

arLCD Arduino Library
1.01

Generated by Doxygen 1.8.3.1

Fri Apr 19 2013 10:39:56

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	EzLCD3 Class Reference	7
4.1.1	Detailed Description	10
4.1.2	Member Enumeration Documentation	10
4.1.2.1	Commands	10
4.1.3	Constructor & Destructor Documentation	12
4.1.3.1	EzLCD3	12
4.1.3.2	~EzLCD3	12
4.1.4	Member Function Documentation	12
4.1.4.1	analogMeter	12
4.1.4.2	analogMeterColor	12
4.1.4.3	arc	13
4.1.4.4	arc	13
4.1.4.5	begin	13
4.1.4.6	box	13
4.1.4.7	button	14
4.1.4.8	calibrate	14
4.1.4.9	checkbox	14
4.1.4.10	choice	14
4.1.4.11	circle	15
4.1.4.12	cliparea	15
4.1.4.13	clipenable	15
4.1.4.14	cls	15

4.1.4.15	cls	15
4.1.4.16	color	15
4.1.4.17	color	16
4.1.4.18	colorId	16
4.1.4.19	colorId	16
4.1.4.20	debugWrite	16
4.1.4.21	dial	16
4.1.4.22	digitalMeter	17
4.1.4.23	echo	17
4.1.4.24	echo	17
4.1.4.25	ellipse	17
4.1.4.26	ellipse	17
4.1.4.27	ezLCDUpgrade	17
4.1.4.28	fill	17
4.1.4.29	findEzLCD	18
4.1.4.30	font	18
4.1.4.31	font	18
4.1.4.32	fonto	18
4.1.4.33	fonto	18
4.1.4.34	fontw	18
4.1.4.35	getPixel	18
4.1.4.36	getWidgetValue	19
4.1.4.37	groupBox	19
4.1.4.38	height	19
4.1.4.39	image	19
4.1.4.40	image	20
4.1.4.41	isChecked	20
4.1.4.42	isHWSerial	20
4.1.4.43	isPressed	20
4.1.4.44	light	20
4.1.4.45	light	21
4.1.4.46	light	21
4.1.4.47	light	21
4.1.4.48	line	21
4.1.4.49	line	21
4.1.4.50	lineTtype	21
4.1.4.51	lineType	22
4.1.4.52	lineWidth	22
4.1.4.53	lineWidth	22
4.1.4.54	parseHex	22

4.1.4.55	picture	22
4.1.4.56	picture	22
4.1.4.57	pie	23
4.1.4.58	plot	23
4.1.4.59	plot	23
4.1.4.60	point	23
4.1.4.61	printAligned	23
4.1.4.62	printString	24
4.1.4.63	printStringId	24
4.1.4.64	progressBar	24
4.1.4.65	radioButton	24
4.1.4.66	rect	25
4.1.4.67	reset	25
4.1.4.68	slider	25
4.1.4.69	staticText	25
4.1.4.70	string	26
4.1.4.71	theme	26
4.1.4.72	touchS	26
4.1.4.73	touchX	26
4.1.4.74	touchY	26
4.1.4.75	touchZone	26
4.1.4.76	waitNoTouch	27
4.1.4.77	waitTouch	27
4.1.4.78	width	27
4.1.4.79	wquiet	27
4.1.4.80	write	27
4.1.4.81	wstack	27
4.1.4.82	wstate	27
4.1.4.83	wvalue	28
4.1.4.84	xmax	28
4.1.4.85	xy	28
4.1.4.86	xy_restore	28
4.1.4.87	xy_store	28
4.1.4.88	xyAligned	28
4.1.4.89	xyGet	29
4.1.4.90	ymax	29
4.1.5	Member Data Documentation	29
4.1.5.1	m_pStream	29
4.2	EzLCD3_HW Class Reference	29
4.2.1	Detailed Description	30

4.2.2	Constructor & Destructor Documentation	30
4.2.2.1	EzLCD3_HW	30
4.2.3	Member Function Documentation	30
4.2.3.1	begin	30
4.3	EzLCD3_SW Class Reference	30
4.3.1	Detailed Description	31
4.3.2	Constructor & Destructor Documentation	31
4.3.2.1	EzLCD3_SW	31
4.3.3	Member Function Documentation	31
4.3.3.1	begin	31
5	File Documentation	33
5.1	C:/Users/Segler/arduino-1.0.4/libraries/arLCD/examples.lst File Reference	33
5.2	C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.cpp File Reference	33
5.2.1	Macro Definition Documentation	33
5.2.1.1	DEFAULT_TIMEOUT	33
5.2.1.2	PARSE_TIMEOUT	33
5.2.1.3	sdebug	33
5.2.1.4	sdebugln	33
5.2.1.5	SEMPLE_SEP	33
5.3	C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.h File Reference	33
5.3.1	Macro Definition Documentation	34
5.3.1.1	CLEAR	34
5.3.1.2	EZLCD_ENUM_ALIGNMENT	34
5.3.1.3	EZLCD_ENUM_CHOICE	34
5.3.1.4	EZLCD_ENUM_COMBALIGN	34
5.3.1.5	EZLCD_ENUM_ORIENTATION	35
5.3.1.6	EZLCD_ENUM_WIDGETS	35
5.3.1.7	EZLCD_GLOBALENUMS	35
5.3.1.8	EZLCD_PROCESSING_PRIMITIVES	35
5.3.1.9	EZLCD_SERIALDEBUG	35
5.3.1.10	EZM_BAUD_RATE	35
5.3.1.11	FIFO	35
5.3.1.12	LIFO	35
5.3.2	Enumeration Type Documentation	35
5.3.2.1	anonymous enum	35
6	Example Documentation	37
6.1	analogMeter.ino	37
6.2	arc.ino	37
6.3	button.ino	38

6.4	button_interrupt.ino	38
6.5	checkbox.ino	39
6.6	choice.ino	40
6.7	circle.ino	40
6.8	cls.ino	40
6.9	color.ino	41
6.10	colorid.ino	41
6.11	dial.ino	41
6.12	digitalMeter.ino	42
6.13	ellipse.ino	43
6.14	font.ino	43
6.15	groupbox.ino	44
6.16	image.ino	44
6.17	light.ino	44
6.18	line.ino	45
6.19	lineType.ino	45
6.20	lineWidth.ino	46
6.21	pie.ino	46
6.22	plot.ino	46
6.23	point.ino	47
6.24	printExample.ino	47
6.25	progress.ino	47
6.26	radio.ino	48
6.27	radio_interrupt.ino	49
6.28	rect.ino	50
6.29	slider.ino	50
6.30	theme.ino	51
6.31	touch.ino	52
Index		52

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Print	
EzLCD3	7
EzLCD3_HW	29
EzLCD3_SW	30

Chapter 2

Class Index

2.1 Class List

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

EzLCD3	7
EzLCD3_HW	29
EzLCD3_SW	30

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

```
C:/Users/Segler/arduino-1.0.4/libraries/arLCD/examples.lst . . . . . 33
C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.cpp . . . . . 33
C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.h . . . . . 33
```

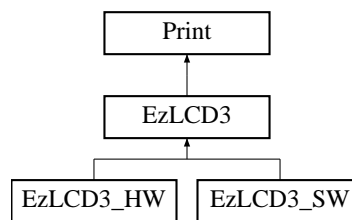

Chapter 4

Class Documentation

4.1 EzLCD3 Class Reference

```
#include <ezLCD.h>
```

Inheritance diagram for EzLCD3:



Public Types

- enum `Commands` {
 `Command` = 0, `Status` = 1, `Clr_Screen` = 2, `Ping` = 3,
 `zBeep` = 4, `Light` = 5, `Color` = 6, `eColor_ID` = 7,
 `Font` = 10, `Fontw` = 11, `Font_Orient` = 12, `Line_Width` = 13,
 `Line_Type` = 14, `eXY` = 15, `StringID` = 16, `Plot` = 17,
 `eLine` = 18, `Box` = 19, `zCircle` = 20, `Arc` = 21,
 `Pie` = 22, `Picture` = 24, `Print` = 25, `Beep_Freq` = 26,
 `Get_Pixel` = 27, `Calibrate` = 28, `zReset` = 29, `Rec_Macro` = 30,
 `Play_Macro` = 31, `Stop_Macro` = 32, `Pause_Macro` = 33, `Loop_Macro` = 34,
 `Speed_Macro` = 35, `Peri` = 36, `ConfigIO` = 37, `IO` = 38,
 `IOG` = 39, `Security` = 40, `Location` = 41, `Upgrade` = 43,
 `Parameters` = 45, `ClipEnable` = 46, `ClipArea` = 47, `Comment` = 50,
 `Fsgetcwd` = 51, `Fschdir` = 52, `Fsmkdir` = 53, `Fsrmdir` = 54,
 `Fsdir` = 55, `Fscopy` = 56, `Fsrename` = 57, `Fsremove` = 58,
 `Fsmore` = 59, `Format` = 60, `If` = 61, `Cmd` = 62,
 `Set_Button` = 70, `Set_CheckBox` = 71, `Set_Gbox` = 72, `Set_RadioButton` = 73,
 `Set_DMeter` = 74, `DMeter_Value` = 75, `Set_AMeter` = 76, `AMeter_Value` = 77,
 `AMeter_Color` = 78, `Set_TouchZone` = 79, `Set_Dial` = 80, `Set_Slider` = 82,
 `Set_Progress` = 85, `Progress_Value` = 86, `Set_StaticText` = 87, `StaticText_Value` = 88,
 `Choice` = 89, `Widget_Theme` = 90, `Widget_Values` = 91, `Widget_State` = 92,
 `Mode` = 98, `Comport` = 99, `Xmax` = 100, `Ymax` = 101,
 `Wait` = 102, `Waitn` = 103, `Waitt` = 104, `Threshold` = 105,
 `Verbose` = 106, `Lecho` = 107, `Xtouch` = 110, `Ytouch` = 111,
 `Stouch` = 112, `Wquiet` = 113, `Wstack` = 114 }

Public Member Functions

- [EzLCD3](#) ()
- [~EzLCD3](#) ()
- void [debugWrite](#) (char b)
- virtual void [begin](#) (long baud)=0
- virtual size_t [write](#) (uint8_t)
- void [reset](#) ()
- void [cls](#) (int id=0)
- void [cls](#) (const char *color)
- void [wquiet](#) (void)
- unsigned int [wstack](#) (int cmd)
- bool [echo](#) ()
- void [echo](#) (bool val)
- void [ezLCDUpgrade](#) (void)
- int [light](#) ()
- void [light](#) (int brightness)
- void [light](#) (int brightness, unsigned long timeout)
- void [light](#) (int brightness, unsigned long timeout, int dimmed)
- void [color](#) (int id)
- void [color](#) (uint8_t *red, uint8_t *green, uint8_t *blue)
- void [colorId](#) (int id, uint8_t red, uint8_t green, uint8_t blue)
- void [colorId](#) (int id, uint8_t *red, uint8_t *green, uint8_t *blue)
- void [font](#) (int id=0)
- void [font](#) (const char *fontname)
- void [fontw](#) (int id, const char *fontname)
- void [fonto](#) (int orientation)
- int [fonto](#) ()
- void [calibrate](#) ()
- void [lineWidth](#) (int width)
- int [lineWidth](#) ()
- void [lineType](#) (int type)
- int [lineType](#) ()
- uint16_t [xmax](#) ()
- uint16_t [width](#) ()
- uint16_t [ymax](#) ()
- uint16_t [height](#) ()
- void [xy](#) (uint16_t x, uint16_t y)
- void [xyAligned](#) (uint32_t align)
- void [xyGet](#) (uint16_t *x, uint16_t *y)
- void [xy_store](#) (int id)
- void [xy_restore](#) (int id)
- void [string](#) (int id, const char *str)
- uint16_t [getPixel](#) (uint16_t x, uint16_t y)
- void [plot](#) ()
- void [plot](#) (uint16_t x, uint16_t y)
- void [line](#) (uint16_t x, uint16_t y)
- void [box](#) (uint16_t width, uint16_t height, bool fill=false)
- void [circle](#) (uint16_t radius, bool fill=false)
- void [arc](#) (uint16_t radius, int16_t start, int16_t end)
- void [pie](#) (uint16_t radius, int16_t start, int16_t end)
- void [picture](#) (int id, uint16_t x, uint16_t y, uint16_t option=0)
- void [picture](#) (const char *filename, uint16_t x, uint16_t y, uint16_t option=0)
- void [image](#) (int id, uint16_t x, uint16_t y, uint16_t option=0)
- void [image](#) (const char *filename, uint16_t x, uint16_t y, uint16_t option=0)

