APO Light Control

Generated by Doxygen 1.8.14

Contents

Chapter 1

Apo Light Control

1.1 Introduction

This is the documentation for the project Apo Light Control by Klára and Edward. You can find the source code on `FEL Gitlab` (if you have the access rights, ofc ^-^).

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

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

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Module Documentation

6.1 Screens module

Screen base class and implementations.

Classes

- class ListScreen

  Unit screen.
- class Screen

  Class representing one screen.
- class UnitScreen

  Unit screen.

6.1.1 Detailed Description

Screen base class and implementations.

6.2 Utilities module

Utility functions.

Namespaces

- Colour

    Anything related to handling colours.
- IOTools

    Tools connected to IO.

6.2.1 Detailed Description

Utility functions.

6.3 Networking and threading module

Networking and threading.

Classes

- class ControlMessageQueue

  Thread-safe FIFO queue for control messages.
- class RWMutex

  Read/Write mutex class.
- class NetworkHandler

  Class that handles all network communication.

6.3.1 Detailed Description

Networking and threading.

6.4 Unit module

MZ board and light units.

Namespaces

- LedController

  Namespace that handles LEDs.

Classes

- class DeviceInput

  Handler device input.
- class Display

  A class that handles the (one-way) interaction with the device display and provides methods for rendering shapes, text, and other.
- class Mapper

  A class that handles the mapping of peripheral physical regions to virtual addresses.
- class LightUnit

  A class representing a light unit.

6.4.1 Detailed Description

MZ board and light units.

Chapter 7

Namespace Documentation

7.1 Colour Namespace Reference

Anything related to handling colours.

Enumerations

- enum {
  BLACK = 0, **WHITE** = 0xFFFF, **RED** = 0xF800, **ORANGE** = 0xFC00,
  YELLOW = 0xFF80, **LIME** = 0xB7E0, **GREEN** = 0x4FE0, **DARK_GREEN** = 0x4D24,
  BLUE = 0x051F, **PURPLE** = 0x881F, **BROWN** = 0x9260, **DARK_BLUE** = 0x08CB,
  TURUOISE = 0x6694, **UGLY** = 0xEC1A, **WEIRD_RED** = 0xA165, **DARK_GREY** = 0x2945,
  LIGHT_GREY = 0xD69A, **ALMOST_GOLD** = 0xFE03 }

Functions

- uint8_t getR (uint32_t rgb)

  Extracts the red component from RGB888.
- uint8_t getG (uint32_t rgb)

  Extracts the green component from RGB888.
- uint8_t getB (uint32_t rgb)

  Extracts the blue component from RGB888.
- uint32_t setR (uint32_t value, uint8_t newValue)

  Sets the red component of RGB888.
- uint32_t setG (uint32_t value, uint8_t newValue)

  Sets the green component of RGB888.
- uint32_t setB (uint32_t value, uint8_t newValue)

  Sets the blue component of RGB888.
- uint32_t changeR (uint32_t value, int16_t change)

  Changes the red component of RGB888.
- uint32_t changeG (uint32_t value, int16_t change)

  Changes the green component of RGB888.
- uint32_t changeB (uint32_t value, int16_t change)

  Changes the blue component of RGB888.
- uint32_t fromRGB (uint8_t r, uint8_t g, uint8_t b)

Creates an RGB888 colour from its separate components.

- std::string toRGBString (uint32_t rgb)

  Creates an rgb string representation of the colour.
- std::string toHexString (uint32_t rgb)

  Creates a hex string representation of the colour.
- uint16_t rgb888to565 (uint32_t rgb888)

  Converts an RGB888 colour to an RGB565 colour.
- uint32_t rgb565to888 (uint16_t rgb565)

  Converts an RGB565 colour to an RGB888 colour.

7.1.1 Detailed Description

Anything related to handling colours.

7.1.2 Function Documentation

7.1.2.1 changeB()

```
uint32_t Colour::changeB (
            uint32_t value,
            int16_t change )
```

Changes the blue component of RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

value	The RGB888 colour to be changed.
change	The change in the blue component.

7.1.2.2 changeG()

```
uint32_t Colour::changeG (
             uint32_t value,
             int16_t change )
```

Changes the green component of RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

value	The RGB888 colour to be changed.
change	The change in the green component.

7.1.2.3 changeR()

```
uint32_t Colour::changeR (
             uint32_t value,
             int16_t change )
```

Changes the red component of RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

value	The RGB888 colour to be changed.
change	The change of the red component.

Parameters

Parameters

Returns

```
uint8_t Colour::getB (
            uint32_t rgb )
```

Extracts the blue component from RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

rgb The rgb colour.

Returns

The blue component.

7.1.2.6 getG()

```
uint8_t Colour::getG (
            uint32_t rgb )
```

Extracts the green component from RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

rgb	The rgb colour.

Returns

The green component.

7.1.2.7 getR()

```
uint8_t Colour::getR (
            uint32_t rgb )
```

Extracts the red component from RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

rgb	The rgb colour.

Returns

The red component.

[redacted]

Converts an RGB888 colour to an RGB565 colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

rgb888	The RGB888 colour.

Returns

The resulting RGB565 colour.

See also

rgb565to888

7.1.2.10 setB()

```
uint32_t Colour::setB (
            uint32_t value,
            uint8_t newValue )
```

Sets the blue component of RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

value	The RGB888 colour to be changed.
newValue	The blue component.

7.1.2.11 setG()

```
uint32_t Colour::setG (
            uint32_t value,
            uint8_t newValue )
```

Sets the green component of RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

value	The RGB888 colour to be changed.
newValue	The green component.

7.1.2.12 setR()

```
uint32_t Colour::setR (
             uint32_t value,
             uint8_t newValue )
```

Sets the red component of RGB888.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

value	The RGB888 colour to be changed.
newValue	The red component.

7.1.2.13 toHexString()

```
std::string Colour::toHexString (
             uint32_t rgb )
```

Creates a hex string representation of the colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

rgb	The RGB888 colour.

Returns

The resulting string.

[redacted]

The entry point of the whole application.

Variables

- std::list< LightUnit > unitList

  The list of all currently connected units.
- ControlMessageQueue controlQueue

  The control message queue.
- std::unordered_map< std::string, std::array< uint16_t, 256 > > uiIcons

  The map of UI icons.

7.2.1 Detailed Description

Namesapce representing the entry point of the application.

It stores a list of connected units, a control message queue and ui icons map. Also internally runs and synchronizes the main loop and network loop and handles network messages.

7.2.2 Function Documentation

7.2.2.1 run()

```
int Engine::run (
            int argc,
            char ** argv )
```

The entry point of the whole application.

This function starts a network thread, connects the new unit to the network and loads ui icons. =1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

in	*argc*	The number of arguments.
in	*argv*	The command line arguments. The format should be '"Unit Description" path_to_icon16x16.ppm'.

7.3 IOTools Namespace Reference

Tools connected to IO.

