# Graphics Libraries Help

MPLAB Harmony Integrated Software Framework

# Volume V: MPLAB Harmony Framework Reference

This volume provides API reference information for the framework libraries included in your installation of MPLAB Harmony.

## Description


This volume is a programmer reference that details the interfaces to the libraries that comprise MPLAB Harmony and explains how to use the libraries individually to accomplish the tasks for which they were designed.

# Graphics Libraries Help

This topic provides information about the graphics libraries that are available in MPLAB Harmony.

Currently, MPLAB Harmony provides two solutions for developing graphics firmware:

- The MPLAB Harmony Graphics Composer (MHGC) Suite - The MPLAB Harmony Graphics Composer Suite (MHGC) is a free, modular graphics stack and tools suite for use with Microchip PIC32 microcontrollers. The MHGC tool provides an easy to use GUI that works within the MPLAB X IDE environment.
- The SEGGER emWin Graphics Library - In addition to the standard Graphics Library, the MPLAB Harmony Integrated Software Framework also offers a third-party graphics library, emWin, from SEGGER Microcontroller GmbH & Co. KG. The SEGGER emWin Graphics Library provides an efficient, processor and LCD controller-independent Graphical User Interface (GUI) for applications that operate with a graphical LCD.

## MPLAB Harmony Graphics Composer (MHGC) Suite

This section describes the MPLAB Harmony Graphics Composer (MHGC) Suite.

### Introduction

This section provides an overview of the MPLAB Harmony Graphics Composer (MHGC) Suite.

#### Description

The MPLAB Harmony Graphics Composer (MHGC) Suite is a free, modular graphics stack and tools suite for use with Microchip PIC32 microcontrollers. The MHGC suite provides an easy to use GUI that works within the MPLAB X IDE environment. This is tightly coupled with MPLAB Harmony Configurator (MHC), code development, and other integrated debug features. The tools provide a simplified interface to create graphics content, target specific processor / display and touch interface hardware, and generate code. In most cases, no additional programming to support graphics is required at all, which reduces development time. For more information about the MHGC, refer to MPLAB Harmony Graphics Composer User's Guide.

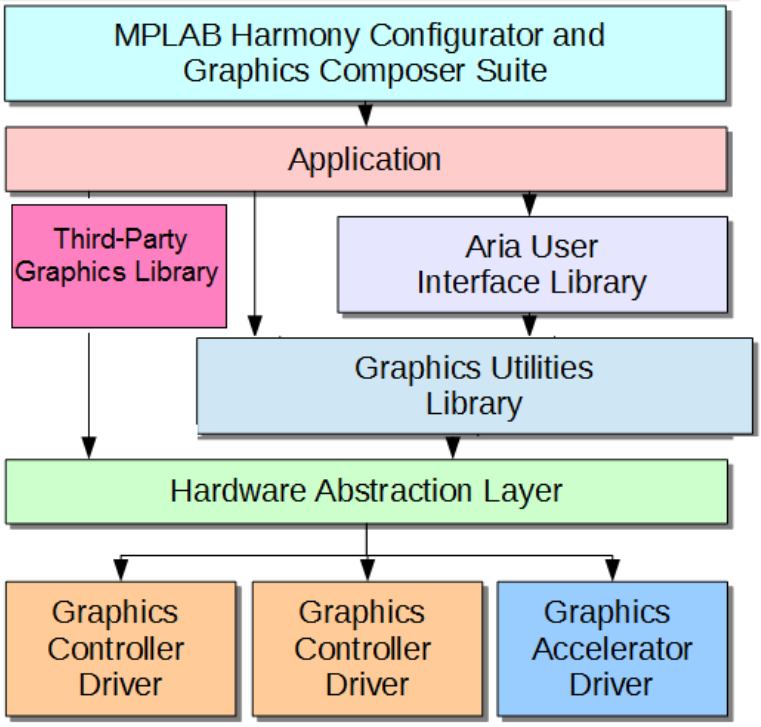
The MPLAB Harmony Graphics Stack consists of several layers that directly build on the capabilities of lower layers to provide a robust framework for displaying rich graphics on supported display devices. Higher layers can be removed as needed if their capabilities are not required.

### Graphics Stack Architecture

This section describes the graphics stack architecture.

#### Description

The following is a diagram of the graphics stack architecture.




The functionality of each layer is summarized below, the details have been provided in the help document in the corresponding sections:

- Graphics Controller Driver – Software that talks directly to hardware. Multiple drivers for internal, external and no-controller options are available. These can be customized with the new Display Manager interface. No other software in the stack should have hardware access.
- Graphics Accelerator Driver- Software that interfaces with graphics accelerator hardware.
- Hardware Abstraction Layer (HAL) – A software layer that serves as a gate-keeper for all graphics controller and accelerator drivers. This layer is configured at initialization by the underlying graphics drivers and provides functionality such as: buffer management, primitive shape drawing, hardware abstraction, and draw state management. The presence of this layer serves as a means of protection for the drivers, frame buffers, and draw state in order to prevent state mismanagement by the application.
- Graphics Utilities Library – This library is primarily responsible for managing and decoding assets such as images, fonts, and strings. It provides the means for interacting with asset data, complex data decoding, data decompression, and string asset look-up. It also abstractly handles accessing external memory sources during asset decoding.
- Aria User Interface Library – This library provides the capability for interface generation, management, and interaction. This library provides the building blocks for constructing a user interface in the form of “Widgets” or user interface elements. These consist of things like buttons, check-boxes, images, etc. This library also handles user interaction events for things like touch actions.
- Third Party Graphics Library – The third party library can be used with the harmony framework to perform the graphics operations if desired by the user. The third party library has access to the hardware abstraction layer (HAL), which has been configured to supply the frame buffer to be filled in by the third-party graphics library.
- MPLAB Harmony Graphics Composer (MHGC) – This tools suite provides the capability to design a user interface using a graphical drag and drop interface. The tool can write all of the code needed to initialize, configure, and manage an Aria library context. The tools include:
  - Graphics Asset Converter – new engine for importing multiple external image types, estimating and optimizing size,
  - Image Editor - Enabling palette, compression, format changes and editing of images without external tools
  - Resource Manager – Tabulated totals of memory usage for images, fonts and other elements used within the graphics design. These can be used to optimize a specific design to fit within a given device Flash memory.
  - WYSIWYG GUI editor – Enables drag and drop capability to visualize your particular design
  - Event Manager – Enables the user to customize the experience of touch and logical (application) events and to interact with graphical attributes
  - Tree Manager – Enables the user to select the drawing priority and establish parent / child relationships so that objects can be grouped as desired
  - String and Font Manager – Used to input strings in multiple languages for potential reuse, and optimization of fonts and memory requirements

The Graphics Library architecture components, display drivers, libAria APIs and demonstration applications are placed into the following locations during the installation of MPLAB Harmony:

- `/microchip/harmony/<version>/framework/gfx/hal`
- `/microchip/harmony/<version>/framework/gfx/utils`
- `/microchip/harmony/<version>/framework/gfx/libaria`
- `/microchip/harmony/<version>/apps/gfx/`

## Graphics Composer Suite Salient Features

This section provides the highlights of the new MPLAB Harmony Graphics Composer Suite.

### Description

Multiple new features have been added to the tools starting with the introduction of MPLAB Harmony v2.02.

These tools were reworked based on:

- New hardware capabilities from Microchip PIC32 products
- Numerous customer requests for updated features.

The tools include:

- New graphics import engine (GAC)
- Capability of image conversion and compression
- Hardware Abstraction Layer (HAL), support for GPU
- Entirely new graphics library (Aria User Interface Library)
- Localization font and string manager; font filtering
- Resource utilization manager
- 24-bit color (32-bit with alpha channel) and multi-layer support
- Multiple new widgets, support for primitive touch gestures
- Multiple new applications to demonstrate features (old apps will be retired)
- Tree drawing support, parent child association
- Revised WYSIWYG engine, updated accuracy and screen elements
- Revised clipping and object drawing support
- New interface for graphics events, and external / logical macros
- Integration of the Display Manager for automatic generation of display drivers

In addition to the rich features offered on introduction, the new architecture enables new capabilities that are planned in future releases.

These new features include:

- Integration of the PIC32MZ DA LCD driver (GLCD)
- Support for the PIC32MZ DA GPU library
- Mechanism for the motion/movement engine
- Mechanism for the simulation engine
- Editor/user customization of widgets
- Integrated image editor
- Touch screen gesture editing
- Memory-saving global pallet functions, optimized use of SRAM buffers

## Graphics Composer Suite Goals

This section describes the MPLAB Harmony Graphics Composer Suite goals.

### Description

The graphics tools and stack were designed with several goals in mind:

- Tight Integration Experience – Design and code generator tools are tightly integrated with the development environment for one-touch project generation
- User Experience – Libraries and tools are easy to learn and use
- Powerful User Interface Library – User interface library builds upon and expands previous capabilities to offer increased functionality
- Complete Code Generation – Can generate code for library initialization, library management, touch integration, color schemes, event handling, and screen macros
- Powerful Asset Converter – Can output several image formats, performs auto palette generation for image compression, supports run-length encoding, and supports several popular image asset formats. Also supports automatic font character inclusion and rasterization.
- Expanded Color Mode Support – The stack can manage frame buffers using 8-bit to 32-bit color
- Enhanced Resource Configuration – Tools provide the capability to manage assets more completely
- Text localization – The stack provides the capability to easily integrate international language characters into a design and seamlessly change between defined languages at run-time
- Abstract Hardware Support – Graphics controllers and accelerators should be able to be added or removed without any change to the application
- Enable future features – Including simulator and motion functions

## Graphics Composer Porting

This section describes the MPLAB Harmony Graphics Composer porting process.

### Description

Within MPLAB Harmony releases earlier than v2.02, the MPLAB Harmony Graphics Composer tool offered only a subset of the features currently provided. This feature subset included tools that were removed in the current release. These removed tools include:

- Graphics Object Library (GOL)
- Graphic Asset Converter (GAC)

Customers who must have legacy access to these tools can still find them in a parallel version of MPLAB Harmony. These tools will continue to be available within the v1.xx versions for some time.

- On update of the tools, any designs previously constructed using graphics will be offered an update. This process will port the old design to the new structure. While the process is automatic, it may take some engineering to refine items like widget compatibility. Most objects, their events, parameters and locations will be ported into the new design without further intervention.

## *Aria User Interface Library*

This section describes the Aria User Interface for the graphics capabilities and operations.

### Description

## Introduction

Introduces the Aria User Interface Library.

### Description

The Aria User Interface Library is primarily responsible for presenting a visual means of interaction between a user and an application. The library provides the building blocks to construct a complex user interface and is responsible for managing the interface once created. It is also responsible for responding to external input from users other sources and reacting appropriately. The goals of this library are to be:

- Able to provide a simple but powerful user experience
- Customizable to the needs of the application
- Light and flexible with regards to resource consumption
- Easily extensible to meet future design needs

### Definitions

**Alignment** – Indicates the placement of objects within a given bounding area

**Bounding Rectangle** – The rectangle that an object occupies in a given space

**Context** – A discrete instance of the user interface library

**Event** – An indication of some kind of occurrence that may require attention

**Layer** – Directly related to the layers offered by the Hardware Abstraction Layer. Aria layers also function as direct children to a screen. Widgets are added to layers and become part of the overall widget tree.

**Margin** – A buffer area at the edge of a bounding rectangle

**Occlusion** – The state of being completely obstructed by another entity.

**Rasterize** – The process of translating a user interface model from a logical mathematical representation into a visual image.

**Scheme** – A list of colors that can be referenced for drawing purposes

**Screen** – The root node of a widget tree. Represents a discrete configuration of layers and widgets. Can have a unique life cycle for custom memory management.

**String** – A logical array of linguistic characters

**Widget** – An abstract object that is part of a user interface

**Widget Tree** – A tree data structure of widgets that, when rendered, generates a user interface image.

### Overview

The Aria user interface library is primarily responsible for:

- HAL Configuration
- Widget Tree Management
- Event Management

- Input Handling
- Scene Rendering

**HAL Configuration**

The Aria User Interface library is context-based similar to other portions of the graphics stack. For ease of use, the library is responsible for creating and managing a HAL context internally. This releases the application from having to interact with the HAL API at all.

The context contains all of the information required to manage the state of the library. It contains the screen state, the event list, the input state, and various other settings.

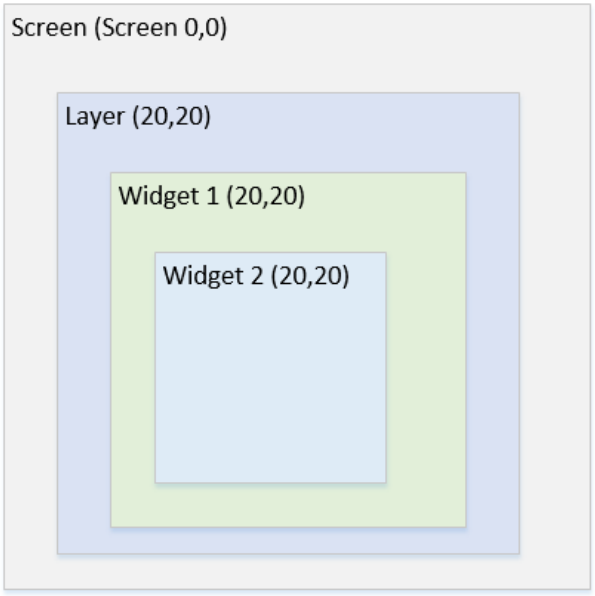
**Widget Tree Management**

The widget tree is a tree data structure comprised primarily of widgets. At its root is a screen object. Each of the screen object's direct children is a layer object. Any descendants of a layer are widgets.

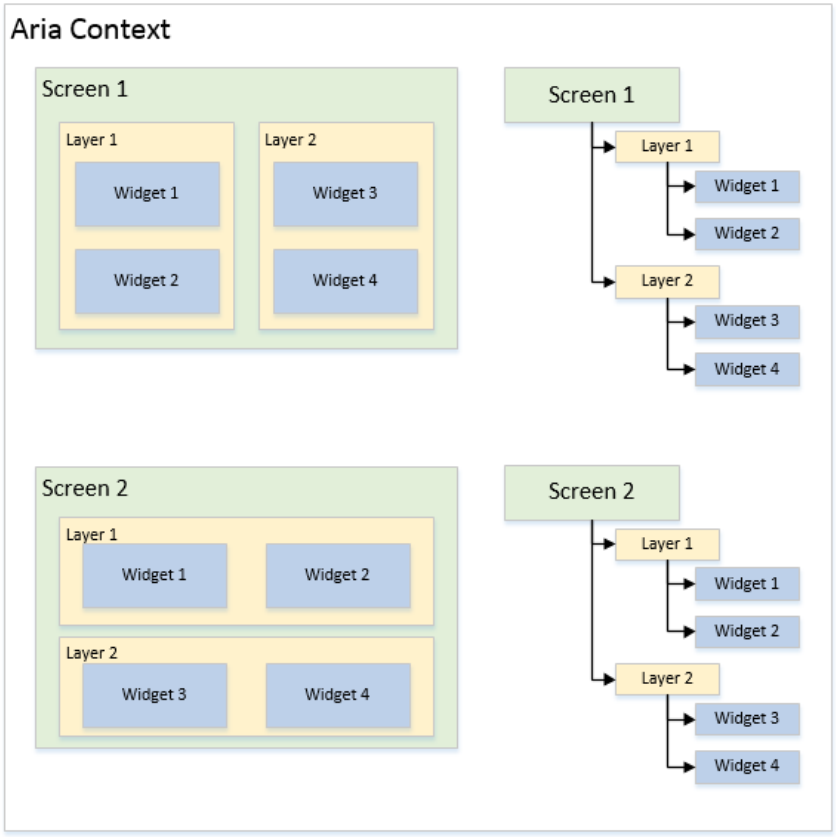
**Heterogeneous Space**

When dealing with objects in a tree it is helpful to understand objects do not live in the same coordinate space as their ancestors or descendants. Each level of the tree represents a unique area of spatial coordinates with the root coordinate space, or the screen space, being the physical space of the display device. Each space is a two dimensional Cartesian coordinate system in both the positive and negative directions.

For example, assume a widget is a child of a layer which is a child of a screen. The screen position is (0,0) in physical space. The layer position is (20,20). Widget 1 is at (20,20) and widget 2 is at (20,20). All of these coordinates aside from the screen are relative. Each widget is (20,20) offset from its parent. However, Widget 2 is not at (20,20) in physical space, it's actually (60,60). In global space each space builds on its parent, but it's entirely relative.




The following is two screen examples showing different visual representations with identical logical (tree) representations.




Using a tree to manage the logical state of a user interface provides numerous benefits.

- **Position Inheritance** – Child coordinate systems logically inherit from their parents but are not directly affected by them. Thus, if some ancestor changes position in its space, children likewise change overall position but their positions in their relative space do not change. This allows for easy manipulation of large portions of the user interface through very small changes.
- **Intelligent Rendering** – The tree structure allows for fast analysis of widget states when determining how and when to render the overall scene. Nodes in the tree track changes in the states of widgets and their descendants. This allows the library to use intelligent methods to cull portions of the user interface to avoid the processing overhead of redrawing widgets that have not changed.
- **Effect Propagation** – As with coordinate propagation, effects can also be inherited along tree branches. For instance, if some node in the tree is made invisible, all descendants are also made invisible. If a node is made partially transparent, then that transparency is propagated to all descendant nodes.

## Screens

A screen is defined as the logical root of the user interface scene. Its direct descendants are always one or more layers, as seen in the above diagram. Its size always matches the physical dimensions of the display device used.

## Life Cycle

The life cycle of a screen can be configured to better manage run-time memory usage. The relevant options are:

**Persistent** – By default screens will create their widget tree when shown and will free the memory consumed by their widget tree when they are hidden. A screen that is marked as persistent will not free their widget tree when hidden. This allows widgets in a screen to maintain their state when the screen is no longer visible. The downside is that more heap memory is consumed.

**Create At Startup** – By default screens are only created when they are shown thus keeping run-time memory usage to a minimum. However, the application may want to access widgets in a screen before it is shown for the first time. This option will cause the screen to allocate all of its memory when the screen is first added to the user interface library context.

## Orientation

Because the Hardware Abstraction Layer supports dynamic orthogonal orientation, screens can take advantage of this feature. Thus, screens have the option to set a magnitude of rotation to some factor of 90 degrees.

## Layers

User interface layers serve several functions. They function as the de-facto root parent for widgets, they directly configure hardware layers in the graphics driver, and they manage per-layer effects.

In the simplest case the hardware supports a single layer that is the same dimension as the physical display. More advanced cases may have several layers that can have unique coordinates and dimensions within the physical display space.

When a screen becomes active, it iterates over all of its layers and applies the settings of each through the appropriate HAL APIs to set up the

display state for that particular screen. Two screens may have different layer counts, layouts, and buffer counts.

## Schemes

Schemes in Aria are simply collections of colors with given names. If a scheme is assigned to a widget then that scheme will be referenced by that widget during rendering. Aria has an internal scheme that all widgets use by default in the event that a scheme is not assigned.

Below is a list of scheme colors and a description of how it is often used. There is no restriction on how a widget references a scheme. The color names are merely a recommendation.

**Scheme Colors**

- **Base** – Default area fill
- **Highlight** – Light embossing
- **Highlight Light** – Very light embossing
- **Shadow** – Dark embossing
- **Shadow Dark** – Very dark embossing
- **Foreground** – A foreground color
- **Foreground Inactive** – Foreground color when inactive
- **Foreground Disabled** – Foreground color when disabled
- **Background** – A background color, usually to differentiate from Base
- **Background Inactive** – Background when inactive
- **Background Disabled** – Background when disabled
- **Text** – Text color
- **Text Highlight** – Text color background when highlighted
- **Text Highlight Text** – Text color when highlighted
- **Text Inactive** – Text color when inactive
- **Text Disabled** – Text color when disabled

## Widgets

A widget is an abstract representation of an object in the user interface. In its most basic form it is a rectangle that is capable of drawing a border, a background color, and containing child widgets. More specific implementations extend the basic widget implementation to provide advanced functionality.

The Aria library relies heavily on function pointers to take advantage of some object oriented programming concepts like inheritance and polymorphism.

Widgets are typically created by calling their specific “new” function. For instance: “laWidget_New()” will allocate a new basic widget and return a pointer to it (similar to calling new in C++). Calling this function will automatically initialize the widget by calling the constructor for that widget. Deleting widgets is done through the use of the function “laWidget_Delete()”.

Widgets can then be added to layers or other widgets as desired.

**Edit Widgets**

Edit widgets are a special class of widget that inherits from the EditWidget base implementation instead of Widget. These widgets are capable of becoming the active “edit” widget which means that they will receive any edit events raised by a widget capable of issuing edit events, such as a key pad.

**Widget Implementations**

The following are descriptions of the widgets offered by Aria:

- Button
  - Standard button type widget.
  - Can have text and image icon.
  - Has a toggle mode.
- Check Box
  - Standard check box widget.
  - Has built in image for checked and unchecked state.
  - Can use custom image for checked and unchecked state.
- Circle
  - Widget that draws a circle
- Draw Surface
  - Widget that has a callback during its paint loop
  - Allows application to make raw HAL draw calls during Aria’s paint loop
- Gradient
  - Widget that draws linearly interpolated gradient for its background
  - Can use as a parent for other widgets to achieve custom backgrounds
- Group Box

    - Widget that functions as a decorated container
    - Offers a line border and a horizontally aligned title
- Image
    - Widget that draws an image
    - Image is clipped to the bounds of the widget.
    - Image can be vertically or horizontally aligned to the bounds of the widget
- Image Sequence
    - Widget that functions as an automatic image slideshow renderer
    - Can add a sequence of widgets and a list of time delays
    - Can automatically cycle through list of images without application input
- Key Pad
    - A grid of button widgets
    - Buttons can be configured to send edit events to the library edit API
- Label
    - Widget that draws a string
    - Can be aligned vertically and horizontally
- Line
    - Widget that draws a line between two specified coordinates
- List
    - A list box of strings
    - Strings can have icons
    - Can be configured to have single, sequential, or multi-selection state
- List Wheel
    - A rotating wheel of strings
    - Cycles seamlessly through the list
    - Responds to drag input
- Panel Widget
    - Panels are containers of other widgets, including daughter panels, in support of a parent-child tree of widgets, with the Panel widget as the parent
- Progress Bar
    - Widget that fills in a direction based on a given percentage
- Radio Button
    - A button that can belong to a group of radio buttons
    - Only one button in group can be selected at any one time
- Rectangle
    - Widget that draws a rectangle
- Scroll Bar
    - A scroll bar that has a configurable scroll range.
    - Normally embedded in other widgets like the list box
- Slider
    - A widget that slides between a min and max value
- Text Field
    - A field of text that can be modified by edit event inputs
- Touch Text
    - A widget that draws lines based on input events
    - Helps to demonstrate input functionality
- Window
    - A container that can be decorated with a title bar
    - Title bar can have title text and an icon

## Event Management

The Aria state maintains an internal list of events that must get serviced frequently. This is done by called by “laUpdate()”. This is known as the ‘update loop’.

## Input Handling

The user interface library has no knowledge of existing hardware but it must provide the means for the user to interact with the scene. Aria thus provides several generic APIs to allow any source to inject input events into the system. These events are stored in the internal event list and are handled during the next update phase.

## Scene Rendering

The widget tree is a logical representation of the state of the user interface. The library must be capable of transforming this information into a visual representation that can be sent to the graphics display. The actual rendering is handled by the HAL. The individual widgets contain the algorithms necessary to render themselves but Aria is responsible for telling the widgets when to render themselves. This is known as the ‘paint loop’

The library is responsible for evaluating the widget tree to detect widgets that indicate invalid visual states and managing the redraw. It is essential that widgets only redraw when necessary to avoid needlessly consuming processing resources. It is also important that the library not attempt to draw too much at once as that may starve the rest of the application.

## How to Use the Library

```
// initialize the HAL layer
GFX_Initialize();

// initialize the user interface library
laInitialize();

// create a ui context and set active
laContext* uiContext;

iuContext = laContext_Create(0, 0, 0, GFX_COLOR_MODE_RGB_565, NULL);
laContext_SetActive(uiContext);

// create a screen
laScreen* screen;

screen = laScreen_New(LA_FALSE, LA_FALSE, &screenCreate);

// add screen to context
laContext_AddScreen(screen);

// this would be done inside a function called “screenCreate()”
// create layer
laLayer* layer0 = laLayer_New();
laWidget_SetPosition((laWidget*)layer0, 0, 0);
laWidget_SetSize((laWidget*)layer0, 480, 272);

// create a buffer in the layer
laLayer_SetBufferCount(layer0, 1);

// set the layer to the screen
laScreen_SetLayer(screen, 0, layer0);

// create a child widget
laButtonWidget* ButtonWidget1 = laButtonWidget_New();
laWidget_SetPosition((laWidget*)ButtonWidget1, 411, 201);
laWidget_SetSize((laWidget*)ButtonWidget1, 60, 60);
laWidget_SetLocalRedraw((laWidget*)ButtonWidget1, LA_TRUE);
laWidget_SetDrawBackground((laWidget*)ButtonWidget1, LA_FALSE);
laWidget_SetBorderType((laWidget*)ButtonWidget1, LA_WIDGET_BORDER_NONE);
laButtonWidget_SetPressedOffset(ButtonWidget1, 0);
laButtonWidget_SetReleasedEventCallback(ButtonWidget1, &ButtonWidget1_ReleasedEvent);

// add child to parent (layer 0)
laWidget_AddChild((laWidget*)layer0, (laWidget*)ButtonWidget1);

// do this inside application update loop
// update HAL
GFX_Update()

// set ui context as active
laContext_SetActive(uiContext);

// update context (argument is update time in ms)
laUpdate(0);
```

## Aria User Interface Library Interface

### a) Functions

| | Name | Description |
|---|---|---|
| | laContext_AddScreen | Add screen to the list of screens in the current context |
| | laContext_Create | Creates an instance of the Aria user interface library |
| | laContext_Destroy | Destroys an Aria instance |
| | laContext_GetActive | Returns the current active context. |
| | laContext_GetActiveScreen | Returns the active screen of the current context |
| | laContext_GetActiveScreenIndex | Return the index of the active screen |
| | laContext_GetColorMode | Returns the color mode of the current context |
| | laContext_GetDefaultScheme | Returns the pointer to the default scheme of the current context |
| | laContext_GetEditWidget | Gets the widget that is currently recieving all widget edit events. |
| | laContext_GetFocusWidget | Return a pointer to the widget in focus |
| | laContext_GetPremptionLevel | Returns the preemption level for the screen |
| | laContext_GetScreenRect | Returns the display rectangle structure of the physical display |
| | laContext_GetStringLanguage | Returns the language index of the current context |
| | laContext_GetStringTable | Get a pointer to the GFXU_StringTableAsset structure that maintains the strings, associated fonts, etc |
| | laContext_HideActiveScreen | Hide the active screen |
| | laContext_RedrawAll | Forces the library to redraw the currently active screen in its entirety. |
| | laContext_RemoveScreen | Remove the specified screen from the list of screens in the current context |
| | laContext_SetActive | Make the specified context active |
| | laContext_SetActiveScreen | Change the active screen to the one specified by the index argument |
| | laContext_SetActiveScreenChangedCallback | Set the callback function pointer when the screen change event occurs |
| | laContext_SetEditWidget | Sets the currently active edit widget. |
| | laContext_SetFocusWidget | Set into focus the widget specified as the argument |
| | laContext_SetLanguageChangedCallback | Set the callback function pointer when the language change event occurs |
| | laContext_SetStringLanguage | Set the language index of the current context |
| | laContext_SetStringTable | Set the StringTable pointer to the specified new StringTableAsset structure |
| | laContext_Update | Runs the update loop for a library instance. |
| | laEditWidget_Accept | This is function laEditWidget_Accept. |
| | laEditWidget_Append | This is function laEditWidget_Append. |
| | laEditWidget_Backspace | This is function laEditWidget_Backspace. |
| | laEditWidget_Clear | This is function laEditWidget_Clear. |
| | laEditWidget_EndEdit | This is function laEditWidget_EndEdit. |
| | laEditWidget_Set | This is function laEditWidget_Set. |
| | laEditWidget_StartEdit | This is function laEditWidget_StartEdit. |
| | laList_Assign | Assignes a new pointer to an index in the list |
| | laList_Clear | Removes all nodes from a given list |
| | laList_Copy | Creates a duplicate of an existing list |
| | laList_Create | Initializes a new linked list |
| | laList_Destroy | Removes all nodes from a given list and frees the data of each node |
| | laList_Find | Retrieves the index of a value from the list |
| | laList_Get | Retrieves a value from the list |
| | laList_InsertAt | Inserts an item into a list at a given index. All existing from index are shifted right one place. |
| | laList_PopBack | Removes the last value from the list |
| | laList_PopFront | Removes the first value from the list |
| | laList_PushBack | Pushes a new node onto the back of the list |
| | laList_PushFront | Pushes a new node onto the front of the list |

| | | |
|---|---|---|
| | laList_Remove | Removes an item from the list |
| | laList_RemoveAt | Removes an item from the list at an index |
| | laString_Allocate | Attempts to resize the local data buffer for a string. |
| | laString_Append | Appends a string onto the end of another string |
| | laString_Capacity | Returns the capacity of a string |
| | laString_CharAt | Extracts the code point for the character in a string at a given index. |
| | laString_Clear | Sets a string's length to zero and its string table reference to NULL. Does not free any associated data and preserves capacity. |
| | laString_Compare | Compares two string objects |
| | laString_CompareBuffer | Compares a string object and a GFXU_CHAR* buffer |
| | laString_Copy | Copies the values from one string into another |
| | laString_CreateFromBuffer | Creates a string object from a GFXU_CHAR buffer and a font asset pointer |
| | laString_CreateFromCharBuffer | Creates a string object from a const char* buffer and a font asset pointer. This method provides compatibility with standard c-style strings. Input string will be converted from 8-bit with to 32-bit width. |
| | laString_CreateFromID | Creates a string object that simply references a string in the string table. |
| | laString_Delete | Deletes all memory associated with a string object |
| | laString_Destroy | Destroys a string object. This frees the strings internal data buffer, if it exists, sets its string table reference to null, and clears all supporting attributes. |
| | laString_Draw | Wrapper around GFX Utility string draw function for Aria user interface library. Internal use only. |
| | laString_ExtractFromTable | Extracts a read-only string from the string table into a modifiable string object. This relies on the active context to indicate which string table to reference as well as which language entry to extract. |
| | laString_GetCharIndexAtPoint | Given an offset in pixels returns the corresponding character index. |
| | laString_GetCharOffset | Returns the offset of a given character index in pixels. |
| | laString_GetCharWidth | Given a character index, gets the width of that character. Only accurate if the string has a font associated with it and that font contains all the characters in the string in question. |
| | laString_GetHeight | Returns the height of a string by referencing its associated font asset data. |
| | laString_GetRect | Calculates the rectangle for a given string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. |
| | laString_Initialize | Initializes a string struct to default |
| | laString_Insert | Inserts a string into another string at a given index |
| | laString_Length | Calculates the length of a string in characters |
| | laString_New | Allocates a memory for a new string |
| | laString_ReduceLength | Reduces the length of a string. This simply slides the null terminator to the left and does not affect the string's capacity value. |
| | laString_Set | Attempts to set the local data buffer of a string to an input buffer |
| | laString_SetCapacity | Attempts to adjust the capacity of a string |
| | laString_ToCharBuffer | Extracts the data buffer from a string and copies it into the provided buffer argument. |
| | laButtonWidget_GetHAlignment | Gets the horizontal alignment setting for a button |
| | laButtonWidget_GetImageMargin | Gets the distance between the icon and the text |
| | laButtonWidget_GetImagePosition | Gets the position of the button icon |
| | laButtonWidget_GetPressed | Gets the pressed state of a button |
| | laButtonWidget_GetPressedEventCallback | Gets the callback associated with the button pressed event |
| | laButtonWidget_GetPressedImage | Gets the pressed image asset pointer for a button |
| | laButtonWidget_GetPressedOffset | Gets the offset of the button internals when pressed |
| | laButtonWidget_GetReleasedEventCallback | Gets the callback for the button released event |
| | laButtonWidget_GetReleasedImage | Gets the currently used released icon |
| | laButtonWidget_GetText | Gets the text for a button. If the button's string has local data then a duplicate of the string will be allocated. The caller is responsible for managing the memory for the duplicated string. If the button string is a string table reference then only the reference ID is copied. |

| | | |
|---|---|---|
| | laButtonWidget_GetToggleable | Gets the value of this button's toggle flag |
| | laButtonWidget_GetVAlignment | Gets the vertical alignment setting for a button |
| | laButtonWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laButtonWidget_SetHAlignment | Sets the horizontal alignment value for a button |
| | laButtonWidget_SetImageMargin | Sets the distance between the icon and text |
| | laButtonWidget_SetImagePosition | Sets the position of the button icon |
| | laButtonWidget_SetPressed | Sets the pressed state for a button. |
| | laButtonWidget_SetPressedEventCallback | Sets the pressed event callback for the button |
| | laButtonWidget_SetPressedImage | Sets the image to be used as a pressed icon |
| | laButtonWidget_SetPressedOffset | Sets the offset of the button internals when pressed |
| | laButtonWidget_SetReleasedEventCallback | Sets the callback for the button released event |
| | laButtonWidget_SetReleasedImage | Sets the image to be used as the released icon |
| | laButtonWidget_SetText | Sets the text for a button. If the input string has local data then the data will be copied into the button's local string, causing a memory allocation. If the input string is a string table reference then only the reference will be copied. The input string can be safely modified and the button string will not be affected. |
| | laButtonWidget_SetToggleable | Enables the toggle mode for a button. When pressed, toggle buttons will stay down until pressed again. |
| | laButtonWidget_SetVAlignment | Sets the vertical alignment for a button |
| | laCheckBoxWidget_GetChecked | Gets the checked state of the check box |
| | laCheckBoxWidget_GetCheckedEventCallback | Gets the checked event callback |
| | laCheckBoxWidget_GetCheckedImage | Gets the checked image of the check box |
| | laCheckBoxWidget_GetHAlignment | Gets the horizontal alignment of the check box |
| | laCheckBoxWidget_GetImageMargin | Gets the distance between the image and the text |
| | laCheckBoxWidget_GetImagePosition | Gets the image position of the check box |
| | laCheckBoxWidget_GetText | Gets a copy of the checkbox text. If the text has local data the data will be duplicated. The caller is responsible for managing the memory as appropriate. |
| | laCheckBoxWidget_GetUncheckedEventCallback | Gets the unchecked event callback |
| | laCheckBoxWidget_GetUncheckedImage | Gets the unchecked image of the check box |
| | laCheckBoxWidget_GetVAlignment | Gets the vertical alignment of the check box |
| | laCheckBoxWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laCheckBoxWidget_SetChecked | Sets the checked state of the check box |
| | laCheckBoxWidget_SetCheckedEventCallback | Sets the checked event callback |
| | laCheckBoxWidget_SetCheckedImage | Sets the checked image of the check box |
| | laCheckBoxWidget_SetHAlignment | Sets the horizontal alignment of the check box. |
| | laCheckBoxWidget_SetImagePosition | Sets the image position of the check box |
| | laCheckBoxWidget_SetText | Sets the checkbox text to the input string. If the string has local data the data will be duplicated and copied to the checkboxes internal string. |
| | laCheckBoxWidget_SetUncheckedEventCallback | Sets the unchecked event callback |
| | laCheckBoxWidget_SetUncheckedImage | Sets the unchecked image of the check box |
| | laCheckBoxWidget_SetVAlignment | Sets the vertical alignment of the check box |
| | laCircleWidget_GetOrigin | Gets the origin coordiates of a circle widget |
| | laCircleWidget_GetRadius | Gets the radius of a circle widget |
| | laCircleWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laCircleWidget_SetOrigin | Sets the origin coordiates of a circle widget |
| | laCircleWidget_SetRadius | Sets the radius of a circle widget |
| | laDraw_1x2BevelBorder | Internal utility function to draw a 1x2 bevel border |
| | laDraw_2x1BevelBorder | Internal utility function to draw a 2x1 bevel border |
| | laDraw_2x2BevelBorder | Internal utility function to draw a 2x2 bevel border |
| | laDraw_LineBorder | Internal utility function to draw a basic line border |

| | | |
|---|---|---|
| | laDrawSurfaceWidget_GetDrawCallback | Returns the pointer to the currently set draw callback. |
| | laDrawSurfaceWidget_New | Allocates memory for a new DrawSurface widget. |
| | laDrawSurfaceWidget_SetDrawCallback | Sets the draw callback pointer for the draw surface widget. |
| | laEvent_AddEvent | Add the mentioned event callback to the list of events maintained by the current context |
| | laEvent_ClearList | Clear the event list maintained by the current context. |
| | laEvent_GetCount | Returns the number of events listed in the current context |
| | laEvent_ProcessEvents | Processes the screen change as well as touch events |
| | laEvent_SetFilter | Set callback pointer for current context filter event |
| | laGradientWidget_GetDirection | Gets the gradient direction value for this widget. |
| | laGradientWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laGradientWidget_SetDirection | Sets the gradient direction value for this widget. |
| | laGroupBoxWidget_GetAlignment | Gets the horizontal alignmnet for the group box title text |
| | laGroupBoxWidget_GetText | Gets the text value for the group box. |
| | laGroupBoxWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laGroupBoxWidget_SetAlignment | Sets the alignment for the group box title text |
| | laGroupBoxWidget_SetText | Sets the text value for the group box. |
| | laImageSequenceWidget_GetImage | Gets the image asset pointer for an entry. |
| | laImageSequenceWidget_GetImageChangedEventCallback | Gets the image changed event callback pointer. |
| | laImageSequenceWidget_GetImageCount | Gets the number of image entries for this widget. |
| | laImageSequenceWidget_GetImageDelay | Gets the image delay for an entry. |
| | laImageSequenceWidget_GetImageHAlignment | Gets the horizontal alignment for an image entry |
| | laImageSequenceWidget_GetImageVAlignment | Sets the vertical alignment for an image entry |
| | laImageSequenceWidget_GetRepeat | Indicates if the widget will repeat through the image entries. |
| | laImageSequenceWidget_IsPlaying | Indicates if the widget is currently cycling through the image entries. |
| | laImageSequenceWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laImageSequenceWidget_Play | Starts the widget automatically cycling through the image entries. |
| | laImageSequenceWidget_Rewind | Resets the current image sequence display index to zero. |
| | laImageSequenceWidget_SetImage | Sets the image asset pointer for an entry. |
| | laImageSequenceWidget_SetImageChangedEventCallback | Sets the image changed event callback pointer. This callback is called whenever the active display index is changed. |
| | laImageSequenceWidget_SetImageCount | Sets the number of image entries for this widget. An image entry that is null will show nothing. |
| | laImageSequenceWidget_SetImageDelay | Sets the image delay for an entry. |
| | laImageSequenceWidget_SetImageHAlignment | Sets the horizontal alignment for an image entry. |
| | laImageSequenceWidget_SetImageVAlignment | Sets the vertical alignment value for an image entry |
| | laImageSequenceWidget_SetRepeat | Sets the repeat flag for the widget |
| | laImageSequenceWidget_ShowImage | Sets the active display index to the indicated value. |
| | laImageSequenceWidget_ShowNextImage | Sets the active display index to the next index value. |
| | laImageSequenceWidget_ShowPreviousImage | Sets the active display index to the previous index value. |
| | laImageSequenceWidget_Stop | Stops the widget from automatically cycling through the image entries. |
| | laImageWidget_GetHAlignment | Gets the image horizontal alignment value. |
| | laImageWidget_GetImage | Gets the image asset pointer for the widget. |
| | laImageWidget_GetVAlignment | Gets the image vertical alignment value. |
| | laImageWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laImageWidget_SetHAlignment | Sets the image horizontal alignment value. |
| | laImageWidget_SetImage | Sets the image asset pointer for the widget. |
| | laImageWidget_SetVAlignment | Sets the image vertical alignment value. |
| | laInput_GetEnabled | Returns the input enabled status of the current context |

| | | |
|---|---|---|
| | laInput_InjectTouchDown | Register and track the touch down event and queue it for handling by associated widgets |
| | laInput_InjectTouchMoved | Register and track the touch moved event and queue it for handling by associated widgets |
| | laInput_InjectTouchUp | Register and track the touch up event and queue it for handling by associated widgets |
| | laInput_SetEnabled | Sets the input status of the current context with the specified input argument |
| | laKeyPadWidget_GetKeyAction | Gets the key pad cell action for a cell at row/column |
| | laKeyPadWidget_GetKeyClickEventCallback | Gets the current key click event callback pointer |
| | laKeyPadWidget_GetKeyDrawBackground | Gets the background type for a key pad cell at row/column |
| | laKeyPadWidget_GetKeyEnabled | Gets the enabled flag for a cell at a given row/column |
| | laKeyPadWidget_GetKeyImageMargin | Gets the key pad cell image margin value |
| | laKeyPadWidget_GetKeyImagePosition | Gets the image position for a key pad cell |
| | laKeyPadWidget_GetKeyPressedImage | Gets the pressed icon image asset pointer for the display image for a key pad cell |
| | laKeyPadWidget_GetKeyReleasedImage | Gets the released icon image asset pointer for the display image for a key pad cell |
| | laKeyPadWidget_GetKeyText | Returns a copy of the display text for a given cell at row/column |
| | laKeyPadWidget_GetKeyValue | Gets the edit text value for a given key pad cell. |
| | laKeyPadWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laKeyPadWidget_SetKeyAction | Sets the cell action type for a key pad cell at row/column |
| | laKeyPadWidget_SetKeyClickEventCallback | Sets the current key click event callback pointer |
| | laKeyPadWidget_SetKeyEnabled | Sets the enabled flag for a cell at the given row/column |
| | laKeyPadWidget_SetKeyImageMargin | Sets the key pad cell image margin value for a given cell at row/column |
| | laKeyPadWidget_SetKeyImagePosition | |
| | laKeyPadWidget_SetKeyPressedImage | Sets the pressed icon image asset pointer for a key pad cell |
| | laKeyPadWidget_SetKeyReleasedImage | Sets the released icon image asset pointer for a key pad cell |
| | laKeyPadWidget_SetKeyText | Sets the display text for a given cell at row/column |
| | laKeyPadWidget_SetKeyValue | Sets the edit value for a given key pad cell. |
| | laLabelWidget_GetHAlignment | Gets the text horizontal alignment value. |
| | laLabelWidget_GetText | Gets the text value for the label. |
| | laLabelWidget_GetVAlignment | Gets the current vertical text alignment |
| | laLabelWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laLabelWidget_SetHAlignment | Sets the text horizontal alignment value |
| | laLabelWidget_SetText | Sets the text value for the label. |
| | laLabelWidget_SetVAlignment | Sets the vertical text alignment value |
| | laLayer_Delete | Destructor for the layer object |
| | laLayer_GetAlphaAmount | Get's the amount of alpha blending for a given layer |
| | laLayer_GetAlphaEnable | Gets the layer alpha enable flag |
| | laLayer_GetBufferCount | Return the buffer count for the current layer |
| | laLayer_GetMaskColor | Returns the mask color value for the current layer |
| | laLayer_GetMaskEnable | Gets the layer mask enable flag |
| | laLayer_GetVSync | Gets the layer's vsync flag setting |
| | laLayer_New | Constructor for a new layer |
| | laLayer_SetAlphaAmount | Set's the amount of alpha blending for a given layer |
| | laLayer_SetAlphaEnable | Sets the layer alpha enable flag to the specified value |
| | laLayer_SetBufferCount | Set the buffer count for the current layer to the specified value |
| | laLayer_SetMaskColor | Set the mask color value for the current layer to the specified value |
| | laLayer_SetMaskEnable | Sets the layer mask enable flag to the specified value |
| | laLayer_SetVSync | Sets the layer's vsync flag. |
| | laLineWidget_GetEndPoint | Gets the coordinates for the second point of the line. |
| | laLineWidget_GetStartPoint | Gets the coordinates for the first point of the line. |

| | | |
|---|---|---|
| | laLineWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laLineWidget_SetEndPoint | Sets the cooridnate for the second point of the line |
| | laLineWidget_SetStartPoint | Sets the cooridnate for the first point of the line |
| | laListWheelWidget_AppendItem | Appends a new item entry to the list. The initial value of the item will be empty. |
| | laListWheelWidget_GetAlignment | Gets the horizontal alignment for the list widget |
| | laListWheelWidget_GetIconMargin | Gets the icon margin value for the list wheel widget |
| | laListWheelWidget_GetIconPosition | Sets the icon position for the list wheel widget. |
| | laListWheelWidget_GetItemCount | Gets the number of items currently contained in the list |
| | laListWheelWidget_GetItemIcon | Gets the pointer to the image asset for the icon for the item at the given index. |
| | laListWheelWidget_GetItemText | Gets the text value for an item in the list. |
| | laListWheelWidget_GetSelectedItem | Returns the index of the currently selected item. |
| | laListWheelWidget_GetSelectedItemChangedEventCallback | Gets the callback for the item selected changed event |
| | laListWheelWidget_InsertItem | Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right. |
| | laListWheelWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laListWheelWidget_RemoveAllItems | Attempts to remove all items from the list. |
| | laListWheelWidget_RemoveItem | Attempts to remove an item from the list. |
| | laListWheelWidget_SelectNextItem | Attempts to move the selected item index to the next item in the list. |
| | laListWheelWidget_SelectPreviousItem | Attempts to move the selected item index to the previous item in the list. |
| | laListWheelWidget_SetAlignment | Sets the horizontal alignment mode for the list widget. |
| | laListWheelWidget_SetIconMargin | Sets the icon margin value for the list widget. |
| | laListWheelWidget_SetIconPosition | Sets the icon position for the list wheel widget |
| | laListWheelWidget_SetItemIcon | Sets the icon pointer for a given index. |
| | laListWheelWidget_SetItemText | Sets the text value for an item in the list. |
| | laListWheelWidget_SetSelectedItem | Attempts to set the selectedi tem index |
| | laListWheelWidget_SetSelectedItemChangedEventCallback | |
| | laListWidget_AppendItem | Appends a new item entry to the list. The initial value of the item will be empty. |
| | laListWidget_DeselectAll | Attempts to set all item states as not selected. |
| | laListWidget_GetAlignment | Gets the horizontal alignment for the list widget |
| | laListWidget_GetAllowEmptySelection | Returns true of the list allows an empty selection set |
| | laListWidget_GetFirstSelectedItem | Returns the lowest selected item index. |
| | laListWidget_GetIconMargin | Gets the icon margin value for the list widget |
| | laListWidget_GetIconPosition | Gets the icon position for the list |
| | laListWidget_GetItemCount | Gets the number of items currently contained in the list |
| | laListWidget_GetItemIcon | Gets the pointer to the image asset for the icon for the item at the given index. |
| | laListWidget_GetItemSelected | Returns true if the item at the given index is currently selected. |
| | laListWidget_GetItemText | Gets the text value for an item in the list. |
| | laListWidget_GetLastSelectedItem | Returns the highest selected item index. |
| | laListWidget_GetSelectedItemChangedEventCallback | Gets the callback for the item selected changed event |
| | laListWidget_GetSelectionCount | Returns the number of selected items in the list. |
| | laListWidget_GetSelectionMode | Gets the selection mode for the list |
| | laListWidget_InsertItem | Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right. |
| | laListWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laListWidget_RemoveAllItems | Attempts to remove all items from the list. |
| | laListWidget_RemoveItem | Attempts to remove an item from the list. |
| | laListWidget_SelectAll | Attempts to set all item states to selected. |

| | | |
|---|---|---|
| | laListWidget_SetAlignment | Sets the horizontal alignment mode for the list widget. |
| | laListWidget_SetAllowEmptySelection | Configures the list to allow an empty selection set. |
| | laListWidget_SetIconMargin | Sets the icon margin value for the list widget. |
| | laListWidget_SetIconPosition | Sets the icon position for the list widget |
| | laListWidget_SetItemIcon | Sets the icon pointer for a given index. |
| | laListWidget_SetItemSelected | Attempts to set the item at idx as selected. |
| | laListWidget_SetItemText | Sets the text value for an item in the list. |
| | laListWidget_SetItemVisible | |
| | laListWidget_SetSelectedItemChangedEventCallback | Sets the callback for the item selected changed event |
| | laListWidget_SetSelectionMode | Set the list selection mode |
| | laListWidget_ToggleItemSelected | Attempts to toggle the selected state of the item at idx. |
| | laProgressBarWidget_GetDirection | Gets the fill direction value for a progress bar widget |
| | laProgressBarWidget_GetValue | Gets the percentage value for a progress bar. |
| | laProgressBarWidget_GetValueChangedEventCallback | Gets the currently set value changed event callback. |
| | laProgressBarWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laProgressBarWidget_SetDirection | Sets the fill direction for a progress bar widget |
| | laProgressBarWidget_SetValue | Sets the percentage value for a progress bar. Valid values are 0 - 100. |
| | laProgressBarWidget_SetValueChangedCallback | Sets the desired value changed event callback pointer |
| | laRadioButtonGroup_AddButton | Add a button widget to the button list of the selected Radio button group. |
| | laRadioButtonGroup_Create | This function creates a GFX_GOL_RADIOBUTTON group with the provided button list. |
| | laRadioButtonGroup_Destroy | This function destroys the GFX_GOL_RADIOBUTTON group |
| | laRadioButtonGroup_RemoveButton | Remove a button widget to the button list of the selected Radio button group. |
| | laRadioButtonGroup_SelectButton | Select the button widget specified from the button list for the Radio button group. |
| | laRadioButtonWidget_GetDeselectedEventCallback | Gets the current radio button deselected event callback |
| | laRadioButtonWidget_GetGroup | Returns the pointer to the currently set radio button group. |
| | laRadioButtonWidget_GetHAlignment | Gets the horizontal alignment setting for a button |
| | laRadioButtonWidget_GetImageMargin | Gets the distance between the icon and the text |
| | laRadioButtonWidget_GetImagePosition | Gets the current image position setting for the radio button |
| | laRadioButtonWidget_GetSelected | Returns true if this radio button is currently selected |
| | laRadioButtonWidget_GetSelectedEventCallback | Gets the current radio button selected event callback |
| | laRadioButtonWidget_GetSelectedImage | Gets the selected image asset pointer for a button |
| | laRadioButtonWidget_GetText | Gets the text value for the button. |
| | laRadioButtonWidget_GetUnselectedImage | Gets the image asset pointer currently used as the unselected icon |
| | laRadioButtonWidget_GetVAlignment | Sets the vertical alignment for a button |
| | laRadioButtonWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laRadioButtonWidget_SetDeselectedEventCallback | Sets the deselected callback pointer |
| | laRadioButtonWidget_SetHAlignment | Sets the horizontal alignment value for a button |
| | laRadioButtonWidget_SetImagePosition | Sets the image relative position setting for the radio button |
| | laRadioButtonWidget_SetSelected | Sets this button as selected. |
| | laRadioButtonWidget_SetSelectedEventCallback | Sets the radio button selected event callback |
| | laRadioButtonWidget_SetSelectedImage | Sets the image to be used as a selected icon |
| | laRadioButtonWidget_SetText | Sets the text value for the button. |
| | laRadioButtonWidget_SetUnselectedImage | Sets the asset pointer for the radio button's unselected image icon |
| | laRadioButtonWidget_SetVAlignment | Sets the vertical alignment for a button |
| | laRectangleWidget_GetThickness | Gets the rectangle border thickness setting |
| | laRectangleWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |

| | | |
|---|---|---|
| | laScheme_Initialize | Initialize the scheme to the default values as per the specified color mode. |
| | laScreen_Delete | Frees all memory for all layers and widgets for this screen |
| | laScreen_GetHideEventCallback | Returns the hide call back event function pointer for the specified screen |
| | laScreen_GetLayerIndex | Returns the index of the layer for the screen specified. |
| | laScreen_GetOrientation | Returns the orientation object associated with the specified screen |
| | laScreen_GetShowEventCallback | Returns the show call back event function pointer for the specified screen |
| | laScreen_Hide | Hide the currently active screen<br>This function has been deprecated in favor of laContext_SetActiveScreen |
| | laScreen_New | Create a new screen, initialize it to the values specified. |
| | laScreen_SetHideEventCallback | Set the hide call back event function pointer for the specified screen |
| | laScreen_SetLayer | Assigns the provided layer pointer to the screen at the given index<br>This function has been deprecated in favor of laContext_SetActiveScreen |
| | laScreen_SetOrientation | Sets the orientation object to the specified screen |
| | laScreen_SetShowEventCallback | Set the show call back event function pointer for the specified screen |
| | laScreen_Show | Make the specified screen active and show it on the display |
| | laScrollBarWidget_GetExtentValue | Gets the current scroll bar extent value |
| | laScrollBarWidget_GetMaxinumValue | Gets the maximum scroll value |
| | laScrollBarWidget_GetOrientation | Gets the orientation value for the scroll bar |
| | laScrollBarWidget_GetScrollPercentage | Gets the current scroll value as a percentage |
| | laScrollBarWidget_GetScrollValue | Gets the current scroll value |
| | laScrollBarWidget_GetStepSize | Gets the current discreet step size |
| | laScrollBarWidget_GetValueChangedEventCallback | Gets the current value changed callback function pointer |
| | laScrollBarWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laScrollBarWidget_SetExtentValue | Sets the scroll bar extent value |
| | laScrollBarWidget_SetMaximumValue | Sets the maximum scroll value |
| | laScrollBarWidget_SetOrientation | Sets the orientation value of the scroll bar |
| | laScrollBarWidget_SetScrollPercentage | Sets the current scroll value using a percentage. Percentage should be a value from 0 - 100 |
| | laScrollBarWidget_SetScrollValue | Sets the current scroll value |
| | laScrollBarWidget_SetStepSize | Sets the current step size |
| | laScrollBarWidget_SetValueChangedEventCallback | Sets the value changed event callback pointer |
| | laScrollBarWidget_StepBackward | Moves the scroll value back by the current step size |
| | laScrollBarWidget_StepForward | Moves the scroll value forward by the current step size |
| | laSliderWidget_GetGripSize | Gets the current grip size of the slider |
| | laSliderWidget_GetMaxinumValue | Gets the maximum value for the slider |
| | laSliderWidget_GetMininumValue | Gets the minimum value for the slider |
| | laSliderWidget_GetOrientation | Gets the orientation value for the slider |
| | laSliderWidget_GetSliderPercentage | Gets the slider value as a percentage |
| | laSliderWidget_GetSliderValue | Gets the current slider value |
| | laSliderWidget_GetValueChangedEventCallback | Gets the current value changed event callback pointer |
| | laSliderWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laSliderWidget_SetGripSize | Sets the grip size of the slider |
| | laSliderWidget_SetMaximumValue | Sets the maximum value for the slider |
| | laSliderWidget_SetMinimumValue | Sets the minimum value for the slider |
| | laSliderWidget_SetOrientation | |
| | laSliderWidget_SetSliderPercentage | Sets the slider value using a percentage. Value must be from 0 - 100. |
| | laSliderWidget_SetSliderValue | Sets the current slider value |
| | laSliderWidget_SetValueChangedEventCallback | Sets the value changed event callback pointer |
| | laSliderWidget_Step | Moves the slider by a given amount |

| | | |
|---|---|---|
| | laTextFieldWidget_GetAlignment | Gets the text horizontal alignment value. |
| | laTextFieldWidget_GetCursorDelay | Gets the current cursor delay. |
| | laTextFieldWidget_GetCursorEnabled | Gets the cursor enabled value |
| | laTextFieldWidget_GetCursorPosition | Gets the current edit cursor position |
| | laTextFieldWidget_GetText | Gets the text value for the box. |
| | laTextFieldWidget_GetTextChangedEventCallback | Gets the current text changed event callback pointer |
| | laTextFieldWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laTextFieldWidget_SetAlignment | Sets the text horizontal alignment value |
| | laTextFieldWidget_SetCursorDelay | Sets the cursor delay value |
| | laTextFieldWidget_SetCursorEnabled | Sets the cursor enabled value flag |
| | laTextFieldWidget_SetCursorPosition | Sets the position of the cursor |
| | laTextFieldWidget_SetText | Sets the text value for the box. |
| | laTextFieldWidget_SetTextChangedEventCallback | Sets the text changed event callback pointer |
| | laTouchTest_AddPoint | Adds a point to the touch test widget. The point will then be displayed. |
| | laTouchTest_ClearPoints | Clears all of the existing touch points |
| | laTouchTestWidget_GetPointAddedEventCallback | Gets the current point added event callback |
| | laTouchTestWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laTouchTestWidget_SetPointAddedEventCallback | Sets the point added event callback |
| | laUtils_ArrangeRectangle | Calculates the position of a rectangle within the given bounds and in accordance with the given parameters. A use case for this is when an image and a text rectangle must be arranged in a button box. This version of the algorithm will calculate the location of the image rectangle. |
| | laUtils_ArrangeRectangleRelative | Calculates the position of a rectangle within the given bounds and in accordance with the given parameters. A use case for this is when an image and a text rectangle must be arranged in a button box. This version of the algorithm will calculate the location of the text rectangle. |
| | laUtils_ChildIntersectsParent | Performs an intersection test between a parent widget and a child widget |
| | laUtils_ClipRectToParent | Clips a rectangle to the parent of a widget |
| | laUtils_GetLayer | Finds the root parent of a widget, which should be a layer |
| | laUtils_ListOcclusionCullTest | Performs an occlusion test on a list of widgets an a rectangular area. This attempts to find only the topmost widgets for the given area. If a widget is completely occluded then it is removed from the list. Any widgets that remain in the list should be redrawn by the rasterizer. |
| | laUtils_OcclusionCullTest | Performs an occlusion test for a widget in the tree. A widget is occluded if it is completely covered by one or more widgets. This is useful for culling widgets before the rasterizing phase. |
| | laUtils_Pick | Finds the top-most visible widget in the tree at the given coordinates. |
| | laUtils_PickRect | Finds all of the visible widgets in the given rectangular area. |
| | laUtils_PointScreenToLocalSpace | Converts a point from layer space into the local space of a widget |
| | laUtils_RectFromParentSpace | Converts a rectangle from widget parent space to widget local space |
| | laUtils_RectToLayerSpace | Converts a rectangle from widget local space to layer space |
| | laUtils_RectToParentSpace | Converts a rectangle from widget local space to widget parent space. Widget must be a child of a layer for this to function. |
| | laUtils_RectToScreenSpace | Converts a rectangle from widget local space to screen space |
| | laWidget_AddChild | Adds the child to the parent widget specified in the argument |
| | laWidget_Delete | Delete the widget object specified |
| | laWidget_GetAlphaAmount | Return the widget's global alpha amount |
| | laWidget_GetAlphaEnable | Return the alpha enable property of the widget |
| | laWidget_GetBorderType | Return the border type associated with the widget object |
| | laWidget_GetChildAtIndex | Fetches the child at the specified index from the children list of the specified parent widget |
| | laWidget_GetChildCount | Returns the size of the children list of the specified parent widget |
| | laWidget_GetCumulativeAlphaAmount | Calculates the cumulative alpha amount for a hierarchy of widgets |

| | | |
|---|---|---|
| | laWidget_GetCumulativeAlphaEnable | Determines if this or any ancestor widget has alpha enabled |
| | laWidget_GetEnabled | Returns the boolean value of the widget enabled property |
| | laWidget_GetHeight | Returns the widget rectangles height |
| | laWidget_GetIndexOfChild | Fetches the index of the child from the children list of the specified parent widget |
| | laWidget_GetMargin | Returns the margin value associated with the widget in the laMargin pointer |
| | laWidget_GetScheme | Returns the scheme associated with the specified widget |
| | laWidget_GetVisible | Returns the boolean value of the widget visible property |
| | laWidget_GetWidth | Returns the widget rectangles width |
| | laWidget_GetX | Returns the widget rectangles upper left corner x-coordinate |
| | laWidget_GetY | Returns the widget rectangles upper left corner y-coordinate |
| | laWidget_HasFocus | Checks if the widget specified has focus in the current context |
| | laWidget_Invalidate | Invalidates the specified widget. |
| | laWidget_isOpaque | Returns true if the widget is considered opaque. |
| | laWidget_New | Create a new widget. |
| | laWidget_OverrideTouchDownEvent | Replace the TouchDownEvent callback for the widget with the new function pointer specified |
| | laWidget_OverrideTouchMovedEvent | Replace the TouchMovedEvent callback for the widget with the new function pointer specified |
| | laWidget_OverrideTouchUpEvent | Replace the TouchUpEvent callback for the widget with the new function pointer specified |
| | laWidget_RectToLayerSpace | Transforms a widget rectangle from local space to its root layer space. |
| | laWidget_RectToParentSpace | Returns the rectangle containing the parent of the widget specified |
| | laWidget_RectToScreenSpace | Transforms a widget rectangle from local space to screen space coordinates. |
| | laWidget_RemoveChild | Removes the child from the parent widget specified in the argument |
| | laWidget_Resize | Changes the widget size by the new defined width and height increments. |
| | laWidget_SetAlphaAmount | Set the widget's global alpha amount to the specified alpha amount |
| | laWidget_SetAlphaEnable | Set the alpha enable property of the widget with the boolean value specified |
| | laWidget_SetBorderType | Set the border type associated with the widget object |
| | laWidget_SetEnabled | Sets the boolean value of the widget enabled property |
| | laWidget_SetFocus | Set the widget into focus for the current context. |
| | laWidget_SetHeight | Sets the widget's height value |
| | laWidget_SetMargins | Set the margin value for left, right, top and bottom margins associated with the widget |
| | laWidget_SetParent | Sets the parent of the child widget to that specified in the argument list |
| | laWidget_SetPosition | Changes the widget position to the new defined x and y coordinates. |
| | laWidget_SetScheme | Sets the scheme variable for the specified widget |
| | laWidget_SetSize | Changes the widget size to the new defined width and height dimensions. |
| | laWidget_SetVisible | Sets the boolean value of the widget visible property |
| | laWidget_SetWidth | Sets the widget's width value |
| | laWidget_SetX | Sets the widget's x coordinate position |
| | laWidget_SetY | Sets the widget's y coordinate position |
| | laWidget_Translate | Changes the widget position by moving the widget by the defined x and y increments. |
| | laWindowWidget_GetIcon | Gets the currently used window icon |
| | laWindowWidget_GetIconMargin | Gets the current image icon margin |
| | laWindowWidget_GetTitle | Gets the title text for this window. |
| | laWindowWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laWindowWidget_SetIcon | Sets the image to be used as a window icon |
| | laWindowWidget_SetIconMargin | Sets the image icon margin |

| | | |
|---|---|---|
| | laWindowWidget_SetTitle | Sets the title text for the window. |
| | laCheckBoxWidget_SetImageMargin | Sets the distance between the image and the text |
| | laContext_SetPreemptionLevel | Set the preemption level to the specified value |
| | laKeyPadWidget_SetKeyBackgroundType | Sets the background type for a key pad cell at row/column |
| | laLayer_GetAllowInputPassThrough | Gets the layer's input passthrough setting |
| | laLayer_GetEnabled | Returns the boolean value of the layer enabled property |
| | laLayer_SetAllowInputPassthrough | Sets the layer's input passthrough flag. |
| | laLayer_SetEnabled | Sets the boolean value of the layer enabled property |
| | laListWheelWidget_GetFlickInitSpeed | Returns the flick init speed for the wheel. |
| | laListWheelWidget_GetIndicatorArea | Returns the spacing for the selected item indicator bars. |
| | laListWheelWidget_GetMaxMomentum | Returns the maximum momentum value for the wheel. |
| | laListWheelWidget_GetMomentumFalloffRate | Returns the momentum falloff rate for the wheel. |
| | laListWheelWidget_GetRotationUpdateRate | Returns the wheel rotation update rate. |
| | laListWheelWidget_GetShaded | Returns true if the list is using gradient shading to illustrate depth |
| | laListWheelWidget_GetShowIndicators | Returns true if the list is displaying its selected item indicators |
| | laListWheelWidget_GetVisibleItemCount | Returns the list's visible item count |
| | laListWheelWidget_SetFlickInitSpeed | Configures the flick init speed for the list wheel |
| | laListWheelWidget_SetIndicatorArea | Configures the display area for the list selection indicator bars |
| | laListWheelWidget_SetMaxMomentum | Configures the maximum momentum value for the wheel |
| | laListWheelWidget_SetMomentumFalloffRate | Configures the momentum falloff rate for the wheel |
| | laListWheelWidget_SetRotationUpdateRate | Configures the rotation update rate for a wheel |
| | laListWheelWidget_SetShaded | Configures the list to use gradient or flat background shading |
| | laListWheelWidget_SetShowIndicators | Configures the list to display the selected item indicator bars |
| | laListWheelWidget_SetVisibleItemCount | Sets the number of visible items in the list. Must be greater than or equal to three and must be an odd number. |
| | laRadioButtonWidget_SetImageMargin | Sets the distance between the icon and text |
| | laScreen_GetMirrored | Returns the mirror setting for the specified screen |
| | laScreen_SetMirrored | Sets the mirror setting for the specified screen |
| | laString_Remove | Removes a number of characters from a string at a given index |
| | laTextFieldWidget_SetClearOnFirstEdit | Sets the flag to indicate that the text field will be cleared on first edit. |
| | laUtils_PickFromLayer | Finds the top-most visible widget in a layer at the given coordinates. |
| | laUtils_PointToLayerSpace | Converts a point from widget space into layer space |
| | laUtils_ScreenToMirroredSpace | Takes a point in screen space and returns a transformed version in mirrored space. |
| | laUtils_ScreenToOrientedSpace | Takes a point in screen space and returns a transformed version in oriented space. |
| | laUtils_WidgetLocalRect | Returns the bounding rectangle of a widget in local space |
| | laWidget_GetBackgroundType | Return the property value 'background type' associated with the widget object |
| | laWidget_GetOptimizationFlags | Returns the optimization flags for the widget |
| | laWidget_SetBackgroundType | Set the property value 'background type' associated with the widget object |
| | laWidget_SetOptimizationFlags | Sets the optimizations for a widget |
| | laRectangleWidget_SetThickness | Sets the rectangle border thickness setting |
| | laString_DrawClipped | Wrapper around GFX Utility string draw function for Aria user interface library. Draws only a clipped area of a string. Internal use only. |
| | laString_IsEmpty | Returns a boolean indicating if the provided string contains data or has a link to the string table. |
| | laUtils_GetNextHighestWidget | Gets the next highest Z order widget in the tree from 'wgt' |
| | laUtils_RectFromLayerSpace | Converts a rectangle from layer space to widget local space |
| | laUtils_WidgetIsOccluded | This is function laUtils_WidgetIsOccluded. |
| | laUtils_WidgetLayerRect | Returns the bounding rectangle of a widget in layer space |
| | laWindowWidget_GetIconRect | This is function laWindowWidget_GetIconRect. |
| | laWindowWidget_GetTextRect | This is function laWindowWidget_GetTextRect. |
| | laWindowWidget_GetTitleBarRect | internal use only |

| | | |
|---|---|---|
| | laContext_IsDrawing | Indicates if any layers of the active screen are currently drawing a frame. |
| | laContext_IsLayerDrawing | Indicates if the layer at the given index of the active screen is currently drawing. |
| | laLayer_IsDrawing | Queries a layer to find out if it is currently drawing a frame. |
| | laLayer_GetInputRect | Gets the layer's input rectangle. |
| | laLayer_GetInputRectLocked | Gets the layer's input rect locked flag |
| | laLayer_SetInputRect | Sets the layer's input rect dimensions. |
| | laLayer_SetInputRectLocked | Sets the layer's input rect locked flag. |
| | laScreen_GetLayerSwapSync | Returns the layer swap sync setting for the specified screen |
| | laScreen_SetLayerSwapSync | Sets the layer swap sync setting for the specified screen |
| | laWidget_DeleteAllDescendants | Deletes all of the descendants of the given widget. |
| | laImageWidget_SetCallBackEnd | This is function laImageWidget_SetCallBackEnd. |
| | laImageWidget_SetCallBackStart | This is function laImageWidget_SetCallBackStart. |
| | laString_DrawSubStringClipped | Wrapper around GFX Utility string draw function for Aria user interface library. Draws the substring between the start and end offset, and draws only the section of the string within the clipping rectangle. Internal use only. |
| | laString_GetLineRect | Calculates the rectangle for a line in a string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. The line ends when a line feed or end of string is reached. |
| | laString_GetMultiLineRect | Calculates the rectangle for a given multi-line string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. The height the sum of the heights of the bounding rectangles for each line and the width is the widest among the bounding rectangles. |

## b) Data Types and Constants

| | Name | Description |
|---|---|---|
| | laBool_t | libaria bool values |
| | laContext_t | An instance of the Aria user interface library. |
| | laEditWidget_t | Specifies the edit widget structure to manage all properties and events associated with edit widgets |
| | laList_t | Linked list definition |
| | laListNode_t | Linked list node definition |
| | laMargin_t | libaria margin values |
| | laPreemptionLevel | libaria pre-emption level values |
| | laRelativePosition | libaria relative position values |
| | laResult_t | libaria results (success and failure codes). |
| | laString_t | String definition |
| | GFXU_StringTableAsset | Describes a string table asset. There is typically only ever one of these defined at any one time.<br>header - standard asset header languageCount - the number of languages in the string table stringCount - the number of strings in the string table stringIndexTable - the pointer to the string index table. the string index table is a table that contains all of the unique strings defined in the string table. fontTable - the font table contains an array of pointers to all defined font assets that the string table references fontIndexTable - the font index table is a table... more |
| | laBool | libaria bool values |
| | laContext | An instance of the Aria user interface library. |
| | laContext_ActiveScreenChangedCallback_FnPtr | Callback pointer for the active screen change notification. |
| | laContext_LanguageChangedCallback_FnPtr | Callback pointer for when the language change event occurs. |
| | laEditWidget | Specifies the edit widget structure to manage all properties and events associated with edit widgets |
| | laEditWidget_Accept_FnPtr | This is type laEditWidget_Accept_FnPtr. |
| | laEditWidget_Append_FnPtr | This is type laEditWidget_Append_FnPtr. |
| | laEditWidget_Backspace_FnPtr | This is type laEditWidget_Backspace_FnPtr. |
| | laEditWidget_Clear_FnPtr | This is type laEditWidget_Clear_FnPtr. |
| | laEditWidget_EndEdit_FnPtr | This is type laEditWidget_EndEdit_FnPtr. |

| | | |
|---|---|---|
| | laEditWidget_Set_FnPtr | This is type laEditWidget_Set_FnPtr. |
| | laEditWidget_StartEdit_FnPtr | This is type laEditWidget_StartEdit_FnPtr. |
| | laHAlignment | libaria horizontal alignment values |
| | laList | Linked list definition |
| | laListNode | Linked list node definition |
| | laMargin | libaria margin values |
| | laResult | libaria results (success and failure codes). |
| | laScreen | The structure to maintain the screen related variables and event handling |
| | laString | String definition |
| | laVAlignment | libaria vertical alignment values |
| | laBorderType_t | Specifies the different border types used for the widgets in the library |
| | laButtonState_t | Controls the button pressed state |
| | laButtonWidget_t | Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed an released state. |
| | laCheckBoxWidget_t | Implementation of a checkbox widget. |
| | laCircleWidget_t | Implementation of a circle widget. |
| | laDrawSurfaceWidget_t | Implementation of a Drawsurface widget. |
| | laEvent_t | Basic UI event definition |
| | laEventID_t | Defines internal event type IDs |
| | laEventState_t | Structure to manage the event lists, state and call back pointers |
| | laGestureID_t | Placeholder for eventual gesture support. |
| | laGradientWidget_t | Gradient widget struct definition. |
| | laGradientWidgetDirection_t | Implementation of a gradient widget. |
| | laGroupBoxWidget_t | Group box struct definition. |
| | laImageSequenceEntry_t | Image sequence entry definition |
| | laImageSequenceWidget_t | Image sequence widget struct definition |
| | laImageWidget_t | Image widget struct definition |
| | laInput_TouchDownEvent_t | Register and handle the touch press detect event |
| | laInput_TouchMovedEvent_t | Register and handle the touch coordinates changed event |
| | laInput_TouchUpEvent_t | Register and handle the touch release detect event |
| | laInputState_t | Maintain a history of touch states; currently libaria keeps track of the last touch state only. |
| | laKey_t | All values possible for key entry from the libaria keyboard widget |
| | laKeyPadCell_t | Defines a key pad cell struct |
| | laKeyPadCellAction_t | Defines an assigned action to a key pad cell |
| | laKeyPadWidget_t | Defines a key pad widget struct |
| | laLabelWidget_t | Implementation of a label widget struct |
| | laLayer_t | Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent. |
| | laLayerBuffer_t | Structure to maintain the buffer type and track the buffer location for each layer |
| | laLayerBufferType_t | Defines the type of a layer. If the layer has an explicit address then Aria tries to set that through the HAL when the layer is being set up. |
| | laLineWidget_t | Defines the implementation of a line widget struct |
| | laListItem_t | Defines a list item struct |
| | laListWheelItem_t | Implementation of a list wheel widget item struct |
| | laListWheelWidget_t | Implementation of a list wheel widget struct |
| | laListWidget_SelectionMode_t | Defines the list selection modes |
| | laListWidget_t | Defines the implementation of a list widget |
| | laMouseButton_t | All values possible for mouse key entry from the libaria mouse input |
| | laProgressBarDirection_t | Defines the valid values for the progress bar widget fill directions. |
| | laProgressBarWidget_t | Implementation of a progressbar widget struct |
| | laRadioButtonGroup_t | Defines the structure used for the Radio Button group. |
| | laRadioButtonWidget_t | Implementation of a radio button widget struct |
| | laRectangleWidget_t | Implementation of a rectangle widget struct |
| | laScheme_t | This structure specifies the style scheme components of an object. |
| | laScreen_t | The structure to maintain the screen related variables and event handling |

| | | |
|---|---|---|
| | laScreenOrientation_t | Possible values for screen orientation. |
| | laScrollBarOrientation_t | Defines the scroll bar direction values |
| | laScrollBarState_t | Defines the various scroll bar state values |
| | laScrollBarWidget_t | Implementation of a scroll bar widget. |
| | laSliderOrientation_t | Slider orientations |
| | laSliderState_t | Describes various slider states |
| | laSliderWidget_t | Implementation of a slider widget struct |
| | laTextFieldWidget_t | Implementation of a text field widget. |
| | laTouchState_t | Manage the touch input state and track the touch coordinate |
| | laTouchTestState_t | Touch test states |
| | laTouchTestWidget_t | Implementation of a touch test widget struct |
| | laWidget_t | Specifies Graphics widget structure to manage all properties and events associated with the widget |
| | laWidgetDirtyState_t | Specifies the different dirty states the widget can be assigned |
| | laWidgetDrawState_t | Specifies the different draw states the widget can be assigned |
| | laWidgetEvent_t | Basic widget event definition |
| | laWidgetType_t | Specifies the different widget types used in the library |
| | laWindowWidget_t | Implementation of a window widget struct |
| | GFX_Point | A two dimensional Cartesian point. |
| | GFX_Rect | A rectangle definition. |
| | laBorderType | Specifies the different border types used for the widgets in the library |
| | laButtonState | Controls the button pressed state |
| | laButtonWidget | Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed an released state. |
| | laButtonWidget_PressedEvent | This is type laButtonWidget_PressedEvent. |
| | laButtonWidget_ReleasedEvent | This is type laButtonWidget_ReleasedEvent. |
| | laCheckBoxWidget | Implementation of a checkbox widget. |
| | laCheckBoxWidget_CheckedEvent | This is type laCheckBoxWidget_CheckedEvent. |
| | laCheckBoxWidget_UncheckedEvent | This is type laCheckBoxWidget_UncheckedEvent. |
| | laCircleWidget | Implementation of a circle widget. |
| | laDrawSurfaceWidget | Implementation of a Drawsurface widget. |
| | laDrawSurfaceWidget_DrawCallback | This is type laDrawSurfaceWidget_DrawCallback. |
| | laEvent | Basic UI event definition |
| | laEvent_FilterEvent | Function pointer to define an event filter. Event filters allow a receiver to discard undesirable events |
| | laEventID | Defines internal event type IDs |
| | laEventState | Structure to manage the event lists, state and call back pointers |
| | laGestureID | Placeholder for eventual gesture support. |
| | laGradientWidget | Gradient widget struct definition. |
| | laGradientWidgetDirection | Implementation of a gradient widget. |
| | laGroupBoxWidget | Group box struct definition. |
| | laImageSequenceEntry | Image sequence entry definition |
| | laImageSequenceImageChangedEvent_FnPtr | This is type laImageSequenceImageChangedEvent_FnPtr. |
| | laImageSequenceWidget | Image sequence widget struct definition |
| | laImageWidget | Image widget struct definition |
| | laInput_TouchDownEvent | Register and handle the touch press detect event |
| | laInput_TouchMovedEvent | Register and handle the touch coordinates changed event |
| | laInput_TouchUpEvent | Register and handle the touch release detect event |
| | laInputState | Maintain a history of touch states; currently libaria keeps track of the last touch state only. |
| | laKey | All values possible for key entry from the libaria keyboard widget |
| | laKeyPadCell | Defines a key pad cell struct |
| | laKeyPadCellAction | Defines an assigned action to a key pad cell |
| | laKeyPadWidget | Defines a key pad widget struct |
| | laKeyPadWidget_KeyClickEvent | This is type laKeyPadWidget_KeyClickEvent. |
| | laLabelWidget | Implementation of a label widget struct |

| | | |
|---|---|---|
| | laLayer | Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent. |
| | laLayerBuffer | Structure to maintain the buffer type and track the buffer location for each layer |
| | laLayerBufferType | Defines the type of a layer. If the layer has an explicit address then Aria tries to set that through the HAL when the layer is being set up. |
| | laLineWidget | Defines the implementation of a line widget struct |
| | laListItem | Defines a list item struct |
| | laListWheelItem | Implementation of a list wheel widget item struct |
| | laListWheelWidget | Implementation of a list wheel widget struct |
| | laListWheelWidget_SelectedItemChangedEvent | This is type laListWheelWidget_SelectedItemChangedEvent. |
| | laListWidget | Defines the implementation of a list widget |
| | laListWidget_ItemSelectedChangedEvent | This is type laListWidget_ItemSelectedChangedEvent. |
| | laListWidget_SelectedItemChangedEvent | This is type laListWidget_SelectedItemChangedEvent. |
| | laListWidget_SelectionMode | Defines the list selection modes |
| | laMouseButton | All values possible for mouse key entry from the libaria mouse input |
| | laProgressBar | This is type laProgressBar. |
| | laProgressBar_ValueChangedEventCallback | This is type laProgressBar_ValueChangedEventCallback. |
| | laProgressBarDirection | Defines the valid values for the progress bar widget fill directions. |
| | laProgressBarWidget | Implementation of a progressbar widget struct |
| | laRadioButtonGroup | Defines the structure used for the Radio Button group. |
| | laRadioButtonWidget | Implementation of a radio button widget struct |
| | laRadioButtonWidget_DeselectedEvent | This is type laRadioButtonWidget_DeselectedEvent. |
| | laRadioButtonWidget_SelectedEvent | This is type laRadioButtonWidget_SelectedEvent. |
| | laRectangleWidget | Implementation of a rectangle widget struct |
| | laScheme | This structure specifies the style scheme components of an object. |
| | laScreen_CreateCallback_FnPtr | Callback pointer for a new screen create event notification. This is called when the library attempts to create a screen. |
| | laScreen_ShowHideCallback_FnPtr | Callback pointer for the active screen show or hide event change notification. |
| | laScreenOrientation | Possible values for screen orientation. |
| | laScrollBarOrientation | Defines the scroll bar direction values |
| | laScrollBarState | Defines the various scroll bar state values |
| | laScrollBarWidget | Implementation of a scroll bar widget. |
| | laScrollBarWidget_ValueChangedEvent | This is type laScrollBarWidget_ValueChangedEvent. |
| | laSliderOrientation | Slider orientations |
| | laSliderState | Describes various slider states |
| | laSliderWidget | Implementation of a slider widget struct |
| | laSliderWidget_ValueChangedEvent | This is type laSliderWidget_ValueChangedEvent. |
| | laTextFieldWidget | Implementation of a text field widget. |
| | laTextFieldWidget_TextChangedCallback | This is type laTextFieldWidget_TextChangedCallback. |
| | laTouchState | Manage the touch input state and track the touch coordinate |
| | laTouchTestState | Touch test states |
| | laTouchTestWidget | Implementation of a touch test widget struct |
| | laTouchTestWidget_PointAddedEventCallback | This is type laTouchTestWidget_PointAddedEventCallback. |
| | laWidget | Specifies Graphics widget structure to manage all properties and events associated with the widget |
| | laWidget_Constructor_FnPtr | This is type laWidget_Constructor_FnPtr. |
| | laWidget_Destructor_FnPtr | This is type laWidget_Destructor_FnPtr. |
| | laWidget_DrawFunction_FnPtr | This is type laWidget_DrawFunction_FnPtr. |
| | laWidget_Focus_FnPtr | This is type laWidget_Focus_FnPtr. |
| | laWidget_Moved_FnPtr | This is type laWidget_Moved_FnPtr. |
| | laWidget_Paint_FnPtr | This is type laWidget_Paint_FnPtr. |
| | laWidget_Resized_FnPtr | This is type laWidget_Resized_FnPtr. |
| | laWidget_TouchDownEvent_FnPtr | This is type laWidget_TouchDownEvent_FnPtr. |
| | laWidget_TouchMovedEvent_FnPtr | This is type laWidget_TouchMovedEvent_FnPtr. |
| | laWidget_TouchUpEvent_FnPtr | This is type laWidget_TouchUpEvent_FnPtr. |
| | laWidget_Update_FnPtr | This is type laWidget_Update_FnPtr. |

| | | |
|---|---|---|
| | laWidgetDirtyState | Specifies the different dirty states the widget can be assigned |
| | laWidgetDrawState | Specifies the different draw states the widget can be assigned |
| | laWidgetEvent | Basic widget event definition |
| | laWidgetType | Specifies the different widget types used in the library |
| | laWindowWidget | Implementation of a window widget struct |
| | laBackgroundType_t | Specifies the different background types used for the widgets in the library |
| | laWidgetOptimizationFlags_t | Specifies the different draw optimization flags for a widget |
| | laBackgroundType | Specifies the different background types used for the widgets in the library |
| | laWidget_LanguageChangingEvent_FnPtr | This is type laWidget_LanguageChangingEvent_FnPtr. |
| | laWidgetOptimizationFlags | Specifies the different draw optimization flags for a widget |
| | LA_DEFAULT_SCHEME_COLOR_MODE | This is macro LA_DEFAULT_SCHEME_COLOR_MODE. |
| | LA_STRING_NULLIDX | This is macro LA_STRING_NULLIDX. |
| | DEFAULT_BORDER_MARGIN | This is macro DEFAULT_BORDER_MARGIN. |
| | LA_IMAGESEQ_RESTART | This is macro LA_IMAGESEQ_RESTART. |
| | LA_INPUT_PRIMARY_ID | This is macro LA_INPUT_PRIMARY_ID. |
| | LA_MAX_TOUCH_STATES | This is macro LA_MAX_TOUCH_STATES. |
| | LA_TOUCHTEST_MEMORY_SIZE | This is macro LA_TOUCHTEST_MEMORY_SIZE. |
| | NUM_BUTTONS | This is macro NUM_BUTTONS. |
| | NUM_KEYS | This is macro NUM_KEYS. |
| | laLayer_AddDamageRect | Adds a damaged rectangle to the list. Damage rectangles are used in minimal redraw algorithms. |
| | laEventResult | Defines what happened when processing an event |
| | laLayerFrameState | Defines the frame state of a layer. Certain actions must only be performed at the start of a new frame and other actions must wait until the end of the current frame. |
| | laRectArray | This is type laRectArray. |
| | laWidget_InvalidateBorderAreas_FnPtr | This is type laWidget_InvalidateBorderAreas_FnPtr. |
| | laEventResult_t | Defines what happened when processing an event |
| | laLayerFrameState_t | Defines the frame state of a layer. Certain actions must only be performed at the start of a new frame and other actions must wait until the end of the current frame. |
| | laContextFrameState_t | Possible values for context frame state. |
| | laContextFrameState | Possible values for context frame state. |
| | laImageWidget_DrawEventCallback | |
| | laContextUpdateState_t | Possible values for context update state. |
| | laWidgetUpdateState_t | Specifies the different update states the widget can be assigned |
| | laContextUpdateState | Possible values for context update state. |
| | laWidgetUpdateState | Specifies the different update states the widget can be assigned |

## Description

This section Aria User Interface Library Interface.