- void [fill](#) (bool isFilled)
- void [point](#) (int x, int y)
- void [line](#) (int x1, int y1, int x2, int y2)
- void [rect](#) (int x, int y, int [width](#), int [height](#))
- void [ellipse](#) (int x, int y, int diameter)
- void [ellipse](#) (int x, int y, int [width](#), int [height](#))
- void [arc](#) (int x, int y, int diameter, int start, int stop)
- void [clipenable](#) (bool enable)
- void [cliparea](#) (uint16_t left, uint16_t top, uint16_t right, uint16_t bottom)
- void [printStringId](#) (int id, uint32_t alignment=0)
- void [printAligned](#) (const char *str, uint32_t alignment=0)
- void [printString](#) (char *str)
- void [waitTouch](#) (unsigned long timeout=(unsigned long)-1)
- void [waitNoTouch](#) (unsigned long timeout=(unsigned long)-1)
- int [choice](#) (const char *str, int [theme](#), unsigned long timeout=(unsigned long)-1)
- void [theme](#) (int index, int embossDkColor, int embossLtColor, int textColor0, int textColor1, int textColorDisabled, int color0, int color1, int colorDisabled, int commonBkColor, int [fontw](#))
- void [button](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, uint16_t align, uint16_t radius, int [theme](#), int strid)
- void [touchZone](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option)
- void [checkbox](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int [theme](#), int strid)
- void [radioButton](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int [theme](#), int strid)
- void [groupBox](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int [theme](#), int strid)
- void [progressBar](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int initial, int range, int [theme](#), int suffix)
- void [staticText](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint32_t option, int [theme](#), int strid)
- void [digitalMeter](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int initial, int digits, int dotpos, int [theme](#))
- void [analogMeter](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int initial, int min, int max, int [theme](#), int strid, int type)
- void [dial](#) (int id, uint16_t x, uint16_t y, uint16_t radius, uint16_t option, int resolution, int initial, int max, int [theme](#))
- void [slider](#) (int id, uint16_t x, uint16_t y, uint16_t [width](#), uint16_t [height](#), uint16_t option, int max, int resolution, int initial, int [theme](#))
- void [analogMeterColor](#) (int id, int color1, int color2, int color3, int color4, int color5, int color6)
- void [wvalue](#) (int id, int value)
- bool [isPressed](#) (int id)
- bool [isChecked](#) (int id)
- unsigned int [wstate](#) (int id)
- int [getWidgetValue](#) (int id)
- int [touchS](#) ()
- int [touchX](#) ()
- int [touchY](#) ()

Protected Member Functions

- void [findEzLCD](#) ()
- bool [isHWSerial](#) ()
- unsigned int [parseHex](#) (unsigned long timeout)

Protected Attributes

- Stream * [m_pStream](#)

4.1.1 Detailed Description

Communicates with the Arduino via software and/or hardware serial port. This is a virtual class so the user cannot instantiate it. Instead the user will instantiate an [EzLCD3_HW](#) or [EZLCD3xx_SW](#) classes. See the end of this header file.

4.1.2 Member Enumeration Documentation

4.1.2.1 enum EzLCD3::Commands

Numerical values for the EarthSEMPLE commands. Provided here for users who wish to compose EarthSEMPLE commands manually. (This is a low- level asset that is not required for the common uses of the device)

Enumerator

Command Direct command.

Status

Clr_Screen Clear to provided color.

Ping Return Pong

zBeep Beep provided duration (frequency fixed)

Light 0 (off) to 100 (on)

Color

eColor_ID

Font Font number.

Fontw Font number widget.

Font_Orient Horizontal or vertical.

Line_Width 1 or 3.

Line_Type 1=dot dot 2=dash dash.

eXY X and Y.

StringID SID ASCII String or File Name that ends with 0.

Plot Place Pixel at X and Y.

eLine Draw a line to X and Y.

Box Draws a Box to X and Y optional fill.

zCircle Draws a Circle with Radius optional fill

Arc Draws an Arc with Radius and Begin Angle to End Angle.

Pie Draws a Pie figure with Radius and Begin Angle to End Angle and fills it.

Picture Places a Picture on display.

Print Places the string on display which ends with 0.

Beep_Freq Set the beeper frequency.

Get_Pixel get pixel >

Calibrate Calibrate touch screen.

zReset Reset.

Rec_Macro Record Macro to flash drive.

Play_Macro Play Macro.

Stop_Macro Stop Macro.

Pause_Macro Pause n msec.

Loop_Macro Loop on Macro.

Speed_Macro Set the macro speed.

Peri

ConfigIO

IO

IOG

Security Set drive security string.

Location LID Location Vlaue.

Upgrade

Parameters

ClipEnable Set clip Enable.

ClipArea Set clip area.

Comment

Fsgetcwd

Fschdir

Fsmkdir

Fsrmdir

Fsdir

Fscopy

Fsrename

Fsremove

Fsmore

Format Format Flash Drive if string1 = "ezLCD"

If

Cmd

Set_Button Widget Button.

Set_CheckBox Widget Checkbox.

Set_Gbox Widget Group Box.

Set_RadioButton Widget Radio Button.

Set_DMeter Widget Digital Meter.

DMeter_Value Set DMeter value.

Set_AMeter Widget Analog Meter.

AMeter_Value Set AMeter value.

AMeter_Color Set AMeter color

Set_TouchZone touch zone

Set_Dial Widget RoundDial.

Set_Slider Widget Slider.

Set_Progress Widget Progress bar.

Progress_Value Progress value.

Set_StaticText Widget Static text.

StaticText_Value Static text Value.

Choice Widget get choice.

Widget_Theme Widget Scheme.

Widget_Values Widget Values (Slider and Dial in this version).

Widget_State Widget State (Button, checkbox, radiobutton in this version).

Mode

Comport

Xmax Return Xmax width.
Ymax Return Ymax height.
Wait Wait for touch.
Waitn Wait for no touch.
Waitt Wait for touch.
Threshold Touch threshold.
Verbose Controls the verbose mode.
Lecho Controls the echo mode.
Xtouch return touchX.
Ytouch return touchY.
Stouch return touchS.
Wquiet
Wstack

4.1.3 Constructor & Destructor Documentation

4.1.3.1 EzLCD3::EzLCD3 ()

Class constructor

4.1.3.2 EzLCD3::~~EzLCD3 ()

Class destructor

4.1.4 Member Function Documentation

4.1.4.1 void EzLCD3::analogMeter (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint16_t *option*, int *initial*, int *min*, int *max*, int *theme*, int *strid*, int *type*)

Creates/alter an analog meter widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>strid</i>	String ID to use for text.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>initial</i>	Initial numeric value of the meter.
in	<i>min</i>	Minimum reading of the meter.
in	<i>max</i>	Maximum reading of the meter.
in	<i>option</i>	Option.
in	<i>type</i>	type.

4.1.4.2 void EzLCD3::analogMeterColor (int *id*, int *color1*, int *color2*, int *color3*, int *color4*, int *color5*, int *color6*)

Set value for an analog meter widget.

Parameters

in	<i>id</i>	Widget ID.
in	<i>color1</i>	color1
in	<i>color2</i>	color2
in	<i>color3</i>	color3
in	<i>color4</i>	color4
in	<i>color5</i>	color5
in	<i>color6</i>	color6

4.1.4.3 void EzLCD3::arc (uint16_t radius, int16_t start, int16_t end)

Draw an arc with the specified radius, start angle and end angle.

Parameters

in	<i>radius</i>	Arc radius in pixels.
in	<i>start</i>	Start angle in degrees.
in	<i>end</i>	End angle in degrees.

4.1.4.4 void EzLCD3::arc (int x, int y, int diameter, int start, int stop)

Graphics primitives similar to Processing and GLCD.

Draw an arc with the specified radius, start angle and end angle at x y.

Parameters

in	<i>x</i>	X
in	<i>y</i>	Y
in	<i>diameter</i>	diameter of arc
in	<i>start</i>	Start angle in degrees.
in	<i>stop</i>	End angle in degrees.

4.1.4.5 virtual void EzLCD3::begin (long baud) [pure virtual]

Initialize communication at the specified baud rate and wait for ezLCD to get ready to accept commands. Implementation depends on hardware vs software serial.

Parameters

in	<i>baud</i>	Baud rate
----	-------------	-----------

Implemented in [EzLCD3_SW](#), and [EzLCD3_HW](#).

4.1.4.6 void EzLCD3::box (uint16_t width, uint16_t height, bool fill = false)

Draw a box from the current x,y with specified width, height, and fill.

Parameters

in	<i>width</i>	Box width in pixels.
in	<i>height</i>	Box height in pixels.
in	<i>fill</i>	Set to <code>true</code> to fill the solid with box. Default is <code>false</code> which only draws the box outline.

4.1.4.7 `void EzLCD3::button (int id, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t option, uint16_t align, uint16_t radius, int theme, int strid)`

Draw/alter a button widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>strid</i>	String ID to use for text.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>radius</i>	Controls roundness of the button edges. Radius of 0 (default) means button corners are perfect right angles. Radius of half the size of the button results in an edge that is round.
in	<i>align</i>	Text alignment. Allowed values are CENTER (default), LEFT, RIGHT or BOTTOM
in	<i>option</i>	Option.

4.1.4.8 `void EzLCD3::calibrate ()`

Invoke a touch screen calibrate.

4.1.4.9 `void EzLCD3::checkbox (int id, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t option, int theme, int strid)`

Draw/alter a checkbox.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>strid</i>	String ID to use for text.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>option</i>	Option.

4.1.4.10 `int EzLCD3::choice (const char * str, int theme, unsigned long timeout = (unsigned long)-1)`

Show a YES, NO and CANCEL prompt, wait for input and return the result.

Parameters

in	<i>str</i>	String to display in the prompt
in	<i>theme</i>	Theme to use for drawing the prompt.
in	<i>timeout</i>	Timeout value in milliseconds before giving up on waiting. Default is very long.

Return values

YES	User pressed Yes.
NO	User pressed No.
CANCEL	User pressed Cancel.

4.1.4.11 void EzLCD3::circle (uint16_t *radius*, bool *fill* = false)

Draw a circle at current x,y with the specified radius and fill.

Parameters

in	<i>radius</i>	Circle radius in pixels.
in	<i>fill</i>	Set to <code>true</code> to fill the circle solid with color. Default is <code>false</code> which only draws the circle outline

4.1.4.12 void EzLCD3::cliparea (uint16_t *left*, uint16_t *top*, uint16_t *right*, uint16_t *bottom*)

Set the clip area to protect the surrounding area from change. The parameters are limiting coordinates of the clip area.

Parameters

in	<i>left</i>	Left edge of the clip area in pixels.
in	<i>top</i>	Top edge of the clip area in pixels.
in	<i>right</i>	Right edge of the clip area in pixels.
in	<i>bottom</i>	Bottom edge of the clip area in pixels.

4.1.4.13 void EzLCD3::clipenable (bool *enable*)

Enable or disable the clip area.

Parameters

in	<i>enable</i>	<code>true</code> to enable, <code>false</code> to disable.
----	---------------	-------------------------------------------------------------

4.1.4.14 void EzLCD3::cls (int *id* = 0)

Clear screen to a specific color id and clear widgets.

Parameters

in	<i>id</i>	Numeric value for the color. 0 (black) by default.
----	-----------	----------------------------------------------------

4.1.4.15 void EzLCD3::cls (const char * *color*)

Clear screen to a specific color and clear widgets.