Functions

- bool fileExists (const std::string &path)

  Checks whether a file exists.
- bool loadImage16x16 (const std::string &path, uint16_t buffer[256])

  Loads a ppm 16x16 image.

7.3.1 Detailed Description

Tools connected to IO.

7.3.2 Function Documentation

7.3.2.1 fileExists()

```
bool IOTools::fileExists (
             const std::string & path )
```

Checks whether a file exists.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

path	Path to file.

Returns

True if the file exists, false otherwise.

7.3.2.2 loadImage16x16()

```
bool IOTools::loadImage16x16 (
             const std::string & path,
             uint16_t buffer[256] )
```

Loads a ppm 16x16 image.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

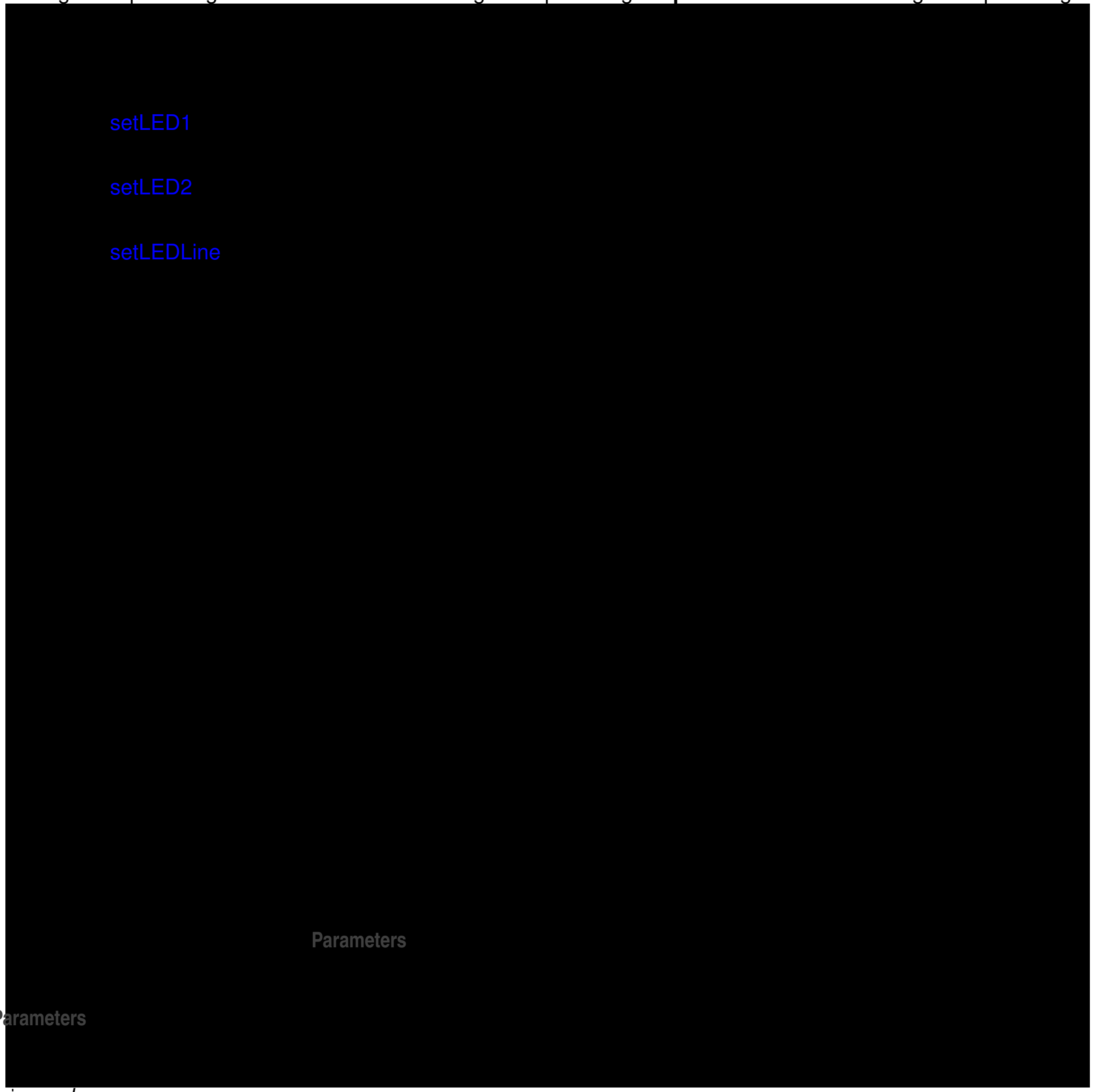
path	Path to file.
buffer	An array to store the image into.

Returns

True if the image was loaded successfully, false otherwise.

7.4 LedController Namespace Reference

Namespace that handles LEDs.



Parameters

Parameters

in *color*

7.4.2.2 setLED2()

```
void LedController::setLED2 (
             uint32_t color )
```

Sets the color of the second led.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]Parameters

Parameters

in *color*

7.4.2.3 setLEDLine()

```
void LedController::setLEDLine (
             uint32_t bits )
```

Sets the on/off state of ledline leds.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

`in`	*bits*	Each bit represents the state of one led.

Chapter 8

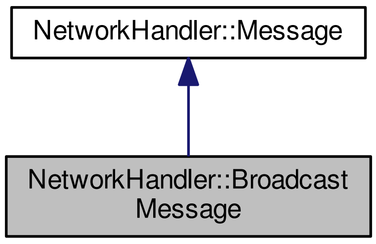
Class Documentation

8.1 NetworkHandler::BroadcastMessage Struct Reference

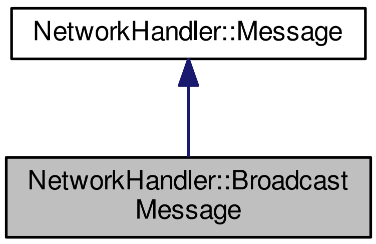
Broadcast message. Type 0.

```
#include <NetworkHandler.h>
```

Inheritance diagram for NetworkHandler::BroadcastMessage:



Collaboration diagram for NetworkHandler::BroadcastMessage:



Public Attributes

- uint32_t rgbCeiling

  RGB ceiling value of sender unit.
- uint32_t rgbWall

  RGB wall value of sender unit.
- char description [16]

  Description of sender unit.
- uint16_t image [256]

  Icon of sender unitt.

8.1.1 Detailed Description

Broadcast message. Type 0.

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

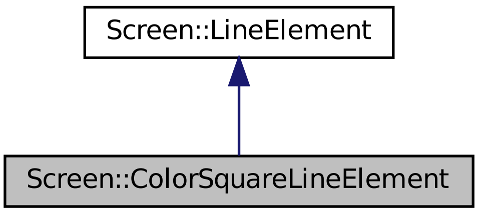
- Network/NetworkHandler.h

8.2 Screen::ColorSquareLineElement Class Reference

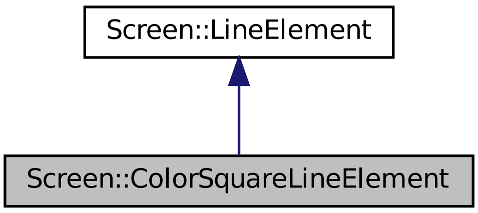
Color square.