### *a) Functions*

#### laContext_AddScreen Function

Add screen to the list of screens in the current context

## File

libaria_context.h

## C

```
LIB_EXPORT laResult laContext_AddScreen(laScreen* screen);
```

## Returns

laResult

## Description

Add screen to the list of screens in the current context

## Function

laResult laContext_AddScreen(laScreen* screen)

### laContext_Create Function

Creates an instance of the Aria user interface library

## File

libaria_context.h

## C

```
LIB_EXPORT laContext* laContext_Create(GFX_Driver driver, GFX_Display display, GFX_Processor processor,
GFX_ColorMode mode, GFXU_MemoryIntf* memoryIntf);
```

## Returns

laContext* - a valid context pointer or NULL

## Preconditions

Should have called laInitialize() before attempting to create a context

## Parameters

| Parameters | Description |
|---|---|
| GFX_Driver | the graphics controller the library will initialize the HAL with |
| GFX_Display | the graphics display the library will initialize the HAL with |
| GFX_ColorMode | the color mode the library will use and initialize the HAL with |
| GFXU_MemoryIntf* | the memory interface the library will use and will initialize the HAL with |

## Function

laContext* laContext_Create(laArray*)

### laContext_Destroy Function

Destroys an Aria instance

## File

libaria_context.h

## C

```
LIB_EXPORT laResult laContext_Destroy(laContext* context);
```

## Returns

laResult - indicates if the instance was successfully destroyed

## Parameters

| Parameters | Description |
|---|---|
| laContext* | a valid Aria pointer |

## Function

laResult laContext_Destroy(laContext*)

### laContext_GetActive Function

Returns the current active context.

## File

libaria_context.h

### C

```
LIB_EXPORT laContext* laContext_GetActive();
```

### Returns

laContext*

### Function

laContext* laContext_GetActive()

## laContext_GetActiveScreen Function

Returns the active screen of the current context

### File

libaria_context.h

### C

```
LIB_EXPORT laScreen* laContext_GetActiveScreen();
```

### Returns

laScreen*

### Description

Returns the active screen of the current context

### Function

laScreen* laContext_GetActiveScreen()

## laContext_GetActiveScreenIndex Function

Return the index of the active screen

### File

libaria_context.h

### C

```
LIB_EXPORT int32_t laContext_GetActiveScreenIndex();
```

### Returns

int32_t

### Description

Return the index of the active screen

### Function

int32_t laContext_GetActiveScreenIndex()

## laContext_GetColorMode Function

Returns the color mode of the current context

### File

libaria_context.h

### C

```
LIB_EXPORT GFX_ColorMode laContext_GetColorMode();
```

### Returns

GFX_ColorMode

### Function

GFX_ColorMode laContext_GetColorMode()

## laContext_GetDefaultScheme Function

Returns the pointer to the default scheme of the current context

### File

libaria_context.h

### C

```
LIB_EXPORT laScheme* laContext_GetDefaultScheme();
```

### Returns

laScheme*

### Description

Returns the pointer to the default scheme of the current context

### Function

laScheme* laContext_GetDefaultScheme()

## laContext_GetEditWidget Function

Gets the widget that is currently recieving all widget edit events.

### File

libaria_context.h

### C

```
LIB_EXPORT laEditWidget* laContext_GetEditWidget();
```

### Returns

laEditWidget*

### Description

Edit widgets are widgets that inherit the 'edit widget' API function list. These widgets are capable of receiving edit events from other widgets that are edit event broadcasters. A broadcaster could be a 'key pad' and a receiver could be a 'text edit' box.

### Function

laEditWidget* laContext_GetEditWidget()

## laContext_GetFocusWidget Function

Return a pointer to the widget in focus

### File

libaria_context.h

### C

```
LIB_EXPORT laWidget* laContext_GetFocusWidget();
```

### Returns

laWidget*

### Description

The focus widget is the widget that is currently receiving all input events. This can happen when the user initiates a touch down event on the widget and is currently dragging their finger on the display. The widget will receive all touch moved events until a touch up event is received.

### Function

laWidget* laContext_GetFocusWidget()

## laContext_GetPremptionLevel Function

Returns the preemption level for the screen

### File

libaria_context.h

### C

```
LIB_EXPORT laPreemptionLevel laContext_GetPremptionLevel();
```

### Returns

laPreemptionLevel

### Description

Returns the preemption level for the screen

### Function

laPreemptionLevel laContext_GetPreemptionLevel()

## laContext_GetScreenRect Function

Returns the display rectangle structure of the physical display

### File

libaria_context.h

### C

```
LIB_EXPORT GFX_Rect laContext_GetScreenRect();
```

### Returns

GFX_Rect

### Description

Returns the display rectangle - width height and upper left corner coordinates of the physical display

### Function

LIB_EXPORT GFX_Rect laContext_GetScreenRect()

## laContext_GetStringLanguage Function

Returns the language index of the current context

### File

libaria_context.h

### C

```
LIB_EXPORT uint32_t laContext_GetStringLanguage();
```

### Returns

uint32_t

### Description

Returns the language index of the current context

### Function

uint32_t laContext_GetStringLanguage()

### laContext_GetStringTable Function

Get a pointer to the GFXU_StringTableAsset structure that maintains the strings, associated fonts, etc

#### File

libaria_context.h

#### C

```
LIB_EXPORT GFXU_StringTableAsset* laContext_GetStringTable();
```

#### Returns

GFXU_StringTableAsset*

#### Description

Get a pointer to the GFXU_StringTableAsset structure that maintains the strings, associated fonts, etc

#### Function

GFXU_StringTableAsset* laContext_GetStringTable()

### laContext_HideActiveScreen Function

Hide the active screen

#### File

libaria_context.h

#### C

```
LIB_EXPORT GFX_DEPRECATED laResult laContext_HideActiveScreen();
```

#### Returns

void

#### Description

Hide the active screen. If the screen's persistent flag is set to true then the memory for the screen's widgets will not be deallocated. This will maintain the state of the screen.

#### Function

laResult laContext_HideActiveScreen()

### laContext_RedrawAll Function

Forces the library to redraw the currently active screen in its entirety.

#### File

libaria_context.h

#### C

```
LIB_EXPORT void laContext_RedrawAll();
```

#### Returns

void

#### Function

void laContext_RedrawAll()

### laContext_RemoveScreen Function

Remove the specified screen from the list of screens in the current context

#### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_RemoveScreen(laScreen* screen);
```

### Returns

laResult

### Description

Remove the specified screen from the list of screens in the current context

### Function

laResult laContext_RemoveScreen(laScreen* screen)

## laContext_SetActive Function

Make the specified context active

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetActive(laContext* context);
```

### Returns

laResult - LA_SUCCESS if the context was successfully set as active

### Function

laResult laContext_SetActive(laContext* context)

## laContext_SetActiveScreen Function

Change the active screen to the one specified by the index argument

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetActiveScreen(uint32_t id);
```

### Returns

void

### Description

This operation will tear down the existing layer state of the driver if necessary and rebuild the frame buffers if the existing buffers can not be reused. This operation can be potentially slow and expensive. Widgets can be used to simulate screen transitions as applicable.

### Function

laResult laContext_SetActiveScreen(uint32_t id)

## laContext_SetActiveScreenChangedCallback Function

Set the callback function pointer when the screen change event occurs

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetActiveScreenChangedCallback(laContext_ActiveScreenChangedCallback_FnPtr
cb);
```

### Returns

laResult

### Description

Set the callback function pointer when the screen change event occurs

### Function

laResult laContext_SetActiveScreenChangedCallback(laContext_ActiveScreenChangedCallback_FnPtr cb)

## laContext_SetEditWidget Function

Sets the currently active edit widget.

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetEditWidget(laWidget* widget);
```

### Returns

laResult

### Parameters

| Parameters | Description |
|---|---|
| laWidget* | a widget that inherits the edit widget API and has its 'editable' flag set to true. |

### Function

laResult laContext_SetEditWidget(laWidget* widget)

## laContext_SetFocusWidget Function

Set into focus the widget specified as the argument

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetFocusWidget(laWidget* widget);
```

### Returns

laResult

### Description

Set into focus the widget specified as the argument

### Function

laResult laContext_SetFocusWidget(laWidget* widget)

## laContext_SetLanguageChangedCallback Function

Set the callback function pointer when the language change event occurs

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetLanguageChangedCallback(laContext_LanguageChangedCallback_FnPtr cb);
```

### Returns

laResult

### Description

Set the callback function pointer when the language change event occurs

### Function

laResult laContext_SetLanguageChangedCallback(laContext_LanguageChangedCallback_FnPtr cb)

## laContext_SetStringLanguage Function

Set the language index of the current context

### File

libaria_context.h

### C

```
LIB_EXPORT void laContext_SetStringLanguage(uint32_t id);
```

### Returns

void

### Description

Set the language index of the current context

### Function

void laContext_SetStringLanguage(uint32_t id)

## laContext_SetStringTable Function

Set the StringTable pointer to the specified new StringTableAsset structure

### File

libaria_context.h

### C

```
LIB_EXPORT void laContext_SetStringTable(GFXU_StringTableAsset* table);
```

### Returns

void

### Description

Set the StringTable pointer to the specified new StringTableAsset structure

### Function

void laContext_SetStringTable( GFXU_StringTableAsset* table)

## laContext_Update Function

Runs the update loop for a library instance.

### File

libaria_context.h

### C

```
LIB_EXPORT void laContext_Update(uint32_t dt);
```

### Returns

void

### Description

The update loop allows the library to service its event array and allows any intelligent widgets to perform active update tasks. This should be run periodically, but not often enough to starve other processes. Running too little may result in a loss of UI responsiveness.

### Parameters

| Parameters | Description |
|---|---|
| uint32_t dt | a delta time representing how much time has passed since the last time laContext_Update has been called. This is typically in milliseconds. |

### Function

void laContext_Update(uint32_t dt)

## laEditWidget_Accept Function

### File

libaria_editwidget.h

### C

```
void laEditWidget_Accept();
```

### Description

This is function laEditWidget_Accept.

## laEditWidget_Append Function

### File

libaria_editwidget.h

### C

```
void laEditWidget_Append(laString str);
```

### Description

This is function laEditWidget_Append.

## laEditWidget_Backspace Function

### File

libaria_editwidget.h

### C

```
void laEditWidget_Backspace();
```

### Description

This is function laEditWidget_Backspace.

## laEditWidget_Clear Function

### File

libaria_editwidget.h

### C

```
void laEditWidget_Clear();
```

### Description

This is function laEditWidget_Clear.

## laEditWidget_EndEdit Function

### File

libaria_editwidget.h

### C

```
void laEditWidget_EndEdit();
```

### Description

This is function laEditWidget_EndEdit.

## laEditWidget_Set Function

### File

libaria_editwidget.h

### C

```
void laEditWidget_Set(laString str);
```

### Description

This is function laEditWidget_Set.

## laEditWidget_StartEdit Function

### File

libaria_editwidget.h

### C

```
laResult laEditWidget_StartEdit();
```

### Description

This is function laEditWidget_StartEdit.

## laList_Assign Function

Assignes a new pointer to an index in the list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_Assign(laList* list, size_t idx, void* val);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |
| size_t idx | the index to modify |
| void* val | the new value of the node |

### Function

int32_t laList_Assign( laList* list, size_t idx, void* val)

## laList_Clear Function

Removes all nodes from a given list

### File

libaria_list.h

### C

```
LIB_EXPORT void laList_Clear(laList* list);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laList* list | the list to modify |

### Function

void laList_Clear( laList* list)

## laList_Copy Function

Creates a duplicate of an existing list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_Copy(laList* l, laList* r);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* l | the source list |
| laList* r | the result list |

### Function

int32_t laList_Copy( laList* l, laList* r)

## laList_Create Function

Initializes a new linked list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_Create(laList* list);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to initilaize |

### Function

int32_t laList_Create( laList* list)

## laList_Destroy Function

Removes all nodes from a given list and frees the data of each node

### File

libaria_list.h

### C

```
LIB_EXPORT void laList_Destroy(laList* list);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laList* list | the list to modify |

### Function

void laList_Destroy( laList* list)

## laList_Find Function

Retrieves the index of a value from the list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_Find(laList* list, void* val);
```

### Returns

int32_t - the index of the value searched for

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to reference |
| void* val | the value to search for |

### Function

int32_t laList_Find( laList* list, void* val)

## laList_Get Function

Retrieves a value from the list

### File

libaria_list.h

### C

```
LIB_EXPORT void* laList_Get(laList* list, uint32_t idx);
```

### Returns

void* - the retrieved value

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to reference |
| uint32_t idx | the index of the value to retrieve |

### Function

void* laList_Get( laList* list, uint32_t idx)

## laList_InsertAt Function

Inserts an item into a list at a given index. All existing from index are shifted right one place.

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_InsertAt(laList* list, void* val, uint32_t idx);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |
| void* val | the value to insert |
| uint32_t idx | the position to insert the value |

### Function

int32_t laList_InsertAt( laList* list,
void* val,
uint32_t idx)

## laList_PopBack Function

Removes the last value from the list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_PopBack(laList* list);
```

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |

### Function

void laList_PopBack( laList* list)

## laList_PopFront Function

Removes the first value from the list

### File

libaria_list.h

### C

```
LIB_EXPORT void laList_PopFront(laList* list);
```

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |

### Function

void laList_PopFront( laList* list)

## laList_PushBack Function

Pushes a new node onto the back of the list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_PushBack(laList* list, void* val);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |
| void* val | the new value of the node |

### Function

int32_t laList_PushBack( laList* list, void* val)

## laList_PushFront Function

Pushes a new node onto the front of the list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_PushFront(laList* list, void*);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |
| void* val | the new value of the node |

### Function

int32_t laList_PushFront( laList* list, void* val)

## laList_Remove Function

Removes an item from the list

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_Remove(laList* list, void*);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |
| void* val | the value to remove |

### Function

int32_t laList_Remove( laList* list, void*)

## laList_RemoveAt Function

Removes an item from the list at an index

### File

libaria_list.h

### C

```
LIB_EXPORT int32_t laList_RemoveAt(laList* list, uint32_t idx);
```

### Returns

int32_t - 0 if success, -1 if failure

### Parameters

| Parameters | Description |
|---|---|
| laList* list | pointer to the list to modify |
| uint32_t idx | the index of the value to remove |

### Function

int32_t laList_Remove(laList* list, uint32_t idx)

## laString_Allocate Function

Attempts to resize the local data buffer for a string.

### File

libaria_string.h

### C

```
LIB_EXPORT laResult laString_Allocate(laString* str, uint32_t size);
```

### Returns

laResult - LA_SUCCESS if the function succeeded

### Remarks

If size is zero then the memory will be freed and the function will return success.

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to modify |
| uint32_t size | the desired size of the string |

### Function

void laString_Allocate( laString* str, uint32_t size)

## laString_Append Function

Appends a string onto the end of another string

### File

libaria_string.h

### C

```
LIB_EXPORT laResult laString_Append(laString* dst, const laString* src);
```

### Returns

laResult - LA_SUCCESS if the operation succeeded

### Parameters

| Parameters | Description |
|---|---|
| laString* dst | the destination string |
| const laString* src | the source string |

### Function

void laString_Append( laString* dst, const laString* src)

## laString_Capacity Function

Returns the capacity of a string

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_Capacity(const laString* str);
```

### Returns

uint32_t - the capacity of a string in characters

### Parameters

| Parameters | Description |
|---|---|
| const laString* str | the string to reference |

### Function

uint32_t laString_Capacity(const laString* str)

## laString_CharAt Function

Extracts the code point for the character in a string at a given index.

### File

libaria_string.h

### C

```
LIB_EXPORT GFXU_CHAR laString_CharAt(const laString* str, uint32_t idx);
```

### Returns

GFXU_CHAR - the code point of the character

### Parameters

| Parameters | Description |
|---|---|
| const laString* str | the string to reference |
| uint32_t idx | the character index to reference |

### Function

GFXU_CHAR laString_CharAt(const laString* str, uint32_t idx)

## laString_Clear Function

Sets a string's length to zero and its string table reference to NULL. Does not free any associated data and preserves capacity.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_Clear(laString* str);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to modify |

### Function

void laString_Clear( laString* str)

## laString_Compare Function

Compares two string objects

### File

libaria_string.h

### C

```
LIB_EXPORT int32_t laString_Compare(const laString* lstr, const laString* rstr);
```

### Returns

int32_t - the result of the string comparison, 0 if the strings are equal see strcmp() for more information

### Parameters

| Parameters | Description |
|---|---|
| const laString* lstr | the left argument |
| const laString* rstr | the right argument |

### Function

int32_t laString_Compare(const laString* lstr, const laString* rstr)

## laString_CompareBuffer Function

Compares a string object and a GFXU_CHAR* buffer

### File

libaria_string.h

### C

```
LIB_EXPORT int32_t laString_CompareBuffer(const laString* str, const GFXU_CHAR* buffer);
```

### Returns

int32_t - the result of the string comparison, 0 if the strings are equal see strcmp() for more information

### Parameters

| Parameters | Description |
|---|---|
| const laString* lstr | the string |
| const GFXU_CHAR* buffer | the GFXU_CHAR buffer |

### Function

int32_t laString_Compare(const laString* lstr, const GFXU_CHAR* buffer)

## laString_Copy Function

Copies the values from one string into another

### File

libaria_string.h

### C

```
LIB_EXPORT laResult laString_Copy(laString* dst, const laString* src);
```

### Returns

laResult - LA_SUCCESS if the function succeeded

### Remarks

Makes duplicate of a given string. Destination will have the same length and data but may not have the same overall capacity. The source may have lots of unused space and the destination may not match to avoid waste. Caller is responsible for the allocated memory but does not need to preserve the input string to maintain the destination string buffer state.

Caller must also ensure that the font contains all the glyphs for the string or rendering may not be possible.

### Parameters

| Parameters | Description |
|---|---|
| laString* dst | the destination string object |
| laString* src | the source string object |

### Function

laResult laString_Copy(laString* dst, const laString* src)

## laString_CreateFromBuffer Function

Creates a string object from a GFXU_CHAR buffer and a font asset pointer

### File

libaria_string.h

### C

```
LIB_EXPORT laString laString_CreateFromBuffer(const GFXU_CHAR* chr, GFXU_FontAsset* fnt);
```

### Returns

laString - created string object

### Remarks

Makes an internal copy of the input buffer for the string object. Caller is responsible for the allocated memory but does not need to preserve the input buffer to maintain the string buffer state.

Caller must also ensure that the font contains all the glyphs for the string or rendering may not be possible.

### Parameters

| Parameters | Description |
|---|---|
| const GFXU_CHAR* chr | pointer to a GFXU_CHAR buffer, can be NULL |
| GFXU_FontAsset* fnt | pointer to a font asset, can be NULL |

### Function

laString laString_CreateFromBuffer(const GFXU_CHAR* chr, GFXU_FontAsset* fnt)

## laString_CreateFromCharBuffer Function

Creates a string object from a const char* buffer and a font asset pointer. This method provides compatibility with standard c-style strings. Input string will be converted from 8-bit with to 32-bit width.

### File

libaria_string.h

### C

```
LIB_EXPORT laString laString_CreateFromCharBuffer(const char* chr, GFXU_FontAsset* fnt);
```

### Returns

laString - created string object

### Remarks

Makes an internal copy of the input buffer for the string object. Caller is responsible for the allocated memory but does not need to preserve the input buffer to maintain the string buffer state.

Caller must also ensure that the font contains all the glyphs for the string or rendering may not be possible.

### Parameters

| Parameters | Description |
|---|---|
| const char* chr | pointer to a const char* buffer, can be NULL |
| GFXU_FontAsset* fnt | pointer to a font asset, can be NULL |

### Function

laString laString_CreateFromCharBuffer(const char* chr, GFXU_FontAsset* fnt)

## laString_CreateFromID Function

Creates a string object that simply references a string in the string table.

### File

libaria_string.h

### C

```
LIB_EXPORT laString laString_CreateFromID(uint32_t id);
```

### Returns

laString - created string object

### Remarks

Allocates no memory.

### Parameters

| Parameters | Description |
|---|---|
| uint32_t id | the string table id to use |

### Function

laString laString_CreateFromID(uint32_t id)

## laString_Delete Function

Deletes all memory associated with a string object

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_Delete(laString** str);
```

### Returns

void

### Remarks

Will free local string data and the memory for the string pointer itself, setting the pointer to NULL if successful

### Parameters

| Parameters | Description |
|---|---|
| laString** str | pointer to a pointer to a string object |

### Function

void laString_Delete( laString** str)

## laString_Destroy Function

Destroys a string object. This frees the strings internal data buffer, if it exists, sets its string table reference to null, and clears all supporting attributes.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_Destroy(laString* str);
```

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to modify |

## Function

void laString_Destroy( laString* str)

### laString_Draw Function

Wrapper around GFX Utility string draw function for Aria user interface library. Internal use only.

## File

libaria_string.h

## C

```
LIB_EXPORT void laString_Draw(laString* str, int32_t x, int32_t y, GFXU_ExternalAssetReader** reader);
```

## Returns

void

## Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to draw |
| int32_t x | x position to render at |
| int32_t y | y position to render at |
| GFXU_ExternalAssetReader** reader | external reader state machine, if string font is located external |

## Function

void laString_Draw( laString* str,
int32_t x,
int32_t y,
GFXU_ExternalAssetReader** reader)

### laString_ExtractFromTable Function

Extracts a read-only string from the string table into a modifiable string object. This relies on the active context to indicate which string table to reference as well as which language entry to extract.

## File

libaria_string.h

## C

```
LIB_EXPORT void laString_ExtractFromTable(laString* dst, uint32_t table_index);
```

## Returns

void

## Remarks

Caller is responsible for the allocated memory but does not need to preserve the input buffer to maintain the string buffer state.

## Parameters

| Parameters | Description |
|---|---|
| laString* dst | the destination string object |
| uint32_t table_index | the table index to extract |

## Function

void laString_ExtractFromTable( laString* dst, uint32_t table_index)

### laString_GetCharIndexAtPoint Function

Given an offset in pixels returns the corresponding character index.

## File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_GetCharIndexAtPoint(laString* str, int32_t x);
```

### Returns

uint32_t - character index

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| int32_t x | x offset in pixels |

### Function

uint32_t laString_GetCharIndexAtPoint( laString* str, int32_t x)

## laString_GetCharOffset Function

Returns the offset of a given character index in pixels.

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_GetCharOffset(laString* str, uint32_t idx);
```

### Returns

uint32_t - the offset in pixels

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| uint32_t idx | the character index offset to calculate |

### Function

uint32_t laString_GetCharOffset( laString* str, uint32_t idx)

## laString_GetCharWidth Function

Given a character index, gets the width of that character. Only accurate if the string has a font associated with it and that font contains all the characters in the string in question.

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_GetCharWidth(laString* str, uint32_t idx);
```

### Returns

uint32_t - character width in pixels

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| uint32_t x | character index to reference |

### Function

uint32_t laString_GetCharWidth( laString* str, uint32_t idx)

## laString_GetHeight Function

Returns the height of a string by referencing its associated font asset data.

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_GetHeight(laString* str);
```

### Returns

uint32_t - the height of the string

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |

### Function

uint32_t laString_GetHeight( laString* str)

## laString_GetRect Function

Calculates the rectangle for a given string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_GetRect(laString* str, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| GFX_Rect* rect | the calculated string rectangle result |

### Function

void laString_GetRect( laString* str, GFX_Rect* rect)

## laString_Initialize Function

Initializes a string struct to default

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_Initialize(laString* str);
```

### Returns

void

### Remarks

Allocates no memory

### Parameters

| Parameters | Description |
|---|---|
| laString* str | pointer to a string object |

### Function

void laString_Initialize( laString* str)

## laString_Insert Function

Inserts a string into another string at a given index

### File

libaria_string.h

### C

```
LIB_EXPORT laResult laString_Insert(laString* dst, const laString* src, uint32_t idx);
```

### Returns

laResult - LA_SUCCESS if the operation succeeded

### Parameters

| Parameters | Description |
|---|---|
| laString* dst | the destination string |
| const laString* src | the source string |
| uint32_t idx | the insertion index |

### Function

void laString_Insert( laString* dst,const laString* src, uint32_t idx)

## laString_Length Function

Calculates the length of a string in characters

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_Length(const laString* str);
```

### Returns

uint32_t - the number of characters in the string

### Parameters

| Parameters | Description |
|---|---|
| const laString* str | the string to reference |

### Function

uint32_t laString_Length(const laString* str)

## laString_New Function

Allocates a memory for a new string

### File

libaria_string.h

### C

```
LIB_EXPORT laString* laString_New(laString* src);
```

### Returns

laString* - pointer to the newly allocated string

### Remarks

Caller is responsible for freeing the memory allocated by this function

### Parameters

| Parameters | Description |
|---|---|
| laString* src | a string to copy, can be NULL |

### Function

laString* laString_New(laString* src)

## laString_ReduceLength Function

Reduces the length of a string. This simply slides the null terminator to the left and does not affect the string's capacity value.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_ReduceLength(laString* str, uint32_t length);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to modify |
| uint32_t length | the new desired length in characters |

### Function

void laString_ReduceLength( laString* str, uint32_t length)

## laString_Set Function

Attempts to set the local data buffer of a string to an input buffer

### File

libaria_string.h

### C

```
LIB_EXPORT laResult laString_Set(laString* str, const GFXU_CHAR* buffer);
```

### Returns

laResult - LA_SUCCESS if the function succeeded

### Remarks

Makes an internal copy of the input buffer for the string object. Caller is responsible for the allocated memory but does not need to preserve the input buffer to maintain the string buffer state.

Caller must also ensure that the font contains all the glyphs for the string or rendering may not be possible.

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to modify |
| const GFXU_CHAR* buffer | the input buffer |

### Function

laResult laString_Set(laString* str, const GFXU_CHAR* buffer)

## laString_SetCapacity Function

Attempts to adjust the capacity of a string

### File

libaria_string.h

### C

```
LIB_EXPORT laResult laString_SetCapacity(laString* str, uint32_t cap);
```

### Returns

laResult - LA_SUCCESS if the operation succeeded

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to modify |
| uint32_t cap | the new desired capacity |

### Function

void laString_SetCapacity( laString* str, uint32_t cap)

## laString_ToCharBuffer Function

Extracts the data buffer from a string and copies it into the provided buffer argument.

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_ToCharBuffer(const laString* str, GFXU_CHAR* buffer, uint32_t size);
```

### Returns

uint32_t - the number of characters copied

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| GFXU_CHAR* buffer | the destination buffer |
| uint32_t size | the max size of the destination buffer |

### Function

uint32_t laString_ToCharBuffer(const laString* str,
GFXU_CHAR* buffer,
uint32_t size)

## laButtonWidget_GetHAlignment Function

Gets the horizontal alignment setting for a button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laHAlignment laButtonWidget_GetHAlignment(laButtonWidget* btn);
```

### Returns

laHAlignment - the horizontal alignment value

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

laHAlignment laButtonWidget_GetHAlignment(laButtonWidget* btn)

## laButtonWidget_GetImageMargin Function

Gets the distance between the icon and the text

### File

libaria_widget_button.h

### C

```
LIB_EXPORT uint32_t laButtonWidget_GetImageMargin(laButtonWidget* btn);
```

### Returns

uint32_t - the distance value

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |

### Function

uint32_t laButtonWidget_GetImageMargin( laButtonWidget* btn)

## laButtonWidget_GetImagePosition Function

Gets the position of the button icon

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laRelativePosition laButtonWidget_GetImagePosition(laButtonWidget* btn);
```

### Returns

laRelativePosition

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

laRelativePosition laButtonWidget_GetImagePosition(laButtonWidget* btn)

## laButtonWidget_GetPressed Function

Gets the pressed state of a button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laBool laButtonWidget_GetPressed(laButtonWidget* btn);
```

### Returns

laBool - the button pressed state

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

laBool laButtonWidget_GetPressed(laButtonWidget* btn)

## laButtonWidget_GetPressedEventCallback Function

Gets the callback associated with the button pressed event

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laButtonWidget_PressedEvent laButtonWidget_GetPressedEventCallback(laButtonWidget* btn);
```

### Returns

laButtonWidget_PressedEvent

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |

### Function

laButtonWidget_PressedEvent laButtonWidget_GetPressedEventCallback(laButtonWidget* btn)

## laButtonWidget_GetPressedImage Function

Gets the pressed image asset pointer for a button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laButtonWidget_GetPressedImage(laButtonWidget* btn);
```

### Returns

GFXU_ImageAsset* - the pressed asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

GFXU_ImageAsset* laButtonWidget_GetPressedImage(laButtonWidget* btn)

## laButtonWidget_GetPressedOffset Function

Gets the offset of the button internals when pressed

### File

libaria_widget_button.h

### C

```
LIB_EXPORT int32_t laButtonWidget_GetPressedOffset(laButtonWidget* btn);
```

### Returns

int32_t - the distance value

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |

### Function

int32_t laButtonWidget_GetPressedOffset( laButtonWidget* btn)

## laButtonWidget_GetReleasedEventCallback Function

Gets the callback for the button released event

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laButtonWidget_ReleasedEvent laButtonWidget_GetReleasedEventCallback(laButtonWidget* btn);
```

### Returns

laButtonWidget_ReleasedEvent

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |

### Function

laButtonWidget_ReleasedEvent laButtonWidget_GetReleasedEventCallback(laButtonWidget* btn)

## laButtonWidget_GetReleasedImage Function

Gets the currently used released icon

### File

libaria_widget_button.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laButtonWidget_GetReleasedImage(laButtonWidget* btn);
```

### Returns

GFXU_ImageAsset* - the released asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

GFXU_ImageAsset* laButtonWidget_GetReleasedImage(laButtonWidget* btn)

## laButtonWidget_GetText Function

Gets the text for a button. If the button's string has local data then a duplicate of the string will be allocated. The caller is responsible for managing the memory for the duplicated string. If the button string is a string table reference then only the reference ID is copied.

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_GetText(laButtonWidget* btn, laString* str);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |
| laString* str | pointer to a string to copy the button string into |

### Function

laResult laButtonWidget_GetText(laButtonWidget* btn, laString* str)

## laButtonWidget_GetToggleable Function

Gets the value of this button's toggle flag

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laBool laButtonWidget_GetToggleable(laButtonWidget* btn);
```

### Returns

laBool - the value of the toggle flag

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

laBool laButtonWidget_GetToggleable(laButtonWidget* btn)

## laButtonWidget_GetVAlignment Function

Gets the vertical alignment setting for a button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laVAlignment laButtonWidget_GetVAlignment(laButtonWidget* btn);
```

### Returns

laVAlignment - the vertical alignment setting for the button

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to reference |

### Function

laVAlignment laButtonWidget_GetVAlignment(laButtonWidget* btn)

## laButtonWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laButtonWidget* laButtonWidget_New();
```

### Returns

laButtonWidget* - pointer to a new button widget instance

### Description

Creates a new button widget instance. Invokes the button constructor

### Remarks

Caller is responsible for managing the memory allocated by this function until the widget is added to a valid widget tree.

### Function

laButtonWidget* laButtonWidget_New()

## laButtonWidget_SetHAlignment Function

Sets the horizontal alignment value for a button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetHAlignment(laButtonWidget* btn, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to modify |
| laHAlignment align | the desired alignment value |

### Function

laResult laButtonWidget_SetHAlignment(laButtonWidget* btn,
laHAlignment align)

## laButtonWidget_SetImageMargin Function

Sets the distance between the icon and text

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetImageMargin(laButtonWidget* btn, uint32_t mg);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| uint32_t | the distance value |

### Function

laResult laButtonWidget_SetImageMargin(laButtonWidget* btn,
uint32_t mg)

## laButtonWidget_SetImagePosition Function

Sets the position of the button icon

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetImagePosition(laButtonWidget* btn, laRelativePosition pos);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| laRelativePosition pos | the desired image position |

### Function

laResult laButtonWidget_SetImagePosition(laButtonWidget* btn,
laRelativePosition pos)

## laButtonWidget_SetPressed Function

Sets the pressed state for a button.

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetPressed(laButtonWidget* btn, laBool pressed);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to modify |
| laBool pressed | the pressed state |

### Function

laResult laButtonWidget_SetPressed(laButtonWidget* btn, laBool pressed)

## laButtonWidget_SetPressedEventCallback Function

Sets the pressed event callback for the button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetPressedEventCallback(laButtonWidget* btn, laButtonWidget_PressedEvent
cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| laButtonWidget_PressedEvent cb | a valid callback pointer or NULL |

### Function

laResult laButtonWidget_SetPressedEventCallback(laButtonWidget* btn,
laButtonWidget_PressedEvent cb)

## laButtonWidget_SetPressedImage Function

Sets the image to be used as a pressed icon

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetPressedImage(laButtonWidget* btn, GFXU_ImageAsset* img);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| GFXU_ImageAsset* img | pointer to an image asset |

### Function

laResult laButtonWidget_SetPressedImage(laButtonWidget* btn,
GFXU_ImageAsset* img)

## laButtonWidget_SetPressedOffset Function

Sets the offset of the button internals when pressed

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetPressedOffset(laButtonWidget* btn, int32_t offs);
```

### Returns

laResult - the operation result

### Description

This value will be applied to all of the contents of the button when it is pressed. This helps to visualize the button being pressed.

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| int32_t | the distance value |

### Function

laResult laButtonWidget_SetPressedOffset(laButtonWidget* btn, int32_t offs)

## laButtonWidget_SetReleasedEventCallback Function

Sets the callback for the button released event

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetReleasedEventCallback(laButtonWidget* btn,
laButtonWidget_ReleasedEvent cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| laButtonWidget_ReleasedEvent cb | a valid callback pointer or NULL |

### Function

laResult laButtonWidget_SetReleasedEventCallback(laButtonWidget* btn,
laButtonWidget_ReleasedEvent cb)

## laButtonWidget_SetReleasedImage Function

Sets the image to be used as the released icon

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetReleasedImage(laButtonWidget* btn, GFXU_ImageAsset* img);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the widget |
| GFXU_ImageAsset* img | the image asset to be used |

### Function

laResult laButtonWidget_SetReleasedImage(laButtonWidget* btn,
GFXU_ImageAsset* img)

## laButtonWidget_SetText Function

Sets the text for a button. If the input string has local data then the data will be copied into the button's local string, causing a memory allocation. If the input string is a string table reference then only the reference will be copied. The input string can be safely modified and the button string will not be affected.

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetText(laButtonWidget* btn, laString str);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to modify |
| laString str | the string to set to the button |

### Function

laResult laButtonWidget_SetText(laButtonWidget* btn, laString str)

## laButtonWidget_SetToggleable Function

Enables the toggle mode for a button. When pressed, toggle buttons will stay down until pressed again.

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetToggleable(laButtonWidget* btn, laBool toggleable);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the button to modify |
| laBool toggleable | the desired togglestate |

### Function

laResult laButtonWidget_SetToggleable(laButtonWidget* btn,
laBool toggleable)

## laButtonWidget_SetVAlignment Function

Sets the vertical alignment for a button

### File

libaria_widget_button.h

### C

```
LIB_EXPORT laResult laButtonWidget_SetVAlignment(laButtonWidget* btn, laVAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laButtonWidget* btn | the btn to modify |
| laVAlignment align | the desired vertical alignment setting |

### Function

laResult laButtonWidget_SetVAlignment(laButtonWidget* btn,
laVAlignment align)

## laCheckBoxWidget_GetChecked Function

Gets the checked state of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laBool laCheckBoxWidget_GetChecked(laCheckBoxWidget* cbox);
```

### Returns

laBool - the checked flag value

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laBool laCheckBoxWidget_GetChecked(laCheckBoxWidget* cbox)

## laCheckBoxWidget_GetCheckedEventCallback Function

Gets the checked event callback

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laCheckBoxWidget_CheckedEvent laCheckBoxWidget_GetCheckedEventCallback(laCheckBoxWidget* cbox);
```

### Returns

laCheckBoxWidget_CheckedEvent - a valid callback pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laCheckBoxWidget_CheckedEvent laCheckBoxWidget_GetCheckedEventCallback(laCheckBoxWidget* cbox)

## laCheckBoxWidget_GetCheckedImage Function

Gets the checked image of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laCheckBoxWidget_GetCheckedImage(laCheckBoxWidget* btn);
```

### Returns

GFXU_ImageAsset* - the current checked image asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

GFXU_ImageAsset* laCheckBoxWidget_GetCheckedImage(laCheckBoxWidget* btn)

## laCheckBoxWidget_GetHAlignment Function

Gets the horizontal alignment of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laHAlignment laCheckBoxWidget_GetHAlignment(laCheckBoxWidget* cbox);
```

### Returns

laHAlignment - the current halign value

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laHAlignment laCheckBoxWidget_GetHAlignment(laCheckBoxWidget* cbox)

## laCheckBoxWidget_GetImageMargin Function

Gets the distance between the image and the text

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT uint32_t laCheckBoxWidget_GetImageMargin(laCheckBoxWidget* btn);
```

### Returns

uint32_t - the current image margin value

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

uint32_t laCheckBoxWidget_GetImageMargin( laCheckBoxWidget* btn)

## laCheckBoxWidget_GetImagePosition Function

Gets the image position of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laRelativePosition laCheckBoxWidget_GetImagePosition(laCheckBoxWidget* btn);
```

### Returns

laRelativePosition - the current image position value

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laRelativePosition laCheckBoxWidget_GetImagePosition(laCheckBoxWidget* btn)

## laCheckBoxWidget_GetText Function

Gets a copy of the checkbox text. If the text has local data the data will be duplicated. The caller is responsible for managing the memory as appropriate.

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_GetText(laCheckBoxWidget* cbox, laString* str);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| str | pointer to an laString object |

### Function

laResult laCheckBoxWidget_GetText(laCheckBoxWidget* cbox, laString* str)

## laCheckBoxWidget_GetUncheckedEventCallback Function

Gets the unchecked event callback

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laCheckBoxWidget_UncheckedEvent laCheckBoxWidget_GetUncheckedEventCallback(laCheckBoxWidget*
cbox);
```

### Returns

laCheckBoxWidget_UncheckedEvent - a valid callback pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laCheckBoxWidget_UncheckedEvent laCheckBoxWidget_GetUncheckedEventCallback(laCheckBoxWidget* cbox)

## laCheckBoxWidget_GetUncheckedImage Function

Gets the unchecked image of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laCheckBoxWidget_GetUncheckedImage(laCheckBoxWidget* btn);
```

### Returns

GFXU_ImageAsset* - the current unchecked image asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

GFXU_ImageAsset* laCheckBoxWidget_GetUncheckedImage(laCheckBoxWidget* btn)

## laCheckBoxWidget_GetVAlignment Function

Gets the vertical alignment of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laVAlignment laCheckBoxWidget_GetVAlignment(laCheckBoxWidget* cbox);
```

### Returns

laVAlignment

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laVAlignment laCheckBoxWidget_GetVAlignment(laCheckBoxWidget* cbox)

## laCheckBoxWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is

added to a widget tree.

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laCheckBoxWidget* laCheckBoxWidget_New();
```

### Returns

laCheckBoxWidget*

### Function

laCheckBoxWidget* laCheckBoxWidget_New()

## laCheckBoxWidget_SetChecked Function

Sets the checked state of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetChecked(laCheckBoxWidget* cbox, laBool checked);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| laBool checked | the desired checked value |

### Function

laResult laCheckBoxWidget_SetChecked(laCheckBoxWidget* cbox,
laBool checked)

## laCheckBoxWidget_SetCheckedEventCallback Function

Sets the checked event callback

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetCheckedEventCallback(laCheckBoxWidget* cbox,
laCheckBoxWidget_CheckedEvent cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| laCheckBoxWidget_CheckedEvent cb | a valid callback pointer or NULL |

### Function

laResult laCheckBoxWidget_SetCheckedEventCallback(laCheckBoxWidget* cbox,
laCheckBoxWidget_CheckedEvent cb)

## laCheckBoxWidget_SetCheckedImage Function

Sets the checked image of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetCheckedImage(laCheckBoxWidget* btn, GFXU_ImageAsset* img);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| GFXU_ImageAsset* img | the desired checked image asset pointer |

### Function

laResult laCheckBoxWidget_SetCheckedImage(laCheckBoxWidget* btn,
GFXU_ImageAsset* img)

## laCheckBoxWidget_SetHAlignment Function

Sets the horizontal alignment of the check box.

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetHAlignment(laCheckBoxWidget* cbox, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| laHAlignment align | the desired halign value |

### Function

laResult laCheckBoxWidget_SetHAlignment(laCheckBoxWidget* cbox,
laHAlignment align)

## laCheckBoxWidget_SetImagePosition Function

Sets the image position of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetImagePosition(laCheckBoxWidget* btn, laRelativePosition pos);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

| | |
|---|---|
| laRelativePosition pos | the desired image position value |

### Function

laResult laCheckBoxWidget_SetImagePosition(laCheckBoxWidget* btn,
laRelativePosition pos)

## laCheckBoxWidget_SetText Function

Sets the checkbox text to the input string. If the string has local data the data will be duplicated and copied to the checkboxes internal string.

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetText(laCheckBoxWidget* cbox, laString str);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |

### Function

laResult laCheckBoxWidget_SetText(laCheckBoxWidget* cbox, laString str)

## laCheckBoxWidget_SetUncheckedEventCallback Function

Sets the unchecked event callback

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetUncheckedEventCallback(laCheckBoxWidget* cbox,
laCheckBoxWidget_UncheckedEvent cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| laCheckBoxWidget_UncheckedEvent cb | a valid callback pointer or NULL |

### Function

laResult laCheckBoxWidget_SetUncheckedEventCallback(laCheckBoxWidget* cbox,
laCheckBoxWidget_UncheckedEvent cb)

## laCheckBoxWidget_SetUncheckedImage Function

Sets the unchecked image of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetUncheckedImage(laCheckBoxWidget* btn, GFXU_ImageAsset* img);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| GFXU_ImageAsset* img | the desired unchecked image asset pointer |

### Function

laResult laCheckBoxWidget_SetUncheckedImage(laCheckBoxWidget* btn,
GFXU_ImageAsset* img)

## laCheckBoxWidget_SetVAlignment Function

Sets the vertical alignment of the check box

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetVAlignment(laCheckBoxWidget* cbox, laVAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| laVAlignment align | the valign value |

### Function

laResult laCheckBoxWidget_SetVAlignment(laCheckBoxWidget* cbox,
laVAlignment align)

## laCircleWidget_GetOrigin Function

Gets the origin coordiates of a circle widget

### File

libaria_widget_circle.h

### C

```
LIB_EXPORT laResult laCircleWidget_GetOrigin(laCircleWidget* cir, int32_t* x, int32_t* y);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCircleWidget* cir | the widget |
| int32_t* x | pointer to an integer pointer to store x |
| int32_t* y | pointer to an integer pointer to store y |

### Function

laResult laCircleWidget_GetOrigin(laCircleWidget* cir, int32_t* x, int32_t* y)

## laCircleWidget_GetRadius Function

Gets the radius of a circle widget

### File

libaria_widget_circle.h

### C

```
LIB_EXPORT uint32_t laCircleWidget_GetRadius(laCircleWidget* cir);
```

### Returns

uint32_t

### Parameters

| Parameters | Description |
|---|---|
| laCircleWidget* cir | the widget |

### Function

uint32_t laCircleWidget_GetRadius( laCircleWidget* cir)

## laCircleWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_circle.h

### C

```
LIB_EXPORT laCircleWidget* laCircleWidget_New();
```

### Returns

laCircleWidget*

### Function

laCircleWidget* laCircleWidget_New()

## laCircleWidget_SetOrigin Function

Sets the origin coordiates of a circle widget

### File

libaria_widget_circle.h

### C

```
LIB_EXPORT laResult laCircleWidget_SetOrigin(laCircleWidget* cir, int32_t x, int32_t y);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCircleWidget* cir | the widget |
| int32_t x | the desired x origin coordinate |
| int32_t y | the desired y origin coordinate |

### Function

laResult laCircleWidget_SetOrigin(laCircleWidget* cir, int32_t x, int32_t y)

## laCircleWidget_SetRadius Function

Sets the radius of a circle widget

### File

libaria_widget_circle.h

### C

```
LIB_EXPORT laResult laCircleWidget_SetRadius(laCircleWidget* cir, uint32_t rad);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCircleWidget* cir | the widget |
| uint32_t red | the desired radius value |

### Function

laResult laCircleWidget_SetRadius(laCircleWidget* cir, uint32_t rad)

## laDraw_1x2BevelBorder Function

Internal utility function to draw a 1x2 bevel border

### File

libaria_draw.h

### C

```
LIB_EXPORT void laDraw_1x2BevelBorder(GFX_Rect* rect, GFX_Color topColor, GFX_Color bottomInnerColor,
GFX_Color bottomOuterColor);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* rect | the rect to draw (screen space) |
| GFX_Color topColor | the color of the top left lines |
| GFX_Color bottomInnerColor | the color of the bottom inner line |
| GFX_Color bottomOuterColor | the color of the bottom outer line |

### Function

void laDraw_1x2BevelBorder( GFX_Rect* rect,
GFX_Color topColor,
GFX_Color bottomInnerColor,
GFX_Color bottomOuterColor)

## laDraw_2x1BevelBorder Function

Internal utility function to draw a 2x1 bevel border

### File

libaria_draw.h

### C

```
LIB_EXPORT void laDraw_2x1BevelBorder(GFX_Rect* rect, GFX_Color topOuterColor, GFX_Color topInnerColor,
GFX_Color bottomOuterColor);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* rect | the rect to draw (screen space) |
| GFX_Color topOuterColor | the color of the top outer line |
| GFX_Color topInnerColor | the color of the top inner line |
| GFX_Color bottomOuterColor | the color of the bottom lines |

### Function

void laDraw_2x1BevelBorder( GFX_Rect* rect,
GFX_Color topOuterColor,

GFX_Color topInnerColor,
GFX_Color bottomOuterColor)

## laDraw_2x2BevelBorder Function

Internal utility function to draw a 2x2 bevel border

### File

libaria_draw.h

### C

```
LIB_EXPORT void laDraw_2x2BevelBorder(GFX_Rect* rect, GFX_Color topOuterColor, GFX_Color topInnerColor,
GFX_Color bottomInnerColor, GFX_Color bottomOuterColor);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* rect | the rect to draw (screen space) |
| GFX_Color topOuterColor | the color of the top outer line |
| GFX_Color topInnerColor | the color of the top inner line |
| GFX_Color bottomInnerColor | the color of the bottom inner line |
| GFX_Color bottomOuterColor | the color of the bottom outer line |

### Function

void laDraw_2x2BevelBorder( GFX_Rect* rect,
GFX_Color topOuterColor,
GFX_Color topInnerColor,
GFX_Color bottomInnerColor,
GFX_Color bottomOuterColor)

## laDraw_LineBorder Function

Internal utility function to draw a basic line border

### File

libaria_draw.h

### C

```
LIB_EXPORT void laDraw_LineBorder(GFX_Rect* rect, GFX_Color color);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* rect | the rect to draw (screen space) |
| GFX_Color color | the color to draw |

### Function

void laDraw_LineBorder( GFX_Rect* rect, GFX_Color color)

## laDrawSurfaceWidget_GetDrawCallback Function

Returns the pointer to the currently set draw callback.

### File

libaria_widget_drawsurface.h

### C

```
LIB_EXPORT laDrawSurfaceWidget_DrawCallback laDrawSurfaceWidget_GetDrawCallback(laDrawSurfaceWidget* sfc);
```

### Returns

laDrawSurfaceWidget_DrawCallback - a valid callback pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laDrawSurfaceWidget* sfc | the widget |

### Function

laDrawSurfaceWidget_DrawCallback laDrawSurfaceWidget_GetDrawCallback(laDrawSurfaceWidget* sfc)

## laDrawSurfaceWidget_New Function

Allocates memory for a new DrawSurface widget.

### File

libaria_widget_drawsurface.h

### C

```
LIB_EXPORT laDrawSurfaceWidget* laDrawSurfaceWidget_New();
```

### Returns

laDrawSurfaceWidget*

### Description

Allocates memory for a new DrawSurface widget. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### Function

laDrawSurfaceWidget* laDrawSurfaceWidget_New()

## laDrawSurfaceWidget_SetDrawCallback Function

Sets the draw callback pointer for the draw surface widget.

### File

libaria_widget_drawsurface.h

### C

```
LIB_EXPORT laResult laDrawSurfaceWidget_SetDrawCallback(laDrawSurfaceWidget* sfc,
laDrawSurfaceWidget_DrawCallback cb);
```

### Returns

laResult - the result of the operation

### Description

Sets the draw callback pointer for the draw surface widget. This callback will be called during Aria's paint loop and allows the application to perform HAL draw calls. The application should not adjust HAL layer, buffer, or context options in any way during this phase.

The callback should return GFX_TRUE if it has completed drawing. Returning GFX_FALSE will indicate to the renderer that the DrawSurface requires more time to draw and will call it again during the next paint loop.

### Parameters

| Parameters | Description |
|---|---|
| laDrawSurfaceWidget* sfc | the widget |
| laDrawSurfaceWidget_DrawCallback | a valid callback pointer or NULL |

### Function

laResult laDrawSurfaceWidget_SetDrawCallback(laDrawSurfaceWidget* sfc,
laDrawSurfaceWidget_DrawCallback cb)

## laEvent_AddEvent Function

Add the mentioned event callback to the list of events maintained by the current context

### File

libaria_event.h

### C

```
laResult laEvent_AddEvent(laEvent* evt);
```

### Returns

laResult

### Description

Add the mentioned event callback to the list of events maintained by the current context

### Function

laResult laEvent_AddEvent(laEvent* evt)

## laEvent_ClearList Function

Clear the event list maintained by the current context.

### File

libaria_event.h

### C

```
laResult laEvent_ClearList();
```

### Returns

laResult

### Description

Clear the event list maintained by the current context.

### Function

laResult laEvent_ClearList()

## laEvent_GetCount Function

Returns the number of events listed in the current context

### File

libaria_event.h

### C

```
LIB_EXPORT uint32_t laEvent_GetCount();
```

### Returns

uint32_t

### Description

Returns the number of events listed in the current context

### Function

uint32_t laEvent_GetCount()

## laEvent_ProcessEvents Function

Processes the screen change as well as touch events

### File

libaria_event.h

### C

```
laResult laEvent_ProcessEvents();
```

### Returns

laResult

### Description

When a screen change event occurs, the specific screen change event handler has to be called as well as some generic maintenance for the screen change like destroying or hiding screen resources needs to be done. This function handles these tasks. It also handles similarly the touch events for individual widgets in the same manner.

### Function

laResult laEvent_ProcessEvents()

## laEvent_SetFilter Function

Set callback pointer for current context filter event

### File

libaria_event.h

### C

```
LIB_EXPORT laResult laEvent_SetFilter(laEvent_FilterEvent cb);
```

### Returns

laResult

### Description

Set callback pointer for current context filter event

### Function

laResult laEvent_SetFilter(laEvent_FilterEvent cb)

## laGradientWidget_GetDirection Function

Gets the gradient direction value for this widget.

### File

libaria_widget_gradient.h

### C

```
LIB_EXPORT laGradientWidgetDirection laGradientWidget_GetDirection(laGradientWidget* grad);
```

### Returns

laGradientWidgetDirection - the current gradient direction

### Parameters

| Parameters | Description |
|---|---|
| laGradientWidget* grad | the widget |

### Function

laGradientWidgetDirection laGradientWidget_GetDirection(laGradientWidget* grad)

## laGradientWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_gradient.h

### C

```
LIB_EXPORT laGradientWidget* laGradientWidget_New();
```

### Returns

laGradientWidget*

### Function

laGradientWidget* laGradientWidget_New()

## laGradientWidget_SetDirection Function

Sets the gradient direction value for this widget.

### File

libaria_widget_gradient.h

### C

```
LIB_EXPORT laResult laGradientWidget_SetDirection(laGradientWidget* grad, laGradientWidgetDirection dir);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laGradientWidget* grad | the widget |
| laGradientWidgetDirection dir | the desired gradient direction |

### Function

laResult laGradientWidget_SetDirection(laGradientWidget* grad, laGradientWidgetDirection dir)

## laGroupBoxWidget_GetAlignment Function

Gets the horizontal alignmnet for the group box title text

### File

libaria_widget_groupbox.h

### C

```
LIB_EXPORT laHAlignment laGroupBoxWidget_GetAlignment(laGroupBoxWidget* box);
```

### Returns

laHAlignment - the current halign value

### Parameters

| Parameters | Description |
|---|---|
| laGroupBoxWidget* box | the widget |

### Function

laHAlignment laGroupBoxWidget_GetAlignment(laGroupBoxWidget* box)

## laGroupBoxWidget_GetText Function

Gets the text value for the group box.

### File

libaria_widget_groupbox.h

### C

```
LIB_EXPORT laResult laGroupBoxWidget_GetText(laGroupBoxWidget* box, laString* str);
```

### Returns

laResult

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laGroupBoxWidget* box | the widget |
| laString* str | a pointer to an laString object |

### Function

laResult laGroupBoxWidget_GetText(laGroupBoxWidget* lbl, laString* str)

## laGroupBoxWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_groupbox.h

### C

```
LIB_EXPORT laGroupBoxWidget* laGroupBoxWidget_New();
```

### Returns

laGroupBoxWidget*

### Description

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### Function

laGroupBoxWidget* laGroupBoxWidget_New()

## laGroupBoxWidget_SetAlignment Function

Sets the alignment for the group box title text

### File

libaria_widget_groupbox.h

### C

```
LIB_EXPORT laResult laGroupBoxWidget_SetAlignment(laGroupBoxWidget* box, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laGroupBoxWidget* box | the widget |
| laHAlignment | the desired halign value |

### Function

laResult laGroupBoxWidget_SetAlignment(laGroupBoxWidget* box,
laHAlignment align)

## laGroupBoxWidget_SetText Function

Sets the text value for the group box.

### File

libaria_widget_groupbox.h

### C

```
LIB_EXPORT laResult laGroupBoxWidget_SetText(laGroupBoxWidget* box, laString str);
```

### Returns

void

### Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

### Parameters

| Parameters | Description |
|---|---|
| laGroupBoxWidget* box | the widget |
| laString str | an laString object |

### Function

void laGroupBoxWidget_SetText( laGroupBoxWidget* box, laString str)

## laImageSequenceWidget_GetImage Function

Gets the image asset pointer for an entry.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laImageSequenceWidget_GetImage(laImageSequenceWidget* img, uint32_t idx);
```

### Returns

GFXU_ImageAsset* - the image asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |

### Function

GFXU_ImageAsset* laImageSequenceWidget_GetImage(laImageSequenceWidget* img,
uint32_t idx)

## laImageSequenceWidget_GetImageChangedEventCallback Function

Gets the image changed event callback pointer.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laImageSequenceImageChangedEvent_FnPtr
laImageSequenceWidget_GetImageChangedEventCallback(laImageSequenceWidget* img);
```

### Returns

laImageSequenceImageChangedEvent_FnPtr - a valid callback pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

### Function

laImageSequenceImageChangedEvent_FnPtr laImageSequenceWidget_GetImageChangedEventCallback(laImageSequenceWidget* img)

## laImageSequenceWidget_GetImageCount Function

Gets the number of image entries for this widget.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT uint32_t laImageSequenceWidget_GetImageCount(laImageSequenceWidget* img);
```

### Returns

uint32_t - the number of entries for this sequence widget

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

### Function

uint32_t laImageSequenceWidget_GetImageCount( laImageSequenceWidget* img)

## laImageSequenceWidget_GetImageDelay Function

Gets the image delay for an entry.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT uint32_t laImageSequenceWidget_GetImageDelay(laImageSequenceWidget* img, uint32_t idx);
```

### Returns

uint32_t - the delay value

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |

### Function

uint32_t laImageSequenceWidget_GetImageDelay( laImageSequenceWidget* img,
uint32_t idx)

## laImageSequenceWidget_GetImageHAlignment Function

Gets the horizontal alignment for an image entry

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laHAlignment laImageSequenceWidget_GetImageHAlignment(laImageSequenceWidget* img, uint32_t idx);
```

### Returns

laHAlignment - the halign value

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |

### Function

laHAlignment laImageSequenceWidget_GetImageHAlignment(laImageSequenceWidget* img,
uint32_t idx)

## laImageSequenceWidget_GetImageVAlignment Function

Sets the vertical alignment for an image entry

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laVAlignment laImageSequenceWidget_GetImageVAlignment(laImageSequenceWidget* img, uint32_t idx);
```

### Returns

laVAlignment - the valign value

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |

### Function

laVAlignment laImageSequenceWidget_GetImageVAlignment(laImageSequenceWidget* img,
uint32_t idx)

## laImageSequenceWidget_GetRepeat Function

Indicates if the widget will repeat through the image entries.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laBool laImageSequenceWidget_GetRepeat(laImageSequenceWidget* img);
```

### Returns

laBool - indicates if the widget is automatically repeating

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

### Function

laBool laImageSequenceWidget_GetRepeat(laImageSequenceWidget* img)

## laImageSequenceWidget_IsPlaying Function

Indicates if the widget is currently cycling through the image entries.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laBool laImageSequenceWidget_IsPlaying(laImageSequenceWidget* img);
```

## Returns

laBool - indicates if the widget is automatically cycling

## Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

## Function

laBool laImageSequenceWidget_IsPlaying(laImageSequenceWidget* img)

### laImageSequenceWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

## File

libaria_widget_imagesequence.h

## C

```
LIB_EXPORT laImageSequenceWidget* laImageSequenceWidget_New();
```

## Returns

laImageSequenceWidget*

## Description

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

## Function

laImageSequenceWidget* laImageSequenceWidget_New()

### laImageSequenceWidget_Play Function

Starts the widget automatically cycling through the image entries.

## File

libaria_widget_imagesequence.h

## C

```
LIB_EXPORT laResult laImageSequenceWidget_Play(laImageSequenceWidget* img);
```

## Returns

laResult - the result of the operation

## Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

## Function

laResult laImageSequenceWidget_Play(laImageSequenceWidget* img)

### laImageSequenceWidget_Rewind Function

Resets the current image sequence display index to zero.

## File

libaria_widget_imagesequence.h

## C

```
LIB_EXPORT laResult laImageSequenceWidget_Rewind(laImageSequenceWidget* img);
```

## Returns

laResult - the result of the operation

## Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

## Function

laResult laImageSequenceWidget_Rewind(laImageSequenceWidget* img)

### laImageSequenceWidget_SetImage Function

Sets the image asset pointer for an entry.

## File

libaria_widget_imagesequence.h

## C

```
LIB_EXPORT laResult laImageSequenceWidget_SetImage(laImageSequenceWidget* img, uint32_t idx,
GFXU_ImageAsset* imgAst);
```

## Returns

laResult - the result of the operation

## Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |
| GFXU_ImageAsset* imgAst | the image asset pointer |

## Function

laResult laImageSequenceWidget_SetImage(laImageSequenceWidget* img,

uint32_t idx,

GFXU_ImageAsset* imgAst)

### laImageSequenceWidget_SetImageChangedEventCallback Function

Sets the image changed event callback pointer. This callback is called whenever the active display index is changed.

## File

libaria_widget_imagesequence.h

## C

```
LIB_EXPORT laResult laImageSequenceWidget_SetImageChangedEventCallback(laImageSequenceWidget* img,
laImageSequenceImageChangedEvent_FnPtr cb);
```

## Returns

laResult

## Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| laImageSequenceImageChangedEvent_FnPtr cb | a valid callback pointer or NULL |

## Function

laResult laImageSequenceWidget_SetImageChangedEventCallback(laImageSequenceWidget* img,

laImageSequenceImageChangedEvent_FnPtr cb)

## laImageSequenceWidget_SetImageCount Function

Sets the number of image entries for this widget. An image entry that is null will show nothing.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_SetImageCount(laImageSequenceWidget* img, uint32_t count);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t count | the desired number of entries |

### Function

laResult laImageSequenceWidget_SetImageCount(laImageSequenceWidget* img,
uint32_t count)

## laImageSequenceWidget_SetImageDelay Function

Sets the image delay for an entry.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_SetImageDelay(laImageSequenceWidget* img, uint32_t idx, uint32_t delay);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |
| uint32_t delay | the delay value |

### Function

laResult laImageSequenceWidget_SetImageDelay(laImageSequenceWidget* img,
uint32_t idx,
uint32_t delay)

## laImageSequenceWidget_SetImageHAlignment Function

Sets the horizontal alignment for an image entry.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_SetImageHAlignment(laImageSequenceWidget* img, uint32_t idx, laHAlignment align);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |
| laHAlignment align | the halign value |

### Function

laResult laImageSequenceWidget_SetImageHAlignment(laImageSequenceWidget* img,
uint32_t idx,
laHAlignment align)

## laImageSequenceWidget_SetImageVAlignment Function

Sets the vertical alignment value for an image entry

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_SetImageVAlignment(laImageSequenceWidget* img, uint32_t idx,
laVAlignment align);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the index |
| laVAlignment align | the vertical alignment setting |

### Function

laResult laImageSequenceWidget_SetImageVAlignment(laImageSequenceWidget* img,
uint32_t idx,
laVAlignment align)

## laImageSequenceWidget_SetRepeat Function

Sets the repeat flag for the widget

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_SetRepeat(laImageSequenceWidget* img, laBool repeat);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| laBool repeat | the desired repeat setting |

### Function

laResult laImageSequenceWidget_SetRepeat(laImageSequenceWidget* img,
laBool repeat)

## laImageSequenceWidget_ShowImage Function

Sets the active display index to the indicated value.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_ShowImage(laImageSequenceWidget* img, uint32_t idx);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |
| uint32_t idx | the desired index |

### Function

laResult laImageSequenceWidget_ShowImage(laImageSequenceWidget* img,
uint32_t idx)

## laImageSequenceWidget_ShowNextImage Function

Sets the active display index to the next index value.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_ShowNextImage(laImageSequenceWidget* img);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

### Function

laResult laImageSequenceWidget_ShowNextImage(laImageSequenceWidget* img)

## laImageSequenceWidget_ShowPreviousImage Function

Sets the active display index to the previous index value.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_ShowPreviousImage(laImageSequenceWidget* img);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

### Function

laResult laImageSequenceWidget_ShowPreviousImage(laImageSequenceWidget* img)

## laImageSequenceWidget_Stop Function

Stops the widget from automatically cycling through the image entries.

### File

libaria_widget_imagesequence.h

### C

```
LIB_EXPORT laResult laImageSequenceWidget_Stop(laImageSequenceWidget* img);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laImageSequenceWidget* img | the widget |

### Function

laResult laImageSequenceWidget_Stop(laImageSequenceWidget* img)

## laImageWidget_GetHAlignment Function

Gets the image horizontal alignment value.

### File

libaria_widget_image.h

### C

```
LIB_EXPORT laHAlignment laImageWidget_GetHAlignment(laImageWidget* img);
```

### Returns

laHAlignment - the horizontal alignment value

### Parameters

| Parameters | Description |
|---|---|
| laImageWidget* img | the widget |

### Function

laHAlignment laImageWidget_GetHAlignment(laImageWidget* img)

## laImageWidget_GetImage Function

Gets the image asset pointer for the widget.

### File

libaria_widget_image.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laImageWidget_GetImage(laImageWidget* img);
```

### Returns

GFXU_ImageAsset* - the image asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laImageWidget* img | the widget |

### Function

GFXU_ImageAsset* laImageWidget_GetImage(laImageWidget* img)

## laImageWidget_GetVAlignment Function

Gets the image vertical alignment value.

### File

libaria_widget_image.h

### C

```
LIB_EXPORT laVAlignment laImageWidget_GetVAlignment(laImageWidget* img);
```

### Returns

laVAlignment - the vertical alignment setting

### Parameters

| Parameters | Description |
|---|---|
| laImageWidget* img | the widget |

### Function

laVAlignment laImageWidget_GetVAlignment(laImageWidget* img)

## laImageWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_image.h

### C

```
LIB_EXPORT laImageWidget* laImageWidget_New();
```

### Returns

laImageWidget* - the widget

### Function

laImageWidget* laImageWidget_New()

## laImageWidget_SetHAlignment Function

Sets the image horizontal alignment value.