Parameters

in	<i>color</i>	Null-terminated string describing color.
----	--------------	------------------------------------------

4.1.4.16 void EzLCD3::color (int *id*)

Sets the current color by color ID.

Parameters

in	<i>id</i>	Numeric color ID.
----	-----------	-------------------

4.1.4.17 void EzLCD3::color (uint8_t * *red*, uint8_t * *green*, uint8_t * *blue*)

Return current color.

Parameters

out	<i>red</i>	Red value in 0-255.
out	<i>green</i>	Green value in 0-255.
out	<i>blue</i>	Blue value in 0-255.

4.1.4.18 void EzLCD3::colorId (int *id*, uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Sets a color index to a specific RGB value

Parameters

in	<i>id</i>	Numeric color ID.
in	<i>red</i>	Red value in 0-255.
in	<i>green</i>	Green value in 0-255.
in	<i>blue</i>	Blue value in 0-255.

4.1.4.19 void EzLCD3::colorId (int *id*, uint8_t * *red*, uint8_t * *green*, uint8_t * *blue*)

Get RGB value of a color index

Parameters

in	<i>id</i>	Numeric color ID.
out	<i>red</i>	Red value in 0-255.
out	<i>green</i>	Green value in 0-255.
out	<i>blue</i>	Blue value in 0-255.

4.1.4.20 void EzLCD3::debugWrite (char *b*)

4.1.4.21 void EzLCD3::dial (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *radius*, uint16_t *option*, int *resolution*, int *initial*, int *max*, int *theme*)

Create/alter a dial widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>radius</i>	Radius of the dial in pixels.
in	<i>resolution</i>	Resolution of the dial in degrees.
in	<i>initial</i>	Initial numeric position of the dial.
in	<i>max</i>	Maximum numeric position of the dial.
in	<i>option</i>	1=draw, 2=disabled, 3=ring, 4=accuracy

4.1.4.22 `void EzLCD3::digitalMeter (int id, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t option, int initial, int digits, int dotpos, int theme)`

Creates/alter a digital meter widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>initial</i>	Initial numeric value of the meter.
in	<i>digits</i>	Total number of digits.
in	<i>dotpos</i>	Dot position counting digits from the left.
in	<i>option</i>	Option

4.1.4.23 `bool EzLCD3::echo () [inline]`

Returns whether the class was configured for echo

Return values

<i>true</i>	The class is configured for echo.
<i>false</i>	The class is configured for NO echo

4.1.4.24 `void EzLCD3::echo (bool val)`

Enable or disable echo

Parameters

in	<i>val</i>	true to enable echo. false to disable echo.
----	------------	---------------------------------------------

4.1.4.25 `void EzLCD3::ellipse (int x, int y, int diameter)`

Graphics primitives similar to Processing and GLCD.

4.1.4.26 `void EzLCD3::ellipse (int x, int y, int width, int height)`

Graphics primitives similar to Processing and GLCD.

4.1.4.27 `void EzLCD3::ezLCDUpgrade (void)`

4.1.4.28 `void EzLCD3::fill (bool isFilled)`

Graphics primitives similar to Processing and GLCD.

Parameters

in	<i>isFilled</i>	set true filled boxes and circles
----	-----------------	-----------------------------------

4.1.4.29 void EzLCD3::findEzLCD () [protected]

Establish communication with the ezLCD by sending "ping" commands and waiting for a response.

4.1.4.30 void EzLCD3::font (int *id* = 0)

Set current font to an internal factory font.

Parameters

<i>in</i>	<i>id</i>	Factory font index. Currently 0 to 5. Default is 0, default medium font.
-----------	-----------	--------------------------------------------------------------------------

4.1.4.31 void EzLCD3::font (const char * *fontname*)

Set current font to a programmable font (ezf file) from flash drive

Parameters

<i>in</i>	<i>fontname</i>	Font filename on the ezLCD filesystem.
-----------	-----------------	----------------------------------------

4.1.4.32 void EzLCD3::fonto (int *orientation*)

Set current font orientation to horizontal or vertical

Parameters

<i>in</i>	<i>orientation</i>	HORIZONTAL for horizontal orientation. VERTICAL for vertical orientation.
-----------	--------------------	---------------------------------------------------------------------------

4.1.4.33 int EzLCD3::fonto ()

Returns current font orientation.

Return values

<i>HORIZONTAL</i>	Horizontal orientation.
<i>VERTICAL</i>	Vertical orientation.

4.1.4.34 void EzLCD3::fontw (int *id*, const char * *fontname*)

Set font index (used by themes) to a programmable font (ezf file) from flash drive.

Parameters

<i>in</i>	<i>id</i>	Font ID.
<i>in</i>	<i>fontname</i>	Font filename on the ezLCD filesystem.

4.1.4.35 uint16_t EzLCD3::getPixel (uint16_t *x*, uint16_t *y*)

Get pixel value at x y

4.1.4.36 int EzLCD3::getValue (int *id*)

Get the value of the widget with the given id Used to get the value of a slider or dial

Parameters

in	<i>id</i>	Widget ID you want to check (slider or dial).
----	-----------	-----------------------------------------------

Returns

returns the numeric value for the widget.

Return values

-1	There is no information of the slider with the specified ID having it's position changed.
----	-------------------------------------------------------------------------------------------

4.1.4.37 void EzLCD3::groupBox (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint16_t *option*, int *theme*, int *strid*)

Create/alter a groupox.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>option</i>	Option
in	<i>theme</i>	Theme ID to use.
in	<i>strid</i>	String ID to use for text.

4.1.4.38 uint16_t EzLCD3::height () [inline]

Return the screen height.

Returns

Screen height in pixels.

4.1.4.39 void EzLCD3::image (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *option* = 0)

Draw an image by id on ezLCD. Same as [picture\(\)](#) function.

Parameters

in	<i>id</i>	ID of the image to display.
in	<i>x</i>	x-coordinate in pixels of where to draw.
in	<i>y</i>	y-coordinate in pixels of where to draw.
in	<i>option</i>	option

4.1.4.40 `void EzLCD3::image (const char * filename, uint16_t x, uint16_t y, uint16_t option = 0)`

Draw an image from file on the ezLCD. Same as `picture()` function.

Parameters

in	<i>filename</i>	Filename of the image to display. Must include extension.
in	<i>x</i>	x-coordinate in pixels of where to draw.
in	<i>y</i>	y-coordinate in pixels of where to draw.
in	<i>option</i>	option .

4.1.4.41 `bool EzLCD3::isChecked (int id)`

return true if the checkbox or radio button with the given id is checked

Parameters

in	<i>id</i>	Widget ID of the checkbox you want to check.
----	-----------	----------------------------------------------

Return values

<i>true</i>	ezLCD signaled that the checkbox was checked.
<i>false</i>	not checked return is only meaningful if id is for a valid checkbox

4.1.4.42 `bool EzLCD3::isHWSerial ()` [protected]

Return whether the hardware or software serial class is being used.

Return values

<i>true</i>	HardwareSerial class is being used.
<i>false</i>	SoftwareSerial class is being used.

4.1.4.43 `bool EzLCD3::isPressed (int id)`

return true if the button with the given id is pressed

Parameters

in	<i>id</i>	Widget ID of the button you want to check.
----	-----------	--------------------------------------------

Return values

<i>true</i>	ezLCD signaled that the button is pressed.
<i>false</i>	not pressed return is only meaningful if id is for a valid checkbox

4.1.4.44 `int EzLCD3::light ()`

Return current brightness setting.

Returns

Brightness in 0-100.

4.1.4.45 void EzLCD3::light (int *brightness*)

Set brightness.

Parameters

in	<i>brightness</i>	Brightness in 0-100.
----	-------------------	----------------------

4.1.4.46 void EzLCD3::light (int *brightness*, unsigned long *timeout*)

Set brightness & timeout

Parameters

in	<i>brightness</i>	Brightness in 0-100.
in	<i>timeout</i>	Timeout value in minutes before dimming

4.1.4.47 void EzLCD3::light (int *brightness*, unsigned long *timeout*, int *dimmed*)

Set brightness, timeout, dimmed brightness

Parameters

in	<i>brightness</i>	Brightness in 0-100.
in	<i>timeout</i>	Timeout value in minutes before dimming
in	<i>dimmed</i>	Dimmed brightness level in 0-100

4.1.4.48 void EzLCD3::line (uint16_t *x*, uint16_t *y*)

Draw a line from the current x,y to the specified x,y

Parameters

in	<i>x</i>	x-coordinate in pixels to draw the line to.
in	<i>y</i>	y-coordinate in pixels to draw the line to.

4.1.4.49 void EzLCD3::line (int *x1*, int *y1*, int *x2*, int *y2*)

Graphics primitives similar to Processing and GLCD.

Draw a line from x1 y1 to x2 y2 with the current color.

Parameters

in	<i>x1</i>	x1-coordinate in pixels
in	<i>y1</i>	y1-coordinate in pixels
in	<i>x2</i>	x2-coordinate in pixels
in	<i>y2</i>	y2-coordinate in pixels

4.1.4.50 int EzLCD3::lineTtype ()

Return current line type.

Returns

0 when solid, higher number for dash and even higher for dot.

4.1.4.51 void EzLCD3::lineType (int *type*)

Set line type to solid, dot or dash.

Parameters

<i>in</i>	<i>type</i>	0 is solid, increasing number increases spacing between dots.
-----------	-------------	---------------------------------------------------------------

4.1.4.52 void EzLCD3::lineWidth (int *width*)

Set current line width

Parameters

<i>in</i>	<i>width</i>	Line width in pixels. Can be 1 or 3 pixels.
-----------	--------------	---------------------------------------------

4.1.4.53 int EzLCD3::lineWidth ()

Returns current line width

Returns

Line width in pixels. 1 or 3 pixels.

4.1.4.54 unsigned int EzLCD3::parseHex (unsigned long *timeout*) [protected]

Alternative to Stream parseInt that can parse hexadecimal numbers

Return values

<i>the</i>	integer value of sequence of hex charactres
------------	---------------------------------------------

4.1.4.55 void EzLCD3::picture (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *option* = 0)

Draw a picture by id on ezLCD. Same as [image\(\)](#) function.

Parameters

<i>in</i>	<i>id</i>	ID of the picture to display.
<i>in</i>	<i>x</i>	x-coordinate in pixels of where to draw.
<i>in</i>	<i>y</i>	y-coordinate in pixels of where to draw.
<i>in</i>	<i>option</i>	option.

4.1.4.56 void EzLCD3::picture (const char * *filename*, uint16_t *x*, uint16_t *y*, uint16_t *option* = 0)

Draw a picture from file on the ezLCD. Same as [image\(\)](#) function.

Parameters

in	<i>filename</i>	Filename of the picture to display. Must include extension.
in	<i>x</i>	x-coordinate in pixels of where to draw.
in	<i>y</i>	y-coordinate in pixels of where to draw.
in	<i>option</i>	option.

4.1.4.57 `void EzLCD3::pie (uint16_t radius, int16_t start, int16_t end)`

Draw a pie with the specified radius, start angle and end angle.

Parameters

in	<i>radius</i>	Arc radius in pixels.
in	<i>start</i>	Start angle in degrees.
in	<i>end</i>	End angle in degrees.

4.1.4.58 `void EzLCD3::plot ()`

Draw a pixel at the current x and y with the current color.

4.1.4.59 `void EzLCD3::plot (uint16_t x, uint16_t y)`

Draw a pixel at a specified x,y with current color.

Parameters

in	<i>x</i>	x-coordinate in pixels.
in	<i>y</i>	y-coordinate in pixels.

4.1.4.60 `void EzLCD3::point (int x, int y)`

Graphics primitives similar to Processing and GLCD.

Draw a pixel at the current x and y with the current color.

Parameters

in	<i>x</i>	x-coordinate in pixels
in	<i>y</i>	y-coordinate in pixels

4.1.4.61 `void EzLCD3::printAligned (const char * str, uint32_t alignment = 0)`

Print string to the display at the current x,y with given alignment.