`#include <Screen.h>`

Inheritance diagram for Screen::ColorSquareLineElement:



Collaboration diagram for Screen::ColorSquareLineElement:



ColorSquareLineElement color alignRight

ColorSquareLineElement

ColorSquareLineElement color marginLeft marginRight alignRight

SpaceLineElement

renderSelf Display display

color

```
uint16_t color = 0,
int margin = 0,
bool alignRight = false )
```

ColorSquareLineElement constructor.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

`in`	*color*
`in`	*margin*
`in`	*alignRight*

8.2.2.2 ColorSquareLineElement() [2/2]

```
Screen::ColorSquareLineElement::ColorSquareLineElement (
            uint16_t color,
            int marginLeft,
            int marginRight,
            bool alignRight = false )
```

SpaceLineElement constructor with different margins.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

in	*color*
in	*marginLeft*
in	*marginRight*
in	*alignRight*

8.2.3 Member Function Documentation

8.2.3.1 renderSelf()

```
int Screen::ColorSquareLineElement::renderSelf (
            Display * display,
            int x,
            int y ) [virtual]
```

Renders itself at position x, y to display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

in	*display*	Display to render to.
in	*x*	X position to render to.
in	*y*	Y position to render to.

Returns

Number of pixels this elements takes.

Implements Screen::LineElement.

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

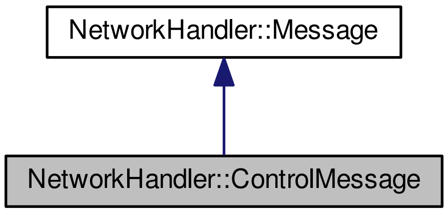
- DisplayUtils/Screen.h
- DisplayUtils/Screen.cpp

8.3 NetworkHandler::ControlMessage Struct Reference

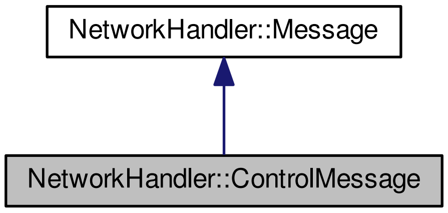
Control message. Types 1 and 2.

`#include <NetworkHandler.h>`

Inheritance diagram for NetworkHandler::ControlMessage:



Collaboration diagram for NetworkHandler::ControlMessage:



Public Attributes

- int16_t valuesCeiling [3]

  Ceiling values to increment/set for the recieving unit.
- int16_t valuesWall [3]

  Wall values to increment/set for the recieving unit.

8.3.1 Detailed Description

Control message. Types 1 and 2.

Type 1 increments values. Type 2 sets values.

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

- Network/NetworkHandler.h

8.4 ControlMessageQueue::ControlMessageInfo Struct Reference

Element of the ControlMessageQueue.

```
#include <ControlMessageQueue.h>
```

Public Member Functions

- ControlMessageInfo ()

  Empty constructor constructs invalid ControlMessageInfo.
- ControlMessageInfo (uint32_t ip, int type=-1)

  ControlMessageInfo constructor with ip and optional type.

Public Attributes

- uint32_t ip

  IP address.
- int type

  Message type.
- int16_t valuesCeiling [3]

  Values ceiling.
- int16_t valuesWall [3]

  Values wall.

8.4.1 Detailed Description

Element of the ControlMessageQueue.

See also

NetworkHandler::Message
NetworkHandler::ControlMessage

8.4.2 Constructor & Destructor Documentation

8.4.2.1 ControlMessageInfo()

```
ControlMessageQueue::ControlMessageInfo::ControlMessageInfo (
              uint32_t ip,
              int type = -1 )  [inline]
```

ControlMessageInfo constructor with ip and optional type.

If type is set to 2, this constructor also sets all values to -1. =1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

in	*ip*
in	*type*

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

- Misc/ControlMessageQueue.h

8.5 ControlMessageQueue Class Reference

Thread-safe FIFO queue for control messages.

```
#include <ControlMessageQueue.h>
```

Classes

- struct ControlMessageInfo

  Element of the ControlMessageQueue.

Public Types

- typedef struct ControlMessageQueue::ControlMessageInfo ControlMessageInfo

  Element of the ControlMessageQueue.

Public Member Functions

- bool hasChanged ()

  Checks if the queue has changed.
- size_t size ()

  Returns number of elements in the queue.
- void enqueue (ControlMessageInfo info)

  Equeues an element at the back.
- ControlMessageInfo dequeue ()

  Dequeues an element from the front.

8.5.1 Detailed Description

Thread-safe FIFO queue for control messages.

8.5.2 Member Typedef Documentation

8.5.2.1 ControlMessageInfo

```
typedef struct ControlMessageQueue::ControlMessageInfo ControlMessageQueue::ControlMessageInfo
```

Element of the ControlMessageQueue.

See also

NetworkHandler::Message
NetworkHandler::ControlMessage

8.5.3 Member Function Documentation

8.5.3.1 dequeue()

```
ControlMessageQueue::ControlMessageInfo ControlMessageQueue::dequeue ( )
```

Dequeues an element from the front.

Returns

The front element. If there are no elements, returns ControlMessageInfo();

8.5.3.2 enqueue()

```
void ControlMessageQueue::enqueue (
            ControlMessageQueue::ControlMessageInfo info )
```

Equeues an element at the back.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

`in`	*info*

8.5.3.3 hasChanged()

```
bool ControlMessageQueue::hasChanged ( )
```

Checks if the queue has changed.

Returns

Value indicating wether the queue has changed since the last call to this function.

8.5.3.4 size()

```
size_t ControlMessageQueue::size ( )
```

Returns number of elements in the queue.

Returns

Number of elements in the queue.

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

- Misc/ControlMessageQueue.h
- Misc/ControlMessageQueue.cpp

8.6 DeviceInput Class Reference

Handler device input.

```
#include <DeviceInput.h>
```

Public Member Functions

- DeviceInput ()
- ∼DeviceInput ()
- void update ()

  Gets the input (knobs state) from the device.

Public Attributes

- int8_t RGBDelta [3]

  The change in the device knob positions.
- bool RGBPressed [3]

  Wether given device knob is pressed or not.

8.6.1 Detailed Description

Handler device input.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 DeviceInput()

```
DeviceInput::DeviceInput ( )
```

Constructor.

8.6.2.2 ∼DeviceInput()

```
DeviceInput::∼DeviceInput ( )
```

Destructor.