### File

libaria_widget_image.h

### C

```
LIB_EXPORT laResult laImageWidget_SetHAlignment(laImageWidget* img, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laImageWidget* img | the widget |
| laHAlignment align | the horizontal alignment value |

### Function

laResult laImageWidget_SetHAlignment(laImageWidget* img,

laHAlignment align)

### laImageWidget_SetImage Function

Sets the image asset pointer for the widget.

#### File

libaria_widget_image.h

#### C

```
LIB_EXPORT laResult laImageWidget_SetImage(laImageWidget* img, GFXU_ImageAsset* imgAst);
```

#### Returns

laResult - the operation result

#### Parameters

| Parameters | Description |
|---|---|
| llaImageWidget* img | the widget |
| GFXU_ImageAsset* imgAst | the image asset pointer |

#### Function

laResult laImageWidget_SetImage(laImageWidget* img,
GFXU_ImageAsset* imgAst)

### laImageWidget_SetVAlignment Function

Sets the image vertical alignment value.

#### File

libaria_widget_image.h

#### C

```
LIB_EXPORT laResult laImageWidget_SetVAlignment(laImageWidget* img, laVAlignment align);
```

#### Returns

laResult - the operation result

#### Parameters

| Parameters | Description |
|---|---|
| laImageWidget* img | the widget |
| laVAlignment | the vertical alignment setting |

#### Function

laResult laImageWidget_SetVAlignment(laImageWidget* img,
laVAlignment align)

### laInput_GetEnabled Function

Returns the input enabled status of the current context

#### File

libaria_input.h

#### C

```
LIB_EXPORT laBool laInput_GetEnabled();
```

#### Returns

laBool

#### Description

Returns the input enabled status of the current context

### Function

laBool laInput_GetEnabled()

## laInput_InjectTouchDown Function

Register and track the touch down event and queue it for handling by associated widgets

### File

libaria_input.h

### C

```
LIB_EXPORT laResult laInput_InjectTouchDown(uint32_t id, int32_t x, int32_t y);
```

### Returns

laResult

### Description

Register and track the touch down event and queue it for handling by associated widgets

### Function

laResult laInput_InjectTouchDown(uint32_t id, int32_t x, int32_t y)

## laInput_InjectTouchMoved Function

Register and track the touch moved event and queue it for handling by associated widgets

### File

libaria_input.h

### C

```
LIB_EXPORT laResult laInput_InjectTouchMoved(uint32_t id, int32_t x, int32_t y);
```

### Returns

laResult

### Description

Register and track the touch moved event and queue it for handling by associated widgets

### Function

laResult laInput_InjectTouchMoved(uint32_t id, int32_t x, int32_t y)

## laInput_InjectTouchUp Function

Register and track the touch up event and queue it for handling by associated widgets

### File

libaria_input.h

### C

```
LIB_EXPORT laResult laInput_InjectTouchUp(uint32_t id, int32_t x, int32_t y);
```

### Returns

laResult

### Description

Register and track the touch up event and queue it for handling by associated widgets

### Function

laResult laInput_InjectTouchUp(uint32_t id, int32_t x, int32_t y)

## laInput_SetEnabled Function

Sets the input status of the current context with the specified input argument

### File

libaria_input.h

### C

```
LIB_EXPORT laResult laInput_SetEnabled(laBool enable);
```

### Returns

laResult

### Description

Sets the input status of the current context with the specified input argument

### Function

laResult laInput_SetEnabled(laBool enable)

## laKeyPadWidget_GetKeyAction Function

Gets the key pad cell action for a cell at row/column

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laKeyPadCellAction laKeyPadWidget_GetKeyAction(laKeyPadWidget* pad, uint32_t row, uint32_t col);
```

### Returns

laKeyPadCellAction - the cell action value

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

### Function

laKeyPadCellAction laKeyPadWidget_GetKeyAction(laKeyPadWidget* pad,
uint32_t row,
uint32_t col)

## laKeyPadWidget_GetKeyClickEventCallback Function

Gets the current key click event callback pointer

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laKeyPadWidget_KeyClickEvent laKeyPadWidget_GetKeyClickEventCallback(laKeyPadWidget* pad);
```

### Returns

laKeyPadWidget_KeyClickEvent - the callback pointer

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |

## Function

laKeyPadWidget_KeyClickEvent laKeyPadWidget_GetKeyClickEventCallback(laKeyPadWidget* pad)

### laKeyPadWidget_GetKeyDrawBackground Function

Gets the background type for a key pad cell at row/column

## File

libaria_widget_keypad.h

## C

```
LIB_EXPORT laBackgroundType laKeyPadWidget_GetKeyDrawBackground(laKeyPadWidget* pad, uint32_t row, uint32_t
col);
```

## Returns

laBackgroundType - the cell background type

## Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

## Function

laBackgroundType laKeyPadWidget_GetKeyBackgroundType(laKeyPadWidget* pad,
uint32_t row,
uint32_t col)

### laKeyPadWidget_GetKeyEnabled Function

Gets the enabled flag for a cell at a given row/column

## File

libaria_widget_keypad.h

## C

```
LIB_EXPORT laBool laKeyPadWidget_GetKeyEnabled(laKeyPadWidget* pad, uint32_t row, uint32_t col);
```

## Returns

laBool - the flag value

## Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

## Function

laBool laKeyPadWidget_GetKeyEnabled(laKeyPadWidget* pad,
uint32_t row,
uint32_t col)

### laKeyPadWidget_GetKeyImageMargin Function

Gets the key pad cell image margin value

## File

libaria_widget_keypad.h

### C

```
LIB_EXPORT uint32_t laKeyPadWidget_GetKeyImageMargin(laKeyPadWidget* pad, uint32_t row, uint32_t col);
```

### Returns

uint32_t - the margin value

### Description

The image margin value is the space between the image and the text

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

### Function

uint32_t laKeyPadWidget_GetKeyImageMargin( laKeyPadWidget* pad,

uint32_t row,

uint32_t col)

## laKeyPadWidget_GetKeyImagePosition Function

Gets the image position for a key pad cell

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laRelativePosition laKeyPadWidget_GetKeyImagePosition(laKeyPadWidget* pad, uint32_t row,
uint32_t col);
```

### Returns

laRelativePosition - the image position

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

### Function

laRelativePosition laKeyPadWidget_GetKeyImagePosition(laKeyPadWidget* pad,

uint32_t row,

uint32_t col)

## laKeyPadWidget_GetKeyPressedImage Function

Gets the pressed icon image asset pointer for the display image for a key pad cell

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laKeyPadWidget_GetKeyPressedImage(laKeyPadWidget* pad, uint32_t row, uint32_t
col);
```

### Returns

GFXU_ImageAsset* - pointer to the icon image asset

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

### Function

GFXU_ImageAsset* laKeyPadWidget_GetKeyPressedImage(laKeyPadWidget* pad,

uint32_t row,

uint32_t col)

## laKeyPadWidget_GetKeyReleasedImage Function

Gets the released icon image asset pointer for the display image for a key pad cell

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laKeyPadWidget_GetKeyReleasedImage(laKeyPadWidget* pad, uint32_t row, uint32_t
col);
```

### Returns

GFXU_ImageAsset* - pointer to the icon image asset

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

### Function

GFXU_ImageAsset* laKeyPadWidget_GetKeyReleasedImage(laKeyPadWidget* pad,

uint32_t row,

uint32_t col)

## laKeyPadWidget_GetKeyText Function

Returns a copy of the display text for a given cell at row/column

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_GetKeyText(laKeyPadWidget* pad, uint32_t row, uint32_t col, laString*
str);
```

### Returns

laResult - the result of the operation

### Description

This function allocates memory for the input string argument. The application becomes responsible for the management of the memory after function completion.

The input string does not need to be initialized in any fashion before calling this function.

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |

| | |
|---|---|
| uint32_t col | the indicated column |
| laString* str | a pointer to an laString object |

### Function

laResult laKeyPadWidget_GetKeyText(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laString* str)

## laKeyPadWidget_GetKeyValue Function

Gets the edit text value for a given key pad cell.

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laString* laKeyPadWidget_GetKeyValue(laKeyPadWidget* pad, uint32_t row, uint32_t col);
```

### Returns

laString* - an initialized string containing a copy of the key pad cell edit value text

### Description

This function allocates memory and returns a valid laString pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |

### Function

laString* laKeyPadWidget_GetKeyValue(laKeyPadWidget* pad,
uint32_t row,
uint32_t col)

## laKeyPadWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laKeyPadWidget* laKeyPadWidget_New(uint32_t rows, uint32_t cols);
```

### Returns

laKeyPadWidget*

### Parameters

| Parameters | Description |
|---|---|
| uint32_t | number of rows to create number of columns to create |

### Function

laKeyPadWidget* laKeyPadWidget_New(uint32_t rows, uint32_t cols)

## laKeyPadWidget_SetKeyAction Function

Sets the cell action type for a key pad cell at row/column

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyAction(laKeyPadWidget* pad, uint32_t row, uint32_t col,
laKeyPadCellAction action);
```

### Returns

laResult - the result of the operation

### Description

The cell action is the action that is dispatched to the Aria edit event system. This event will then be received by the active edit event receptor widget if one exists.

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| laKeyPadCellAction action | the desired edit action |

### Function

laResult laKeyPadWidget_SetKeyAction(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laKeyPadCellAction action)

## laKeyPadWidget_SetKeyClickEventCallback Function

Sets the current key click event callback pointer

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyClickEventCallback(laKeyPadWidget* pad,
laKeyPadWidget_KeyClickEvent cb);
```

### Returns

laResult - the result of the operation

### Description

The key click event callback pointer is issued any time a button is interacted with.

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| laKeyPadWidget_KeyClickEvent cb | the callback pointer |

### Function

laResult laKeyPadWidget_SetKeyClickEventCallback(laKeyPadWidget* pad,
laKeyPadWidget_KeyClickEvent cb)

## laKeyPadWidget_SetKeyEnabled Function

Sets the enabled flag for a cell at the given row/column

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyEnabled(laKeyPadWidget* pad, uint32_t row, uint32_t col, laBool enabled);
```

### Returns

laResult - the result of the operation

### Description

The enabled flag controls the visibility and interactibility of a key pad cell. This enables the key pad to be configured to match such examples as a phone dialer key pad with twelve buttons total but the buttons to the left and right of the zero button not being drawn.

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| laBool enabled | the flag value |

### Function

laResult laKeyPadWidget_SetKeyEnabled(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laBool enabled)

## laKeyPadWidget_SetKeyImageMargin Function

Sets the key pad cell image margin value for a given cell at row/column

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyImageMargin(laKeyPadWidget* pad, uint32_t row, uint32_t col, uint32_t mg);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| uint32_t mg | the desired margin value |

### Function

laResult laKeyPadWidget_SetKeyImageMargin(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
uint32_t mg)

## laKeyPadWidget_SetKeyImagePosition Function

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyImagePosition(laKeyPadWidget* pad, uint32_t row, uint32_t col, laRelativePosition pos);
```

## Returns

laResult - the result of the operation

## Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| laRelativePosition pos | the desired image position |

## Function

laResult laKeyPadWidget_SetKeyImagePosition(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laRelativePosition pos)

### laKeyPadWidget_SetKeyPressedImage Function

Sets the pressed icon image asset pointer for a key pad cell

## File

libaria_widget_keypad.h

## C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyPressedImage(laKeyPadWidget* pad, uint32_t row, uint32_t col,
GFXU_ImageAsset* img);
```

## Returns

laResult - the result of the operation

## Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| GFXU_ImageAsset* img | pointer to an image asset |

## Function

laResult laKeyPadWidget_SetKeyPressedImage(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
GFXU_ImageAsset* img)

### laKeyPadWidget_SetKeyReleasedImage Function

Sets the released icon image asset pointer for a key pad cell

## File

libaria_widget_keypad.h

## C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyReleasedImage(laKeyPadWidget* pad, uint32_t row, uint32_t col,
GFXU_ImageAsset* img);
```

## Returns

laResult - the result of the operation

## Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| GFXU_ImageAsset* img | pointer to an image asset |

## Function

laResult laKeyPadWidget_SetKeyReleasedImage(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
GFXU_ImageAsset* img)

### laKeyPadWidget_SetKeyText Function

Sets the display text for a given cell at row/column

## File

libaria_widget_keypad.h

## C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyText(laKeyPadWidget* pad, uint32_t row, uint32_t col, laString str);
```

## Returns

laResult - the result of the operation

## Description

Sets the display text for a given cell at row/column

## Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| laString str | an laString object |

## Function

laResult laKeyPadWidget_SetKeyText(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laString str)

### laKeyPadWidget_SetKeyValue Function

Sets the edit value for a given key pad cell.

## File

libaria_widget_keypad.h

## C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyValue(laKeyPadWidget* pad, uint32_t row, uint32_t col, laString str);
```

## Returns

laResult - the result of the operation

## Description

The edit value for a key pad cell is the value that is passed to the Aria edit event management system. This may be different than the displayed

text of the cell or when the cell is using a picture icon and has no display text.

An input string that references the string table is a valid use case and the edit text will change as the active string table language changes.

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| laString str | the string to set the key value to |

### Function

laResult laKeyPadWidget_SetKeyValue(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laString str)

## laLabelWidget_GetHAlignment Function

Gets the text horizontal alignment value.

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laHAlignment laLabelWidget_GetHAlignment(laLabelWidget* lbl);
```

### Returns

laHAlignment - the horizontal alignment value

### Parameters

| Parameters | Description |
|---|---|
| laLabelWidget* | the widget |

### Function

laHAlignment laLabelWidget_GetHAlignment(laLabelWidget* lbl)

## laLabelWidget_GetText Function

Gets the text value for the label.

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laResult laLabelWidget_GetText(laLabelWidget* lbl, laString* str);
```

### Returns

laResult - the operation result

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laLabelWidget* lbl | the widget |
| laString* str | a pointer to an laString object |

### Function

laResult laLabelWidget_GetText(laLabelWidget* lbl, laString* str)

## laLabelWidget_GetVAlignment Function

Gets the current vertical text alignment

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laVAlignment laLabelWidget_GetVAlignment(laLabelWidget* lbl);
```

### Returns

laVAlignment - the vertical alignment setting

### Parameters

| Parameters | Description |
|---|---|
| laLabelWidget* | the widget |

### Function

laVAlignment laLabelWidget_GetVAlignment(laLabelWidget* lbl)

## laLabelWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laLabelWidget* laLabelWidget_New();
```

### Returns

laLabelWidget*

### Function

laLabelWidget* laLabelWidget_New()

## laLabelWidget_SetHAlignment Function

Sets the text horizontal alignment value

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laResult laLabelWidget_SetHAlignment(laLabelWidget* lbl, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laLabelWidget* | the widget |
| laHAlignment align | the horizontal alignment value |

### Function

laResult laLabelWidget_SetHAlignment(laLabelWidget* lbl,
laHAlignment align)

## laLabelWidget_SetText Function

Sets the text value for the label.

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laResult laLabelWidget_SetText(laLabelWidget* lbl, laString str);
```

### Returns

laResult - the operation result

### Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

### Parameters

| Parameters | Description |
|---|---|
| laLabelWidget* lbl | the widget |
| laString str | an laString object |

### Function

laResult laLabelWidget_SetText(laLabelWidget* lbl, laString str)

## laLabelWidget_SetVAlignment Function

Sets the vertical text alignment value

### File

libaria_widget_label.h

### C

```
LIB_EXPORT laResult laLabelWidget_SetVAlignment(laLabelWidget* lbl, laVAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laLabelWidget* | the widget |
| laVAlignment align | the vertical alignment setting |

### Function

laResult laLabelWidget_SetVAlignment(laLabelWidget* lbl,
laVAlignment align)

## laLayer_Delete Function

Destructor for the layer object

### File

libaria_layer.h

### C

```
LIB_EXPORT void laLayer_Delete(laLayer* layer);
```

### Returns

void

### Description

Destructor for the layer object

### Function

void laLayer_Delete( laLayer* layer)

## laLayer_GetAlphaAmount Function

Get's the amount of alpha blending for a given layer

### File

libaria_layer.h

### C

```
LIB_EXPORT uint32_t laLayer_GetAlphaAmount(const laLayer* layer);
```

### Returns

uint32_t - an alpha channel value from 0 - 255

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |

### Function

uint32_t laLayer_GetAlphaAmount(const laLayer* layer)

## laLayer_GetAlphaEnable Function

Gets the layer alpha enable flag

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_GetAlphaEnable(const laLayer* layer);
```

### Returns

laBool - the value of the alpha enable flag

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* | the layer |

### Function

laBool laLayer_GetAlphaEnable(const laLayer* layer)

## laLayer_GetBufferCount Function

Return the buffer count for the current layer

### File

libaria_layer.h

### C

```
LIB_EXPORT uint32_t laLayer_GetBufferCount(const laLayer* layer);
```

### Returns

uint32_t - the current number of buffers for the layer

### Description

Return the buffer count for the current layer

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |

### Function

uint32_t laLayer_GetBufferCount(const laLayer* layer)

## laLayer_GetMaskColor Function

Returns the mask color value for the current layer

### File

libaria_layer.h

### C

```
LIB_EXPORT GFX_Color laLayer_GetMaskColor(const laLayer* layer);
```

### Returns

GFX_Color - the layer mask color value

### Description

Returns the mask color value for the current layer

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |

### Function

GFX_Color laLayer_GetMaskColor(const laLayer* layer)

## laLayer_GetMaskEnable Function

Gets the layer mask enable flag

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_GetMaskEnable(const laLayer* layer);
```

### Returns

laBool - the value of the mask enable flag

### Description

Gets the layer mask enable flag

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |

### Function

laBool laLayer_GetMaskEnable(const laLayer* layer)

## laLayer_GetVSync Function

Gets the layer's vsync flag setting

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_GetVSync(const laLayer* layer);
```

### Returns

laBool - the state of the layer's vsync flag

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

laBool laLayer_GetVSync(const laLayer* layer)

## laLayer_New Function

Constructor for a new layer

### File

libaria_layer.h

### C

```
LIB_EXPORT laLayer* laLayer_New();
```

### Returns

laLayer*

### Description

Constructor for a new layer, returns the layer object

### Remarks

Allocates memory for a layer using the active context memory interface. Once added to a screen the it becomes the responsibility of the framework to free the memory.

### Function

laLayer* laLayer_New()

## laLayer_SetAlphaAmount Function

Set's the amount of alpha blending for a given layer

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetAlphaAmount(laLayer* layer, uint32_t amount);
```

### Returns

laResult - success if the operation succeeded

### Description

Set's the amount of alpha blending for a given layer

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |
| uint32_t amount | an alpha amount from 0 - 255 |

### Function

laResult laLayer_SetAlphaAmount(laLayer* layer, uint32_t amount)

## laLayer_SetAlphaEnable Function

Sets the layer alpha enable flag to the specified value

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetAlphaEnable(laLayer* layer, laBool enable);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |
| laBool enable | the desired value of the flag |

### Function

laResult laLayer_SetAlphaEnable(laLayer* layer, laBool enable)

## laLayer_SetBufferCount Function

Set the buffer count for the current layer to the specified value

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetBufferCount(laLayer* layer, uint32_t count);
```

### Returns

laResult - the result of the operation

### Description

Set the buffer count for the current layer to the specified value

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |
| uint32_t count | the desired number of buffers |

### Function

laResult laLayer_SetBufferCount(laLayer* layer, uint32_t count)

## laLayer_SetMaskColor Function

Set the mask color value for the current layer to the specified value

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetMaskColor(laLayer* layer, GFX_Color color);
```

### Returns

laResult - the result of the operation

### Description

Set the mask color value for the current layer to the specified value

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |
| GFX_color color | the desired mask color value |

### Function

void laLayer_SetMaskColor( laLayer* layer, GFX_Color color)

## laLayer_SetMaskEnable Function

Sets the layer mask enable flag to the specified value

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetMaskEnable(laLayer* layer, laBool enable);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |
| laBool enable | the desired value of the flag |

### Function

laResult laLayer_SetMaskEnable(laLayer* layer, laBool enable)

## laLayer_SetVSync Function

Sets the layer's vsync flag.

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetVSync(laLayer* layer, laBool enable);
```

### Returns

laResult - the result of the operation

### Description

Sets the layer's vsync flag.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

void laLayer_SetVSync( laLayer* layer, laBool enable)

## laLineWidget_GetEndPoint Function

Gets the coordinates for the second point of the line.

### File

libaria_widget_line.h

### C

```
LIB_EXPORT laResult laLineWidget_GetEndPoint(laLineWidget* line, int32_t* x, int32_t* y);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLineWidget* line | the widget |
| int32_t* x | pointer to an int to store the x coordinate |
| int32_t* y | pointer to tan int to store the y coordinate |

### Function

laResult laLineWidget_GetEndPoint(laLineWidget* line, int32_t* x, int32_t* y)

## laLineWidget_GetStartPoint Function

Gets the coordinates for the first point of the line.

### File

libaria_widget_line.h

### C

```
LIB_EXPORT laResult laLineWidget_GetStartPoint(laLineWidget* line, int32_t* x, int32_t* y);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLineWidget* line | the widget |
| int32_t* x | pointer to an int to store the x coordinate |
| int32_t* y | pointer to tan int to store the y coordinate |

### Function

laResult laLineWidget_GetStartPoint(laLineWidget* line, int32_t* x, int32_t* y)

## laLineWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_line.h

### C

```
LIB_EXPORT laLineWidget* laLineWidget_New();
```

### Returns

laLineWidget*

### Function

laLineWidget* laLineWidget_New()

## laLineWidget_SetEndPoint Function

Sets the cooridnate for the second point of the line

### File

libaria_widget_line.h

### C

```
LIB_EXPORT laResult laLineWidget_SetEndPoint(laLineWidget* line, int32_t x, int32_t y);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLineWidget* line | the widget |
| int32_t x | the x coordinate value |
| int32_t y | the y coordinate value |

### Function

laResult laLineWidget_SetEndPoint(laLineWidget* line, int32_t x, int32_t y)

## laLineWidget_SetStartPoint Function

Sets the cooridnate for the first point of the line

### File

libaria_widget_line.h

### C

```
LIB_EXPORT laResult laLineWidget_SetStartPoint(laLineWidget* line, int32_t x, int32_t y);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLineWidget* line | the widget |
| int32_t x | the x coordinate value |
| int32_t y | the y coordinate value |

### Function

laResult laLineWidget_SetStartPoint(laLineWidget* line, int32_t x, int32_t y)

## laListWheelWidget_AppendItem Function

Appends a new item entry to the list. The initial value of the item will be empty.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_AppendItem(laListWheelWidget* whl);
```

### Returns

uint32_t - the index of the newly appended item

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_AppendItem( laListWheelWidget* whl)

## laListWheelWidget_GetAlignment Function

Gets the horizontal alignment for the list widget

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laHAlignment laListWheelWidget_GetAlignment(laListWheelWidget* whl);
```

### Returns

laHAlignment - the current list halign mode

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laHAlignment laListWheelWidget_GetAlignment(laListWheelWidget* whl)

## laListWheelWidget_GetIconMargin Function

Gets the icon margin value for the list wheel widget

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetIconMargin(laListWheelWidget* whl);
```

### Returns

uint32_t - the icon margin value

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetIconMargin( laListWheelWidget* whl)

## laListWheelWidget_GetIconPosition Function

Sets the icon position for the list wheel widget.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laRelativePosition laListWheelWidget_GetIconPosition(laListWheelWidget* whl);
```

### Returns

laRelativePosition - the current icon position

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laRelativePosition laListWheelWidget_GetIconPosition(laListWheelWidget* whl)

## laListWheelWidget_GetItemCount Function

Gets the number of items currently contained in the list

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetItemCount(laListWheelWidget* whl);
```

### Returns

uint32_t - the number of items in the list

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetItemCount( laListWheelWidget* whl)

## laListWheelWidget_GetItemIcon Function

Gets the pointer to the image asset for the icon for the item at the given index.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laListWheelWidget_GetItemIcon(laListWheelWidget* whl, uint32_t index);
```

### Returns

GFXU_ImageAsset* - the image asset pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the index to consider |

### Function

GFXU_ImageAsset* laListWheelWidget_GetItemIcon(laListWheelWidget* whl,
uint32_t index)

## laListWheelWidget_GetItemText Function

Gets the text value for an item in the list.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_GetItemText(laListWheelWidget* whl, uint32_t idx, laString* str);
```

### Returns

laResult - the operation result

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the index to consider |
| laString* str | a pointer to an laString object |

## Function

laResult laListWheelWidget_GetItemText(laListWheelWidget* whl,
uint32_t idx,
laString* str)

### laListWheelWidget_GetSelectedItem Function

Returns the index of the currently selected item.

## File

libaria_widget_listwheel.h

## C

```
LIB_EXPORT int32_t laListWheelWidget_GetSelectedItem(laListWheelWidget* whl);
```

## Returns

int32_t - the index of the selected item or -1 if an error occured

## Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

## Function

int32_t laListWheelWidget_GetSelectedItem( laListWheelWidget* whl)

### laListWheelWidget_GetSelectedItemChangedEventCallback Function

Gets the callback for the item selected changed event

## File

libaria_widget_listwheel.h

## C

```
LIB_EXPORT laListWheelWidget_SelectedItemChangedEvent
laListWheelWidget_GetSelectedItemChangedEventCallback(laListWheelWidget* whl);
```

## Returns

laListWheelWidget_SelectedItemChangedEvent - the current pointer to the callback or NULL

## Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

## Function

laListWheelWidget_SelectedItemChangedEvent laListWheelWidget_GetSelectedItemChangedEventCallback(laListWheelWidget* whl)

### laListWheelWidget_InsertItem Function

Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right.

## File

libaria_widget_listwheel.h

## C

```
LIB_EXPORT uint32_t laListWheelWidget_InsertItem(laListWheelWidget* whl, uint32_t idx);
```

## Returns

uint32_t - the index of the inserted item

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the desired index of the new item |

### Function

uint32_t laListWheelWidget_InsertItem( laListWheelWidget* whl, uint32_t idx)

## laListWheelWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laListWheelWidget* laListWheelWidget_New();
```

### Returns

laListWheelWidget*

### Function

laListWheelWidget* laListWheelWidget_New()

## laListWheelWidget_RemoveAllItems Function

Attempts to remove all items from the list.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_RemoveAllItems(laListWheelWidget* whl);
```

### Returns

laResult - the operation result

### Remarks

All memory owned by each item string will be freed automatically.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laResult laListWheelWidget_RemoveAllItems(laListWheelWidget* whl)

## laListWheelWidget_RemoveItem Function

Attempts to remove an item from the list.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_RemoveItem(laListWheelWidget* whl, uint32_t idx);
```

### Returns

laResult - the operation result

### Remarks

The memory owned by the string item will be freed automatically.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the index to remove from the list |

### Function

laResult laListWheelWidget_RemoveItem(laListWheelWidget* whl, uint32_t idx)

## laListWheelWidget_SelectNextItem Function

Attempts to move the selected item index to the next item in the list.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SelectNextItem(laListWheelWidget* whl);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laResult laListWheelWidget_SelectNextItem(laListWheelWidget* whl)

## laListWheelWidget_SelectPreviousItem Function

Attempts to move the selected item index to the previous item in the list.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SelectPreviousItem(laListWheelWidget* whl);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laResult laListWheelWidget_SelectPreviousItem(laListWheelWidget* whl)

## laListWheelWidget_SetAlignment Function

Sets the horizontal alignment mode for the list widget.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetAlignment(laListWheelWidget* whl, laHAlignment align);
```

### Returns

laResult

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| laHAlignment align | the desired halign mode |

### Function

laResult laListWheelWidget_SetAlignment(laListWheelWidget* whl,
laHAlignment align)

## laListWheelWidget_SetIconMargin Function

Sets the icon margin value for the list widget.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetIconMargin(laListWheelWidget* whl, uint32_t mg);
```

### Returns

laResult - the operation result

### Description

The icon margin value is the distance between the icon image and the text.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t mg | the margin value |

### Function

laResult laListWheelWidget_SetIconMargin(laListWheelWidget* whl, uint32_t mg)

## laListWheelWidget_SetIconPosition Function

Sets the icon position for the list wheel widget

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetIconPosition(laListWheelWidget* whl, laRelativePosition pos);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| laRelativePosition pos | the relative position setting |

### Function

laResult laListWheelWidget_SetIconPosition(laListWheelWidget* whl,
laRelativePosition pos)

## laListWheelWidget_SetItemIcon Function

Sets the icon pointer for a given index.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetItemIcon(laListWheelWidget* whl, uint32_t index, GFXU_ImageAsset*
img);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the index to configure |
| GFXU_ImageAsset* | the image asset pointer to use as the icon |

### Function

laResult laListWheelWidget_SetItemIcon(laListWheelWidget* whl,
uint32_t index,
GFXU_ImageAsset* img)

## laListWheelWidget_SetItemText Function

Sets the text value for an item in the list.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetItemText(laListWheelWidget* whl, uint32_t index, laString str);
```

### Returns

laResult - the operation result

### Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the index to consider |
| laString str | an laString object |

### Function

laResult laListWheelWidget_SetItemText(laListWheelWidget* whl,
uint32_t index,
laString str)

## laListWheelWidget_SetSelectedItem Function

Attempts to set the selectedi tem index

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetSelectedItem(laListWheelWidget* whl, uint32_t idx);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t idx | the desired selected item index |

### Function

laResult laListWheelWidget_SetSelectedItem(laListWheelWidget* whl,
uint32_t idx)

## laListWheelWidget_SetSelectedItemChangedEventCallback Function

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetSelectedItemChangedEventCallback(laListWheelWidget* whl,
laListWheelWidget_SelectedItemChangedEvent cb);
```

### Returns

laResult - the operation result

### Description

This callback is called whenver the wheel's selected item changes.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| laListWheelWidget_SelectedItemChangedEvent | the desired pointer to callback or NULL |

### Function

laResult laListWheelWidget_SetSelectedItemChangedEventCallback(laListWheelWidget* whl,
laListWheelWidget_SelectedItemChangedEvent cb)

## laListWidget_AppendItem Function

Appends a new item entry to the list. The initial value of the item will be empty.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_AppendItem(laListWidget* lst);
```

### Returns

uint32_t - the index of the newly appended item

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWidget_AppendItem( laListWidget* lst)

## laListWidget_DeselectAll Function

Attempts to set all item states as not selected.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_DeselectAll(laListWidget* lst);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laResult laListWidget_DeselectAll(laListWidget* lst)

## laListWidget_GetAlignment Function

Gets the horizontal alignment for the list widget

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laHAlignment laListWidget_GetAlignment(laListWidget* lst);
```

### Returns

laHAlignment - the current list halign mode

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laHAlignment laListWidget_GetAlignment(laListWidget* lst)

## laListWidget_GetAllowEmptySelection Function

Returns true of the list allows an empty selection set

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laBool laListWidget_GetAllowEmptySelection(laListWidget* lst);
```

### Returns

laBool - true if the list allows an empty selection set

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laBool laListWidget_GetAllowEmptySelection(laListWidget* lst)

## laListWidget_GetFirstSelectedItem Function

Returns the lowest selected item index.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_GetFirstSelectedItem(laListWidget* lst);
```

### Returns

uint32_t - the lowest selected item index or -1 if nothing is selected.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWidget_GetFirstSelectedItem( laListWidget* lst)

## laListWidget_GetIconMargin Function

Gets the icon margin value for the list widget

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_GetIconMargin(laListWidget* lst);
```

### Returns

uint32_t - the icon margin value

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWidget_GetIconMargin( laListWidget* lst)

## laListWidget_GetIconPosition Function

Gets the icon position for the list

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laRelativePosition laListWidget_GetIconPosition(laListWidget* lst);
```

### Returns

laRelativePosition - the current icon position

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laRelativePosition laListWidget_GetIconPosition(laListWidget* lst)

## laListWidget_GetItemCount Function

Gets the number of items currently contained in the list

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_GetItemCount(laListWidget* lst);
```

### Returns

uint32_t - the number of items in the list

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWidget_GetItemCount( laListWidget* lst)

## laListWidget_GetItemIcon Function

Gets the pointer to the image asset for the icon for the item at the given index.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laListWidget_GetItemIcon(laListWidget* lst, uint32_t idx);
```

### Returns

GFXU_ImageAsset* - the image asset pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to consider |

### Function

GFXU_ImageAsset* laListWidget_GetItemIcon(laListWidget* lst,
uint32_t idx)

## laListWidget_GetItemSelected Function

Returns true if the item at the given index is currently selected.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laBool laListWidget_GetItemSelected(laListWidget* lst, uint32_t idx);
```

### Returns

laBool - the selection state of the item

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to consider |

### Function

laBool laListWidget_GetItemSelected(laListWidget* lst,
uint32_t idx)

## laListWidget_GetItemText Function

Gets the text value for an item in the list.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_GetItemText(laListWidget* lst, uint32_t idx, laString* str);
```

### Returns

laResult - the operation result

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to consider |
| laString* str | a pointer to an laString object |

### Function

laResult laListWidget_GetItemText(laListWidget* lst,
uint32_t idx,
laString* str)

## laListWidget_GetLastSelectedItem Function

Returns the highest selected item index.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_GetLastSelectedItem(laListWidget* lst);
```

### Returns

uint32_t - the highest selected item index or -1 if nothing is selected.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWidget_GetLastSelectedItem( laListWidget* lst)

## laListWidget_GetSelectedItemChangedEventCallback Function

Gets the callback for the item selected changed event

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laListWidget_SelectedItemChangedEvent
```

```
laListWidget_GetSelectedItemChangedEventCallback(laListWidget* lst);
```

### Returns

laListWidget_SelectedItemChangedEvent - the current pointer to callback or NULL

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laListWidget_SelectedItemChangedEvent laListWidget_GetSelectedItemChangedEventCallback(laListWidget* lst)

## laListWidget_GetSelectionCount Function

Returns the number of selected items in the list.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_GetSelectionCount(laListWidget* lst);
```

### Returns

uint32_t - the number of selected items

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWidget_GetSelectionCount( laListWidget* lst)

## laListWidget_GetSelectionMode Function

Gets the selection mode for the list

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laListWidget_SelectionMode laListWidget_GetSelectionMode(laListWidget* lst);
```

### Returns

laListWidget_SelectionMode - the list selection mode

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laListWidget_SelectionMode laListWidget_GetSelectionMode(laListWidget* lst)

## laListWidget_InsertItem Function

Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT uint32_t laListWidget_InsertItem(laListWidget* lst, uint32_t idx);
```

### Returns

uint32_t - the index of the inserted item

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the desired index of the new item |

### Function

uint32_t laListWidget_InsertItem( laListWidget* lst, uint32_t idx)

## laListWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laListWidget* laListWidget_New();
```

### Returns

laListWidget* lst - the widget

### Function

laListWidget* laListWidget_New()

## laListWidget_RemoveAllItems Function

Attempts to remove all items from the list.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_RemoveAllItems(laListWidget* lst);
```

### Returns

laResult - the operation result

### Remarks

All memory owned by each item string will be freed automatically.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laResult laListWidget_RemoveAllItems(laListWidget* lst)

## laListWidget_RemoveItem Function

Attempts to remove an item from the list.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_RemoveItem(laListWidget* lst, uint32_t idx);
```

### Returns

laResult - the operation result

### Remarks

The memory owned by the string item will be freed automatically.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to remove from the list |

### Function

laResult laListWidget_RemoveItem(laListWidget* lst, uint32_t idx)

## laListWidget_SelectAll Function

Attempts to set all item states to selected.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SelectAll(laListWidget* lst);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

laResult laListWidget_SelectAll(laListWidget* lst)

## laListWidget_SetAlignment Function

Sets the horizontal alignment mode for the list widget.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetAlignment(laListWidget* lst, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| laHAlignment align | the desired halign mode |

### Function

laResult laListWidget_SetAlignment(laListWidget* lst,
laHAlignment align)

## laListWidget_SetAllowEmptySelection Function

Configures the list to allow an empty selection set.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetAllowEmptySelection(laListWidget* lst, laBool allow);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| laBool allow | the desired empty selection set mode |

### Function

laResult laListWidget_SetAllowEmptySelection(laListWidget* lst,
laBool allow)

## laListWidget_SetIconMargin Function

Sets the icon margin value for the list widget.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetIconMargin(laListWidget* lst, uint32_t mg);
```

### Returns

laResult - the operation result

### Description

The icon margin value is the distance between the icon image and the text.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t mg | the margin value |

### Function

laResult laListWidget_SetIconMargin(laListWidget* lst, uint32_t mg)

## laListWidget_SetIconPosition Function

Sets the icon position for the list widget

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetIconPosition(laListWidget* lst, laRelativePosition pos);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| laRelativePosition pos | the relative position setting |

### Function

laResult laListWidget_SetIconPosition(laListWidget* lst,
laRelativePosition pos)

## laListWidget_SetItemIcon Function

Sets the icon pointer for a given index.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetItemIcon(laListWidget* lst, uint32_t idx, GFXU_ImageAsset* img);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to configure |
| GFXU_ImageAsset* | the image asset pointer to use as the icon |

### Function

laResult laListWidget_SetItemIcon(laListWidget* lst,
uint32_t idx,
GFXU_ImageAsset* img)

## laListWidget_SetItemSelected Function

Attempts to set the item at idx as selected.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetItemSelected(laListWidget* lst, uint32_t idx, laBool selected);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to consider |
| laBool | the select state to set to the item |

### Function

laResult laListWidget_SetItemSelected(laListWidget* lst,
uint32_t idx,
laBool selected)

## laListWidget_SetItemText Function

Sets the text value for an item in the list.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetItemText(laListWidget* lst, uint32_t index, laString str);
```

### Returns

laResult - the operation result

### Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to consider |
| laString str | an laString object |

### Function

laResult laListWidget_SetItemText(laListWidget* lst,
uint32_t index,
laString str)

## laListWidget_SetItemVisible Function

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetItemVisible(laListWidget* lst, uint32_t idx);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t index | the index to modify |

### Function

laResult laListWidget_SetItemVisible(laListWidget* lst,
uint32_t index)

## laListWidget_SetSelectedItemChangedEventCallback Function

Sets the callback for the item selected changed event

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetSelectedItemChangedEventCallback(laListWidget* lst,
laListWidget_SelectedItemChangedEvent cb);
```

### Returns

laResult - the operation result

### Description

This callback is called whenver an items selected state is modified.

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| laListWidget_SelectedItemChangedEvent | the desired pointer to callback or NULL |

### Function

laResult laListWidget_SetSelectedItemChangedEventCallback(laListWidget* lst,
laListWidget_SelectedItemChangedEvent cb)

## laListWidget_SetSelectionMode Function

Set the list selection mode

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_SetSelectionMode(laListWidget* lst, laListWidget_SelectionMode mode);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| laListWidget_SelectionMode mode | the desired list selection mode |

### Function

laResult laListWidget_SetSelectionMode(laListWidget* lst,
laListWidget_SelectionMode mode)

## laListWidget_ToggleItemSelected Function

Attempts to toggle the selected state of the item at idx.

### File

libaria_widget_list.h

### C

```
LIB_EXPORT laResult laListWidget_ToggleItemSelected(laListWidget* lst, uint32_t idx);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t idx | the index to consider |

### Function

laResult laListWidget_ToggleItemSelected(laListWidget* lst,
uint32_t idx)

## laProgressBarWidget_GetDirection Function

Gets the fill direction value for a progress bar widget

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT laProgressBarDirection laProgressBarWidget_GetDirection(laProgressBarWidget* bar);
```

### Returns

laProgressBarDirection - the fill direction value

### Parameters

| Parameters | Description |
|---|---|
| laProgressBarWidget* bar | the widget |

### Function

laProgressBarDirection laProgressBarWidget_GetDirection(laProgressBarWidget* bar)

## laProgressBarWidget_GetValue Function

Gets the percentage value for a progress bar.

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT uint32_t laProgressBarWidget_GetValue(laProgressBarWidget* bar);
```

### Returns

uint32_t

### Parameters

| Parameters | Description |
|---|---|
| laProgressBarWidget* bar | the widget |

### Function

uint32_t laProgressBarWidget_GetValue( laProgressBarWidget* bar)

## laProgressBarWidget_GetValueChangedEventCallback Function

Gets the currently set value changed event callback.

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT laProgressBar_ValueChangedEventCallback
laProgressBarWidget_GetValueChangedEventCallback(laProgressBarWidget* bar);
```

### Returns

laProgressBar_ValueChangedEventCallback - the current callback pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laProgressBarWidget* bar | the widget |

### Function

laProgressBar_ValueChangedEventCallback laProgressBarWidget_GetValueChangedEventCallback(laProgressBarWidget* bar)

## laProgressBarWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT laProgressBarWidget* laProgressBarWidget_New();
```

### Returns

laProgressBarWidget*

### Function

laProgressBarWidget* laProgressBarWidget_New()

## laProgressBarWidget_SetDirection Function

Sets the fill direction for a progress bar widget

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT laResult laProgressBarWidget_SetDirection(laProgressBarWidget* bar, laProgressBarDirection dir);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laProgressBarWidget* bar | the widget |
| laProgressBarDirection dir | the desired fill direction |

### Function

laResult laProgressBarWidget_SetDirection(laProgressBarWidget* bar,
laProgressBarDirection dir)

## laProgressBarWidget_SetValue Function

Sets the percentage value for a progress bar. Valid values are 0 - 100.

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT laResult laProgressBarWidget_SetValue(laProgressBarWidget* bar, uint32_t value);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laProgressBarWidget* bar | the widget |
| uint32_t value | the desired value |

### Function

laResult laProgressBarWidget_SetValue(laProgressBarWidget* bar, uint32_t value)

## laProgressBarWidget_SetValueChangedCallback Function

Sets the desired value changed event callback pointer

### File

libaria_widget_progressbar.h

### C

```
LIB_EXPORT laResult laProgressBarWidget_SetValueChangedCallback(laProgressBarWidget* bar,
laProgressBar_ValueChangedEventCallback cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laProgressBarWidget* bar | the widget |
| laProgressBar_ValueChangedEventCallback | a valid callback pointer or NULL |

### Function

laResult laProgressBarWidget_SetValueChangedCallback(laProgressBarWidget* bar, laProgressBar_ValueChangedEventCallback cb)

## laRadioButtonGroup_AddButton Function

Add a button widget to the button list of the selected Radio button group.

### File

libaria_radiobutton_group.h

### C

```
LIB_EXPORT laResult laRadioButtonGroup_AddButton(laRadioButtonGroup* grp, laRadioButtonWidget* btn);
```

### Returns

laResult

### Description

Add a button widget to the button list of the selected Radio button group. The function makes sure the radio button grp is valid and the button widget to be added is not already a part of the group. The button is then added as the last button in the group button list

### Function

laResult laRadioButtonGroup_AddButton(laRadioButtonGroup* grp,
laRadioButtonWidget* btn)

## laRadioButtonGroup_Create Function

This function creates a GFX_GOL_RADIOBUTTON group with the provided button list.

### File

libaria_radiobutton_group.h

### C

```
LIB_EXPORT laResult laRadioButtonGroup_Create(laRadioButtonGroup** grp);
```

### Returns

laResult

### Description

This function creates a GFX_GOL_RADIOBUTTON group with the given pointer and the button list provided within the laRadioButtonGroup object.

### Function

laResult laRadioButtonGroup_Create(laRadioButtonGroup** grp)

## laRadioButtonGroup_Destroy Function

This function destroys the GFX_GOL_RADIOBUTTON group

### File

libaria_radiobutton_group.h

### C

```
LIB_EXPORT void laRadioButtonGroup_Destroy(laRadioButtonGroup* grp);
```

### Returns

void

### Description

This function destroys he GFX_GOL_RADIOBUTTON group with the given pointer. It frees the memory allocated to the button group and clears the button list.

### Function

void laRadioButtonGroup_Destroy( laRadioButtonGroup* grp)

## laRadioButtonGroup_RemoveButton Function

Remove a button widget to the button list of the selected Radio button group.

### File

libaria_radiobutton_group.h

### C

```
LIB_EXPORT laResult laRadioButtonGroup_RemoveButton(laRadioButtonGroup* grp, laRadioButtonWidget* btn);
```

### Returns

laResult

### Description

Remove a button widget to the button list of the selected Radio button group. The function makes sure the radio button grp is valid and the button widget to be removed is a part of the group. The button is then removed properly making sure to handle the list correctly. If the list size is 0, the group is destroyed.

### Function

laResult laRadioButtonGroup_RemoveButton(laRadioButtonGroup* grp,
laRadioButtonWidget* btn);

## laRadioButtonGroup_SelectButton Function

Select the button widget specified from the button list for the Radio button group.

### File

libaria_radiobutton_group.h

### C

```
LIB_EXPORT laResult laRadioButtonGroup_SelectButton(laRadioButtonGroup* grp, laRadioButtonWidget* btn);
```

### Returns

laResult

## Description

Select the button widget specified from the button list for the Radio button group. The function makes sure the specified button widget is a part of the group. It deselects the currently selected button widget and reassigns the focus to the button widget specified.

## Function

laResult laRadioButtonGroup_SelectButton(laRadioButtonGroup* grp,
laRadioButtonWidget* btn)

### laRadioButtonWidget_GetDeselectedEventCallback Function

Gets the current radio button deselected event callback

## File

libaria_widget_radiobutton.h

## C

```
LIB_EXPORT laRadioButtonWidget_DeselectedEvent
laRadioButtonWidget_GetDeselectedEventCallback(laRadioButtonWidget* btn);
```

## Returns

laRadioButtonWidget_DeselectedEvent - a valid callback pointer or NULL

## Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

## Function

laRadioButtonWidget_DeselectedEvent laRadioButtonWidget_GetDeselectedEventCallback(laRadioButtonWidget* btn)

### laRadioButtonWidget_GetGroup Function

Returns the pointer to the currently set radio button group.

## File

libaria_widget_radiobutton.h

## C

```
LIB_EXPORT laRadioButtonGroup* laRadioButtonWidget_GetGroup(laRadioButtonWidget* btn);
```

## Returns

laRadioButtonGroup* - the currently assigned radio button group

## Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

## Function

laRadioButtonGroup* laRadioButtonWidget_GetGroup(laRadioButtonWidget* btn)

### laRadioButtonWidget_GetHAlignment Function

Gets the horizontal alignment setting for a button

## File

libaria_widget_radiobutton.h

## C

```
LIB_EXPORT laHAlignment laRadioButtonWidget_GetHAlignment(laRadioButtonWidget* btn);
```

### Returns

laHAlignment - the horizontal alignment value

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

laHAlignment laRadioButtonWidget_GetHAlignment(laRadioButtonWidget* btn)

## laRadioButtonWidget_GetImageMargin Function

Gets the distance between the icon and the text

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT uint32_t laRadioButtonWidget_GetImageMargin(laRadioButtonWidget* btn);
```

### Returns

uint32_t - the distance value

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

uint16_t laRadioButtonWidget_GetImageMargin( laRadioButtonWidget* btn)

## laRadioButtonWidget_GetImagePosition Function

Gets the current image position setting for the radio button

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laRelativePosition laRadioButtonWidget_GetImagePosition(laRadioButtonWidget* btn);
```

### Returns

laRelativePosition - the current image relative position

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

laRelativePosition laRadioButtonWidget_GetImagePosition(laRadioButtonWidget* btn)

## laRadioButtonWidget_GetSelected Function

Returns true if this radio button is currently selected

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laBool laRadioButtonWidget_GetSelected(laRadioButtonWidget* btn);
```

### Returns

laBool - true if this button is currently selected

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

laBool laRadioButtonWidget_GetSelected(laRadioButtonWidget* btn)

## laRadioButtonWidget_GetSelectedEventCallback Function

Gets the current radio button selected event callback

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laRadioButtonWidget_SelectedEvent
laRadioButtonWidget_GetSelectedEventCallback(laRadioButtonWidget* btn);
```

### Returns

laRadioButtonWidget_SelectedEvent - a valid callback pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

laRadioButtonWidget_SelectedEvent laRadioButtonWidget_GetSelectedEventCallback(laRadioButtonWidget* btn)

## laRadioButtonWidget_GetSelectedImage Function

Gets the selected image asset pointer for a button

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laRadioButtonWidget_GetSelectedImage(laRadioButtonWidget* btn);
```

### Returns

GFXU_ImageAsset* - the selected asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

GFXU_ImageAsset* laRadioButtonWidget_GetSelectedImage(laRadioButtonWidget* btn)

## laRadioButtonWidget_GetText Function

Gets the text value for the button.

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_GetText(laRadioButtonWidget* btn, laString* str);
```

### Returns

laResult - the operation result

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laString* str | a pointer to an laString object |

### Function

laResult laRadioButtonWidget_GetText(laRadioButtonWidget* btn,
laString* str)

## laRadioButtonWidget_GetUnselectedImage Function

Gets the image asset pointer currently used as the unselected icon

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laRadioButtonWidget_GetUnselectedImage(laRadioButtonWidget* btn);
```

### Returns

GFXU_ImageAsset* - the selected asset pointer

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

GFXU_ImageAsset* laRadioButtonWidget_GetUnselectedImage(laRadioButtonWidget* btn)

## laRadioButtonWidget_GetVAlignment Function

Sets the vertical alignment for a button

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laVAlignment laRadioButtonWidget_GetVAlignment(laRadioButtonWidget* btn);
```

### Returns

laVAlignment align - the desired vertical alignment setting

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

laVAlignment laRadioButtonWidget_GetVAlignment(laRadioButtonWidget* btn)

## laRadioButtonWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laRadioButtonWidget* laRadioButtonWidget_New();
```

### Returns

laProgressBarWidget*

### Function

laRadioButtonWidget* laRadioButtonWidget_New()

## laRadioButtonWidget_SetDeselectedEventCallback Function

Sets the deselected callback pointer

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetDeselectedEventCallback(laRadioButtonWidget* btn,
laRadioButtonWidget_DeselectedEvent cb);
```

### Returns

laResult - the operation result

### Description

This callback is called when this radio button is deselected

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laRadioButtonWidget_DeselectedEvent | a valid callback pointer or NULL |

### Function

laResult laRadioButtonWidget_SetDeselectedEventCallback(laRadioButtonWidget* btn,
laRadioButtonWidget_DeselectedEvent cb)

## laRadioButtonWidget_SetHAlignment Function

Sets the horizontal alignment value for a button

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetHAlignment(laRadioButtonWidget* btn, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laHAlignment align | the desired alignment value |

### Function

laResult laRadioButtonWidget_SetHAlignment(laRadioButtonWidget* btn,
laHAlignment align)

## laRadioButtonWidget_SetImagePosition Function

Sets the image relative position setting for the radio button

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetImagePosition(laRadioButtonWidget* btn, laRelativePosition pos);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laRelativePosition pos | the desired image position |

### Function

laResult laRadioButtonWidget_SetImagePosition(laRadioButtonWidget* btn,
laRelativePosition pos)

## laRadioButtonWidget_SetSelected Function

Sets this button as selected.

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetSelected(laRadioButtonWidget* btn);
```

### Returns

laResult - the operation result

### Description

If this button belongs to a radio button group then this function will potentially unselect another button and become selected.

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |

### Function

laResult laRadioButtonWidget_SetSelected(laRadioButtonWidget* btn)

## laRadioButtonWidget_SetSelectedEventCallback Function

Sets the radio button selected event callback

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetSelectedEventCallback(laRadioButtonWidget* btn,
laRadioButtonWidget_SelectedEvent cb);
```

### Returns

laResult - the operation result

## Description

This callback is called when the radio button becomes selected

## Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laRadioButtonWidget_SelectedEvent | a valid callback pointer or NULL |

## Function

laResult laRadioButtonWidget_SetSelectedEventCallback(laRadioButtonWidget* btn,
laRadioButtonWidget_SelectedEvent cb)

### laRadioButtonWidget_SetSelectedImage Function

Sets the image to be used as a selected icon

## File

libaria_widget_radiobutton.h

## C

```
LIB_EXPORT laResult laRadioButtonWidget_SetSelectedImage(laRadioButtonWidget* btn, GFXU_ImageAsset* img);
```

## Returns

laResult - the operation result

## Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| GFXU_ImageAsset* img | the desired image asset pointer or NULL |

## Function

laResult laRadioButtonWidget_SetSelectedImage(laRadioButtonWidget* btn,
GFXU_ImageAsset* img)

### laRadioButtonWidget_SetText Function

Sets the text value for the button.

## File

libaria_widget_radiobutton.h

## C

```
LIB_EXPORT laResult laRadioButtonWidget_SetText(laRadioButtonWidget* btn, laString str);
```

## Returns

laResult - the operation result

## Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

## Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laString str | an laString object |

## Function

laResult laRadioButtonWidget_SetText(laRadioButtonWidget* btn,
laString str)

## laRadioButtonWidget_SetUnselectedImage Function

Sets the asset pointer for the radio button's unselected image icon

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetUnselectedImage(laRadioButtonWidget* btn, GFXU_ImageAsset* img);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| GFXU_ImageAsset* img | the desired image asset pointer or NULL |

### Function

laResult laRadioButtonWidget_SetUnselectedImage(laRadioButtonWidget* btn,
GFXU_ImageAsset* img)

## laRadioButtonWidget_SetVAlignment Function

Sets the vertical alignment for a button

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetVAlignment(laRadioButtonWidget* btn, laVAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| laVAlignment align | the desired vertical alignment setting |

### Function

laResult laRadioButtonWidget_SetVAlignment(laRadioButtonWidget* btn,
laVAlignment align)

## laRectangleWidget_GetThickness Function

Gets the rectangle border thickness setting

### File

libaria_widget_rectangle.h

### C

```
LIB_EXPORT int32_t laRectangleWidget_GetThickness(laRectangleWidget* rect);
```

### Returns

int32_t - the border thickness setting

### Parameters

| Parameters | Description |
|---|---|
| laRectangleWidget* rect | the widget |

### Function

int32_t laRectangleWidget_GetThickness( laRectangleWidget* rect)

## laRectangleWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_rectangle.h

### C

```
LIB_EXPORT laRectangleWidget* laRectangleWidget_New();
```

### Returns

laRectangleWidget*

### Function

laRectangleWidget* laRectangleWidget_New()

## laScheme_Initialize Function

Initialize the scheme to the default values as per the specified color mode.

### File

libaria_scheme.h

### C

```
LIB_EXPORT void laScheme_Initialize(laScheme* scheme, GFX_ColorMode mode);
```

### Returns

void

### Description

Initialize the scheme to the default values as per the specified color mode.

### Parameters

| Parameters | Description |
|---|---|
| laSceme* scheme | the scheme to modify |
| GFX_ColorMode | the color mode to use |

### Function

void laScheme_Initialize( laScheme* scheme, GFX_ColorMode mode)

## laScreen_Delete Function

Frees all memory for all layers and widgets for this screen

### File

libaria_screen.h

### C

```
LIB_EXPORT void laScreen_Delete(laScreen* scr);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to destroy |

### Function

void laScreen_Delete( laScreen* scr)

## laScreen_GetHideEventCallback Function

Returns the hide call back event function pointer for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laScreen_ShowHideCallback_FnPtr laScreen_GetHideEventCallback(laScreen* scr);
```

### Returns

laScreen_ShowHideCallback_FnPtr

### Description

Returns the hide call back event function pointer for the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to reference |

### Function

aScreen_ShowHideCallback_FnPtr laScreen_GetHideEventCallback( laScreen* scr)

## laScreen_GetLayerIndex Function

Returns the index of the layer for the screen specified.

### File

libaria_screen.h

### C

```
LIB_EXPORT int32_t laScreen_GetLayerIndex(laScreen* scr, laLayer* layer);
```

### Returns

uint32_t - the index of the layer

### Description

Returns the index of the layer for the screen specified.

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to reference |
| laLayer* layer | the layer to search for |

### Function

int32_t laScreen_GetLayerIndex( laScreen* scr, laLayer* layer)

## laScreen_GetOrientation Function

Returns the orientation object associated with the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laScreenOrientation laScreen_GetOrientation(laScreen* scr);
```

### Returns

laScreenOrientation - the screen orientation

### Description

Returns the orientation object associated with the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to reference |

### Function

laScreenOrientation laScreen_GetOrientation(laScreen* scr)

## laScreen_GetShowEventCallback Function

Returns the show call back event function pointer for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laScreen_ShowHideCallback_FnPtr laScreen_GetShowEventCallback(laScreen* scr);
```

### Returns

laScreen_ShowHideCallback_FnPtr

### Description

Returns the show call back event function pointer for the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to reference |

### Function

laScreen_ShowHideCallback_FnPtr laScreen_GetShowEventCallback(laScreen* scr)

## laScreen_Hide Function

Hide the currently active screen

This function has been deprecated in favor of laContext_SetActiveScreen

### File

libaria_screen.h

### C

```
LIB_EXPORT GFX_DEPRECATED laResult laScreen_Hide(laScreen* scr);
```

### Returns

laResult

### Description

The function makes sure that the specified screen is currently active, hides the screen by calling the hide callback function pointer, if the persistent flag is not marked for that screen, delete the screen and free memory. Reset or turn off the Layers allocated for the screen.

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to hide |

### Function

laResult laScreen_Hide(laScreen* scr)

## laScreen_New Function

Create a new screen, initialize it to the values specified.

### File

libaria_screen.h

### C

```
LIB_EXPORT laScreen* laScreen_New(laBool persistent, laBool createAtStartup, laScreen_CreateCallback_FnPtr cb);
```

### Returns

void

### Description

Create a new screen, initialize it to the values specified. The key properties to specify include screen persistence, call backs for screen creation, initialize the screen to default values either specified through MHGC or manually by user.

### Parameters

| Parameters | Description |
|---|---|
| laBool persistent | indicates that the screen should not free the memory of its layers when it is hidden |
| laBool createAtStartup | indicates that the screen should be created as soon as possible to make its widgets accessible to the application |
| laScreen_CreateCallback_FnPtr cb | the function that should be called to initialize the screen at a later time |

### Function

laScreen* laScreen_New(laBool persistent,
laBool createAtStartup,
laScreen_CreateCallback_FnPtr cb)

## laScreen_SetHideEventCallback Function

Set the hide call back event function pointer for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laResult laScreen_SetHideEventCallback(laScreen* scr, laScreen_ShowHideCallback_FnPtr cb);
```

### Returns

laResult

## Description

Set the hide call back event function pointer for the specified screen

## Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to modify laScreen_ShowHideCallback_FnPtr |

## Function

laResult laScreen_SetHideEventCallback(laScreen* scr,
laScreen_ShowHideCallback_FnPtr cb)

## laScreen_SetLayer Function

Assigns the provided layer pointer to the screen at the given index

This function has been deprecated in favor of laContext_SetActiveScreen

## File

libaria_screen.h

## C

```
LIB_EXPORT laResult laScreen_SetLayer(laScreen* scr, uint32_t idx, laLayer* layer);
```

## Returns

laResult - the result of the operation

## Description

Screens contain an internal list of layer pointers. This API assigns a layer to a screen. If the screen is currently active the library attempts to immediately enable the new layer in the HAL.

## Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to modify |
| uint32_t idx | the index of the layer |
| laLayer* layer | the layer pointer to assign to the screen |

## Function

laResult laScreen_SetLayer(laScreen* scr, uint32_t idx, laLayer* layer)

## laScreen_SetOrientation Function

Sets the orientation object to the specified screen

## File

libaria_screen.h

## C

```
LIB_EXPORT laResult laScreen_SetOrientation(laScreen* scr, laScreenOrientation ori);
```

## Returns

laResult - the result of the operation

## Description

Sets the orientation object to the specified screen

## Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to modify |

| | |
|---|---|
| laScreenOrientation | the new orientation setting |

### Function

laResult laScreen_SetOrientation(laScreen* scr, laScreenOrientation ori)

## laScreen_SetShowEventCallback Function

Set the show call back event function pointer for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laResult laScreen_SetShowEventCallback(laScreen* scr, laScreen_ShowHideCallback_FnPtr cb);
```

### Returns

laResult - the result of the operation

### Description

Set the show call back event function pointer for the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to modify |
| laScreen_ShowHideCallback_FnPtr | the function pointer to use |

### Function

laResult laScreen_SetShowEventCallback(laScreen* scr,
laScreen_ShowHideCallback_FnPtr cb)

## laScreen_Show Function

Make the specified screen active and show it on the display

### File

libaria_screen.h

### C

```
LIB_EXPORT GFX_DEPRECATED laResult laScreen_Show(laScreen* scr);
```

### Returns

void

### Description

The function makes sure that the specified screen is not already active, creates it if it is is not already created, sets the appropriate color mode, make it active and call the show callback function pointer.

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to show |

### Function

laResult laScreen_Show(laScreen* scr)

## laScrollBarWidget_GetExtentValue Function

Gets the current scroll bar extent value

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT uint32_t laScrollBarWidget_GetExtentValue(laScrollBarWidget* bar);
```

### Returns

uint32_t - the extent value

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

uint32_t laScrollBarWidget_GetExtentValue( laScrollBarWidget* bar)

## laScrollBarWidget_GetMaxinumValue Function

Gets the maximum scroll value

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT uint32_t laScrollBarWidget_GetMaxinumValue(laScrollBarWidget* bar);
```

### Returns

uint32_t - the maximum scroll value

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

uint32_t laScrollBarWidget_GetMaxinumValue( laScrollBarWidget* bar)

## laScrollBarWidget_GetOrientation Function

Gets the orientation value for the scroll bar

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laScrollBarOrientation laScrollBarWidget_GetOrientation(laScrollBarWidget* bar);
```

### Returns

laScrollBarOrientation - the orientation value

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

laScrollBarOrientation laScrollBarWidget_GetOrientation(laScrollBarWidget* bar)

## laScrollBarWidget_GetScrollPercentage Function

Gets the current scroll value as a percentage

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT uint32_t laScrollBarWidget_GetScrollPercentage(laScrollBarWidget* bar);
```

### Returns

uint32_t - the scroll percentage

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

uint32_t laScrollBarWidget_GetScrollPercentage( laScrollBarWidget* bar)

## laScrollBarWidget_GetScrollValue Function

Gets the current scroll value

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT uint32_t laScrollBarWidget_GetScrollValue(laScrollBarWidget* bar);
```

### Returns

uint32_t - the scroll value

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

uint32_t laScrollBarWidget_GetScrollValue( laScrollBarWidget* bar)

## laScrollBarWidget_GetStepSize Function

Gets the current discreet step size

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT uint32_t laScrollBarWidget_GetStepSize(laScrollBarWidget* bar);
```

### Returns

uint32_t - the current step size

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

uint32_t laScrollBarWidget_GetStepSize( laScrollBarWidget* bar)

## laScrollBarWidget_GetValueChangedEventCallback Function

Gets the current value changed callback function pointer

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laScrollBarWidget_ValueChangedEvent
laScrollBarWidget_GetValueChangedEventCallback(laScrollBarWidget* bar);
```

### Returns

laScrollBarWidget_ValueChangedEvent - a valid pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

laScrollBarWidget_ValueChangedEvent laScrollBarWidget_GetValueChangedEventCallback(laScrollBarWidget* bar)

## laScrollBarWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laScrollBarWidget* laScrollBarWidget_New();
```

### Returns

laScrollBarWidget*

### Function

laScrollBarWidget* laScrollBarWidget_New()

## laScrollBarWidget_SetExtentValue Function

Sets the scroll bar extent value

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetExtentValue(laScrollBarWidget* bar, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| uint32_t val | the extent value |

### Function

laResult laScrollBarWidget_SetExtentValue(laScrollBarWidget* bar,
uint32_t val)

## laScrollBarWidget_SetMaximumValue Function

Sets the maximum scroll value

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetMaximumValue(laScrollBarWidget* bar, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| uint32_t val | the desired maximum scroll value |

### Function

laResult laScrollBarWidget_SetMaximumValue(laScrollBarWidget* bar,
uint32_t val)

## laScrollBarWidget_SetOrientation Function

Sets the orientation value of the scroll bar

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetOrientation(laScrollBarWidget* bar, laScrollBarOrientation align,
laBool swapDimensions);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| laScrollBarOrientation | the desired orientation value |

### Function

laResult laScrollBarWidget_SetOrientation(laScrollBarWidget* bar,
laScrollBarOrientation align)

## laScrollBarWidget_SetScrollPercentage Function

Sets the current scroll value using a percentage. Percentage should be a value from 0 - 100

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetScrollPercentage(laScrollBarWidget* bar, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| uint32_t val | a value from 0 - 100 |

### Function

laResult laScrollBarWidget_SetScrollPercentage(laScrollBarWidget* bar,
uint32_t val)

## laScrollBarWidget_SetScrollValue Function

Sets the current scroll value

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetScrollValue(laScrollBarWidget* bar, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| uint32_t | the desired scroll value |

### Function

laResult laScrollBarWidget_SetScrollValue(laScrollBarWidget* bar,
uint32_t val)

## laScrollBarWidget_SetStepSize Function

Sets the current step size

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetStepSize(laScrollBarWidget* bar, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| uint32_t val | the desired step size |

### Function

laResult laScrollBarWidget_SetStepSize(laScrollBarWidget* bar,
uint32_t val)

## laScrollBarWidget_SetValueChangedEventCallback Function

Sets the value changed event callback pointer

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_SetValueChangedEventCallback(laScrollBarWidget* bar,
laScrollBarWidget_ValueChangedEvent cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |
| laScrollBarWidget_ValueChangedEvent | a valid pointer or NULL |

### Function

laResult laScrollBarWidget_SetValueChangedEventCallback(laScrollBarWidget* bar,
laScrollBarWidget_ValueChangedEvent cb)

## laScrollBarWidget_StepBackward Function

Moves the scroll value back by the current step size

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_StepBackward(laScrollBarWidget* bar);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

laResult laScrollBarWidget_StepBackward(laScrollBarWidget* bar)

## laScrollBarWidget_StepForward Function

Moves the scroll value forward by the current step size

### File

libaria_widget_scrollbar.h

### C

```
LIB_EXPORT laResult laScrollBarWidget_StepForward(laScrollBarWidget* bar);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laScrollBarWidget* bar | the widget |

### Function

laResult laScrollBarWidget_StepForward(laScrollBarWidget* bar)

## laSliderWidget_GetGripSize Function

Gets the current grip size of the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT uint32_t laSliderWidget_GetGripSize(laSliderWidget* sld);
```

### Returns

uint32_t - the current grip size

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

uint32_t laSliderWidget_GetGripSize( laSliderWidget* sld)

## laSliderWidget_GetMaxinumValue Function

Gets the maximum value for the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT uint32_t laSliderWidget_GetMaxinumValue(laSliderWidget* sld);
```

### Returns

uint32_t - the maximum value for the slider

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

uint32_t laSliderWidget_GetMaxinumValue( laSliderWidget* sld)

## laSliderWidget_GetMininumValue Function

Gets the minimum value for the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT uint32_t laSliderWidget_GetMininumValue(laSliderWidget* sld);
```

### Returns

uint32_t - the minimum slider value

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

uint32_t laSliderWidget_GetMininumValue( laSliderWidget* sld)

## laSliderWidget_GetOrientation Function

Gets the orientation value for the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laSliderOrientation laSliderWidget_GetOrientation(laSliderWidget* sld);
```

### Returns

laSliderOrientation

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

laSliderOrientation laSliderWidget_GetOrientation(laSliderWidget* sld)

## laSliderWidget_GetSliderPercentage Function

Gets the slider value as a percentage

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT uint32_t laSliderWidget_GetSliderPercentage(laSliderWidget* sld);
```

### Returns

uint32_t - the slider value as a percentage

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

uint32_t laSliderWidget_GetSliderPercentage( laSliderWidget* sld)

## laSliderWidget_GetSliderValue Function

Gets the current slider value

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT int32_t laSliderWidget_GetSliderValue(laSliderWidget* sld);
```

### Returns

uint32_t - the current slider value

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

uint32_t laSliderWidget_GetSliderValue( laSliderWidget* sld)

## laSliderWidget_GetValueChangedEventCallback Function

Gets the current value changed event callback pointer

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laSliderWidget_ValueChangedEvent laSliderWidget_GetValueChangedEventCallback(laSliderWidget*
sld);
```

### Returns

laSliderWidget_ValueChangedEvent - a valid callback or NULL

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

### Function

laSliderWidget_ValueChangedEvent laSliderWidget_GetValueChangedEventCallback(laSliderWidget* sld)

## laSliderWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laSliderWidget* laSliderWidget_New();
```

### Returns

laSliderWidget*

### Function

laSliderWidget* laSliderWidget_New()

## laSliderWidget_SetGripSize Function

Sets the grip size of the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetGripSize(laSliderWidget* sld, uint32_t size);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| uint32_t size | the desired grip size |

### Function

laResult laSliderWidget_SetGripSize(laSliderWidget* sld,

uint32_t size)

## laSliderWidget_SetMaximumValue Function

Sets the maximum value for the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetMaximumValue(laSliderWidget* sld, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| uint32_t val | the desired maximum value for the slider |

### Function

laResult laSliderWidget_SetMaximumValue(laSliderWidget* sld,
uint32_t val)

## laSliderWidget_SetMinimumValue Function

Sets the minimum value for the slider

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetMinimumValue(laSliderWidget* sld, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| uint32_t val | the desired minimum value |

### Function

laResult laSliderWidget_SetMinimumValue(laSliderWidget* sld,
uint32_t val)

## laSliderWidget_SetOrientation Function

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetOrientation(laSliderWidget* sld, laSliderOrientation align, laBool
swapDimensions);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |

| | |
|---|---|
| laSliderOrientation | the desired slider orientation |
| laBool | indicates if the width and height of the slider should be swapped |

### Function

laResult laSliderWidget_SetOrientation(laSliderWidget* sld,
laSliderOrientation align,
laBool swapDimensions)

## laSliderWidget_SetSliderPercentage Function

Sets the slider value using a percentage. Value must be from 0 - 100.

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetSliderPercentage(laSliderWidget* sld, uint32_t val);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| uint32_t val | a percentage value from 0 - 100 |

### Function

laResult laSliderWidget_SetSliderPercentage(laSliderWidget* sld,
uint32_t val)

## laSliderWidget_SetSliderValue Function

Sets the current slider value

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetSliderValue(laSliderWidget* sld, int32_t val);
```

### Returns

laResult - the operation result

### Description

Must be between slider min and max

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| uint32_t val | the desired slider value |

### Function

laResult laSliderWidget_SetSliderValue(laSliderWidget* sld,
int32_t val)

## laSliderWidget_SetValueChangedEventCallback Function

Sets the value changed event callback pointer

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_SetValueChangedEventCallback(laSliderWidget* sld,
laSliderWidget_ValueChangedEvent cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| laSliderWidget_ValueChangedEvent | a valid pointer or NULL |

### Function

laResult laSliderWidget_SetValueChangedEventCallback(laSliderWidget* sld,
laSliderWidget_ValueChangedEvent cb)

## laSliderWidget_Step Function

Moves the slider by a given amount

### File

libaria_widget_slider.h

### C

```
LIB_EXPORT laResult laSliderWidget_Step(laSliderWidget* sld, int32_t amount);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laSliderWidget* sld | the widget |
| int32_t amount | the amount by which to adjust the current slider value |

### Function

laResult laSliderWidget_Step(laSliderWidget* sld, int32_t amount)

## laTextFieldWidget_GetAlignment Function

Gets the text horizontal alignment value.

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laHAlignment laTextFieldWidget_GetAlignment(laTextFieldWidget* txt);
```

### Returns

laHAlignment - the horizontal alignment value

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |

### Function

laHAlignment laTextFieldWidget_GetAlignment(laTextFieldWidget* txt)

## laTextFieldWidget_GetCursorDelay Function

Gets the current cursor delay.

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT uint32_t laTextFieldWidget_GetCursorDelay(laTextFieldWidget* txt);
```

### Returns

uint32_t - the current delay value

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |

### Function

uint32_t laTextFieldWidget_GetCursorDelay( laTextFieldWidget* txt)

## laTextFieldWidget_GetCursorEnabled Function

Gets the cursor enabled value

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laBool laTextFieldWidget_GetCursorEnabled(laTextFieldWidget* txt);
```

### Returns

laBool - the cursor enabled flag value

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |

### Function

laBool laTextFieldWidget_GetCursorEnabled(laTextFieldWidget* txt)

## laTextFieldWidget_GetCursorPosition Function

Gets the current edit cursor position

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT uint32_t laTextFieldWidget_GetCursorPosition(laTextFieldWidget* txt);
```

### Returns

uint32_t - the index of the cursor

### Description

This cursor will appear to the left of the character at index of the string

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |

### Function

uint32_t laTextFieldWidget_GetCursorPosition( laTextFieldWidget* txt)

## laTextFieldWidget_GetText Function

Gets the text value for the box.

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_GetText(laTextFieldWidget* txt, laString* str);
```

### Returns

laResult - the operation result

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| laString* str | a pointer to an laString object |

### Function

laResult laTextFieldWidget_GetText(laTextFieldWidget* txt, laString* str)

## laTextFieldWidget_GetTextChangedEventCallback Function

Gets the current text changed event callback pointer

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laTextFieldWidget_TextChangedCallback
laTextFieldWidget_GetTextChangedEventCallback(laTextFieldWidget* txt);
```

### Returns

laTextFieldWidget_TextChangedCallback - a valid pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |

### Function

laTextFieldWidget_TextChangedCallback laTextFieldWidget_GetTextChangedEventCallback(laTextFieldWidget* txt)

## laTextFieldWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laTextFieldWidget* laTextFieldWidget_New();
```

### Returns

laTextFieldWidget*

### Function

laTextFieldWidget* laTextFieldWidget_New()

## laTextFieldWidget_SetAlignment Function

Sets the text horizontal alignment value

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetAlignment(laTextFieldWidget* txt, laHAlignment align);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| laHAlignment | the horizontal alignment value |

### Function

laResult laTextFieldWidget_SetAlignment(laTextFieldWidget* txt,
laHAlignment align)

## laTextFieldWidget_SetCursorDelay Function

Sets the cursor delay value

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetCursorDelay(laTextFieldWidget* txt, uint32_t dt);
```

### Returns

laResult - the operation result

### Description

This value is typically expressed in milliseconds

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| uint32_t dt | the cursor delay value |

### Function

laResult laTextFieldWidget_SetCursorDelay(laTextFieldWidget* txt,
uint32_t dt)

## laTextFieldWidget_SetCursorEnabled Function

Sets the cursor enabled value flag

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetCursorEnabled(laTextFieldWidget* txt, laBool en);
```

### Returns

laResult - the operation result

### Description

The cursor enabled flag controls whether the cursor will display or not

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| laBool en | the desired flag state |

### Function

laResult laTextFieldWidget_SetCursorEnabled(laTextFieldWidget* txt,
laBool en)

## laTextFieldWidget_SetCursorPosition Function

Sets the position of the cursor

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetCursorPosition(laTextFieldWidget* txt, uint32_t pos);
```

### Returns

laResult - the operation result

### Description

The cursor will appear to the left of the character at pos

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| uint32_t pos | the position of the cursor in character indices |

### Function

laResult laTextFieldWidget_SetCursorPosition(laTextFieldWidget* txt,
uint32_t pos)

## laTextFieldWidget_SetText Function

Sets the text value for the box.

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetText(laTextFieldWidget* txt, laString str);
```

### Returns

laResult - the operation result

### Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |

| | |
|---|---|
| laString str | an laString object |

### Function

laResult laTextFieldWidget_SetText(laTextFieldWidget* txt, laString str)

## laTextFieldWidget_SetTextChangedEventCallback Function

Sets the text changed event callback pointer

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetTextChangedEventCallback(laTextFieldWidget* txt,
laTextFieldWidget_TextChangedCallback cb);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| laTextFieldWidget_TextChangedCallback | a valid pointer or NULL |

### Function

laResult laTextFieldWidget_SetTextChangedEventCallback(laTextFieldWidget* txt,
laTextFieldWidget_TextChangedCallback cb)

## laTouchTest_AddPoint Function

Adds a point to the touch test widget. The point will then be displayed.

### File

libaria_widget_touchtest.h

### C

```
LIB_EXPORT laResult laTouchTest_AddPoint(laTouchTestWidget* tch, GFX_Point* pnt);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laTouchTestWidget* tch | the widget |
| GFX_Point* pnt | a pointer to the point to add |

### Function

laResult laTouchTest_AddPoint(laTouchTestWidget* tch, GFX_Point* pnt)

## laTouchTest_ClearPoints Function

Clears all of the existing touch points

### File

libaria_widget_touchtest.h

### C

```
LIB_EXPORT laResult laTouchTest_ClearPoints(laTouchTestWidget* tch);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laTouchTestWidget* tch | the widget |

### Function

laResult laTouchTest_ClearPoints(laTouchTestWidget* tch)

## laTouchTestWidget_GetPointAddedEventCallback Function

Gets the current point added event callback

### File

libaria_widget_touchtest.h

### C

```
LIB_EXPORT laTouchTestWidget_PointAddedEventCallback
laTouchTestWidget_GetPointAddedEventCallback(laTouchTestWidget* txt);
```

### Returns

laTouchTestWidget_PointAddedEventCallback - a valid pointer or NULL

### Parameters

| Parameters | Description |
|---|---|
| laTouchTestWidget* tch | the widget |

### Function

laTouchTestWidget_PointAddedEventCallback laTouchTestWidget_GetPointAddedEventCallback(laTouchTestWidget* txt)

## laTouchTestWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_touchtest.h

### C

```
LIB_EXPORT laTouchTestWidget* laTouchTestWidget_New();
```

### Returns

laTouchTestWidget*

### Function

laTouchTestWidget* laTouchTestWidget_New()

## laTouchTestWidget_SetPointAddedEventCallback Function

Sets the point added event callback

### File

libaria_widget_touchtest.h

### C

```
LIB_EXPORT laResult laTouchTestWidget_SetPointAddedEventCallback(laTouchTestWidget* txt,
laTouchTestWidget_PointAddedEventCallback cb);
```

### Returns

laResult - the operation result

## Parameters

| Parameters | Description |
|---|---|
| laTouchTestWidget* tch | the widget |
| laTouchTestWidget_PointAddedEventCallback cb | a valid pointer or NULL |

## Function

laResult laTouchTestWidget_SetPointAddedEventCallback(laTouchTestWidget* txt,
laTouchTestWidget_PointAddedEventCallback cb)

### laUtils_ArrangeRectangle Function

Calculates the position of a rectangle within the given bounds and in accordance with the given parameters. A use case for this is when an image and a text rectangle must be arranged in a button box. This version of the algorithm will calculate the location of the image rectangle.

## File

libaria_utils.h

## C

```
void laUtils_ArrangeRectangle(GFX_Rect* sub, GFX_Rect obj, GFX_Rect bounds, laHAlignment hAlignment,
laVAlignment vAlignment, laRelativePosition position, uint8_t leftMargin, uint8_t topMargin, uint8_t
rightMargin, uint8_t bottomMargin, uint16_t rectMargin);
```

## Returns

void

## Remarks

The x and y position of sub will be manipulated by this function. The dimensions of the rectangle should be set before calling and should remain unchanged after execution.

## Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* sub | the bounds of the subject rectangle (image) |
| GFX_Rect obj | the bounds of the object rectangle (text) |
| GFX_Rect bounds | the bounds of the bounding rectangle (widget) |
| laHAlignment hAlignment | the horizontal alignment of the rects |
| laVAlignment vAlignment | the vertical alignment of the rects |
| laRelativePosition position | the relative position of the rectangles |
| uint8_t leftMargin | the left margin of the bounding rectangle |
| uint8_t topMargin | the top margin of the bounding rectangle |
| uint8_t rightMargin | the right margin of the bounding rectangle |
| uint8_t bottomMargin | the bottom margin of the bounding rectangle |
| uint16_t rectMargin | the distance between the image and the text rects |

## Function

void laUtils_ArrangeRectangle( GFX_Rect* sub,
GFX_Rect obj,
GFX_Rect bounds,
laHAlignment hAlignment,
laVAlignment vAlignment,
laRelativePosition position,
uint8_t leftMargin,
uint8_t topMargin,
uint8_t rightMargin,
uint8_t bottomMargin,
uint16_t rectMargin)

## laUtils_ArrangeRectangleRelative Function

Calculates the position of a rectangle within the given bounds and in accordance with the given parameters. A use case for this is when an image and a text rectangle must be arranged in a button box. This version of the algorithm will calculate the location of the text rectangle.

### File

libaria_utils.h

### C

```
void laUtils_ArrangeRectangleRelative(GFX_Rect* sub, GFX_Rect obj, GFX_Rect bounds, laHAlignment
hAlignment, laVAlignment vAlignment, laRelativePosition position, uint8_t leftMargin, uint8_t topMargin,
uint8_t rightMargin, uint8_t bottomMargin, uint16_t rectMargin);
```

### Returns

void

### Remarks

The x and y position of sub will be manipulated by this function. The dimensions of the rectangle should be set before calling and should remain unchanged after execution.

### Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* sub | the bounds of the subject rectangle (text) |
| GFX_Rect obj | the bounds of the object rectangle (image) |
| GFX_Rect bounds | the bounds of the bounding rectangle (widget) |
| laHAlignment hAlignment | the horizontal alignment of the rects |
| laVAlignment vAlignment | the vertical alignment of the rects |
| laRelativePosition position | the relative position of the rectangles |
| uint8_t leftMargin | the left margin of the bounding rectangle |
| uint8_t topMargin | the top margin of the bounding rectangle |
| uint8_t rightMargin | the right margin of the bounding rectangle |
| uint8_t bottomMargin | the bottom margin of the bounding rectangle |
| uint16_t rectMargin | the distance between the image and the text rects |

### Function

void laUtils_ArrangeRectangleRelative( GFX_Rect* sub,
GFX_Rect obj,
GFX_Rect bounds,
laHAlignment hAlignment,
laVAlignment vAlignment,
laRelativePosition position,
uint8_t leftMargin,
uint8_t topMargin,
uint8_t rightMargin,
uint8_t bottomMargin,
uint16_t rectMargin)

## laUtils_ChildIntersectsParent Function

Performs an intersection test between a parent widget and a child widget

### File

libaria_utils.h

### C

```
laBool laUtils_ChildIntersectsParent(laWidget* parent, laWidget* child);
```

### Returns

laBool - result of the intersection test

### Parameters

| Parameters | Description |
|---|---|
| laWidget* parent | the parent widget |
| laWidget* child | the child widget |

### Function

laBool laUtils_ChildIntersectsParent(laWidget* parent, laWidget* child)

## laUtils_ClipRectToParent Function

Clips a rectangle to the parent of a widget

### File

libaria_utils.h

### C

```
void laUtils_ClipRectToParent(laWidget* widget, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Rect* rect | the rectangle to clip |

### Function

void laUtils_ClipRectToParent( laWidget* widget, GFX_Rect* rect)

## laUtils_GetLayer Function

Finds the root parent of a widget, which should be a layer

### File

libaria_utils.h

### C

```
laLayer* laUtils_GetLayer(laWidget* widget);
```

### Returns

laLayer* - the widget's owning layer

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |

### Function

laLayer* laUtils_GetLayer(laWidget* widget)

## laUtils_ListOcclusionCullTest Function

Performs an occlusion test on a list of widgets an a rectangular area. This attempts to find only the topmost widgets for the given area. If a widget is completely occluded then it is removed from the list. Any widgets that remain in the list should be redrawn by the rasterizer.

