arLCD Arduino Library

1.01

Generated by Doxygen 1.8.3.1

Fri Apr 19 2013 10:39:56

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

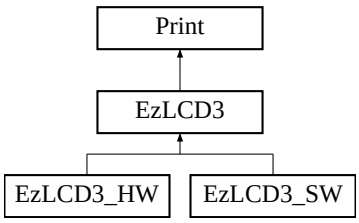
Chapter 4

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 lineTtype ()
- 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 EarthSEMPL commands. Provided here for users who wish to compose EarthSEMPL 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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*strid*	String ID to use for text.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*initial*	Initial numeric value of the meter.
in	*min*	Minimum reading of the meter.
in	*max*	Maximum reading of the meter.
in	*option*	Option.
in	*type*	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	*id*	Widget ID.
in	*color1*	color1
in	*color2*	color2
in	*color3*	color3
in	*color4*	color4
in	*color5*	color5
in	*color6*	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	*radius*	Arc radius in pixels.
in	*start*	Start angle in degrees.
in	*end*	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	*x*	X
in	*y*	Y
in	*diameter*	diameter of arc
in	*start*	Start angle in degrees.
in	*stop*	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	*baud*	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	*width*	Box width in pixels.
in	*height*	Box height in pixels.
in	*fill*	Set to true to fill the solid with box. Default is false 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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*strid*	String ID to use for text.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*radius*	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	*align*	Text alignment. Allowed values are `CENTER` (default), `LEFT`, `RIGHT` or `BOTTOM`
in	*option*	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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*strid*	String ID to use for text.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*option*	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	*str*	String to display in the prompt
in	*theme*	Theme to use for drawing the prompt.
in	*timeout*	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	*radius*	Circle radius in pixels.
in	*fill*	Set to true to fill the circle solid with color. Default is false 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	*left*	Left edge of the clip area in pixels.
in	*top*	Top edge of the clip area in pixels.
in	*right*	Right edge of the clip area in pixels.
in	*bottom*	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	*enable*	true to enable, false to disable.

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

Clear screen to a specific color id and clear widgets.

Parameters

in	*id*	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	*color*	Null-terminated string describing color.

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

Sets the current color by color ID.

Parameters

in	*id*	Numeric color ID.

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

Return current color.

Parameters

out	*red*	Red value in 0-255.
out	*green*	Green value in 0-255.
out	*blue*	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	*id*	Numeric color ID.
in	*red*	Red value in 0-255.
in	*green*	Green value in 0-255.
in	*blue*	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	*id*	Numeric color ID.
out	*red*	Red value in 0-255.
out	*green*	Green value in 0-255.
out	*blue*	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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*radius*	Radius of the dial in pixels.
in	*resolution*	Resolution of the dial in degrees.
in	*initial*	Initial numeric position of the dial.
in	*max*	Maximum numeric position of the dial.
in	*option*	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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*initial*	Initial numeric value of the meter.
in	*digits*	Total number of digits.
in	*dotpos*	Dot position counting digits from the left.
in	*option*	Option

4.1.4.23 bool EzLCD3::echo ( ) [inline]

Returns whether the class was configured for echo

Return values

true	The class is configured for echo.
false	The class is configured for NO echo

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

Enable or disable echo

Parameters

in	*val*	`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	*isFilled*	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

in	*id*	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

in	*fontname*	Font filename on the ezLCD filesystem.

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

Set current font orientation to horizontal or vertical

Parameters

in	*orientation*	HORIZONTAL for horizontal orientation. VERTICAL for vertical orientation.

4.1.4.33 int EzLCD3::fonto ( )

Returns current font orientation.

Return values

HORIZONTAL	Horizontal orientation.
VERTICAL	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

in	*id*	Font ID.
in	*fontname*	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::getWidgetValue (int *id*)

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

Parameters

in	*id*	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	*id*	Widget ID to assign.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*option*	Option
in	*theme*	Theme ID to use.
in	*strid*	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	*id*	ID of the image to display.
in	*x*	x-coordinate in pixels of where to draw.
in	*y*	y-coordinate in pixels of where to draw.
in	*option*	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	*filename*	Filename of the image to display. Must include extension.
in	*x*	x-coordinate in pixels of where to draw.
in	*y*	y-coordinate in pixels of where to draw.
in	*option*	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	*id*	Widget ID of the checkbox you want to check.

Return values

true	ezLCD signaled that the checkbox was checked.
false	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

true	`HardwareSerial` class is being used.
false	`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	*id*	Widget ID of the button you want to check.

Return values

true	ezLCD signaled that the button is pressed.
false	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	*brightness*	Brightness in 0-100.