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

- MZApi/DeviceInput.h
- MZApi/DeviceInput.cpp

8.7 Display Class Reference

A class that handles the (one-way) interaction with the device display and provides methods for rendering shapes, text, and other.

```
#include <Display.h>
```

Public Member Functions

- Display (uint16_t bgColour, uint16_t fgColour, uint16_t highlightColour, font_descriptor_t font)
  The display constructor taking colours and fonts as parameters.
- ∼Display ()
  The display destructor.
- void handleInput (int8_t rgbDelta[3], bool knobsPressed[3])
  Reacts to input from the device.
- void switchScreen (Screen ∗newScreen)
  Changes the display screen.
- bool toPreviousScreen (bool keepAlive=false)
  Returns to the previous screen.
- void setColours (uint16_t bgColour, uint16_t fgColour, uint16_t highlightColour)
  Sets the base colours for the display - background, foreground and highlight.
- void setFont (font_descriptor_t font)
  Sets the font for the display.
- size_t textWidth (std::string &text)
  Calculates text width in pixels.
- void clearScreen (uint16_t colour)
  Sets the whole display to one colour.
- void setPixel (int x, int y, uint16_t colour)
  Sets one pixel to a given colour.
- void renderRectangle (int left, int top, int right, int bottom, uint16_t colour)
  Renders an axis-aligned rectangle with given corner points in a given colour.
- void renderColourSquare (int topX, int topY, uint16_t colour)
  Renders an axis-aligned rectangle with given position in a given colour.
- size_t renderText (int topX, int topY, std::string text, uint16_t colour)
  Renders an axis-aligned text-line starting at a given position in a given colour.
- size_t renderIcon (uint16_t ∗buffer, int topX, int topY, int exponent=0)
  Renders an icon starting at a given position.
- void redraw ()
  Renders the display buffer on the device.

Public Attributes

- uint16_t fgColour

  Current theme text colour.
- uint16_t bgColour

  Current theme background colour.
- uint16_t selectColour

  Current theme selected background colour.
- size_t lineMax

  The maximum number of lines that fit on the display.

Static Public Attributes

- static const size_t width = 480

  The display width.
- static const size_t height = 320

  The display height.

8.7.1 Detailed Description

A class that handles the (one-way) interaction with the device display and provides methods for rendering shapes, text, and other.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 Display()

```
Display::Display (
            uint16_t bgColour,
            uint16_t fgColour,
            uint16_t highlightColour,
            font_descriptor_t font )
```

The display constructor taking colours and fonts as parameters.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

bgColour	The colour used for background.
fgColour	The colour used for foreground.
highlightColour	The colour used for highlighted items, such as the sleected ones.
font	The font to be used for displayed text.

8.7.3 Member Function Documentation

8.7.3.1 clearScreen()

```
void Display::clearScreen (
             uint16_t colour )
```

Sets the whole display to one colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

colour	The colour used as background.

8.7.3.2 handleInput()

```
void Display::handleInput (
             int8_t rgbDelta[3],
             bool knobsPressed[3] )
```

Reacts to input from the device.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

rgbDelta	The change in knobs position.
knobsPressed	The high/low state of the knobs.

8.7.3.3 renderColourSquare()

```
void Display::renderColourSquare (
             int topX,
             int topY,
             uint16_t colour )
```

Renders an axis-aligned rectangle with given position in a given colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

topX	The x coordinate of the left edge.
topY	The y coordinate of the top edge.
colour	The colour of the rectangle.

8.7.3.4 renderIcon()

```
size_t Display::renderIcon (
             uint16_t * buffer,
             int topX,
             int topY,
             int exponent = 0 )
```

Renders an icon starting at a given position.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

buffer	The 16x16 image to be rendered.
topX	The x coordinate of the top-left corner.
topY	The y coordinate of the top-left corner.
exponent	The exponent to use for scaling by powers of two.

Returns

Width rendered icon in pixels.

8.7.3.5 renderRectangle()

```
void Display::renderRectangle (
             int left,
             int top,
             int right,
             int bottom,
             uint16_t colour )
```

Renders an axis-aligned rectangle with given corner points in a given colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

left	The x coordinate of the left edge.
top	The y coordinate of the top edge.
right	The x cooridnate of the right edge.
bottom	The y coordinate of the bottom edgge.
colour	The colour of the rectangle.

8.7.3.6 renderText()

```
size_t Display::renderText (
             int topX,
             int topY,
             std::string text,
             uint16_t colour )
```

Renders an axis-aligned text-line starting at a given position in a given colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

topX	The x coordinate of the top-left corner.
topY	The y coordinate of the top-left corner.
text	The text to be rendered.
colour	The colour of the text.

Returns

Width rendered text in pixels.

Parameters

Parameters

Returns

8.7.3.8 setFont()

```
void Display::setFont (
             font_descriptor_t font )
```

Sets the font for the display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

---

font	The font to be used for displayed text.

8.7.3.9 setPixel()

```
void Display::setPixel (
             int x,
             int y,
             uint16_t colour )
```

Sets one pixel to a given colour.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

x	The pixel x coordinate.
y	The pixel y coordinate.
colour	The colour of the pixel.

8.7.3.10 switchScreen()

```
void Display::switchScreen (
             Screen * newScreen )
```

Changes the display screen.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

newScreen	The new screen.

8.7.3.11 textWidth()

```
size_t Display::textWidth (
            std::string & text )
```

Calculates text width in pixels.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

text	The text to calculate width for.

Returns

Width of the passed text in pixels.

8.7.3.12 toPreviousScreen()

```
bool Display::toPreviousScreen (
            bool keepAlive = false )
```

Returns to the previous screen.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

keepAlive	Whether the current screen should be preserved so that it can be returned to later.

Returns

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

- MZApi/Display.h
- MZApi/Display.cpp

8.8 font_descriptor_t Struct Reference

Bitmap font struct.

```
#include <font_types.h>
```

Public Attributes

- char ∗ name

  font name
- int maxwidth

  max width in pixels
- unsigned int height

  height in pixels
- int ascent

  ascent (baseline) height
- int firstchar

  first character in bitmap
- int size

  font size in characters
- const font_bits_t ∗ bits

  16-bit right-padded bitmap data
- const uint32_t ∗ offset

  offsets into bitmap data
- const unsigned char ∗ width

  character widths or 0 if fixed
- int defaultchar

  default char (not glyph index)
- int32_t bits_size

  words of MWIMAGEBITS bits

8.8.1 Detailed Description

Bitmap font struct.

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

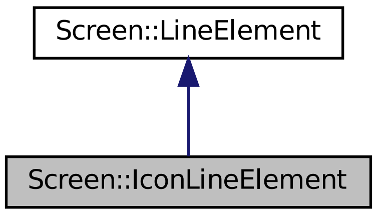
- DisplayUtils/font_types.h

8.9 Screen::IconLineElement Class Reference

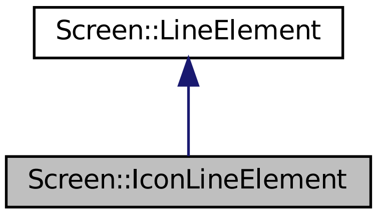
Scaled icon.

```
#include <Screen.h>
```

Inheritance diagram for Screen::IconLineElement:



Collaboration diagram for Screen::IconLineElement:



Public Member Functions

- IconLineElement (uint16_t ∗pIcon=NULL, int scaleExponent=0, int margin=0, bool alignRight=false)

  IconLineElement constructor.
- IconLineElement (uint16_t ∗pIcon, int scaleExponent, int marginLeft, int marginRight, bool alignRight=false)

  IconLineElement constructor with different margins.
- int renderSelf (Display ∗display, int x, int y)

  Renders itself at position x, y to display.