### File

libaria_utils.h

### C

```
void laUtils_ListOcclusionCullTest(laList* list, GFX_Rect rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laList* list | the widget list to test |
| GFX_Rect rect | the occlusion area |

### Function

void laUtils_ListOcclusionCullTest( laList* list, GFX_Rect rect)

## laUtils_OcclusionCullTest Function

Performs an occlusion test for a widget in the tree. A widget is occluded if it is completely covered by one or more widgets. This is useful for culling widgets before the rasterizing phase.

### File

libaria_utils.h

### C

```
laBool laUtils_OcclusionCullTest(laWidget* widget);
```

### Returns

laBool - result of the occlusion test

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the widget to test |

### Function

laBool laUtils_OcclusionCullTest(laWidget* widget)

## laUtils_Pick Function

Finds the top-most visible widget in the tree at the given coordinates.

### File

libaria_utils.h

### C

```
LIB_EXPORT laWidget* laUtils_Pick(int32_t x, int32_t y);
```

### Returns

laWidget* - the result widget

### Parameters

| Parameters | Description |
|---|---|
| int32_t x | the x coordinate of the pick point |
| int32_t y | the y coordinate of the pick point |

### Function

laWidget* laUtils_Pick(int32_t x, int32_t y)

## laUtils_PickRect Function

Finds all of the visible widgets in the given rectangular area.

### File

libaria_utils.h

### C

```
void laUtils_PickRect(laLayer* layer, GFX_Rect rect, laList* list);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer to analyze |
| GFX_Rect | the rectangle pick area |
| laList* list | the result list |

### Function

void laUtils_PickRect( laLayer* layer, GFX_Rect rect, laList* list)

## laUtils_PointScreenToLocalSpace Function

Converts a point from layer space into the local space of a widget

### File

libaria_utils.h

### C

```
void laUtils_PointScreenToLocalSpace(laWidget* widget, GFX_Point* pnt);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Point* pnt | the point to convert |

### Function

void laUtils_PointLayerToLocalSpace( laWidget* widget, GFX_Point* pnt)

## laUtils_RectFromParentSpace Function

Converts a rectangle from widget parent space to widget local space

### File

libaria_utils.h

### C

```
void laUtils_RectFromParentSpace(laWidget* widget, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Rect* rect | the rectangle to convert |

### Function

void laUtils_RectFromParentSpace( laWidget* widget, GFX_Rect* rect)

## laUtils_RectToLayerSpace Function

Converts a rectangle from widget local space to layer space

### File

libaria_utils.h

### C

```
void laUtils_RectToLayerSpace(laWidget* widget, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Rect* rect | the rectangle to convert |

### Function

void laUtils_RectToLayerSpace( laWidget* widget, GFX_Rect* rect)

## laUtils_RectToParentSpace Function

Converts a rectangle from widget local space to widget parent space. Widget must be a child of a layer for this to function.

### File

libaria_utils.h

### C

```
void laUtils_RectToParentSpace(laWidget* widget, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Rect* rect | the rectangle to convert |

### Function

void laUtils_RectToParentSpace( laWidget* widget, GFX_Rect* rect)

## laUtils_RectToScreenSpace Function

Converts a rectangle from widget local space to screen space

### File

libaria_utils.h

### C

```
void laUtils_RectToScreenSpace(laWidget* widget, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Rect* rect | the rectangle to convert |

### Function

void laUtils_RectToScreenSpace( laWidget* widget, GFX_Rect* rect)

## laWidget_AddChild Function

Adds the child to the parent widget specified in the argument

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_AddChild(laWidget* parent, laWidget* child);
```

### Returns

laResult - the operation result

### Description

The function checks to see if the child and parent are valid, removes the child from its current parents children list, and assigns the child to the parent widget specified. The child is attached at the end of the list of the parent widgets children list.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* parent | the parent widget |
| laWidget* child | the child to add |

### Function

laResult laWidget_AddChild(laWidget* parent, laWidget* child)

## laWidget_Delete Function

Delete the widget object specified

### File

libaria_widget.h

### C

```
LIB_EXPORT void laWidget_Delete(laWidget* wgt);
```

### Returns

void

### Description

Delete a widget object specified, de-allocate memory for the widget through the current active context. All child widgets are also destructed and freed.

### Function

void laWidget_Delete( laWidget* wgt)

## laWidget_GetAlphaAmount Function

Return the widget's global alpha amount

### File

libaria_widget.h

### C

```
LIB_EXPORT uint32_t laWidget_GetAlphaAmount(laWidget* wgt);
```

### Returns

uint32_t - the widget's global alpha amount

### Description

Return the widget's global alpha amount

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

uint32_t laWidget_GetAlphaAmount( laWidget* wgt)

## laWidget_GetAlphaEnable Function

Return the alpha enable property of the widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_GetAlphaEnable(laWidget* wgt);
```

### Returns

laBool - the widget's alpha enable flag value

### Description

Return the alpha enable property of the widget

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

laBool laWidget_GetAlphaEnable(laWidget* wgt)

## laWidget_GetBorderType Function

Return the border type associated with the widget object

### File

libaria_widget.h

### C

```
LIB_EXPORT laBorderType laWidget_GetBorderType(laWidget* wgt);
```

### Returns

laBorderType - the current widget border type

### Description

Return the border type associated with the widget object

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBorderType laWidget_GetBorderType(laWidget* wgt)

## laWidget_GetChildAtIndex Function

Fetches the child at the specified index from the children list of the specified parent widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laWidget* laWidget_GetChildAtIndex(laWidget* parent, uint32_t idx);
```

### Returns

laWidget* - a valid child pointer or NULL

### Description

Fetches the child at the specified index from the children list of the specified parent widget

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| uint32_t idx | the desired child index |

### Function

laWidget* laWidget_GetChildAtIndex(laWidget* parent, uint32_t idx)

## laWidget_GetChildCount Function

Returns the size of the children list of the specified parent widget

### File

libaria_widget.h

### C

```
LIB_EXPORT uint32_t laWidget_GetChildCount(laWidget* parent);
```

### Returns

uint32_t - the number of children of this widget

### Description

Returns the size of the children list of the specified parent widget

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

uint32_t laWidget_GetChildCount( laWidget* parent)

## laWidget_GetCumulativeAlphaAmount Function

Calculates the cumulative alpha amount for a hierarchy of widgets

### File

libaria_widget.h

### C

```
LIB_EXPORT uint32_t laWidget_GetCumulativeAlphaAmount(laWidget* wgt);
```

### Returns

uint32_t - the cumulative blending amount

### Description

Alpha blending amounts are cumulative from parent to child. If a parent is blended at 50% then logically a child should also implicitely be blended at 50%. If a child further explictely enables blending at 50% then the cumulative amount is 25%.

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

uint32_t laWidget_GetCumulativeAlphaAmount( laWidget* wgt)

## laWidget_GetCumulativeAlphaEnable Function

Determines if this or any ancestor widget has alpha enabled

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_GetCumulativeAlphaEnable(laWidget* wgt);
```

### Returns

laBool - whether the widget has alpha enabled

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

laBool laWidget_GetCumulativeAlphaEnable(laWidget* wgt)

## laWidget_GetEnabled Function

Returns the boolean value of the widget enabled property

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_GetEnabled(laWidget* wgt);
```

### Returns

laBool - the value of the enabled flag

### Description

Returns the boolean value of the widget enabled property. The widget enable flag often governs things like appearing 'greyed out' and prohibits user interactiong if it is false. Widgets must individually support this flag.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBool laWidget_GetEnabled(laWidget* wgt)

## laWidget_GetHeight Function

Returns the widget rectangles height

### File

libaria_widget.h

### C

```
LIB_EXPORT int32_t laWidget_GetHeight(laWidget* wgt);
```

### Returns

uint32_t - the widget's width value

### Description

Returns the widget rectangles height

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

int32_t laWidget_GetHeight( laWidget* wgt)

## laWidget_GetIndexOfChild Function

Fetches the index of the child from the children list of the specified parent widget

### File

libaria_widget.h

### C

```
LIB_EXPORT int32_t laWidget_GetIndexOfChild(laWidget* parent, laWidget* child);
```

### Returns

int32_t - the index of the given child pointer or -1 if not found

### Description

Traverses the children list of the specified parent widget and finds the index of the child widget specified.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* parent | the parent widget |
| laWidget* child | the child widget |

### Function

int32_t laWidget_GetIndexOfChild( laWidget* parent, laWidget* child)

## laWidget_GetMargin Function

Returns the margin value associated with the widget in the laMargin pointer

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_GetMargin(laWidget* wgt, laMargin* mg);
```

### Returns

laResult - the operation result

### Description

Returns the margin value associated with the widget in the laMargin pointer

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laMargin* mg | a pointer to an laMargin object to store the margin values |

### Function

laResult laWidget_GetMargin (laWidget* wgt, laMargin* mg)

## laWidget_GetScheme Function

Returns the scheme associated with the specified widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laScheme* laWidget_GetScheme(laWidget* wgt);
```

### Returns

laScheme* - a pointer to the active scheme for a widget

### Description

Returns the scheme associated with the specified widget

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laScheme* laWidget_GetScheme(laWidget* wgt)

## laWidget_GetVisible Function

Returns the boolean value of the widget visible property

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_GetVisible(laWidget* wgt);
```

### Returns

laBool - the flag value

### Description

Returns the boolean value of the widget visible property. Widgets that are invisible will be skipped during the rendering phase. All descendants also logically become invisible when an ancestor does.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBool laWidget_GetVisible(laWidget* wgt)

## laWidget_GetWidth Function

Returns the widget rectangles width

### File

libaria_widget.h

### C

```
LIB_EXPORT int32_t laWidget_GetWidth(laWidget* wgt);
```

### Returns

uint32_t - the widget's y coordinate value

### Description

Returns the widget rectangles width

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

int32_t laWidget_GetWidth( laWidget* wgt)

## laWidget_GetX Function

Returns the widget rectangles upper left corner x-coordinate

### File

libaria_widget.h

### C

```
LIB_EXPORT int32_t laWidget_GetX(laWidget* wgt);
```

### Returns

uint32_t

### Description

Returns the widget rectangles upper left corner x-coordinate

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

int32_t laWidget_GetX( laWidget* wgt)

## laWidget_GetY Function

Returns the widget rectangles upper left corner y-coordinate

### File

libaria_widget.h

### C

```
LIB_EXPORT int32_t laWidget_GetY(laWidget* wgt);
```

### Returns

uint32_t - the y value

### Description

Returns the widget rectangles upper left corner y-coordinate

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |

### Function

int32_t laWidget_GetY( laWidget* wgt)

## laWidget_HasFocus Function

Checks if the widget specified has focus in the current context

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_HasFocus(laWidget* wgt);
```

### Returns

laBool - true of the widget currently has context focus

### Description

Checks if the widget specified has focus in the current context

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBool laWidget_HasFocus(laWidget* wgt)

## laWidget_Invalidate Function

Invalidates the specified widget.

### File

libaria_widget.h

### C

```
LIB_EXPORT void laWidget_Invalidate(laWidget* wgt);
```

### Returns

void

### Description

This function invalidates the specified widget. Invalid widgets are redrawn during the next paint loop call. This function may also invalidate the widget's parent, siblings, ancestors, or cousins.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

void laWidget_Invalidate( laWidget* wgt)

## laWidget_isOpaque Function

Returns true if the widget is considered opaque.

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_isOpaque(laWidget* wgt);
```

### Returns

laBool - true if the widget is fully opaque

### Description

Opacity is determined by a number of factors including: cumulative alpha amount, background type, and the opaque optimization flag.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBool laWidget_isOpaque(laWidget* wgt)

## laWidget_New Function

Create a new widget.

### File

libaria_widget.h

### C

```
LIB_EXPORT laWidget* laWidget_New();
```

### Returns

lawidget*

### Description

Create a new widget, alocate memory for the widget through the current active context. Returns a widget object pointer. Application is responsible for managing the widget pointer until the widget is added to a widget tree.

### Function

laWidget* laWidget_New()

## laWidget_OverrideTouchDownEvent Function

Replace the TouchDownEvent callback for the widget with the new function pointer specified

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_OverrideTouchDownEvent(laWidget* wgt, laWidget_TouchDownEvent_FnPtr ptr);
```

### Returns

laResult - the operation result

### Description

This function will replace the current touch down event handler for a widget. Widgets may have their own internal override for this function and replacing it will break their internal capabilties.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laWidget_TouchDownEvent_FnPtr | a valid pointer or NULL |

### Function

laResult laWidget_OverrideTouchDownEvent(laWidget* wgt,
laWidget_TouchDownEvent_FnPtr ptr)

## laWidget_OverrideTouchMovedEvent Function

Replace the TouchMovedEvent callback for the widget with the new function pointer specified

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_OverrideTouchMovedEvent(laWidget* wgt, laWidget_TouchMovedEvent_FnPtr ptr);
```

### Returns

laResult - the operation result

### Description

This function will replace the current touch moved event handler for a widget. Widgets may have their own internal override for this function and replacing it will break their internal capabilties.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laWidget_TouchMovedEvent_FnPtr | a valid pointer or NULL |

### Function

laResult laWidget_OverrideTouchMovedEvent(laWidget* wgt,
laWidget_TouchMovedEvent_FnPtr ptr)

## laWidget_OverrideTouchUpEvent Function

Replace the TouchUpEvent callback for the widget with the new function pointer specified

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_OverrideTouchUpEvent(laWidget* wgt, laWidget_TouchUpEvent_FnPtr ptr);
```

### Returns

laResult - the operation result

### Description

This function will replace the current touch up event handler for a widget. Widgets may have their own internal override for this function and replacing it will break their internal capabilties.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laWidget_TouchUpEvent_FnPtr | a valid pointer or NULL |

### Function

laResult laWidget_OverrideTouchUpEvent(laWidget* wgt,
laWidget_TouchUpEvent_FnPtr ptr)

## laWidget_RectToLayerSpace Function

Transforms a widget rectangle from local space to its root layer space.

### File

libaria_widget.h

### C

```
LIB_EXPORT GFX_Rect laWidget_RectToLayerSpace(laWidget* wgt);
```

### Returns

GFX_Rect - the transformed rectangle

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

GFX_Rect laWidget_RectToLayerSpace(laWidget* wgt)

## laWidget_RectToParentSpace Function

Returns the rectangle containing the parent of the widget specified

### File

libaria_widget.h

### C

```
LIB_EXPORT GFX_Rect laWidget_RectToParentSpace(laWidget* wgt);
```

### Returns

GFX_Rect - the widget rectangle in parent space

### Description

Returns the rectangle containing the parent of the widget specified If the widget and the parent are not null, the rectangle defining the parent widget with its upper left corner x and y coordinates is returned

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

GFX_Rect laWidget_RectToParentSpace(laWidget* wgt)

## laWidget_RectToScreenSpace Function

Transforms a widget rectangle from local space to screen space coordinates.

### File

libaria_widget.h

### C

```
LIB_EXPORT GFX_Rect laWidget_RectToScreenSpace(laWidget* wgt);
```

### Returns

GFX_Rect - the transformed rectangle

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

GFX_Rect laWidget_RectToScreenSpace(laWidget* wgt)

## laWidget_RemoveChild Function

Removes the child from the parent widget specified in the argument

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_RemoveChild(laWidget* parent, laWidget* child);
```

### Returns

laResult - the operation result

### Description

The function checks to see if the child and parent are valid, removes the child from its current parents children list

### Parameters

| Parameters | Description |
|---|---|
| laWidget* parent | the parent widget |
| laWidget* child | the child to remove |

### Function

laResult laWidget_RemoveChild(laWidget* parent, laWidget* child)

## laWidget_Resize Function

Changes the widget size by the new defined width and height increments.

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_Resize(laWidget* wgt, int32_t width, int32_t height);
```

### Returns

laResult - the operation result

### Description

Changes the widget size by the new defined width and height increments.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| int32_t width | the amount to change the width by |
| int32_t height | the amount ot change the height by |

### Function

void laWidget_Resize( laWidget* wgt, int32_t width, int32_t height)

## laWidget_SetAlphaAmount Function

Set the widget's global alpha amount to the specified alpha amount

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetAlphaAmount(laWidget* wgt, uint32_t alpha);
```

### Returns

laResult - the result of the operation

### Description

Set the widget's global alpha amount to the specified alpha amount. Widgets may enable alpha blending even for color modes that don't support an alpha channel.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| uint32_t alpha | the desired global alpha amount |

### Function

laResult laWidget_SetAlphaAmount(laWidget* wgt, uint32_t alpha)

## laWidget_SetAlphaEnable Function

Set the alpha enable property of the widget with the boolean value specified

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetAlphaEnable(laWidget* wgt, laBool enable);
```

### Returns

laResult - the result of the operation

## Description

Set the alpha enable property of the widget with the boolean value specified

## Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |
| laBool enable | the desired alpha enable flag value |

## Function

laResult laWidget_SetAlphaEnable(laWidget* wgt, laBool enable)

### laWidget_SetBorderType Function

Set the border type associated with the widget object

## File

libaria_widget.h

## C

```
LIB_EXPORT laResult laWidget_SetBorderType(laWidget* wgt, laBorderType type);
```

## Returns

laResult - the operation result

## Description

Set the border type associated with the widget object

## Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laBorderType type | the desired border type |

## Function

laResult laWidget_SetBorderType(laWidget* wgt, laBorderType type)

### laWidget_SetEnabled Function

Sets the boolean value of the widget enabled property

## File

libaria_widget.h

## C

```
LIB_EXPORT laResult laWidget_SetEnabled(laWidget* wgt, laBool enable);
```

## Returns

laResult - the operation result

## Description

Sets the boolean value of the widget enabled property

## Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laBool | the desired enabled flag value |

## Function

laResult laWidget_SetEnabled(laWidget* wgt, laBool enable)

## laWidget_SetFocus Function

Set the widget into focus for the current context.

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetFocus(laWidget* wgt);
```

### Returns

laResult - the operation result

### Description

Set the widget into focus for the current context. The input events etc are received by the widget once it is in focus

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laResult laWidget_SetFocus(laWidget* wgt)

## laWidget_SetHeight Function

Sets the widget's height value

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetHeight(laWidget* wgt, int32_t height);
```

### Returns

laResult - result of the operation

### Description

Sets the widget's height value

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |
| int32_t height | the desired height value, must be > 0 |

### Function

laResult laWidget_SetHeight(laWidget* wgt, int32_t height)

## laWidget_SetMargins Function

Set the margin value for left, right, top and bottom margins associated with the widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetMargins(laWidget* wgt, uint32_t left, uint32_t top, uint32_t right,
uint32_t bottom);
```

## Returns

laResult - the operation result

## Description

Set the margin value for left, right, top and bottom margins associated with the widget. Margins are a generic property and it is up to the individual widget to implement them (or not).

## Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| uint32_t left | the left margin value |
| uint32_t top | the top margin value |
| uint32_t right | the right margin value |
| uint32_t bottom | the bottom margin value |

## Function

laResult laWidget_SetMargins(laWidget* wgt,
uint32_t left,
uint32_t top,
uint32_t right,
uint32_t bottom)

### laWidget_SetParent Function

Sets the parent of the child widget to that specified in the argument list

## File

libaria_widget.h

## C

```
LIB_EXPORT laResult laWidget_SetParent(laWidget* wgt, laWidget* parent);
```

## Returns

laResult - the operation result

## Description

The function checks to see if the child and parent are valid, removes the child from its current parents children list, and assigns the child to the parent widget specified. The child is attached at the end of the list of the parent widgets children list.

## Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laWidget* parent | the desired parent widget |

## Function

laResult laWidget_SetParent(laWidget* wgt, laWidget* parent)

### laWidget_SetPosition Function

Changes the widget position to the new defined x and y coordinates.

## File

libaria_widget.h

## C

```
LIB_EXPORT laResult laWidget_SetPosition(laWidget* wgt, int32_t x, int32_t y);
```

### Returns

laResult - the operation result

### Description

Changes the widget position to the new defined x and y coordinates. Moving widgets can be expensive as it needs to repaint multiple areas of its parent widget.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| int32_t x | the new x coordinate |
| int32_t y | the new y coordinate |

### Function

void laWidget_SetPosition( laWidget* wgt, int32_t x, int32_t y)

## laWidget_SetScheme Function

Sets the scheme variable for the specified widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetScheme(laWidget* wgt, laScheme* scheme);
```

### Returns

laResult - the operation result

### Description

Sets the scheme variable for the specified widget. The scheme defines the appearance of the widget. Setting this to NULL may result in undefined behavior if the widget doesn't properly support a NULL scheme.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laScheme* scheme | a pointer to a scheme or NULL |

### Function

void laWidget_SetScheme( laWidget* wgt, laScheme* scheme)

## laWidget_SetSize Function

Changes the widget size to the new defined width and height dimensions.

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetSize(laWidget* wgt, uint32_t width, uint32_t height);
```

### Returns

laResult - the operation result

### Description

Changes the widget size to the new width and height dimensions.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| int32_t width | the new width size |
| int32_t height | the new height size |

### Function

void laWidget_SetSize( laWidget* wgt, uint32_t width, uint32_t height)

## laWidget_SetVisible Function

Sets the boolean value of the widget visible property

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetVisible(laWidget* wgt, laBool visible);
```

### Returns

laResult - the operation result

### Description

Sets the boolean value of the widget visible property

### Remarks

This value has no effect on layer objects. Use laLayer_SetEnabled instead.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laBool | the desired setting |

### Function

laResult laWidget_SetVisible(laWidget* wgt, laBool visible)

## laWidget_SetWidth Function

Sets the widget's width value

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetWidth(laWidget* wgt, int32_t width);
```

### Returns

laResult - result of the operation

### Description

Sets the widget's width value

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |
| int32_t width | the desired width value, must be > 0 |

### Function

laResult laWidget_SetWidth(laWidget* wgt, int32_t width)

## laWidget_SetX Function

Sets the widget's x coordinate position

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetX(laWidget* wgt, int32_t x);
```

### Returns

laResult - result of the operation

### Description

Sets the widget's x coordinate position

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |
| int32_t x | the desired x value |

### Function

laResult laWidget_SetX(laWidget* wgt, int32_t x)

## laWidget_SetY Function

Sets the widget's y coordinate position

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetY(laWidget* wgt, int32_t y);
```

### Returns

laResult - result of the operation

### Description

Sets the widget's y coordinate position

### Parameters

| Parameters | Description |
|---|---|
| lawidget* wgt | the widget |
| int32_t y | the desired y value |

### Function

laResult laWidget_SetY(laWidget* wgt, int32_t y)

## laWidget_Translate Function

Changes the widget position by moving the widget by the defined x and y increments.

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_Translate(laWidget* wgt, int32_t x, int32_t y);
```

### Returns

laResult - the operation result

### Description

Changes the widget position by moving the widget by the defined x and y increments.Moving widgets can be expensive as it needs to repaint multiple areas of its parent widget.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| int32_t x | the amount to move in x |
| int32_t y | the amount to move in y |

### Function

void laWidget_Translate( laWidget* wgt, int32_t x, int32_t y)

## laWindowWidget_GetIcon Function

Gets the currently used window icon

### File

libaria_widget_window.h

### C

```
LIB_EXPORT GFXU_ImageAsset* laWindowWidget_GetIcon(laWindowWidget* win);
```

### Returns

GFXU_ImageAsset*

### Parameters

| Parameters | Description |
|---|---|
| laWindowWidget* win | the widget |

### Function

GFXU_ImageAsset* laWindowWidget_GetIcon(laWindowWidget* win)

## laWindowWidget_GetIconMargin Function

Gets the current image icon margin

### File

libaria_widget_window.h

### C

```
LIB_EXPORT uint32_t laWindowWidget_GetIconMargin(laWindowWidget* win);
```

### Returns

uint32_t - the icon margin

### Parameters

| Parameters | Description |
|---|---|
| laWindowWidget* win | the widget |

### Function

uint32_t laWindowWidget_GetIconMargin( laWindowWidget* win)

## laWindowWidget_GetTitle Function

Gets the title text for this window.

### File

libaria_widget_window.h

### C

```
LIB_EXPORT laResult laWindowWidget_GetTitle(laWindowWidget* win, laString* str);
```

### Returns

laResult - the operation result

### Description

This function allocates memory and initializes the input string pointer. The caller is responsible for managing the memory once this function returns.

### Parameters

| Parameters | Description |
|---|---|
| laWindowWidget* win | the widget |
| laString* str | a pointer to an laString object |

### Function

laResult laWindowWidget_GetTitle(laWindowWidget* win, laString* str)

## laWindowWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree.

### File

libaria_widget_window.h

### C

```
LIB_EXPORT laWindowWidget* laWindowWidget_New();
```

### Returns

laWindowWidget*

### Function

laWindowWidget* laWindowWidget_New()

## laWindowWidget_SetIcon Function

Sets the image to be used as a window icon

### File

libaria_widget_window.h

### C

```
LIB_EXPORT laResult laWindowWidget_SetIcon(laWindowWidget* win, GFXU_ImageAsset* img);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laWindowWidget* win | the widget |
| GFXU_ImageAsset* | pointer to an image asset |

### Function

laResult laWindowWidget_SetIcon(laWindowWidget* win,
GFXU_ImageAsset* img)

## laWindowWidget_SetIconMargin Function

Sets the image icon margin

### File

libaria_widget_window.h

### C

```
LIB_EXPORT laResult laWindowWidget_SetIconMargin(laWindowWidget* win, uint32_t mg);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laWindowWidget* win | the widget |
| uint32_t mg | the image icon margin value |

### Function

laResult laWindowWidget_SetIconMargin(laWindowWidget* win, uint32_t mg)

## laWindowWidget_SetTitle Function

Sets the title text for the window.

### File

libaria_widget_window.h

### C

```
LIB_EXPORT laResult laWindowWidget_SetTitle(laWindowWidget* win, laString str);
```

### Returns

laResult - the operation result

### Description

This function copies the contents of the input string into its internal string buffer. The input string can then be freed or altered without affecting the label's internal string value.

### Parameters

| Parameters | Description |
|---|---|
| laWindowWidget* win | the widget |
| laString str | an laString object |

### Function

laResult laWindowWidget_SetTitle(laWindowWidget* win, laString str)

## laCheckBoxWidget_SetImageMargin Function

Sets the distance between the image and the text

### File

libaria_widget_checkbox.h

### C

```
LIB_EXPORT laResult laCheckBoxWidget_SetImageMargin(laCheckBoxWidget* btn, uint32_t mg);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laCheckBoxWidget* cbox | the widget |
| uint32_t mg | the desired image margin value |

### Function

laResult laCheckBoxWidget_SetImageMargin(laCheckBoxWidget* btn, uint32_t mg)

## laContext_SetPreemptionLevel Function

Set the preemption level to the specified value

### File

libaria_context.h

### C

```
LIB_EXPORT laResult laContext_SetPreemptionLevel(laPreemptionLevel level);
```

### Returns

laResult

### Description

Set the preemption level to the specified value

### Function

laResult laContext_SetPreemptionLevel(laPreemptionLevel level)

## laKeyPadWidget_SetKeyBackgroundType Function

Sets the background type for a key pad cell at row/column

### File

libaria_widget_keypad.h

### C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyBackgroundType(laKeyPadWidget* pad, uint32_t row, uint32_t col,
laBackgroundType type);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laKeyPadWidget* pad | the widget |
| uint32_t row | the indicated row |
| uint32_t col | the indicated column |
| laBackgroundType type | the desired background type |

### Function

laResult laKeyPadWidget_SetKeyBackgroundType(laKeyPadWidget* pad,
uint32_t row,
uint32_t col,
laBackgroundType type)

## laLayer_GetAllowInputPassThrough Function

Gets the layer's input passthrough setting

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_GetAllowInputPassThrough(const laLayer* layer);
```

### Returns

laBool - the state of the layer's passthrough flag

### Description

The input passthrough setting is used to prohibit or allow input events to pass through a layer. If a layer is opaque or semi-opaque input events should probably not be allowed to pass through. If the layer is completely transparent then input events may be allowed to pass through to interact with widgets on layers further back in the hierarchy.

An application that disables this is responsible for ensuring that it is modified when the dimensions of the layer change.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

laBool laLayer_GetAllowInputPassThrough(const laLayer* layer)

## laLayer_GetEnabled Function

Returns the boolean value of the layer enabled property

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_GetEnabled(const laLayer* layer);
```

### Returns

laBool - the flag value

### Description

Returns the boolean value of the layer enabled property

### Parameters

| Parameters | Description |
|---|---|
| laLayer* | the layer |

### Function

laBool laLayer_GetEnabled(const laLayer* layer)

## laLayer_SetAllowInputPassthrough Function

Sets the layer's input passthrough flag.

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetAllowInputPassthrough(laLayer* layer, laBool enable);
```

### Returns

laResult - the result of the operation

### Description

Sets the layer's input passthrough flag.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

laResult laLayer_SetAllowInputPassthrough(laLayer* layer, laBool enable)

## laLayer_SetEnabled Function

Sets the boolean value of the layer enabled property

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetEnabled(laLayer* widget, laBool enable);
```

### Returns

laResult - the result of the operation

### Description

Sets the boolean value of the layer enabled property

### Remarks

The enabled flag for a layer will often control the hardware setting for layer usage, depending on the display driver

### Parameters

| Parameters | Description |
|---|---|
| laLayer* | the layer |
| laBool | the desired enabled value |

### Function

laResult laLayer_SetEnabled(laLayer* widget, laBool enable)

## laListWheelWidget_GetFlickInitSpeed Function

Returns the flick init speed for the wheel.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetFlickInitSpeed(laListWheelWidget* whl);
```

### Returns

uint32_t - the flick init speed value

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetFlickInitSpeed( laListWheelWidget* whl)

## laListWheelWidget_GetIndicatorArea Function

Returns the spacing for the selected item indicator bars.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetIndicatorArea(laListWheelWidget* whl);
```

### Returns

uint32_t - the display area

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetIndicatorArea( laListWheelWidget* whl)

## laListWheelWidget_GetMaxMomentum Function

Returns the maximum momentum value for the wheel.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetMaxMomentum(laListWheelWidget* whl);
```

### Returns

uint32_t - the maximum momentum value.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetMaxMomentum( laListWheelWidget* whl)

## laListWheelWidget_GetMomentumFalloffRate Function

Returns the momentum falloff rate for the wheel.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetMomentumFalloffRate(laListWheelWidget* whl);
```

### Returns

uint32_t - the momentum falloff rate value.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetMomentumFalloffRate( laListWheelWidget* whl)

## laListWheelWidget_GetRotationUpdateRate Function

Returns the wheel rotation update rate.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetRotationUpdateRate(laListWheelWidget* whl);
```

### Returns

uint32_t - the rotation update rate value.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

uint32_t laListWheelWidget_GetRotationUpdateRate( laListWheelWidget* whl)

## laListWheelWidget_GetShaded Function

Returns true if the list is using gradient shading to illustrate depth

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laBool laListWheelWidget_GetShaded(laListWheelWidget* whl);
```

### Returns

laBool - true gradient shading is being used

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laBool laListWheelWidget_GetShaded(laListWheelWidget* whl)

## laListWheelWidget_GetShowIndicators Function

Returns true if the list is displaying its selected item indicators

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laBool laListWheelWidget_GetShowIndicators(laListWheelWidget* whl);
```

### Returns

laBool - true if the indicators are being shown

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |

### Function

laBool laListWheelWidget_GetShowIndicators(laListWheelWidget* whl)

## laListWheelWidget_GetVisibleItemCount Function

Returns the list's visible item count

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT uint32_t laListWheelWidget_GetVisibleItemCount(laListWheelWidget* whl);
```

### Returns

uint32_t - the number of visible items

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |

### Function

uint32_t laListWheelWidget_GetVisibleItemCount( laListWheelWidget* whl)

## laListWheelWidget_SetFlickInitSpeed Function

Configures the flick init speed for the list wheel

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetFlickInitSpeed(laListWheelWidget* whl, uint32_t speed);
```

### Returns

laResult - the operation result

### Description

The flick init speed is the drag distance needed to move the wheel into momentum mode. It is the distance that must be covered from one Aria update frame to another.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t speed | the flick init speed value |

### Function

laResult laListWheelWidget_SetFlickInitSpeed(laListWheelWidget* whl,
uint32_t speed)

## laListWheelWidget_SetIndicatorArea Function

Configures the display area for the list selection indicator bars

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetIndicatorArea(laListWheelWidget* whl, uint32_t area);
```

### Returns

laResult - the operation result

### Description

This space is measured from the middle of the widget outward.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t area | the display area for the indicator bars |

### Function

laResult laListWheelWidget_SetIndicatorArea(laListWheelWidget* whl,

uint32_t area)

## laListWheelWidget_SetMaxMomentum Function

Configures the maximum momentum value for the wheel

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetMaxMomentum(laListWheelWidget* whl, uint32_t max);
```

### Returns

laResult - the operation result

### Description

When a wheel is in momenum mode addition drag/flick gestures will add more momentum to the wheel. The maximum momentum value governs the maximum speed at which the wheel can rotate at any single point in time.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t max | the maximum momentum value |

### Function

laResult laListWheelWidget_SetMaxMomentum(laListWheelWidget* whl,

uint32_t max)

## laListWheelWidget_SetMomentumFalloffRate Function

Configures the momentum falloff rate for the wheel

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetMomentumFalloffRate(laListWheelWidget* whl, uint32_t rate);
```

### Returns

laResult - the operation result

### Description

When a wheel is in momenum mode and during each rotation update tick the wheel will reduce its current momentum value by this falloff percentage. The higher this value is the faster a wheel will slow down. The wheel is limited to integer math so the lowest this value can be is one.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t max | the momentum falloff value |

### Function

laResult laListWheelWidget_SetMomentumFalloffRate(laListWheelWidget* whl,

uint32_t rate)

## laListWheelWidget_SetRotationUpdateRate Function

Configures the rotation update rate for a wheel

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetRotationUpdateRate(laListWheelWidget* whl, uint32_t ms);
```

### Returns

laResult - the operation result

### Description

When a wheel is in momenum mode it may be too costly to update with every Aria update loop call. This value can delay a wheel update. For instance, if Aria is updating every 20ms, the wheel can be set to update every 60ms and it will update approximately every three to four Aria updates. This can cut down on the number of repaints the wheel needs to perform and can also slow the wheel down it if is rotating too fast for the application to handle. This value is typically expressed in milliseconds.

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| uint32_t ms | the desired rotation update rate |

### Function

laResult laListWheelWidget_SetRotationUpdateRate(laListWheelWidget* whl,
uint32_t ms)

## laListWheelWidget_SetShaded Function

Configures the list to use gradient or flat background shading

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetShaded(laListWheelWidget* whl, laBool b);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| laBool b | true if gradient shading should be used |

### Function

laResult laListWheelWidget_SetShaded(laListWheelWidget* whl,
laBool b)

## laListWheelWidget_SetShowIndicators Function

Configures the list to display the selected item indicator bars

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetShowIndicators(laListWheelWidget* whl, laBool b);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWheelWidget* whl | the widget |
| laBool b | configures the indicator bar display state |

### Function

laResult laListWheelWidget_SetShowIndicators(laListWheelWidget* whl,
laBool b)

## laListWheelWidget_SetVisibleItemCount Function

Sets the number of visible items in the list. Must be greater than or equal to three and must be an odd number.

### File

libaria_widget_listwheel.h

### C

```
LIB_EXPORT laResult laListWheelWidget_SetVisibleItemCount(laListWheelWidget* whl, uint32_t cnt);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laListWidget* lst | the widget |
| uint32_t cnt | the desired number of items |

### Function

laResult laListWheelWidget_SetVisibleItemCount(laListWheelWidget* whl,
uint32_t cnt)

## laRadioButtonWidget_SetImageMargin Function

Sets the distance between the icon and text

### File

libaria_widget_radiobutton.h

### C

```
LIB_EXPORT laResult laRadioButtonWidget_SetImageMargin(laRadioButtonWidget* btn, uint32_t mg);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laRadioButtonWidget* btn | the widget |
| uint32_t mg | the distance value |

### Function

laResult laRadioButtonWidget_SetImageMargin(laRadioButtonWidget* btn,
uint32_t mg)

## laScreen_GetMirrored Function

Returns the mirror setting for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laBool laScreen_GetMirrored(laScreen* scr);
```

### Returns

laBool - the mirror setting

### Description

Returns the mirror setting for the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to reference |

### Function

laBool laScreen_GetMirrored(laScreen* scr)

## laScreen_SetMirrored Function

Sets the mirror setting for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laResult laScreen_SetMirrored(laScreen* scr, laBool mirr);
```

### Returns

laResult - the result of the operation

### Description

Sets the mirror setting for the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to modify |
| laBool | the mirror setting |

### Function

laResult laScreen_SetMirrored(laScreen* scr, laBool mirr)

## laString_Remove Function

Removes a number of characters from a string at a given index

### File

libaria_string.h

### C

```
LIB_EXPORT uint32_t laString_Remove(laString* str, uint32_t idx, uint32_t count);
```

### Returns

uint32_t - the number of characters removed

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to operate on |
| uint32_t idx | the index to remove from the number of characters to remove |

### Function

uint32_t laString_Remove( laString* str, uint32_t idx, uint32_t count)

## laTextFieldWidget_SetClearOnFirstEdit Function

Sets the flag to indicate that the text field will be cleared on first edit.

### File

libaria_widget_textfield.h

### C

```
LIB_EXPORT laResult laTextFieldWidget_SetClearOnFirstEdit(laTextFieldWidget* txt, laBool clear);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laTextFieldWidget* txt | the widget |
| laBool clear | the desired flag state |

### Function

laResult laTextFieldWidget_SetClearOnFirstEdit(laTextFieldWidget* txt,
laBool clear)

## laUtils_PickFromLayer Function

Finds the top-most visible widget in a layer at the given coordinates.

### File

libaria_utils.h

### C

```
LIB_EXPORT laWidget* laUtils_PickFromLayer(const laLayer* layer, int32_t x, int32_t y);
```

### Returns

laWidget* - the result widget

### Parameters

| Parameters | Description |
|---|---|
| int32_t x | the x coordinate of the pick point |
| int32_t y | the y coordinate of the pick point |

### Function

laWidget* laUtils_PickFromLayer(const laLayer* layer, int32_t x, int32_t y)

## laUtils_PointToLayerSpace Function

Converts a point from widget space into layer space

### File

libaria_utils.h

### C

```
void laUtils_PointToLayerSpace(laWidget* widget, GFX_Point* pnt);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Point* pnt | the point to convert |

### Function

void laUtils_PointToLayerSpace( laWidget* widget, GFX_Point* pnt)

## laUtils_ScreenToMirroredSpace Function

Takes a point in screen space and returns a transformed version in mirrored space.

### File

libaria_utils.h

### C

```
GFX_Point laUtils_ScreenToMirroredSpace(const GFX_Point* pnt, const GFX_Rect* rect, GFX_Orientation ori);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Point* point | the point to transform |
| const GFX_Rect* rect | the screen dimensionrectangle |
| GFX_Orientation | the orientation setting |

### Function

void laUtils_ScreenToMirroredSpace(const GFX_Point* point,
const GFX_Rect* rect,
GFX_Orientation ori)

## laUtils_ScreenToOrientedSpace Function

Takes a point in screen space and returns a transformed version in oriented space.

### File

libaria_utils.h

### C

```
GFX_Point laUtils_ScreenToOrientedSpace(const GFX_Point* pnt, const GFX_Rect* rect, GFX_Orientation ori);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Point* point | the point to transform |
| const GFX_Rect* rect | the screen dimensionrectangle |
| GFX_Orientation | the orientation setting |

### Function

void laUtils_ScreenToOrientedSpace(const GFX_Point* point,
const GFX_Rect* rect,
GFX_Orientation ori)

## laUtils_WidgetLocalRect Function

Returns the bounding rectangle of a widget in local space

### File

libaria_utils.h

### C

```
GFX_Rect laUtils_WidgetLocalRect(laWidget* widget);
```

### Returns

GFX_Rect - the bounding rectangle

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |

### Function

GFX_Rect laUtils_WidgetLocalRect(laWidget* widget)

## laWidget_GetBackgroundType Function

Return the property value 'background type' associated with the widget object

### File

libaria_widget.h

### C

```
LIB_EXPORT laBackgroundType laWidget_GetBackgroundType(laWidget* wgt);
```

### Returns

laBackgroundType - the current background type

### Description

Return the property value 'background type' associated with the widget object The background type property decides if the widget background is drawn and re-drawn. If background is none, the entire parent widget will be re-drawn in the event that the widget gets dirty and needs re-drawing.

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBackgroundType laWidget_GetBackgroundType(laWidget* wgt)

## laWidget_GetOptimizationFlags Function

Returns the optimization flags for the widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laBool laWidget_GetOptimizationFlags(laWidget* wgt);
```

### Returns

laBool - the flag value

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laBool laWidget_GetOptimizationFlags(laWidget* wgt)

## laWidget_SetBackgroundType Function

Set the property value 'background type' associated with the widget object

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetBackgroundType(laWidget* wgt, laBackgroundType type);
```

### Returns

laResult - the operation result

### Description

Set the property value 'draw background' associated with the widget object

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |
| laBackgroundType type | the desired background type |

### Function

laResult laWidget_SetBackgroundType(laWidget* wgt, laBackgroundType type)

## laWidget_SetOptimizationFlags Function

Sets the optimizations for a widget

### File

libaria_widget.h

### C

```
LIB_EXPORT laResult laWidget_SetOptimizationFlags(laWidget* wgt, uint32_t flags);
```

### Returns

laResult - the operation result

### Description

See the optimizations enum for a descriptions of the individual flags

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget |

### Function

laResult laWidget_SetOptimizationFlags(laWidget* wgt, uint32_t flags)

## laRectangleWidget_SetThickness Function

Sets the rectangle border thickness setting

### File

libaria_widget_rectangle.h

### C

```
LIB_EXPORT laResult laRectangleWidget_SetThickness(laRectangleWidget* rect, int32_t thk);
```

### Returns

laResult - the operation result

### Parameters

| Parameters | Description |
|---|---|
| laRectangleWidget* rect | the widget |
| int32_t thk | the thickness setting |

### Function

laResult laRectangleWidget_SetThickness(laRectangleWidget* rect,
int32_t thk)

## laString_DrawClipped Function

Wrapper around GFX Utility string draw function for Aria user interface library. Draws only a clipped area of a string. Internal use only.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_DrawClipped(laString* str, int32_t strX, int32_t strY, int32_t strWidth, int32_t
strHeight, int32_t x, int32_t y, GFXU_ExternalAssetReader** reader);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to draw |
| int32_t strX | clipped x position |
| int32_t strY | clipped y position |
| int32_t strWidth | clipped rectangle width |
| int32_t strHeight | clipped rectangle height |
| int32_t x | x position to render at |
| int32_t y | y position to render at |
| GFXU_ExternalAssetReader** reader | external reader state machine, if string font is located external |

### Function

void laString_DrawClipped( laString* str,
int32_t strX,
int32_t strY,
int32_t strWidth,
int32_t strHeight,
int32_t x,
int32_t y,
GFXU_ExternalAssetReader** reader)

## laString_IsEmpty Function

Returns a boolean indicating if the provided string contains data or has a link to the string table.

### File

libaria_string.h

### C

```
LIB_EXPORT laBool laString_IsEmpty(const laString* str);
```

### Returns

laBool - LA_TRUE if the string has data, LA_FALSE if not

### Parameters

| Parameters | Description |
|---|---|
| const laString* str | the string to analyze |

### Function

laBool laString_IsEmpty(laString* str)

## laUtils_GetNextHighestWidget Function

Gets the next highest Z order widget in the tree from 'wgt'

### File

libaria_utils.h

### C

```
laWidget* laUtils_GetNextHighestWidget(laWidget* wgt);
```

### Returns

laWidget* - the next highest widget or NULL if 'wgt' is already the highest

### Parameters

| Parameters | Description |
|---|---|
| laWidget* wgt | the widget to analyze |

### Function

laBool laUtils_GetNextHighestWidget(laWidget* wgt)

## laUtils_RectFromLayerSpace Function

Converts a rectangle from layer space to widget local space

### File

libaria_utils.h

### C

```
void laUtils_RectFromLayerSpace(laWidget* widget, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |
| GFX_Rect* rect | the rectangle to convert |

### Function

void laUtils_RectFromLayerSpace( laWidget* widget, GFX_Rect* rect)

## laUtils_WidgetIsOccluded Function

### File

libaria_utils.h

### C

```
laBool laUtils_WidgetIsOccluded(laWidget* wgt, const GFX_Rect* rect);
```

### Description

This is function laUtils_WidgetIsOccluded.

## laUtils_WidgetLayerRect Function

Returns the bounding rectangle of a widget in layer space

### File

libaria_utils.h

### C

```
GFX_Rect laUtils_WidgetLayerRect(laWidget* widget);
```

### Returns

GFX_Rect - the bounding rectangle

### Parameters

| Parameters | Description |
|---|---|
| laWidget* widget | the subject widget |

### Function

GFX_Rect laUtils_WidgetLayerRect(laWidget* widget)

## laWindowWidget_GetIconRect Function

### File

libaria_widget_window.h

### C

```
void laWindowWidget_GetIconRect(laWindowWidget* win, GFX_Rect* imgRect, GFX_Rect* imgSrcRect);
```

### Description

This is function laWindowWidget_GetIconRect.

## laWindowWidget_GetTextRect Function

### File

libaria_widget_window.h

### C

```
void laWindowWidget_GetTextRect(laWindowWidget* win, GFX_Rect* textRect, GFX_Rect* drawRect);
```

### Description

This is function laWindowWidget_GetTextRect.

## laWindowWidget_GetTitleBarRect Function

### File

libaria_widget_window.h

### C

```
void laWindowWidget_GetTitleBarRect(laWindowWidget* win, GFX_Rect* barRect);
```

### Description

internal use only

## laContext_IsDrawing Function

Indicates if any layers of the active screen are currently drawing a frame.

### File

libaria_context.h

### C

```
LIB_EXPORT laBool laContext_IsDrawing();
```

### Returns

laResult

### Description

Indicates if any layers are currently drawing a frame. Because frame updates can happen long after making changes to the UI state it is best to only make updates to the state of a layer tree only when the layer is not drawing.

Requires an active context and active screen.

### Function

laBool laContext_IsDrawing()

## laContext_IsLayerDrawing Function

Indicates if the layer at the given index of the active screen is currently drawing.

### File

libaria_context.h

### C

```
LIB_EXPORT laBool laContext_IsLayerDrawing(uint32_t idx);
```

### Returns

laResult

### Description

Indicates if the layer at the given index is currently drawing a frame. Because frame updates can happen long after making changes to the UI state it is best to only make updates to the state of a layer tree only when the layer is not drawing.

Requires an active context and active screen.

### Parameters

| Parameters | Description |
|---|---|
| uint32_t idx | the index of the layer to query |

### Function

laBool laContext_IsLayerDrawing(uint32_t idx)

## laLayer_IsDrawing Function

Queries a layer to find out if it is currently drawing a frame.

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_IsDrawing(laLayer* layer);
```

### Returns

laBool - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |

### Function

laBool laLayer_IsDrawing(laLayer* layer)

## laLayer_GetInputRect Function

Gets the layer's input rectangle.

### File

libaria_layer.h

### C

```
LIB_EXPORT GFX_Rect laLayer_GetInputRect(const laLayer* layer);
```

### Returns

GFX_Rect - the input rectangle

### Description

Gets the layer's input rectangle.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

GFX_Rect laLayer_GetInputRect(const laLayer* layer)

## laLayer_GetInputRectLocked Function

Gets the layer's input rect locked flag

### File

libaria_layer.h

### C

```
LIB_EXPORT laBool laLayer_GetInputRectLocked(const laLayer* layer);
```

### Returns

laBool - the state of the layer's input rect locked flag

### Description

This flag controls whether the layer input rectangle is locked to match the size of the layer's actual dimensions.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

laBool laLayer_GetInputRectLocked(const laLayer* layer)

## laLayer_SetInputRect Function

Sets the layer's input rect dimensions.

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetInputRect(laLayer* layer, int32_t x, int32_t y, int32_t width, int32_t
height);
```

### Returns

laResult - the result of the operation

### Description

Sets the layer's input rect dimensions. This rectangle controls the input area of the layer. Some use cases may require a layer to accept input even if the input is outside of the physical dimensions of the layer. One example is a touch glass that is larger than the size of a display. Widgets may need to be placed in this invisible external area and still be capable of receiving input events.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |
| int32_t x | the x position of the rectangle |
| int32_t y | the y position of the rectangle |
| int32_t width | the width of the rectangle |
| int32_t height | the height of the rectangle |

### Function

laResult laLayer_SetInputRect(laLayer* layer, laBool enable)

## laLayer_SetInputRectLocked Function

Sets the layer's input rect locked flag.

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_SetInputRectLocked(laLayer* layer, laBool locked);
```

### Returns

laResult - the result of the operation

### Description

Sets the layer's input rect locked flag. This flag controls whether the input rectangle is locked to match the size of the layer's actual dimensions. When enabled, any change to the layer's size will be propagated to the input area as well. The default value is true.

### Parameters

| Parameters | Description |
|---|---|
| const laLayer* layer | the layer |

### Function

laResult laLayer_SetInputRectLocked(laLayer* layer, laBool enable)

## laScreen_GetLayerSwapSync Function

Returns the layer swap sync setting for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laBool laScreen_GetLayerSwapSync(laScreen* scr);
```

### Returns

laBool - the sync setting

### Description

Returns the layer swap sync setting for the specified screen

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to reference |

### Function

laBool laScreen_GetLayerSwapSync(laScreen* scr)

## laScreen_SetLayerSwapSync Function

Sets the layer swap sync setting for the specified screen

### File

libaria_screen.h

### C

```
LIB_EXPORT laResult laScreen_SetLayerSwapSync(laScreen* scr, laBool sync);
```

### Returns

laResult - the result of the operation

### Description

Layer synchronization allows for the configuration timing of the buffer swap chain. In the case where multiple layers are being modified at the same time, it is often desirable to have the updates appear on the display at the same time. Layer sync will gate all layer swapping until all dirty layers have finished drawing. All layers will then swap at same time.

### Parameters

| Parameters | Description |
|---|---|
| laScreen* scr | the screen to modify |
| laBool | the sync setting |

### Function

laResult laScreen_SetLayerSwapSync(laScreen* scr, laBool sync)

## laWidget_DeleteAllDescendants Function

Deletes all of the descendants of the given widget.

### File

libaria_widget.h

### C

```
LIB_EXPORT void laWidget_DeleteAllDescendants(laWidget* wgt);
```

### Returns

void

### Description

All descendants of this widget are removed and deleted.

### Function

void laWidget_DeleteAllDescendants( laWidget* wgt)

## laImageWidget_SetCallBackEnd Function

### File

libaria_widget_image.h

### C

```
void laImageWidget_SetCallBackEnd(laImageWidget* image, laImageWidget_DrawEventCallback cb);
```

### Description

This is function laImageWidget_SetCallBackEnd.

## laImageWidget_SetCallBackStart Function

### File

libaria_widget_image.h

### C

```
void laImageWidget_SetCallBackStart(laImageWidget* image, laImageWidget_DrawEventCallback cb);
```

### Description

This is function laImageWidget_SetCallBackStart.

## laString_DrawSubStringClipped Function

Wrapper around GFX Utility string draw function for Aria user interface library. Draws the substring between the start and end offset, and draws only the section of the string within the clipping rectangle. Internal use only.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_DrawSubStringClipped(laString* str, uint32_t start, uint32_t end, int32_t clipX,
int32_t clipY, int32_t clipWidth, int32_t clipHeight, int32_t x, int32_t y, GFXU_ExternalAssetReader**
reader);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to draw |
| uint32_t start | the start position of the substring |
| uint32_t end | the end position of the substring |
| int32_t strX | clipped x position |
| int32_t strY | clipped y position |
| int32_t strWidth | clipped rectangle width |
| int32_t strHeight | clipped rectangle height |
| int32_t x | x position to render at |
| int32_t y | y position to render at |

| | |
|---|---|
| GFXU_ExternalAssetReader** reader | external reader state machine, if string font is located external |

### Function

void laString_DrawSubStringClipped( laString* str,
int32_t strX,
int32_t strY,
int32_t strWidth,
int32_t strHeight,
int32_t x,
int32_t y,
GFXU_ExternalAssetReader** reader)

## laString_GetLineRect Function

Calculates the rectangle for a line in a string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. The line ends when a line feed or end of string is reached.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_GetLineRect(laString* str, uint32_t start, GFX_Rect* rect, uint32_t * end);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| uint32_t start | the start offset of the line in the string |
| GFX_Rect* rect | the calculated string rectangle result |
| uint32_t * end | the calculated end of the line (including line feed or end of string) |

### Function

void laString_GetLineRect( laString* str, uint32_t offset, GFX_Rect* rect, uint32_t * endoffset)

## laString_GetMultiLineRect Function

Calculates the rectangle for a given multi-line string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. The height the sum of the heights of the bounding rectangles for each line and the width is the widest among the bounding rectangles.

### File

libaria_string.h

### C

```
LIB_EXPORT void laString_GetMultiLineRect(laString* str, GFX_Rect* rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| laString* str | the string to reference |
| GFX_Rect* rect | the calculated string rectangle result |

### Function

void laString_GetMultiLineRect( laString* str, GFX_Rect* rect)

### *b) Data Types and Constants*

#### laContext_t Structure

An instance of the Aria user interface library.

## File

libaria_context.h

## C

```
struct laContext_t {
  GFX_Display displayIndex;
  void* gfxContext;
  GFXU_MemoryIntf memIntf;
  laPreemptionLevel preemptLevel;
  laArray screenList;
  laScreen* activeScreen;
  uint32_t frameState;
  uint32_t currentLayer;
  laInputState input;
  GFXU_StringTableAsset* stringTable;
  uint32_t languageID;
  uint32_t widgetIDs;
  laScheme defaultScheme;
  GFX_ColorMode colorMode;
  laWidget* focus;
  laEditWidget* edit;
  laContext_ActiveScreenChangedCallback_FnPtr screenChangedCB;
  laContext_LanguageChangedCallback_FnPtr languageChangedCB;
};
```

## Members

| Members | Description |
|---|---|
| GFX_Display displayIndex; | the display the library is using |
| void* gfxContext; | the HAL context the library owns |
| GFXU_MemoryIntf memIntf; | the memory interface the library is using |
| laPreemptionLevel preemptLevel; | the preemption level the library is using |
| laArray screenList; | the list of the screens in the context |
| laScreen* activeScreen; | the currently active screen |
| uint32_t frameState; | the context frame state |
| uint32_t currentLayer; | the current drawing layer |
| laInputState input; | the input state of the instance |
| GFXU_StringTableAsset* stringTable; | the string table for the instance |
| uint32_t languageID; | the currently active language |
| uint32_t widgetIDs; | the next unique widget ID |
| laScheme defaultScheme; | an internal default scheme that widgets use by default if the user doesn't set one |
| GFX_ColorMode colorMode; | the color mode the library uses for all layers |
| laWidget* focus; | the widget that currently has focus |
| laEditWidget* edit; | the widget that is currently receiving edit events |
| laContext_ActiveScreenChangedCallback_FnPtr screenChangedCB; | screen changed callback |
| laContext_LanguageChangedCallback_FnPtr languageChangedCB; | language changed callback |

## Description

Structure: laContext

The context represents an discrete instance of Aria user interface library. The library is designed to be multi-instance and fully re-entrant. The entire state of the library is stored and referenced through the context pointer.

## Remarks

None.

## laList_t Structure

Linked list definition

### File

libaria_list.h

### C

```
struct laList_t {
  laListNode* head;
  laListNode* tail;
  size_t size;
};
```

### Description

Structure: laList_t

### Remarks

None.

## laPreemptionLevel Enumeration

libaria pre-emption level values

### File

libaria_common.h

### C

```
enum laPreemptionLevel {
  LA_PREEMPTION_LEVEL_0,
  LA_PREEMPTION_LEVEL_1,
  LA_PREEMPTION_LEVEL_2
};
```

### Members

| Members | Description |
|---|---|
| LA_PREEMPTION_LEVEL_0 | draw cycle always completes |
| LA_PREEMPTION_LEVEL_1 | preempts after each widget fully draws |
| LA_PREEMPTION_LEVEL_2 | preempts after each widget draw step completes |

### Description

Enumeration: laPreemptionLevel

libaria pre-emption level values

### Remarks

None.

## laRelativePosition Enumeration

libaria relative position values

### File

libaria_common.h

### C

```
enum laRelativePosition {
  LA_RELATIVE_POSITION_LEFTOF,
  LA_RELATIVE_POSITION_ABOVE,
  LA_RELATIVE_POSITION_RIGHTOF,
  LA_RELATIVE_POSITION_BELOW,
  LA_RELATIVE_POSITION_BEHIND
};
```

## Description

Enumeration: laRelativePosition

libaria relative position values

## Remarks

None.

### GFXU_StringTableAsset Structure

Describes a string table asset. There is typically only ever one of these defined at any one time.

header - standard asset header languageCount - the number of languages in the string table stringCount - the number of strings in the string table stringIndexTable - the pointer to the string index table. the string index table is a table that contains all of the unique strings defined in the string table. fontTable - the font table contains an array of pointers to all defined font assets that the string table references fontIndexTable - the font index table is a table that maps strings to font indices which can then be used to get an actual font pointer from the font table encodingMode - indicates how strings are encoded in the stringIndexTable

## File

gfxu_string.h

## C

```
typedef struct GFXU_StringTableAsset_t {
  GFXU_AssetHeader header;
  uint32_t languageCount;
  uint32_t stringCount;
  uint8_t* stringIndexTable;
  GFXU_FontAsset** fontTable;
  uint8_t* fontIndexTable;
  GFXU_StringEncodingMode encodingMode;
} GFXU_StringTableAsset;
```

## Description

Structure: GFXU_StringTableAsset_t

### laBool Enumeration

libaria bool values

## File

libaria_common.h

## C

```
typedef enum laBool_t {
  LA_FALSE = 0,
  LA_TRUE
} laBool;
```

## Description

Enumeration: laBool

libaria bool values

## Remarks

None.

### laContext Type

An instance of the Aria user interface library.

## File

libaria_string.h

## C

```
typedef struct laContext_t laContext;
```

## Description

Structure: laContext

The context represents an discrete instance of Aria user interface library. The library is designed to be multi-instance and fully re-entrant. The entire state of the library is stored and referenced through the context pointer.

## Remarks

None.

### laContext_ActiveScreenChangedCallback_FnPtr Type

Callback pointer for the active screen change notification.

## File

libaria_context.h

## C

```
typedef void (* laContext_ActiveScreenChangedCallback_FnPtr)(int32_t, int32_t);
```

## Description

Type: laContext_ActiveScreenChangedCallback_FnPtr

Callback pointer for the active screen change notification.

### laContext_LanguageChangedCallback_FnPtr Type

Callback pointer for when the language change event occurs.

## File

libaria_context.h

## C

```
typedef void (* laContext_LanguageChangedCallback_FnPtr)(uint32_t);
```

## Description

Type: laContext_LanguageChangedCallback_FnPtr

Callback pointer for when the language change event occurs.

### laEditWidget Structure

Specifies the edit widget structure to manage all properties and events associated with edit widgets

## File

libaria_editwidget.h

## C

```
typedef struct laEditWidget_t {
  laWidget widget;
  laEditWidget_StartEdit_FnPtr startEdit;
  laEditWidget_EndEdit_FnPtr endEdit;
  laEditWidget_Clear_FnPtr clear;
  laEditWidget_Accept_FnPtr accept;
  laEditWidget_Set_FnPtr set;
  laEditWidget_Append_FnPtr append;
  laEditWidget_Backspace_FnPtr backspace;
} laEditWidget;
```

## Description

Structure: laEditWidget_t

Edit widgets are a subset of normal widgets that are capable of receiving edit events from the UI kernel. Specialized widgets are capable of broadcasting edit events and the active edit event will react to them.

## Remarks

None.

### laEditWidget_Accept_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef void (* laEditWidget_Accept_FnPtr)(laEditWidget*);
```

#### Description

This is type laEditWidget_Accept_FnPtr.

### laEditWidget_Append_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef void (* laEditWidget_Append_FnPtr)(laEditWidget*, laString);
```

#### Description

This is type laEditWidget_Append_FnPtr.

### laEditWidget_Backspace_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef void (* laEditWidget_Backspace_FnPtr)(laEditWidget*);
```

#### Description

This is type laEditWidget_Backspace_FnPtr.

### laEditWidget_Clear_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef void (* laEditWidget_Clear_FnPtr)(laEditWidget*);
```

#### Description

This is type laEditWidget_Clear_FnPtr.

### laEditWidget_EndEdit_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef void (* laEditWidget_EndEdit_FnPtr)(laEditWidget*);
```

#### Description

This is type laEditWidget_EndEdit_FnPtr.

### laEditWidget_Set_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef void (* laEditWidget_Set_FnPtr)(laEditWidget*, laString);
```

#### Description

This is type laEditWidget_Set_FnPtr.

### laEditWidget_StartEdit_FnPtr Type

#### File

libaria_editwidget.h

#### C

```
typedef laResult (* laEditWidget_StartEdit_FnPtr)(laEditWidget*);
```

#### Description

This is type laEditWidget_StartEdit_FnPtr.

### laHAlignment Enumeration

libaria horizontal alignment values

#### File

libaria_common.h

#### C

```
typedef enum {
  LA_HALIGN_LEFT,
  LA_HALIGN_CENTER,
  LA_HALIGN_RIGHT
} laHAlignment;
```

#### Description

Enumeration: laHAlignment

libaria horizontal alignment values

#### Remarks

None.

### laList Type

Linked list definition

#### File

libaria_utils.h

#### C

```
typedef struct laList_t laList;
```

#### Description

Structure: laList_t

#### Remarks

None.

## laListNode Structure

Linked list node definition

### File

libaria_list.h

### C

```
typedef struct laListNode_t {
  struct laListNode_t* next;
  void* val;
} laListNode;
```

### Description

Structure: laListNode_t

### Remarks

None.

## laMargin Structure

libaria margin values

### File

libaria_common.h

### C

```
typedef struct laMargin_t {
  uint8_t left;
  uint8_t top;
  uint8_t right;
  uint8_t bottom;
} laMargin;
```

### Description

Enumeration: laMargin

libaria margin values

### Remarks

None.

## laResult Enumeration

libaria results (success and failure codes).

### File

libaria_common.h

### C

```
typedef enum laResult_t {
  LA_FAILURE = -1,
  LA_SUCCESS = 0
} laResult;
```

### Description

Enumeration: laResult

Various definitions for success and failure codes.

### Remarks

None.

## laScreen Structure

The structure to maintain the screen related variables and event handling

### File

libaria_screen.h

### C

```
typedef struct laScreen_t {
  uint32_t id;
  laString name;
  laBool persistent;
  laScreen_CreateCallback_FnPtr createCB;
  laBool created;
  laLayer* layers[LA_MAX_LAYERS];
  laScreenOrientation orientation;
  laBool mirrored;
  laBool layerSwapSync;
  laScreen_ShowHideCallback_FnPtr showCB;
  laScreen_ShowHideCallback_FnPtr hideCB;
} laScreen;
```

### Members

| Members | Description |
|---|---|
| uint32_t id; | the id of the screen |
| laString name; | the name of the screen |
| laBool persistent; | indicates that the screen should not free its widgets when it hides |
| laScreen_CreateCallback_FnPtr createCB; | the function that is called to create the contents of the screen |
| laBool created; | indicates if the screen currently exists |
| laLayer* layers[LA_MAX_LAYERS]; | the layer array for the screen |
| laScreenOrientation orientation; | the orientation of the screen |
| laBool mirrored; | the mirror flag of the screen |
| laBool layerSwapSync; | the layerSwapSync flag of the screen |
| laScreen_ShowHideCallback_FnPtr showCB; | a callback that is called when the screen is shown |
| laScreen_ShowHideCallback_FnPtr hideCB; | a callback that is called when the screen is hidden |

### Description

Structure: laScreen_t

Maintains the layers associated with the screen. Marks the screen as persistent or not, which either destroys the screen when changed or preserves it for future reloading. Allocates and manages the event handling when screen change / show / hide events occur.

### Remarks

None.

## laString Structure

String definition

### File

libaria_string.h

### C

```
typedef struct laString_t {
  GFXU_CHAR* data;
  uint32_t capacity;
  uint32_t length;
  GFXU_FontAsset* font;
  int32_t table_index;
} laString;
```

## Members

| Members | Description |
|---|---|
| GFXU_CHAR* data; | local string data storage |
| uint32_t capacity; | actual memory capacity of the string |
| uint32_t length; | actual length of the string, typically this is capacity - 1, but can be less. |
| GFXU_FontAsset* font; | the font that contains the glyph raster data for this string |
| int32_t table_index; | if this is not LA_STRING_NULLIDX then this string is referencing an index in the string table. string table references are read-only but can be extracted to local modifiable versions |

## Description

Structure: laString_t

## Remarks

None.

### laVAlignment Enumeration

libaria vertical alignment values

## File

libaria_common.h

## C

```
typedef enum {
  LA_VALIGN_TOP,
  LA_VALIGN_MIDDLE,
  LA_VALIGN_BOTTOM
} laVAlignment;
```

## Description

Enumeration: laVAlignment

libaria vertical alignment values

## Remarks

None.

### laButtonWidget_t Structure

Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed an released state.

## File

libaria_widget_button.h

## C

```
struct laButtonWidget_t {
  laWidget widget;
  laButtonState state;
  uint8_t toggleable;
  laString text;
  laHAlignment halign;
  laVAlignment valign;
  GFXU_ImageAsset* pressedImage;
  GFXU_ImageAsset* releasedImage;
  laRelativePosition imagePosition;
  uint32_t imageMargin;
  int32_t pressedOffset;
  laButtonWidget_PressedEvent pressedEvent;
  laButtonWidget_ReleasedEvent releasedEvent;
  GFXU_ExternalAssetReader* reader;
};
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | base widget header |
| laButtonState state; | button state |
| uint8_t toggleable; | indicates if the button is toggleable |
| laString text; | the string that holds the button text |
| laHAlignment halign; | horizontal alignment of the button |
| laVAlignment valign; | vertical alignment of the button |
| GFXU_ImageAsset* pressedImage; | button pressed icon image |
| GFXU_ImageAsset* releasedImage; | button released icon image |
| laRelativePosition imagePosition; | icon position in relation to text |
| uint32_t imageMargin; | distance between text and icon |
| int32_t pressedOffset; | pressed text offset distance |
| laButtonWidget_PressedEvent pressedEvent; | pressed event callback |
| laButtonWidget_ReleasedEvent releasedEvent; | released event callback |
| GFXU_ExternalAssetReader* reader; | external asset reader state |

## Description

Structure: laButtonWidget_t

## Remarks

None.

### laLayer_t Structure

Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent.

## File

libaria_layer.h

## C

```
struct laLayer_t {
  laWidget widget;
  laScreen* screen;
  laBool deleting;
  uint32_t bufferCount;
  laLayerBuffer buffers[GFX_MAX_BUFFER_COUNT];
  laBool alphaEnable;
  laBool maskEnable;
  GFX_Color maskColor;
  laBool vsync;
  laRectArray prevDamageRects;
  laRectArray currentDamageRects;
  laRectArray pendingDamageRects;
  laRectArray scratchRectList;
  laRectArray frameRectList;
  uint32_t frameRectIdx;
  GFX_Rect clippedDrawingRect;
  laBool drawingPrev;
  laLayerFrameState frameState;
  uint32_t layerDrawCount;
  uint32_t frameDrawCount;
  GFX_Rect inputRect;
  laBool inputRectLocked;
  laBool allowInputPassThrough;
  uint32_t deltaTime;
};
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | base widget |
| laScreen* screen; | owning screen pointer |

| | |
|---|---|
| laBool deleting; | flag indicating that no changes should be made to the layer because it is in the process of being deleted |
| uint32_t bufferCount; | number of buffers in the layer |
| laLayerBuffer buffers[GFX_MAX_BUFFER_COUNT]; | buffer array |
| laBool alphaEnable; | layer-based alpha blending enable flag |
| laBool maskEnable; | layer-based color masking enable flag |
| GFX_Color maskColor; | layer-based masking color value |
| laBool vsync; | layer vsync flag |
| laRectArray prevDamageRects; | previous damaged rectangle list |
| laRectArray currentDamageRects; | queued damaged rectangle list |
| laRectArray pendingDamageRects; | pending damaged rectangle list these are rectangles added during a frame in progress |
| laRectArray scratchRectList; | used for rectangle culling phase |
| laRectArray frameRectList; | this of rects to draw for a frame GFX_Rect currentDrawingRect; // the current damage rectangle |
| GFX_Rect clippedDrawingRect; | the current damage rectangle clipped to the currently rendering widget |
| laBool drawingPrev; | indicates if the layer is currently drawing from its previous rectangle array |
| laLayerFrameState frameState; | the current frame render state of the layer |
| uint32_t layerDrawCount; | the number of times this layer has drawn |
| uint32_t frameDrawCount; | the number of widgets that have rendered on this layer this frame |
| GFX_Rect inputRect; | layer input area |
| laBool inputRectLocked; | input area matches layer dimensions |
| laBool allowInputPassThrough; | indicates that input events should be propagated through the layer node to left siblings |
| uint32_t deltaTime; | stores delta time for updates that happen during rendering |

## Description

Structure: laLayer_t

## Remarks

None.

### laRadioButtonGroup_t Structure

Defines the structure used for the Radio Button group.

## File

libaria_radiobutton_group.h

## C

```
struct laRadioButtonGroup_t {
  laArray buttonList;
  laBool initialized;
  laRadioButtonWidget* selected;
};
```

## Description

Structure laRadioButtonGroup_t

Defines the parameters required for a Radio Button group. Marks the current selected Radio button within the group

## Remarks

None.

### GFX_Point Type

A two dimensional Cartesian point.

## File

libaria_utils.h

### C

```
typedef struct GFX_Point_t GFX_Point;
```

### Description

Structure: GFX_Point_t

## GFX_Rect Type

A rectangle definition.

### File

libaria_utils.h

### C

```
typedef struct GFX_Rect_t GFX_Rect;
```

### Description

Structure: GFX_Rect_t

## laBorderType Enumeration

Specifies the different border types used for the widgets in the library

### File

libaria_widget.h

### C

```
typedef enum laBorderType_t {
  LA_WIDGET_BORDER_NONE,
  LA_WIDGET_BORDER_LINE,
  LA_WIDGET_BORDER_BEVEL,
  LA_WIDGET_BORDER_LAST = LA_WIDGET_BORDER_BEVEL
} laBorderType;
```

### Description

Enumeration: laBorderType_t

Specifies the different border types used for the widgets in the library

### Remarks

None.

## laButtonState Enumeration

Controls the button pressed state

### File

libaria_widget_button.h

### C

```
typedef enum laButtonState_t {
  LA_BUTTON_STATE_UP,
  LA_BUTTON_STATE_DOWN,
  LA_BUTTON_STATE_TOGGLED
} laButtonState;
```

### Description

Enumeration: laButtonState_t

## laButtonWidget Type

Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed an released state.

### File

libaria_widget_keypad.h

### C

```
typedef struct laButtonWidget_t laButtonWidget;
```

### Description

Structure: laButtonWidget_t

### Remarks

None.

## laButtonWidget_PressedEvent Type

### File

libaria_widget_button.h

### C

```
typedef void (* laButtonWidget_PressedEvent)(laButtonWidget*);
```

### Description

This is type laButtonWidget_PressedEvent.

## laButtonWidget_ReleasedEvent Type

### File

libaria_widget_button.h

### C

```
typedef void (* laButtonWidget_ReleasedEvent)(laButtonWidget*);
```

### Description

This is type laButtonWidget_ReleasedEvent.

## laCheckBoxWidget Structure

Implementation of a checkbox widget.