Parameters

in	<i>str</i>	Null-terminated string to print.
in	<i>alignment</i>	Text alignment. Allowed values are LEFTTOP, TOPLEFT, TOP, RIGHTTOP, TOPRIGHT, LEFT, CENTER, RIGHT, LEFTBOTTOM, BOTTOMLEFT, BOTTOM, RIGHTBOTTOM, BOTTOMRIGHT, or NONE.

4.1.4.62 void EzLCD3::printString (char * *str*)

Print string direct to LCD skipping Arduino filtering faster set xy before printing

Parameters

in	<i>str</i>	Null-terminated string to print.
----	------------	----------------------------------

4.1.4.63 void EzLCD3::printStringId (int *id*, uint32_t *alignment* = 0)

Print a string from the string array on ezLCD by id.

Parameters

in	<i>id</i>	String ID.
in	<i>alignment</i>	Text alignment. Allowed values are LEFTTOP, TOPLEFT, TOP, RIGHTTOP, TOPRIGHT, LEFT, CENTER, RIGHT, LEFTBOTTOM, BOTTOMLEFT, BOTTOM, RIGHTBOTTOM, BOTTOMRIGHT, or NONE.

4.1.4.64 void EzLCD3::progressBar (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint16_t *option*, int *initial*, int *range*, int *theme*, int *suffix*)

Creates/alter a progress widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>initial</i>	Initial numeric reading of the progress bar. Default is 0.
in	<i>range</i>	Maximum allowed value of the progress bar. Default is 100.
in	<i>option</i>	Option
in	<i>suffix</i>	Char to display at end of number text % default

4.1.4.65 void EzLCD3::radioButton (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint16_t *option*, int *theme*, int *strid*)

Draw/alter a radio button widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>strid</i>	String ID to use for text.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>option</i>	Option:

4.1.4.66 void EzLCD3::rect (int *x*, int *y*, int *width*, int *height*)

Graphics primitives similar to Processing and GLCD.

Draw a rectangle at x y of width and height with the current color.

And filled if fill is set true

Parameters

in	<i>x</i>	x-coordinate in pixels
in	<i>y</i>	y-coordinate in pixels
in	<i>width</i>	width of rect in pixels
in	<i>height</i>	height of rect in pixels

4.1.4.67 void EzLCD3::reset ()

Reset ezLCD and re-establish communication.

4.1.4.68 void EzLCD3::slider (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint16_t *option*, int *max*, int *resolution*, int *initial*, int *theme*)

Crete/alter a slider widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>resolution</i>	per step
in	<i>initial</i>	Initial numeric value of the slider.
in	<i>max</i>	Maximum numeric value of the slider.
in	<i>option</i>	Option

4.1.4.69 void EzLCD3::staticText (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint32_t *option*, int *theme*, int *strid*)

Create/alter a static text widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>theme</i>	Theme ID to use.
in	<i>strid</i>	String ID to use for text.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>option</i>	Option

4.1.4.70 void EzLCD3::string (int *id*, const char * *str*)

Store string at an index in the string array on ezLCD.

Parameters

in	<i>id</i>	String ID at which to store.
in	<i>str</i>	Null-terminated string to store.

4.1.4.71 void EzLCD3::theme (int *index*, int *embossDkColor*, int *embossLtColor*, int *textColor0*, int *textColor1*, int *textColorDisabled*, int *color0*, int *color1*, int *colorDisabled*, int *commonBkColor*, int *fontw*)

Set up a widget theme

Parameters

in	<i>index</i>	Theme Index
in	<i>embossDkColor</i>	Dark Emboss color used for 3-D effect of objects.
in	<i>embossLtColor</i>	Light Emboss color used for 3-D effect of objects .
in	<i>textColor0</i>	For text, Useage may vary from one widget to another, whether the widget is in focus or not.
in	<i>textColor1</i>	For text when pressed. Useage may vary from one widget to another.
in	<i>textColorDisabled</i>	Color of objects that are disabled.
in	<i>color0</i>	For objects. Usage may vary from one object to another, whether the widget is in focus or not.
in	<i>color1</i>	For objects when pressed. Useage may vary from one object to another.
in	<i>colorDisabled</i>	Used to render objects that are disabled.
in	<i>commonBkColor</i>	Used to hide objects from screen but still active.
in	<i>fontw</i>	Font id associated with this theme.

4.1.4.72 int EzLCD3::touchS ()

Return the touch status 0 = not currently pressed, 3= pressed, 4 = released

4.1.4.73 int EzLCD3::touchX ()

Return the x (horizontal) coordinate of the last location where the screen was touched.

4.1.4.74 int EzLCD3::touchY ()

Return the y (vertical) coordinate of the last location where the screen was touched.

4.1.4.75 void EzLCD3::touchZone (int *id*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *height*, uint16_t *option*)

Draw/alter a touchZone widget.

Parameters

in	<i>id</i>	Widget ID to assign.
in	<i>x</i>	Starting x-coordinate in pixels.
in	<i>y</i>	Starting y-coordinate in pixels.
in	<i>width</i>	Width in pixels.
in	<i>height</i>	Height in pixels.
in	<i>option</i>	Option.

4.1.4.76 `void EzLCD3::waitNoTouch (unsigned long timeout = (unsigned long)-1)`

Wait for release of touch.

Parameters

<i>in</i>	<i>timeout</i>	Timeout value in milliseconds before we give up on waiting. Default is very long.
-----------	----------------	-----------------------------------------------------------------------------------

4.1.4.77 `void EzLCD3::waitTouch (unsigned long timeout = (unsigned long)-1)`

Wait for touch.

Parameters

<i>in</i>	<i>timeout</i>	Timeout value in milliseconds before we give up on waiting. Default is very long.
-----------	----------------	-----------------------------------------------------------------------------------

4.1.4.78 `uint16_t EzLCD3::width () [inline]`

Return the screen width.

Returns

Screen width in pixels.

4.1.4.79 `void EzLCD3::wquiet (void)`

turn off widget output used for polling and interrupts .

4.1.4.80 `size_t EzLCD3::write (uint8_t value) [inline],[virtual]`

4.1.4.81 `unsigned int EzLCD3::wstack (int cmd)`

Gets one value off widget stack.

Parameters

<i>in</i>	<i>cmd</i>	= FIFO will return in first in first out order
<i>in</i>	<i>cmd</i>	= LIFO will return in last in first out order
<i>in</i>	<i>cmd</i>	= CLEAR will clear the stack

Return values

<i>value</i>	one unsigned int off stack high byte will be widget ID low byte status <code>lcd.wstack (LIFO)</code> will return the last widget pushed and state
--------------	----------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.4.82 `unsigned int EzLCD3::wstate (int id)`

Get the state of the widget with the given id Used to get the value of widget

Parameters

<i>in</i>	<i>id</i>	Widget ID you want to check .
-----------	-----------	-------------------------------

Return values

<i>value</i>	of widget
--------------	-----------

4.1.4.83 void EzLCD3::wvalue (int *id*, int *value*)

Set value for a widget.

Parameters

in	<i>id</i>	Widget ID.
in	<i>value</i>	Numeric value to set.

4.1.4.84 uint16_t EzLCD3::xmax ()

Return the maximum allowed x-coordinate.

Returns

Maximum allowed x-coordinate in pixels.

4.1.4.85 void EzLCD3::xy (uint16_t *x*, uint16_t *y*)

Set the drawing cursor to location x,y on screen.

Parameters

in	<i>x</i>	x-coordinate in pixels.
in	<i>y</i>	y-coordinate in pixels.

4.1.4.86 void EzLCD3::xy_restore (int *id*)

Restore current x and y from the x,y array on ezLCD.

Parameters

in	<i>id</i>	Index from which x and y are restored.
----	-----------	----------------------------------------

4.1.4.87 void EzLCD3::xy_store (int *id*)

Store current x and y into x,y array on ezLCD.

Parameters

in	<i>id</i>	Index where x and y is stored.
----	-----------	--------------------------------

4.1.4.88 void EzLCD3::xyAligned (uint32_t *align*)

Set the drawing cursor to a preset aligned location.

Parameters

in	align	Alignment. Allowed values are LEFTTOP, TOPLEFT, TOP, RIGHTTOP, TOPRIGHT, LEFT, CENTER, RIGHT, LEFTBOTTOM, BOTTOMLEFT, BOTTOM, RIGHTBOTTOM, BOTTOMRIGHT
----	-------	--------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.4.89 void EzLCD3::xyGet (uint16_t * x, uint16_t * y)

Return current x,y drawing location.

Parameters

out	x	x-coordinate in pixels.
out	y	y-coordinate in pixels.

4.1.4.90 uint16_t EzLCD3::ymax ()

Return the maximum allowed y-coordinate.

Returns

Maximum allowed y-coordinate in pixels.

4.1.5 Member Data Documentation

4.1.5.1 Stream* EzLCD3::m_pStream [protected]

Pointer to an instance of a serial communication class which is either `HardwareSerial` (provided by Arduino) or `SoftwareSerial` (provided in Arduino 1.0 and written by David A. Mellis).

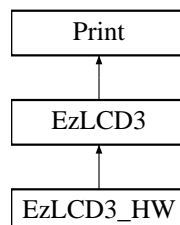
The documentation for this class was generated from the following files:

- [C:/Users/Segler/arduino-1.0.4/libraries/arduino/ezLCD.h](#)
- [C:/Users/Segler/arduino-1.0.4/libraries/arduino/ezLCD.cpp](#)

4.2 EzLCD3_HW Class Reference

```
#include <ezLCD.h>
```

Inheritance diagram for EzLCD3_HW:



Public Member Functions

- [EzLCD3_HW](#) ()
- void [begin](#) (long baud)

Additional Inherited Members

4.2.1 Detailed Description

Class derived from [EzLCD3](#) that uses the Arduino's hardware serial class `HardwareSerial`.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 EzLCD3_HW::EzLCD3_HW ()

Class constructor. Requires no parameters because hardware serial implies `pin 0` for receive and `pin 1` for transmit.

4.2.3 Member Function Documentation

4.2.3.1 void EzLCD3_HW::begin (long *baud*) [virtual]

Initialize communication at the specified baud rate and wait for ezLCD to get ready to accept commands. Implementation depends on hardware vs software serial.

Parameters

<i>in</i>	<i>baud</i>	Baud rate
-----------	-------------	-----------

Implements [EzLCD3](#).

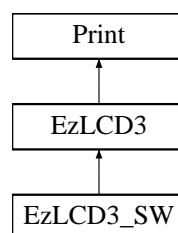
The documentation for this class was generated from the following files:

- C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.h
- C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.cpp

4.3 EzLCD3_SW Class Reference

```
#include <ezLCD.h>
```

Inheritance diagram for EzLCD3_SW:



Public Member Functions

- [EzLCD3_SW](#) (int rx, int tx)
- void [begin](#) (long baud)

Additional Inherited Members

4.3.1 Detailed Description

Class that derived from [EzLCD3](#) that uses Arduino's software serial class `SoftwareSerial`.

Examples:

[analogMeter.ino](#), [arc.ino](#), [button.ino](#), [button_interrupt.ino](#), [checkbox.ino](#), [choice.ino](#), [circle.ino](#), [cls.ino](#), [color.ino](#), [colorid.ino](#), [dial.ino](#), [digitalMeter.ino](#), [ellipse.ino](#), [font.ino](#), [groupbox.ino](#), [image.ino](#), [light.ino](#), [line.ino](#), [lineType.ino](#), [lineWidth.ino](#), [pie.ino](#), [plot.ino](#), [point.ino](#), [printExample.ino](#), [progress.ino](#), [radio.ino](#), [radio_interrupt.ino](#), [rect.ino](#), [slider.ino](#), [theme.ino](#), and [touch.ino](#).

4.3.2 Constructor & Destructor Documentation

4.3.2.1 EzLCD3_SW::EzLCD3_SW (int rx, int tx)

Class constructor

Parameters

<code>in</code>	<code>rx</code>	Receive pin to be used.
<code>in</code>	<code>tx</code>	Transmit pin to be used.