Protected Attributes

- uint16_t ∗ pIcon

  Pointer to the icon buffer.
- int scaleExponent

  Exponent to scale icon by.

Additional Inherited Members

8.9.1 Detailed Description

Scaled icon.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 IconLineElement() [1/2]

```
Screen::IconLineElement::IconLineElement (
             uint16_t * pIcon = NULL,
             int scaleExponent = 0,
             int margin = 0,
             bool alignRight = false )
```

IconLineElement constructor.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

in	*pIcon*
in	*scaleExponent*
in	*margin*
in	*alignRight*

8.9.2.2 IconLineElement() [2/2]

```
Screen::IconLineElement::IconLineElement (
             uint16_t * pIcon,
             int scaleExponent,
             int marginLeft,
             int marginRight,
             bool alignRight = false )
```

IconLineElement constructor with different margins.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

in	*pIcon*	
in	*scaleExponent*	
in	*marginLeft*	
in	*marginRight*	
in	*alignRight*	

8.9.3 Member Function Documentation

8.9.3.1 renderSelf()

```
int Screen::IconLineElement::renderSelf (
             Display * display,
             int x,
             int y )  [virtual]
```

Renders itself at position x, y to display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

in	*display*	Display to render to.
in	*x*	X position to render to.
in	*y*	Y position to render to.

Returns

Number of pixels this elements takes.

Implements Screen::LineElement.

8.9.4 Member Data Documentation

8.9.4.1 scaleExponent

```
int Screen::IconLineElement::scaleExponent  [protected]
```

Exponent to scale icon by.

See also

Display::renderIcon

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

- DisplayUtils/Screen.h
- DisplayUtils/Screen.cpp

8.10 LightUnit Class Reference

A class representing a light unit.

```
#include <LightUnit.h>
```

Public Member Functions

- LightUnit ()

  Default LightUnit constructor.
- LightUnit (const char description[16])

  LightUnit constructor with unit description.
- LightUnit (unsigned long ip, const char description[16], const uint16_t image[256])

  LightUnit constructor with ip, unit description and image as parameters.
- LightUnit (unsigned long ip, const char description[16], const uint16_t image[256], uint32_t rgbCeiling, uint32_t rgbWall)

  LightUnit constructor with complete information.
- ∼LightUnit ()

Public Attributes

- uint32_t rgbCeiling = 0

  The RGB888 colour of the ceiling.
- uint32_t rgbWall = 0

  The RGB888 colour of the wall.
- char description [17]

  Unit dscription.
- uint16_t image [256]

  Unit icon.
- unsigned long ip = 0

  Unit IP.
- std::chrono::steady_clock::time_point lastNetworkBroadcastTimePoint

  Time of the last recieved broadcast.
- std::mutex mutex_change

  Mutex to prevent simultaneous changing of variables.
- std::atomic_bool screenActive

  Flag to prevent unit ereasure from unitList when it's active in UnitScreen.

[redacted]

```
              unsigned long ip,
              const char description[16],
              const uint16_t image[256] )
```

LightUnit constructor with ip, unit description and image as parameters.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

ip	The IP address of the light unit.
description	Description of the light unit.
image	Icon for the light unit.

8.10.2.3 LightUnit() [3/3]

```
LightUnit::LightUnit (
            unsigned long ip,
            const char description[16],
            const uint16_t image[256],
            uint32_t rgbCeiling,
            uint32_t rgbWall )
```

LightUnit constructor with complete information.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

ip	The IP address of the light unit.
description	Description of the light unit.
image	Icon for the light unit.
rgbCeiling	RGB888 colour of the light unit ceiling.
rgbWall	RGB888 colour of the light unit wall.

8.10.2.4 ∼LightUnit()

```
LightUnit::∼LightUnit ( )
```

LightUnit destructor.

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

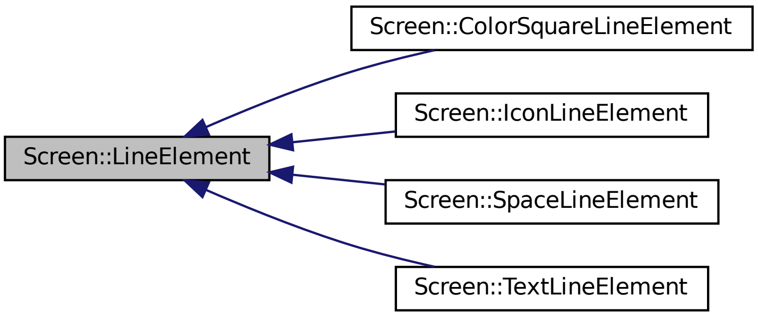
- Unit/LightUnit.h
- Unit/LightUnit.cpp

8.11 Screen::LineElement Class Reference

Line element base class.

```
#include <Screen.h>
```

Inheritance diagram for Screen::LineElement:



Public Member Functions

- LineElement (int margin=0, bool alignRight=false)

  LineElement constructor.
- LineElement (int marginLeft, int marginRight, bool alignRight=false)

  LineElement constructor with different margins.
- virtual int renderSelf (Display ∗display, int x, int y)=0

  Renders itself at position x, y to display.

Public Attributes

- bool alignRight

  Wether to align left or right.

Protected Attributes

- int marginLeft

  Left margin of this element. Can be negative.
- int marginRight

  Right margin of this element. Can be negative.

8.11.1 Detailed Description

Line element base class.

LineElement

Parameters

Parameters

```
int marginLeft,
int marginRight,
bool alignRight = false )
```

LineElement constructor with different margins.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

`in`	*marginLeft*
`in`	*marginRight*
`in`	*alignRight*

8.11.3 Member Function Documentation

8.11.3.1 renderSelf()

```
virtual int Screen::LineElement::renderSelf (
            Display * display,
            int x,
            int y )  [pure virtual]
```

Renders itself at position x, y to display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

`in`	*display*	Display to render to.
`in`	*x*	X position to render to.
`in`	*y*	Y position to render to.

Returns

Number of pixels this elements takes.

Implemented in Screen::IconLineElement, Screen::TextLineElement, Screen::ColorSquareLineElement, and Screen::SpaceLineElement.