### File

libaria_widget_checkbox.h

### C

```
typedef struct laCheckBoxWidget_t {
  laWidget widget;
  laBool checked;
  laString text;
  laHAlignment halign;
  laVAlignment valign;
  GFXU_ImageAsset* checkedImage;
  GFXU_ImageAsset* uncheckedImage;
  laRelativePosition imagePosition;
  uint32_t imageMargin;
  GFXU_ExternalAssetReader* reader;
  laCheckBoxWidget_CheckedEvent checkedEvent;
  laCheckBoxWidget_CheckedEvent uncheckedEvent;
} laCheckBoxWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | base class properties |
| laBool checked; | the state of the box |
| laString text; | the text of the box |

| | |
|---|---|
| laHAlignment halign; | the horizontal alignment of the box contents |
| laVAlignment valign; | the vertical alignment of the box contents |
| GFXU_ImageAsset* checkedImage; | pointer to a custom image to use for the checked image |
| GFXU_ImageAsset* uncheckedImage; | pointer to a custom image to use for the unchecked image |
| laRelativePosition imagePosition; | postition of the image relative to the text of the box |
| uint32_t imageMargin; | the distance between the image and the text |
| GFXU_ExternalAssetReader* reader; | an external asset reader pointer |
| laCheckBoxWidget_CheckedEvent checkedEvent; | callback for checked events |
| laCheckBoxWidget_CheckedEvent uncheckedEvent; | callback for unchecked events |

## Description

Structure: laCheckBoxWidget_t

A check box widget contains an interactive two-state box indicating on or off. The check box may also contain descriptive text. Custom images for the check box may be used in place of the default box graphic.

## Remarks

None.

### laCheckBoxWidget_CheckedEvent Type

## File

libaria_widget_checkbox.h

## C

```
typedef void (* laCheckBoxWidget_CheckedEvent)(laCheckBoxWidget*);
```

## Description

This is type laCheckBoxWidget_CheckedEvent.

### laCheckBoxWidget_UncheckedEvent Type

## File

libaria_widget_checkbox.h

## C

```
typedef void (* laCheckBoxWidget_UncheckedEvent)(laCheckBoxWidget*);
```

## Description

This is type laCheckBoxWidget_UncheckedEvent.

### laCircleWidget Structure

Implementation of a circle widget.

## File

libaria_widget_circle.h

## C

```
typedef struct laCircleWidget_t {
  laWidget widget;
  int32_t x;
  int32_t y;
  int32_t radius;
} laCircleWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | base widget header |

| | |
|---|---|
| int32_t x; | the origin x cooridnate |
| int32_t y; | the origin y coordinate |
| int32_t radius; | the radius of the circle |

## Description

Structure: laCircleWidget_t

A circle widget draws a circle of the specified origin and radius inside the widget bounds. All coordinates are expressed in local widget space.

The color of the circle is determined by the widget scheme's 'foreground' color.

## Remarks

None.

### laDrawSurfaceWidget Structure

Implementation of a Drawsurface widget.

## File

libaria_widget_drawsurface.h

## C

```
typedef struct laDrawSurfaceWidget_t {
  laWidget widget;
  laDrawSurfaceWidget_DrawCallback cb;
} laDrawSurfaceWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | the widget base class |
| laDrawSurfaceWidget_DrawCallback cb; | the draw callback |

## Description

Structure: laDrawSurfaceWidget_t

A draw surface widget is a special widget that allows an application to perform custom HAL draw calls during Aria's paint loop. To use, create and add a draw surface widget to the desired place in the widget tree. Then register for the callback through the API 'laDrawSurfaceWidget_SetDrawCallback'. This callback occurs during the paint loop. The application should then be free to adjust the HAL draw state and issue draw calls as desired. The HAL layer, buffer, or context state must not be adjusted in any way.

It is also important to not stall for too long during the draw callback.

## Remarks

None.

### laDrawSurfaceWidget_DrawCallback Type

## File

libaria_widget_drawsurface.h

## C

```
typedef laBool (* laDrawSurfaceWidget_DrawCallback)(laDrawSurfaceWidget* sfc, GFX_Rect* bounds);
```

## Description

This is type laDrawSurfaceWidget_DrawCallback.

### laEvent Structure

Basic UI event definition

## File

libaria_event.h

## C

```
typedef struct laEvent_t {
```

```
  laEventID id;
} laEvent;
```

### Description

Structure: laEvent_t

## laEvent_FilterEvent Type

Function pointer to define an event filter. Event filters allow a receiver to discard undesirable events

### File

libaria_event.h

### C

```
typedef laBool (* laEvent_FilterEvent)(laEvent*);
```

### Description

Function Pointer: laEvent_FilterEvent

## laEventID Enumeration

Defines internal event type IDs

### File

libaria_event.h

### C

```
typedef enum laEventID_t {
  LA_EVENT_NONE,
  LA_EVENT_SCREEN_CHANGE,
  LA_EVENT_TOUCH_DOWN,
  LA_EVENT_TOUCH_UP,
  LA_EVENT_TOUCH_MOVED
} laEventID;
```

### Members

| Members | Description |
|---|---|
| LA_EVENT_NONE | internal events |

### Description

Enumeration: laEventID

## laEventState Structure

Structure to manage the event lists, state and call back pointers

### File

libaria_event.h

### C

```
typedef struct laEventState_t {
  OSAL_SEM_HANDLE_TYPE eventCountSem;
  OSAL_MUTEX_HANDLE_TYPE eventLock;
  laList events;
  laEvent_FilterEvent filter;
} laEventState;
```

### Description

Structure: laEventState_t

### Remarks

None.

## laGestureID Enumeration

Placeholder for eventual gesture support.

### File

libaria_input.h

### C

```
typedef enum laGestureID_t {
  LA_GESTURE_NONE = 0
} laGestureID;
```

### Description

Enumeration: laGestureID

### Remarks

None.

## laGradientWidget Structure

Gradient widget struct definition.

### File

libaria_widget_gradient.h

### C

```
typedef struct laGradientWidget_t {
  laWidget widget;
  laGradientWidgetDirection dir;
} laGradientWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laGradientWidgetDirection dir; | gradient direction |

### Description

Enumeration: laGradientWidget_t

### Remarks

None.

## laGradientWidgetDirection Enumeration

Implementation of a gradient widget.

### File

libaria_widget_gradient.h

### C

```
typedef enum laGradientWidgetDirection_t {
  LA_GRADIENT_DIRECTION_RIGHT,
  LA_GRADIENT_DIRECTION_DOWN,
  LA_GRADIENT_DIRECTION_LEFT,
  LA_GRADIENT_DIRECTION_UP
} laGradientWidgetDirection;
```

### Description

Enumeration: laGradientWidgetDirection_t

A gradient widget is similar to a panel widget with the exception that it can draw a gradient color for its background. This operation can be more costly than drawing a solid color and should be used sparingly.

Gradient uses 'foreground' and 'foreground inactive' as its interpolated background draw colors.

## Remarks

None.

### laGroupBoxWidget Structure

Group box struct definition.

## File

libaria_widget_groupbox.h

## C

```
typedef struct laGroupBoxWidget_t {
  laWidget widget;
  laString text;
  laHAlignment halign;
  GFXU_ExternalAssetReader* reader;
} laGroupBoxWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laString text; | group box title text |
| laHAlignment halign; | group box text alignment |
| GFXU_ExternalAssetReader* reader; | asset reader |

## Description

Enumeration: laGroupBoxWidget_t

A group box is a widget that is similar to a basic panel but provides a line border and title text. Used for grouping and describing widgets of similar function.

## Remarks

None.

### laImageSequenceEntry Structure

Image sequence entry definition

## File

libaria_widget_imagesequence.h

## C

```
typedef struct laImageSequenceEntry_t {
  GFXU_ImageAsset* image;
  uint32_t delay;
  laHAlignment halign;
  laVAlignment valign;
} laImageSequenceEntry;
```

## Members

| Members | Description |
|---|---|
| GFXU_ImageAsset* image; | image asset pointer |
| uint32_t delay; | how many time units to display this entry |
| laHAlignment halign; | the horizontal alignment for this entry |
| laVAlignment valign; | the vertical alignment for this entry |

## Description

Enumeration: laImageSequenceEntry_t

Defines a single entry for the image sequence widget

## Remarks

None.

## laImageSequenceImageChangedEvent_FnPtr Type

### File

libaria_widget_imagesequence.h

### C

```
typedef void (* laImageSequenceImageChangedEvent_FnPtr)(laImageSequenceWidget*);
```

### Description

This is type laImageSequenceImageChangedEvent_FnPtr.

## laImageSequenceWidget Structure

Image sequence widget struct definition

### File

libaria_widget_imagesequence.h

### C

```
typedef struct laImageSequenceWidget_t {
  laWidget widget;
  uint32_t count;
  laImageSequenceEntry* images;
  int32_t activeIdx;
  laBool playing;
  uint32_t time;
  laBool repeat;
  laImageSequenceImageChangedEvent_FnPtr cb;
  GFXU_ExternalAssetReader* reader;
} laImageSequenceWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| uint32_t count; | number of image entries for this widget |
| laImageSequenceEntry* images; | image entry array |
| int32_t activeIdx; | currently displayed entry |
| laBool playing; | indicates that the widget is automatically cycling |
| uint32_t time; | current cycle time |
| laBool repeat; | indicates that the sequence should repeat when it reaches the end of the sequence |
| laImageSequenceImageChangedEvent_FnPtr cb; | callback when the image changes |
| GFXU_ExternalAssetReader* reader; | asset reader pointer |

### Description

Enumeration: laImageSequenceWidget_t

An image sequence widget is similar to an image widget with the additional capability of showing a sequence of images and automating the transition between them.

This widget is dependent on the time value provided to laUpdate. If laUpdate is not provided with time information this widget will not be able to automatically cycle.

### Remarks

None.

## laImageWidget Structure

Image widget struct definition

### File

libaria_widget_image.h

### C

```
typedef struct laImageWidget_t {
  laWidget widget;
  laHAlignment halign;
  laVAlignment valign;
  GFXU_ImageAsset* image;
  GFXU_ExternalAssetReader* reader;
  laImageWidget_DrawEventCallback ImageDrawStart;
  laImageWidget_DrawEventCallback ImageDrawEnd;
} laImageWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laHAlignment halign; | image horizontal alignment |
| laVAlignment valign; | image vertical alignment |
| GFXU_ImageAsset* image; | pointer to image asset |
| GFXU_ExternalAssetReader* reader; | asset reader |

### Description

Enumeration: laImageWidget_t

An image widget displays an image asset.

### Remarks

None.

## laInput_TouchDownEvent Structure

Register and handle the touch press detect event

### File

libaria_input.h

### C

```
typedef struct laInput_TouchDownEvent_t {
  int32_t touchID;
  int32_t x;
  int32_t y;
} laInput_TouchDownEvent;
```

### Description

Structure: laInput_TouchDownEvent_t

Register and handle the touch press detect event

### Remarks

None.

## laInput_TouchMovedEvent Structure

Register and handle the touch coordinates changed event

### File

libaria_input.h

### C

```
typedef struct laInput_TouchMovedEvent_t {
  int32_t touchID;
  int32_t prevX;
  int32_t prevY;
  int32_t x;
  int32_t y;
} laInput_TouchMovedEvent;
```

### Description

Structure: laInput_TouchMovedEvent_t

Register and handle the touch coordinates changed event

### Remarks

None.

## laInput_TouchUpEvent Structure

Register and handle the touch release detect event

### File

libaria_input.h

### C

```
typedef struct laInput_TouchUpEvent_t {
  int32_t touchID;
  int32_t x;
  int32_t y;
} laInput_TouchUpEvent;
```

### Description

Structure: laInput_TouchUpEvent_t

Register and handle the touch release detect event

### Remarks

None.

## laInputState Structure

Maintain a history of touch states; currently libaria keeps track of the last touch state only.

### File

libaria_input.h

### C

```
typedef struct laInputState_t {
  laBool enabled;
  laTouchState touch[LA_MAX_TOUCH_STATES];
} laInputState;
```

### Description

Structure: laInputState_t

Maintain a history of touch states; currently libaria keeps track of the last touch state only.

### Remarks

None.

## laKey Enumeration

All values possible for key entry from the libaria keyboard widget

### File

libaria_input.h

### C

```
typedef enum laKey_t {
  KEY_NULL = 0,
  KEY_ESCAPE,
  KEY_F1,
  KEY_F2,
  KEY_F3,
  KEY_F4,
```

```
    KEY_F5,
    KEY_F6,
    KEY_F7,
    KEY_F8,
    KEY_F9,
    KEY_F10,
    KEY_F11,
    KEY_F12,
    KEY_PRINTSCREEN,
    KEY_SCROLLLOCK,
    KEY_PAUSE,
    KEY_1,
    KEY_2,
    KEY_3,
    KEY_4,
    KEY_5,
    KEY_6,
    KEY_7,
    KEY_8,
    KEY_9,
    KEY_0,
    KEY_BACKQUOTE,
    KEY_TAB,
    KEY_CAPSLOCK,
    KEY_BRACKET_LEFT,
    KEY_BRACKET_RIGHT,
    KEY_SLASH,
    KEY_SEMICOLON,
    KEY_QUOTE,
    KEY_BACKSLASH,
    KEY_EQUALS,
    KEY_BACKSPACE,
    KEY_MINUS,
    KEY_COMMA,
    KEY_ENTER,
    KEY_PERIOD,
    KEY_A,
    KEY_B,
    KEY_C,
    KEY_D,
    KEY_E,
    KEY_F,
    KEY_G,
    KEY_H,
    KEY_I,
    KEY_J,
    KEY_K,
    KEY_L,
    KEY_M,
    KEY_N,
    KEY_O,
    KEY_P,
    KEY_Q,
    KEY_R,
    KEY_S,
    KEY_T,
    KEY_U,
    KEY_V,
    KEY_W,
    KEY_X,
    KEY_Y,
    KEY_Z,
    KEY_SPACE,
    KEY_LCTRL,
    KEY_RCTRL,
    KEY_LSHIFT,
    KEY_RSHIFT,
    KEY_LALT,
    KEY_RALT,
    KEY_LMETA,
    KEY_RMETA,
    KEY_INSERT,
    KEY_HOME,
    KEY_PAGEUP,
    KEY_END,
```

```
  KEY_PAGEDOWN,
  KEY_RIGHT,
  KEY_LEFT,
  KEY_DOWN,
  KEY_UP,
  KEY_NUMLOCK,
  KEY_KP_DIVIDE,
  KEY_KP_MULTIPLY,
  KEY_KP_MINUS,
  KEY_KP_PLUS,
  KEY_KP_ENTER,
  KEY_KP_1,
  KEY_KP_2,
  KEY_KP_3,
  KEY_KP_4,
  KEY_KP_5,
  KEY_KP_6,
  KEY_KP_7,
  KEY_KP_8,
  KEY_KP_9,
  KEY_KP_0,
  KEY_KP_PERIOD,
  KEY_LAST = KEY_KP_PERIOD
} laKey;
```

## Description

Enumeration: laKey

All values possible for key entry from the libaria keyboard widget

## Remarks

None.

### laKeyPadCell Structure

Defines a key pad cell struct

## File

libaria_widget_keypad.h

## C

```
typedef struct laKeyPadCell_t {
  laBool enabled;
  laButtonWidget* button;
  laKeyPadCellAction action;
  laString value;
} laKeyPadCell;
```

## Members

| Members | Description |
|---|---|
| laBool enabled; | indicates if the cell should be drawn |
| laButtonWidget* button; | the button that handles the cell input events and rendering |
| laKeyPadCellAction action; | the action that occurs when the cell is activated |
| laString value; | the value that is passed to the edit event system |

## Description

Structure: laKeyPadCell_t

A key pad is made up of an array of key pad cells. Each cell is individually an laButtonWidget, an action, a value, and a few other options.

## Remarks

None.

### laKeyPadCellAction Enumeration

Defines an assigned action to a key pad cell

### File

libaria_widget_keypad.h

### C

```
typedef enum laKeyPadCellAction_t {
  LA_KEYPAD_CELL_ACTION_NONE,
  LA_KEYPAD_CELL_ACTION_APPEND,
  LA_KEYPAD_CELL_ACTION_SET,
  LA_KEYPAD_CELL_ACTION_BACKSPACE,
  LA_KEYPAD_CELL_ACTION_CLEAR,
  LA_KEYPAD_CELL_ACTION_ACCEPT
} laKeyPadCellAction;
```

### Description

Structure: laKeyPadCellAction_t

### Remarks

None.

## laKeyPadWidget Structure

Defines a key pad widget struct

### File

libaria_widget_keypad.h

### C

```
typedef struct laKeyPadWidget_t {
  laWidget widget;
  uint32_t rows;
  uint32_t cols;
  laKeyPadActionTrigger trigger;
  laKeyPadCell* cells;
  laKeyPadWidget_KeyClickEvent clickEvt;
  GFXU_ExternalAssetReader* reader;
} laKeyPadWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| uint32_t rows; | number of button rows |
| uint32_t cols; | number of button columns |
| laKeyPadActionTrigger trigger; | trigger for action and events |
| laKeyPadCell* cells; | key cell array |
| laKeyPadWidget_KeyClickEvent clickEvt; | key click callback event |
| GFXU_ExternalAssetReader* reader; | asset reader |

### Description

Structure: laKeyPadCell_t

A key pad is a widget that is comprised of an array of laButtonWidgets. This widget serves to issue edit events based on application or input interaction. Receptor edit widgets can then receive these edit events and react accordingly.

### Remarks

None.

## laKeyPadWidget_KeyClickEvent Type

### File

libaria_widget_keypad.h

### C

```
typedef void (* laKeyPadWidget_KeyClickEvent)(laKeyPadWidget*, laButtonWidget*, uint32_t, uint32_t);
```

## Description

This is type laKeyPadWidget_KeyClickEvent.

### laLabelWidget Structure

Implementation of a label widget struct

## File

libaria_widget_label.h

## C

```
typedef struct laLabelWidget_t {
  laWidget widget;
  laString text;
  laHAlignment halign;
  laVAlignment valign;
  GFXU_ExternalAssetReader* reader;
} laLabelWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laString text; | string to draw |
| laHAlignment halign; | horizontal alignment of string |
| laVAlignment valign; | vertical alignment of string |
| GFXU_ExternalAssetReader* reader; | asset reader |

## Description

Structure: laLabelWidget_t

A label widget is a simple widget that draws a string of text.

## Remarks

None.

### laLayer Type

Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent.

## File

libaria_utils.h

## C

```
typedef struct laLayer_t laLayer;
```

## Description

Structure: laLayer_t

## Remarks

None.

### laLayerBuffer Structure

Structure to maintain the buffer type and track the buffer location for each layer

## File

libaria_layer.h

## C

```
typedef struct laLayerBuffer_t {
  laLayerBufferType type;
  void* address;
} laLayerBuffer;
```

## Description

Structure: laLayerBuffer_t

Structure to maintain the buffer type and track the buffer location for each layer

## Remarks

None.

### laLayerBufferType Enumeration

Defines the type of a layer. If the layer has an explicit address then Aria tries to set that through the HAL when the layer is being set up.

## File

libaria_layer.h

## C

```
typedef enum laLayerBufferType_t {
  LA_BUFFER_TYPE_AUTO,
  LA_BUFFER_TYPE_ADDRESS
} laLayerBufferType;
```

## Description

Enumeration: laLayerBufferType_t

## Remarks

None.

### laLineWidget Structure

Defines the implementation of a line widget struct

## File

libaria_widget_line.h

## C

```
typedef struct laLineWidget_t {
  laWidget widget;
  int32_t x1;
  int32_t y1;
  int32_t x2;
  int32_t y2;
} laLineWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| int32_t x1; | point 1 x |
| int32_t y1; | point 1 y |
| int32_t x2; | point 2 x |
| int32_t y2; | point 2 y |

## Description

Structure: laLineWidget_t

A line widget draws a simple line shape within the confines of its bounding rectangle. All coordinates are expressed in local widget space.

The color of the line is determined by the widget scheme's 'foreground' color.

## Remarks

None.

### laListItem Structure

Defines a list item struct

### File

libaria_widget_list.h

### C

```
typedef struct laListItem_t {
  laString string;
  GFXU_ImageAsset* icon;
  laBool selected;
  GFX_Rect rowRect;
} laListItem;
```

### Members

| Members | Description |
|---|---|
| laString string; | list item string |
| GFXU_ImageAsset* icon; | list item icon |
| laBool selected; | list item selected flag |
| GFX_Rect rowRect; | list item row rectangle |

### Description

Structure: laListItem_t

### Remarks

None.

## laListWheelItem Structure

Implementation of a list wheel widget item struct

### File

libaria_widget_listwheel.h

### C

```
typedef struct laListWheelItem_t {
  laString string;
  GFXU_ImageAsset* icon;
} laListWheelItem;
```

### Description

Structure: laListWheelItem_t

A list wheel item contains either a text string, an icon, or both

### Remarks

None.

## laListWheelWidget Structure

Implementation of a list wheel widget struct

### File

libaria_widget_listwheel.h

### C

```
typedef struct laListWheelWidget_t {
  laWidget widget;
  laArray items;
  int32_t selectedItem;
  int32_t visibleItems;
  int32_t topItem;
  laHAlignment halign;
  laRelativePosition iconPos;
  uint32_t iconMargin;
  laBool showIndicators;
  uint32_t indicatorArea;
  uint32_t shaded;
```

```
  int32_t cycleDistance;
  int32_t cycleDelta;
  int32_t firstTouchY;
  int32_t touchY;
  int32_t lastTouchY;
  laBool stillTouching;
  int32_t minFlickDelta;
  int32_t momentum;
  int32_t maxMomentum;
  int32_t momentumFalloff;
  int32_t rotation;
  int32_t rotationCounter;
  int32_t rotationTick;
  laBool snapPending;
  laListWheelIndicatorFill indicatorFill;
  laListWheelZoomEffects zoomEffects;
  laBool autoHideWheel;
  laBool hideWheel;
  struct {
    int32_t y;
    int32_t per;
    uint32_t nextItem;
  } paintState;
  laListWheelWidget_SelectedItemChangedEvent cb;
  laBorderType borderTypeCache;
  laBackgroundType backgroundTypeCache;
  GFXU_ExternalAssetReader* reader;
} laListWheelWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laArray items; | list of items for the wheel |
| int32_t selectedItem; | currently selected item |
| int32_t visibleItems; | number of visible items in the wheel must be odd and >= 3 |
| int32_t topItem; | the current top item |
| laHAlignment halign; | the horizontal alignment of the items |
| laRelativePosition iconPos; | the icon position of the items |
| uint32_t iconMargin; | the icon margin of the items |
| laBool showIndicators; | controls the visibility of the horizontal indicator bars in the center of the widget |
| uint32_t indicatorArea; | controls the distance between the indicator bars |
| uint32_t shaded; | determins if the background of the widget uses gradient shading to show depth |
| int32_t cycleDistance; | determins the amount of drag distance needed to cycle between items |
| int32_t cycleDelta; | tracks the current amount of drag distance |
| int32_t firstTouchY; | these track drag movement over time |
| int32_t minFlickDelta; | amount of distance that must be dragged in a single frame to trigger momentum mode |
| int32_t momentum; | current momentum value |
| int32_t maxMomentum; | maximum momentum value |
| int32_t momentumFalloff; | momentum falloff rate |
| int32_t rotation; | determines actual rotation of the wheel |
| int32_t rotationCounter; | time-based limiter for rotation calculations |
| int32_t rotationTick; | rotation time accumulator |
| laListWheelIndicatorFill indicatorFill; | the indicator's fill type |
| laListWheelZoomEffects zoomEffects; | zoomEffects |
| laBool autoHideWheel; | auto hides the wheel |
| laBool hideWheel; | flag to hide/show the wheel |
| laListWheelWidget_SelectedItemChangedEvent cb; | item changed callback |
| laBorderType borderTypeCache; | Copy of border type, used to restore borders on auto-hide |
| laBackgroundType backgroundTypeCache; | Copy of background type, used to restore borders on auto-hide |
| GFXU_ExternalAssetReader* reader; | asset reader |

## Description

Structure: laListWheelWidget_t

A list wheel widget is a widget that is similar to a normal list widget but can be dragged up or down to cycle through a single active value. This widget is also capable of momentum and motion over time.

## Remarks

None.

### laListWheelWidget_SelectedItemChangedEvent Type

## File

libaria_widget_listwheel.h

## C

```
typedef void (* laListWheelWidget_SelectedItemChangedEvent)(laListWheelWidget*, uint32_t idx);
```

## Description

This is type laListWheelWidget_SelectedItemChangedEvent.

### laListWidget Structure

Defines the implementation of a list widget

## File

libaria_widget_list.h

## C

```
typedef struct laListWidget_t {
  laWidget widget;
  laListWidget_SelectionMode mode;
  laBool allowEmpty;
  laArray items;
  laHAlignment halign;
  laRelativePosition iconPos;
  uint32_t iconMargin;
  uint32_t itemDown;
  laScrollBarWidget* scrollbar;
  struct {
    laListItem* item;
    GFX_Rect itemRect;
    int32_t y;
    uint32_t nextItem;
  } paintState;
  laListWidget_ItemSelectedChangedEvent cb;
  GFXU_ExternalAssetReader* reader;
} laListWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | list base class |
| laListWidget_SelectionMode mode; | list selection mode |
| laBool allowEmpty; | indicates if the list must always have at least one selected item |
| laArray items; | list containing the list items |
| laHAlignment halign; | horizontal alignment of the list |
| laRelativePosition iconPos; | icon position for the list icons |
| uint32_t iconMargin; | margin for the list icons |
| uint32_t itemDown; | tracks whether an input event is in process |
| laScrollBarWidget* scrollbar; | internal scrollbar for this widget |
| laListWidget_ItemSelectedChangedEvent cb; | item selected changed event |

## Description

Structure: laListWidget_t

A list widget is a widget that contains a series of vertical nodes. Each node can have text, an image, or both, and can be selected or not. The list has a built-in scrollbar. This allows the list to be larger than the visible area of the widget.

### Remarks

None.

## laListWidget_ItemSelectedChangedEvent Type

### File

libaria_widget_list.h

### C

```
typedef void (* laListWidget_ItemSelectedChangedEvent)(laListWidget*, uint32_t idx, laBool selected);
```

### Description

This is type laListWidget_ItemSelectedChangedEvent.

## laListWidget_SelectedItemChangedEvent Type

### File

libaria_widget_list.h

### C

```
typedef void (* laListWidget_SelectedItemChangedEvent)(laListWidget*, uint32_t idx, laBool selected);
```

### Description

This is type laListWidget_SelectedItemChangedEvent.

## laListWidget_SelectionMode Enumeration

Defines the list selection modes

### File

libaria_widget_list.h

### C

```
typedef enum laListWidget_SelectionMode_t {
  LA_LIST_WIDGET_SELECTION_MODE_SINGLE,
  LA_LIST_WIDGET_SELECTION_MODE_MULTIPLE,
  LA_LIST_WIDGET_SELECTION_MODE_CONTIGUOUS
} laListWidget_SelectionMode;
```

### Description

Enumeration: laListWidget_SelectionMode_t

Single - a single selection from the list is allowed at any one time Multiple - any number of selected items is allowed at any one time Contiguous - any number of selected items in a contiguous serious is allowed at any one time

### Remarks

None.

## laMouseButton Enumeration

All values possible for mouse key entry from the libaria mouse input

### File

libaria_input.h

### C

```
typedef enum laMouseButton_t {
  BUTTON_NONE = 0,
  BUTTON_LEFT,
  BUTTON_MIDDLE,
  BUTTON_RIGHT,
```

```
  BUTTON_WHEEL_UP,
  BUTTON_WHEEL_DOWN,
  BUTTON_LAST = BUTTON_WHEEL_DOWN
} laMouseButton;
```

## Description

Enumeration: laMouseButton

All values possible for mouse key entry from the libaria mouse input

## Remarks

None.

### laProgressBar Type

## File

libaria_widget_progressbar.h

## C

```
typedef struct laProgressBarWidget_t laProgressBar;
```

## Description

This is type laProgressBar.

### laProgressBar_ValueChangedEventCallback Type

## File

libaria_widget_progressbar.h

## C

```
typedef void (* laProgressBar_ValueChangedEventCallback)(laProgressBar*, uint32_t);
```

## Description

This is type laProgressBar_ValueChangedEventCallback.

### laProgressBarDirection Enumeration

Defines the valid values for the progress bar widget fill directions.

## File

libaria_widget_progressbar.h

## C

```
typedef enum laProgressBarDirection_t {
  LA_PROGRESSBAR_DIRECTION_RIGHT,
  LA_PROGRESSBAR_DIRECTION_UP,
  LA_PROGRESSBAR_DIRECTION_LEFT,
  LA_PROGRESSBAR_DIRECTION_DOWN
} laProgressBarDirection;
```

## Description

Enumeration: laProgressBarDirection_t

## Remarks

None.

### laProgressBarWidget Structure

Implementation of a progressbar widget struct

## File

libaria_widget_progressbar.h

## C

```
typedef struct laProgressBarWidget_t {
  laWidget widget;
  laProgressBarDirection direction;
  uint32_t value;
  laProgressBar_ValueChangedEventCallback cb;
} laProgressBarWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | base widget class |
| laProgressBarDirection direction; | the fill direction of the bar |
| uint32_t value; | fill percentage |
| laProgressBar_ValueChangedEventCallback cb; | value changed callback |

## Description

Structure: laProgressBarDirection_t

A progress bar widget is a widget that can fill itself with a color based on a given percentage from 0-100. This is often used to visually illustrate the progress of some other activity over time.

## Remarks

None.

### laRadioButtonGroup Type

Defines the structure used for the Radio Button group.

## File

libaria_widget_radiobutton.h

## C

```
typedef struct laRadioButtonGroup_t laRadioButtonGroup;
```

## Description

Structure laRadioButtonGroup_t

Defines the parameters required for a Radio Button group. Marks the current selected Radio button within the group

## Remarks

None.

### laRadioButtonWidget Structure

Implementation of a radio button widget struct

## File

libaria_widget_radiobutton.h

## C

```
typedef struct laRadioButtonWidget_t {
  laWidget widget;
  laBool selected;
  laString text;
  laHAlignment halign;
  laVAlignment valign;
  GFXU_ImageAsset* selectedImage;
  GFXU_ImageAsset* unselectedImage;
  laRelativePosition imagePosition;
  uint32_t imageMargin;
  uint32_t circleButtonSize;
  laRadioButtonWidget_SelectedEvent selectedEvent;
  laRadioButtonWidget_DeselectedEvent deselectedEvent;
  struct {
    uint8_t enabled;
  } paintData;
```

```
  GFXU_ExternalAssetReader* reader;
  laRadioButtonGroup* group;
} laRadioButtonWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laBool selected; | indicates if the radio button is selected |
| laString text; | radio button text |
| laHAlignment halign; | horizontal alignment |
| laVAlignment valign; | vertical alignment |
| GFXU_ImageAsset* selectedImage; | button custom selected image |
| GFXU_ImageAsset* unselectedImage; | buton custom unselected image |
| laRelativePosition imagePosition; | image icon relative position |
| uint32_t imageMargin; | image margin |
| uint32_t circleButtonSize; | size of radio circle button in pixels |
| laRadioButtonWidget_SelectedEvent selectedEvent; | button selected event callback |
| laRadioButtonWidget_DeselectedEvent deselectedEvent; | button deselected event callback |
| GFXU_ExternalAssetReader* reader; | asset reader |
| laRadioButtonGroup* group; | radio button group |

## Description

Enumeration: laRadioButtonWidget_t

A radio button is similar to a checkbox widget in that it has an on and off state. It is further capable of being added to a radio button group. This group provides a mutually exclusive selection capability so that only one radio button may be selected at any one time.

## Remarks

None.

### laRadioButtonWidget_DeselectedEvent Type

## File

libaria_widget_radiobutton.h

## C

```
typedef void (* laRadioButtonWidget_DeselectedEvent)(laRadioButtonWidget*);
```

## Description

This is type laRadioButtonWidget_DeselectedEvent.

### laRadioButtonWidget_SelectedEvent Type

## File

libaria_widget_radiobutton.h

## C

```
typedef void (* laRadioButtonWidget_SelectedEvent)(laRadioButtonWidget*);
```

## Description

This is type laRadioButtonWidget_SelectedEvent.

### laRectangleWidget Structure

Implementation of a rectangle widget struct

## File

libaria_widget_rectangle.h

### C

```
typedef struct laRectangleWidget_t {
  laWidget widget;
  int32_t thickness;
} laRectangleWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| int32_t thickness; | rectangle border thickness |

### Description

Enumeration: laRectangleWidget_t

A rectangle widget draws a basic rectangle of a given thickness using the widget's bounding box as the dimensions.

### Remarks

None.

## laScheme Structure

This structure specifies the style scheme components of an object.

### File

libaria_scheme.h

### C

```
typedef struct laScheme_t {
  GFX_Color base;
  GFX_Color highlight;
  GFX_Color highlightLight;
  GFX_Color shadow;
  GFX_Color shadowDark;
  GFX_Color foreground;
  GFX_Color foregroundInactive;
  GFX_Color foregroundDisabled;
  GFX_Color background;
  GFX_Color backgroundInactive;
  GFX_Color backgroundDisabled;
  GFX_Color text;
  GFX_Color textHighlight;
  GFX_Color textHighlightText;
  GFX_Color textInactive;
  GFX_Color textDisabled;
} laScheme;
```

### Description

Enumeration: laScheme_t

A scheme is a collection of colors that can be referenced by widgets or other objects. While the color names strive to be intuitive they aren't always used in the manner in which they describe.

### Remarks

None.

## laScreen_CreateCallback_FnPtr Type

Callback pointer for a new screen create event notification. This is called when the library attempts to create a screen.

### File

libaria_screen.h

### C

```
typedef void (* laScreen_CreateCallback_FnPtr)(laScreen*);
```

## Description

Type: laScreen_CreateCallback_FnPtr

### laScreen_ShowHideCallback_FnPtr Type

Callback pointer for the active screen show or hide event change notification.

## File

libaria_screen.h

## C

```
typedef void (* laScreen_ShowHideCallback_FnPtr)(laScreen*);
```

## Description

Type: laScreen_ShowHideCallback_FnPtr

Callback pointer for the active screen show or hide event change notification.

### laScreenOrientation Enumeration

Possible values for screen orientation.

## File

libaria_screen.h

## C

```
typedef enum laScreenOrientation_t {
  LA_SCREEN_ORIENTATION_0 = 0x0,
  LA_SCREEN_ORIENTATION_90,
  LA_SCREEN_ORIENTATION_180,
  LA_SCREEN_ORIENTATION_270
} laScreenOrientation;
```

## Description

Enumeration: laScreenOrientation_t

Possible values for screen orientation.

## Remarks

None.

### laScrollBarOrientation Enumeration

Defines the scroll bar direction values

## File

libaria_widget_scrollbar.h

## C

```
typedef enum laScrollBarOrientation_t {
  LA_SCROLLBAR_ORIENT_VERTICAL,
  LA_SCROLLBAR_ORIENT_HORIZONTAL
} laScrollBarOrientation;
```

## Description

Enumeration: laScrollBarOrientation_t

## Remarks

None.

### laScrollBarState Enumeration

Defines the various scroll bar state values

### File

libaria_widget_scrollbar.h

### C

```
typedef enum laScrollBarState_t {
  LA_SCROLLBAR_STATE_NONE,
  LA_SCROLLBAR_STATE_TOP_PRESSED,
  LA_SCROLLBAR_STATE_TOP_INSIDE,
  LA_SCROLLBAR_STATE_BOTTOM_PRESSED,
  LA_SCROLLBAR_STATE_BOTTOM_INSIDE,
  LA_SCROLLBAR_STATE_HANDLE_DOWN
} laScrollBarState;
```

### Description

Enumeration: laScrollBarState_t

### Remarks

None.

## laScrollBarWidget Structure

Implementation of a scroll bar widget.

### File

libaria_widget_scrollbar.h

### C

```
typedef struct laScrollBarWidget_t {
  laWidget widget;
  laScrollBarState state;
  laScrollBarOrientation alignment;
  uint32_t max;
  uint32_t extent;
  uint32_t value;
  uint32_t step;
  laScrollBarWidget_ValueChangedEvent valueChangedEvent;
  GFX_Point handleDownOffset;
} laScrollBarWidget;
```

### Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laScrollBarState state; | scrollbar input state |
| laScrollBarOrientation alignment; | scroll bar direction |
| uint32_t max; | maximum scroll value |
| uint32_t extent; | visible space/handle size |
| uint32_t value; | current scroll value |
| uint32_t step; | discreet scroll stepping value |
| laScrollBarWidget_ValueChangedEvent valueChangedEvent; | value changed callback |

### Description

Structure: laScrollBarWidget_t

A scroll bar is a widget that is capable of displaying a range and a scroll handle. The handle can grow and shrink in size depending on the scroll range and visible scroll space and can be interacted with to scroll through the available space.

### Remarks

None.

## laScrollBarWidget_ValueChangedEvent Type

### File

libaria_widget_scrollbar.h

### C

```
typedef void (* laScrollBarWidget_ValueChangedEvent)(laScrollBarWidget*);
```

### Description

This is type laScrollBarWidget_ValueChangedEvent.

## laSliderOrientation Enumeration

Slider orientations

### File

libaria_widget_slider.h

### C

```
typedef enum laSliderOrientation_t {
  LA_SLIDER_ORIENT_VERTICAL,
  LA_SLIDER_ORIENT_HORIZONTAL
} laSliderOrientation;
```

### Description

Enumeration: laSliderOrientation_t

### Remarks

None.

## laSliderState Enumeration

Describes various slider states

### File

libaria_widget_slider.h

### C

```
typedef enum laSliderState_t {
  LA_SLIDER_STATE_NONE,
  LA_SLIDER_STATE_HANDLE_DOWN,
  LA_SLIDER_STATE_AREA_DOWN
} laSliderState;
```

### Description

Enumeration: laSliderState_t

### Remarks

None.

## laSliderWidget Structure

Implementation of a slider widget struct

### File

libaria_widget_slider.h

### C

```
typedef struct laSliderWidget_t {
  laWidget widget;
  laSliderState state;
  laSliderOrientation alignment;
  int32_t min;
```

```
  int32_t max;
  int32_t value;
  uint32_t grip;
  laSliderWidget_ValueChangedEvent valueChangedEvent;
  GFX_Point handleDownOffset;
} laSliderWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laSliderState state; | slider state |
| laSliderOrientation alignment; | slider alignment |
| int32_t min; | slider min value |
| int32_t max; | slider max value |
| int32_t value; | slider current value |
| uint32_t grip; | slider grip size |
| laSliderWidget_ValueChangedEvent valueChangedEvent; | value changed event |

## Description

Structure: laSliderWidget_t

A slider bar is a widget that is capable of displaying a range and a slider handle. The slider can be moved between two discreet values and can have a variable min and max range.

## Remarks

None.

### laSliderWidget_ValueChangedEvent Type

## File

libaria_widget_slider.h

## C

```
typedef void (* laSliderWidget_ValueChangedEvent)(laSliderWidget*);
```

## Description

This is type laSliderWidget_ValueChangedEvent.

### laTextFieldWidget Structure

Implementation of a text field widget.

## File

libaria_widget_textfield.h

## C

```
typedef struct laTextFieldWidget_t {
  laEditWidget editWidget;
  laString text;
  laHAlignment halign;
  uint32_t cursorPos;
  uint32_t cursorDelay;
  uint32_t cursorTime;
  laBool cursorEnable;
  laBool cursorVisible;
  laBool clearOnFirstEdit;
  laTextFieldWidget_TextChangedCallback textChangedEvent;
  GFXU_ExternalAssetReader* reader;
} laTextFieldWidget;
```

## Members

| Members | Description |
|---|---|
| laEditWidget editWidget; | edit widget base class |

| | |
|---|---|
| laString text; | the text to edit |
| laHAlignment halign; | horizontal alignment |
| uint32_t cursorPos; | current cursor position |
| uint32_t cursorDelay; | cursor blink delay |
| uint32_t cursorTime; | current cursor tick counter |
| laBool cursorEnable; | cursor enabled flag |
| laBool cursorVisible; | cursor visbility flag |
| laBool clearOnFirstEdit; | needs clear on first edit |
| laTextFieldWidget_TextChangedCallback textChangedEvent; | text changed event |
| GFXU_ExternalAssetReader* reader; | asset reader |

## Description

Enumeration: laTextFieldWidget_t

A text field widget is a widget that is capable of displaying a single line of editable text. This widget is capable of receiving edit events from the Aria edit event system. It can also display a blinking cursor.

## Remarks

None.

### laTextFieldWidget_TextChangedCallback Type

## File

libaria_widget_textfield.h

## C

```
typedef void (* laTextFieldWidget_TextChangedCallback)(laTextFieldWidget*);
```

## Description

This is type laTextFieldWidget_TextChangedCallback.

### laTouchState Structure

Manage the touch input state and track the touch coordinate

## File

libaria_input.h

## C

```
typedef struct laTouchState_t {
  uint32_t valid;
  int32_t x;
  int32_t y;
} laTouchState;
```

## Description

Structure: laTouchState

Manage the touch input state and track the touch coordinate

## Remarks

None.

### laTouchTestState Enumeration

Touch test states

## File

libaria_widget_touchtest.h

## C

```
typedef enum laTouchTestState_t {
```

```
    LA_TOUCHTEST_STATE_UP,
    LA_TOUCHTEST_STATE_DOWN
} laTouchTestState;
```

## Description

Enumeration: laTouchTestState_t

## Remarks

None.

### laTouchTestWidget Structure

Implementation of a touch test widget struct

## File

libaria_widget_touchtest.h

## C

```
typedef struct laTouchTestWidget_t {
  laWidget widget;
  laTouchTestState state;
  GFX_Point pnts[LA_TOUCHTEST_MEMORY_SIZE];
  uint32_t size;
  uint32_t start;
  uint32_t next;
  laTouchTestWidget_PointAddedEventCallback cb;
} laTouchTestWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | widget base class |
| laTouchTestState state; | touch test state |
| GFX_Point pnts[LA_TOUCHTEST_MEMORY_SIZE]; | touch point array |
| uint32_t size; | current number of valid touch points |
| uint32_t start; | first valid touch point |
| uint32_t next; | next available touch point entry |
| laTouchTestWidget_PointAddedEventCallback cb; | point added callback |

## Description

Structure: laTouchTestWidget_t

The touch test widget is a specialized widget that displays intersecting lines based on input events. This can help visualize touch interaction and aid determining accurate input coordinates.

## Remarks

None.

### laTouchTestWidget_PointAddedEventCallback Type

## File

libaria_widget_touchtest.h

## C

```
typedef void (* laTouchTestWidget_PointAddedEventCallback)(laTouchTestWidget*, GFX_Point*);
```

## Description

This is type laTouchTestWidget_PointAddedEventCallback.

### laWidget Structure

Specifies Graphics widget structure to manage all properties and events associated with the widget

## File

libaria_widget.h

## C

```
typedef struct laWidget_t {
  uint32_t id;
  laWidgetType type;
  laBool editable;
  laBool visible;
  laBool enabled;
  GFX_Rect rect;
  uint32_t cornerRadius;
  laMargin margin;
  laBorderType borderType;
  laBackgroundType backgroundType;
  uint32_t optimizationFlags;
  uint32_t drawCount;
  GFX_PixelBuffer* cache;
  laBool cacheInvalid;
  laBool alphaEnabled;
  uint32_t alphaAmount;
  uint32_t dirtyState;
  uint32_t drawState;
  laWidget_DrawFunction_FnPtr drawFunc;
  laScheme* scheme;
  laBool root;
  laWidget* parent;
  laArray children;
  laWidget_Destructor_FnPtr destructor;
  laWidget_Moved_FnPtr moved;
  laWidget_Resized_FnPtr resized;
  laWidget_Focus_FnPtr focusGained;
  laWidget_Focus_FnPtr focusLost;
  laWidget_Update_FnPtr update;
  laWidget_Paint_FnPtr paint;
  laWidget_TouchDownEvent_FnPtr touchDown;
  laWidget_TouchUpEvent_FnPtr touchUp;
  laWidget_TouchMovedEvent_FnPtr touchMoved;
  laWidget_LanguageChangingEvent_FnPtr languageChangeEvent;
  laWidget_InvalidateBorderAreas_FnPtr invalidateBorderAreas;
} laWidget;
```

## Members

| Members | Description |
|---|---|
| uint32_t id; | the id of the widget |
| laWidgetType type; | the type of the widget |
| laBool editable; | indicates if this widget implements the editable interface |
| laBool visible; | the widget visible flag |
| laBool enabled; | the widget enabled flag |
| GFX_Rect rect; | the bounding rectangle of the widget |
| uint32_t cornerRadius; | corner radius, draws round corners if > 0 |
| laMargin margin; | the margin settings for the widget |
| laBorderType borderType; | the widget border type |
| laBackgroundType backgroundType; | the widget background type |
| uint32_t optimizationFlags; | optimization flags |
| uint32_t drawCount; | number of times this widget has been drawn for the active screen |
| GFX_PixelBuffer* cache; | the local framebuffer cache for the widget this can be used to avoid costly parent redraw operations at the cost of using more memory |
| laBool cacheInvalid; | indicates that the local cache is invalid and needs to be refilled |
| laBool alphaEnabled; | indicates that the global alpha blending setting is enabled for this widget |
| uint32_t alphaAmount; | the global alpha amount to apply to this widget (cumulative with parent widgets) |
| uint32_t dirtyState; | the widget's dirty state |
| uint32_t drawState; | the widget's draw state |
| laWidget_DrawFunction_FnPtr drawFunc; | the next draw function to call |
| laScheme* scheme; | the widget's color scheme |

| | |
|---|---|
| laBool root; | indicates if this widget is a root widget |
| laWidget* parent; | pointer to the widget's parent |
| laArray children; | pointers for the widget's children |
| laWidget_Destructor_FnPtr destructor; | the widget's destructor |
| laWidget_Moved_FnPtr moved; | moved function pointer |
| laWidget_Resized_FnPtr resized; | resized function pointer |
| laWidget_Focus_FnPtr focusGained; | focus gained function pointer |
| laWidget_Focus_FnPtr focusLost; | focus lost function pointer |
| laWidget_Update_FnPtr update; | update function pointer |
| laWidget_Paint_FnPtr paint; | paint function pointer |
| laWidget_TouchDownEvent_FnPtr touchDown; | touch down function pointer |
| laWidget_TouchUpEvent_FnPtr touchUp; | touch up function pointer |
| laWidget_TouchMovedEvent_FnPtr touchMoved; | touch moved function pointer |
| laWidget_LanguageChangingEvent_FnPtr languageChangeEvent; | languaged event pointer |

## Description

Structure: laWidget_t

Specifies Graphics widget structure to manage all properties and events associated with the widget. It also contains information about the parent and children for the widget to manage the tree structure that the library supports.

## Remarks

None.

### laWidget_Constructor_FnPtr Type

## File

libaria_widget.h

## C

```
typedef void (* laWidget_Constructor_FnPtr)(laWidget*);
```

## Description

This is type laWidget_Constructor_FnPtr.

### laWidget_Destructor_FnPtr Type

## File

libaria_widget.h

## C

```
typedef void (* laWidget_Destructor_FnPtr)(laWidget*);
```

## Description

This is type laWidget_Destructor_FnPtr.

### laWidget_DrawFunction_FnPtr Type

## File

libaria_widget.h

## C

```
typedef void (* laWidget_DrawFunction_FnPtr)(void*);
```

## Description

This is type laWidget_DrawFunction_FnPtr.

### laWidget_Focus_FnPtr Type

#### File

libaria_widget.h

#### C

```
typedef void (* laWidget_Focus_FnPtr)(laWidget*);
```

#### Description

This is type laWidget_Focus_FnPtr.

### laWidget_Moved_FnPtr Type

#### File

libaria_widget.h

#### C

```
typedef void (* laWidget_Moved_FnPtr)(laWidget*);
```

#### Description

This is type laWidget_Moved_FnPtr.

### laWidget_Paint_FnPtr Type

#### File

libaria_widget.h

#### C

```
typedef void (* laWidget_Paint_FnPtr)(laWidget*);
```

#### Description

This is type laWidget_Paint_FnPtr.

### laWidget_Resized_FnPtr Type

#### File

libaria_widget.h

#### C

```
typedef void (* laWidget_Resized_FnPtr)(laWidget*);
```

#### Description

This is type laWidget_Resized_FnPtr.

### laWidget_TouchDownEvent_FnPtr Type

#### File

libaria_widget.h

#### C

```
typedef void (* laWidget_TouchDownEvent_FnPtr)(laWidget*, laInput_TouchDownEvent*);
```

#### Description

This is type laWidget_TouchDownEvent_FnPtr.

## laWidget_TouchMovedEvent_FnPtr Type

### File

libaria_widget.h

### C

```
typedef void (* laWidget_TouchMovedEvent_FnPtr)(laWidget*, laInput_TouchMovedEvent*);
```

### Description

This is type laWidget_TouchMovedEvent_FnPtr.

## laWidget_TouchUpEvent_FnPtr Type

### File

libaria_widget.h

### C

```
typedef void (* laWidget_TouchUpEvent_FnPtr)(laWidget*, laInput_TouchUpEvent*);
```

### Description

This is type laWidget_TouchUpEvent_FnPtr.

## laWidget_Update_FnPtr Type

### File

libaria_widget.h

### C

```
typedef laWidgetUpdateState (* laWidget_Update_FnPtr)(laWidget*, uint32_t);
```

### Description

This is type laWidget_Update_FnPtr.

## laWidgetDirtyState Enumeration

Specifies the different dirty states the widget can be assigned

### File

libaria_widget.h

### C

```
typedef enum laWidgetDirtyState_t {
  LA_WIDGET_DIRTY_STATE_CLEAN,
  LA_WIDGET_DIRTY_STATE_CHILD,
  LA_WIDGET_DIRTY_STATE_DIRTY
} laWidgetDirtyState;
```

### Description

Enumeration: laWidgetDirtyState_t

Specifies the different dirty states the widget can be assigned This decides whether the particular widget would be re-drawn or not. Dirty widget are re-drawn and clean are not painted over.

### Remarks

None.

## laWidgetDrawState Enumeration

Specifies the different draw states the widget can be assigned

### File

libaria_widget.h

### C

```
typedef enum laWidgetDrawState_t {
  LA_WIDGET_DRAW_STATE_READY,
  LA_WIDGET_DRAW_STATE_DONE
} laWidgetDrawState;
```

### Description

Enumeration: laWidgetDrawState_t

Specifies the different draw states the widget can be assigned

### Remarks

None.

## laWidgetEvent Structure

Basic widget event definition

### File

libaria_event.h

### C

```
typedef struct laWidgetEvent_t {
  laEventID id;
  laWidget* source;
  laWidget* target;
  laBool accepted;
} laWidgetEvent;
```

### Description

Structure: laWidgetEvent_t

## laWidgetType Enumeration

Specifies the different widget types used in the library

### File

libaria_widget.h

### C

```
typedef enum laWidgetType_t {
  LA_WIDGET_WIDGET,
  LA_WIDGET_LAYER,
  LA_WIDGET_ARC,
  LA_WIDGET_BAR_GRAPH,
  LA_WIDGET_BUTTON,
  LA_WIDGET_CHECKBOX,
  LA_WIDGET_CIRCLE,
  LA_WIDGET_CIRCULAR_GAUGE,
  LA_WIDGET_CIRCULAR_SLIDER,
  LA_WIDGET_DRAWSURFACE,
  LA_WIDGET_IMAGE,
  LA_WIDGET_IMAGEPLUS,
  LA_WIDGET_IMAGESEQUENCE,
  LA_WIDGET_GRADIENT,
  LA_WIDGET_GROUPBOX,
  LA_WIDGET_KEYPAD,
  LA_WIDGET_LABEL,
  LA_WIDGET_LINE,
  LA_WIDGET_LINE_GRAPH,
  LA_WIDGET_LIST,
  LA_WIDGET_LISTWHEEL,
  LA_WIDGET_PIE_CHART,
  LA_WIDGET_PROGRESSBAR,
  LA_WIDGET_RADIAL_MENU,
```

```
  LA_WIDGET_RADIOBUTTON,
  LA_WIDGET_RECTANGLE,
  LA_WIDGET_SCROLLBAR,
  LA_WIDGET_SLIDER,
  LA_WIDGET_TEXTFIELD,
  LA_WIDGET_TOUCHTEST,
  LA_WIDGET_WINDOW
} laWidgetType;
```

## Description

Enumeration: laWidgetType_t

This enumeration specifies the different widget types used in the library.

## Remarks

None.

### laWindowWidget Structure

Implementation of a window widget struct

## File

libaria_widget_window.h

## C

```
typedef struct laWindowWidget_t {
  laWidget widget;
  laString title;
  GFXU_ImageAsset* icon;
  uint32_t iconMargin;
  struct {
    GFX_Rect barRect;
  } paintData;
  GFXU_ExternalAssetReader* reader;
} laWindowWidget;
```

## Members

| Members | Description |
|---|---|
| laWidget widget; | base widget class |
| laString title; | title text |
| GFXU_ImageAsset* icon; | title icon |
| uint32_t iconMargin; | title icon margin |
| GFXU_ExternalAssetReader* reader; | asset reader |

## Description

Structure: laWindowWidget_t

A window widget is an extension of a basic panel. It adds a title bar with text and an icon.

## Remarks

None.

### laBackgroundType Enumeration

Specifies the different background types used for the widgets in the library

## File

libaria_widget.h

## C

```
typedef enum laBackgroundType_t {
  LA_WIDGET_BACKGROUND_NONE,
  LA_WIDGET_BACKGROUND_FILL,
  LA_WIDGET_BACKGROUND_CACHE,
  LA_WIDGET_BACKGROUND_LAST = LA_WIDGET_BACKGROUND_CACHE
} laBackgroundType;
```

## Description

Enumeration: laBackgroundType_t

Specifies the different background types used for the widgets in the library

None - No background fill. Widget must defer to its parent to erase dirty pixels. This may cause additional overhead as clean pixels may be repainted as well.

Fill - a scheme color is used to fill the widget rectangle.

Cache - a local framebuffer cache is maintained by the widget and used to clean up dirty pixels. Will not cause a parent repaint event but will use additional memory to contain the cache.

## Remarks

None.

### laWidget_LanguageChangingEvent_FnPtr Type

## File

libaria_widget.h

## C

```
typedef void (* laWidget_LanguageChangingEvent_FnPtr)(laWidget*);
```

## Description

This is type laWidget_LanguageChangingEvent_FnPtr.

### laWidgetOptimizationFlags Enumeration

Specifies the different draw optimization flags for a widget

## File

libaria_widget.h

## C

```
typedef enum laWidgetOptimizationFlags_t {
  LA_WIDGET_OPT_LOCAL_REDRAW = 0x1,
  LA_WIDGET_OPT_DRAW_ONCE = 0x2,
  LA_WIDGET_OPT_OPAQUE = 0x4
} laWidgetOptimizationFlags;
```

## Members

| Members | Description |
|---|---|
| LA_WIDGET_OPT_LOCAL_REDRAW = 0x1 | local redraw If a widget has no background then normally the parent would need to redraw to erase the contents of the widget. This flag indicates to the renderer to not redraw the parent event if the widget has no background |
| LA_WIDGET_OPT_DRAW_ONCE = 0x2 | draw once Indicates that a widget should draw once per screen show event all other attempts to invalidate or paint a widget will be rejected |
| LA_WIDGET_OPT_OPAQUE = 0x4 | opaque Indicates that a widget is fully opaque regardless of its background setting. This is often used for cases like image widgets where the image fills the entire widget space but you don't want the overhead of drawing a background behind it as well. This flag helps widgets without backgrounds to pass occlusion tests. |

## Description

Enumeration: laWidgetOptimizationFlags_t

Specifies the different draw optimization flags for a widget

## Remarks

None.

### LA_DEFAULT_SCHEME_COLOR_MODE Macro

## File

libaria_common.h

### C

```
#define LA_DEFAULT_SCHEME_COLOR_MODE GFX_COLOR_MODE_RGB_565
```

### Description

This is macro LA_DEFAULT_SCHEME_COLOR_MODE.

## LA_STRING_NULLIDX Macro

### File

libaria_string.h

### C

```
#define LA_STRING_NULLIDX -1
```

### Description

This is macro LA_STRING_NULLIDX.

## DEFAULT_BORDER_MARGIN Macro

### File

libaria_widget.h

### C

```
#define DEFAULT_BORDER_MARGIN 4
```

### Description

This is macro DEFAULT_BORDER_MARGIN.

## LA_IMAGESEQ_RESTART Macro

### File

libaria_widget_imagesequence.h

### C

```
#define LA_IMAGESEQ_RESTART -1
```

### Description

This is macro LA_IMAGESEQ_RESTART.

## LA_INPUT_PRIMARY_ID Macro

### File

libaria_input.h

### C

```
#define LA_INPUT_PRIMARY_ID 0
```

### Description

This is macro LA_INPUT_PRIMARY_ID.

## LA_MAX_TOUCH_STATES Macro

### File

libaria_input.h

### C

```
#define LA_MAX_TOUCH_STATES 2
```

### Description

This is macro LA_MAX_TOUCH_STATES.

## LA_TOUCHTEST_MEMORY_SIZE Macro

### File

libaria_widget_touchtest.h

### C

```
#define LA_TOUCHTEST_MEMORY_SIZE 20
```

### Description

This is macro LA_TOUCHTEST_MEMORY_SIZE.

## NUM_BUTTONS Macro

### File

libaria_input.h

### C

```
#define NUM_BUTTONS BUTTON_LAST + 1
```

### Description

This is macro NUM_BUTTONS.

## NUM_KEYS Macro

### File

libaria_input.h

### C

```
#define NUM_KEYS KEY_LAST + 1
```

### Description

This is macro NUM_KEYS.

## laLayer_AddDamageRect Function

Adds a damaged rectangle to the list. Damage rectangles are used in minimal redraw algorithms.

### File

libaria_layer.h

### C

```
LIB_EXPORT laResult laLayer_AddDamageRect(laLayer* layer, const GFX_Rect* rect, laBool noCombine);
```

### Returns

laResult - the result of the operation

### Parameters

| Parameters | Description |
|---|---|
| laLayer* layer | the layer |
| const GFX_Rect* rect | the rectangle |

### Function

laResult laLayer_AddDamageRect(laLayer* layer, const GFX_Rect* rect)

## laEventResult Enumeration

Defines what happened when processing an event

### File

libaria_event.h

### C

```
typedef enum laEventResult_t {
  LA_EVENT_HANDLED,
  LA_EVENT_DEFERRED,
  LA_EVENT_RESET_QUEUE
} laEventResult;
```

### Members

| Members | Description |
|---|---|
| LA_EVENT_HANDLED | the event was handled |
| LA_EVENT_DEFERRED | the event needs to wait |
| LA_EVENT_RESET_QUEUE | the entire event queue should be flushed and reset |

### Description

Enumeration: laEventResult

## laLayerFrameState Enumeration

Defines the frame state of a layer. Certain actions must only be performed at the start of a new frame and other actions must wait until the end of the current frame.

### File

libaria_layer.h

### C

```
typedef enum laLayerFrameState_t {
  LA_LAYER_FRAME_READY,
  LA_LAYER_FRAME_PREFRAME,
  LA_LAYER_FRAME_IN_PROGRESS,
  LA_LAYER_FRAME_COMPLETE
} laLayerFrameState;
```

### Description

Enumeration: laLayerFrameState

### Remarks

None.

## laRectArray Type

### File

libaria_widget.h

### C

```
typedef struct laRectArray_t laRectArray;
```

### Description

This is type laRectArray.

## laWidget_InvalidateBorderAreas_FnPtr Type

### File

libaria_widget.h

### C

```
typedef void (* laWidget_InvalidateBorderAreas_FnPtr)(laWidget*);
```

### Description

This is type laWidget_InvalidateBorderAreas_FnPtr.

## laContextFrameState Enumeration

Possible values for context frame state.

### File

libaria_context.h

### C

```
typedef enum laContextFrameState_t {
  LA_CONTEXT_FRAME_READY = 0,
  LA_CONTEXT_FRAME_PREFRAME,
  LA_CONTEXT_FRAME_PRELAYER,
  LA_CONTEXT_FRAME_DRAWING,
  LA_CONTEXT_FRAME_POSTLAYER
} laContextFrameState;
```

### Description

Enumeration: laContextFrameState_t

Possible values for context frame state.

### Remarks

None.

## laImageWidget_DrawEventCallback Type

### File

libaria_widget_image.h

### C

```
typedef void (* laImageWidget_DrawEventCallback)(laImageWidget*);
```

### Section

Data Types and Constants

## laContextUpdateState Enumeration

Possible values for context update state.

### File

libaria_context.h

### C

```
typedef enum laContextUpdateState_t {
  LA_CONTEXT_UPDATE_DONE = 0,
  LA_CONTEXT_UPDATE_PENDING
} laContextUpdateState;
```

### Description

Enumeration: laContextUpdateState

Possible values for context update state.

### Remarks

None.

## laWidgetUpdateState Enumeration

Specifies the different update states the widget can be assigned

### File

libaria_widget.h

### C