4.3.3 Member Function Documentation

4.3.3.1 void EzLCD3_SW::begin (long baud) [virtual]

Initialize communication at the specified baud rate and wait for ezLCD to get ready to accept commands. Implementation depends on hardware vs software serial.

Parameters

<code>in</code>	<code>baud</code>	Baud rate
-----------------	-------------------	-----------

Implements [EzLCD3](#).

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

- [C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.h](#)
- [C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.cpp](#)

Chapter 5

File Documentation

5.1 C:/Users/Segler/arduino-1.0.4/libraries/arLCD/examples.lst File Reference

5.2 C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.cpp File Reference

```
#include <SoftwareSerial.h>
#include <Arduino.h>
#include "ezLCD.h"
```

Macros

- #define `sdebug`(str)
- #define `sdebugln`(str)
- #define `DEFAULT_TIMEOUT` 100
- #define `PARSE_TIMEOUT` 100
- #define `SEMPLEP` ''

5.2.1 Macro Definition Documentation

5.2.1.1 #define `DEFAULT_TIMEOUT` 100

5.2.1.2 #define `PARSE_TIMEOUT` 100

5.2.1.3 #define `sdebug(str)`

ezlcd3xx.c Source file for the ezlcd3xx Arduino library for ezLCD-3xx devices by EarthLCD.com. The library communicates with the ezLCD via a hardware and/or serial port

5.2.1.4 #define `sdebugln(str)`

5.2.1.5 #define `SEMPLEP` ''

5.3 C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.h File Reference

```
#include <SoftwareSerial.h>
#include "Print.h"
```

Classes

- class [EzLCD3](#)
- class [EzLCD3_HW](#)
- class [EzLCD3_SW](#)

Macros

- `#define EZLCD_PROCESSING_PRIMITIVES`
- `#define EZM_BAUD_RATE 38400`
- `#define EZLCD_SERIALDEBUG 0`
- `#define EZLCD_GLOBALENUMS 1`
- `#define EZLCD_ENUM_ORIENTATION enum { HORIZONTAL=0, VERTICAL=1 };`
- `#define EZLCD_ENUM_ALIGNMENT enum { CENTER=0x100, TOP=0x200, RIGHT=0x400, BOTTOM=0x800, LEFT=0x1000 };`
- `#define EZLCD_ENUM_COMBALIGN`
- `#define EZLCD_ENUM_WIDGETS enum { NONE=0, DOWNSIZE=0x1, BOTH = DOWNSIZE | CENTER };`
- `#define EZLCD_ENUM_CHOICE enum { YES=1, NO=0, CANCEL=-1 };`
- `#define FIFO 0`
- `#define LIFO 1`
- `#define CLEAR 2`

Enumerations

- enum {
[BLACK](#), [GRAY](#), [SILVER](#), [WHITE](#),
[RED](#), [MAROON](#), [YELLOW](#), [OLIVE](#),
[LIME](#), [GREEN](#), [AQUA](#), [TEAL](#),
[BLUE](#), [NAVY](#), [FUCHSIA](#), [PURPLE](#) }

5.3.1 Macro Definition Documentation

5.3.1.1 `#define CLEAR 2`

5.3.1.2 `#define EZLCD_ENUM_ALIGNMENT enum { CENTER=0x100, TOP=0x200, RIGHT=0x400, BOTTOM=0x800, LEFT=0x1000 };`

Enum values for alignment

5.3.1.3 `#define EZLCD_ENUM_CHOICE enum { YES=1, NO=0, CANCEL=-1 };`

Enum for choice

5.3.1.4 `#define EZLCD_ENUM_COMBALIGN`

Value:

```
enum { LEFTTOP = TOP | LEFT, RIGHTTOP = TOP | RIGHT }; \
enum { TOPLEFT = LEFTTOP, TOPRIGHT = RIGHTTOP }; \
enum { LEFTBOTTOM = BOTTOM | LEFT, RIGHTBOTTOM = BOTTOM | RIGHT }; \
enum { BOTTOMLEFT = LEFTBOTTOM, BOTTOMRIGHT = RIGHTBOTTOM };
```

Enum values for convenient combined alignment

5.3.1.5 `#define EZLCD_ENUM_ORIENTATION enum { HORIZONTAL=0, VERTICAL=1 };`

Enum values for orientation

5.3.1.6 `#define EZLCD_ENUM_WIDGETS enum { NONE=0, DOWNSIZE=0x1, BOTH = DOWNSIZE | CENTER };`

Enum for widget and picture options

5.3.1.7 `#define EZLCD_GLOBALENUMS 1`

Set to 1 to globally declare numeric constants used by the [EzLCD3](#) class. Set to 0 to declare the numeric constants as enums local to the [EzLCD3](#) class

5.3.1.8 `#define EZLCD_PROCESSING_PRIMITIVES`

ezlcd3xx.h Header for the ezlcd3xx Arduino library for ezLCD-3xx devices by EarthLCD.com The library communicates with the ezLCD via a hardware and/or serial port

5.3.1.9 `#define EZLCD_SERIALDEBUG 0`

Set to 1 to enable debugging the library over the hardware serial port. When doing so, make sure you are using [EzLCD3_SW](#) class and not using the hardware serial in any other way. Set to 0 to disable serial debugging, which frees up the hardware serial port and additional processing, and results in smaller code. Keep this at 0 when you are using [EzLCD3_HW](#) class.

5.3.1.10 `#define EZM_BAUD_RATE 38400`

Examples:

[analogMeter.ino](#), [arc.ino](#), [button.ino](#), [button_interrupt.ino](#), [checkbox.ino](#), [choice.ino](#), [circle.ino](#), [cls.ino](#), [color.ino](#), [colorid.ino](#), [dial.ino](#), [digitalMeter.ino](#), [ellipse.ino](#), [font.ino](#), [groupbox.ino](#), [image.ino](#), [light.ino](#), [line.ino](#), [lineType.ino](#), [lineWidth.ino](#), [pie.ino](#), [plot.ino](#), [point.ino](#), [printExample.ino](#), [progress.ino](#), [radio.ino](#), [radio_interrupt.ino](#), [rect.ino](#), [slider.ino](#), [theme.ino](#), and [touch.ino](#).

5.3.1.11 `#define FIFO 0`

5.3.1.12 `#define LIFO 1`

5.3.2 Enumeration Type Documentation

5.3.2.1 anonymous enum

Enum for standard colors

Enumerator

BLACK

GRAY

SILVER

WHITE

RED

MAROON

YELLOW

OLIVE

LIME

GREEN

AQUA

TEAL

BLUE

NAVY

FUCHSIA

PURPLE

Chapter 6

Example Documentation

6.1 analogMeter.ino

analog meter demo

```
/*
 * analogMeter.ino displays analog meter
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

int xPos = 50; // horizontal position
int yPos = 30; // vertical position
int width = 200;
int height = 200;
int option = 1; // 1=draw, 2=disabled, 3=ring, 4=accuracy
int type = 0; // 0=full, 1=half, 2=quarter.

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls( WHITE );
    lcd.fontw( 1, "0" );
    lcd.string( 1, "ALOG_0" ); // stringId 1
    lcd.theme( 1, 155, 152, 3, 3, 3, 1, 4, 5, 0, 16 );
    lcd.analogMeter( 1, xPos, yPos, width, height, option, 0, 0, 1023, 1, 1, type );
}

void loop()
{
    int value = analogRead(0);
    lcd.wvalue(1, value);
}
```

6.2 arc.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls();
}

void loop(){
    int x = random(0,319);
    int y = random(0,239);
    lcd.xy(x,y);
}
```

```

    lcd.color(random(1,200));
    lcd.arc(random(1,100), random(1,360), random(1,360) );
}

```

6.3 button.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );

int x1Pos = 10; // horizontal position for buttton 1
int x2Pos = 210; // horizontal position for buttton 2
int yPos = 70; // vertical position of both buttons
int width = 100;
int height = 100;
int radius = 20;
int alignment = 0; // 0=centered, 1=right, 2=left, 3=bottom, 4=top
int option = 1; // 1=draw, 2=disabled, 3=toggle pressed, 4=toggle not pressed,
// 5=toggle pressed disabled, 6=toggle not pressed disabled.

void setup()
{
    Serial.begin(9600);
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls( 0 );
    lcd.font(0);
    lcd.println("    Button Test To Turn On LED On D13");
    lcd.fontw( 1, "sans24" );
    lcd.theme( 1, 9, 3, 0, 0, 0, 8, 8, 8, 1, 1 );
    lcd.theme( 2, 5, 20, 3, 3, 3, 4, 4, 4, 2, 1 );
    lcd.string( 1, "ON" ); // stringId 1
    lcd.string( 2, "OFF" ); // stringId 2
    lcd.button( 1, x1Pos, yPos, width, height, option, alignment, radius, 1, 1 );
    lcd.button( 2, x2Pos, yPos, width, height, option, alignment, radius, 2, 2 );

    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, LOW );
}

void loop()
{
    if( lcd.isPressed(1) ) // if Button 1 is pressed:
        digitalWrite( LED_PIN, HIGH ); // turn LED on
    else if( lcd.isPressed(2) ) // if Button 2 was pressed:
        digitalWrite( LED_PIN, LOW ); // turn LED off
}

```

6.4 button_interrupt.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );
volatile boolean ezLCDInt = false; // flag to indicate interrupt

int x1Pos = 10; // horizontal position for buttton 1
int x2Pos = 210; // horizontal position for buttton 2
int yPos = 70; // vertical position of both buttons
int width = 100;
int height = 100;
int radius = 20;
int alignment = 0; // 0=centered, 1=right, 2=left, 3=bottom, 4=top
int option = 1; // 1=draw, 2=disabled, 3=toggle pressed, 4=toggle not pressed,
// 5=toggle pressed disabled, 6=toggle not pressed disabled.

void setup()
{
    Serial.begin(9600);
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls( 0 );
    lcd.font(0);
    lcd.println("    Button Test To Turn On LED On D13\r");
    lcd.println("                Using Interrupts\r");
}

```

```

    lcd.fontw( 1, "sans24" );
    lcd.theme( 1, 9, 3, 0, 0, 0, 8, 8, 8, 1, 1 );
    lcd.theme( 2, 5, 20, 3, 3, 3, 4, 4, 4, 2, 1 );
    lcd.string( 1, "ON" ); // stringId 1
    lcd.string( 2, "OFF" ); // stringId 2
    lcd.button( 1, x1Pos, yPos, width, height, option, alignment, radius, 1, 1 );
    lcd.button( 2, x2Pos, yPos, width, height, option, alignment, radius, 2, 2 );

    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, LOW );

    attachInterrupt(0, ezLCDevent, LOW);
}

void loop()
{
    if( ezLCDInt )
    {
        Serial.print("$");
        ezLCDInt = false;
        if( lcd.isPressed(1) ) // if Button 1 is pressed:
            digitalWrite( LED_PIN, HIGH ); // turn LED on
        else if( lcd.isPressed(2) ) // if Button 2 was pressed:
            digitalWrite( LED_PIN, LOW ); // turn LED off
    }
}

void ezLCDevent( void ) {
    ezLCDInt = true;
}

```

6.5 checkbox.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

int xPos = 30; // horizontal position for widget
int yPos = 30; // vertical position for widget
int width = 225;
int height = 50;
int option = 1; // 1=draw unchecked, 2=disabled, 3=draw checked, 4=redraw

void setup()
{
    Serial.begin(9600);
    lcd.begin( EZM_BAUD_RATE );
    lcd.fontw( 1, "sans24" );
    lcd.cls(BLACK);
    lcd.string( 1, "Flash LED Faster" ); // stringId 1
    lcd.checkbox( 1, xPos, yPos, width, height, option, 1, 1 );

    pinMode(LED_BUILTIN, OUTPUT );
    digitalWrite( LED_BUILTIN, LOW );
}

int rate = 500; // blink delay

void loop()
{
    if( lcd.isChecked(1) ) // if checkbox 1 is checked
        rate = 200; // reduce delay
    else // if checkbox 1 is unchecked
        rate = 500;
    blink(rate);
}

void blink(int rate)
{
    digitalWrite( LED_BUILTIN, HIGH ); // turn LED on
    delay(rate);
    digitalWrite( LED_BUILTIN, LOW ); // turn LED off
    delay(rate);
}