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

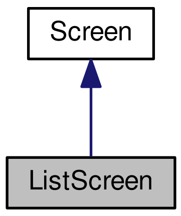
- DisplayUtils/Screen.h
- DisplayUtils/Screen.cpp

8.12 ListScreen Class Reference

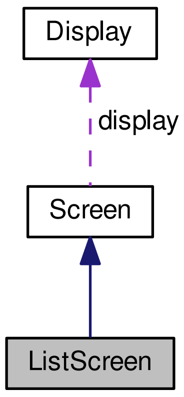
Unit screen.

```
#include <ListScreen.h>
```

Inheritance diagram for ListScreen:



Collaboration diagram for ListScreen:



Public Member Functions

- ListScreen (Display *display)

  ListScreen constructor.
- void renderScreen ()

  Renders this screen to display.
- void handleKnobChange (int8_t RGBDelta[3])

  Handles knob changes.
- void handleKnobPress (bool RGBPressed[3])

  Handles knob presses.

Additional Inherited Members

8.12.1 Detailed Description

Unit screen.

This screen renders list of all connected units and allows selection to pass to Unit screen.

See also

UnitScreen

8.12.2 Constructor & Destructor Documentation

8.12.2.1 ListScreen()

```
ListScreen::ListScreen (
            Display * display )
```

ListScreen constructor.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

`in`	*display*	Display to be bound to.

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

- DisplayUtils/ListScreen.h
- DisplayUtils/ListScreen.cpp

8.13 Mapper Class Reference

A class that handles the mapping of peripheral physical regions to virtual addresses.

```
#include <Mapper.h>
```

Public Member Functions

- Mapper (off_t base, size_t size)

Public Attributes

- unsigned char $*$ mem_base = NULL

  Base addres to which the mapping is bound.

8.13.1 Detailed Description

A class that handles the mapping of peripheral physical regions to virtual addresses.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 Mapper()

```
Mapper::Mapper (
             off_t base,
             size_t size )
```

Constructor for Mapper. =1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|Parameters

Parameters

base	The base region of the peripheral.
size	Address range for the region.

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

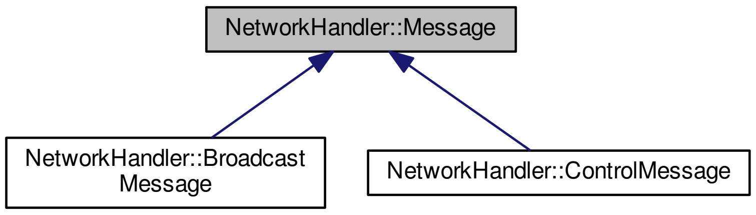
- MZApi/Mapper.h
- MZApi/Mapper.cpp

8.14 NetworkHandler::Message Struct Reference

Base message struct.

`#include <NetworkHandler.h>`

Inheritance diagram for NetworkHandler::Message:


NetworkHandler::Message
NetworkHandler::Broadcast Message
NetworkHandler::ControlMessage

Public Attributes

- uint32_t magic

  Magic number to detect packets.
- uint32_t version

  Version.
- uint32_t msgType

  Type of message.

8.14.1 Detailed Description

Base message struct.

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

- Network/NetworkHandler.h

8.15 NetworkHandler Class Reference

Class that handles all network communication.

```
#include <NetworkHandler.h>
```

Classes

- struct BroadcastMessage

  Broadcast message. Type 0.
- struct ControlMessage

  Control message. Types 1 and 2.
- struct Message

  Base message struct.
- struct RecievedMessage

  Struct that represents a recieved message.

Public Types

- typedef struct NetworkHandler::Message Message

  Base message struct.
- typedef NetworkHandler::BroadcastMessage BroadcastMessage

  Broadcast message. Type 0.
- typedef NetworkHandler::ControlMessage ControlMessage

  Control message. Types 1 and 2.
- typedef struct NetworkHandler::RecievedMessage RecievedMessage

  Struct that represents a recieved message.

Public Member Functions

- bool broadcastMessage (const BroadcastMessage ∗message)

  Broadcasts message.
- bool broadcastUnit (LightUnit &unit)

  Builds and broadcasts message for given unit.
- bool sendMessage (const ControlMessage ∗message, uint32_t ip)

  Sends control message.
- BroadcastMessage buildBroadcastMessage (LightUnit &unit)

  Builds broadcast message for unit.
- ControlMessage buildControlMessage (int type, int16_t valuesCeiling[ ], int16_t valuesWall[ ])

  Builds control message.
- RecievedMessage recieveMessage ()

  Waits for a message on the socket. Can timeout.

8.15.1 Detailed Description

Class that handles all network communication.

8.15.2 Member Typedef Documentation

8.15.2.1 ControlMessage

```
typedef NetworkHandler::ControlMessage NetworkHandler::ControlMessage
```

Control message. Types 1 and 2.

Type 1 increments values. Type 2 sets values.

8.15.3 Member Function Documentation

8.15.3.1 broadcastMessage()

```
bool NetworkHandler::broadcastMessage (
             const BroadcastMessage * message )
```

Broadcasts message.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

`in`	*message*

Returns

Value indicating if the message was sent successfully.

LightUnit

Parameters

Parameters

Returns

NetworkHandler::BroadcastMessage
LightUnit

Builds broadcast message for unit.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

`in` *unit*

Returns

Built broadcast message.

8.15.3.4 buildControlMessage()

```
NetworkHandler::ControlMessage NetworkHandler::buildControlMessage (
                int type,
                int16_t valuesCeiling[ ],
                int16_t valuesWall[ ] )
```

Builds control message.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

in	*type*	Type of the message
in	*valuesCeiling*	Ceiling values to set in the message.
in	*valuesWall*	Wall values to set in the message.

See also

Message

Returns

Built control message.

8.15.3.5 recieveMessage()

```
NetworkHandler::RecievedMessage NetworkHandler::recieveMessage ( )
```

Waits for a message on the socket. Can timeout.

Returns

RecievedMessage only valid when ip != 0.

8.15.3.6 sendMessage()

```
bool NetworkHandler::sendMessage (
            const ControlMessage * message,
            uint32_t ip )
```

Sends control message.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]Parameters

Parameters

in	*message*
in	*ip*

Returns

Value indicating if the message was sent successfully.

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

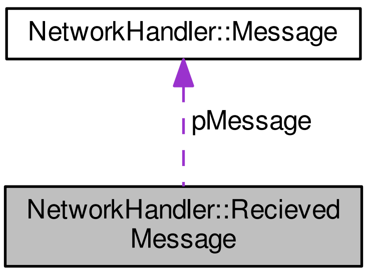
- Network/NetworkHandler.h
- Network/NetworkHandler.cpp

8.16 NetworkHandler::RecievedMessage Struct Reference

Struct that represents a recieved message.

`#include <NetworkHandler.h>`

Collaboration diagram for NetworkHandler::RecievedMessage:



Public Member Functions

- ∼RecievedMessage ()

  Desctructor. Deletes pMessage.

Public Attributes

- uint32_t ip

  IP address of the sender. Set to 0 to represent invalid message (e.g. on timeout).
- std::chrono::steady_clock::time_point timePoint

  Time point at which the message was recieved.
- Message ∗ pMessage

  Pointer to the message.