```
typedef enum laWidgetUpdateState_t {
  LA_WIDGET_UPDATE_STATE_DONE,
  LA_WIDGET_UPDATE_STATE_PENDING
} laWidgetUpdateState;
```

### Description

Enumeration: laWidgetUpdateState_t

Specifies the different update states the widget can be assigned

### Remarks

None.

## Files

### Files

| Name | Description |
|---|---|
| libaria_common.h | This file defines the common macros and definitions used by the gfx definition and implementation headers. |
| libaria_context.h | Context definition for the Aria user interface library. |
| libaria_draw.h | Internal standard drawing help function definitions. |
| libaria_editwidget.h | |
| libaria_event.h | Defines events that are used in the UI library. Events are created and stored for later processing during a library context's update loop. |
| libaria_global.h | This file contains global definitions used by the Aria user interface library. |
| libaria_input.h | |
| libaria_layer.h | Aria layers map directly to layers provided by the Graphics Hardware Abstraction layer. HAL layers map directly to hardware layers provided by graphics hardware. UI layers are logical containers for widgets and provide many of the same features. |
| libaria_list.h | A linked list implementation for the Aria user interface library |
| libaria_math.h | This is file libaria_math.h. |
| libaria_radiobutton_group.h | |
| libaria_scheme.h | A scheme is a collection of colors that can be referenced by one or more widgets. Widgets may use schemes in different ways. While the color names strive to be intuitive they aren't always used in the manner in which they describe. |
| libaria_screen.h | A screen describes the state of a set of layers. It can be orthogonally rotated and its life-cycle can be configured. |
| libaria_string.h | A string library implementation for the Aria user interface library. |
| libaria_utils.h | General internal utilities for the library |
| libaria_widget.h | |
| libaria_widget_button.h | Defines a button widget |
| libaria_widget_checkbox.h | |
| libaria_widget_circle.h | |
| libaria_widget_drawsurface.h | |
| libaria_widget_gradient.h | |
| libaria_widget_groupbox.h | |
| libaria_widget_image.h | |
| libaria_widget_imagesequence.h | |
| libaria_widget_keypad.h | |
| libaria_widget_label.h | |
| libaria_widget_line.h | |

| | |
|---|---|
| libaria_widget_list.h | |
| libaria_widget_listwheel.h | |
| libaria_widget_progressbar.h | |
| libaria_widget_radiobutton.h | |
| libaria_widget_rectangle.h | |
| libaria_widget_scrollbar.h | |
| libaria_widget_slider.h | |
| libaria_widget_textfield.h | |
| libaria_widget_touchtest.h | |
| libaria_widget_window.h | Window Widget |

## Description

### *libaria_common.h*

This file defines the common macros and definitions used by the gfx definition and implementation headers.

## Enumerations

| | Name | Description |
|---|---|---|
| | laBool_t | libaria bool values |
| | laPreemptionLevel | libaria pre-emption level values |
| | laRelativePosition | libaria relative position values |
| | laResult_t | libaria results (success and failure codes). |
| | laBool | libaria bool values |
| | laHAlignment | libaria horizontal alignment values |
| | laResult | libaria results (success and failure codes). |
| | laVAlignment | libaria vertical alignment values |

## Macros

| | Name | Description |
|---|---|---|
| | LA_DEFAULT_SCHEME_COLOR_MODE | This is macro LA_DEFAULT_SCHEME_COLOR_MODE. |

## Structures

| | Name | Description |
|---|---|---|
| | laMargin_t | libaria margin values |
| | laMargin | libaria margin values |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This file defines the common macros and definitions used by the gfx definition and the implementation header.

## Remarks

The directory in which this file resides should be added to the compiler's search path for header files.

## File Name

libaria_common.h

## Company

Microchip Technology Inc.

### *libaria_context.h*

Context definition for the Aria user interface library.

## Enumerations

| | Name | Description |
|---|---|---|
| | laContextFrameState_t | Possible values for context frame state. |

| | | |
|---|---|---|
| | laContextUpdateState_t | Possible values for context update state. |
| | laContextFrameState | Possible values for context frame state. |
| | laContextUpdateState | Possible values for context update state. |

## Functions

| | Name | Description |
|---|---|---|
| | laContext_AddScreen | Add screen to the list of screens in the current context |
| | laContext_Create | Creates an instance of the Aria user interface library |
| | laContext_Destroy | Destroys an Aria instance |
| | laContext_GetActive | Returns the current active context. |
| | laContext_GetActiveScreen | Returns the active screen of the current context |
| | laContext_GetActiveScreenIndex | Return the index of the active screen |
| | laContext_GetColorMode | Returns the color mode of the current context |
| | laContext_GetDefaultScheme | Returns the pointer to the default scheme of the current context |
| | laContext_GetEditWidget | Gets the widget that is currently recieving all widget edit events. |
| | laContext_GetFocusWidget | Return a pointer to the widget in focus |
| | laContext_GetPremptionLevel | Returns the preemption level for the screen |
| | laContext_GetScreenRect | Returns the display rectangle structure of the physical display |
| | laContext_GetStringLanguage | Returns the language index of the current context |
| | laContext_GetStringTable | Get a pointer to the GFXU_StringTableAsset structure that maintains the strings, associated fonts, etc |
| | laContext_HideActiveScreen | Hide the active screen |
| | laContext_IsDrawing | Indicates if any layers of the active screen are currently drawing a frame. |
| | laContext_IsLayerDrawing | Indicates if the layer at the given index of the active screen is currently drawing. |
| | laContext_RedrawAll | Forces the library to redraw the currently active screen in its entirety. |
| | laContext_RemoveScreen | Remove the specified screen from the list of screens in the current context |
| | laContext_SetActive | Make the specified context active |
| | laContext_SetActiveScreen | Change the active screen to the one specified by the index argument |
| | laContext_SetActiveScreenChangedCallback | Set the callback function pointer when the screen change event occurs |
| | laContext_SetEditWidget | Sets the currently active edit widget. |
| | laContext_SetFocusWidget | Set into focus the widget specified as the argument |
| | laContext_SetLanguageChangedCallback | Set the callback function pointer when the language change event occurs |
| | laContext_SetPreemptionLevel | Set the preemption level to the specified value |
| | laContext_SetStringLanguage | Set the language index of the current context |
| | laContext_SetStringTable | Set the StringTable pointer to the specified new StringTableAsset structure |
| | laContext_Update | Runs the update loop for a library instance. |

## Structures

| | Name | Description |
|---|---|---|
| | laContext_t | An instance of the Aria user interface library. |

## Types

| | Name | Description |
|---|---|---|
| | laContext_ActiveScreenChangedCallback_FnPtr | Callback pointer for the active screen change notification. |
| | laContext_LanguageChangedCallback_FnPtr | Callback pointer for when the language change event occurs. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_context.h

## Company

Microchip Technology Inc.

## *libaria_draw.h*

Internal standard drawing help function definitions.

### Functions

| | Name | Description |
|---|---|---|
| | laDraw_1x2BevelBorder | Internal utility function to draw a 1x2 bevel border |
| | laDraw_2x1BevelBorder | Internal utility function to draw a 2x1 bevel border |
| | laDraw_2x2BevelBorder | Internal utility function to draw a 2x2 bevel border |
| | laDraw_LineBorder | Internal utility function to draw a basic line border |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

### File Name

libaria_draw.h

### Company

Microchip Technology Inc.

## *libaria_editwidget.h*

### Functions

| | Name | Description |
|---|---|---|
| | laEditWidget_Accept | This is function laEditWidget_Accept. |
| | laEditWidget_Append | This is function laEditWidget_Append. |
| | laEditWidget_Backspace | This is function laEditWidget_Backspace. |
| | laEditWidget_Clear | This is function laEditWidget_Clear. |
| | laEditWidget_EndEdit | This is function laEditWidget_EndEdit. |
| | laEditWidget_Set | This is function laEditWidget_Set. |
| | laEditWidget_StartEdit | This is function laEditWidget_StartEdit. |

### Structures

| | Name | Description |
|---|---|---|
| | laEditWidget_t | Specifies the edit widget structure to manage all properties and events associated with edit widgets |
| | laEditWidget | Specifies the edit widget structure to manage all properties and events associated with edit widgets |

### Types

| | Name | Description |
|---|---|---|
| | laEditWidget_Accept_FnPtr | This is type laEditWidget_Accept_FnPtr. |
| | laEditWidget_Append_FnPtr | This is type laEditWidget_Append_FnPtr. |
| | laEditWidget_Backspace_FnPtr | This is type laEditWidget_Backspace_FnPtr. |
| | laEditWidget_Clear_FnPtr | This is type laEditWidget_Clear_FnPtr. |
| | laEditWidget_EndEdit_FnPtr | This is type laEditWidget_EndEdit_FnPtr. |
| | laEditWidget_Set_FnPtr | This is type laEditWidget_Set_FnPtr. |
| | laEditWidget_StartEdit_FnPtr | This is type laEditWidget_StartEdit_FnPtr. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements the routines to enable edit of library widgets.

### File Name

libaria_editwidget.h

## Company

Microchip Technology Inc.

### *libaria_event.h*

Defines events that are used in the UI library. Events are created and stored for later processing during a library context's update loop.

## Enumerations

| | Name | Description |
|---|---|---|
| | laEventID_t | Defines internal event type IDs |
| | laEventResult_t | Defines what happened when processing an event |
| | laEventID | Defines internal event type IDs |
| | laEventResult | Defines what happened when processing an event |

## Functions

| | Name | Description |
|---|---|---|
| | laEvent_AddEvent | Add the mentioned event callback to the list of events maintained by the current context |
| | laEvent_ClearList | Clear the event list maintained by the current context. |
| | laEvent_GetCount | Returns the number of events listed in the current context |
| | laEvent_ProcessEvents | Processes the screen change as well as touch events |
| | laEvent_SetFilter | Set callback pointer for current context filter event |

## Structures

| | Name | Description |
|---|---|---|
| | laEvent_t | Basic UI event definition |
| | laEventState_t | Structure to manage the event lists, state and call back pointers |
| | laWidgetEvent_t | Basic widget event definition |
| | laEvent | Basic UI event definition |
| | laEventState | Structure to manage the event lists, state and call back pointers |
| | laWidgetEvent | Basic widget event definition |

## Types

| | Name | Description |
|---|---|---|
| | laEvent_FilterEvent | Function pointer to define an event filter. Event filters allow a receiver to discard undesirable events |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_event.h

## Company

Microchip Technology Inc.

### *libaria_global.h*

This file contains global definitions used by the Aria user interface library.

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_global.h

## Company

Microchip Technology Inc.

## *libaria_input.h*

### Enumerations

| | Name | Description |
|---|---|---|
| | laGestureID_t | Placeholder for eventual gesture support. |
| | laKey_t | All values possible for key entry from the libaria keyboard widget |
| | laMouseButton_t | All values possible for mouse key entry from the libaria mouse input |
| | laGestureID | Placeholder for eventual gesture support. |
| | laKey | All values possible for key entry from the libaria keyboard widget |
| | laMouseButton | All values possible for mouse key entry from the libaria mouse input |

### Functions

| | Name | Description |
|---|---|---|
| | laInput_GetEnabled | Returns the input enabled status of the current context |
| | laInput_InjectTouchDown | Register and track the touch down event and queue it for handling by associated widgets |
| | laInput_InjectTouchMoved | Register and track the touch moved event and queue it for handling by associated widgets |
| | laInput_InjectTouchUp | Register and track the touch up event and queue it for handling by associated widgets |
| | laInput_SetEnabled | Sets the input status of the current context with the specified input argument |

### Macros

| | Name | Description |
|---|---|---|
| | LA_INPUT_PRIMARY_ID | This is macro LA_INPUT_PRIMARY_ID. |
| | LA_MAX_TOUCH_STATES | This is macro LA_MAX_TOUCH_STATES. |
| | NUM_BUTTONS | This is macro NUM_BUTTONS. |
| | NUM_KEYS | This is macro NUM_KEYS. |

### Structures

| | Name | Description |
|---|---|---|
| | laInput_TouchDownEvent_t | Register and handle the touch press detect event |
| | laInput_TouchMovedEvent_t | Register and handle the touch coordinates changed event |
| | laInput_TouchUpEvent_t | Register and handle the touch release detect event |
| | laInputState_t | Maintain a history of touch states; currently libaria keeps track of the last touch state only. |
| | laTouchState_t | Manage the touch input state and track the touch coordinate |
| | laInput_TouchDownEvent | Register and handle the touch press detect event |
| | laInput_TouchMovedEvent | Register and handle the touch coordinates changed event |
| | laInput_TouchUpEvent | Register and handle the touch release detect event |
| | laInputState | Maintain a history of touch states; currently libaria keeps track of the last touch state only. |
| | laTouchState | Manage the touch input state and track the touch coordinate |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

### File Name

libaria_input.h

### Company

Microchip Technology Inc.

## *libaria_layer.h*

Aria layers map directly to layers provided by the Graphics Hardware Abstraction layer. HAL layers map directly to hardware layers provided by graphics hardware. UI layers are logical containers for widgets and provide many of the same features.

## Enumerations

| | Name | Description |
|---|---|---|
| | laLayerBufferType_t | Defines the type of a layer. If the layer has an explicit address then Aria tries to set that through the HAL when the layer is being set up. |
| | laLayerFrameState_t | Defines the frame state of a layer. Certain actions must only be performed at the start of a new frame and other actions must wait until the end of the current frame. |
| | laLayerBufferType | Defines the type of a layer. If the layer has an explicit address then Aria tries to set that through the HAL when the layer is being set up. |
| | laLayerFrameState | Defines the frame state of a layer. Certain actions must only be performed at the start of a new frame and other actions must wait until the end of the current frame. |

## Functions

| | Name | Description |
|---|---|---|
| | laLayer_AddDamageRect | Adds a damaged rectangle to the list. Damage rectangles are used in minimal redraw algorithms. |
| | laLayer_Delete | Destructor for the layer object |
| | laLayer_GetAllowInputPassThrough | Gets the layer's input passthrough setting |
| | laLayer_GetAlphaAmount | Get's the amount of alpha blending for a given layer |
| | laLayer_GetAlphaEnable | Gets the layer alpha enable flag |
| | laLayer_GetBufferCount | Return the buffer count for the current layer |
| | laLayer_GetEnabled | Returns the boolean value of the layer enabled property |
| | laLayer_GetInputRect | Gets the layer's input rectangle. |
| | laLayer_GetInputRectLocked | Gets the layer's input rect locked flag |
| | laLayer_GetMaskColor | Returns the mask color value for the current layer |
| | laLayer_GetMaskEnable | Gets the layer mask enable flag |
| | laLayer_GetVSync | Gets the layer's vsync flag setting |
| | laLayer_IsDrawing | Queries a layer to find out if it is currently drawing a frame. |
| | laLayer_New | Constructor for a new layer |
| | laLayer_SetAllowInputPassthrough | Sets the layer's input passthrough flag. |
| | laLayer_SetAlphaAmount | Set's the amount of alpha blending for a given layer |
| | laLayer_SetAlphaEnable | Sets the layer alpha enable flag to the specified value |
| | laLayer_SetBufferCount | Set the buffer count for the current layer to the specified value |
| | laLayer_SetEnabled | Sets the boolean value of the layer enabled property |
| | laLayer_SetInputRect | Sets the layer's input rect dimensions. |
| | laLayer_SetInputRectLocked | Sets the layer's input rect locked flag. |
| | laLayer_SetMaskColor | Set the mask color value for the current layer to the specified value |
| | laLayer_SetMaskEnable | Sets the layer mask enable flag to the specified value |
| | laLayer_SetVSync | Sets the layer's vsync flag. |

## Structures

| | Name | Description |
|---|---|---|
| | laLayer_t | Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent. |
| | laLayerBuffer_t | Structure to maintain the buffer type and track the buffer location for each layer |
| | laLayerBuffer | Structure to maintain the buffer type and track the buffer location for each layer |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_layer.h

## Company

Microchip Technology Inc.

## *libaria_list.h*

A linked list implementation for the Aria user interface library

### Functions

| | Name | Description |
|---|---|---|
| | laList_Assign | Assignes a new pointer to an index in the list |
| | laList_Clear | Removes all nodes from a given list |
| | laList_Copy | Creates a duplicate of an existing list |
| | laList_Create | Initializes a new linked list |
| | laList_Destroy | Removes all nodes from a given list and frees the data of each node |
| | laList_Find | Retrieves the index of a value from the list |
| | laList_Get | Retrieves a value from the list |
| | laList_InsertAt | Inserts an item into a list at a given index. All existing from index are shifted right one place. |
| | laList_PopBack | Removes the last value from the list |
| | laList_PopFront | Removes the first value from the list |
| | laList_PushBack | Pushes a new node onto the back of the list |
| | laList_PushFront | Pushes a new node onto the front of the list |
| | laList_Remove | Removes an item from the list |
| | laList_RemoveAt | Removes an item from the list at an index |

### Structures

| | Name | Description |
|---|---|---|
| | laList_t | Linked list definition |
| | laListNode_t | Linked list node definition |
| | laListNode | Linked list node definition |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This is a linked list implementation that is used internally by the Aria user interface library. All of the memory operations are handled by the memory interface that is provided by the active libaria context. Applications that wish to use this implementation must ensure that the appropriate libaria context is active when calling these functions.

### File Name

libaria_list.h

### Company

Microchip Technology Inc.

## *libaria_math.h*

This is file libaria_math.h.

## *libaria_radiobutton_group.h*

### Functions

| | Name | Description |
|---|---|---|
| | laRadioButtonGroup_AddButton | Add a button widget to the button list of the selected Radio button group. |
| | laRadioButtonGroup_Create | This function creates a GFX_GOL_RADIOBUTTON group with the provided button list. |
| | laRadioButtonGroup_Destroy | This function destroys the GFX_GOL_RADIOBUTTON group |
| | laRadioButtonGroup_RemoveButton | Remove a button widget to the button list of the selected Radio button group. |
| | laRadioButtonGroup_SelectButton | Select the button widget specified from the button list for the Radio button group. |

### Structures

| | Name | Description |
|---|---|---|
| | laRadioButtonGroup_t | Defines the structure used for the Radio Button group. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements functions to control a radio button group.

## File Name

libaria_radiobutton_group.h

## Company

Microchip Technology Inc.

### *libaria_scheme.h*

A scheme is a collection of colors that can be referenced by one or more widgets. Widgets may use schemes in different ways. While the color names strive to be intuitive they aren't always used in the manner in which they describe.

## Functions

| | Name | Description |
|---|---|---|
| | laScheme_Initialize | Initialize the scheme to the default values as per the specified color mode. |

## Structures

| | Name | Description |
|---|---|---|
| | laScheme_t | This structure specifies the style scheme components of an object. |
| | laScheme | This structure specifies the style scheme components of an object. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_scheme.h

## Company

Microchip Technology Inc.

### *libaria_screen.h*

A screen describes the state of a set of layers. It can be orthogonally rotated and its life-cycle can be configured.

## Enumerations

| | Name | Description |
|---|---|---|
| | laScreenOrientation_t | Possible values for screen orientation. |
| | laScreenOrientation | Possible values for screen orientation. |

## Functions

| | Name | Description |
|---|---|---|
| | laScreen_Delete | Frees all memory for all layers and widgets for this screen |
| | laScreen_GetHideEventCallback | Returns the hide call back event function pointer for the specified screen |
| | laScreen_GetLayerIndex | Returns the index of the layer for the screen specified. |
| | laScreen_GetLayerSwapSync | Returns the layer swap sync setting for the specified screen |
| | laScreen_GetMirrored | Returns the mirror setting for the specified screen |
| | laScreen_GetOrientation | Returns the orientation object associated with the specified screen |
| | laScreen_GetShowEventCallback | Returns the show call back event function pointer for the specified screen |
| | laScreen_Hide | Hide the currently active screen<br>This function has been deprecated in favor of laContext_SetActiveScreen |
| | laScreen_New | Create a new screen, initialize it to the values specified. |
| | laScreen_SetHideEventCallback | Set the hide call back event function pointer for the specified screen |
| | laScreen_SetLayer | Assigns the provided layer pointer to the screen at the given index<br>This function has been deprecated in favor of laContext_SetActiveScreen |
| | laScreen_SetLayerSwapSync | Sets the layer swap sync setting for the specified screen |

| | | |
|---|---|---|
| | laScreen_SetMirrored | Sets the mirror setting for the specified screen |
| | laScreen_SetOrientation | Sets the orientation object to the specified screen |
| | laScreen_SetShowEventCallback | Set the show call back event function pointer for the specified screen |
| | laScreen_Show | Make the specified screen active and show it on the display |

## Structures

| | Name | Description |
|---|---|---|
| | laScreen_t | The structure to maintain the screen related variables and event handling |
| | laScreen | The structure to maintain the screen related variables and event handling |

## Types

| | Name | Description |
|---|---|---|
| | laScreen_CreateCallback_FnPtr | Callback pointer for a new screen create event notification. This is called when the library attempts to create a screen. |
| | laScreen_ShowHideCallback_FnPtr | Callback pointer for the active screen show or hide event change notification. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_screen.h

## Company

Microchip Technology Inc.

### *libaria_string.h*

A string library implementation for the Aria user interface library.

## Functions

| | Name | Description |
|---|---|---|
| | laString_Allocate | Attempts to resize the local data buffer for a string. |
| | laString_Append | Appends a string onto the end of another string |
| | laString_Capacity | Returns the capacity of a string |
| | laString_CharAt | Extracts the code point for the character in a string at a given index. |
| | laString_Clear | Sets a string's length to zero and its string table reference to NULL. Does not free any associated data and preserves capacity. |
| | laString_Compare | Compares two string objects |
| | laString_CompareBuffer | Compares a string object and a GFXU_CHAR* buffer |
| | laString_Copy | Copies the values from one string into another |
| | laString_CreateFromBuffer | Creates a string object from a GFXU_CHAR buffer and a font asset pointer |
| | laString_CreateFromCharBuffer | Creates a string object from a const char* buffer and a font asset pointer. This method provides compatibility with standard c-style strings. Input string will be converted from 8-bit with to 32-bit width. |
| | laString_CreateFromID | Creates a string object that simply references a string in the string table. |
| | laString_Delete | Deletes all memory associated with a string object |
| | laString_Destroy | Destroys a string object. This frees the strings internal data buffer, if it exists, sets its string table reference to null, and clears all supporting attributes. |
| | laString_Draw | Wrapper around GFX Utility string draw function for Aria user interface library. Internal use only. |
| | laString_DrawClipped | Wrapper around GFX Utility string draw function for Aria user interface library. Draws only a clipped area of a string. Internal use only. |
| | laString_DrawSubStringClipped | Wrapper around GFX Utility string draw function for Aria user interface library. Draws the substring between the start and end offset, and draws only the section of the string within the clipping rectangle. Internal use only. |
| | laString_ExtractFromTable | Extracts a read-only string from the string table into a modifiable string object. This relies on the active context to indicate which string table to reference as well as which language entry to extract. |
| | laString_GetCharIndexAtPoint | Given an offset in pixels returns the corresponding character index. |

| | | |
|---|---|---|
| | laString_GetCharOffset | Returns the offset of a given character index in pixels. |
| | laString_GetCharWidth | Given a character index, gets the width of that character. Only accurate if the string has a font associated with it and that font contains all the characters in the string in question. |
| | laString_GetHeight | Returns the height of a string by referencing its associated font asset data. |
| | laString_GetLineRect | Calculates the rectangle for a line in a string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. The line ends when a line feed or end of string is reached. |
| | laString_GetMultiLineRect | Calculates the rectangle for a given multi-line string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. The height the sum of the heights of the bounding rectangles for each line and the width is the widest among the bounding rectangles. |
| | laString_GetRect | Calculates the rectangle for a given string object. References the associated font for the height but must perform a summation for each character in the string by doing a font meta-data lookup. |
| | laString_Initialize | Initializes a string struct to default |
| | laString_Insert | Inserts a string into another string at a given index |
| | laString_IsEmpty | Returns a boolean indicating if the provided string contains data or has a link to the string table. |
| | laString_Length | Calculates the length of a string in characters |
| | laString_New | Allocates a memory for a new string |
| | laString_ReduceLength | Reduces the length of a string. This simply slides the null terminator to the left and does not affect the string's capacity value. |
| | laString_Remove | Removes a number of characters from a string at a given index |
| | laString_Set | Attempts to set the local data buffer of a string to an input buffer |
| | laString_SetCapacity | Attempts to adjust the capacity of a string |
| | laString_ToCharBuffer | Extracts the data buffer from a string and copies it into the provided buffer argument. |

## Macros

| | Name | Description |
|---|---|---|
| | LA_STRING_NULLIDX | This is macro LA_STRING_NULLIDX. |

## Structures

| | Name | Description |
|---|---|---|
| | laString_t | String definition |
| | laString | String definition |

## Types

| | Name | Description |
|---|---|---|
| | laContext | An instance of the Aria user interface library. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This is a string library implementation that is used internally by the Aria user interface library. All of the memory operations are handled by the memory interface that is provided by the active libaria context. Applications that wish to use this implementation must ensure that the appropriate libaria context is active when calling these functions.

This implementation relies on the GFXU_CHAR definition for characters provided by the GFX Utils library. This chararcter definition is 32 bits in size and allows libaria to support international character code points and Unicode encoding standards.

## File Name

libaria_string.h

## Company

Microchip Technology Inc.

### *libaria_utils.h*

General internal utilities for the library

## Functions

| | Name | Description |
|---|---|---|
| | laUtils_ArrangeRectangle | Calculates the position of a rectangle within the given bounds and in accordance with the given parameters. A use case for this is when an image and a text rectangle must be arranged in a button box. This version of the algorithm will calculate the location of the image rectangle. |
| | laUtils_ArrangeRectangleRelative | Calculates the position of a rectangle within the given bounds and in accordance with the given parameters. A use case for this is when an image and a text rectangle must be arranged in a button box. This version of the algorithm will calculate the location of the text rectangle. |
| | laUtils_ChildIntersectsParent | Performs an intersection test between a parent widget and a child widget |
| | laUtils_ClipRectToParent | Clips a rectangle to the parent of a widget |
| | laUtils_GetLayer | Finds the root parent of a widget, which should be a layer |
| | laUtils_GetNextHighestWidget | Gets the next highest Z order widget in the tree from 'wgt' |
| | laUtils_ListOcclusionCullTest | Performs an occlusion test on a list of widgets an a rectangular area. This attempts to find only the topmost widgets for the given area. If a widget is completely occluded then it is removed from the list. Any widgets that remain in the list should be redrawn by the rasterizer. |
| | laUtils_OcclusionCullTest | Performs an occlusion test for a widget in the tree. A widget is occluded if it is completely covered by one or more widgets. This is useful for culling widgets before the rasterizing phase. |
| | laUtils_Pick | Finds the top-most visible widget in the tree at the given coordinates. |
| | laUtils_PickFromLayer | Finds the top-most visible widget in a layer at the given coordinates. |
| | laUtils_PickRect | Finds all of the visible widgets in the given rectangular area. |
| | laUtils_PointScreenToLocalSpace | Converts a point from layer space into the local space of a widget |
| | laUtils_PointToLayerSpace | Converts a point from widget space into layer space |
| | laUtils_RectFromLayerSpace | Converts a rectangle from layer space to widget local space |
| | laUtils_RectFromParentSpace | Converts a rectangle from widget parent space to widget local space |
| | laUtils_RectToLayerSpace | Converts a rectangle from widget local space to layer space |
| | laUtils_RectToParentSpace | Converts a rectangle from widget local space to widget parent space. Widget must be a child of a layer for this to function. |
| | laUtils_RectToScreenSpace | Converts a rectangle from widget local space to screen space |
| | laUtils_ScreenToMirroredSpace | Takes a point in screen space and returns a transformed version in mirrored space. |
| | laUtils_ScreenToOrientedSpace | Takes a point in screen space and returns a transformed version in oriented space. |
| | laUtils_WidgetIsOccluded | This is function laUtils_WidgetIsOccluded. |
| | laUtils_WidgetLayerRect | Returns the bounding rectangle of a widget in layer space |
| | laUtils_WidgetLocalRect | Returns the bounding rectangle of a widget in local space |

## Types

| | Name | Description |
|---|---|---|
| | GFX_Point | A two dimensional Cartesian point. |
| | GFX_Rect | A rectangle definition. |
| | laLayer | Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent. |
| | laList | Linked list definition |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_utils.h

## Company

Microchip Technology Inc.

### *libaria_widget.h*

## Enumerations

| | Name | Description |
|---|---|---|
| | laBackgroundType_t | Specifies the different background types used for the widgets in the library |

| | | |
|---|---|---|
| | laBorderType_t | Specifies the different border types used for the widgets in the library |
| | laWidgetDirtyState_t | Specifies the different dirty states the widget can be assigned |
| | laWidgetDrawState_t | Specifies the different draw states the widget can be assigned |
| | laWidgetOptimizationFlags_t | Specifies the different draw optimization flags for a widget |
| | laWidgetType_t | Specifies the different widget types used in the library |
| | laWidgetUpdateState_t | Specifies the different update states the widget can be assigned |
| | laBackgroundType | Specifies the different background types used for the widgets in the library |
| | laBorderType | Specifies the different border types used for the widgets in the library |
| | laWidgetDirtyState | Specifies the different dirty states the widget can be assigned |
| | laWidgetDrawState | Specifies the different draw states the widget can be assigned |
| | laWidgetOptimizationFlags | Specifies the different draw optimization flags for a widget |
| | laWidgetType | Specifies the different widget types used in the library |
| | laWidgetUpdateState | Specifies the different update states the widget can be assigned |

## Functions

| | Name | Description |
|---|---|---|
| | laWidget_AddChild | Adds the child to the parent widget specified in the argument |
| | laWidget_Delete | Delete the widget object specified |
| | laWidget_DeleteAllDescendants | Deletes all of the descendants of the given widget. |
| | laWidget_GetAlphaAmount | Return the widget's global alpha amount |
| | laWidget_GetAlphaEnable | Return the alpha enable property of the widget |
| | laWidget_GetBackgroundType | Return the property value 'background type' associated with the widget object |
| | laWidget_GetBorderType | Return the border type associated with the widget object |
| | laWidget_GetChildAtIndex | Fetches the child at the specified index from the children list of the specified parent widget |
| | laWidget_GetChildCount | Returns the size of the children list of the specified parent widget |
| | laWidget_GetCumulativeAlphaAmount | Calculates the cumulative alpha amount for a hierarchy of widgets |
| | laWidget_GetCumulativeAlphaEnable | Determines if this or any ancestor widget has alpha enabled |
| | laWidget_GetEnabled | Returns the boolean value of the widget enabled property |
| | laWidget_GetHeight | Returns the widget rectangles height |
| | laWidget_GetIndexOfChild | Fetches the index of the child from the children list of the specified parent widget |
| | laWidget_GetMargin | Returns the margin value associated with the widget in the laMargin pointer |
| | laWidget_GetOptimizationFlags | Returns the optimization flags for the widget |
| | laWidget_GetScheme | Returns the scheme associated with the specified widget |
| | laWidget_GetVisible | Returns the boolean value of the widget visible property |
| | laWidget_GetWidth | Returns the widget rectangles width |
| | laWidget_GetX | Returns the widget rectangles upper left corner x-coordinate |
| | laWidget_GetY | Returns the widget rectangles upper left corner y-coordinate |
| | laWidget_HasFocus | Checks if the widget specified has focus in the current context |
| | laWidget_Invalidate | Invalidates the specified widget. |
| | laWidget_isOpaque | Returns true if the widget is considered opaque. |
| | laWidget_New | Create a new widget. |
| | laWidget_OverrideTouchDownEvent | Replace the TouchDownEvent callback for the widget with the new function pointer specified |
| | laWidget_OverrideTouchMovedEvent | Replace the TouchMovedEvent callback for the widget with the new function pointer specified |
| | laWidget_OverrideTouchUpEvent | Replace the TouchUpEvent callback for the widget with the new function pointer specified |
| | laWidget_RectToLayerSpace | Transforms a widget rectangle from local space to its root layer space. |
| | laWidget_RectToParentSpace | Returns the rectangle containing the parent of the widget specified |
| | laWidget_RectToScreenSpace | Transforms a widget rectangle from local space to screen space coordinates. |
| | laWidget_RemoveChild | Removes the child from the parent widget specified in the argument |
| | laWidget_Resize | Changes the widget size by the new defined width and height increments. |
| | laWidget_SetAlphaAmount | Set the widget's global alpha amount to the specified alpha amount |
| | laWidget_SetAlphaEnable | Set the alpha enable property of the widget with the boolean value specified |
| | laWidget_SetBackgroundType | Set the property value 'background type' associated with the widget object |
| | laWidget_SetBorderType | Set the border type associated with the widget object |

| | | |
|---|---|---|
| | laWidget_SetEnabled | Sets the boolean value of the widget enabled property |
| | laWidget_SetFocus | Set the widget into focus for the current context. |
| | laWidget_SetHeight | Sets the widget's height value |
| | laWidget_SetMargins | Set the margin value for left, right, top and bottom margins associated with the widget |
| | laWidget_SetOptimizationFlags | Sets the optimizations for a widget |
| | laWidget_SetParent | Sets the parent of the child widget to that specified in the argument list |
| | laWidget_SetPosition | Changes the widget position to the new defined x and y coordinates. |
| | laWidget_SetScheme | Sets the scheme variable for the specified widget |
| | laWidget_SetSize | Changes the widget size to the new defined width and height dimensions. |
| | laWidget_SetVisible | Sets the boolean value of the widget visible property |
| | laWidget_SetWidth | Sets the widget's width value |
| | laWidget_SetX | Sets the widget's x coordinate position |
| | laWidget_SetY | Sets the widget's y coordinate position |
| | laWidget_Translate | Changes the widget position by moving the widget by the defined x and y increments. |

## Macros

| | Name | Description |
|---|---|---|
| | DEFAULT_BORDER_MARGIN | This is macro DEFAULT_BORDER_MARGIN. |

## Structures

| | Name | Description |
|---|---|---|
| | laWidget_t | Specifies Graphics widget structure to manage all properties and events associated with the widget |
| | laWidget | Specifies Graphics widget structure to manage all properties and events associated with the widget |

## Types

| | Name | Description |
|---|---|---|
| | laRectArray | This is type laRectArray. |
| | laWidget_Constructor_FnPtr | This is type laWidget_Constructor_FnPtr. |
| | laWidget_Destructor_FnPtr | This is type laWidget_Destructor_FnPtr. |
| | laWidget_DrawFunction_FnPtr | This is type laWidget_DrawFunction_FnPtr. |
| | laWidget_Focus_FnPtr | This is type laWidget_Focus_FnPtr. |
| | laWidget_InvalidateBorderAreas_FnPtr | This is type laWidget_InvalidateBorderAreas_FnPtr. |
| | laWidget_LanguageChangingEvent_FnPtr | This is type laWidget_LanguageChangingEvent_FnPtr. |
| | laWidget_Moved_FnPtr | This is type laWidget_Moved_FnPtr. |
| | laWidget_Paint_FnPtr | This is type laWidget_Paint_FnPtr. |
| | laWidget_Resized_FnPtr | This is type laWidget_Resized_FnPtr. |
| | laWidget_TouchDownEvent_FnPtr | This is type laWidget_TouchDownEvent_FnPtr. |
| | laWidget_TouchMovedEvent_FnPtr | This is type laWidget_TouchMovedEvent_FnPtr. |
| | laWidget_TouchUpEvent_FnPtr | This is type laWidget_TouchUpEvent_FnPtr. |
| | laWidget_Update_FnPtr | This is type laWidget_Update_FnPtr. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements top level widget control functions.

## File Name

libaria_widget.h

## Company

Microchip Technology Inc.

### *libaria_widget_button.h*

Defines a button widget

## Enumerations

| | Name | Description |
|---|---|---|
| | laButtonState_t | Controls the button pressed state |
| | laButtonState | Controls the button pressed state |

## Functions

| | Name | Description |
|---|---|---|
| | laButtonWidget_GetHAlignment | Gets the horizontal alignment setting for a button |
| | laButtonWidget_GetImageMargin | Gets the distance between the icon and the text |
| | laButtonWidget_GetImagePosition | Gets the position of the button icon |
| | laButtonWidget_GetPressed | Gets the pressed state of a button |
| | laButtonWidget_GetPressedEventCallback | Gets the callback associated with the button pressed event |
| | laButtonWidget_GetPressedImage | Gets the pressed image asset pointer for a button |
| | laButtonWidget_GetPressedOffset | Gets the offset of the button internals when pressed |
| | laButtonWidget_GetReleasedEventCallback | Gets the callback for the button released event |
| | laButtonWidget_GetReleasedImage | Gets the currently used released icon |
| | laButtonWidget_GetText | Gets the text for a button. If the button's string has local data then a duplicate of the string will be allocated. The caller is responsible for managing the memory for the duplicated string. If the button string is a string table reference then only the reference ID is copied. |
| | laButtonWidget_GetToggleable | Gets the value of this button's toggle flag |
| | laButtonWidget_GetVAlignment | Gets the vertical alignment setting for a button |
| | laButtonWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laButtonWidget_SetHAlignment | Sets the horizontal alignment value for a button |
| | laButtonWidget_SetImageMargin | Sets the distance between the icon and text |
| | laButtonWidget_SetImagePosition | Sets the position of the button icon |
| | laButtonWidget_SetPressed | Sets the pressed state for a button. |
| | laButtonWidget_SetPressedEventCallback | Sets the pressed event callback for the button |
| | laButtonWidget_SetPressedImage | Sets the image to be used as a pressed icon |
| | laButtonWidget_SetPressedOffset | Sets the offset of the button internals when pressed |
| | laButtonWidget_SetReleasedEventCallback | Sets the callback for the button released event |
| | laButtonWidget_SetReleasedImage | Sets the image to be used as the released icon |
| | laButtonWidget_SetText | Sets the text for a button. If the input string has local data then the data will be copied into the button's local string, causing a memory allocation. If the input string is a string table reference then only the reference will be copied. The input string can be safely modified and the button string will not be affected. |
| | laButtonWidget_SetToggleable | Enables the toggle mode for a button. When pressed, toggle buttons will stay down until pressed again. |
| | laButtonWidget_SetVAlignment | Sets the vertical alignment for a button |

## Structures

| | Name | Description |
|---|---|---|
| | laButtonWidget_t | Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed an released state. |

## Types

| | Name | Description |
|---|---|---|
| | laButtonWidget_PressedEvent | This is type laButtonWidget_PressedEvent. |
| | laButtonWidget_ReleasedEvent | This is type laButtonWidget_ReleasedEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

## File Name

libaria_widget_button.h

## Company

Microchip Technology Inc.

### *libaria_widget_checkbox.h*

## Functions

| | Name | Description |
|---|---|---|
| | laCheckBoxWidget_GetChecked | Gets the checked state of the check box |
| | laCheckBoxWidget_GetCheckedEventCallback | Gets the checked event callback |
| | laCheckBoxWidget_GetCheckedImage | Gets the checked image of the check box |
| | laCheckBoxWidget_GetHAlignment | Gets the horizontal alignment of the check box |
| | laCheckBoxWidget_GetImageMargin | Gets the distance between the image and the text |
| | laCheckBoxWidget_GetImagePosition | Gets the image position of the check box |
| | laCheckBoxWidget_GetText | Gets a copy of the checkbox text. If the text has local data the data will be duplicated. The caller is responsible for managing the memory as appropriate. |
| | laCheckBoxWidget_GetUncheckedEventCallback | Gets the unchecked event callback |
| | laCheckBoxWidget_GetUncheckedImage | Gets the unchecked image of the check box |
| | laCheckBoxWidget_GetVAlignment | Gets the vertical alignment of the check box |
| | laCheckBoxWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laCheckBoxWidget_SetChecked | Sets the checked state of the check box |
| | laCheckBoxWidget_SetCheckedEventCallback | Sets the checked event callback |
| | laCheckBoxWidget_SetCheckedImage | Sets the checked image of the check box |
| | laCheckBoxWidget_SetHAlignment | Sets the horizontal alignment of the check box. |
| | laCheckBoxWidget_SetImageMargin | Sets the distance between the image and the text |
| | laCheckBoxWidget_SetImagePosition | Sets the image position of the check box |
| | laCheckBoxWidget_SetText | Sets the checkbox text to the input string. If the string has local data the data will be duplicated and copied to the checkboxes internal string. |
| | laCheckBoxWidget_SetUncheckedEventCallback | Sets the unchecked event callback |
| | laCheckBoxWidget_SetUncheckedImage | Sets the unchecked image of the check box |
| | laCheckBoxWidget_SetVAlignment | Sets the vertical alignment of the check box |

## Structures

| | Name | Description |
|---|---|---|
| | laCheckBoxWidget_t | Implementation of a checkbox widget. |
| | laCheckBoxWidget | Implementation of a checkbox widget. |

## Types

| | Name | Description |
|---|---|---|
| | laCheckBoxWidget_CheckedEvent | This is type laCheckBoxWidget_CheckedEvent. |
| | laCheckBoxWidget_UncheckedEvent | This is type laCheckBoxWidget_UncheckedEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements button widget functions.

## File Name

libaria_widget_button.h

## Company

Microchip Technology Inc.

## *libaria_widget_circle.h*

### Functions

| | Name | Description |
|---|---|---|
| | laCircleWidget_GetOrigin | Gets the origin coordiates of a circle widget |
| | laCircleWidget_GetRadius | Gets the radius of a circle widget |
| | laCircleWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laCircleWidget_SetOrigin | Sets the origin coordiates of a circle widget |
| | laCircleWidget_SetRadius | Sets the radius of a circle widget |

### Structures

| | Name | Description |
|---|---|---|
| | laCircleWidget_t | Implementation of a circle widget. |
| | laCircleWidget | Implementation of a circle widget. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements circle drawing widget functions.

### File Name

libaria_widget_circle.h

### Company

Microchip Technology Inc.

## *libaria_widget_drawsurface.h*

### Functions

| | Name | Description |
|---|---|---|
| | laDrawSurfaceWidget_GetDrawCallback | Returns the pointer to the currently set draw callback. |
| | laDrawSurfaceWidget_New | Allocates memory for a new DrawSurface widget. |
| | laDrawSurfaceWidget_SetDrawCallback | Sets the draw callback pointer for the draw surface widget. |

### Structures

| | Name | Description |
|---|---|---|
| | laDrawSurfaceWidget_t | Implementation of a Drawsurface widget. |
| | laDrawSurfaceWidget | Implementation of a Drawsurface widget. |

### Types

| | Name | Description |
|---|---|---|
| | laDrawSurfaceWidget_DrawCallback | This is type laDrawSurfaceWidget_DrawCallback. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements surface container drawing functions.

### File Name

libaria_widget_drawsurface.h

### Company

Microchip Technology Inc.

## *libaria_widget_gradient.h*

### Enumerations

| | Name | Description |
|---|---|---|
| | laGradientWidgetDirection_t | Implementation of a gradient widget. |
| | laGradientWidgetDirection | Implementation of a gradient widget. |

### Functions

| | Name | Description |
|---|---|---|
| | laGradientWidget_GetDirection | Gets the gradient direction value for this widget. |
| | laGradientWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laGradientWidget_SetDirection | Sets the gradient direction value for this widget. |

### Structures

| | Name | Description |
|---|---|---|
| | laGradientWidget_t | Gradient widget struct definition. |
| | laGradientWidget | Gradient widget struct definition. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements gradient drawing widget functions.

### File Name

libaria_widget_gradient.h

### Company

Microchip Technology Inc.

## *libaria_widget_groupbox.h*

### Functions

| | Name | Description |
|---|---|---|
| | laGroupBoxWidget_GetAlignment | Gets the horizontal alignmnet for the group box title text |
| | laGroupBoxWidget_GetText | Gets the text value for the group box. |
| | laGroupBoxWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laGroupBoxWidget_SetAlignment | Sets the alignment for the group box title text |
| | laGroupBoxWidget_SetText | Sets the text value for the group box. |

### Structures

| | Name | Description |
|---|---|---|
| | laGroupBoxWidget_t | Group box struct definition. |
| | laGroupBoxWidget | Group box struct definition. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements group box widget functions.

### File Name

libaria_widget_groupbox.h

### Company

Microchip Technology Inc.

## *libaria_widget_image.h*

### Functions

| | Name | Description |
|---|---|---|
| | laImageWidget_GetHAlignment | Gets the image horizontal alignment value. |
| | laImageWidget_GetImage | Gets the image asset pointer for the widget. |
| | laImageWidget_GetVAlignment | Gets the image vertical alignment value. |
| | laImageWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laImageWidget_SetCallBackEnd | This is function laImageWidget_SetCallBackEnd. |
| | laImageWidget_SetCallBackStart | This is function laImageWidget_SetCallBackStart. |
| | laImageWidget_SetHAlignment | Sets the image horizontal alignment value. |
| | laImageWidget_SetImage | Sets the image asset pointer for the widget. |
| | laImageWidget_SetVAlignment | Sets the image vertical alignment value. |

### Structures

| | Name | Description |
|---|---|---|
| | laImageWidget_t | Image widget struct definition |
| | laImageWidget | Image widget struct definition |

### Types

| | Name | Description |
|---|---|---|
| | laImageWidget_DrawEventCallback | |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements image widget functions.

### File Name

libaria_widget_image.h

### Company

Microchip Technology Inc.

## *libaria_widget_imagesequence.h*

### Functions

| | Name | Description |
|---|---|---|
| | laImageSequenceWidget_GetImage | Gets the image asset pointer for an entry. |
| | laImageSequenceWidget_GetImageChangedEventCallback | Gets the image changed event callback pointer. |
| | laImageSequenceWidget_GetImageCount | Gets the number of image entries for this widget. |
| | laImageSequenceWidget_GetImageDelay | Gets the image delay for an entry. |
| | laImageSequenceWidget_GetImageHAlignment | Gets the horizontal alignment for an image entry |
| | laImageSequenceWidget_GetImageVAlignment | Sets the vertical alignment for an image entry |
| | laImageSequenceWidget_GetRepeat | Indicates if the widget will repeat through the image entries. |
| | laImageSequenceWidget_IsPlaying | Indicates if the widget is currently cycling through the image entries. |
| | laImageSequenceWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laImageSequenceWidget_Play | Starts the widget automatically cycling through the image entries. |
| | laImageSequenceWidget_Rewind | Resets the current image sequence display index to zero. |
| | laImageSequenceWidget_SetImage | Sets the image asset pointer for an entry. |
| | laImageSequenceWidget_SetImageChangedEventCallback | Sets the image changed event callback pointer. This callback is called whenever the active display index is changed. |
| | laImageSequenceWidget_SetImageCount | Sets the number of image entries for this widget. An image entry that is null will show nothing. |

| | | |
|---|---|---|
| | laImageSequenceWidget_SetImageDelay | Sets the image delay for an entry. |
| | laImageSequenceWidget_SetImageHAlignment | Sets the horizontal alignment for an image entry. |
| | laImageSequenceWidget_SetImageVAlignment | Sets the vertical alignment value for an image entry |
| | laImageSequenceWidget_SetRepeat | Sets the repeat flag for the widget |
| | laImageSequenceWidget_ShowImage | Sets the active display index to the indicated value. |
| | laImageSequenceWidget_ShowNextImage | Sets the active display index to the next index value. |
| | laImageSequenceWidget_ShowPreviousImage | Sets the active display index to the previous index value. |
| | laImageSequenceWidget_Stop | Stops the widget from automatically cycling through the image entries. |

## Macros

| | Name | Description |
|---|---|---|
| | LA_IMAGESEQ_RESTART | This is macro LA_IMAGESEQ_RESTART. |

## Structures

| | Name | Description |
|---|---|---|
| | laImageSequenceEntry_t | Image sequence entry definition |
| | laImageSequenceWidget_t | Image sequence widget struct definition |
| | laImageSequenceEntry | Image sequence entry definition |
| | laImageSequenceWidget | Image sequence widget struct definition |

## Types

| | Name | Description |
|---|---|---|
| | laImageSequenceImageChangedEvent_FnPtr | This is type laImageSequenceImageChangedEvent_FnPtr. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements image sequence (slide show) widget drawing functions.

## File Name

libaria_widget_imagesequence.h

## Company

Microchip Technology Inc.

### *libaria_widget_keypad.h*

## Enumerations

| | Name | Description |
|---|---|---|
| | laKeyPadCellAction_t | Defines an assigned action to a key pad cell |
| | laKeyPadCellAction | Defines an assigned action to a key pad cell |

## Functions

| | Name | Description |
|---|---|---|
| | laKeyPadWidget_GetKeyAction | Gets the key pad cell action for a cell at row/column |
| | laKeyPadWidget_GetKeyClickEventCallback | Gets the current key click event callback pointer |
| | laKeyPadWidget_GetKeyDrawBackground | Gets the background type for a key pad cell at row/column |
| | laKeyPadWidget_GetKeyEnabled | Gets the enabled flag for a cell at a given row/column |
| | laKeyPadWidget_GetKeyImageMargin | Gets the key pad cell image margin value |
| | laKeyPadWidget_GetKeyImagePosition | Gets the image position for a key pad cell |
| | laKeyPadWidget_GetKeyPressedImage | Gets the pressed icon image asset pointer for the display image for a key pad cell |
| | laKeyPadWidget_GetKeyReleasedImage | Gets the released icon image asset pointer for the display image for a key pad cell |
| | laKeyPadWidget_GetKeyText | Returns a copy of the display text for a given cell at row/column |
| | laKeyPadWidget_GetKeyValue | Gets the edit text value for a given key pad cell. |
| | laKeyPadWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laKeyPadWidget_SetKeyAction | Sets the cell action type for a key pad cell at row/column |

| | | |
|---|---|---|
| | laKeyPadWidget_SetKeyBackgroundType | Sets the background type for a key pad cell at row/column |
| | laKeyPadWidget_SetKeyClickEventCallback | Sets the current key click event callback pointer |
| | laKeyPadWidget_SetKeyEnabled | Sets the enabled flag for a cell at the given row/column |
| | laKeyPadWidget_SetKeyImageMargin | Sets the key pad cell image margin value for a given cell at row/column |
| | laKeyPadWidget_SetKeyImagePosition | |
| | laKeyPadWidget_SetKeyPressedImage | Sets the pressed icon image asset pointer for a key pad cell |
| | laKeyPadWidget_SetKeyReleasedImage | Sets the released icon image asset pointer for a key pad cell |
| | laKeyPadWidget_SetKeyText | Sets the display text for a given cell at row/column |
| | laKeyPadWidget_SetKeyValue | Sets the edit value for a given key pad cell. |

## Structures

| | Name | Description |
|---|---|---|
| | laKeyPadCell_t | Defines a key pad cell struct |
| | laKeyPadWidget_t | Defines a key pad widget struct |
| | laKeyPadCell | Defines a key pad cell struct |
| | laKeyPadWidget | Defines a key pad widget struct |

## Types

| | Name | Description |
|---|---|---|
| | laButtonWidget | Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed an released state. |
| | laKeyPadWidget_KeyClickEvent | This is type laKeyPadWidget_KeyClickEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements keypad widget functions.

## File Name

libaria_widget_keypad.h

## Company

Microchip Technology Inc.

### *libaria_widget_label.h*

## Functions

| | Name | Description |
|---|---|---|
| | laLabelWidget_GetHAlignment | Gets the text horizontal alignment value. |
| | laLabelWidget_GetText | Gets the text value for the label. |
| | laLabelWidget_GetVAlignment | Gets the current vertical text alignment |
| | laLabelWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laLabelWidget_SetHAlignment | Sets the text horizontal alignment value |
| | laLabelWidget_SetText | Sets the text value for the label. |
| | laLabelWidget_SetVAlignment | Sets the vertical text alignment value |

## Structures

| | Name | Description |
|---|---|---|
| | laLabelWidget_t | Implementation of a label widget struct |
| | laLabelWidget | Implementation of a label widget struct |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements label (text) widget functions.

## File Name

libaria_widget_label.h

## Company

Microchip Technology Inc.

### *libaria_widget_line.h*

## Functions

| | Name | Description |
|---|---|---|
| | laLineWidget_GetEndPoint | Gets the coordinates for the second point of the line. |
| | laLineWidget_GetStartPoint | Gets the coordinates for the first point of the line. |
| | laLineWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laLineWidget_SetEndPoint | Sets the cooridnate for the second point of the line |
| | laLineWidget_SetStartPoint | Sets the cooridnate for the first point of the line |

## Structures

| | Name | Description |
|---|---|---|
| | laLineWidget_t | Defines the implementation of a line widget struct |
| | laLineWidget | Defines the implementation of a line widget struct |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements line draw widget functions.

## File Name

libaria_widget_line.h

## Company

Microchip Technology Inc.

### *libaria_widget_list.h*

## Enumerations

| | Name | Description |
|---|---|---|
| | laListWidget_SelectionMode_t | Defines the list selection modes |
| | laListWidget_SelectionMode | Defines the list selection modes |

## Functions

| | Name | Description |
|---|---|---|
| | laListWidget_AppendItem | Appends a new item entry to the list. The initial value of the item will be empty. |
| | laListWidget_DeselectAll | Attempts to set all item states as not selected. |
| | laListWidget_GetAlignment | Gets the horizontal alignment for the list widget |
| | laListWidget_GetAllowEmptySelection | Returns true of the list allows an empty selection set |
| | laListWidget_GetFirstSelectedItem | Returns the lowest selected item index. |
| | laListWidget_GetIconMargin | Gets the icon margin value for the list widget |
| | laListWidget_GetIconPosition | Gets the icon position for the list |
| | laListWidget_GetItemCount | Gets the number of items currently contained in the list |
| | laListWidget_GetItemIcon | Gets the pointer to the image asset for the icon for the item at the given index. |
| | laListWidget_GetItemSelected | Returns true if the item at the given index is currently selected. |
| | laListWidget_GetItemText | Gets the text value for an item in the list. |
| | laListWidget_GetLastSelectedItem | Returns the highest selected item index. |
| | laListWidget_GetSelectedItemChangedEventCallback | Gets the callback for the item selected changed event |
| | laListWidget_GetSelectionCount | Returns the number of selected items in the list. |
| | laListWidget_GetSelectionMode | Gets the selection mode for the list |

| | | |
|---|---|---|
| | laListWidget_InsertItem | Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right. |
| | laListWidget_New | Allocates memory for and initializes a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laListWidget_RemoveAllItems | Attempts to remove all items from the list. |
| | laListWidget_RemoveItem | Attempts to remove an item from the list. |
| | laListWidget_SelectAll | Attempts to set all item states to selected. |
| | laListWidget_SetAlignment | Sets the horizontal alignment mode for the list widget. |
| | laListWidget_SetAllowEmptySelection | Configures the list to allow an empty selection set. |
| | laListWidget_SetIconMargin | Sets the icon margin value for the list widget. |
| | laListWidget_SetIconPosition | Sets the icon position for the list widget |
| | laListWidget_SetItemIcon | Sets the icon pointer for a given index. |
| | laListWidget_SetItemSelected | Attempts to set the item at idx as selected. |
| | laListWidget_SetItemText | Sets the text value for an item in the list. |
| | laListWidget_SetItemVisible | |
| | laListWidget_SetSelectedItemChangedEventCallback | Sets the callback for the item selected changed event |
| | laListWidget_SetSelectionMode | Set the list selection mode |
| | laListWidget_ToggleItemSelected | Attempts to toggle the selected state of the item at idx. |

## Structures

| | Name | Description |
|---|---|---|
| | laListItem_t | Defines a list item struct |
| | laListWidget_t | Defines the implementation of a list widget |
| | laListItem | Defines a list item struct |
| | laListWidget | Defines the implementation of a list widget |

## Types

| | Name | Description |
|---|---|---|
| | laListWidget_ItemSelectedChangedEvent | This is type laListWidget_ItemSelectedChangedEvent. |
| | laListWidget_SelectedItemChangedEvent | This is type laListWidget_SelectedItemChangedEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements list box widget functions.

## File Name

libaria_widget_list.h

## Company

Microchip Technology Inc.

### *libaria_widget_listwheel.h*

## Functions

| | Name | Description |
|---|---|---|
| | laListWheelWidget_AppendItem | Appends a new item entry to the list. The initial value of the item will be empty. |
| | laListWheelWidget_GetAlignment | Gets the horizontal alignment for the list widget |
| | laListWheelWidget_GetFlickInitSpeed | Returns the flick init speed for the wheel. |
| | laListWheelWidget_GetIconMargin | Gets the icon margin value for the list wheel widget |
| | laListWheelWidget_GetIconPosition | Sets the icon position for the list wheel widget. |
| | laListWheelWidget_GetIndicatorArea | Returns the spacing for the selected item indicator bars. |
| | laListWheelWidget_GetItemCount | Gets the number of items currently contained in the list |
| | laListWheelWidget_GetItemIcon | Gets the pointer to the image asset for the icon for the item at the given index. |
| | laListWheelWidget_GetItemText | Gets the text value for an item in the list. |

| | | |
|---|---|---|
| | laListWheelWidget_GetMaxMomentum | Returns the maximum momentum value for the wheel. |
| | laListWheelWidget_GetMomentumFalloffRate | Returns the momentum falloff rate for the wheel. |
| | laListWheelWidget_GetRotationUpdateRate | Returns the wheel rotation update rate. |
| | laListWheelWidget_GetSelectedItem | Returns the index of the currently selected item. |
| | laListWheelWidget_GetSelectedItemChangedEventCallback | Gets the callback for the item selected changed event |
| | laListWheelWidget_GetShaded | Returns true if the list is using gradient shading to illustrate depth |
| | laListWheelWidget_GetShowIndicators | Returns true if the list is displaying its selected item indicators |
| | laListWheelWidget_GetVisibleItemCount | Returns the list's visible item count |
| | laListWheelWidget_InsertItem | Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right. |
| | laListWheelWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laListWheelWidget_RemoveAllItems | Attempts to remove all items from the list. |
| | laListWheelWidget_RemoveItem | Attempts to remove an item from the list. |
| | laListWheelWidget_SelectNextItem | Attempts to move the selected item index to the next item in the list. |
| | laListWheelWidget_SelectPreviousItem | Attempts to move the selected item index to the previous item in the list. |
| | laListWheelWidget_SetAlignment | Sets the horizontal alignment mode for the list widget. |
| | laListWheelWidget_SetFlickInitSpeed | Configures the flick init speed for the list wheel |
| | laListWheelWidget_SetIconMargin | Sets the icon margin value for the list widget. |
| | laListWheelWidget_SetIconPosition | Sets the icon position for the list wheel widget |
| | laListWheelWidget_SetIndicatorArea | Configures the display area for the list selection indicator bars |
| | laListWheelWidget_SetItemIcon | Sets the icon pointer for a given index. |
| | laListWheelWidget_SetItemText | Sets the text value for an item in the list. |
| | laListWheelWidget_SetMaxMomentum | Configures the maximum momentum value for the wheel |
| | laListWheelWidget_SetMomentumFalloffRate | Configures the momentum falloff rate for the wheel |
| | laListWheelWidget_SetRotationUpdateRate | Configures the rotation update rate for a wheel |
| | laListWheelWidget_SetSelectedItem | Attempts to set the selectedi tem index |
| | laListWheelWidget_SetSelectedItemChangedEventCallback | |
| | laListWheelWidget_SetShaded | Configures the list to use gradient or flat background shading |
| | laListWheelWidget_SetShowIndicators | Configures the list to display the selected item indicator bars |
| | laListWheelWidget_SetVisibleItemCount | Sets the number of visible items in the list. Must be greater than or equal to three and must be an odd number. |

## Structures

| | Name | Description |
|---|---|---|
| | laListWheelItem_t | Implementation of a list wheel widget item struct |
| | laListWheelWidget_t | Implementation of a list wheel widget struct |
| | laListWheelItem | Implementation of a list wheel widget item struct |
| | laListWheelWidget | Implementation of a list wheel widget struct |

## Types

| | Name | Description |
|---|---|---|
| | laListWheelWidget_SelectedItemChangedEvent | This is type laListWheelWidget_SelectedItemChangedEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements list wheel (drawing-style list box) widget functions.

## File Name

libaria_widget_listwheel.h

## Company

Microchip Technology Inc.

## *libaria_widget_progressbar.h*

### Enumerations

| | Name | Description |
|---|---|---|
| | laProgressBarDirection_t | Defines the valid values for the progress bar widget fill directions. |
| | laProgressBarDirection | Defines the valid values for the progress bar widget fill directions. |

### Functions

| | Name | Description |
|---|---|---|
| | laProgressBarWidget_GetDirection | Gets the fill direction value for a progress bar widget |
| | laProgressBarWidget_GetValue | Gets the percentage value for a progress bar. |
| | laProgressBarWidget_GetValueChangedEventCallback | Gets the currently set value changed event callback. |
| | laProgressBarWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laProgressBarWidget_SetDirection | Sets the fill direction for a progress bar widget |
| | laProgressBarWidget_SetValue | Sets the percentage value for a progress bar. Valid values are 0 - 100. |
| | laProgressBarWidget_SetValueChangedCallback | Sets the desired value changed event callback pointer |

### Structures

| | Name | Description |
|---|---|---|
| | laProgressBarWidget_t | Implementation of a progressbar widget struct |
| | laProgressBarWidget | Implementation of a progressbar widget struct |

### Types

| | Name | Description |
|---|---|---|
| | laProgressBar | This is type laProgressBar. |
| | laProgressBar_ValueChangedEventCallback | This is type laProgressBar_ValueChangedEventCallback. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements progress bar widget functions.

### File Name

libaria_widget_progressbar.h

### Company

Microchip Technology Inc.

## *libaria_widget_radiobutton.h*

### Functions

| | Name | Description |
|---|---|---|
| | laRadioButtonWidget_GetDeselectedEventCallback | Gets the current radio button deselected event callback |
| | laRadioButtonWidget_GetGroup | Returns the pointer to the currently set radio button group. |
| | laRadioButtonWidget_GetHAlignment | Gets the horizontal alignment setting for a button |
| | laRadioButtonWidget_GetImageMargin | Gets the distance between the icon and the text |
| | laRadioButtonWidget_GetImagePosition | Gets the current image position setting for the radio button |
| | laRadioButtonWidget_GetSelected | Returns true if this radio button is currently selected |
| | laRadioButtonWidget_GetSelectedEventCallback | Gets the current radio button selected event callback |
| | laRadioButtonWidget_GetSelectedImage | Gets the selected image asset pointer for a button |
| | laRadioButtonWidget_GetText | Gets the text value for the button. |
| | laRadioButtonWidget_GetUnselectedImage | Gets the image asset pointer currently used as the unselected icon |
| | laRadioButtonWidget_GetVAlignment | Sets the vertical alignment for a button |
| | laRadioButtonWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |

| | | |
|---|---|---|
| | laRadioButtonWidget_SetDeselectedEventCallback | Sets the deselected callback pointer |
| | laRadioButtonWidget_SetHAlignment | Sets the horizontal alignment value for a button |
| | laRadioButtonWidget_SetImageMargin | Sets the distance between the icon and text |
| | laRadioButtonWidget_SetImagePosition | Sets the image relative position setting for the radio button |
| | laRadioButtonWidget_SetSelected | Sets this button as selected. |
| | laRadioButtonWidget_SetSelectedEventCallback | Sets the radio button selected event callback |
| | laRadioButtonWidget_SetSelectedImage | Sets the image to be used as a selected icon |
| | laRadioButtonWidget_SetText | Sets the text value for the button. |
| | laRadioButtonWidget_SetUnselectedImage | Sets the asset pointer for the radio button's unselected image icon |
| | laRadioButtonWidget_SetVAlignment | Sets the vertical alignment for a button |

## Structures

| | Name | Description |
|---|---|---|
| | laRadioButtonWidget_t | Implementation of a radio button widget struct |
| | laRadioButtonWidget | Implementation of a radio button widget struct |

## Types

| | Name | Description |
|---|---|---|
| | laRadioButtonGroup | Defines the structure used for the Radio Button group. |
| | laRadioButtonWidget_DeselectedEvent | This is type laRadioButtonWidget_DeselectedEvent. |
| | laRadioButtonWidget_SelectedEvent | This is type laRadioButtonWidget_SelectedEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements radio button widget functions.

## File Name

libaria_widget_radiobutton.h

## Company

Microchip Technology Inc.

### *libaria_widget_rectangle.h*

## Functions

| | Name | Description |
|---|---|---|
| | laRectangleWidget_GetThickness | Gets the rectangle border thickness setting |
| | laRectangleWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laRectangleWidget_SetThickness | Sets the rectangle border thickness setting |

## Structures

| | Name | Description |
|---|---|---|
| | laRectangleWidget_t | Implementation of a rectangle widget struct |
| | laRectangleWidget | Implementation of a rectangle widget struct |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements rectangle drawing widget functions.

## File Name

libaria_widget_rectangle.h

## Company

Microchip Technology Inc.

## *libaria_widget_scrollbar.h*

### Enumerations

| | Name | Description |
|---|---|---|
| | laScrollBarOrientation_t | Defines the scroll bar direction values |
| | laScrollBarState_t | Defines the various scroll bar state values |
| | laScrollBarOrientation | Defines the scroll bar direction values |
| | laScrollBarState | Defines the various scroll bar state values |

### Functions

| | Name | Description |
|---|---|---|
| | laScrollBarWidget_GetExtentValue | Gets the current scroll bar extent value |
| | laScrollBarWidget_GetMaxinumValue | Gets the maximum scroll value |
| | laScrollBarWidget_GetOrientation | Gets the orientation value for the scroll bar |
| | laScrollBarWidget_GetScrollPercentage | Gets the current scroll value as a percentage |
| | laScrollBarWidget_GetScrollValue | Gets the current scroll value |
| | laScrollBarWidget_GetStepSize | Gets the current discreet step size |
| | laScrollBarWidget_GetValueChangedEventCallback | Gets the current value changed callback function pointer |
| | laScrollBarWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laScrollBarWidget_SetExtentValue | Sets the scroll bar extent value |
| | laScrollBarWidget_SetMaximumValue | Sets the maximum scroll value |
| | laScrollBarWidget_SetOrientation | Sets the orientation value of the scroll bar |
| | laScrollBarWidget_SetScrollPercentage | Sets the current scroll value using a percentage. Percentage should be a value from 0 - 100 |
| | laScrollBarWidget_SetScrollValue | Sets the current scroll value |
| | laScrollBarWidget_SetStepSize | Sets the current step size |
| | laScrollBarWidget_SetValueChangedEventCallback | Sets the value changed event callback pointer |
| | laScrollBarWidget_StepBackward | Moves the scroll value back by the current step size |
| | laScrollBarWidget_StepForward | Moves the scroll value forward by the current step size |

### Structures

| | Name | Description |
|---|---|---|
| | laScrollBarWidget_t | Implementation of a scroll bar widget. |
| | laScrollBarWidget | Implementation of a scroll bar widget. |

### Types

| | Name | Description |
|---|---|---|
| | laScrollBarWidget_ValueChangedEvent | This is type laScrollBarWidget_ValueChangedEvent. |

### Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements scroll bar widget functions.

### File Name

libaria_widget_scrollbar.h

### Company

Microchip Technology Inc.

## *libaria_widget_slider.h*

### Enumerations

| | Name | Description |
|---|---|---|
| | laSliderOrientation_t | Slider orientations |

| | | |
|---|---|---|
| | laSliderState_t | Describes various slider states |
| | laSliderOrientation | Slider orientations |
| | laSliderState | Describes various slider states |

## Functions

| | Name | Description |
|---|---|---|
| | laSliderWidget_GetGripSize | Gets the current grip size of the slider |
| | laSliderWidget_GetMaxinumValue | Gets the maximum value for the slider |
| | laSliderWidget_GetMininumValue | Gets the minimum value for the slider |
| | laSliderWidget_GetOrientation | Gets the orientation value for the slider |
| | laSliderWidget_GetSliderPercentage | Gets the slider value as a percentage |
| | laSliderWidget_GetSliderValue | Gets the current slider value |
| | laSliderWidget_GetValueChangedEventCallback | Gets the current value changed event callback pointer |
| | laSliderWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laSliderWidget_SetGripSize | Sets the grip size of the slider |
| | laSliderWidget_SetMaximumValue | Sets the maximum value for the slider |
| | laSliderWidget_SetMinimumValue | Sets the minimum value for the slider |
| | laSliderWidget_SetOrientation | |
| | laSliderWidget_SetSliderPercentage | Sets the slider value using a percentage. Value must be from 0 - 100. |
| | laSliderWidget_SetSliderValue | Sets the current slider value |
| | laSliderWidget_SetValueChangedEventCallback | Sets the value changed event callback pointer |
| | laSliderWidget_Step | Moves the slider by a given amount |

## Structures

| | Name | Description |
|---|---|---|
| | laSliderWidget_t | Implementation of a slider widget struct |
| | laSliderWidget | Implementation of a slider widget struct |

## Types

| | Name | Description |
|---|---|---|
| | laSliderWidget_ValueChangedEvent | This is type laSliderWidget_ValueChangedEvent. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements slider control widget functions.

## File Name

libaria_widget_slider.h

## Company

Microchip Technology Inc.

### *libaria_widget_textfield.h*

## Functions

| | Name | Description |
|---|---|---|
| | laTextFieldWidget_GetAlignment | Gets the text horizontal alignment value. |
| | laTextFieldWidget_GetCursorDelay | Gets the current cursor delay. |
| | laTextFieldWidget_GetCursorEnabled | Gets the cursor enabled value |
| | laTextFieldWidget_GetCursorPosition | Gets the current edit cursor position |
| | laTextFieldWidget_GetText | Gets the text value for the box. |
| | laTextFieldWidget_GetTextChangedEventCallback | Gets the current text changed event callback pointer |
| | laTextFieldWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laTextFieldWidget_SetAlignment | Sets the text horizontal alignment value |

| | | |
|---|---|---|
| | laTextFieldWidget_SetClearOnFirstEdit | Sets the flag to indicate that the text field will be cleared on first edit. |
| | laTextFieldWidget_SetCursorDelay | Sets the cursor delay value |
| | laTextFieldWidget_SetCursorEnabled | Sets the cursor enabled value flag |
| | laTextFieldWidget_SetCursorPosition | Sets the position of the cursor |
| | laTextFieldWidget_SetText | Sets the text value for the box. |
| | laTextFieldWidget_SetTextChangedEventCallback | Sets the text changed event callback pointer |

## Structures

| | Name | Description |
|---|---|---|
| | laTextFieldWidget_t | Implementation of a text field widget. |
| | laTextFieldWidget | Implementation of a text field widget. |

## Types

| | Name | Description |
|---|---|---|
| | laTextFieldWidget_TextChangedCallback | This is type laTextFieldWidget_TextChangedCallback. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements text field widget functions.

## File Name

libaria_widget_textfield.h

## Company

Microchip Technology Inc.

### *libaria_widget_touchtest.h*

## Enumerations

| | Name | Description |
|---|---|---|
| | laTouchTestState_t | Touch test states |
| | laTouchTestState | Touch test states |

## Functions

| | Name | Description |
|---|---|---|
| | laTouchTest_AddPoint | Adds a point to the touch test widget. The point will then be displayed. |
| | laTouchTest_ClearPoints | Clears all of the existing touch points |
| | laTouchTestWidget_GetPointAddedEventCallback | Gets the current point added event callback |
| | laTouchTestWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laTouchTestWidget_SetPointAddedEventCallback | Sets the point added event callback |

## Macros

| | Name | Description |
|---|---|---|
| | LA_TOUCHTEST_MEMORY_SIZE | This is macro LA_TOUCHTEST_MEMORY_SIZE. |

## Structures

| | Name | Description |
|---|---|---|
| | laTouchTestWidget_t | Implementation of a touch test widget struct |
| | laTouchTestWidget | Implementation of a touch test widget struct |

## Types

| | Name | Description |
|---|---|---|
| | laTouchTestWidget_PointAddedEventCallback | This is type laTouchTestWidget_PointAddedEventCallback. |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements graphical touch test (box) widget functions.

## File Name

libaria_widget_touchtest.h

## Company

Microchip Technology Inc.

### *libaria_widget_window.h*

Window Widget

## Functions

| | Name | Description |
|---|---|---|
| | laWindowWidget_GetIcon | Gets the currently used window icon |
| | laWindowWidget_GetIconMargin | Gets the current image icon margin |
| | laWindowWidget_GetIconRect | This is function laWindowWidget_GetIconRect. |
| | laWindowWidget_GetTextRect | This is function laWindowWidget_GetTextRect. |
| | laWindowWidget_GetTitle | Gets the title text for this window. |
| | laWindowWidget_GetTitleBarRect | internal use only |
| | laWindowWidget_New | Allocates memory for a new widget of this type. The application is responsible for the managment of this memory until the widget is added to a widget tree. |
| | laWindowWidget_SetIcon | Sets the image to be used as a window icon |
| | laWindowWidget_SetIconMargin | Sets the image icon margin |
| | laWindowWidget_SetTitle | Sets the title text for the window. |

## Structures

| | Name | Description |
|---|---|---|
| | laWindowWidget_t | Implementation of a window widget struct |
| | laWindowWidget | Implementation of a window widget struct |

## Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements window container widget functions.

## File Name

libaria_widget_window.h

## Company

Microchip Technology Inc.

### *Hardware Abstraction Layer (HAL)*

This topic describes the Hardware Abstraction Layer (HAL) of the MPLAB Harmony Graphics Composer (MHGC) Suite, which is a component of MHGC Suite.

### Introduction

This section introduces the Hardware Abstraction Layer (HAL) of the MPLAB Harmony Graphics Composer (MHGC) Suite.

## Description

The HAL serves to abstract the details of the hardware away from the application and protect the graphics state from mismanagement. This layer is designed to be similar to industry-standard graphics APIs, such as OpenGL from SGI, and DirectX from Microsoft. Applications that use graphics should only communicate with this layer at a minimum, and should not attempt to communicate with display drivers directly.

Before looking at the operation and structure of the HAL, the following definitions of the different keywords and concepts explained within the HAL are provided.

## Hardware Abstraction Layer Definitions:

- *Alpha blending*: An operation that combines two colors into a single color, based on one or more percentage values
- *Blit*: Writing an area of pixel data to a buffer

- *Buffer swap*: Cycling through a buffer chain, therefore changing the read and write buffer pointers
- *Cache coherent*: Data that must always be current in memory, such as data that is accessed by a peripheral, which should use coherent memory
- *Clipping*: Comparing a point to a rectangle, or a rectangle to a rectangle, to assess whether the point is contained inside the rectangle or conforming the area of one rectangle to fit inside another
- *Color masking*: An operation that compares a color value with a color mask value. If the values are equal, the color is ignored and not rendered to the write buffer.
- *Color mode*: Defines how pixel data is stored in memory. Some color modes consume less memory than others.
- *Context*: An instance of the hardware abstraction layer. Combines a display, a driver, and possibly a graphics accelerator.
- *Display*: A display device capable of rendering color data
- *Draw Target*: An application-defined area of memory to be used as the target for draw operations. This is often different than the active frame buffer and can be used for off-screen rendering operations.
- *Driver*: A software program that communicates directly with, and manages, hardware
- *Double buffer*: A display configuration in which multiple frame buffers are chained together to avoid screen tearing artifacts
- *Frame buffer*: An area of memory that contains pixel data. Pixel data is one of several color modes with each mode consuming various quantities of memory.
- *Heap*: A pool of memory that can be dynamically allocated
- *HSync*: A refresh state of a display device when the device is being refreshed outside the horizontal viewing area
- *Layer*: A rectangular area of space that contains one or more frame buffers. Can directly correspond to a hardware-managed layer.
- *Pixel*: A single color value stored in a predefined mode. Usually 8 to 32 bits in size.
- *Pixel buffer*: A struct that describes a rectangle of pixel data. May or may not actually contain valid pixel data.
- *Point*: A two dimensional Cartesian coordinate consisting of a horizontal “x” value and a vertical “y” value
- *Raster operation*: Any operation or algorithm that writes pixel data to the write buffer
- *Read buffer*: A buffer that is currently being used to feed display data to display hardware
- *Size*: A two dimensional measurement of magnitude consisting of a “width” and a “height” value
- *VSync*: A refresh state of a display device when the device is being refreshed outside the vertical viewing area
- *Write buffer*: A buffer that is designated as the receiver of raster operations

## HAL Features

What does the Hardware Abstraction Layer do?

The HAL serves four main purposes:

- Configure abstract graphics and display hardware
- Managing frame buffer memory
- Manage draw state
- Draw shapes

**Graphics and Display Hardware Configuration**

The HAL serves as an intermediary between the higher level stack layers and the hardware drivers. Drivers are expected to conform to the HAL specification and applications interface with drivers through a simple set of APIs. The main purpose of this is to allow the framework to use various hardware drivers without ever having to make changes to the application. Each driver will interface with the HAL according to its specific needs and capabilities.

**Frame Buffer Memory Management**

Memory management is handled by the HAL for all drivers, libraries, and applications. This may include; buffer creation, buffer resizing, freeing buffer memory, buffer swapping, etc. The application simply requests buffer functions through the HAL APIs. Drivers may restrict how buffers are managed based on specific graphics controller needs and capabilities.

## Draw State Management

The HAL maintains a state that indicates how raster operations should be performed.

## Shape Drawing

The HAL provides APIs for basic pixel, line, circle, and rectangle drawing. These are rendered according to the draw state.

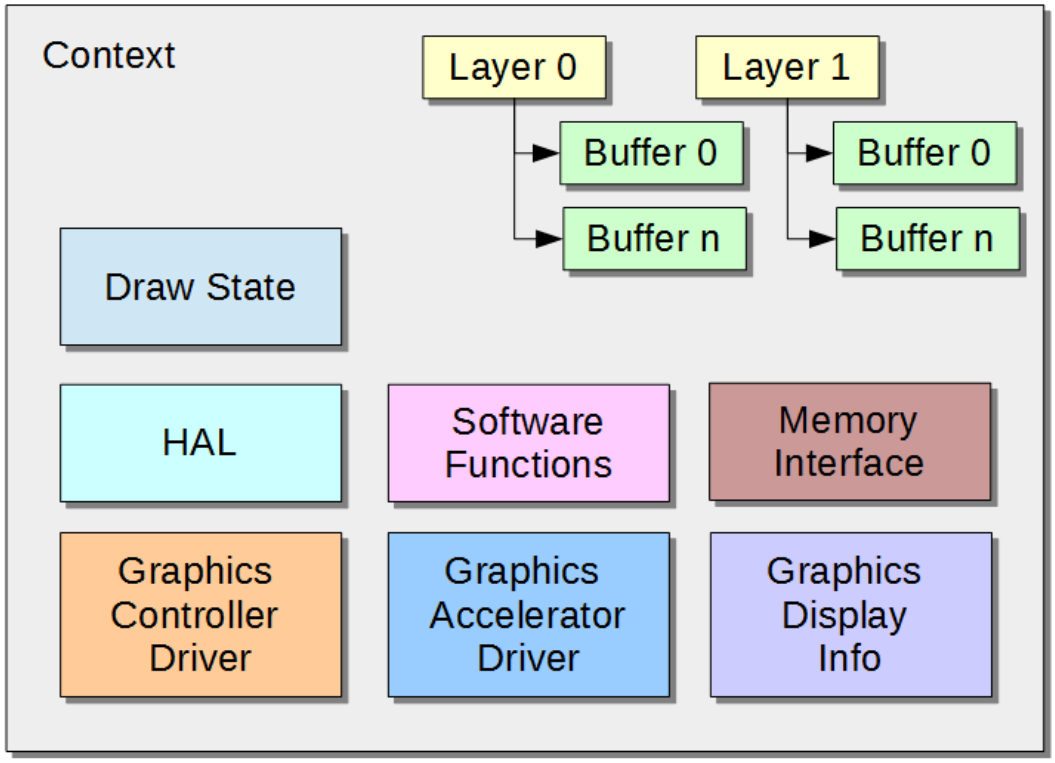
## HAL Context

What is a context?

The HAL is designed to be multi-instance, which means it supports multiple drivers and displays concurrently. This is accomplished by using a *context* or *layer* state. A context is essentially the combination of a single display descriptor, and a single graphics driver. Graphics processors may also be part of a context. The HAL allows only a single context to be *active* or in use at any one time, but the application may switch contexts at any time in order to act on another state.

**Note:** Graphics drivers must also be able to support multi-instancing in order for the application to use multiple contexts.

The following block diagram shows a high-level representation of what the context contains. A description of each block follows the diagram.




## Graphics Display Info

The application uses a “Display” definition to obtain knowledge about the displays that are available through the HAL. This definition contains information such as:

- *Name*: A short identifier for the display
- *Color Modes*: The color modes the display supports
- *Size*: A width and height
- *Timing settings*: Values for the front porches, back porches, pulse widths, etc.

Display definitions are generated through the use of HConfig and Freemarker templates. These definitions are created during the code generation phase of MHC and are expected to exist at run-time.

## Graphics Controller Driver

The application uses a “Driver” definition to obtain information from the HAL about the available drivers in the system.

The driver definition consists of the following information:

- *Name*: A short identifier for the controller
- *Color Modes*: The color modes the controller supports
- *Layer Count*: The number of hardware layers the controller supports

Driver integration with the HAL will be covered in a subsequent section.

## Graphics Accelerator Driver

A Graphics Processing Unit (GPU) may be present in the system. If one exists, the context must reroute GPU supported raster operations to the graphics accelerator driver for handling.

## Layer

A HAL layer is a representation of a hardware based display layer most likely provided by a graphics controller. Applications are capable of using one or more of these layers up to the max value indicated by a hardware driver. Layers have a width, height, and a position in absolute space on the display.

Layers may have one or more frame buffers associated with them. Layers with two buffers are often called *double buffered*. Multiple buffers of a layer are connected to form a buffer chain, and are cycled through as needed via pointer swapping. Layers have, at all times, one buffer considered to be the *read buffer* and one buffer considered to be the *write buffer*. In a single buffer layer, the read and write buffers are the same. When drawing single buffer layers, rendering artifacts such as screen tearing may be observed. This is because a single buffer may be written to, and read from at the same time. Double buffering alleviates screen tearing as all raster operations are performed on the hidden *write buffer* and the buffers are only swapped once the *write buffer* has been fully crafted. Further, by acting during display blanking periods, the driver can swap the read and write buffer pointers during periods when the display is not actively drawing. This should completely eliminate screen tearing.

A context has one active layer at all times and all operations are performed on the active layer.

## Frame Buffer

An extension of a pixel buffer, frame buffers are used by layers to track frame buffer states. Frame buffers contain a pixel buffer, but also contain the following:

- *Pixel Buffer State*: An indication of the origin of the data for the pixel buffer. This can indicate that the buffer contains no pixel data, that the pixel data was allocated from the heap, or that the buffer and associated pixel data is owned and managed by the graphics driver. The latter state is used to prevent the application from freeing buffers managed by the graphic driver.
- *Coherent*: An indication that the buffer should be allocated from cache coherent memory when it is dynamically allocated

## Memory Interface

By default, the HAL uses standard library memory management functions, such as malloc, free, calloc, etc. However, in the presence of memory peripherals, the application may want the Graphics Stack to utilize a custom memory manager instead. This is accomplished by providing a memory interface definition.

This definition simply provides alternate function pointers for standard memory allocation functions.

## Draw State

The context's draw state is simply a list of hints that the context feeds into raster operation functions such as a line draw. The state indicates what the draw color is, if alpha blending is enabled, if the final raster point should be adjusted for orientation or mirroring, if there is a masking or transparency color enabled, etc.

## HAL

One of the most important functions of the context is to provide hardware abstraction. By default, all raster operations are handled in software, or in the Software Functions module. However, if a GPU exists, any supported raster operation requests must be rerouted to the GPU driver for handling. In other cases, the driver may need to restrict context options or handle an operation in a manner that is different from the default implementation. Therefore, the driver may change the function routing in the context's HAL state as it sees fit. However, if the driver implements non-default functionality, it must ensure that overall functionality of the context is not compromised.

## Software Functions

The HAL contains a series of default implementation functions for most operations. These are represented by the Software Functions module.

## Color Support

The HAL is able to create and manage a context using one of several color formats.

- GS8: 8-bit gray scale
- RGB_332: 8-bit, 256 colors
- RGB_565: 16-bit, 65536 colors
- RGB_5551: 15-bit color, 1-bit alpha, 32767 colors
- RGB_888: 24-bit color, 16 million colors
- RGBA_8888: 24-bit color, 8-bit alpha, 16 million colors
- ARGB_8888: 24-bit color 8-bit alpha, 16 million colors

All buffers that are created by the context will use this color mode. This can affect the sizes of the frame buffers that will be created.

## HAL State Management

The HAL is primarily interacted with through the GFX_Get and GFX_Set functions. These variable argument functions always take as the first argument an operation ID. Then, follow a variable number of supporting arguments to either set or get data. For example:

```
GFX_Set(GFXF_DRAW_COLOR, 0xFFFF);
```

This code would set the current draw color for the active context to white, assuming a 16-bit color space. The first argument is one of the values listed in the GFX_FLAG enum and the second is the argument expected by that operation.

To get the current draw color the code would appear as follows:

```
GFX_Get(GFXF_DRAW_COLOR, &color);
```

These get and set functions can return these status values:

- GFX_FAILURE: An error occurred during this operation
- GFX_SUCCESS: The operation was successful
- GFX_UNSUPPORTED: The operation is not supported by the context

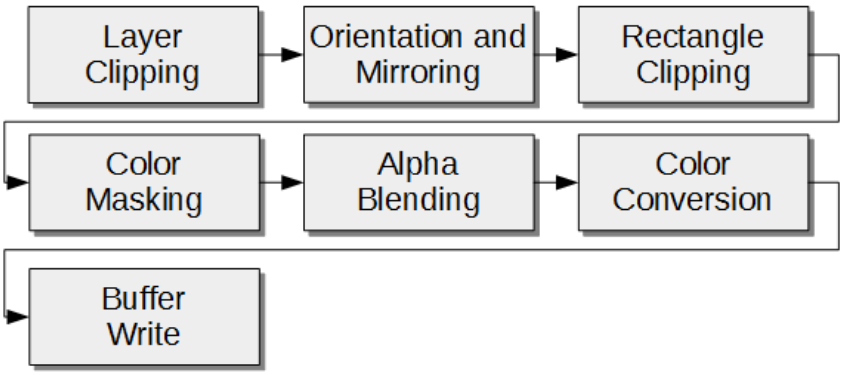
**Notes:**

1. All features may not be supported by all drivers. Software fall backs and default implementations may be provided for some features when hardware support is not available.
2. Most flags have a get and set mode. A few are get only and a few are set only. For detailed interface information, refer to `gfx_common.h`.

## Pixel Transformation Pipeline

Overview of the Pixel Transformation Pipeline.

The HAL uses a multi-stage pixel rendering pipeline to apply the various effects that may have been enabled by the application. The stages are shown in the following figure:




## Stage Description

- *Layer Clipping*: The pixel is clipped to the destination layer. It is rejected if it falls outside the layer. Writing outside of the layer can cause memory out-of-bounds exceptions.
- *Orientation and Mirroring*: The pixel's destination point is rotated and mirrored according to the draw state
- *Rectangle Clipping*: The pixel is containment evaluated with the context's clipping rectangle and rejected if it is out of bounds
- *Color Masking*: The pixel is compared to the context's color mask value. If the color matches the value it is rejected
- *Alpha Blending*: The source pixel is blended with the destination pixel. Both the color's alpha channel and the global alpha blending value is taken into consideration. A color without an alpha channel is upscaled to 32-bits and its alpha channel is set to max.
- *Color Conversion*: The color is converted to the color mode of the destination buffer. This only applies to blits.
- *Buffer Write*: The result color is written to the frame buffer at the potentially transformed point

**Note:**

1. These stages can be disabled in the GFX Options in the MHC option tree. Disabling them can increase speed but can cause the program to become unstable or draw incorrectly.
2. This flow is meant to show how the stages might interact but the exact order of execution is dependent on the state of the HAL and the operation being performed.

## Using The Library

To access the HAL, simply include the header file `gfx.h` in your application. This is assuming that the appropriate flags have been checked in the configuration.

The HAL APIs typically fall into one of several groups:

- *Initialization*: Interfaces in the MHC configuration that are responsible for setting up the state of the HAL
- *Context Management*: Interfaces that create or destroy a graphics context
- *Context Maintenance*: Interfaces that allow the context to perform tasks such as HAL or driver state updates
- *Draw State Management*: This consists of two generic interfaces that allow the application to manage the state of a context. This is accomplished by indicating the get/set operation from a predefined list of option IDs, and sending the appropriate arguments into the variable argument functions
- *Blitting and Shape Drawing*: These interfaces perform raster operations on the active frame buffer according to the current draw state of the HAL

The following sample code displays how to initialize the HAL, create a context, create some layers and buffers, and draw a rectangle.

**Note:** The following code example is not performing any return value checking.