```

6.6 choice.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  Serial.begin(9600);
  lcd.begin( EZM_BAUD_RATE );
  lcd.fontw( 0, "sans24" );
  lcd.theme( 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 0 );

  pinMode( LED_PIN, OUTPUT );
  digitalWrite( LED_PIN, LOW );
}

void loop()
{
  lcd.cls(BLACK);

  int result = lcd.choice( "\"Got Milk\"", 0 );
  Serial.println(result);
  if(result == 1 )
    digitalWrite( LED_PIN, HIGH ); // turn LED on
  else if(result == 0 )
    digitalWrite( LED_PIN, HIGH ); // turn LED on

  delay(2000);
}
```

6.7 circle.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
  int x = 160;
  int y = 120;
  int size = 20;
  lcd.xy(x,y);
  for(int i=0; i < 100; i++)
  {
    lcd.color(i);
    lcd.circle( size );
    size += 4;
    delay(100);
  }
}

void loop(){}
```

6.8 cls.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls(); // clear screen to black
  lcd.rect(0,0,100,100);
  delay(1000);
  lcd.cls(RED); // clear screen to red
}
```



```
void loop(){ }
```

6.9 color.ino

```
/*
 * color.ino displays pre-defined colors
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
  lcd.fill(true);
  for(int i=0; i < 16; i++)
  {
    lcd.color(i);
    lcd.rect(i*16,0, 16, 100);
  }
  for(int i=16; i < 168; i++)
  {
    lcd.color(i);
    int pos = (i-16);
    lcd.rect(pos*2,120, 2, 100);
  }
}

void loop(){}
```

6.10 colorId.ino

```
/*
 * colorId.ino assigns a color to an id
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
  lcd.colorId(168, 222, 126, 93); // clear the LCD
  lcd.color(168); // peach
  lcd.fill(true); // set color 168
  lcd.rect(0,0,200,200); // draw filled rectangle
}

void loop(){}
```

6.11 dial.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );

int xPos = 100; // horizontal position
int yPos = 85; // vertical position
int radius = 75;
int option = 1; // 1=draw, 2=disabled.
```

```

int resolution = 25;
int value = 250;
int max = 500;
int id = 1;

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.fontw( 1, "sans24" );
  lcd.theme( 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 0 );
  lcd.cls(BLACK);
  lcd.color(WHITE);
  lcd.dial( id, xPos, yPos, radius, option, resolution, value, max, 0 );

  pinMode( LED_PIN, OUTPUT );
  digitalWrite( LED_PIN, LOW );
}

void loop()
{
  int value = lcd.getWidgetValue(id);
  blink(value);
}

void blink(int rate)
{
  digitalWrite( LED_BUILTIN, HIGH ); // turn LED on
  delay(rate);
  digitalWrite( LED_BUILTIN, LOW ); // turn LED off
  delay(rate);
}

```

6.12 digitalMeter.ino

```

/*
 * digitalMeter.ino displays analog meter
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

int xPos = 50; // horizontal position
int yPos = 50; // vertical position
int width = 100;
int height = 30;
int option = 14; // 1=draw, 2=disabled, 3=ring, 4=accuracy
                // 1=left, 2=disabled, 3=right, 4=center, 11=left framed, 12=disable framed,
                // 13=right framed, 14=center framed, 6=redraw.

int digits = 3;
int dp = 2;

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.font( 0 );
  lcd.theme( 1, 155, 152, 3, 130, 0, 0, 1, 147, 153, 1 );
  lcd.cls( WHITE );
  lcd.color(BLACK);
  lcd.xy(30,85);
  lcd.print("ALog pin 0 volts");
  lcd.digitalMeter( 1, xPos,yPos, width, height, option, 0, digits, dp, 1);
}

void loop()
{
  float value = analogRead(0);
  float volts = (5.00 * value) / 1023.0;
  lcd.wvalue(1, value);
  delay(2000); //wait two seconds before updating
}

```

6.13 ellipse.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
  int x=50;
  int y=50;
  int size = 100;
  for(int i=0; i < 100; i++)
  {
    lcd.color(i);
    lcd.ellipse(x, y, size );
    x = x + 2;
    y = x;
    size += 4;
    delay(100);
  }
}

void loop(){}

```

6.14 font.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

const int nbrFonts = 10;
char *fonts[] = {
  "kin", "sans","serif", "lcd", "blip", "core", "mac", "neur","olde", "squine"};

int nbrSizes[] = {4,4,4,4,2,2,2,2,2,2 } ;
char *sizes[] = { "72","48","36","24" };

char fontName[16];

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls(WHITE); // clear screen to white
  lcd.color(BLACK);
}

void loop()
{
  // show the two predefined fonts:
  lcd.font(0);
  lcd.println("This is font 0");
  lcd.println("ABCDEFGHGIJKlmnopqrst0123456789");
  lcd.println();
  lcd.font(1);
  lcd.println("This is font 1");
  lcd.println("ABCDEFGHGIJKlmnopqrst0123456789");
  delay(4000);
  lcd.cls(WHITE);

  // show the programmable (ezf file) fonts
  for(int i=0; i < nbrFonts; i++)
  {
    for(int size = 0; size < nbrSizes[i]; size++ )
    {
      strcpy(fontName, fonts[i]); // copy the base name
      strcat(fontName, sizes[size]); // append the size
      lcd.font(fontName);
      lcd.println(fontName);
    }
    delay(1000);
    lcd.cls(WHITE);
  }
}

```

6.15 groupbox.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );

int xPos = 35; // horizontal position
int yPos = 50; // vertical position
int width = 250;
int height = 120;
int option = 4; // 1=left,2=disabled,3=right,4=center.

void setup()
{
    Serial.begin(9600);
    lcd.begin( EZM_BAUD_RATE );

    lcd.string( 1, "Left Align");
    lcd.string( 2, "Disabled");
    lcd.string( 3, "Right Align");
    lcd.string( 4, "Center Align");

    lcd.fontw( 1, "sans24" );
    lcd.theme( 1, 155, 152, 3, 3, 1, 0, 1, 0, 0, 0 );
    lcd.cls( 0 );

    lcd.groupBox(1, xPos, yPos, width, height, option, 1, 1 );
}

void loop()
{
}
}
```

6.16 image.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls();
    lcd.image("logo200.gif",10,10);
    delay(2000);
    lcd.cls();
    lcd.image("logo200.gif",10,10,1);
}

void loop(){}
}
```

6.17 light.ino

```
/*
 * light.ino displays backlight control
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls(WHITE); // clear screen to white
    lcd.color(BLACK);
    lcd.font(1);
}
}
```

```

void loop()
{
    lcd.println("Backlight in steps of 10%");
    // set level in steps of 10%
    for(int i=0; i <= 100; i += 10)
    {
        lcd.light(i); // set the backlight level
        lcd.print("Backlight set to ");
        lcd.println(i);
        delay(2000);
    }
    lcd.cls(WHITE); // clear screen to white
}

```

6.18 line.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls();
    lcd.color(RED);
    for(int i=0; i < 100; i++)
    {
        int x = random(0,319);
        int y = random(0,239);
        lcd.line(x,y); //draw line from the previous xy location
        delay(100);
    }
    lcd.cls();
}

void loop()
{
    int color = random(0,168);
    lcd.color(color);
    //draw line specifying all coordinates
    int x = random(0,200);
    int y = random(0,100);
    int length = random(20, 100);
    lcd.line(x,y, x+length, y+length);
}

```

6.19 lineType.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls();
    lcd.color(WHITE);
    lcd.lineWidth(1);
    for( int type=0; type < 3; type++)
    {
        lcd.lineType(type);
        for(int i=0; i < 10; i++)
        {
            int x = random(0,319);
            int y = random(0,239);
            lcd.line(x,y); //draw line from the previous xy location
            delay(100);
        }
    }
}

void loop(){
}

```

6.20 lineWidth.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
  lcd.color(GREEN);
  lcd.lineWidth(1);
  for(int i=0; i < 30; i++)
  {
    int x = random(0,319);
    int y = random(0,239);
    lcd.line(x,y); //draw line from the previous xy location
    delay(100);
  }
  lcd.lineWidth(3);
  for(int i=0; i < 30; i++)
  {
    int x = random(0,319);
    int y = random(0,239);
    lcd.line(x,y); //draw line from the previous xy location
    delay(100);
  }
}

void loop(){ }
```

6.21 pie.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.color(WHITE);
}
int angle = 10;

void loop()
{
  lcd.cls();
  lcd.xy(160,120);
  lcd.pie(50, 0,angle);
  delay(1000);
  angle = angle + 30;
  if(angle > 360)
    angle = 10;
  lcd.cls();
}
```

6.22 plot.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
}
```

```
void loop()
{
  int color = random(0,168);
  lcd.color(color);
  int x = random(0,319);
  int y = random(0,239);
  lcd.plot(x,y);
}
```

6.23 point.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls();
}

void loop()
{
  int color = random(0,168);
  lcd.color(color);
  int x = random(0,319);
  int y = random(0,239);
  lcd.point(x,y);
}
```

6.24 printExample.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.cls(BLACK);
  lcd.color(WHITE);
  lcd.font( 0 );
  lcd.println("hello, world!");
  lcd.write(65);
  lcd.println();
  lcd.println(65);
  lcd.println(65,DEC);
  lcd.println(65,HEX);
  lcd.println(65,OCT);
  lcd.println(65,BIN);
  lcd.println(3.14);
  lcd.println();
}

void loop() {}
```

6.25 progress.ino

```
#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

int xPos = 25; // horizontal position
int yPos = 50; // vertical position
int width = 250;
int height = 35;
int option = 1; // 1=draw horizontal, 2=horizontal disabled, 3=vertical,
```

```

        // 4=vertical disabled, 5=redraw horizontal,
        // 6=redraw horizontal disabled, 7=redraw vertical,
        // 8=redraw vertical disabled
int value = 0;
int max= 100;

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.fontw( 1, "0" );
  lcd.theme( 1, 155, 152, 3, 3, 1, 0, 1, 0, 0, 0 );
  lcd.cls(BLACK);
  lcd.color(WHITE);
  lcd.string(1,"%"); //set string 1 to % for progress bar
  lcd.progressBar( 1, xPos, yPos, width, height, option, value,max, 1 ,1);
}

void loop()
{
  value = value + 10;
  if(value <= 100)
    lcd.wvalue(1, value);
  delay(500);
}

```

6.26 radio.ino

```

/*
 * radio.ino
 * radio buttons select the blink rate of an LED
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );
volatile boolean ezLCDInt = false;

int xPos = 25; // horizontal position
int yPos = 50; // vertical position
int width = 250;
int height = 35;
int option = 5; // 0=remove, 1=draw, 2=disabled, 3=checked,
// 4=draw first unchecked, 5=draw first checked

void setup()
{
  Serial.begin(9600);
  lcd.begin( EZM_BAUD_RATE );
  lcd.fontw( 1, "sans24" );
  lcd.theme( 1, 9, 3, 0, 0, 0, 8, 8, 8, 1, 1 );
  lcd.cls(BLACK);
  lcd.color(WHITE);
  lcd.xy(20,30);
  lcd.string( 1, "Off" ); // stringId 1
  lcd.string( 2, "Slow" ); // stringId 2
  lcd.string( 3, "Fast" ); // stringId 3
  lcd.radioButton( 1, xPos, yPos, width, height, 5, 1,1 );
  lcd.radioButton( 2, xPos, yPos + 50, width, height, 1, 1, 2 );
  lcd.radioButton( 3, xPos, yPos + 100, width, height, 1, 1, 3 );

  pinMode( LED_PIN, OUTPUT );
  digitalWrite( LED_PIN, LOW );
  Serial.println("ready");
}

int rate = 0; // blink delay, 0 is off

int selected = 0;
int prevSelected = 0;

void loop()
{
  for(int i=1; i <= 3; i++)
  {
    boolean checked = lcd.isChecked(i);
    if(checked) // if radioButton 1 is checked
      selected = i; // store the selected widget
  }
}

```



