y)
(r,g,b)=LCD.colorId(3)
print r,g,b
print LCD.string(65)
print LCD.string(66)
print LCD.color()
print LCD.io(8)


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

Module Index

5.1 Modules

Here is a list of all modules:

Chapter 6

Namespace Index

6.1 Namespace List

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

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

Chapter 8

Class Index

8.1 Class List

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

Chapter 9

Module Documentation

9.1 Commands

Functions

- def ezLCD3xx.verbose

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

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

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

  the ping command
- def ezLCD3xx.backlight

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

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

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

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

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

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

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

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

  The calibrate command will re calibrate the touch screen.

9.1.1 Detailed Description

9.1.2 Function Documentation

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

The backlight command will set backlight brightness and timeout.

Parameters

brightness	1
timeout	2
level	3

9.1.2.2 def ezLCD3xx.cfgio (*self*, *pin*, *function*)

The cfgio command will configure io pins.

Parameters

pin	
function	

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

The io command use to set and clear io pins.

Parameters

pin	
level	

Returns

io level

9.1.2.4 def ezLCD3xx.ping (*self*)

the ping command

Returns

0

9.1.2.5 def ezLCD3xx.play (*self*, *filename*)

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

Parameters

filename	macro filename

9.1.2.6 def ezLCD3xx.reset (*self*)

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

9.1.2.7 def ezLCD3xx.run (*self*, *filename*)

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

Parameters

filename	macro filename

9.1.2.8 def 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 !

9.1.2.9 def ezLCD3xx.verbose (*self*, *state*)

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

Parameters

state	0=off 1=on

9.1.2.10 def ezLCD3xx.wquiet (*self*, *state*)

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

Parameters

state	0=off 1=on

9.1.2.11 def ezLCD3xx.xmax (*self*)

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

Returns

x-horizontal resolution in pixels starting from 0

9.1.2.12 def ezLCD3xx.ymax (*self*)

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

Returns

y-vertical resolution in pixels starting from 0

9.2 Primitve Drawing Commands

Functions

- def ezLCD3xx.cls

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

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

  The colorId command.
- def ezLCD3xx.xy

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

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

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

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

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

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

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

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

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

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

  The clipenable command enables or disables cliparea.

9.2.1 Detailed Description

9.2.2 Function Documentation

9.2.2.1 def 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

9.2.2.2 def 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

9.2.2.3 def 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

9.2.2.4 def 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	

9.2.2.5 def ezLCD3xx.clipEnable (*self*, *enable*)

The clipenable command enables or disables cliparea.

Parameters

enable	0=off 1=on

9.2.2.6 def 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

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

The color command see ezLCD3xx manual for colors.

Parameters

color	number

Returns

color as a tuple

9.2.2.8 def 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

9.2.2.9 def ezLCD3xx.line (*self, x, y*)

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

Parameters

x	
y	

9.2.2.10 def 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)

9.2.2.11 def 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.

9.2.2.12 def 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

9.2.2.13 def 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

9.2.2.14 def 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()
```

9.3 Widgets

Functions

- def ezLCD3xx.ameter

  The ameter widget.
- def ezLCD3xx.ameter_color

  The ameter_color command.
- def ezLCD3xx.dmeter

  The dmeter widget.
- def ezLCD3xx.button

  The button command.
- def ezLCD3xx.choice

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

  The groupBox widget.
- def ezLCD3xx.radioButton

  The radioButton widget.
- def ezLCD3xx.staticText

  The staticText widget.
- def ezLCD3xx.slider

  The slider command.
- def ezLCD3xx.progressBar

  The progressBar command.
- def ezLCD3xx.touchZone

  The touchZone command.
- def ezLCD3xx.dial

  The dial command.
- def ezLCD3xx.theme

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

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

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

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

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

  The wstate command.

9.3.1 Detailed Description

9.3.2 Function Documentation

9.3.2.1 def 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	

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

The ameter_color command.

Parameters

ID	
color1	
color2	
color3	
color4	
color5	
color6	

9.3.2.3 def 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

9.3.2.4 def 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

9.3.2.5 def 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	

9.3.2.6 def 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	

9.3.2.7 def 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

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

The groupBox widget.

Parameters

ID	
x	
y	
width	
height	
options	
theme	
stringID	

9.3.2.9 def ezLCD3xx.progressBar (*self, ID, x, y, width, height, options, value, mmax, theme, stringID*)

The progressBar command.

Parameters

ID	
x	
y	
width	
height	
options	
value	
mmax	
theme	
stringID	

9.3.2.10 def 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	

9.3.2.11 def 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	

9.3.2.12 def 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

9.3.2.13 def ezLCD3xx.string (*self*, *stringNumber*, *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

9.3.2.14 def 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

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

The touchZone command.

Parameters

ID	
x	
y	
width	
height	
options	

9.3.2.16 def 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)
```

9.3.2.17 def ezLCD3xx.wstate (*self*, *ID*, *option*)

The wstate command.

Parameters

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

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

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

Parameters

ID	
value	

9.4 Bitmaps and Fonts

Functions

- def ezLCD3xx.picture

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

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

  print string in current color and font and optional coordinates

9.4.1 Detailed Description

9.4.2 Function Documentation

9.4.2.1 def 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')`

9.4.2.2 def 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)
```

9.4.2.3 def 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')`

9.4.2.4 def 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 10

Namespace Documentation

10.1 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**

10.1.1 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/
```

10.1.2 Function Documentation

10.1.2.1 def ezLCD3xx.__init__ (*self, interface*)

ezLCD object

10.1.2.2 def ezLCD3xx.closeSerial (*self*)

```
close
```

10.1.2.3 def ezLCD3xx.WaitForCR (*self*)

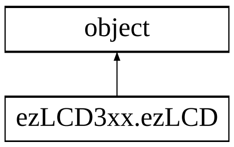
This is a internal use function.

Chapter 11

Class Documentation

11.1 ezLCD3xx.ezLCD Class Reference

Inheritance diagram for ezLCD3xx.ezLCD:



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

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

Index