8.16.1 Detailed Description

Struct that represents a recieved message.

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

- Network/NetworkHandler.h

8.17 RWMutex Class Reference

Read/Write mutex class.

`#include <RWMutex.h>`

Public Member Functions

- void lockRead ()

  Locks this mutex for reading. This is a shared lock.
- void unlockRead ()

  Unlocks one reader lock.
- void lockWrite ()

  Locks write mutex.
- void unlockWrite ()

  Unlocks write mutex.

8.17.1 Detailed Description

Read/Write mutex class.

8.17.2 Member Function Documentation

8.17.2.1 lockRead()

```
void RWMutex::lockRead ( )
```

Locks this mutex for reading. This is a shared lock.

This is a shared mutex and it's implemented as a counter that is incremented on lock and decremented on unlock. This class also implements an anti-writer-starvation mechanism (write-preferring RW lock).

8.17.2.2 lockWrite()

```
void RWMutex::lockWrite ( )
```

Locks write mutex.

This is an exclusive mutex. It is a write-preferring lock.

8.17.2.3 unlockRead()

```
void RWMutex::unlockRead ( )
```

Unlocks one reader lock.

Decrements internal reader counter.

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

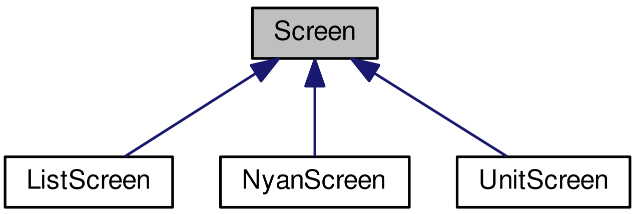
- Misc/RWMutex.h
- Misc/RWMutex.cpp

8.18 Screen Class Reference

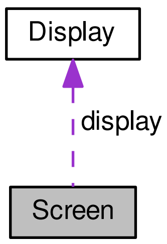
Class representing one screen.

```
#include <Screen.h>
```

Inheritance diagram for Screen:



Collaboration diagram for Screen:



Classes

- class ColorSquareLineElement

    Color square.
- class IconLineElement

    Scaled icon.
- class LineElement

    Line element base class.
- class SpaceLineElement

    Empty space.
- class TextLineElement

    Text.

Public Member Functions

- virtual void renderScreen ()=0

    Renders this screen to display.
- virtual void handleKnobChange (int8_t RGBDelta[3])=0

    Handles knob changes.
- virtual void handleKnobPress (bool RGBPressed[3])=0

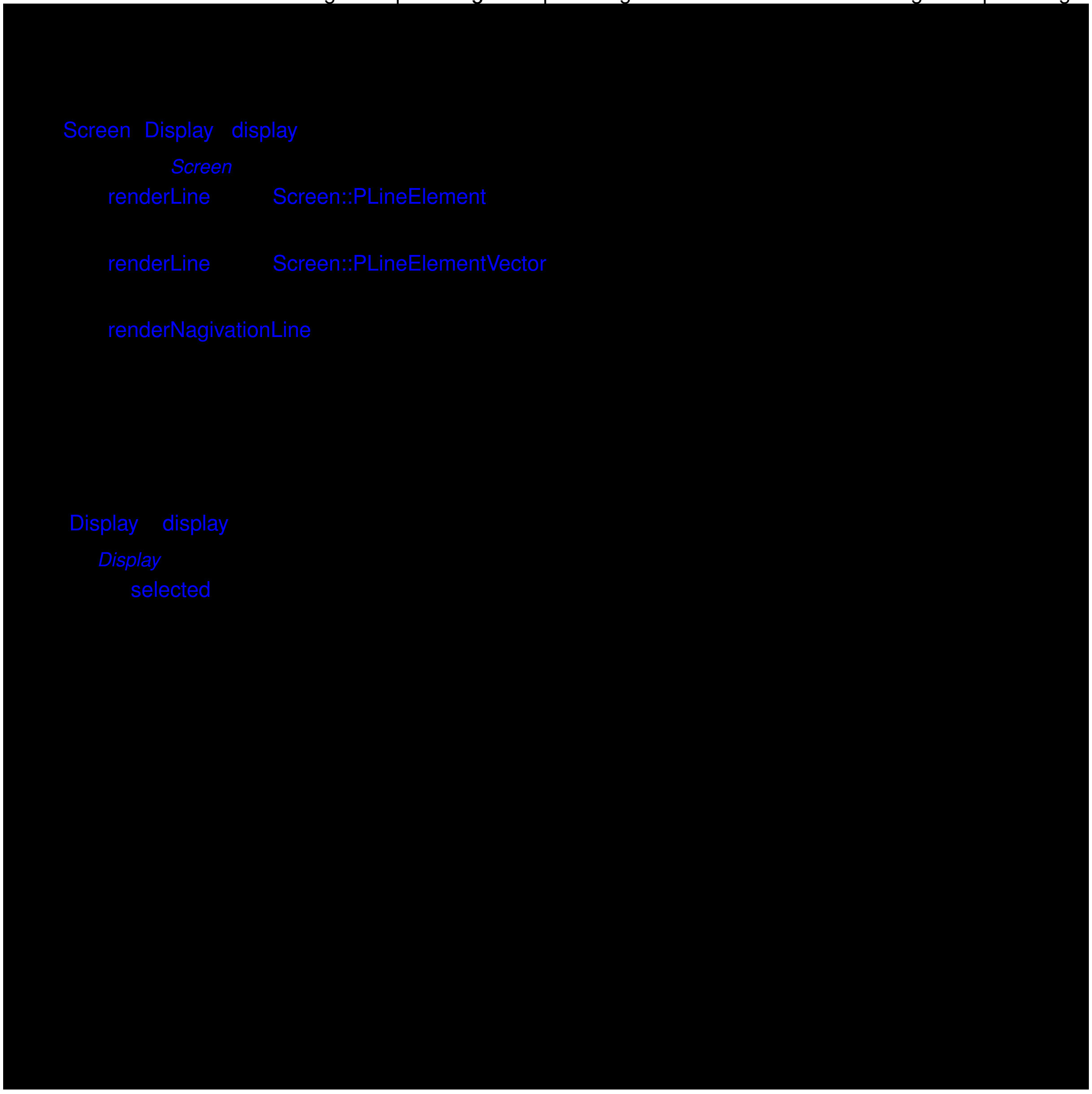
    Handles knob presses.

Protected Types

- typedef std::unique_ptr< LineElement > PLineElement

    One line element pointer.
- typedef std::vector< Screen::PLineElement > PLineElementVector

    Vector of line element pointers.



```
Screen::Screen (
            Display * display )   [protected]
```

Creates Screen and binds it to display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

`in`	*display*

8.18.3 Member Function Documentation

8.18.3.1 renderLine() [1/2]

```
void Screen::renderLine (
            int y,
            Screen::PLineElement element,
            int32_t lineColor = -1 )  [protected]
```

Renders line containing only one element.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]Parameters

Parameters

in	*y*	Y position to render the line at.
in	*element*	Element to render in this line.
in	*lineColor*	Color of this line. Pass -1 to not draw any line background.