```

    }

    if(selected != prevSelected)
    {
        Serial.println(selected);
        prevSelected = selected;
        if(selected == 1)
            rate = 0; // LED is off
        else if(selected == 2)
            rate = 500; // LED blinks slow
        else if(selected == 3)
            rate = 200; // LED blinks fast
    }
    blink();
}

void blink()
{
    digitalWrite( LED_PIN, HIGH ); // turn LED on
    delay(rate);
    digitalWrite( LED_PIN, LOW ); // turn LED off
    delay(rate);
}

```

6.27 radio_interrupt.ino

```

/*
 * radio_interrupt.ino
 * radio buttons select the blink rate of an LED
 * interrupts are used to indicate button presses
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );
volatile boolean ezLCDInt = false;

int xPos = 25; // horizontal position
int yPos = 50; // vertical position
int width = 250;
int height = 35;
int option = 5; // 0=remove, 1=draw, 2=disabled, 3=checked,
// 4=draw first unchecked, 5=draw first checked

void setup()
{
    Serial.begin(9600);
    lcd.begin( EZM_BAUD_RATE );
    lcd.fontw( 1, "sans24" );
    lcd.theme( 1, 9, 3, 0, 0, 0, 8, 8, 8, 1, 1 );
    lcd.cls(BLACK);
    lcd.color(WHITE);
    lcd.xy(20,30);
    lcd.string( 1, "STOP" ); // stringId 1
    lcd.string( 2, "Slow" ); // stringId 2
    lcd.string( 3, "Fast" ); // stringId 3
    lcd.radioButton( 1, xPos, yPos, width, height, 5, 1, 1 );
    lcd.radioButton( 2, xPos, yPos + 50, width, height, 1, 1, 2 );
    lcd.radioButton( 3, xPos, yPos + 100, width, height, 1, 1, 3 );

    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, LOW );
    attachInterrupt(0, ezLCDhandler, LOW);
}

int rate = 0; // blink delay, 0 stops blinking

int selected = 0;
int prevSelected = 0;

void loop()
{
    if(ezLCDInt)
    {
        ezLCDInt = false;
        digitalWrite( LED_PIN, LOW ); // LED off
    }
}

```

```

    if( lcd.isChecked(1) ) {      // if radioButton 1 is checked
        Serial.println("1 checked");
        rate = 0;  // stop
    }
    else if( lcd.isChecked(2) ) {  // if radioButton 2 checked
        Serial.println("2 checked");
        rate = 500;  // slow
    }
    else if( lcd.isChecked(3) ) {  // if radioButton 3 is checked
        Serial.println("3 checked");
        rate = 200;  // fast
    }
}
blink();
}

void blink()
{
    digitalWrite( LED_PIN, HIGH ); // turn LED on
    delay(rate);
    digitalWrite( LED_PIN, LOW );  // turn LED off
    delay(rate);
}

void ezLCDhandler( void )
{
    ezLCDInt = true;
}

```

6.28 rect.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd(10, 11); // create lcd object using pins 10 & 11

void setup()
{
    lcd.begin( EZM_BAUD_RATE );
    lcd.cls();
    int x=2;
    int y=2;
    int width = 300;
    int height = 200;
    for(int i=0; i < 100; i++)
    {
        lcd.color(i);
        lcd.rect(x,y,width, height ); //draw line from the previous xy location
        x = x + 2;
        y = y + 2;
        width = width -4;
        height = height -4;
        delay(100);
    }
}

void loop(){}

```

6.29 slider.ino

```

#include <ezLCD.h>
#include <SoftwareSerial.h>

#define LED_PIN 13

EzLCD3_SW lcd( 10, 11 );

int xPos = 25; // horizontal position
int yPos = 50; // vertical position
int width = 250;
int height = 35;
int option = 5; // 1=draw horizontal, 2=horizontal disabled, 3=vertical,
                // 4=vertical disabled, 5=horizontal slider,
                // 6=horizontal slider disabled, 7=vertical slider,

```

```

// 8=vertical slider disabled
int max= 500;
int resolution = 5;
int value = 200;

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.fontw( 1, "sans24" );
  lcd.theme( 1, 9, 3, 0, 0, 0, 8, 8, 8, 1, 1 );
  lcd.cls(BLACK);
  lcd.color(WHITE);
  lcd.slider( 1, xPos, yPos, width, height, option, max, resolution, value,1 );

  pinMode( LED_PIN, OUTPUT );
  digitalWrite( LED_PIN, LOW );
}

void loop()
{
  int rate = lcd.getWidgetValue(1);
  blink(rate);
}

void blink(int rate)
{
  digitalWrite( LED_BUILTIN, HIGH ); // turn LED on
  delay(rate);
  digitalWrite( LED_BUILTIN, LOW ); // turn LED off
  delay(rate);
}

```

6.30 theme.ino

```

/*
 * theme.ino example showing the use of themes
 * TODO this example is not complete
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

// theme parameters
int embossDkColor    = GREEN;
int embossLtColor    = WHITE;
int textColor0       = BLACK;
int textColor1       = BLACK;
int textColorDisabled = BLACK;
int color0           = LIME;
int color1           = LIME;
int colorDisabled    = LIME;
int commonBkColor    = GRAY;

int x1Pos = 10; // horizontal position for buttton 1
int x2Pos = 110; // horizontal position for buttton 2
int yPos = 40; // vertical position of both buttons
int width = 100;
int height = 100;
int radius = 0;
int alignment = 0; // 0=centered, 1=right, 2=left, 3=bottom, 4=top
int option = 1; // 1=draw, 2=disabled, 3=toggle pressed, 4=toggle not pressed,
// 5=toggle pressed disabled, 6=toggle not pressed disabled.

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.fontw( 1, "sans24" );
  lcd.theme( 1, 9, 3, 0, 0, 0, 8, 8, 8, 1, 1 );
  lcd.cls( 0 );
  lcd.string( 1, "ON" ); // stringId 1
  lcd.string( 2, "OFF" ); // stringId 2

  lcd.theme( 1, 9, 12, 0, 0, 0, 8, 8, 8, 1, 1 ); // text to blue
  lcd.theme( 1, 9, 4, 0, 0, 0, 8, 8, 8, 1, 1 ); // text to red

  lcd.button( 1, x1Pos, yPos, width, height, option, alignment, radius, 1, 1 );
  lcd.button( 2, x2Pos, yPos, width, height, option, alignment, radius, 2, 2 );
}

```

```
void loop()
{
}
```

6.31 touch.ino

```
/*
 * touch.ino example for touch functions
 */

#include <ezLCD.h>
#include <SoftwareSerial.h>

EzLCD3_SW lcd( 10, 11 );

void setup()
{
  lcd.begin( EZM_BAUD_RATE );
  lcd.font(0);
  lcd.cls(WHITE); // clear screen to white
  lcd.color(BLACK);
  lcd.println("Touch the Screen");
}

void loop()
{
  if(lcd.touchS() == 1) // if is touched
  {
    lcd.cls(WHITE); // clear screen to white
    int x = lcd.touchX();
    if(x > 0){
      lcd.print("X touch = ");
      lcd.println(x);
    }
    int y = lcd.touchY();
    if(y > 0){
      lcd.print("Y touch = ");
      lcd.println(y);
    }
    lcd.xy(x,y);
    lcd.circle(25);
  }
}
```

Index

~EzLCD3
EzLCD3, 12

AMeter_Color
EzLCD3, 11

AMeter_Value
EzLCD3, 11

AQUA
ezLCD.h, 36

analogMeter
EzLCD3, 12

analogMeterColor
EzLCD3, 12

Arc
EzLCD3, 10

arc
EzLCD3, 13

BLACK
ezLCD.h, 35

BLUE
ezLCD.h, 36

```
Beep_Freq
EzLCD3, 10
```

```
begin
    EzLCD3, 13
    EzLCD3_HW, 30
    EzLCD3_SW, 31
```

Box
EzLCD3, 10

box
EzLCD3, 13

button
EzLCD3, 14

C:/Users/Segler/arduino-1.0.4/libraries/arLCD/examples.
lst, 33

C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.
cpp, 33

C:/Users/Segler/arduino-1.0.4/libraries/arLCD/ezLCD.h,
33

CLEAR
ezLCD.h, 34

Calibrate
EzLCD3, 10

calibrate
EzLCD3, 14

checkbox
EzLCD3, 14

Choice

EzLCD3, 11

choice
EzLCD3, 14

circle
EzLCD3, 15

ClipArea
EzLCD3, 11

ClipEnable
EzLCD3, 11

cliparea
EzLCD3, 15

clipenable
EzLCD3, 15

Clr_Screen
EzLCD3, 10

cls
EzLCD3, 15

Cmd
EzLCD3, 11

Color
EzLCD3, 10

color
EzLCD3, 15, 16

colorId
EzLCD3, 16

Command
EzLCD3, 10

Commands
EzLCD3, 10

Comment
EzLCD3, 11

Comport
EzLCD3, 11

ConfigIO
EzLCD3, 11

DMeter_Value
EzLCD3, 11

DEFAULT_TIMEOUT
ezLCD.cpp, 33

```
debugWrite
    EzLCD3, 16
```

dial
EzLCD3, 16

digitalMeter
EzLCD3, 17

eColor_ID
EzLCD3, 10

eLine

- EzLCD3, [10](#)
- eXY
 - EzLCD3, [10](#)
- EZLCD_ENUM_CHOICE
 - ezLCD.h, [34](#)
- EZLCD_ENUM_WIDGETS
 - ezLCD.h, [35](#)
- EZLCD_GLOBALENUMS
 - ezLCD.h, [35](#)
- EZLCD_SERIALDEBUG
 - ezLCD.h, [35](#)
- EZM_BAUD_RATE
 - ezLCD.h, [35](#)
- echo
 - EzLCD3, [17](#)
- ellipse
 - EzLCD3, [17](#)
- ezLCD.h
 - AQUA, [36](#)
 - BLACK, [35](#)
 - BLUE, [36](#)
 - FUCHSIA, [36](#)
 - GRAY, [35](#)
 - GREEN, [36](#)
 - LIME, [36](#)
 - MAROON, [35](#)
 - NAVY, [36](#)
 - OLIVE, [36](#)
 - PURPLE, [36](#)
 - RED, [35](#)
 - SILVER, [35](#)
 - TEAL, [36](#)
 - WHITE, [35](#)
 - YELLOW, [35](#)
- EzLCD3
 - AMeter_Color, [11](#)
 - AMeter_Value, [11](#)
 - Arc, [10](#)
 - Beep_Freq, [10](#)
 - Box, [10](#)
 - Calibrate, [10](#)
 - Choice, [11](#)
 - ClipArea, [11](#)
 - ClipEnable, [11](#)
 - Clr_Screen, [10](#)
 - Cmd, [11](#)
 - Color, [10](#)
 - Command, [10](#)
 - Comment, [11](#)
 - Comport, [11](#)
 - ConfigIO, [11](#)
 - DMeter_Value, [11](#)
 - eColor_ID, [10](#)
 - eLine, [10](#)
 - eXY, [10](#)
 - Font, [10](#)
 - Font_Orient, [10](#)
 - Fontw, [10](#)
 - Format, [11](#)
 - Fschdir, [11](#)
 - Fscopy, [11](#)
 - Fsdir, [11](#)
 - Fsgetcwd, [11](#)
 - Fsmkdir, [11](#)
 - Fsmore, [11](#)
 - Fsremove, [11](#)
 - Fsrename, [11](#)
 - Fsrmdir, [11](#)
 - Get_Pixel, [10](#)
 - IO, [11](#)
 - IOG, [11](#)
 - If, [11](#)
 - Lecho, [12](#)
 - Light, [10](#)
 - Line_Type, [10](#)
 - Line_Width, [10](#)
 - Location, [11](#)
 - Loop_Macro, [10](#)
 - Mode, [11](#)
 - Parameters, [11](#)
 - Pause_Macro, [10](#)
 - Peri, [10](#)
 - Picture, [10](#)
 - Pie, [10](#)
 - Ping, [10](#)
 - Play_Macro, [10](#)
 - Plot, [10](#)
 - Print, [10](#)
 - Progress_Value, [11](#)
 - Rec_Macro, [10](#)
 - Security, [11](#)
 - Set_AMeter, [11](#)
 - Set_Button, [11](#)
 - Set_CheckBox, [11](#)
 - Set_DMeter, [11](#)
 - Set_Dial, [11](#)
 - Set_Gbox, [11](#)
 - Set_Progress, [11](#)
 - Set_RadioButton, [11](#)
 - Set_Slider, [11](#)
 - Set_StaticText, [11](#)
 - Set_TouchZone, [11](#)
 - Speed_Macro, [10](#)
 - StaticText_Value, [11](#)
 - Status, [10](#)
 - Stop_Macro, [10](#)
 - Stouch, [12](#)
 - StringID, [10](#)
 - Threshold, [12](#)
 - Upgrade, [11](#)
 - Verbose, [12](#)
 - Wait, [12](#)
 - Waitn, [12](#)
 - Waitt, [12](#)
 - Widget_State, [11](#)
 - Widget_Theme, [11](#)