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

Set brightness & timeout

Parameters

in	*brightness*	Brightness in 0-100.
in	*timeout*	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	*brightness*	Brightness in 0-100.
in	*timeout*	Timeout value in minutes before dimming
in	*dimmed*	Dimmed brighness 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	*x*	x-coordinate in pixels to draw the line to.
in	*y*	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	*x1*	x1-coordinate in pixels
in	*y1*	y1-coordinate in pixels
in	*x2*	x2-coordinate in pixels
in	*y2*	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

in	*type*	0 is solid, increasing number increases spacing between dots.

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

Set current line width

Parameters

in	*width*	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

the	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

in	*id*	ID of the picture to display.
in	*x*	x-coordinate in pixels of where to draw.
in	*y*	y-coordinate in pixels of where to draw.
in	*option*	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	*filename*	Filename of the picture to display. Must include extension.
in	*x*	x-coordinate in pixels of where to draw.
in	*y*	y-coordinate in pixels of where to draw.
in	*option*	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	*radius*	Arc radius in pixels.
in	*start*	Start angle in degrees.
in	*end*	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	*x*	x-coordinate in pixels.
in	*y*	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	*x*	x-coordinate in pixels
in	*y*	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	*str*	Null-terminated string to print.
in	*alignment*	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	*str*	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	*id*	String ID.
in	*alignment*	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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*initial*	Initial numeric reading of the progress bar. Default is 0.
in	*range*	Maximum allowed value of the progress bar. Default is 100.
in	*option*	Option
in	*suffix*	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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*strid*	String ID to use for text.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*option*	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	*x*	x-coordinate in pixels
in	*y*	y-coordinate in pixels
in	*width*	width of rect in pixels
in	*height*	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*)

Ceate/alter a slider widget.

Parameters

in	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*resolution*	per step
in	*initial*	Initial numeric value of the slider.
in	*max*	Maximum numeric value of the slider.
in	*option*	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	*id*	Widget ID to assign.
in	*theme*	Theme ID to use.
in	*strid*	String ID to use for text.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*option*	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	*id*	String ID at which to store.
in	*str*	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	*index*	Theme Index
in	*embossDkColor*	Dark Emboss color used for 3-D effect of objects.
in	*embossLtColor*	Light Emboss color used for 3-D effect of objects .
in	*textColor0*	For text, Useage may vary from one widget to another, whether the widget is in focus or not.
in	*textColor1*	For text when pressed. Useage may vary from one widget to another.
in	*textColor-Disabled*	Color of objects that are disabled.
in	*color0*	For objects. Usage may vary from one object to another, whether the widget is in focus or not.
in	*color1*	For objects when pressed. Useage may vary from one object to another.
in	*colorDisabled*	Used to render objects that are disabled.
in	*commonBkColor*	Used to hide objects from screen but still active.
in	*fontw*	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	*id*	Widget ID to assign.
in	*x*	Starting x-coordinate in pixels.
in	*y*	Starting y-coordinate in pixels.
in	*width*	Width in pixels.
in	*height*	Height in pixels.
in	*option*	Option.

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

Wait for release of touch.

Parameters

in	*timeout*	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

in	*timeout*	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

in	*cmd*	= FIFO will return in first in first out order
in	*cmd*	= LIFO will return in last in first out order
in	*cmd*	= CLEAR will clear the stack

Return values

value	one unsigned int off stack high byte will be widget ID low byte status `lcd.wstack(LIFO)` 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

in	*id*	Widget ID you want to check .

Return values

value	of widget

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

Set value for a widget.

Parameters

in	*id*	Widget ID.
in	*value*	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	*x*	x-coordinate in pixels.
in	*y*	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	*id*	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	*id*	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).

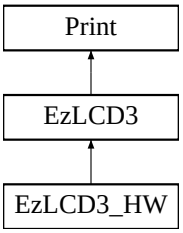
The documentation for this class was generated from the following files:

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

in	*baud*	Baud rate

Implements EzLCD3.

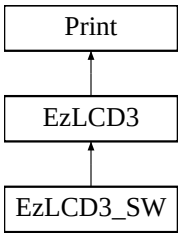
The documentation for this class was generated from the following files:

- 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

in	*rx*	Receive pin to be used.
in	*tx*	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

in	*baud*	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 SEMPL_SEP ' '

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 SEMPL_SEP ' '

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\n\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();                        // clear the LCD
  lcd.colorId(168, 222, 126, 93); // peach
  lcd.color(168);                   // set color 168
  lcd.fill(true);
  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("ABCDEFGHIJKlmnopqrst0123456789");
  lcd.println();
  lcd.font(1);
  lcd.println("This is font 1");
  lcd.println("ABCDEFGHIJKlmnopqrst0123456789");
  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