8.18.3.2 renderLine() [2/2]

```
void Screen::renderLine (
            int y,
            Screen::PLineElementVector & elements,
            int32_t lineColor = -1 )  [protected]
```

Renders line containing elements.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]Parameters

Parameters

in	*y*	Y position to render the line at.
in	*elements*	Elements to render in this line.
in	*lineColor*	Color of this line. Pass -1 to not draw any line background.

8.18.3.3 renderNagivationLine()

```
void Screen::renderNagivationLine (
            std::vector< std::pair< std::string, std::string >> uiInputPairs )  [protected]
```

Renders navigation line.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]Parameters

Parameters

in	*uiInputPairs*	Vector of string pairs (icon_name, text) to render.

See also

Engine::uiIcons

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

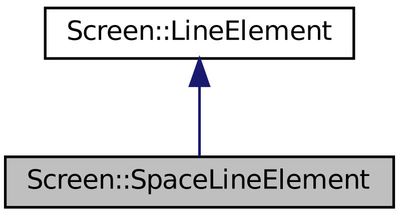
- DisplayUtils/Screen.h
- DisplayUtils/Screen.cpp

8.19 Screen::SpaceLineElement Class Reference

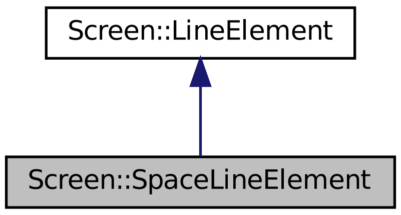
Empty space.

`#include <Screen.h>`

Inheritance diagram for Screen::SpaceLineElement:



Collaboration diagram for Screen::SpaceLineElement:



Public Member Functions

- SpaceLineElement (int size, bool alignRight=false)

  SpaceLineElement constructor.
- int renderSelf (Display ∗display, int x, int y)

  Renders itself at position x, y to display.

Additional Inherited Members

8.19.1 Detailed Description

Empty space.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 SpaceLineElement()

```
Screen::SpaceLineElement::SpaceLineElement (
             int size,
             bool alignRight = false )
```

SpaceLineElement constructor.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters
`in` *size*
`in` *alignRight*

8.19.3 Member Function Documentation

8.19.3.1 renderSelf()

```
int Screen::SpaceLineElement::renderSelf (
            Display * display,
            int x,
            int y )  [virtual]
```

Renders itself at position x, y to display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|**Parameters**

Parameters

in	*display*	Display to render to.
in	*x*	X position to render to.
in	*y*	Y position to render to.

Returns

Number of pixels this elements takes.

Implements Screen::LineElement.

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

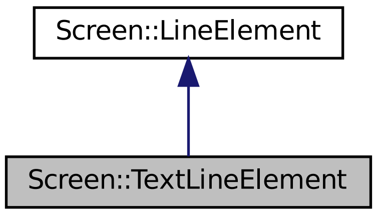
- DisplayUtils/Screen.h
- DisplayUtils/Screen.cpp

8.20 Screen::TextLineElement Class Reference

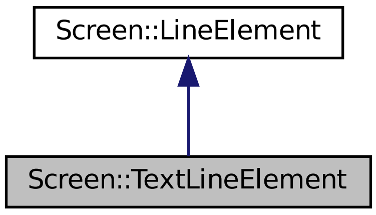
Text.

```
#include <Screen.h>
```

Inheritance diagram for Screen::TextLineElement:



Collaboration diagram for Screen::TextLineElement:



Public Member Functions

- TextLineElement (std::string text="", uint16_t color=0, int margin=0, bool alignRight=false)

  TextLineElement constructor.
- TextLineElement (std::string text, uint16_t color, int marginLeft, int marginRight, bool alignRight=false)

  TextLineElement constructor with different margins.
- int renderSelf (Display ∗display, int x, int y)

  Renders itself at position x, y to display.

Protected Attributes

- std::string text

  The text.
- uint16_t color

  Color of the text.

Additional Inherited Members

8.20.1 Detailed Description

Text.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 TextLineElement() `[1/2]`

```
Screen::TextLineElement::TextLineElement (
             std::string text = "",
             uint16_t color = 0,
             int margin = 0,
             bool alignRight = false )
```

TextLineElement constructor.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]**Parameters**

Parameters

`in`	*text*
`in`	*color*
`in`	*margin*
`in`	*alignRight*

8.20.2.2 TextLineElement() [2/2]

```
Screen::TextLineElement::TextLineElement (
             std::string text,
             uint16_t color,
             int marginLeft,
             int marginRight,
             bool alignRight = false )
```

TextLineElement constructor with different margins.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

`in` *text*	
`in` *color*	
`in` *marginLeft*	
`in` *marginRight*	
`in` *alignRight*	

8.20.3 Member Function Documentation

8.20.3.1 renderSelf()

```
int Screen::TextLineElement::renderSelf (
             Display * display,
             int x,
             int y )  [virtual]
```

Renders itself at position x, y to display.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

`in` *display*	Display to render to.
`in` *x*	X position to render to.
`in` *y*	Y position to render to.

Returns

Number of pixels this elements takes.

Implements Screen::LineElement.

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

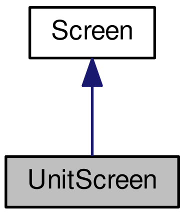
- DisplayUtils/Screen.h
- DisplayUtils/Screen.cpp

8.21 UnitScreen Class Reference

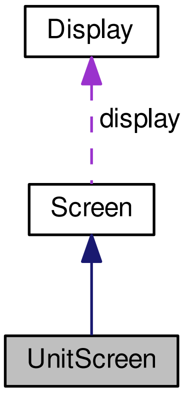
Unit screen.

`#include <UnitScreen.h>`

Inheritance diagram for UnitScreen:



Collaboration diagram for UnitScreen:



Public Member Functions

- UnitScreen (Display ∗display, LightUnit &unit)

  UnitScreen constructor.
- void renderScreen ()

  Renders this screen to display.
- void handleKnobChange (int8_t RGBDelta[3])

  Handles knob changes.
- void handleKnobPress (bool RGBPressed[3])

  Handles knob presses.

Additional Inherited Members

8.21.1 Detailed Description

Unit screen.

This screen renders info about selected unit and allows making changes to units light settings.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 UnitScreen()

```
UnitScreen::UnitScreen (
            Display ∗ display,
            LightUnit & unit )
```

UnitScreen constructor.

=1mm

spread 0pt [l]|X[-1,l]|X[-1,l]|X[-1,l]|Parameters

Parameters

in	*display*	Display to be bound to.
in	*unit*	LightUnit to be bound to.

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

- DisplayUtils/UnitScreen.h
- DisplayUtils/UnitScreen.cpp