- Widget_Values, 11
- Wquiet, 12
- Wstack, 12
- Xmax, 11
- Xtouch, 12
- Ymax, 12
- Ytouch, 12
- zBeep, 10
- zCircle, 10
- zReset, 10
- ezLCD.cpp
 - DEFAULT_TIMEOUT, 33
 - PARSE_TIMEOUT, 33
 - SEMPLEP, 33
 - sdebug, 33
 - sdebugln, 33
- ezLCD.h
 - CLEAR, 34
 - EZLCD_GLOBALENUMS, 35
 - EZLCD_SERIALDEBUG, 35
 - EZM_BAUD_RATE, 35
 - FIFO, 35
 - LIFO, 35
- EzLCD3, 7
 - ~EzLCD3, 12
 - analogMeter, 12
 - analogMeterColor, 12
 - arc, 13
 - begin, 13
 - box, 13
 - button, 14
 - calibrate, 14
 - checkbox, 14
 - choice, 14
 - circle, 15
 - cliparea, 15
 - clipenable, 15
 - cls, 15
 - color, 15, 16
 - colorId, 16
 - Commands, 10
 - debugWrite, 16
 - dial, 16
 - digitalMeter, 17
 - echo, 17
 - ellipse, 17
 - EzLCD3, 12
 - ezLCDUpgrade, 17
 - EzLCD3, 12
 - fill, 17
 - findEzLCD, 17
 - font, 18
 - fonto, 18
 - fontw, 18
 - getPixel, 18
 - getWidgetValue, 18
 - groupBox, 19
 - height, 19
 - image, 19
 - isChecked, 20
 - isHWSerial, 20
 - isPressed, 20
 - light, 20, 21
 - line, 21
 - lineType, 21
 - lineType, 22
 - lineWidth, 22
 - m_pStream, 29
 - parseHex, 22
 - picture, 22
 - pie, 23
 - plot, 23
 - point, 23
 - printAligned, 23
 - printString, 23
 - printStringId, 24
 - progressBar, 24
 - radioButton, 24
 - rect, 24
 - reset, 25
 - slider, 25
 - staticText, 25
 - string, 25
 - theme, 26
 - touchS, 26
 - touchX, 26
 - touchY, 26
 - touchZone, 26
 - waitNoTouch, 27
 - waitTouch, 27
 - width, 27
 - wquiet, 27
 - write, 27
 - wstack, 27
 - wstate, 27
 - wvalue, 28
 - xmax, 28
 - xy, 28
 - xy_restore, 28
 - xy_store, 28
 - xyAligned, 28
 - xyGet, 29
 - ymax, 29
- EzLCD3_HW, 29
 - begin, 30
 - EzLCD3_HW, 30
 - EzLCD3_HW, 30
- EzLCD3_SW, 30
 - begin, 31
 - EzLCD3_SW, 31
 - EzLCD3_SW, 31
- ezLCDUpgrade
 - EzLCD3, 17
- FUCHSIA
 - ezLCD.h, 36
- FIFO

- ezLCD.h, [35](#)
- fill
 - EzLCD3, [17](#)
- findEzLCD
 - EzLCD3, [17](#)
- Font
 - EzLCD3, [10](#)
- font
 - EzLCD3, [18](#)
- Font_Orient
 - EzLCD3, [10](#)
- fonto
 - EzLCD3, [18](#)
- Fontw
 - EzLCD3, [10](#)
- fontw
 - EzLCD3, [18](#)
- Format
 - EzLCD3, [11](#)
- Fschdir
 - EzLCD3, [11](#)
- Fscopy
 - EzLCD3, [11](#)
- Fsdir
 - EzLCD3, [11](#)
- Fsgetcwd
 - EzLCD3, [11](#)
- Fsmkdir
 - EzLCD3, [11](#)
- Fsmore
 - EzLCD3, [11](#)
- Fsremove
 - EzLCD3, [11](#)
- Fsrename
 - EzLCD3, [11](#)
- Fsrmkdir
 - EzLCD3, [11](#)
- GRAY
 - ezLCD.h, [35](#)
- GREEN
 - ezLCD.h, [36](#)
- Get_Pixel
 - EzLCD3, [10](#)
- getPixel
 - EzLCD3, [18](#)
- getWidgetValue
 - EzLCD3, [18](#)
- groupBox
 - EzLCD3, [19](#)
- height
 - EzLCD3, [19](#)
- IO
 - EzLCD3, [11](#)
- IOG
 - EzLCD3, [11](#)
- If
 - EzLCD3, [11](#)
- image
 - EzLCD3, [19](#)
- isChecked
 - EzLCD3, [20](#)
- isHWSerial
 - EzLCD3, [20](#)
- isPressed
 - EzLCD3, [20](#)
- LIME
 - ezLCD.h, [36](#)
- LIFO
 - ezLCD.h, [35](#)
- Lecho
 - EzLCD3, [12](#)
- Light
 - EzLCD3, [10](#)
- light
 - EzLCD3, [20](#), [21](#)
- line
 - EzLCD3, [21](#)
- Line_Type
 - EzLCD3, [10](#)
- Line_Width
 - EzLCD3, [10](#)
- lineTtype
 - EzLCD3, [21](#)
- lineType
 - EzLCD3, [22](#)
- lineWidth
 - EzLCD3, [22](#)
- Location
 - EzLCD3, [11](#)
- Loop_Macro
 - EzLCD3, [10](#)
- MAROON
 - ezLCD.h, [35](#)
- m_pStream
 - EzLCD3, [29](#)
- Mode
 - EzLCD3, [11](#)
- NAVY
 - ezLCD.h, [36](#)
- OLIVE
 - ezLCD.h, [36](#)
- PURPLE
 - ezLCD.h, [36](#)
- PARSE_TIMEOUT
 - ezLCD.cpp, [33](#)
- Parameters
 - EzLCD3, [11](#)
- parseHex
 - EzLCD3, [22](#)
- Pause_Macro

- EzLCD3, [10](#)
- Peri
 - EzLCD3, [10](#)
- Picture
 - EzLCD3, [10](#)
- picture
 - EzLCD3, [22](#)
- Pie
 - EzLCD3, [10](#)
- pie
 - EzLCD3, [23](#)
- Ping
 - EzLCD3, [10](#)
- Play_Macro
 - EzLCD3, [10](#)
- Plot
 - EzLCD3, [10](#)
- plot
 - EzLCD3, [23](#)
- point
 - EzLCD3, [23](#)
- Print
 - EzLCD3, [10](#)
- printAligned
 - EzLCD3, [23](#)
- printString
 - EzLCD3, [23](#)
- printStringId
 - EzLCD3, [24](#)
- Progress_Value
 - EzLCD3, [11](#)
- progressBar
 - EzLCD3, [24](#)
- RED
 - ezLCD.h, [35](#)
- radioButton
 - EzLCD3, [24](#)
- Rec_Macro
 - EzLCD3, [10](#)
- rect
 - EzLCD3, [24](#)
- reset
 - EzLCD3, [25](#)
- SILVER
 - ezLCD.h, [35](#)
- SEMPLE_SEP
 - ezLCD.cpp, [33](#)
- sdebug
 - ezLCD.cpp, [33](#)
- sdebugIn
 - ezLCD.cpp, [33](#)
- Security
 - EzLCD3, [11](#)
- Set_AMeter
 - EzLCD3, [11](#)
- Set_Button
 - EzLCD3, [11](#)
- Set_CheckBox
 - EzLCD3, [11](#)
- Set_DMeter
 - EzLCD3, [11](#)
- Set_Dial
 - EzLCD3, [11](#)
- Set_Gbox
 - EzLCD3, [11](#)
- Set_Progress
 - EzLCD3, [11](#)
- Set_RadioButton
 - EzLCD3, [11](#)
- Set_Slider
 - EzLCD3, [11](#)
- Set_StaticText
 - EzLCD3, [11](#)
- Set_TouchZone
 - EzLCD3, [11](#)
- slider
 - EzLCD3, [25](#)
- Speed_Macro
 - EzLCD3, [10](#)
- StaticText_Value
 - EzLCD3, [11](#)
- staticText
 - EzLCD3, [25](#)
- Status
 - EzLCD3, [10](#)
- Stop_Macro
 - EzLCD3, [10](#)
- Stouch
 - EzLCD3, [12](#)
- string
 - EzLCD3, [25](#)
- StringID
 - EzLCD3, [10](#)
- TEAL
 - ezLCD.h, [36](#)
- theme
 - EzLCD3, [26](#)
- Threshold
 - EzLCD3, [12](#)
- touchS
 - EzLCD3, [26](#)
- touchX
 - EzLCD3, [26](#)
- touchY
 - EzLCD3, [26](#)
- touchZone
 - EzLCD3, [26](#)
- Upgrade
 - EzLCD3, [11](#)
- Verbose
 - EzLCD3, [12](#)
- WHITE

- ezLCD.h, [35](#)
- Wait
 - EzLCD3, [12](#)
- waitNoTouch
 - EzLCD3, [27](#)
- waitTouch
 - EzLCD3, [27](#)
- Waitn
 - EzLCD3, [12](#)
- Waitt
 - EzLCD3, [12](#)
- Widget_State
 - EzLCD3, [11](#)
- Widget_Theme
 - EzLCD3, [11](#)
- Widget_Values
 - EzLCD3, [11](#)
- width
 - EzLCD3, [27](#)
- Wquiet
 - EzLCD3, [12](#)
- wquiet
 - EzLCD3, [27](#)
- write
 - EzLCD3, [27](#)
- Wstack
 - EzLCD3, [12](#)
- wstack
 - EzLCD3, [27](#)
- wstate
 - EzLCD3, [27](#)
- wvalue
 - EzLCD3, [28](#)
- Xmax
 - EzLCD3, [11](#)
- xmax
 - EzLCD3, [28](#)
- Xtouch
 - EzLCD3, [12](#)
- xy
 - EzLCD3, [28](#)
- xy_restore
 - EzLCD3, [28](#)
- xy_store
 - EzLCD3, [28](#)
- xyAligned
 - EzLCD3, [28](#)
- xyGet
 - EzLCD3, [29](#)
- YELLOW
 - ezLCD.h, [35](#)
- Ymax
 - EzLCD3, [12](#)
- ymax
 - EzLCD3, [29](#)
- Ytouch
 - EzLCD3, [12](#)
- zBeep
 - EzLCD3, [10](#)
- zCircle
 - EzLCD3, [10](#)
- zReset
 - EzLCD3, [10](#)