```
// context variable
GFX_Handle* context;

// initialize the HAL layer
GFX_Initialize();

// create a context.  the zeros indicate the display and driver to
// use.  the third argument would be for a custom memory interface
context = GFX_Open(0, 0, NULL);

// make sure the context is active
```

```
GFX_ContextActiveSet(context);

// set the context color mode to RGB_565
GFX_Set(GFXF_COLOR_MODE, GFX_COLOR_MODE_RGB_565);

// make sure the zeroth layer is active, enabled and visible
GFX_Set(GFXF_LAYER_ACTIVE, 0);
GFX_Set(GFXF_LAYER_ENABLED, GFX_TRUE);
GFX_Set(GFXF_LAYER_VISIBLE, GFX_TRUE);

// typically the bottom layer is going to fill the entire
// display area but for demonstration purposes change
// the position and size of the layer
GFX_Set(GFXF_LAYER_POSITION, 100, 100); // x = 100, y = 100
GFX_Set(GFXF_LAYER_SIZE, 320, 200); // width = 320, height = 200

// set the layer to two buffers and set to use coherent memory
GFX_Set(GFXF_LAYER_BUFFER_COUNT, 2);
GFX_Set(GFXF_LAYER_BUFFER_COHERENT, 0, GFX_TRUE);
GFX_Set(GFXF_LAYER_BUFFER_COHERENT, 1, GFX_TRUE);

// allocate the buffers
GFX_Set(GFXF_LAYER_BUFFER_ALLOCATE, 0);
GFX_Set(GFXF_LAYER_BUFFER_ALLOCATE, 1);

// set the draw mode and color
GFX_Set(GFXF_DRAW_MODE, GFX_DRAW_FILL);
GFX_Set(GFXF_DRAW_COLOR, 0xFFFF);

// indicate intent to draw, if this returns GFX_FAILURE then
// draw operations will fail
GFX_Begin();

// fill the entire layer with white
GFX_DrawRect(0, 0, 320, 200); // x, y, width, height

GFX_Set(GFXF_DRAW_COLOR,
        GFX_ColorValue(GFX_COLOR_MODE_RGB_565, GFX_COLOR_MAGENTA));

// draw a smaller magenta rectangle
GFX_DrawRect(10, 10, 100, 100);

// finish drawing
GFX_End();

// swap the buffers
GFX_Set(GFXF_LAYER_SWAP, GFX_TRUE);
```

## Library Interface

### a) Functions

| | Name | Description |
|---|---|---|
| | GFX_AbsoluteValue | Calculates the absolute value of a signed integer. |
| | GFX_ActiveContext | Gets the current set active HAL context. |
| | GFX_Clampf | Clamps a float between a min and max |
| | GFX_Clampi | Clamps an integer between a min and max |
| | GFX_ColorChannelAlpha | Used for getting the alpha color channel of a given color value. |
| | GFX_ColorChannelGreen | Used for getting the green color channel of a given color value. |
| | GFX_ColorChannelRed | Used for getting the red color channel of a given color value. |
| | GFX_ColorConvert | Converts a color value from one mode to another |
| | GFX_ColorLerp | Linear interpolation between two colors |
| | GFX_ColorModeInfoGet | |
| | GFX_ColorValue | Used for getting a color value by name. |
| | GFX_ContextActiveSet | Sets the active context |

| | | |
|---|---|---|
| | GFX_LayerReadBuffer | Gets the pointer to the layer's current read pixel buffer. |
| | GFX_LayerRotate | Swaps the width and height dimensions of a layer. Can be used for run-time display orientation |
| | GFX_LayerSwap | Performs a swap operation on the given layer. This advances the pointers of layer's buffer chain. The current write buffer becomes the new read buffer and a new buffer is chosen as the new write buffer. Has no effect in single buffer environments. |
| | GFX_LayerWriteBuffer | Gets the pointer to the layer's current write pixel buffer. |
| | GFX_Lerp | Performs a linear interpolation of an integer based on a percentage between two signed points. |
| | GFX_Maxf | Returns the larger of two floats. |
| | GFX_Maxi | Returns the larger of two integers. |
| | GFX_Minf | Returns the smaller of two floats. |
| | GFX_Mini | Returns the smaller of two integers. |
| | GFX_Percent | Calculates the percentage of one number when applied to another. Integer based. Accuracy for higher numbers is not guaranteed.<br>The result is the decimal percentage multiplied by 100. |
| | GFX_PercentOf | Calculates the percentage of a number. Returns a whole number with no decimal component. |
| | GFX_PercentWholeRounded | Calculates the percentage of one number when applied to another. Integer based. Accuracy for higher numbers is not guaranteed.<br>The difference between this and GFX_Percent is that the decimal portion of the whole number is rounded off. |
| | GFX_PixelBufferAreaFill | Fills an area of a pixel buffer with a solid color. Caller is responsible for ensuring that the color is the same color format as the destination buffer. |
| | GFX_PixelBufferAreaFill_Unsafe | Fills an area of a pixel buffer with a solid color. Caller is responsible for ensuring that the color is the same color format as the destination buffer. Like GFX_PixelBufferAreaFill but performs no bounds checking. |
| | GFX_PixelBufferAreaGet | Extracts a rectanglar section of pixels from a pixel buffer. |
| | GFX_PixelBufferAreaGet_Unsafe | Extracts a rectanglar section of pixels from a pixel buffer. Like GFX_PixelBufferAreaGet but performs no clipping between the rectangles of the extract area and the source buffer. |
| | GFX_PixelBufferAreaSet | Copies an area of pixels from a source buffer to a destination buffer. If the source buffer format does not match the destination format the data will be converted to match during the copy operation. |
| | GFX_PixelBufferAreaSet_Unsafe | Copies an area of pixels from a source buffer to a destination buffer. If the source buffer format does not match the destination format the data will be converted to match during the copy operation. Like GFX_PixelBufferAreaSet but performs no bounds checking. |
| | GFX_PixelBufferClipRect | Clips a rectangle against a pixel buffer. The result is guaranteed to fit inside the buffer's area. |
| | GFX_PixelBufferConvert | Duplicates a pixel buffer and converts the copy to another color mode. |
| | GFX_PixelBufferCopy | Creates a copy of the input buffer. If the input buffer's pixel data is not null then the data will be duplicated for the result buffer. |
| | GFX_PixelBufferCreate | Initializes a pixel buffer struct. Does not actually allocate any memory. |
| | GFX_PixelBufferDestroy | Destroys a pixel buffer construct. If the buffer's pixels pointer is not zero this function will attempt to free it using the provided GFX_MemoryIntf memory interface. |
| | GFX_PixelBufferGet | Gets the value of the pixel that resides at the provided point in the given buffer. |
| | GFX_PixelBufferGet_Unsafe | Gets the value of the pixel that resides at the provided point in the given buffer. Like GFX_PixelBufferGet but performs no bounds checking. |
| | GFX_PixelBufferGetIndex | Interprets the pixel buffer as a table of indices and looks up a specific index at position 'idx'. Indices may be 1bpp, 4bpp, or 8bpp in size and are indicated by the color mode of the pixel buffer. |
| | GFX_PixelBufferOffsetGet | Gets the offset address of the pixel that resides at the provided point in the given buffer. |
| | GFX_PixelBufferOffsetGet_Unsafe | Gets the offset address of the pixel that resides at the provided point in the given buffer. Similar to GFX_PixelBufferOffsetGet but performs no bounds checking. |
| | GFX_PixelBufferSet | Sets a pixel in a pixel buffer at a point to a specified color. Caller is responsible for ensuring that the input color is in the same color format as the pixel buffer. |
| | GFX_PixelBufferSet_Unsafe | Sets a pixel in a pixel buffer at a point to a specified color. Caller is responsible for ensuring that the input color is in the same color format as the pixel buffer. Like GFX_PixelBufferSet but performs no bounds checking. |
| | GFX_RectClip | Clips a rectangle to the space of another rectangle. The result rectangle is a rectangle that will fit inside both of the given rectangles. |
| | GFX_RectContainsPoint | Determines if a point is inside a rectangle. |
| | GFX_RectContainsRect | Determines if a rectangle is completely inside another rectangle. Still returns true if the edges are touching. |

| | | |
|---|---|---|
| | GFX_RectIntersects | Determines if two rectangles are intersecting |
| | GFX_ScaleInteger | Scales an integer from one number range of 0 -> n0 to another range 0 -> n1 based on percentages. |
| | createDefaultMemIntf | internal use only |
| | GFX_ColorBilerp | Calculates bilinear interpolation between four colors |
| | GFX_ColorBlend_RGBA_8888 | Blends two RGBA8888 colors together using their alpha channel values. |
| | GFX_ColorChannelBlue | Used for getting the blue color channel of a given color value. |
| | GFX_DivideRounding | This is function GFX_DivideRounding. |
| | GFX_DrawBlit | Blits a buffer of pixels into the frame buffer. |
| | GFX_DrawCircle | Draws a circle from using the specified dimensions and the current draw state. |
| | GFX_DrawLine | Draws a line from (x1,y1) to (x2,y2) using the current draw state. |
| | GFX_DrawPixel | Sets the pixel at X and Y using the current draw state. |
| | GFX_DrawRect | Draws a rectangle using the specified dimensions and the current draw state. |
| | GFX_DrawStretchBlit | Blits a buffer of pixels into the frame buffer. |
| | GFX_LayerFromOrientedSpace | Transforms a layer oriented space to screen space. |
| | GFX_LayerPointFromOrientedSpace | Transforms a point from oriented space to screen space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerPointToOrientedSpace | Transforms a point from screen space to oriented space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerRectFromOrientedSpace | Transforms a layer point from oriented space to screen space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerRectToOrientedSpace | Transforms a rectangle from screen space to oriented space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerToOrientedSpace | Transforms a layer from screen space to oriented space. |
| | GFX_PercentOfDec | This is function GFX_PercentOfDec. |
| | GFX_RectClipAdj | This is function GFX_RectClipAdj. |
| | GFX_RectFromPoints | This is function GFX_RectFromPoints. |
| | GFX_RectSplit | This is function GFX_RectSplit. |
| | GFX_RectToPoints | This is function GFX_RectToPoints. |
| | GFX_UtilMirrorPoint | Reorients a point to a given mirrored orientation. |
| | GFX_UtilOrientPoint | Reorients a point to a given orthogonal orientation. |
| | GFX_UtilPointFromOrientedSpace | Transforms a point from an oriented rectangle space to an outer space given a display orientation and a mirroring setting. |
| | GFX_UtilPointToOrientedSpace | Transforms a point to an oriented rectangle space to an outer space given a display orientation and a mirroring setting. |
| | GFX_UtilSizeFromOrientedSpace | Transforms a size tuple from oriented space to screen space |
| | GFX_UtilSizeToOrientedSpace | Transforms a size tuple from screen space to oriented space |
| | GFX_UtilSortPointsX | Sorts two points in the X axis |
| | GFX_UtilSortPointsY | Sorts two points in the Y axis |
| | GFX_RectCombine | Combines the area of two rectangles into a single rectangle. |
| | GFX_DrawDirectBlit | Blits a buffer of pixels into the frame buffer without performing per-pixel operations on the data. |
| | GFX_RectCompare | This is function GFX_RectCompare. |
| | GFX_RectsAreSimilar | This is function GFX_RectsAreSimilar. |
| | GFX_ScaleIntegerSigned | Scales a signed integer from one number range of 0 -> n0 to another range 0 -> n1 based on percentages. |

## b) Data Types and Constants

| | Name | Description |
|---|---|---|
| | GFX_AntialiasMode_t | Enables anti-aliased drawing hint |
| | GFX_BitsPerPixel_t | List of available bits-per-pixel sizes. |
| | GFX_BlendMode_t | Blend mode masks |
| | GFX_BufferSelection_t | Buffer selector used when querying layers for certain buffer states. |
| | GFX_BufferState_t | Frame buffer memory states |
| | GFX_ColorMask_t | Maskable list of color valies. |
| | GFX_ColorMode_t | List of available color modes. |
| | GFX_ColorModeInfo_t | Struct that provides information about a color mode. |

| | | |
|---|---|---|
| | GFX_ColorName_t | Color name reference table |
| | GFX_Context_t | An instance of the hardware abstraction layer. |
| | GFX_DisplayInfo_t | Describes a graphical display device. |
| | GFX_DrawMode_t | Gradient draw modes. |
| | GFX_DrawState_t | A list of drawing hints for shape drawing algorithms |
| | GFX_DriverInfo_t | A driver description structure. |
| | GFX_Flag_t | Hardware abstraction state interface flags. See gfx.h for a comprehensive description. |
| | GFX_FrameBuffer_t | A frame buffer is a wrapper around a pixel buffer construct that is used by display drivers to manage frame buffers. |
| | GFX_HAL_t | Hardware Abstraction Function Table *<br>• This is the core hardware abtraction table that makes everything work. Drivers are<br>• expected to reroute the functionality of this table to hardware specific implementations<br>• to provide accelerated performance and features. |
| | GFX_Layer_t | Layers describe basic display drawing areas and are meant to map directly to graphics controller hardware layers. |
| | GFX_MemoryIntf_t | Custom memory manager interface. |
| | GFX_Orientation_t | Orthogonal orientation settings. |
| | GFX_PixelBuffer_t | A pixel buffer is a wrapper around a basic data pointer. A pixel buffer has a color mode, a pixel count, a rectangular dimension, a pixel count, and a lenght in bytes. |
| | GFX_Point_t | A two dimensional Cartesian point. |
| | GFX_Rect_t | A rectangle definition. |
| | GFX_Size_t | A two dimensional indication of size. Values are signed but should never be negative. |
| | begin_FnPtr | This is type begin_FnPtr. |
| | boolGet_FnPtr | This is type boolGet_FnPtr. |
| | boolSet_FnPtr | This is type boolSet_FnPtr. |
| | brightnessGet_FnPtr | This is type brightnessGet_FnPtr. |
| | brightnessRangeGet_FnPtr | This is type brightnessRangeGet_FnPtr. |
| | brightnessSet_FnPtr | This is type brightnessSet_FnPtr. |
| | colorModeGet_FnPtr | This is type colorModeGet_FnPtr. |
| | colorModeSet_FnPtr | This is type colorModeSet_FnPtr. |
| | destroy_FnPtr | This is type destroy_FnPtr. |
| | drawAlphaValueGet_FnPtr | This is type drawAlphaValueGet_FnPtr. |
| | drawAlphaValueSet_FnPtr | This is type drawAlphaValueSet_FnPtr. |
| | drawBlendModeGet_FnPtr | This is type drawBlendModeGet_FnPtr. |
| | drawBlendModeSet_FnPtr | This is type drawBlendModeSet_FnPtr. |
| | drawBlit_FnPtr | This is type drawBlit_FnPtr. |
| | drawCircle_FnPtr | This is type drawCircle_FnPtr. |
| | drawClipRectGet_FnPtr | This is type drawClipRectGet_FnPtr. |
| | drawClipRectSet_FnPtr | This is type drawClipRectSet_FnPtr. |
| | drawColorGet_FnPtr | This is type drawColorGet_FnPtr. |
| | drawColorSet_FnPtr | This is type drawColorSet_FnPtr. |
| | drawGradientColorGet_FnPtr | This is type drawGradientColorGet_FnPtr. |
| | drawGradientColorSet_FnPtr | This is type drawGradientColorSet_FnPtr. |
| | drawLine_FnPtr | This is type drawLine_FnPtr. |
| | drawLock_FnPtr | This is type drawLock_FnPtr. |
| | drawMaskValueGet_FnPtr | This is type drawMaskValueGet_FnPtr. |
| | drawMaskValueSet_FnPtr | This is type drawMaskValueSet_FnPtr. |
| | drawModeGet_FnPtr | This is type drawModeGet_FnPtr. |
| | drawModeSet_FnPtr | This is type drawModeSet_FnPtr. |
| | drawPaletteGet_FnPtr | This is type drawPaletteGet_FnPtr. |
| | drawPaletteSet_FnPtr | This is type drawPaletteSet_FnPtr. |
| | drawPixel_FnPtr | This is type drawPixel_FnPtr. |
| | drawRect_FnPtr | This is type drawRect_FnPtr. |
| | drawThicknessGet_FnPtr | This is type drawThicknessGet_FnPtr. |
| | drawThicknessSet_FnPtr | This is type drawThicknessSet_FnPtr. |
| | drawUnlock_FnPtr | This is type drawUnlock_FnPtr. |

| | | |
|---|---|---|
| | GFX_AntialiasMode | Enables anti-aliased drawing hint |
| | GFX_BitsPerPixel | List of available bits-per-pixel sizes. |
| | GFX_BlendMode | Blend mode masks |
| | GFX_Bool | This is type GFX_Bool. |
| | GFX_Buffer | This is type GFX_Buffer. |
| | GFX_BufferSelection | Buffer selector used when querying layers for certain buffer states. |
| | GFX_BufferState | Frame buffer memory states |
| | GFX_Calloc_FnPtr | Simple wrapper around the standard calloc function pointer. Used for redirecting memory allocation to other memory management systems. |
| | GFX_Color | This is type GFX_Color. |
| | GFX_ColorMask | Maskable list of color valies. |
| | GFX_ColorMode | List of available color modes. |
| | GFX_ColorModeInfo | Struct that provides information about a color mode. |
| | GFX_ColorName | Color name reference table |
| | GFX_Context | An instance of the hardware abstraction layer. |
| | GFX_Display | This is type GFX_Display. |
| | GFX_DisplayInfo | Describes a graphical display device. |
| | GFX_DrawMode | Gradient draw modes. |
| | GFX_DrawState | A list of drawing hints for shape drawing algorithms |
| | GFX_Driver | This is type GFX_Driver. |
| | GFX_DriverInfo | A driver description structure. |
| | GFX_Flag | Hardware abstraction state interface flags. See gfx.h for a comprehensive description. |
| | GFX_FrameBuffer | A frame buffer is a wrapper around a pixel buffer construct that is used by display drivers to manage frame buffers. |
| | GFX_Free_FnPtr | Simple wrapper around the standard free function pointer. Used for redirecting memory free to other memory management systems. |
| | GFX_HAL | Hardware Abstraction Function Table *<br>• This is the core hardware abtraction table that makes everything work. Drivers are<br>• expected to reroute the functionality of this table to hardware specific implementations<br>• to provide accelerated performance and features. |
| | GFX_Handle | This is type GFX_Handle. |
| | GFX_Layer | Layers describe basic display drawing areas and are meant to map directly to graphics controller hardware layers. |
| | GFX_Malloc_FnPtr | Simple wrapper around the standard malloc function pointer. Used for redirecting memory allocation to other memory management systems. |
| | GFX_Memcpy_FnPtr | Simple wrapper around the standard memcpy function pointer. Used for redirecting memcpy to other memory management systems. |
| | GFX_MemoryIntf | Custom memory manager interface. |
| | GFX_Memset_FnPtr | Simple wrapper around the standard memset function pointer. Used for redirecting memset to other memory management systems. |
| | GFX_Orientation | Orthogonal orientation settings. |
| | GFX_PixelBuffer | A pixel buffer is a wrapper around a basic data pointer. A pixel buffer has a color mode, a pixel count, a rectangular dimension, a pixel count, and a lenght in bytes. |
| | end_FnPtr | This is type end_FnPtr. |
| | GFX_Processor | This is type GFX_Processor. |
| | GFX_Realloc_FnPtr | Simple wrapper around the standard realloc function pointer. Used for redirecting memory allocation to other memory management systems. |
| | GFX_Result | This is type GFX_Result. |
| | GFX_Size | A two dimensional indication of size. Values are signed but should never be negative. |
| | GFX_SyncCallback_FnPtr | This is type GFX_SyncCallback_FnPtr. |
| | layerActiveGet_FnPtr | This is type layerActiveGet_FnPtr. |
| | layerActiveSet_FnPtr | This is type layerActiveSet_FnPtr. |
| | layerAlphaAmountGet_FnPtr | This is type layerAlphaAmountGet_FnPtr. |
| | layerAlphaAmountSet_FnPtr | This is type layerAlphaAmountSet_FnPtr. |
| | layerBufferAddressGet_FnPtr | This is type layerBufferAddressGet_FnPtr. |
| | layerBufferAddressSet_FnPtr | This is type layerBufferAddressSet_FnPtr. |
| | layerBufferAllocate_FnPtr | This is type layerBufferAllocate_FnPtr. |

| | | |
|---|---|---|
| | layerBufferCoherentGet_FnPtr | This is type layerBufferCoherentGet_FnPtr. |
| | layerBufferCoherentSet_FnPtr | This is type layerBufferCoherentSet_FnPtr. |
| | layerBufferCountGet_FnPtr | This is type layerBufferCountGet_FnPtr. |
| | layerBufferCountSet_FnPtr | This is type layerBufferCountSet_FnPtr. |
| | layerBufferFree_FnPtr | This is type layerBufferFree_FnPtr. |
| | layerBufferIsAllocated_FnPtr | This is type layerBufferIsAllocated_FnPtr. |
| | layerMaskColorGet_FnPtr | This is type layerMaskColorGet_FnPtr. |
| | layerMaskColorSet_FnPtr | This is type layerMaskColorSet_FnPtr. |
| | layerPositionGet_FnPtr | This is type layerPositionGet_FnPtr. |
| | layerPositionSet_FnPtr | This is type layerPositionSet_FnPtr. |
| | layerSizeGet_FnPtr | This is type layerSizeGet_FnPtr. |
| | layerSizeSet_FnPtr | This is type layerSizeSet_FnPtr. |
| | layerSwapped_FnPtr | This is type layerSwapped_FnPtr. |
| | orientationGet_FnPtr | This is type orientationGet_FnPtr. |
| | orientationSet_FnPtr | This is type orientationSet_FnPtr. |
| | pixelGet_FnPtr | This is type pixelGet_FnPtr. |
| | pixelSet_FnPtr | This is type pixelSet_FnPtr. |
| | syncCallbackGet_FnPtr | This is type syncCallbackGet_FnPtr. |
| | syncCallbackSet_FnPtr | This is type syncCallbackSet_FnPtr. |
| | syncCallbackSt_FnPtr | This is type syncCallbackSt_FnPtr. |
| | update_FnPtr | This is type update_FnPtr. |
| | GFX_ColorInfo | This is variable GFX_ColorInfo. |
| | GFX_Rect_Zero | This is variable GFX_Rect_Zero. |
| | AGBA_8888_ALPHA_MASK | This is macro AGBA_8888_ALPHA_MASK. |
| | AGBA_8888_BLUE_MASK | This is macro AGBA_8888_BLUE_MASK. |
| | AGBA_8888_GREEN_MASK | This is macro AGBA_8888_GREEN_MASK. |
| | AGBA_8888_RED_MASK | This is macro AGBA_8888_RED_MASK. |
| | GFX_ANTIALIAS_MODE_COUNT | This is macro GFX_ANTIALIAS_MODE_COUNT. |
| | GFX_COLOR_MAX_SIZE | This is macro GFX_COLOR_MAX_SIZE. |
| | GFX_COLOR_MODE_COUNT | This is macro GFX_COLOR_MODE_COUNT. |
| | GFX_COLOR_MODE_IS_ALPHA | This is macro GFX_COLOR_MODE_IS_ALPHA. |
| | GFX_COLOR_MODE_IS_INDEX | This is macro GFX_COLOR_MODE_IS_INDEX. |
| | GFX_COLOR_MODE_IS_PIXEL | This is macro GFX_COLOR_MODE_IS_PIXEL. |
| | GFX_COLOR_MODE_LAST_COLOR | This is macro GFX_COLOR_MODE_LAST_COLOR. |
| | GFX_DRAW_MODE_COUNT | This is macro GFX_DRAW_MODE_COUNT. |
| | GFX_FAILURE | This is macro GFX_FAILURE. |
| | GFX_FALSE | This is macro GFX_FALSE. |
| | GFX_MAX_BUFFER_COUNT | This is macro GFX_MAX_BUFFER_COUNT. |
| | GFX_NULL | This is macro GFX_NULL. |
| | GFX_NUM_FLAGS | This is macro GFX_NUM_FLAGS. |
| | GFX_SUCCESS | This is macro GFX_SUCCESS. |
| | GFX_TRUE | This is macro GFX_TRUE. |
| | GFX_UNSUPPORTED | This is macro GFX_UNSUPPORTED. |
| | RGB_2_BITS | This is macro RGB_2_BITS. |
| | RGB_3_BITS | This is macro RGB_3_BITS. |
| | RGB_332_BLUE_MASK | This is macro RGB_332_BLUE_MASK. |
| | RGB_332_GREEN_MASK | This is macro RGB_332_GREEN_MASK. |
| | RGB_332_RED_MASK | This is macro RGB_332_RED_MASK. |
| | RGB_5_BITS | This is macro RGB_5_BITS. |
| | RGB_565_BLUE_MASK | This is macro RGB_565_BLUE_MASK. |
| | RGB_565_GREEN_MASK | This is macro RGB_565_GREEN_MASK. |
| | RGB_565_RED_MASK | This is macro RGB_565_RED_MASK. |
| | RGB_6_BITS | This is macro RGB_6_BITS. |
| | RGB_8_BITS | This is macro RGB_8_BITS. |
| | RGB_888_BLUE_MASK | This is macro RGB_888_BLUE_MASK. |

| | | |
|---|---|---|
| | RGB_888_GREEN_MASK | This is macro RGB_888_GREEN_MASK. |
| | RGB_888_RED_MASK | This is macro RGB_888_RED_MASK. |
| | RGBA_5551_ALPHA_MASK | This is macro RGBA_5551_ALPHA_MASK. |
| | RGBA_5551_BLUE_MASK | This is macro RGBA_5551_BLUE_MASK. |
| | RGBA_5551_GREEN_MASK | This is macro RGBA_5551_GREEN_MASK. |
| | RGBA_5551_RED_MASK | This is macro RGBA_5551_RED_MASK. |
| | RGBA_8888_ALPHA_MASK | This is macro RGBA_8888_ALPHA_MASK. |
| | RGBA_8888_BLUE_MASK | This is macro RGBA_8888_BLUE_MASK. |
| | RGBA_8888_GREEN_MASK | This is macro RGBA_8888_GREEN_MASK. |
| | RGBA_8888_RED_MASK | This is macro RGBA_8888_RED_MASK. |
| | initialize_FnPtr | This is type initialize_FnPtr. |
| | interrupt_FnPtr | GFX_DRAW_PIPELINE_ENABLED |
| | GFX_DrawPipeline_t | |
| | GFX_PipelineMode_t | Hardware draw path settings. |
| | GFX_ResizeMode_t | This is type GFX_ResizeMode. |
| | GFXU_ImageFlags_t | A list of flags describing an image asset |
| | blendColor_FnPtr | This is type blendColor_FnPtr. |
| | blendGetPoint_FnPtr | This is type blendGetPoint_FnPtr. |
| | drawPipelineModeGet_FnPtr | This is type drawPipelineModeGet_FnPtr. |
| | drawPipelineModeSet_FnPtr | This is type drawPipelineModeSet_FnPtr. |
| | drawResizeModeGet_FnPtr | This is type drawResizeModeGet_FnPtr. |
| | drawResizeModeSet_FnPtr | This is type drawResizeModeSet_FnPtr. |
| | drawStretchBlit_FnPtr | This is type drawStretchBlit_FnPtr. |
| | drawTargetGet_FnPtr | This is type drawTargetGet_FnPtr. |
| | drawTargetSet_FnPtr | This is type drawTargetSet_FnPtr. |
| | GFX_DrawPipeline | |
| | GFX_PipelineMode | Hardware draw path settings. |
| | GFX_ResizeMode | This is type GFX_ResizeMode. |
| | GFXU_ImageFlags | A list of flags describing an image asset |
| | layerSwapPending_FnPtr | This is type layerSwapPending_FnPtr. |
| | maskColor_FnPtr | This is type maskColor_FnPtr. |
| | mirrorPoint_FnPtr | This is type mirrorPoint_FnPtr. |
| | orientPoint_FnPtr | This is type orientPoint_FnPtr. |
| | pixelGetArray_FnPtr | This is type pixelGetArray_FnPtr. |
| | GFX_PIPELINE_MODE_COUNT | This is macro GFX_PIPELINE_MODE_COUNT. |
| | GFX_RESIZE_MODE_COUNT | This is macro GFX_RESIZE_MODE_COUNT. |
| | GFX_ASSERT | This is macro GFX_ASSERT. |
| | GFX_GLOBAL_PALETTE_SIZE | This is macro GFX_GLOBAL_PALETTE_SIZE. |
| | GFX_GlobalPalette | This is type GFX_GlobalPalette. |
| | globalPaletteGet_FnPtr | This is type globalPaletteGet_FnPtr. |
| | globalPaletteSet_FnPtr | This is type globalPaletteSet_FnPtr. |
| | layerEffectSet_FnPtr | This is type layerEffectSet_FnPtr. |
| | GFX_DEPRECATED | This is macro GFX_DEPRECATED. |

## Description

This section describes the Aria User Interface Library Hardware Abstraction Layer interface.

### *a) Functions*

#### GFX_AbsoluteValue Function

Calculates the absolute value of a signed integer.

## File

gfx_math.h

### C

```
LIB_EXPORT uint32_t GFX_AbsoluteValue(int32_t val);
```

### Returns

uint32_t - the absolute value

### Parameters

| Parameters | Description |
|---|---|
| val | the number to consider |

### Function

uint32_t GFX_AbsoluteValue(int32_t val);

## GFX_ActiveContext Function

Gets the current set active HAL context.

### File

gfx_context.h

### C

```
LIB_EXPORT GFX_Context* GFX_ActiveContext();
```

### Returns

GFX_Context* - the active context or NULL

### Function

GFX_Context* GFX_ActiveContext(void)

## GFX_Clampf Function

Clamps a float between a min and max

### File

gfx_math.h

### C

```
LIB_EXPORT float GFX_Clampf(float min, float max, float f);
```

### Returns

float - the clamped value

### Parameters

| Parameters | Description |
|---|---|
| min | the minimum value |
| max | the maximum value |
| i | the float to clamp |

### Function

float GFX_Clampi(float min, float max, float i);

## GFX_Clampi Function

Clamps an integer between a min and max

### File

gfx_math.h

### C

```
LIB_EXPORT int32_t GFX_Clampi(int32_t min, int32_t max, int32_t i);
```

### Returns

int32_t - the clamped value

### Parameters

| Parameters | Description |
|---|---|
| min | the minimum value |
| max | the maximum value |
| i | the number to clamp |

### Function

int32_t GFX_Clampi(int32_t min, int32_t max, int32_t i);

## GFX_ColorChannelAlpha Function

Used for getting the alpha color channel of a given color value.

### File

gfx_color.h

### C

```
LIB_EXPORT uint32_t GFX_ColorChannelAlpha(GFX_Color clr, GFX_ColorMode mode);
```

### Returns

uint32_t - the alpha color channel

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color | the source color value |
| GFX_ColorMode | the source color mode |

### Function

uint32_t GFX_ColorChannelAlpha( GFX_Color clr, GFX_ColorMode mode)

## GFX_ColorChannelGreen Function

Used for getting the green color channel of a given color value.

### File

gfx_color.h

### C

```
LIB_EXPORT uint32_t GFX_ColorChannelGreen(GFX_Color clr, GFX_ColorMode mode);
```

### Returns

uint32_t - the green color channel

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color | the source color value |
| GFX_ColorMode | the source color mode |

### Function

uint32_t GFX_ColorChannelGreen( GFX_Color clr, GFX_ColorMode mode)

## GFX_ColorChannelRed Function

Used for getting the red color channel of a given color value.

### File

gfx_color.h

### C

```
LIB_EXPORT uint32_t GFX_ColorChannelRed(GFX_Color clr, GFX_ColorMode mode);
```

### Returns

uint32_t - the red color channel

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color | the source color value |
| GFX_ColorMode | the source color mode |

### Function

uint32_t GFX_ColorChannelRed( GFX_Color clr, GFX_ColorMode mode)

## GFX_ColorConvert Function

Converts a color value from one mode to another

### File

gfx_color.h

### C

```
LIB_EXPORT GFX_Color GFX_ColorConvert(GFX_ColorMode mode_in, GFX_ColorMode mode_out, GFX_Color color);
```

### Returns

GFX_Color - the result color

### Parameters

| Parameters | Description |
|---|---|
| GFX_ColorMode | the input color mode the output color mode |
| GFX_Color | the source color |

### Function

GFX_Color GFX_ColorConvert(GFX_ColorMode mode_in,
GFX_ColorMode mode_out,
GFX_Color color)

## GFX_ColorLerp Function

Linear interpolation between two colors

### File

gfx_color.h

### C

```
LIB_EXPORT GFX_Color GFX_ColorLerp(GFX_Color l, GFX_Color r, uint32_t percent, GFX_ColorMode mode);
```

### Returns

GFX_Color - the result color

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color | first color input second color input |
| uint32_t | percentage of interpolation [0-100] |
| GFX_ColorMode | input color mode |

### Function

GFX_Color GFX_ColorLerp(GFX_Color l,
GFX_Color r,

uint32_t percent,
GFX_ColorMode mode)

## GFX_ColorModeInfoGet Function

### File

gfx_color.h

### C

```
LIB_EXPORT GFX_ColorModeInfo GFX_ColorModeInfoGet(GFX_ColorMode mode);
```

### Section

Routines

## GFX_ColorValue Function

Used for getting a color value by name.

### File

gfx_color.h

### C

```
LIB_EXPORT GFX_Color GFX_ColorValue(GFX_ColorMode mode, GFX_ColorName name);
```

### Returns

GFX_Color - the color value of the given name in the specified format

### Parameters

| Parameters | Description |
|---|---|
| GFX_ColorMode | the color mode for the return type |
| GFX_ColorName | the name of the color to retrieve |

### Function

GFX_Color GFX_ColorValue(GFX_ColorMode mode, GFX_ColorName name)

## GFX_ContextActiveSet Function

Sets the active context

### File

gfx_context.h

### C

```
void GFX_ContextActiveSet(GFX_Context* const context);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Context* | the new active context or NULL |

### Function

void GFX_ContextActiveSet( GFX_Context* const context)

## GFX_LayerReadBuffer Function

Gets the pointer to the layer's current read pixel buffer.

### File

gfx_layer.h

### C

```
GFX_PixelBuffer* GFX_LayerReadBuffer(GFX_Layer* layer);
```

### Returns

GFX_PixelBuffer* - the pointer to the read pixel buffer

### Parameters

| Parameters | Description |
|---|---|
| GFX_Layer* | the pointer to the layer |

### Function

GFX_PixelBuffer* GFX_LayerReadBuffer(GFX_Layer* layer)

## GFX_LayerRotate Function

Swaps the width and height dimensions of a layer. Can be used for run-time display orientation

### File

gfx_layer.h

### C

```
void GFX_LayerRotate(GFX_Layer* layer);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Layer* | the layer to operate on |

### Function

void GFX_LayerRotate( GFX_Layer* layer)

## GFX_LayerSwap Function

Performs a swap operation on the given layer. This advances the pointers of layer's buffer chain. The current write buffer becomes the new read buffer and a new buffer is chosen as the new write buffer. Has no effect in single buffer environments.

### File

gfx_layer.h

### C

```
void GFX_LayerSwap(GFX_Layer* layer);
```

### Parameters

| Parameters | Description |
|---|---|
| GFX_Layer* | the layer to operate on |

### Function

void GFX_LayerSwap( GFX_Layer* layer)

## GFX_LayerWriteBuffer Function

Gets the pointer to the layer's current write pixel buffer.

### File

gfx_layer.h

### C

```
GFX_PixelBuffer* GFX_LayerWriteBuffer(GFX_Layer* layer);
```

### Returns

GFX_PixelBuffer* - the pointer to the write pixel buffer

### Parameters

| Parameters | Description |
|---|---|
| GFX_Layer* | the pointer to the layer |

### Function

GFX_PixelBuffer* GFX_LayerWriteBuffer(GFX_Layer* layer)

## GFX_Lerp Function

Performs a linear interpolation of an integer based on a percentage between two signed points.

### File

gfx_math.h

### C

```
LIB_EXPORT int32_t GFX_Lerp(int32_t x, int32_t y, uint32_t per);
```

### Returns

int32_t - the interpolated value

### Parameters

| Parameters | Description |
|---|---|
| x | the first point to consider |
| y | the second point to consider |
| per | the percentage of interpolation |

### Function

int32_t GFX_Lerp(int32_t x, int32_t y, uint32_t per);

## GFX_Maxf Function

Returns the larger of two floats.

### File

gfx_math.h

### C

```
LIB_EXPORT float GFX_Maxf(float l, float r);
```

### Returns

float - the larger of the two floats

### Parameters

| Parameters | Description |
|---|---|
| l | the first float to test |
| r | the second float to test |

### Function

float GFX_Maxf(float l, float r);

## GFX_Maxi Function

Returns the larger of two integers.

### File

gfx_math.h

### C

```
LIB_EXPORT int32_t GFX_Maxi(int32_t l, int32_t r);
```

### Returns

int32_t - the larger of the two numbers

### Parameters

| Parameters | Description |
|---|---|
| l | the first number to test |
| r | the second number to test |

### Function

int32_t GFX_Maxi(int32_t l, int32_t r);

## GFX_Minf Function

Returns the smaller of two floats.

### File

gfx_math.h

### C

```
LIB_EXPORT float GFX_Minf(float l, float r);
```

### Returns

float - the smaller of the two floats

### Parameters

| Parameters | Description |
|---|---|
| l | the first float to test |
| r | the second float to test |

### Function

float GFX_Minf(float l, float r);

## GFX_Mini Function

Returns the smaller of two integers.

### File

gfx_math.h

### C

```
LIB_EXPORT int32_t GFX_Mini(int32_t l, int32_t r);
```

### Returns

int32_t - the smaller of the two numbers

### Parameters

| Parameters | Description |
|---|---|
| l | the first number to test |
| r | the second number to test |

### Function

int32_t GFX_Mini(int32_t l, int32_t r);

## GFX_Percent Function

Calculates the percentage of one number when applied to another. Integer based. Accuracy for higher numbers is not guaranteed.

The result is the decimal percentage multiplied by 100.

### File

gfx_math.h

### C

```
LIB_EXPORT uint32_t GFX_Percent(uint32_t l, uint32_t r);
```

### Returns

uint32_t - the percentage represented as a whole number

### Parameters

| Parameters | Description |
|---|---|
| l | the first number of the equation |
| r | the second number of the equation |

### Function

uint32_t GFX_Percent(uint32_t l, uint32_t r);

## GFX_PercentOf Function

Calculates the percentage of a number. Returns a whole number with no decimal component.

### File

gfx_math.h

### C

```
LIB_EXPORT uint32_t GFX_PercentOf(uint32_t num, uint32_t percent);
```

### Returns

uint32_t - the resultant percentage of the number

### Parameters

| Parameters | Description |
|---|---|
| num | the number to consider |
| percent | the percentage to apply |

### Function

uint32_t GFX_PercentOf(uint32_t l, uint32_t percent);

## GFX_PercentWholeRounded Function

Calculates the percentage of one number when applied to another. Integer based. Accuracy for higher numbers is not guaranteed.

The difference between this and GFX_Percent is that the decimal portion of the whole number is rounded off.

### File

gfx_math.h

### C

```
LIB_EXPORT uint32_t GFX_PercentWholeRounded(uint32_t l, uint32_t r);
```

### Returns

uint32_t - the percentage represented as a whole number

### Parameters

| Parameters | Description |
|---|---|
| l | the first number of the equation |
| r | the second number of the equation |

### Function

uint32_t GFX_PercentWholeRounded(uint32_t l, uint32_t r);

## GFX_PixelBufferAreaFill Function

Fills an area of a pixel buffer with a solid color. Caller is responsible for ensuring that the color is the same color format as the destination buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferAreaFill(const GFX_PixelBuffer* const buffer, const GFX_Rect* const
rect, const GFX_Color color);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the buffer to manipulate |
| const GFX_Rect* const rect | the rectangle of the buffer to fill |
| const GFX_Color color | the color to use for the fill operation |

### Function

GFX_Result GFX_PixelBufferAreaFill(const GFX_PixelBuffer* const buffer,
const GFX_Rect* const rect,
const GFX_Color color)

## GFX_PixelBufferAreaFill_Unsafe Function

Fills an area of a pixel buffer with a solid color. Caller is responsible for ensuring that the color is the same color format as the destination buffer. Like GFX_PixelBufferAreaFill but performs no bounds checking.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferAreaFill_Unsafe(const GFX_PixelBuffer* const buffer, const GFX_Rect*
const rect, const GFX_Color color);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the buffer to manipulate |
| const GFX_Rect* const rect | the rectangle of the buffer to fill |
| const GFX_Color color | the color to use for the fill operation |

### Function

GFX_Result GFX_PixelBufferAreaFill_Unsafe(const GFX_PixelBuffer* const buffer,
const GFX_Rect* const rect,
const GFX_Color color)

## GFX_PixelBufferAreaGet Function

Extracts a rectanglar section of pixels from a pixel buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferAreaGet(const GFX_PixelBuffer* const buffer, const GFX_Rect* const
rect, GFX_MemoryIntf* mem_intf, GFX_PixelBuffer* out);
```

### Returns

GFX_Result

## Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the source buffer |
| const GFX_Rect* rect | the area to extract |
| GFX_MemoryIntf* | the memory interface to use for memory operations |
| GFX_PixelBuffer* | the resultant pixel buffer |

## Function

GFX_Result GFX_PixelBufferAreaGet(const GFX_PixelBuffer* const buffer,
const GFX_Rect* const rect,
GFX_MemoryIntf* mem_intf,
GFX_PixelBuffer* out)

### GFX_PixelBufferAreaGet_Unsafe Function

Extracts a rectanglar section of pixels from a pixel buffer. Like GFX_PixelBufferAreaGet but performs no clipping between the rectangles of the extract area and the source buffer.

## File

gfx_pixel_buffer.h

## C

```
LIB_EXPORT GFX_Result GFX_PixelBufferAreaGet_Unsafe(const GFX_PixelBuffer* const buffer, const GFX_Rect*
const rect, GFX_MemoryIntf* mem_intf, GFX_PixelBuffer* out);
```

## Returns

GFX_Result

## Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the source buffer |
| const GFX_Rect* rect | the area to extract |
| GFX_MemoryIntf* | the memory interface to use for memory operations |
| GFX_PixelBuffer* | the resultant pixel buffer |

## Function

GFX_Result GFX_PixelBufferAreaGet_Unsafe(const GFX_PixelBuffer* const buffer,
const GFX_Rect* const rect,
GFX_MemoryIntf* mem_intf,
GFX_PixelBuffer* out)

### GFX_PixelBufferAreaSet Function

Copies an area of pixels from a source buffer to a destination buffer. If the source buffer format does not match the destination format the data will be converted to match during the copy operation.

## File

gfx_pixel_buffer.h

## C

```
LIB_EXPORT GFX_Result GFX_PixelBufferAreaSet(const GFX_PixelBuffer* const source, const GFX_Rect* const
source_rect, const GFX_PixelBuffer* const dest, const GFX_Point* const pnt, GFX_MemoryIntf* mem_intf);
```

## Returns

GFX_Result

## Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const source | the source buffer |

| | |
|---|---|
| const GFX_Rect* const source_rect | the rectangle of the source buffer to use |
| const GFX_PixelBuffer* const dest | the destination buffer to copy to |
| const GFX_Point* const pnt | the location of the destination to copy to |
| GFX_MemoryIntf* | the memory interface to use for memory operations |

### Function

GFX_Result GFX_PixelBufferAreaSet(const GFX_PixelBuffer* const source,
const GFX_Rect* const source_rect,
const GFX_PixelBuffer* const dest,
const GFX_Point* const pnt,
GFX_MemoryIntf* mem_intf)

## GFX_PixelBufferAreaSet_Unsafe Function

Copies an area of pixels from a source buffer to a destination buffer. If the source buffer format does not match the destination format the data will be converted to match during the copy operation. Like GFX_PixelBufferAreaSet but performs no bounds checking.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferAreaSet_Unsafe(const GFX_PixelBuffer* const source, const GFX_Rect*
const source_rect, const GFX_PixelBuffer* const dest, const GFX_Point* const pnt, GFX_MemoryIntf* mem_intf);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const source | the source buffer |
| const GFX_Rect* const source_rect | the rectangle of the source buffer to use |
| const GFX_PixelBuffer* const dest | the destination buffer to copy to |
| const GFX_Point* const pnt | the location of the destination to copy to |
| GFX_MemoryIntf* | the memory interface to use for memory operations |

### Function

GFX_Result GFX_PixelBufferAreaSet_Unsafe(const GFX_PixelBuffer* const source,
const GFX_Rect* const source_rect,
const GFX_PixelBuffer* const dest,
const GFX_Point* const pnt,
GFX_MemoryIntf* mem_intf)

## GFX_PixelBufferClipRect Function

Clips a rectangle against a pixel buffer. The result is guaranteed to fit inside the buffer's area.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferClipRect(const GFX_PixelBuffer* const buffer, const GFX_Rect* const
rect, GFX_Rect* result);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the source buffer |

| | |
|---|---|
| const GFX_Rect* const | the rectangle to analyze |
| GFX_Rect* result | the clipped rectangle |

### Function

GFX_Result GFX_PixelBufferClipRect(const GFX_PixelBuffer* const buffer,
const GFX_Rect* const rect,
GFX_Rect* result)

## GFX_PixelBufferConvert Function

Duplicates a pixel buffer and converts the copy to another color mode.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferConvert(const GFX_PixelBuffer* const source, const GFX_ColorMode
result_mode, GFX_MemoryIntf* mem_intf, GFX_PixelBuffer* result);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const source | the source buffer |
| const GFX_ColorMode result_mode | the desired color mode |
| GFX_MemoryIntf* | the memory interface to use for memory operations |
| GFX_PixelBuffer* | the resultant pixel buffer |

### Function

GFX_Result GFX_PixelBufferConvert(const GFX_PixelBuffer* const source,
const GFX_ColorMode result_mode,
GFX_MemoryIntf* mem_intf,
GFX_PixelBuffer* result)

## GFX_PixelBufferCopy Function

Creates a copy of the input buffer. If the input buffer's pixel data is not null then the data will be duplicated for the result buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferCopy(const GFX_PixelBuffer* const buffer, GFX_MemoryIntf* mem_intf,
GFX_PixelBuffer* result);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the source buffer |
| GFX_MemoryIntf* | the memory interface to use for memory operations |
| GFX_PixelBuffer* | the result buffer |

### Function

GFX_Result GFX_PixelBufferCopy(const GFX_PixelBuffer* const buffer,
GFX_MemoryIntf* mem_intf,
GFX_PixelBuffer* result)

## GFX_PixelBufferCreate Function

Initializes a pixel buffer struct. Does not actually allocate any memory.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferCreate(const int32_t width, const int32_t height, const GFX_ColorMode
mode, const void* const address, GFX_PixelBuffer* buffer);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const int32_t | the width of the buffer the height of the buffer |
| const GFX_ColorMode | the color mode of the buffer |
| const void* | the data addres of the buffer (may be NULL) |
| GFX_PixelBuffer* | pointer of the pixel buffer buffer to initialize |

### Function

GFX_Result GFX_PixelBufferCreate(const int32_t width,

const int32_t height,

const GFX_ColorMode mode,

const void* const address,

GFX_PixelBuffer* buffer)

## GFX_PixelBufferDestroy Function

Destroys a pixel buffer construct. If the buffer's pixels pointer is not zero this function will attempt to free it using the provided GFX_MemoryIntf memory interface.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferDestroy(GFX_PixelBuffer* const buffer, GFX_MemoryIntf* mem_intf);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFX_PixelBuffer* | the buffer to destroy |
| GFX_MemoryIntf* | the memory interface to reference for free() |

### Function

GFX_Result GFX_PixelBufferDestroy(GFX_PixelBuffer* const buffer,

GFX_MemoryIntf* mem_intf)

## GFX_PixelBufferGet Function

Gets the value of the pixel that resides at the provided point in the given buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Color GFX_PixelBufferGet(const GFX_PixelBuffer* const buffer, const GFX_Point* const pnt);
```

### Returns

GFX_Color - the value of the pixel at the point in the source buffer

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* | the source buffer |
| const GFX_Point* | the point for which the offset should be calculated |

### Function

GFX_Color GFX_PixelBufferGet(const GFX_PixelBuffer* const buffer,
const GFX_Point* const pnt)

## GFX_PixelBufferGet_Unsafe Function

Gets the value of the pixel that resides at the provided point in the given buffer. Like GFX_PixelBufferGet but performs no bounds checking.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Color GFX_PixelBufferGet_Unsafe(const GFX_PixelBuffer* const buffer, const GFX_Point* const pnt);
```

### Returns

GFX_Color - the value of the pixel at the point in the source buffer

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* | the source buffer |
| const GFX_Point* | the point for which the offset should be calculated |

### Function

GFX_Color GFX_PixelBufferGet_Unsafe(const GFX_PixelBuffer* const buffer,
const GFX_Point* const pnt)

## GFX_PixelBufferGetIndex Function

Interprets the pixel buffer as a table of indices and looks up a specific index at position 'idx'. Indices may be 1bpp, 4bpp, or 8bpp in size and are indicated by the color mode of the pixel buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Color GFX_PixelBufferGetIndex(const GFX_PixelBuffer* const buffer, const int32_t idx);
```

### Returns

GFX_Color - the resultant value that was retrieved

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const | the input buffer |
| const int32_t | the index to retrieve |

### Function

GFX_Color GFX_PixelBufferGetIndex(const GFX_PixelBuffer* const buffer,
const int32_t idx)

## GFX_PixelBufferOffsetGet Function

Gets the offset address of the pixel that resides at the provided point in the given buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Buffer GFX_PixelBufferOffsetGet(const GFX_PixelBuffer* const buffer, const GFX_Point* const pnt);
```

### Returns

GFX_Buffer - the pointer to the offset point in the source buffer

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* | the source buffer |
| const GFX_Point* | the point for which the offset should be calculated |

### Function

GFX_Buffer GFX_PixelBufferOffsetGet(const GFX_PixelBuffer* const buffer,
const GFX_Point* const pnt)

## GFX_PixelBufferOffsetGet_Unsafe Function

Gets the offset address of the pixel that resides at the provided point in the given buffer. Similar to GFX_PixelBufferOffsetGet but performs no bounds checking.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Buffer GFX_PixelBufferOffsetGet_Unsafe(const GFX_PixelBuffer* const buffer, const GFX_Point* const pnt);
```

### Returns

GFX_Buffer - the pointer to the offset point in the source buffer

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* | the source buffer |
| const GFX_Point* | the point for which the offset should be calculated |

### Function

GFX_Buffer GFX_PixelBufferOffsetGet_Unsafe(const GFX_PixelBuffer* const buffer,
const GFX_Point* const pnt)

## GFX_PixelBufferSet Function

Sets a pixel in a pixel buffer at a point to a specified color. Caller is responsible for ensuring that the input color is in the same color format as the pixel buffer.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferSet(const GFX_PixelBuffer* const buffer, const GFX_Point* const pnt, GFX_Color color);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the buffer to operate on |
| const GFX_Point* const | the location of the pixel to set |
| GFX_Color | the color to set the pixel to. must be the same format as the buffer |

### Function

GFX_Result GFX_PixelBufferSet(const GFX_PixelBuffer* const buffer,
const GFX_Point* const pnt,
GFX_Color color)

## GFX_PixelBufferSet_Unsafe Function

Sets a pixel in a pixel buffer at a point to a specified color. Caller is responsible for ensuring that the input color is in the same color format as the pixel buffer. Like GFX_PixelBufferSet but performs no bounds checking.

### File

gfx_pixel_buffer.h

### C

```
LIB_EXPORT GFX_Result GFX_PixelBufferSet_Unsafe(const GFX_PixelBuffer* const buffer, const GFX_Point* const pnt, GFX_Color color);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| const GFX_PixelBuffer* const buffer | the buffer to operate on |
| const GFX_Point* const | the location of the pixel to set |
| GFX_Color | the color to set the pixel to. must be the same format as the buffer |

### Function

GFX_Result GFX_PixelBufferSet_Unsafe(const GFX_PixelBuffer* const buffer,
const GFX_Point* const pnt,
GFX_Color color)

## GFX_RectClip Function

Clips a rectangle to the space of another rectangle. The result rectangle is a rectangle that will fit inside both of the given rectangles.

### File

gfx_rect.h

### C

```
LIB_EXPORT void GFX_RectClip(const GFX_Rect* l_rect, const GFX_Rect* r_rect, GFX_Rect* result);
```

### Returns

void

### Remarks

result will equals l_rect if the rectangles aren't intersecting

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* l_rect | the subject rectangle |
| const GFX_Rect* r_rect | the object rectangle |
| GFX_Rect* result | the result rectangle |

### Function

void GFX_RectClip(const GFX_Rect* l_rect,
const GFX_Rect* r_rect,
GFX_Rect* result)

## GFX_RectContainsPoint Function

Determines if a point is inside a rectangle.

### File

gfx_rect.h

### C

```
LIB_EXPORT GFX_Bool GFX_RectContainsPoint(const GFX_Rect* rect, const GFX_Point* point);
```

### Returns

GFX_Bool - GFX_TRUE if the point is inside the rectangle

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* rect | the rectangle to test |
| const GFX_Point* point | the point to use for the test |

### Function

GFX_Bool GFX_RectContainsPoint(const GFX_Rect* rect, const GFX_Point* point)

## GFX_RectContainsRect Function

Determines if a rectangle is completely inside another rectangle. Still returns true if the edges are touching.

### File

gfx_rect.h

### C

```
LIB_EXPORT GFX_Bool GFX_RectContainsRect(const GFX_Rect* l_rect, const GFX_Rect* r_rect);
```

### Returns

GFX_Bool - returns GFX_TRUE if r_rect is completly inside l_rect

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* l_rect | the subject rectangle |
| const GFX_Rect* r_rect | the object rectangle |

### Function

GFX_Bool GFX_RectContainsRect(const GFX_Rect* l_rect, const GFX_Rect* r_rect)

## GFX_RectIntersects Function

Determines if two rectangles are intersecting

### File

gfx_rect.h

### C

```
LIB_EXPORT GFX_Bool GFX_RectIntersects(const GFX_Rect* l_rect, const GFX_Rect* r_rect);
```

### Returns

GFX_Bool - returns GFX_TRUE if l_rect and r_rect are intersecting

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* l_rect | rectangle argument |
| const GFX_Rect* r_rect | rectangle argument |

### Function

GFX_Bool GFX_RectIntersects(const GFX_Rect* l_rect, const GFX_Rect* r_rect)

## GFX_ScaleInteger Function

Scales an integer from one number range of 0 -> n0 to another range 0 -> n1 based on percentages.

### File

gfx_math.h

### C

```
LIB_EXPORT uint32_t GFX_ScaleInteger(uint32_t num, uint32_t oldMax, uint32_t newMax);
```

### Returns

uint32_t - the number as defined in the new number range

### Parameters

| Parameters | Description |
|---|---|
| num | the number to consider |
| oldMax | the old range maximum |
| newMax | the new range maximum |

### Function

uint32_t GFX_ScaleInteger(uint32_t num, uint32_t oldMax, uint32_t newMax);

## createDefaultMemIntf Function

### File

gfxu_image_utils.h

### C

```
void createDefaultMemIntf(GFXU_MemoryIntf* memIntf);
```

### Description

internal use only

## GFX_ColorBilerp Function

Calculates bilinear interpolation between four colors

### File

gfx_color.h

### C

```
LIB_EXPORT GFX_Color GFX_ColorBilerp(GFX_Color c00, GFX_Color c01, GFX_Color c10, GFX_Color c11, uint32_t
xper, uint32_t yper, GFX_ColorMode mode);
```

### Returns

GFX_Color - the result color

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color c00 | top left color input |
| GFX_Color c01 | top right color input |

| | |
|---|---|
| GFX_Color c10 | bottom left color input |
| GFX_Color c11 | bottom right color input |
| uint32_t xper | percentage of interpolation in x [0-100] |
| uint32_t yper | percentage of interpolation in y [0-100] |
| GFX_ColorMode | input color mode |

### Function

GFX_Color GFX_ColorBilerp(GFX_Color c00,
GFX_Color c01,
GFX_Color c10,
GFX_Color c11,
uint32_t xper,
uint32_t yper,
GFX_ColorMode mode)

## GFX_ColorBlend_RGBA_8888 Function

Blends two RGBA8888 colors together using their alpha channel values.

### File

gfx_color.h

### C

```
LIB_EXPORT GFX_Color GFX_ColorBlend_RGBA_8888(GFX_Color fore, GFX_Color back);
```

### Returns

GFX_Color - the blended result color

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color | the foreground color the background color |

### Function

GFX_Color GFX_ColorBlend_RGBA_8888(GFX_Color fore, GFX_Color back)

## GFX_ColorChannelBlue Function

Used for getting the blue color channel of a given color value.

### File

gfx_color.h

### C

```
LIB_EXPORT uint32_t GFX_ColorChannelBlue(GFX_Color clr, GFX_ColorMode mode);
```

### Returns

uint32_t - the blue color channel

### Parameters

| Parameters | Description |
|---|---|
| GFX_Color | the source color value |
| GFX_ColorMode | the source color mode |

### Function

uint32_t GFX_ColorChannelBlue( GFX_Color clr, GFX_ColorMode mode)

## GFX_DivideRounding Function

### File

gfx_math.h

### C

```
LIB_EXPORT int32_t GFX_DivideRounding(int32_t num, int32_t denom);
```

### Description

This is function GFX_DivideRounding.

## GFX_DrawBlit Function

Blits a buffer of pixels into the frame buffer.

### File

gfx_draw.h

### C

```
LIB_EXPORT GFX_Result GFX_DrawBlit(GFX_PixelBuffer* buffer, int32_t src_x, int32_t src_y, int32_t
src_width, int32_t src_height, int32_t dest_x, int32_t dest_y);
```

### Returns

GFX_Result - Returns GFX_TRUE if the blit was drawn successfully. Otherwise returns GFX_FALSE.

### Description

A pixel buffer is an array of pixel data that can be applied in bulk, or 'blit'ed, to the frame buffer. Pixel buffers may be of a different color mode and will be converted to match the destination frame buffer before application.

### Parameters

| Parameters | Description |
|---|---|
| buffer | the pointer to the source pixel buffer |
| src_x | the x component of the rectangle of the buffer to blit, usually 0 |
| src_y | the y component of the rectangle of the buffer to blit, usually 0 |
| src_width | width of the rectangle of the buffer to blit, usually the entire with of the source buffer |
| src_height | height of the rectangle of the buffer to blit, usually the entire height of the source buffer |
| dest_x | the x position to blit the source rectangle in the destination buffer<br>the y position to blit the source rectangle in the destination buffer |

### Function

GFX_Result GFX_Result GFX_DrawBlit(GFX_PixelBuffer* buffer,
int32_t src_x,
int32_t src_y,
int32_t src_width,
int32_t src_height,
int32_t dest_x,
int32_t dest_y);

## GFX_DrawCircle Function

Draws a circle from using the specified dimensions and the current draw state.

### File

gfx_draw.h

### C

```
LIB_EXPORT GFX_Result GFX_DrawCircle(int32_t x, int32_t y, int32_t radius);
```

### Returns

GFX_Result - Returns GFX_TRUE if the circle was drawn successfully. Otherwise returns GFX_FALSE.

### Parameters

| Parameters | Description |
|---|---|
| x | the x component of the origin position |
| y | the y component of the origin position |
| radius | the radius of the circle in pixels |

### Function

GFX_Result GFX_Result GFX_DrawCircle(int32_t x,
int32_t y,
int32_t radius);

## GFX_DrawLine Function

Draws a line from (x1,y1) to (x2,y2) using the current draw state.

### File

gfx_draw.h

### C

```
LIB_EXPORT GFX_Result GFX_DrawLine(int32_t x1, int32_t y1, int32_t x2, int32_t y2);
```

### Returns

GFX_Result - Returns GFX_TRUE if the line was drawn successfully. Otherwise returns GFX_FALSE.

### Parameters

| Parameters | Description |
|---|---|
| x1 | the x component of the first coordinate of the line |
| y1 | the y component of the first coordinate of the line |
| x2 | the x component of the second coordinate of the line |
| y2 | the y component of the second coordinate of the line |

### Function

GFX_Result GFX_Result GFX_DrawLine(int32_t x1,
int32_t y1,
int32_t x2,
int32_t y2);

## GFX_DrawPixel Function

Sets the pixel at X and Y using the current draw state.

### File

gfx_draw.h

### C

```
LIB_EXPORT GFX_Result GFX_DrawPixel(int32_t x, int32_t y);
```

### Returns

GFX_Result - Returns GFX_TRUE if the pixel was drawn successfully. Otherwise returns GFX_FALSE.

### Parameters

| Parameters | Description |
|---|---|
| x | the x coordinate of the pixel |
| y | the y coordinate of the pixel |

## Function

GFX_Result GFX_DrawPixel(int32_t x, int32_t y);

### GFX_DrawRect Function

Draws a rectangle using the specified dimensions and the current draw state.

## File

gfx_draw.h

## C

```
LIB_EXPORT GFX_Result GFX_DrawRect(int32_t x, int32_t y, int32_t width, int32_t height);
```

## Returns

GFX_Result - Returns GFX_TRUE if the rectangle was drawn successfully. Otherwise returns GFX_FALSE.

## Description

Draws a rectangle using the coordinates: x,y x + width - 1, y

x, y + height - 1 x + width - 1, y + height - 1

## Parameters

| Parameters | Description |
|---|---|
| x | the x position of the top left point of the rectangle |
| y | the y position of the top left point of the rectangle |
| width | the width of the rectangle in pixels |
| height | the height of the rectangle in pixels |

## Function

GFX_Result GFX_Result GFX_DrawLine(int32_t x,

int32_t x,

int32_t width,

int32_t height);

### GFX_DrawStretchBlit Function

Blits a buffer of pixels into the frame buffer.

## File

gfx_draw.h

## C

```
LIB_EXPORT GFX_Result GFX_DrawStretchBlit(GFX_PixelBuffer* buffer, int32_t src_x, int32_t src_y, int32_t
src_width, int32_t src_height, int32_t dest_x, int32_t dest_y, int32_t dest_width, int32_t dest_height);
```

## Returns

GFX_Result - Returns GFX_TRUE if the blit was drawn successfully. Otherwise returns GFX_FALSE.

## Description

A pixel buffer is an array of pixel data that can be applied in bulk, or 'blit'ed, to the frame buffer. Pixel buffers may be of a different color mode and will be converted to match the destination frame buffer before application. This version can resize the source data before blitting. The option GFXF_RESIZE_METHOD selects the resize technique.

## Parameters

| Parameters | Description |
|---|---|
| buffer | the pointer to the source pixel buffer |
| src_x | the x component of the rectangle of the buffer to blit, usually 0 |
| src_y | the y component of the rectangle of the buffer to blit, usually 0 |
| src_width | width of the rectangle of the buffer to blit, usually the entire with of the source buffer |
| src_height | height of the rectangle of the buffer to blit, usually the entire height of the source buffer |

| | |
|---|---|
| dest_x | the x position to blit the source rectangle in the destination buffer<br>the y position to blit the source rectangle in the destination buffer |
| dest_width | the desired resize width |
| dest_height | the desired resize height |

## Function

GFX_Result GFX_DrawStretchBlit(GFX_PixelBuffer* buffer,
int32_t src_x,
int32_t src_y,
int32_t src_width,
int32_t src_height,
int32_t dest_x,
int32_t dest_y,
int32_t dest_width,
int32_t dest_height);

### GFX_LayerFromOrientedSpace Function

Transforms a layer oriented space to screen space.

## File

gfx_layer.h

## C

```
void GFX_LayerFromOrientedSpace(GFX_Rect* displayRect, GFX_Layer* layer, GFX_Orientation ori, GFX_Bool
mirrored);
```

## Returns

void

## Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* dispayRect | the rectangle of the display |
| GFX_Layert* layer | the layer |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

## Function

void GFX_LayerFromOrientedSpace( GFX_Rect* displayRect,
GFX_Layer* layer,
GFX_Orientation ori,
GFX_Bool mirrored)

### GFX_LayerPointFromOrientedSpace Function

Transforms a point from oriented space to screen space given a layer, a display orientation, and a mirroring setting.

## File

gfx_layer.h

## C

```
LIB_EXPORT GFX_Point GFX_LayerPointFromOrientedSpace(const GFX_Layer* layer, const GFX_Point* point,
GFX_Orientation ori, GFX_Bool mirrored);
```

## Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Layer* layer | the layer |
| const GFX_Point* point | the point |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

### Function

GFX_Point GFX_LayerPointFromOrientedSpace(const GFX_Layer* layer,
const GFX_Point* point,
GFX_Orientation ori,
GFX_Bool mirrored)

## GFX_LayerPointToOrientedSpace Function

Transforms a point from screen space to oriented space given a layer, a display orientation, and a mirroring setting.

### File

gfx_layer.h

### C

```
LIB_EXPORT GFX_Point GFX_LayerPointToOrientedSpace(const GFX_Layer* layer, const GFX_Point* point,
GFX_Orientation ori, GFX_Bool mirrored);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Layer* layer | the layer |
| const GFX_Point* point | the point to transform |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

### Function

GFX_Point GFX_LayerPointToOrientedSpace(const GFX_Rect* layerRect,
const GFX_Point* point,
GFX_Orientation ori,
GFX_Bool mirrored)

## GFX_LayerRectFromOrientedSpace Function

Transforms a layer point from oriented space to screen space given a layer, a display orientation, and a mirroring setting.

### File

gfx_layer.h

### C

```
LIB_EXPORT GFX_Rect GFX_LayerRectFromOrientedSpace(const GFX_Layer* layer, const GFX_Rect* rect,
GFX_Orientation ori, GFX_Bool mirrored);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Layer* layer | the layer |
| const GFX_Rect* rect | the rectangle to transform |

| | |
|---|---|
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

### Function

GFX_Rect GFX_LayerRectFromOrientedSpace(const GFX_Layer* layer,
const GFX_Rect* rect,
GFX_Orientation ori,
GFX_Bool mirrored)

## GFX_LayerRectToOrientedSpace Function

Transforms a rectangle from screen space to oriented space given a layer, a display orientation, and a mirroring setting.

### File

gfx_layer.h

### C

```
LIB_EXPORT GFX_Rect GFX_LayerRectToOrientedSpace(const GFX_Layer* layer, const GFX_Rect* rect,
GFX_Orientation ori, GFX_Bool mirrored);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Layer* layer | the layer |
| const GFX_Rect* rect | the rectangle to transform |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

### Function

GFX_Rect GFX_LayeRectToOrientedSpace(const GFX_Rect* layerRect,
const GFX_Rect* rect,
GFX_Orientation ori,
GFX_Bool mirrored)

## GFX_LayerToOrientedSpace Function

Transforms a layer from screen space to oriented space.

### File

gfx_layer.h

### C

```
void GFX_LayerToOrientedSpace(GFX_Rect* displayRect, GFX_Layer* layer, GFX_Orientation ori, GFX_Bool
mirrored);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| GFX_Rect* dispayRect | the rectangle of the display |
| GFX_Rect* layer | the layer |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

### Function

void GFX_LayerToOrientedSpace( GFX_Rect* displayRect,

GFX_Layer* layer,
GFX_Orientation ori,
GFX_Bool mirrored)

### GFX_PercentOfDec Function

#### File

gfx_math.h

#### C

```
LIB_EXPORT void GFX_PercentOfDec(uint32_t num, uint32_t percent, uint32_t* whl, uint32_t* dec);
```

#### Description

This is function GFX_PercentOfDec.

### GFX_RectClipAdj Function

#### File

gfx_rect.h

#### C

```
LIB_EXPORT GFX_Rect GFX_RectClipAdj(const GFX_Rect* l_rect, const GFX_Rect* r_rect, GFX_Rect* adj);
```

#### Description

This is function GFX_RectClipAdj.

### GFX_RectFromPoints Function

#### File

gfx_rect.h

#### C

```
LIB_EXPORT GFX_Rect GFX_RectFromPoints(const GFX_Point* p1, const GFX_Point* p2);
```

#### Description

This is function GFX_RectFromPoints.

### GFX_RectSplit Function

#### File

gfx_rect.h

#### C

```
LIB_EXPORT uint32_t GFX_RectSplit(const GFX_Rect* sub, const GFX_Rect* obj, GFX_Rect res[4]);
```

#### Description

This is function GFX_RectSplit.

### GFX_RectToPoints Function

#### File

gfx_rect.h

#### C

```
LIB_EXPORT void GFX_RectToPoints(const GFX_Rect* rect, GFX_Point* p1, GFX_Point* p2);
```

#### Description

This is function GFX_RectToPoints.

## GFX_UtilMirrorPoint Function

Reorients a point to a given mirrored orientation.

### File

gfx_util.h

### C

```
GFX_Point GFX_UtilMirrorPoint(const GFX_Point* point, const GFX_Rect* rect, GFX_Orientation ori);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Point* point | the point to reorient |
| const GFX_Rect* rect | the bounding rectangle space |
| GFX_Orientation | the orientation setting |

### Function

void GFX_UtilMirrorPoint(const GFX_Point* point,
const GFX_Rect* rect,
GFX_Orientation ori)

## GFX_UtilOrientPoint Function

Reorients a point to a given orthogonal orientation.

### File

gfx_util.h

### C

```
GFX_Point GFX_UtilOrientPoint(const GFX_Point* point, const GFX_Rect* rect, GFX_Orientation ori);
```

### Returns

GFX_Point

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Point* point | the point to reorient |
| const GFX_Rect* rect | the bounding rectangle space |
| GFX_Orientation | the orientation setting |

### Function

void GFX_UtilOrientPoint(const GFX_Point* point,
const GFX_Rect* rect,
GFX_Orientation ori)

## GFX_UtilPointFromOrientedSpace Function

Transforms a point from an oriented rectangle space to an outer space given a display orientation and a mirroring setting.

### File

gfx_util.h

### C

```
GFX_Point GFX_UtilPointFromOrientedSpace(const GFX_Rect* outerRect, const GFX_Rect* innerRect, const
GFX_Point* pnt, GFX_Orientation ori, GFX_Bool mirrored);
```

## Returns

GFX_Point

## Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* outerRect | the outer rectangle |
| const GFX_Rect* subRect | the inner rectangle |
| const GFX_Point* point | the point |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

## Function

GFX_Point GFX_UtilPointFromOrientedSpace(const GFX_Rect* displayRect,
const GFX_Rect* innerRect,
const GFX_Point* pnt,
GFX_Orientation ori,
GFX_Bool mirrored)

### GFX_UtilPointToOrientedSpace Function

Transforms a point to an oriented rectangle space to an outer space given a display orientation and a mirroring setting.

## File

gfx_util.h

## C

```
GFX_Point GFX_UtilPointToOrientedSpace(const GFX_Rect* outerRect, const GFX_Rect* innerRect, const
GFX_Point* pnt, GFX_Orientation ori, GFX_Bool mirrored);
```

## Returns

GFX_Point

## Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* outerRect | the outer rectangle |
| const GFX_Rect* subRect | the inner rectangle |
| const GFX_Point* point | the point |
| GFX_Orientation | the orientation setting |
| GFX_Bool | the mirroring setting |

## Function

GFX_Point GFX_UtilPointToOrientedSpace(const GFX_Rect* displayRect,
const GFX_Rect* layerRect,
const GFX_Point* point,
GFX_Orientation ori,
GFX_Bool mirrored)

### GFX_UtilSizeFromOrientedSpace Function

Transforms a size tuple from oriented space to screen space

## File

gfx_util.h

## C

```
GFX_Size GFX_UtilSizeFromOrientedSpace(const GFX_Size* size, GFX_Orientation ori);
```

### Returns

GFX_Size

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Size* size | the size dimension |
| GFX_Orientation | the orientation setting |

### Function

GFX_Size GFX_UtilSizeFromOrientedSpace(const GFX_Size* size, GFX_Orientation ori)

## GFX_UtilSizeToOrientedSpace Function

Transforms a size tuple from screen space to oriented space

### File

gfx_util.h

### C

```
GFX_Size GFX_UtilSizeToOrientedSpace(const GFX_Size* size, GFX_Orientation ori);
```

### Returns

GFX_Size

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Size* size | the size dimension |
| GFX_Orientation | the orientation setting |

### Function

void GFX_UtilSizeToOrientedSpace(const GFX_Size* size, GFX_Orientation ori)

## GFX_UtilSortPointsX Function

Sorts two points in the X axis

### File

gfx_util.h

### C

```
void GFX_UtilSortPointsX(GFX_Point* p1, GFX_Point* p2);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| GFX_Point* p1 | the first point to sort |
| GFX_Point* p2 | the second point to sort |

### Function

void GFX_UtilSortPointsX( GFX_Point* p1, GFX_Point* p2)

## GFX_UtilSortPointsY Function

Sorts two points in the Y axis

### File

gfx_util.h

### C

```
void GFX_UtilSortPointsY(GFX_Point* p1, GFX_Point* p2);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| GFX_Point* p1 | the first point to sort |
| GFX_Point* p2 | the second point to sort |

### Function

void GFX_UtilSortPointsY( GFX_Point* p1, GFX_Point* p2)

## GFX_RectCombine Function

Combines the area of two rectangles into a single rectangle.

### File

gfx_rect.h

### C

```
LIB_EXPORT GFX_Rect GFX_RectCombine(const GFX_Rect* l_rect, const GFX_Rect* r_rect);
```

### Returns

void

### Parameters

| Parameters | Description |
|---|---|
| const GFX_Rect* l_rect | the first rectangle |
| const GFX_Rect* r_rect | the second rectangle |

### Function

GFX_Rect GFX_RectCombine(const GFX_Rect* l_rect,
const GFX_Rect* r_rect)

## GFX_DrawDirectBlit Function

Blits a buffer of pixels into the frame buffer without performing per-pixel operations on the data.

### File

gfx_draw.h

### C

```
LIB_EXPORT GFX_Result GFX_DrawDirectBlit(GFX_PixelBuffer* buffer, int32_t src_x, int32_t src_y, int32_t
src_width, int32_t src_height, int32_t dest_x, int32_t dest_y);
```

### Returns

GFX_Result - Returns GFX_TRUE if the blit was drawn successfully. Otherwise returns GFX_FALSE.

### Description

A pixel buffer is an array of pixel data that can be applied in bulk, or 'blit'ed, to the frame buffer. This method will not perform per-pixel operations on the incoming data. The incoming pixel data color format must match the format of the current target buffer. Area clipping operations are still performed if enabled.

### Parameters

| Parameters | Description |
|---|---|
| buffer | the pointer to the source pixel buffer |
| src_x | the x component of the rectangle of the buffer to blit, usually 0 |
| src_y | the y component of the rectangle of the buffer to blit, usually 0 |

| | |
|---|---|
| src_width | width of the rectangle of the buffer to blit, usually the entire with of the source buffer |
| src_height | height of the rectangle of the buffer to blit, usually the entire height of the source buffer |
| dest_x | the x position to blit the source rectangle in the destination buffer<br>the y position to blit the source rectangle in the destination buffer |

### Function

GFX_Result GFX_DrawDirectBlit(GFX_PixelBuffer* buffer,
int32_t src_x,
int32_t src_y,
int32_t src_width,
int32_t src_height,
int32_t dest_x,
int32_t dest_y);

## GFX_RectCompare Function

### File

gfx_rect.h

### C

```
LIB_EXPORT int32_t GFX_RectCompare(const GFX_Rect* l, const GFX_Rect* r);
```

### Description

This is function GFX_RectCompare.

## GFX_RectsAreSimilar Function

### File

gfx_rect.h

### C

```
LIB_EXPORT GFX_Bool GFX_RectsAreSimilar(const GFX_Rect* l, const GFX_Rect* r);
```

### Description

This is function GFX_RectsAreSimilar.

## GFX_ScaleIntegerSigned Function

Scales a signed integer from one number range of 0 -> n0 to another range 0 -> n1 based on percentages.

### File

gfx_math.h

### C

```
LIB_EXPORT int32_t GFX_ScaleIntegerSigned(int32_t num, int32_t oldMax, int32_t newMax);
```

### Returns

int32_t - the number as defined in the new number range

### Parameters

| Parameters | Description |
|---|---|
| num | the number to consider |
| oldMax | the old range maximum |
| newMax | the new range maximum |

### Function

int32_t GFX_ScaleIntegerSigned(int32_t num, int32_t oldMax, int32_t newMax);

### *b) Data Types and Constants*

#### GFX_ColorMode_t Enumeration

List of available color modes.

**File**

gfx_color.h

**C**

```
enum GFX_ColorMode_t {
  GFX_COLOR_MODE_GS_8 = 0x0,
  GFX_COLOR_MODE_RGB_332,
  GFX_COLOR_MODE_RGB_565,
  GFX_COLOR_MODE_RGBA_5551,
  GFX_COLOR_MODE_RGB_888,
  GFX_COLOR_MODE_RGBA_8888,
  GFX_COLOR_MODE_ARGB_8888,
  GFX_COLOR_MODE_YUV,
  GFX_COLOR_MODE_INDEX_1,
  GFX_COLOR_MODE_INDEX_4,
  GFX_COLOR_MODE_INDEX_8,
  GFX_COLOR_MODE_LAST = GFX_COLOR_MODE_INDEX_8
};
```

**Description**

Enumeration: GFX_ColorMode_t

#### GFX_DriverInfo_t Structure

A driver description structure.

**File**

gfx_driver_interface.h

**C**

```
struct GFX_DriverInfo_t {
  char name[16];
  GFX_ColorMask color_formats;
  uint32_t layer_count;
};
```

**Description**

Structure: GFX_DriverInfo_t

name - a short human-readable name. color formats - a mask of supported color formats layer_count - number of layers supported by the driver

**Remarks**

None.

#### GFX_Point_t Structure

A two dimensional Cartesian point.

**File**

gfx_common.h

**C**

```
struct GFX_Point_t {
  int32_t x;
  int32_t y;
};
```

### Description

Structure: GFX_Point_t

## GFX_Rect_t Structure

A rectangle definition.

### File

gfx_common.h

### C

```
struct GFX_Rect_t {
  int32_t x;
  int32_t y;
  int32_t width;
  int32_t height;
};
```

### Description

Structure: GFX_Rect_t

## begin_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* begin_FnPtr)(void);
```

### Description

This is type begin_FnPtr.

## boolGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Bool (* boolGet_FnPtr)(void);
```

### Description

This is type boolGet_FnPtr.

## boolSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* boolSet_FnPtr)(GFX_Bool);
```

### Description

This is type boolSet_FnPtr.

## brightnessGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef uint32_t (* brightnessGet_FnPtr)(void);
```

### Description

This is type brightnessGet_FnPtr.

## brightnessRangeGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* brightnessRangeGet_FnPtr)(uint32_t*, uint32_t*);
```

### Description

This is type brightnessRangeGet_FnPtr.

## brightnessSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* brightnessSet_FnPtr)(uint32_t);
```

### Description

This is type brightnessSet_FnPtr.

## colorModeGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_ColorMode (* colorModeGet_FnPtr)(void);
```

### Description

This is type colorModeGet_FnPtr.

## colorModeSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* colorModeSet_FnPtr)(GFX_ColorMode);
```

### Description

This is type colorModeSet_FnPtr.

## destroy_FnPtr Type

### File

gfx_hal.h

### C

```
typedef void (* destroy_FnPtr)(GFX_Context*);
```

### Description

This is type destroy_FnPtr.

## drawAlphaValueGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef uint32_t (* drawAlphaValueGet_FnPtr)(void);
```

### Description

This is type drawAlphaValueGet_FnPtr.

## drawAlphaValueSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* drawAlphaValueSet_FnPtr)(uint32_t);
```

### Description

This is type drawAlphaValueSet_FnPtr.

## drawBlendModeGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_BlendMode (* drawBlendModeGet_FnPtr)(void);
```

### Description

This is type drawBlendModeGet_FnPtr.

## drawBlendModeSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* drawBlendModeSet_FnPtr)(GFX_BlendMode);
```

### Description

This is type drawBlendModeSet_FnPtr.

## drawBlit_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* drawBlit_FnPtr)(const GFX_PixelBuffer*, const GFX_Rect*, const GFX_Point*, const
GFX_DrawState*);
```

### Description

This is type drawBlit_FnPtr.

### drawCircle_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawCircle_FnPtr)(const GFX_Point*, int32_t, const GFX_DrawState*);
```

#### Description

This is type drawCircle_FnPtr.

### drawClipRectGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawClipRectGet_FnPtr)(GFX_Rect*);
```

#### Description

This is type drawClipRectGet_FnPtr.

### drawClipRectSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawClipRectSet_FnPtr)(const GFX_Rect*);
```

#### Description

This is type drawClipRectSet_FnPtr.

### drawColorGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Color (* drawColorGet_FnPtr)(void);
```

#### Description

This is type drawColorGet_FnPtr.

### drawColorSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawColorSet_FnPtr)(GFX_Color);
```

#### Description

This is type drawColorSet_FnPtr.

### drawGradientColorGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef void (* drawGradientColorGet_FnPtr)(GFX_Color*, GFX_Color*, GFX_Color*, GFX_Color*);
```

#### Description

This is type drawGradientColorGet_FnPtr.

### drawGradientColorSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawGradientColorSet_FnPtr)(GFX_Color, GFX_Color, GFX_Color, GFX_Color);
```

#### Description

This is type drawGradientColorSet_FnPtr.

### drawLine_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawLine_FnPtr)(const GFX_Point*, const GFX_Point*, const GFX_DrawState*);
```

#### Description

This is type drawLine_FnPtr.

### drawLock_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawLock_FnPtr)(void);
```

#### Description

This is type drawLock_FnPtr.

### drawMaskValueGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef uint32_t (* drawMaskValueGet_FnPtr)(void);
```

#### Description

This is type drawMaskValueGet_FnPtr.

### drawMaskValueSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawMaskValueSet_FnPtr)(uint32_t);
```

#### Description

This is type drawMaskValueSet_FnPtr.

### drawModeGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_DrawMode (* drawModeGet_FnPtr)(void);
```

#### Description

This is type drawModeGet_FnPtr.

### drawModeSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawModeSet_FnPtr)(GFX_DrawMode);
```

#### Description

This is type drawModeSet_FnPtr.

### drawPaletteGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawPaletteGet_FnPtr)(GFX_PixelBuffer*);
```

#### Description

This is type drawPaletteGet_FnPtr.

### drawPaletteSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawPaletteSet_FnPtr)(const GFX_PixelBuffer*);
```

#### Description

This is type drawPaletteSet_FnPtr.

### drawPixel_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawPixel_FnPtr)(const GFX_Point*, const GFX_DrawState*);
```

#### Description

This is type drawPixel_FnPtr.

### drawRect_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawRect_FnPtr)(const GFX_Rect*, const GFX_DrawState*);
```

#### Description

This is type drawRect_FnPtr.

### drawThicknessGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef uint32_t (* drawThicknessGet_FnPtr)(void);
```

#### Description

This is type drawThicknessGet_FnPtr.

### drawThicknessSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawThicknessSet_FnPtr)(uint32_t);
```

#### Description

This is type drawThicknessSet_FnPtr.

### drawUnlock_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawUnlock_FnPtr)(void);
```

#### Description

This is type drawUnlock_FnPtr.

### GFX_AntialiasMode Enumeration

Enables anti-aliased drawing hint

### File

gfx_draw.h

### C

```
typedef enum GFX_AntialiasMode_t {
  GFX_ANTIALIAS_OFF = 0x0,
  GFX_ANTIALIAS_ON = 0x1
} GFX_AntialiasMode;
```

### Description

Enumeration: GFX_AntialiasMode_t

### Remarks

None.

## GFX_BitsPerPixel Enumeration

List of available bits-per-pixel sizes.

### File

gfx_color.h

### C

```
typedef enum GFX_BitsPerPixel_t {
  GFX_BPP1,
  GFX_BPP4,
  GFX_BPP8,
  GFX_BPP16,
  GFX_BPP24,
  GFX_BPP32
} GFX_BitsPerPixel;
```

### Description

Enumeration: GFX_BitsPerPixel_t

## GFX_BlendMode Enumeration

Blend mode masks

### File

gfx_common.h

### C

```
typedef enum GFX_BlendMode_t {
  GFX_BLEND_NONE = 0x0,
  GFX_BLEND_CHANNEL = 0x1,
  GFX_BLEND_GLOBAL = 0x2,
  GFX_BLEND_ALL = GFX_BLEND_CHANNEL|GFX_BLEND_GLOBAL
} GFX_BlendMode;
```

### Description

Enumeration: GFX_BlendMode_t

## GFX_Bool Type

### File

gfx_common.h

### C

```
typedef uint32_t GFX_Bool;
```

### Description

This is type GFX_Bool.

## GFX_Buffer Type

### File

gfx_common.h

### C

```
typedef void* GFX_Buffer;
```

### Description

This is type GFX_Buffer.

## GFX_BufferSelection Enumeration

Buffer selector used when querying layers for certain buffer states.

### File

gfx_common.h

### C

```
typedef enum GFX_BufferSelection_t {
  GFX_BUFFER_READ,
  GFX_BUFFER_WRITE
} GFX_BufferSelection;
```

### Description

Enumeration: GFX_BufferSelection_t

## GFX_BufferState Enumeration

Frame buffer memory states

### File

gfx_common.h

### C

```
typedef enum GFX_BufferState_t {
  GFX_BS_NONE = 0x0,
  GFX_BS_ADDRESS,
  GFX_BS_MALLOC,
  GFX_BS_MANAGED
} GFX_BufferState;
```

### Description

Enumeration: GFX_BufferState_t

address - The buffer is set to a discrete address. This could be an address located in DDR memory, a buffer allocated by the application, or some other location.

malloc - The buffer has been dynamically allocated from some form of heap or memory manager. These can be freed as desired.

managed - The buffer is owned by the system and cannot be allocated or freed, it just is. This is common in systems where the memory is owned by the graphics driver or the memory resides on the graphics controller.

### Remarks

None.

## GFX_Calloc_FnPtr Type

Simple wrapper around the standard calloc function pointer. Used for redirecting memory allocation to other memory management systems.

### File

gfx_common.h

### C

```
typedef void* (* GFX_Calloc_FnPtr)(size_t, size_t);
```

### Description

Function pointer

### Function

typedef void* (*GFX_Calloc_FnPtr)(size_t, size_t);

## GFX_Color Type

### File

gfx_common.h

### C

```
typedef uint32_t GFX_Color;
```

### Description

This is type GFX_Color.

## GFX_ColorMask Enumeration

Maskable list of color valies.

### File

gfx_color.h

### C

```
typedef enum GFX_ColorMask_t {
  GFX_COLOR_MASK_GS_8 = 0x1,
  GFX_COLOR_MASK_RGB_332 = 0x4,
  GFX_COLOR_MASK_RGB_565 = 0x8,
  GFX_COLOR_MASK_RGBA_5551 = 0x10,
  GFX_COLOR_MASK_RGB_888 = 0x20,
  GFX_COLOR_MASK_RGBA_8888 = 0x40,
  GFX_COLOR_MASK_ARGB_8888 = 0x80,
  GFX_COLOR_MASK_YUV = 0x100,
  GFX_COLOR_MASK_ALL =
GFX_COLOR_MASK_GS_8|GFX_COLOR_MASK_RGB_332|GFX_COLOR_MASK_RGB_565|GFX_COLOR_MASK_RGBA_5551|GFX_COLOR_MASK_RG
B_888|GFX_COLOR_MASK_RGBA_8888|GFX_COLOR_MASK_ARGB_8888|GFX_COLOR_MASK_YUV
} GFX_ColorMask;
```

### Description

Enumeration: GFX_ColorMask_t

## GFX_ColorMode Type

List of available color modes.

### File

gfx_common.h

### C

```
typedef enum GFX_ColorMode_t GFX_ColorMode;
```

### Description

Enumeration: GFX_ColorMode_t

## GFX_ColorModeInfo Structure

Struct that provides information about a color mode.

### File

gfx_color.h

### C

```
typedef struct GFX_ColorModeInfo_t {
  uint32_t size;
  uint32_t bpp;
  GFX_BitsPerPixel bppOrdinal;
  struct masks {
    uint32_t red;
    uint32_t green;
    uint32_t blue;
    uint32_t alpha;
  } mask;
  struct shifts {
    uint8_t red;
    uint8_t green;
    uint8_t blue;
    uint8_t alpha;
  } shift;
} GFX_ColorModeInfo;
```

### Description

Structure: GFX_ColorModeInfo_t

size - size in bytes bpp - bpp value bppOrdinal - bpp enum value masks - the masks used for extracting individual color channel information

### Remarks

None.

## GFX_ColorName Enumeration

Color name reference table

### File

gfx_color.h

### C

```
typedef enum GFX_ColorName_t {
  GFX_COLOR_BLACK,
  GFX_COLOR_WHITE,
  GFX_COLOR_RED,
  GFX_COLOR_LIME,
  GFX_COLOR_BLUE,
  GFX_COLOR_YELLOW,
  GFX_COLOR_CYAN,
  GFX_COLOR_MAGENTA,
  GFX_COLOR_SILVER,
  GFX_COLOR_DARKGRAY,
  GFX_COLOR_GRAY,
  GFX_COLOR_LIGHTGRAY,
  GFX_COLOR_MAROON,
  GFX_COLOR_OLIVE,
  GFX_COLOR_GREEN,
  GFX_COLOR_PURPLE,
  GFX_COLOR_TEAL,
  GFX_COLOR_NAVY,
  GFX_COLOR_LAST
} GFX_ColorName;
```

### Description

Structure: GFX_ColorName_t

## GFX_Context Structure

An instance of the hardware abstraction layer.

### File

gfx_context.h

## C

```
typedef struct GFX_Context_t {
  GFX_Display display_idx;
  GFX_DisplayInfo* display_info;
  struct {
    uint32_t count;
    uint32_t active_idx;
    GFX_Layer* active;
    GFX_Layer* layers;
  } layer;
  uint32_t brightness;
  GFX_Orientation orientation;
  GFX_Bool mirrored;
  GFX_Bool layerSwapSync;
  GFX_ColorMode colorMode;
  GFX_GlobalPalette globalPalette;
  GFX_DrawState draw;
  GFX_SyncCallback_FnPtr vsyncCB;
  GFX_SyncCallback_FnPtr hsyncCB;
  GFX_HAL hal;
  GFX_MemoryIntf memory;
  void* driver_data;
} GFX_Context;
```

## Members

| Members | Description |
|---|---|
| GFX_SyncCallback_FnPtr vsyncCB; | GFX_DRAW_PIPELINE_ENABLED |

## Description

Stucture: GFX_Context_t

The context is an instance of the hardware abstraction layer. It is essentially the marriage between a graphics driver, a display description, and possibly a graphics processor. It contains the data that describes the layout of the display, the buffers that store the display data, the current draw state, and the function map that controls everything.

Members: display_idx - the display associated with this context display_info - a pointer to the information for the display for this context

layer.count - the number of existing layers layer.active_idx - the index of the active layer layer.active - the pointer to the active layer layer.layers - the array of layers for this context

brightness - the brightness setting for this context orientation - the orientation mode for this context mirrored - the mirror mode for this context

colorMode - the color mode for this context, all buffers of all layers use this mode

draw - the current draw state of this context

vsyncCB - the callback to invoke when the driver enters vsync mode hsyncCB - the callback to invoke when the driver enters hsync mode

hal - the function table for this context

memory - the memory management interface for this context

driver_data - a pointer that can be used for driver-specific data purposes

## Remarks

None.

### GFX_Display Type

## File

gfx_common.h

## C

```
typedef uint32_t GFX_Display;
```

## Description

This is type GFX_Display.

### GFX_DisplayInfo Structure

Describes a graphical display device.

### File

gfx_display.h

### C

```
typedef struct GFX_DisplayInfo_t {
  const char name[16];
  GFX_ColorMask color_formats;
  GFX_Rect rect;
  struct attributes_t {
    int8_t data_width;
    struct horizontal_t {
      int8_t pulse_width;
      int8_t back_porch;
      int8_t front_porch;
    } horz;
    struct vertical_t {
      int8_t pulse_width;
      int8_t back_porch;
      int8_t front_porch;
    } vert;
    int32_t inv_left_shift;
  } attributes;
} GFX_DisplayInfo;
```

### Description

Structure: GFX_DisplayInfo_t

name - a short human-readable name color_formats - mask of color formats this display supports rect - the size of the display

### Remarks

None.

## GFX_DrawMode Enumeration

Gradient draw modes.

### File

gfx_draw.h

### C

```
typedef enum GFX_DrawMode_t {
  GFX_DRAW_LINE = 0x0,
  GFX_DRAW_FILL,
  GFX_DRAW_GRADIENT_LEFT_RIGHT,
  GFX_DRAW_GRADIENT_TOP_BOTTOM
} GFX_DrawMode;
```

### Description

Enumeration: GFX_DrawMode_t

line - draws the outline of a shape fill - draws a filled shape gradient left/right - draws a gradient from left to right, uses the first two gradient colors gradient top/bottom - draws a gradient from top to bottom, uses the first two gradient colors

### Remarks

None.

## GFX_DrawState Structure

A list of drawing hints for shape drawing algorithms

### File

gfx_draw.h

### C

```
typedef struct GFX_DrawState_t {
  GFX_DrawMode mode;
  GFX_Color color;
  GFX_ColorMode colorMode;
```

```
    struct {
      GFX_Color c0;
      GFX_Color c1;
      GFX_Color c2;
      GFX_Color c3;
    } gradient;
    GFX_PixelBuffer palette;
    const GFX_PixelBuffer* target;
    GFX_Rect targetClipRect;
    GFX_BlendMode blendMode;
    GFX_Bool alphaEnable;
    uint32_t globalAlphaValue;
    GFX_Bool maskEnable;
    uint32_t maskValue;
    GFX_Bool antialias;
    uint32_t thickness;
    GFX_Bool clipEnable;
    GFX_Rect clipRect;
    GFX_ResizeMode resizeMode;
    GFX_PipelineMode pipelineMode;
    GFX_DrawPipeline* pipeline;
  } GFX_DrawState;
```

## Description

Structure: GFX_DrawState_t

mode - the shape drawing mode

color - the draw color

gradient - the list of gradient colors

palette - the palette lookup table for blits

alphaEnable - indicates if alpha blending is enabled alphaValue - the desired alpha blending amount

maskEnable - indicates if pixel masking is enabled maskValue - the mask/transparency color value

clipEnable - indicate of pixel clipping is enabled clipRect - the pixel clipping rectangle

## Remarks

None.

### GFX_Driver Type

## File

gfx_common.h

## C

```
typedef uint32_t GFX_Driver;
```

## Description

This is type GFX_Driver.

### GFX_DriverInfo Type

A driver description structure.

## File

gfx_hal.h

## C

```
typedef struct GFX_DriverInfo_t GFX_DriverInfo;
```

## Description

Structure: GFX_DriverInfo_t

name - a short human-readable name. color formats - a mask of supported color formats layer_count - number of layers supported by the driver

## Remarks

None.

## GFX_Flag Enumeration

Hardware abstraction state interface flags. See gfx.h for a comprehensive description.

### File

gfx_common.h

### C

```
typedef enum GFX_Flag_t {
  GFXF_NONE = 0,
  GFXF_DISPLAY_COUNT,
  GFXF_DISPLAY_INFO,
  GFXF_DRIVER_COUNT,
  GFXF_DRIVER_INFO,
  GFXF_BRIGHTNESS_RANGE,
  GFXF_BRIGHTNESS,
  GFXF_VSYNC_CALLBACK,
  GFXF_HSYNC_CALLBACK,
  GFXF_ORIENTATION,
  GFXF_MIRRORED,
  GFXF_COLOR_MODE,
  GFXF_GLOBAL_PALETTE,
  GFXF_LAYER_COUNT,
  GFXF_LAYER_ACTIVE,
  GFXF_LAYER_ENABLED,
  GFXF_LAYER_VISIBLE,
  GFXF_LAYER_VSYNC,
  GFXF_LAYER_INVALID,
  GFXF_LAYER_SWAP_SYNC,
  GFXF_LAYER_SWAP,
  GFXF_LAYER_POSITION,
  GFXF_LAYER_SIZE,
  GFXF_LAYER_ALPHA_ENABLE,
  GFXF_LAYER_ALPHA_AMOUNT,
  GFXF_LAYER_MASK_ENABLE,
  GFXF_LAYER_MASK_COLOR,
  GFXF_LAYER_BUFFER_COUNT,
  GFXF_LAYER_BUFFER_ADDRESS,
  GFXF_LAYER_BUFFER_COHERENT,
  GFXF_LAYER_BUFFER_ALLOCATE,
  GFXF_LAYER_BUFFER_FREE,
  GFXF_DRAW_PIPELINE_MODE,
  GFXF_DRAW_MODE,
  GFXF_DRAW_COLOR,
  GFXF_DRAW_GRADIENT_COLOR,
  GFXF_DRAW_PALETTE,
  GFXF_DRAW_TARGET,
  GFXF_DRAW_THICKNESS,
  GFXF_DRAW_BLEND_MODE,
  GFXF_DRAW_RESIZE_MODE,
  GFXF_DRAW_ALPHA_ENABLE,
  GFXF_DRAW_ALPHA_VALUE,
  GFXF_DRAW_MASK_ENABLE,
  GFXF_DRAW_MASK_VALUE,
  GFXF_DRAW_CLIP_ENABLE,
  GFXF_DRAW_CLIP_RECT,
  GFXF_LAST_FLAG
} GFX_Flag;
```

### Description

Enumeration: GFX_Flag_t

## GFX_FrameBuffer Structure

A frame buffer is a wrapper around a pixel buffer construct that is used by display drivers to manage frame buffers.

### File

gfx_layer.h

### C

```
typedef struct GFX_FrameBuffer_t {
  GFX_PixelBuffer pb;
  GFX_BufferState state;
  GFX_Bool coherent;
  void* driver_data;
} GFX_FrameBuffer;
```

### Description

Structure: GFX_FrameBuffer_t

pb - The pixel buffer description of the frame buffer. state - The state of the frame buffer. coherent - Indicates if the frame buffer is allocated from coherent dynamic memory

### Remarks

None.

## GFX_Free_FnPtr Type

Simple wrapper around the standard free function pointer. Used for redirecting memory free to other memory management systems.

### File

gfx_common.h

### C

```
typedef void (* GFX_Free_FnPtr)(void*);
```

### Description

Function pointer

### Function

typedef void  (*GFX_Free_FnPtr)(void*);

## GFX_HAL Structure

### File

gfx_hal.h

### C

```
typedef struct GFX_HAL_t {
  initialize_FnPtr initialize;
  destroy_FnPtr destroy;
  begin_FnPtr begin;
  end_FnPtr end;
  update_FnPtr update;
  brightnessRangeGet_FnPtr brightnessRangeGet;
  brightnessGet_FnPtr brightnessGet;
  brightnessSet_FnPtr brightnessSet;
  syncCallbackGet_FnPtr vsyncCallbackGet;
  syncCallbackSet_FnPtr vsyncCallbackSet;
  syncCallbackGet_FnPtr hsyncCallbackGet;
  syncCallbackSet_FnPtr hsyncCallbackSet;
  orientationGet_FnPtr orientationGet;
  orientationSet_FnPtr orientationSet;
  boolGet_FnPtr mirroringGet;
  boolSet_FnPtr mirroringSet;
  colorModeGet_FnPtr colorModeGet;
  colorModeSet_FnPtr colorModeSet;
  globalPaletteGet_FnPtr globalPaletteGet;
  globalPaletteSet_FnPtr globalPaletteSet;
  layerActiveGet_FnPtr layerActiveGet;
  layerActiveSet_FnPtr layerActiveSet;
  boolGet_FnPtr layerEnabledGet;
  boolSet_FnPtr layerEnabledSet;
  layerBufferCountGet_FnPtr layerBufferCountGet;
  layerBufferCountSet_FnPtr layerBufferCountSet;
  layerBufferAddressGet_FnPtr layerBufferAddressGet;
```

```
    layerBufferAddressSet_FnPtr layerBufferAddressSet;
    layerBufferCoherentGet_FnPtr layerBufferCoherentGet;
    layerBufferCoherentSet_FnPtr layerBufferCoherentSet;
    layerBufferAllocate_FnPtr layerBufferAllocate;
    layerBufferIsAllocated_FnPtr layerBufferIsAllocated;
    layerBufferFree_FnPtr layerBufferFree;
    boolGet_FnPtr layerVisibleGet;
    boolSet_FnPtr layerVisibleSet;
    boolGet_FnPtr layerVsyncGet;
    boolSet_FnPtr layerVsyncSet;
    boolGet_FnPtr layerInvalidGet;
    boolSet_FnPtr layerInvalidSet;
    boolGet_FnPtr layerSwapSyncGet;
    boolSet_FnPtr layerSwapSyncSet;
    boolGet_FnPtr layerSwapGet;
    boolSet_FnPtr layerSwapSet;
    layerSwapPending_FnPtr layerSwapPending;
    layerSwapped_FnPtr layerSwapped;
    layerPositionGet_FnPtr layerPositionGet;
    layerPositionSet_FnPtr layerPositionSet;
    layerSizeGet_FnPtr layerSizeGet;
    layerSizeSet_FnPtr layerSizeSet;
    boolGet_FnPtr layerAlphaEnableGet;
    layerEffectSet_FnPtr layerAlphaEnableSet;
    layerAlphaAmountGet_FnPtr layerAlphaAmountGet;
    layerAlphaAmountSet_FnPtr layerAlphaAmountSet;
    layerMaskColorGet_FnPtr layerMaskEnableGet;
    layerMaskColorSet_FnPtr layerMaskEnableSet;
    boolGet_FnPtr layerMaskColorGet;
    layerEffectSet_FnPtr layerMaskColorSet;
    orientPoint_FnPtr orientPoint;
    mirrorPoint_FnPtr mirrorPoint;
    drawPipelineModeGet_FnPtr drawPipelineModeGet;
    drawPipelineModeSet_FnPtr drawPipelineModeSet;
    drawModeGet_FnPtr drawModeGet;
    drawModeSet_FnPtr drawModeSet;
    drawColorGet_FnPtr drawColorGet;
    drawColorSet_FnPtr drawColorSet;
    drawGradientColorGet_FnPtr drawGradientColorGet;
    drawGradientColorSet_FnPtr drawGradientColorSet;
    drawPaletteGet_FnPtr drawPaletteGet;
    drawPaletteSet_FnPtr drawPaletteSet;
    drawTargetGet_FnPtr drawTargetGet;
    drawTargetSet_FnPtr drawTargetSet;
    drawBlendModeGet_FnPtr drawBlendModeGet;
    drawBlendModeSet_FnPtr drawBlendModeSet;
    drawResizeModeGet_FnPtr drawResizeModeGet;
    drawResizeModeSet_FnPtr drawResizeModeSet;
    drawAlphaEnableGet_FnPtr drawAlphaEnableGet;
    drawAlphaEnableSet_FnPtr drawAlphaEnableSet;
    drawAlphaValueGet_FnPtr drawAlphaValueGet;
    drawAlphaValueSet_FnPtr drawAlphaValueSet;
    boolGet_FnPtr drawMaskEnableGet;
    boolSet_FnPtr drawMaskEnableSet;
    drawMaskValueGet_FnPtr drawMaskValueGet;
    drawMaskValueSet_FnPtr drawMaskValueSet;
    maskColor_FnPtr maskColor;
    boolGet_FnPtr drawAntialiasGet;
    boolSet_FnPtr drawAntialiasSet;
    drawThicknessGet_FnPtr drawThicknessGet;
    drawThicknessSet_FnPtr drawThicknessSet;
    boolGet_FnPtr drawClipEnableGet;
    boolSet_FnPtr drawClipEnableSet;
    drawClipRectGet_FnPtr drawClipRectGet;
    drawClipRectSet_FnPtr drawClipRectSet;
    blendGetPoint_FnPtr alphaGetPoint;
    blendColor_FnPtr alphaChannelBlend;
    blendColor_FnPtr globalAlphaBlend;
    GFX_DrawPipeline drawPipeline[GFX_PIPELINE_MODE_COUNT];
    interrupt_FnPtr interrupt;
} GFX_HAL;
```

## Members

| Members | Description |
|---|---|
| interrupt_FnPtr interrupt; | GFX_DRAW_PIPELINE_ENABLED |

## Description

Hardware Abstraction Function Table *

- This is the core hardware abtraction table that makes everything work. Drivers are
- expected to reroute the functionality of this table to hardware specific implementations
- to provide accelerated performance and features.

### GFX_Handle Type

## File

gfx_common.h

## C

```
typedef void* GFX_Handle;
```

## Description

This is type GFX_Handle.

### GFX_Layer Structure

Layers describe basic display drawing areas and are meant to map directly to graphics controller hardware layers.

## File

gfx_layer.h

## C

```
typedef struct GFX_Layer_t {
  uint32_t id;
  struct {
    GFX_Rect display;
    GFX_Rect local;
  } rect;
  uint32_t pixel_size;
  GFX_Bool alphaEnable;
  uint32_t alphaAmount;
  GFX_Bool maskEnable;
  uint32_t maskColor;
  uint32_t buffer_count;
  uint32_t buffer_read_idx;
  uint32_t buffer_write_idx;
  GFX_FrameBuffer buffers[GFX_MAX_BUFFER_COUNT];
  GFX_Bool enabled;
  GFX_Bool visible;
  GFX_Bool swap;
  uint32_t swapCount;
  GFX_Bool locked;
  GFX_Bool invalid;
  GFX_Bool vsync;
  void* driver_data;
} GFX_Layer;
```

## Members

| Members | Description |
|---|---|
| GFX_Rect display; | represents area in display space |
| GFX_Rect local; | represents position in local space |

## Description

Structure: GFX_Layer_t

Graphics controllers will typically offer at least one drawing layer for drawing purposes. More advanced controllers may offer several. Layers are often configurable, offering independent positioning, sizing, color formats, and drawing features.

uint32_t id - the unique id of the layer

struct { GFX_Rect display - represents area in display space GFX_Rect local - represents position in local space } rect;

uint32_t pixel_size - size of a layer pixel in bytes

GFX_Bool alphaEnable - indicates if layer alpha blending is enabled uint32_t alphaAmount - indicates the amount of alpha blending

GFX_Bool maskEnable - indicates if layer masking/transparency is enabled uint32_t maskColor - the color to mask

uint32_t buffer_count - the number of buffers this layer owns uint32_t buffer_read_idx - the index of the current read buffer uint32_t buffer_write_idx - the index of the current write buffer GFX_FrameBuffer buffers[GFX_MAX_BUFFER_COUNT] - the layer buffer array

GFX_Bool enabled - indicates if the layer is enabled GFX_Bool visible - indicates if the layer is visible

GFX_Bool swap - indicates if the layer is waiting to advance its buffer chain

GFX_Bool locked - indicates if the layer's buffers are locked for manipulation. this is typically meant to prevent things like swapping before drawing operations have been completed

GFX_Bool vsync - indicates if this layer should swap during the display driver's blanking period. if this is true then it is the responsibility of the display driver to call GFX_LayerSwap on this layer during vblank. If this is false then the layer will swap immediately upon application request.

void* driver_data - this is a pointer that may be allocated by the display driver to store driver-specific per layer data. the driver is responsible for the management of this pointer during the application life cycle

### Remarks

None.

## GFX_Malloc_FnPtr Type

Simple wrapper around the standard malloc function pointer. Used for redirecting memory allocation to other memory management systems.

### File

gfx_common.h

### C

```
typedef void* (* GFX_Malloc_FnPtr)(size_t);
```

### Description

memory abstraction

**********

Function pointer

### Function

typedef void* (*GFX_Malloc_FnPtr)(size_t);

## GFX_Memcpy_FnPtr Type

Simple wrapper around the standard memcpy function pointer. Used for redirecting memcpy to other memory management systems.

### File

gfx_common.h

### C

```
typedef void* (* GFX_Memcpy_FnPtr)(void*, const void*, size_t);
```

### Description

Function pointer

### Function

typedef void* (*GFX_Memcpy_FnPtr)(void*, const void*, size_t);

## GFX_MemoryIntf Structure

Custom memory manager interface.

### File

gfx_common.h

### C

```
typedef struct GFX_MemoryIntf_t {
  GFX_Malloc_FnPtr malloc;
  GFX_Malloc_FnPtr coherent_alloc;
  GFX_Calloc_FnPtr calloc;
  GFX_Realloc_FnPtr realloc;
  GFX_Free_FnPtr free;
  GFX_Free_FnPtr coherent_free;
  GFX_Memset_FnPtr memset;
  GFX_Memcpy_FnPtr memcpy;
} GFX_MemoryIntf;
```

### Description

Structure: GFX_MemoryIntf_t

Applications utilizing the hardware abstraction layer may want to implement or utilize memory managers other than the standard library. This interface is the method for notifying the HAL of that manager.

The application must create a GFX_MemoryIntf struct, populate it with the fuction pointers that point to the custom memory manager, and pass the struct in to GFX_Open when the HAL context is created.

If no GFX_MemoryIntf is provided then the standard library memory management APIs will be used by default.

### Remarks

None.

## GFX_Memset_FnPtr Type

Simple wrapper around the standard memset function pointer. Used for redirecting memset to other memory management systems.

### File

gfx_common.h

### C

```
typedef void* (* GFX_Memset_FnPtr)(void*, int32_t, size_t);
```

### Description

Function pointer

### Function

typedef void* (*GFX_Memset_FnPtr)(void*, int32_t, size_t);

## GFX_Orientation Enumeration

Orthogonal orientation settings.

### File

gfx_common.h

### C

```
typedef enum GFX_Orientation_t {
  GFX_ORIENTATION_0 = 0x0,
  GFX_ORIENTATION_90,
  GFX_ORIENTATION_180,
  GFX_ORIENTATION_270
} GFX_Orientation;
```

### Description

Enumeration: GFX_Orientation_t

## GFX_PixelBuffer Structure

A pixel buffer is a wrapper around a basic data pointer. A pixel buffer has a color mode, a pixel count, a rectangular dimension, a pixel count, and a lenght in bytes.

### File

gfx_pixel_buffer.h

### C

```
typedef struct GFX_PixelBuffer_t {
  GFX_ColorMode mode;
  GFX_Size size;
  int32_t pixel_count;
  uint32_t buffer_length;
  GFX_Buffer pixels;
} GFX_PixelBuffer;
```

### Description

Structure: GFX_PixelBuffer_t

mode - the color mode of the pixel buffer size - the width and height dimension of the pixel buffer pixel_count - the total number of pixels in the buffer buffer_length - the total size of the buffer in bytes pixels - the pointer to the pixel data for the buffer

### Remarks

None.

## end_FnPtr Type

### File

gfx_hal.h

### C

```
typedef void (* end_FnPtr)(void);
```

### Description

This is type end_FnPtr.

## GFX_Processor Type

### File

gfx_common.h

### C

```
typedef uint32_t GFX_Processor;
```

### Description

This is type GFX_Processor.

## GFX_Realloc_FnPtr Type

Simple wrapper around the standard realloc function pointer. Used for redirecting memory allocation to other memory management systems.

### File

gfx_common.h

### C

```
typedef void* (* GFX_Realloc_FnPtr)(void*, size_t);
```

### Description

Function pointer

### Function

typedef void* (*GFX_Realloc_FnPtr)(void*, size_t);

## GFX_Result Type

### File

gfx_common.h

### C

```
typedef int32_t GFX_Result;
```

### Description

This is type GFX_Result.

## GFX_Size Structure

A two dimensional indication of size. Values are signed but should never be negative.

### File

gfx_common.h

### C

```
typedef struct GFX_Size_t {
  int32_t width;
  int32_t height;
} GFX_Size;
```

### Description

Structure: GFX_Size_t

## GFX_SyncCallback_FnPtr Type

### File

gfx_common.h

### C

```
typedef void (* GFX_SyncCallback_FnPtr)(void);
```

### Description

This is type GFX_SyncCallback_FnPtr.

## layerActiveGet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef uint32_t (* layerActiveGet_FnPtr)(void);
```

### Description

This is type layerActiveGet_FnPtr.

## layerActiveSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* layerActiveSet_FnPtr)(uint32_t);
```

### Description

This is type layerActiveSet_FnPtr.

### layerAlphaAmountGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef uint32_t (* layerAlphaAmountGet_FnPtr)(void);
```

#### Description

This is type layerAlphaAmountGet_FnPtr.

### layerAlphaAmountSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerAlphaAmountSet_FnPtr)(uint32_t, GFX_Bool);
```

#### Description

This is type layerAlphaAmountSet_FnPtr.

### layerBufferAddressGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Buffer (* layerBufferAddressGet_FnPtr)(uint32_t);
```

#### Description

This is type layerBufferAddressGet_FnPtr.

### layerBufferAddressSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerBufferAddressSet_FnPtr)(uint32_t, GFX_Buffer);
```

#### Description

This is type layerBufferAddressSet_FnPtr.

### layerBufferAllocate_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerBufferAllocate_FnPtr)(uint32_t);
```

#### Description

This is type layerBufferAllocate_FnPtr.

### layerBufferCoherentGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Bool (* layerBufferCoherentGet_FnPtr)(uint32_t);
```

#### Description

This is type layerBufferCoherentGet_FnPtr.

### layerBufferCoherentSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerBufferCoherentSet_FnPtr)(uint32_t, GFX_Bool);
```

#### Description

This is type layerBufferCoherentSet_FnPtr.

### layerBufferCountGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef uint32_t (* layerBufferCountGet_FnPtr)(void);
```

#### Description

This is type layerBufferCountGet_FnPtr.

### layerBufferCountSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerBufferCountSet_FnPtr)(uint32_t);
```

#### Description

This is type layerBufferCountSet_FnPtr.

### layerBufferFree_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerBufferFree_FnPtr)(uint32_t);
```

#### Description

This is type layerBufferFree_FnPtr.

### layerBufferIsAllocated_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Bool (* layerBufferIsAllocated_FnPtr)(uint32_t);
```

#### Description

This is type layerBufferIsAllocated_FnPtr.

### layerMaskColorGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Color (* layerMaskColorGet_FnPtr)(void);
```

#### Description

This is type layerMaskColorGet_FnPtr.

### layerMaskColorSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerMaskColorSet_FnPtr)(GFX_Color mask, GFX_Bool);
```

#### Description

This is type layerMaskColorSet_FnPtr.

### layerPositionGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerPositionGet_FnPtr)(int32_t*, int32_t*);
```

#### Description

This is type layerPositionGet_FnPtr.

### layerPositionSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerPositionSet_FnPtr)(int32_t, int32_t);
```

#### Description

This is type layerPositionSet_FnPtr.

### layerSizeGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerSizeGet_FnPtr)(int32_t* width, int32_t* height);
```

#### Description

This is type layerSizeGet_FnPtr.

### layerSizeSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* layerSizeSet_FnPtr)(int32_t width, int32_t height);
```

#### Description

This is type layerSizeSet_FnPtr.

### layerSwapped_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef void (* layerSwapped_FnPtr)(GFX_Layer*);
```

#### Description

This is type layerSwapped_FnPtr.

### orientationGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Orientation (* orientationGet_FnPtr)(void);
```

#### Description

This is type orientationGet_FnPtr.

### orientationSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* orientationSet_FnPtr)(GFX_Orientation);
```

#### Description

This is type orientationSet_FnPtr.

### pixelGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Color (* pixelGet_FnPtr)(const GFX_PixelBuffer*, const GFX_Point*);
```

#### Description

This is type pixelGet_FnPtr.

### pixelSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* pixelSet_FnPtr)(const GFX_PixelBuffer*, const GFX_Point*, GFX_Color color);
```

#### Description

This is type pixelSet_FnPtr.

### syncCallbackGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_SyncCallback_FnPtr (* syncCallbackGet_FnPtr)(void);
```

#### Description

This is type syncCallbackGet_FnPtr.

### syncCallbackSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* syncCallbackSet_FnPtr)(GFX_SyncCallback_FnPtr);
```

#### Description

This is type syncCallbackSet_FnPtr.

### syncCallbackSt_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* syncCallbackSt_FnPtr)(GFX_SyncCallback_FnPtr);
```

#### Description

This is type syncCallbackSt_FnPtr.

### update_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* update_FnPtr)(void);
```

#### Description

This is type update_FnPtr.

### GFX_ColorInfo Variable

#### File

gfx_color.h

#### C

```
LIB_EXPORT GFX_ColorModeInfo GFX_ColorInfo[GFX_COLOR_MODE_COUNT];
```

#### Description

This is variable GFX_ColorInfo.

### GFX_Rect_Zero Variable

#### File

gfx_rect.h

#### C

```
const GFX_Rect GFX_Rect_Zero = {0, 0, 0, 0};
```

#### Description

This is variable GFX_Rect_Zero.

### AGBA_8888_ALPHA_MASK Macro

#### File

gfx_color.h

#### C

```
#define AGBA_8888_ALPHA_MASK 0xFF000000
```

#### Description

This is macro AGBA_8888_ALPHA_MASK.

### AGBA_8888_BLUE_MASK Macro

#### File

gfx_color.h

#### C

```
#define AGBA_8888_BLUE_MASK 0xFF
```

#### Description

This is macro AGBA_8888_BLUE_MASK.

### AGBA_8888_GREEN_MASK Macro

#### File

gfx_color.h

#### C

```
#define AGBA_8888_GREEN_MASK 0xFF00
```

#### Description

This is macro AGBA_8888_GREEN_MASK.

### AGBA_8888_RED_MASK Macro

#### File

gfx_color.h

#### C

```
#define AGBA_8888_RED_MASK 0xFF0000
```

#### Description

This is macro AGBA_8888_RED_MASK.

### GFX_ANTIALIAS_MODE_COUNT Macro

#### File

gfx_draw.h

#### C

```
#define GFX_ANTIALIAS_MODE_COUNT (GFX_ANTIALIAS_ON+1)
```

#### Description

This is macro GFX_ANTIALIAS_MODE_COUNT.

### GFX_COLOR_MAX_SIZE Macro

#### File

gfx_color.h

#### C

```
#define GFX_COLOR_MAX_SIZE 4
```

#### Description

This is macro GFX_COLOR_MAX_SIZE.

### GFX_COLOR_MODE_COUNT Macro

#### File

gfx_color.h

#### C

```
#define GFX_COLOR_MODE_COUNT (GFX_COLOR_MODE_LAST + 1)
```

#### Description

This is macro GFX_COLOR_MODE_COUNT.

### GFX_COLOR_MODE_IS_ALPHA Macro

#### File

gfx_color.h

#### C

```
#define GFX_COLOR_MODE_IS_ALPHA(mode) ((mode == GFX_COLOR_MODE_RGBA_5551) || (mode ==
GFX_COLOR_MODE_RGBA_8888) || (mode == GFX_COLOR_MODE_ARGB_8888))
```

#### Description

This is macro GFX_COLOR_MODE_IS_ALPHA.

### GFX_COLOR_MODE_IS_INDEX Macro

#### File

gfx_color.h

#### C

```
#define GFX_COLOR_MODE_IS_INDEX(mode) ((mode >= GFX_COLOR_MODE_INDEX_1) && (mode <= GFX_COLOR_MODE_INDEX_8))
```

#### Description

This is macro GFX_COLOR_MODE_IS_INDEX.

### GFX_COLOR_MODE_IS_PIXEL Macro

#### File

gfx_color.h

#### C

```
#define GFX_COLOR_MODE_IS_PIXEL(mode) ((mode >= GFX_COLOR_MODE_GS_8) && (mode <= GFX_COLOR_MODE_YUV))
```

#### Description

This is macro GFX_COLOR_MODE_IS_PIXEL.

### GFX_COLOR_MODE_LAST_COLOR Macro

#### File

gfx_color.h

#### C

```
#define GFX_COLOR_MODE_LAST_COLOR (GFX_COLOR_MODE_YUV)
```

#### Description

This is macro GFX_COLOR_MODE_LAST_COLOR.

### GFX_DRAW_MODE_COUNT Macro

#### File

gfx_draw.h

#### C

```
#define GFX_DRAW_MODE_COUNT (GFX_DRAW_GRADIENT_TOP_BOTTOM + 1)
```

#### Description

This is macro GFX_DRAW_MODE_COUNT.

### GFX_FAILURE Macro

#### File

gfx_common.h

#### C

```
#define GFX_FAILURE -1
```

#### Description

This is macro GFX_FAILURE.

### GFX_FALSE Macro

#### File

gfx_common.h

#### C

```
#define GFX_FALSE 0
```

#### Description

This is macro GFX_FALSE.

### GFX_MAX_BUFFER_COUNT Macro

#### File

gfx_common.h

#### C

```
#define GFX_MAX_BUFFER_COUNT 2
```

#### Description

This is macro GFX_MAX_BUFFER_COUNT.

### GFX_NULL Macro

#### File

gfx_common.h

#### C

```
#define GFX_NULL 0
```

#### Description

This is macro GFX_NULL.

### GFX_NUM_FLAGS Macro

#### File

gfx_common.h

#### C

```
#define GFX_NUM_FLAGS GFXF_LAST_FLAG
```

#### Description

This is macro GFX_NUM_FLAGS.

### GFX_SUCCESS Macro

#### File

gfx_common.h

#### C

```
#define GFX_SUCCESS 0
```

#### Description

This is macro GFX_SUCCESS.

### GFX_TRUE Macro

#### File

gfx_common.h

#### C

```
#define GFX_TRUE 1
```

#### Description

This is macro GFX_TRUE.

### GFX_UNSUPPORTED Macro

#### File

gfx_common.h

#### C

```
#define GFX_UNSUPPORTED -2
```

#### Description

This is macro GFX_UNSUPPORTED.

### RGB_2_BITS Macro

#### File

gfx_color.h

#### C

```
#define RGB_2_BITS 2
```

#### Description

This is macro RGB_2_BITS.

### RGB_3_BITS Macro

#### File

gfx_color.h

#### C

```
#define RGB_3_BITS 7
```

#### Description

This is macro RGB_3_BITS.

### RGB_332_BLUE_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_332_BLUE_MASK 0x3
```

#### Description

This is macro RGB_332_BLUE_MASK.

### RGB_332_GREEN_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_332_GREEN_MASK 0x1C
```

#### Description

This is macro RGB_332_GREEN_MASK.

### RGB_332_RED_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_332_RED_MASK 0xE0
```

#### Description

This is macro RGB_332_RED_MASK.

### RGB_5_BITS Macro

#### File

gfx_color.h

#### C

```
#define RGB_5_BITS 31
```

#### Description

This is macro RGB_5_BITS.

### RGB_565_BLUE_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_565_BLUE_MASK 0x1F
```

#### Description

This is macro RGB_565_BLUE_MASK.

### RGB_565_GREEN_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_565_GREEN_MASK 0x7E0
```

#### Description

This is macro RGB_565_GREEN_MASK.

### RGB_565_RED_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_565_RED_MASK 0xF800
```

#### Description

This is macro RGB_565_RED_MASK.

### RGB_6_BITS Macro

#### File

gfx_color.h

#### C

```
#define RGB_6_BITS 63
```

#### Description

This is macro RGB_6_BITS.

### RGB_8_BITS Macro

#### File

gfx_color.h

#### C

```
#define RGB_8_BITS 255
```

#### Description

This is macro RGB_8_BITS.

### RGB_888_BLUE_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_888_BLUE_MASK 0xFF
```

#### Description

This is macro RGB_888_BLUE_MASK.

### RGB_888_GREEN_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_888_GREEN_MASK 0xFF00
```

#### Description

This is macro RGB_888_GREEN_MASK.

### RGB_888_RED_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGB_888_RED_MASK 0xFF0000
```

#### Description

This is macro RGB_888_RED_MASK.

### RGBA_5551_ALPHA_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_5551_ALPHA_MASK 0x1
```

#### Description

This is macro RGBA_5551_ALPHA_MASK.

### RGBA_5551_BLUE_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_5551_BLUE_MASK 0x3E
```

#### Description

This is macro RGBA_5551_BLUE_MASK.

### RGBA_5551_GREEN_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_5551_GREEN_MASK 0x7C0
```

#### Description

This is macro RGBA_5551_GREEN_MASK.

### RGBA_5551_RED_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_5551_RED_MASK 0xF800
```

#### Description

This is macro RGBA_5551_RED_MASK.

### RGBA_8888_ALPHA_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_8888_ALPHA_MASK 0xFF
```

#### Description

This is macro RGBA_8888_ALPHA_MASK.

### RGBA_8888_BLUE_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_8888_BLUE_MASK 0xFF00
```

#### Description

This is macro RGBA_8888_BLUE_MASK.

### RGBA_8888_GREEN_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_8888_GREEN_MASK 0xFF0000
```

#### Description

This is macro RGBA_8888_GREEN_MASK.

### RGBA_8888_RED_MASK Macro

#### File

gfx_color.h

#### C

```
#define RGBA_8888_RED_MASK 0xFF000000
```

#### Description

This is macro RGBA_8888_RED_MASK.

### initialize_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* initialize_FnPtr)(GFX_Context*);
```

#### Description

This is type initialize_FnPtr.

### interrupt_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* interrupt_FnPtr)(uint32_t);
```

#### Description

GFX_DRAW_PIPELINE_ENABLED

### blendColor_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Color (* blendColor_FnPtr)(GFX_Color source, GFX_Color dest, GFX_ColorMode mode);
```

#### Description

This is type blendColor_FnPtr.

### blendGetPoint_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Color (* blendGetPoint_FnPtr)(const GFX_PixelBuffer* buffer, const GFX_Point* pnt, const
GFX_DrawState* state);
```

#### Description

This is type blendGetPoint_FnPtr.

### drawPipelineModeGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_PipelineMode (* drawPipelineModeGet_FnPtr)(void);
```

#### Description

This is type drawPipelineModeGet_FnPtr.

### drawPipelineModeSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawPipelineModeSet_FnPtr)(GFX_PipelineMode);
```

#### Description

This is type drawPipelineModeSet_FnPtr.

### drawResizeModeGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_ResizeMode (* drawResizeModeGet_FnPtr)(void);
```

#### Description

This is type drawResizeModeGet_FnPtr.

### drawResizeModeSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawResizeModeSet_FnPtr)(GFX_ResizeMode);
```

#### Description

This is type drawResizeModeSet_FnPtr.

### drawStretchBlit_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawStretchBlit_FnPtr)(const GFX_PixelBuffer*, const GFX_Rect*, const GFX_Rect*,
const GFX_DrawState*);
```

#### Description

This is type drawStretchBlit_FnPtr.

### drawTargetGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* drawTargetGet_FnPtr)(GFX_PixelBuffer**);
```

#### Description

This is type drawTargetGet_FnPtr.

## drawTargetSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* drawTargetSet_FnPtr)(const GFX_PixelBuffer*);
```

### Description

This is type drawTargetSet_FnPtr.

## GFX_DrawPipeline Structure

### File

gfx_hal.h

### C

```
typedef struct GFX_DrawPipeline_t {
  drawPixel_FnPtr drawPixel[GFX_ANTIALIAS_MODE_COUNT];
  drawLine_FnPtr drawLine[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawLine_FnPtr drawHorzLine[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawLine_FnPtr drawVertLine[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawRect_FnPtr drawRect[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawCircle_FnPtr drawCircle[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawArc_FnPtr drawArc[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawEllipse_FnPtr drawEllipse[GFX_DRAW_MODE_COUNT][GFX_ANTIALIAS_MODE_COUNT];
  drawBlit_FnPtr drawBlit;
  drawStretchBlit_FnPtr drawStretchBlit[GFX_RESIZE_MODE_COUNT];
  drawBlit_FnPtr drawDirectBlit;
  pixelGet_FnPtr pixelGet;
  pixelGetArray_FnPtr pixelArrayGet;
  pixelSet_FnPtr pixelSet;
} GFX_DrawPipeline;
```

### Members

| Members | Description |
|---|---|
| drawPixel_FnPtr drawPixel[GFX_ANTIALIAS_MODE_COUNT]; | draw functions |

### Section

Data Types and Constants

## GFX_PipelineMode Enumeration

Hardware draw path settings.

### File

gfx_common.h

### C

```
typedef enum GFX_PipelineMode_t {
  GFX_PIPELINE_SOFTWARE = 0,
  GFX_PIPELINE_GCU = 1,
  GFX_PIPELINE_GPU = 2,
  GFX_PIPELINE_GCUGPU = 3
} GFX_PipelineMode;
```

### Section

Data Types and Constants

**********

**********

**********

Enumeration:

GFX_HardwareMode_t

### GFX_ResizeMode Enumeration

#### File

gfx_draw.h

#### C

```
typedef enum GFX_ResizeMode_t {
  GFX_RESIZE_NEARESTNEIGHBOR = 0x0,
  GFX_RESIZE_BILINEAR
} GFX_ResizeMode;
```

#### Description

This is type GFX_ResizeMode.

### GFXU_ImageFlags Enumeration

A list of flags describing an image asset

#### File

gfxu_image.h

#### C

```
typedef enum GFXU_ImageFlags_t {
  GFXU_IMAGE_USE_MASK = 0x1,
  GFXU_IMAGE_SUPPORTS_CLIPPING = 0x2,
  GFXU_IMAGE_DIRECT_BLIT = 0x4
} GFXU_ImageFlags;
```

#### Description

Enumeration: GFXU_ImageFlags_t

### layerSwapPending_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef void (* layerSwapPending_FnPtr)(GFX_Layer*);
```

#### Description

This is type layerSwapPending_FnPtr.

### maskColor_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Bool (* maskColor_FnPtr)(const GFX_DrawState* state, GFX_Color color);
```

#### Description

This is type maskColor_FnPtr.

### mirrorPoint_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Point (* mirrorPoint_FnPtr)(const GFX_Point* pnt, const GFX_Rect* rect, GFX_Orientation ori);
```

#### Description

This is type mirrorPoint_FnPtr.

### orientPoint_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Point (* orientPoint_FnPtr)(const GFX_Point* pnt, const GFX_Rect* rect, GFX_Orientation ori);
```

#### Description

This is type orientPoint_FnPtr.

### pixelGetArray_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* pixelGetArray_FnPtr)(GFX_BufferSelection source, const GFX_Rect*, GFX_PixelBuffer*);
```

#### Description

This is type pixelGetArray_FnPtr.

### GFX_PIPELINE_MODE_COUNT Macro

#### File

gfx_common.h

#### C

```
#define GFX_PIPELINE_MODE_COUNT (GFX_PIPELINE_GCUGPU + 1)
```

#### Description

This is macro GFX_PIPELINE_MODE_COUNT.

### GFX_RESIZE_MODE_COUNT Macro

#### File

gfx_draw.h

#### C

```
#define GFX_RESIZE_MODE_COUNT (GFX_RESIZE_BILINEAR+1)
```

#### Description

This is macro GFX_RESIZE_MODE_COUNT.

### GFX_ASSERT Macro

#### File

gfx_common.h

#### C

```
#define GFX_ASSERT(x) { }
```

#### Description

This is macro GFX_ASSERT.

### GFX_GLOBAL_PALETTE_SIZE Macro

#### File

gfx_common.h

#### C

```
#define GFX_GLOBAL_PALETTE_SIZE 256
```

#### Description

This is macro GFX_GLOBAL_PALETTE_SIZE.

### GFX_GlobalPalette Type

#### File

gfx_common.h

#### C

```
typedef void* GFX_GlobalPalette;
```

#### Description

This is type GFX_GlobalPalette.

### globalPaletteGet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_GlobalPalette (* globalPaletteGet_FnPtr)(void);
```

#### Description

This is type globalPaletteGet_FnPtr.

### globalPaletteSet_FnPtr Type

#### File

gfx_hal.h

#### C

```
typedef GFX_Result (* globalPaletteSet_FnPtr)(GFX_GlobalPalette);
```

#### Description

This is type globalPaletteSet_FnPtr.

## layerEffectSet_FnPtr Type

### File

gfx_hal.h

### C

```
typedef GFX_Result (* layerEffectSet_FnPtr)(GFX_Bool, GFX_Bool);
```

### Description

This is type layerEffectSet_FnPtr.

## GFX_DEPRECATED Macro

### File

gfx_common.h

### C

```
#define GFX_DEPRECATED __attribute__ ((deprecated))
```

### Description

This is macro GFX_DEPRECATED.

## Files

### Files

| Name | Description |
|---|---|
| gfx_common.h | Common definitions for MPLAB Harmony Graphics Hardware Abstraction Layer |
| gfx_color.h | Contains functions for color information and manipulation operations |
| gfx_context.h | Defines MPLAB Harmony Graphics Hardware Abstraction Layer Context |
| gfx_default_impl.h | This is file gfx_default_impl.h. |
| gfx_display.h | Defines MPLAB Harmony Graphics Hardware Abstraction Layer display information struct |
| gfx_draw.h | Main header file for MPLAB Harmony Graphics Hardware Abstraction primitive draw functions |
| gfx_draw_blit.h | This is file gfx_draw_blit.h. |
| gfx_draw_circle.h | This is file gfx_draw_circle.h. |
| gfx_draw_line.h | This is file gfx_draw_line.h. |
| gfx_draw_pixel.h | This is file gfx_draw_pixel.h. |
| gfx_draw_rect.h | This is file gfx_draw_rect.h. |
| gfx_driver_interface.h | Defines MPLAB Harmony Graphics Hardware Abstraction Layer driver interface struct |
| gfx_hal.h | This is file gfx_hal.h. |
| gfx_interface.h | This is file gfx_interface.h. |
| gfx_layer.h | Defines the graphics layer construct |
| gfx_math.h | Contains some general purpose math functions |
| gfx_pixel_buffer.h | Defines a general purpose pixel buffer construct. |
| gfx_rect.h | Defines general purposes rectangle functions. |
| gfx_util.h | Utility functions for the Hardware Abstraction Layer |

### Description

## *gfx_common.h*

Common definitions for MPLAB Harmony Graphics Hardware Abstraction Layer

### Enumerations

| | Name | Description |
|---|---|---|
| | GFX_BlendMode_t | Blend mode masks |

| | | |
|---|---|---|
| | GFX_BufferSelection_t | Buffer selector used when querying layers for certain buffer states. |
| | GFX_BufferState_t | Frame buffer memory states |
| | GFX_Flag_t | Hardware abstraction state interface flags. See gfx.h for a comprehensive description. |
| | GFX_Orientation_t | Orthogonal orientation settings. |
| | GFX_PipelineMode_t | Hardware draw path settings. |
| | GFX_BlendMode | Blend mode masks |
| | GFX_BufferSelection | Buffer selector used when querying layers for certain buffer states. |
| | GFX_BufferState | Frame buffer memory states |
| | GFX_Flag | Hardware abstraction state interface flags. See gfx.h for a comprehensive description. |
| | GFX_Orientation | Orthogonal orientation settings. |
| | GFX_PipelineMode | Hardware draw path settings. |

## Macros

| | Name | Description |
|---|---|---|
| | GFX_ASSERT | This is macro GFX_ASSERT. |
| | GFX_DEPRECATED | This is macro GFX_DEPRECATED. |
| | GFX_FAILURE | This is macro GFX_FAILURE. |
| | GFX_FALSE | This is macro GFX_FALSE. |
| | GFX_GLOBAL_PALETTE_SIZE | This is macro GFX_GLOBAL_PALETTE_SIZE. |
| | GFX_MAX_BUFFER_COUNT | This is macro GFX_MAX_BUFFER_COUNT. |
| | GFX_NULL | This is macro GFX_NULL. |
| | GFX_NUM_FLAGS | This is macro GFX_NUM_FLAGS. |
| | GFX_PIPELINE_MODE_COUNT | This is macro GFX_PIPELINE_MODE_COUNT. |
| | GFX_SUCCESS | This is macro GFX_SUCCESS. |
| | GFX_TRUE | This is macro GFX_TRUE. |
| | GFX_UNSUPPORTED | This is macro GFX_UNSUPPORTED. |

## Structures

| | Name | Description |
|---|---|---|
| | GFX_MemoryIntf_t | Custom memory manager interface. |
| | GFX_Point_t | A two dimensional Cartesian point. |
| | GFX_Rect_t | A rectangle definition. |
| | GFX_Size_t | A two dimensional indication of size. Values are signed but should never be negative. |
| | GFX_MemoryIntf | Custom memory manager interface. |
| | GFX_Size | A two dimensional indication of size. Values are signed but should never be negative. |

## Types

| | Name | Description |
|---|---|---|
| | GFX_Bool | This is type GFX_Bool. |
| | GFX_Buffer | This is type GFX_Buffer. |
| | GFX_Calloc_FnPtr | Simple wrapper around the standard calloc function pointer. Used for redirecting memory allocation to other memory management systems. |
| | GFX_Color | This is type GFX_Color. |
| | GFX_ColorMode | List of available color modes. |
| | GFX_Display | This is type GFX_Display. |
| | GFX_Driver | This is type GFX_Driver. |
| | GFX_Free_FnPtr | Simple wrapper around the standard free function pointer. Used for redirecting memory free to other memory management systems. |
| | GFX_GlobalPalette | This is type GFX_GlobalPalette. |
| | GFX_Handle | This is type GFX_Handle. |
| | GFX_Malloc_FnPtr | Simple wrapper around the standard malloc function pointer. Used for redirecting memory allocation to other memory management systems. |
| | GFX_Memcpy_FnPtr | Simple wrapper around the standard memcpy function pointer. Used for redirecting memcpy to other memory management systems. |
| | GFX_Memset_FnPtr | Simple wrapper around the standard memset function pointer. Used for redirecting memset to other memory management systems. |
| | GFX_Processor | This is type GFX_Processor. |

| | | |
|---|---|---|
| | GFX_Realloc_FnPtr | Simple wrapper around the standard realloc function pointer. Used for redirecting memory allocation to other memory management systems. |
| | GFX_Result | This is type GFX_Result. |
| | GFX_SyncCallback_FnPtr | This is type GFX_SyncCallback_FnPtr. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Type definitions for common functions.

## File Name

gfx_common.h

## Company

Microchip Technology Inc.

### *gfx_color.h*

Contains functions for color information and manipulation operations

## Enumerations

| | Name | Description |
|---|---|---|
| | GFX_BitsPerPixel_t | List of available bits-per-pixel sizes. |
| | GFX_ColorMask_t | Maskable list of color valies. |
| | GFX_ColorMode_t | List of available color modes. |
| | GFX_ColorName_t | Color name reference table |
| | GFX_BitsPerPixel | List of available bits-per-pixel sizes. |
| | GFX_ColorMask | Maskable list of color valies. |
| | GFX_ColorName | Color name reference table |

## Functions

| | Name | Description |
|---|---|---|
| | GFX_ColorBilerp | Calculates bilinear interpolation between four colors |
| | GFX_ColorBlend_RGBA_8888 | Blends two RGBA8888 colors together using their alpha channel values. |
| | GFX_ColorChannelAlpha | Used for getting the alpha color channel of a given color value. |
| | GFX_ColorChannelBlue | Used for getting the blue color channel of a given color value. |
| | GFX_ColorChannelGreen | Used for getting the green color channel of a given color value. |
| | GFX_ColorChannelRed | Used for getting the red color channel of a given color value. |
| | GFX_ColorConvert | Converts a color value from one mode to another |
| | GFX_ColorLerp | Linear interpolation between two colors |
| | GFX_ColorModeInfoGet | |
| | GFX_ColorValue | Used for getting a color value by name. |

## Macros

| | Name | Description |
|---|---|---|
| | AGBA_8888_ALPHA_MASK | This is macro AGBA_8888_ALPHA_MASK. |
| | AGBA_8888_BLUE_MASK | This is macro AGBA_8888_BLUE_MASK. |
| | AGBA_8888_GREEN_MASK | This is macro AGBA_8888_GREEN_MASK. |
| | AGBA_8888_RED_MASK | This is macro AGBA_8888_RED_MASK. |
| | GFX_COLOR_MAX_SIZE | This is macro GFX_COLOR_MAX_SIZE. |
| | GFX_COLOR_MODE_COUNT | This is macro GFX_COLOR_MODE_COUNT. |
| | GFX_COLOR_MODE_IS_ALPHA | This is macro GFX_COLOR_MODE_IS_ALPHA. |
| | GFX_COLOR_MODE_IS_INDEX | This is macro GFX_COLOR_MODE_IS_INDEX. |
| | GFX_COLOR_MODE_IS_PIXEL | This is macro GFX_COLOR_MODE_IS_PIXEL. |
| | GFX_COLOR_MODE_LAST_COLOR | This is macro GFX_COLOR_MODE_LAST_COLOR. |
| | RGB_2_BITS | This is macro RGB_2_BITS. |
| | RGB_3_BITS | This is macro RGB_3_BITS. |
| | RGB_332_BLUE_MASK | This is macro RGB_332_BLUE_MASK. |

| | | |
|---|---|---|
| | RGB_332_GREEN_MASK | This is macro RGB_332_GREEN_MASK. |
| | RGB_332_RED_MASK | This is macro RGB_332_RED_MASK. |
| | RGB_5_BITS | This is macro RGB_5_BITS. |
| | RGB_565_BLUE_MASK | This is macro RGB_565_BLUE_MASK. |
| | RGB_565_GREEN_MASK | This is macro RGB_565_GREEN_MASK. |
| | RGB_565_RED_MASK | This is macro RGB_565_RED_MASK. |
| | RGB_6_BITS | This is macro RGB_6_BITS. |
| | RGB_8_BITS | This is macro RGB_8_BITS. |
| | RGB_888_BLUE_MASK | This is macro RGB_888_BLUE_MASK. |
| | RGB_888_GREEN_MASK | This is macro RGB_888_GREEN_MASK. |
| | RGB_888_RED_MASK | This is macro RGB_888_RED_MASK. |
| | RGBA_5551_ALPHA_MASK | This is macro RGBA_5551_ALPHA_MASK. |
| | RGBA_5551_BLUE_MASK | This is macro RGBA_5551_BLUE_MASK. |
| | RGBA_5551_GREEN_MASK | This is macro RGBA_5551_GREEN_MASK. |
| | RGBA_5551_RED_MASK | This is macro RGBA_5551_RED_MASK. |
| | RGBA_8888_ALPHA_MASK | This is macro RGBA_8888_ALPHA_MASK. |
| | RGBA_8888_BLUE_MASK | This is macro RGBA_8888_BLUE_MASK. |
| | RGBA_8888_GREEN_MASK | This is macro RGBA_8888_GREEN_MASK. |
| | RGBA_8888_RED_MASK | This is macro RGBA_8888_RED_MASK. |

## Structures

| | Name | Description |
|---|---|---|
| | GFX_ColorModeInfo_t | Struct that provides information about a color mode. |
| | GFX_ColorModeInfo | Struct that provides information about a color mode. |

## Variables

| | Name | Description |
|---|---|---|
| | GFX_ColorInfo | This is variable GFX_ColorInfo. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Color conversion and color channel management

## File Name

gfx_color.h

## Company

Microchip Technology Inc.

### *gfx_context.h*

Defines MPLAB Harmony Graphics Hardware Abstraction Layer Context

## Functions

| | Name | Description |
|---|---|---|
| | GFX_ActiveContext | Gets the current set active HAL context. |
| | GFX_ContextActiveSet | Sets the active context |

## Structures

| | Name | Description |
|---|---|---|
| | GFX_Context_t | An instance of the hardware abstraction layer. |
| | GFX_Context | An instance of the hardware abstraction layer. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

HAL context management functions.

## File Name

gfx_context.h

## Company

Microchip Technology Inc.

### *gfx_default_impl.h*

This is file gfx_default_impl.h.

### *gfx_display.h*

Defines MPLAB Harmony Graphics Hardware Abstraction Layer display information struct

## Structures

| | Name | Description |
|---|---|---|
| | GFX_DisplayInfo_t | Describes a graphical display device. |
| | GFX_DisplayInfo | Describes a graphical display device. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer
Display information.

## File Name

gfx_display.h

## Company

Microchip Technology Inc.

### *gfx_draw.h*

Main header file for MPLAB Harmony Graphics Hardware Abstraction primitive draw functions

## Enumerations

| | Name | Description |
|---|---|---|
| | GFX_AntialiasMode_t | Enables anti-aliased drawing hint |
| | GFX_DrawMode_t | Gradient draw modes. |
| | GFX_ResizeMode_t | This is type GFX_ResizeMode. |
| | GFX_AntialiasMode | Enables anti-aliased drawing hint |
| | GFX_DrawMode | Gradient draw modes. |
| | GFX_ResizeMode | This is type GFX_ResizeMode. |

## Functions

| | Name | Description |
|---|---|---|
| | GFX_DrawBlit | Blits a buffer of pixels into the frame buffer. |
| | GFX_DrawCircle | Draws a circle from using the specified dimensions and the current draw state. |
| | GFX_DrawDirectBlit | Blits a buffer of pixels into the frame buffer without performing per-pixel operations on the data. |
| | GFX_DrawLine | Draws a line from (x1,y1) to (x2,y2) using the current draw state. |
| | GFX_DrawPixel | Sets the pixel at X and Y using the current draw state. |
| | GFX_DrawRect | Draws a rectangle using the specified dimensions and the current draw state. |
| | GFX_DrawStretchBlit | Blits a buffer of pixels into the frame buffer. |

## Macros

| | Name | Description |
|---|---|---|
| | GFX_ANTIALIAS_MODE_COUNT | This is macro GFX_ANTIALIAS_MODE_COUNT. |
| | GFX_DRAW_MODE_COUNT | This is macro GFX_DRAW_MODE_COUNT. |
| | GFX_RESIZE_MODE_COUNT | This is macro GFX_RESIZE_MODE_COUNT. |

## Structures

| | Name | Description |
|---|---|---|
| | GFX_DrawState_t | A list of drawing hints for shape drawing algorithms |
| | GFX_DrawState | A list of drawing hints for shape drawing algorithms |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Shape drawing functions.

## File Name

gfx_draw.h

## Company

Microchip Technology Inc.

### *gfx_draw_blit.h*

This is file gfx_draw_blit.h.

### *gfx_draw_circle.h*

This is file gfx_draw_circle.h.

### *gfx_draw_line.h*

This is file gfx_draw_line.h.

### *gfx_draw_pixel.h*

This is file gfx_draw_pixel.h.

### *gfx_draw_rect.h*

This is file gfx_draw_rect.h.

### *gfx_driver_interface.h*

Defines MPLAB Harmony Graphics Hardware Abstraction Layer driver interface struct

## Structures

| | Name | Description |
|---|---|---|
| | GFX_DriverInfo_t | A driver description structure. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Display driver information, internal use.

## File Name

gfx_draw_interface.h

## Company

Microchip Technology Inc.

## *gfx_hal.h*

### Structures

| | Name | Description |
|---|---|---|
| | GFX_DrawPipeline_t | |
| | GFX_HAL_t | Hardware Abstraction Function Table *<br>• This is the core hardware abtraction table that makes everything work. Drivers are<br>• expected to reroute the functionality of this table to hardware specific implementations<br>• to provide accelerated performance and features. |
| | GFX_DrawPipeline | |
| | GFX_HAL | Hardware Abstraction Function Table *<br>• This is the core hardware abtraction table that makes everything work. Drivers are<br>• expected to reroute the functionality of this table to hardware specific implementations<br>• to provide accelerated performance and features. |

### Types

| | Name | Description |
|---|---|---|
| | begin_FnPtr | This is type begin_FnPtr. |
| | blendColor_FnPtr | This is type blendColor_FnPtr. |
| | blendGetPoint_FnPtr | This is type blendGetPoint_FnPtr. |
| | boolGet_FnPtr | This is type boolGet_FnPtr. |
| | boolSet_FnPtr | This is type boolSet_FnPtr. |
| | brightnessGet_FnPtr | This is type brightnessGet_FnPtr. |
| | brightnessRangeGet_FnPtr | This is type brightnessRangeGet_FnPtr. |
| | brightnessSet_FnPtr | This is type brightnessSet_FnPtr. |
| | colorModeGet_FnPtr | This is type colorModeGet_FnPtr. |
| | colorModeSet_FnPtr | This is type colorModeSet_FnPtr. |
| | destroy_FnPtr | This is type destroy_FnPtr. |
| | drawAlphaValueGet_FnPtr | This is type drawAlphaValueGet_FnPtr. |
| | drawAlphaValueSet_FnPtr | This is type drawAlphaValueSet_FnPtr. |
| | drawBlendModeGet_FnPtr | This is type drawBlendModeGet_FnPtr. |
| | drawBlendModeSet_FnPtr | This is type drawBlendModeSet_FnPtr. |
| | drawBlit_FnPtr | This is type drawBlit_FnPtr. |
| | drawCircle_FnPtr | This is type drawCircle_FnPtr. |
| | drawClipRectGet_FnPtr | This is type drawClipRectGet_FnPtr. |
| | drawClipRectSet_FnPtr | This is type drawClipRectSet_FnPtr. |
| | drawColorGet_FnPtr | This is type drawColorGet_FnPtr. |
| | drawColorSet_FnPtr | This is type drawColorSet_FnPtr. |
| | drawGradientColorGet_FnPtr | This is type drawGradientColorGet_FnPtr. |
| | drawGradientColorSet_FnPtr | This is type drawGradientColorSet_FnPtr. |
| | drawLine_FnPtr | This is type drawLine_FnPtr. |
| | drawLock_FnPtr | This is type drawLock_FnPtr. |
| | drawMaskValueGet_FnPtr | This is type drawMaskValueGet_FnPtr. |
| | drawMaskValueSet_FnPtr | This is type drawMaskValueSet_FnPtr. |
| | drawModeGet_FnPtr | This is type drawModeGet_FnPtr. |
| | drawModeSet_FnPtr | This is type drawModeSet_FnPtr. |
| | drawPaletteGet_FnPtr | This is type drawPaletteGet_FnPtr. |
| | drawPaletteSet_FnPtr | This is type drawPaletteSet_FnPtr. |
| | drawPipelineModeGet_FnPtr | This is type drawPipelineModeGet_FnPtr. |
| | drawPipelineModeSet_FnPtr | This is type drawPipelineModeSet_FnPtr. |
| | drawPixel_FnPtr | This is type drawPixel_FnPtr. |
| | drawRect_FnPtr | This is type drawRect_FnPtr. |
| | drawResizeModeGet_FnPtr | This is type drawResizeModeGet_FnPtr. |
| | drawResizeModeSet_FnPtr | This is type drawResizeModeSet_FnPtr. |

| | | |
|---|---|---|
| | drawStretchBlit_FnPtr | This is type drawStretchBlit_FnPtr. |
| | drawTargetGet_FnPtr | This is type drawTargetGet_FnPtr. |
| | drawTargetSet_FnPtr | This is type drawTargetSet_FnPtr. |
| | drawThicknessGet_FnPtr | This is type drawThicknessGet_FnPtr. |
| | drawThicknessSet_FnPtr | This is type drawThicknessSet_FnPtr. |
| | drawUnlock_FnPtr | This is type drawUnlock_FnPtr. |
| | end_FnPtr | This is type end_FnPtr. |
| | GFX_DriverInfo | A driver description structure. |
| | globalPaletteGet_FnPtr | This is type globalPaletteGet_FnPtr. |
| | globalPaletteSet_FnPtr | This is type globalPaletteSet_FnPtr. |
| | initialize_FnPtr | This is type initialize_FnPtr. |
| | interrupt_FnPtr | GFX_DRAW_PIPELINE_ENABLED |
| | layerActiveGet_FnPtr | This is type layerActiveGet_FnPtr. |
| | layerActiveSet_FnPtr | This is type layerActiveSet_FnPtr. |
| | layerAlphaAmountGet_FnPtr | This is type layerAlphaAmountGet_FnPtr. |
| | layerAlphaAmountSet_FnPtr | This is type layerAlphaAmountSet_FnPtr. |
| | layerBufferAddressGet_FnPtr | This is type layerBufferAddressGet_FnPtr. |
| | layerBufferAddressSet_FnPtr | This is type layerBufferAddressSet_FnPtr. |
| | layerBufferAllocate_FnPtr | This is type layerBufferAllocate_FnPtr. |
| | layerBufferCoherentGet_FnPtr | This is type layerBufferCoherentGet_FnPtr. |
| | layerBufferCoherentSet_FnPtr | This is type layerBufferCoherentSet_FnPtr. |
| | layerBufferCountGet_FnPtr | This is type layerBufferCountGet_FnPtr. |
| | layerBufferCountSet_FnPtr | This is type layerBufferCountSet_FnPtr. |
| | layerBufferFree_FnPtr | This is type layerBufferFree_FnPtr. |
| | layerBufferIsAllocated_FnPtr | This is type layerBufferIsAllocated_FnPtr. |
| | layerEffectSet_FnPtr | This is type layerEffectSet_FnPtr. |
| | layerMaskColorGet_FnPtr | This is type layerMaskColorGet_FnPtr. |
| | layerMaskColorSet_FnPtr | This is type layerMaskColorSet_FnPtr. |
| | layerPositionGet_FnPtr | This is type layerPositionGet_FnPtr. |
| | layerPositionSet_FnPtr | This is type layerPositionSet_FnPtr. |
| | layerSizeGet_FnPtr | This is type layerSizeGet_FnPtr. |
| | layerSizeSet_FnPtr | This is type layerSizeSet_FnPtr. |
| | layerSwapped_FnPtr | This is type layerSwapped_FnPtr. |
| | layerSwapPending_FnPtr | This is type layerSwapPending_FnPtr. |
| | maskColor_FnPtr | This is type maskColor_FnPtr. |
| | mirrorPoint_FnPtr | This is type mirrorPoint_FnPtr. |
| | orientationGet_FnPtr | This is type orientationGet_FnPtr. |
| | orientationSet_FnPtr | This is type orientationSet_FnPtr. |
| | orientPoint_FnPtr | This is type orientPoint_FnPtr. |
| | pixelGet_FnPtr | This is type pixelGet_FnPtr. |
| | pixelGetArray_FnPtr | This is type pixelGetArray_FnPtr. |
| | pixelSet_FnPtr | This is type pixelSet_FnPtr. |
| | syncCallbackGet_FnPtr | This is type syncCallbackGet_FnPtr. |
| | syncCallbackSet_FnPtr | This is type syncCallbackSet_FnPtr. |
| | syncCallbackSt_FnPtr | This is type syncCallbackSt_FnPtr. |
| | update_FnPtr | This is type update_FnPtr. |

## Description

This is file gfx_hal.h.

### *gfx_interface.h*

This is file gfx_interface.h.

## *gfx_layer.h*

Defines the graphics layer construct

### Functions

| | Name | Description |
|---|---|---|
| | GFX_LayerFromOrientedSpace | Transforms a layer oriented space to screen space. |
| | GFX_LayerPointFromOrientedSpace | Transforms a point from oriented space to screen space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerPointToOrientedSpace | Transforms a point from screen space to oriented space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerReadBuffer | Gets the pointer to the layer's current read pixel buffer. |
| | GFX_LayerRectFromOrientedSpace | Transforms a layer point from oriented space to screen space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerRectToOrientedSpace | Transforms a rectangle from screen space to oriented space given a layer, a display orientation, and a mirroring setting. |
| | GFX_LayerRotate | Swaps the width and height dimensions of a layer. Can be used for run-time display orientation |
| | GFX_LayerSwap | Performs a swap operation on the given layer. This advances the pointers of layer's buffer chain. The current write buffer becomes the new read buffer and a new buffer is chosen as the new write buffer. Has no effect in single buffer environments. |
| | GFX_LayerToOrientedSpace | Transforms a layer from screen space to oriented space. |
| | GFX_LayerWriteBuffer | Gets the pointer to the layer's current write pixel buffer. |

### Structures

| | Name | Description |
|---|---|---|
| | GFX_FrameBuffer_t | A frame buffer is a wrapper around a pixel buffer construct that is used by display drivers to manage frame buffers. |
| | GFX_Layer_t | Layers describe basic display drawing areas and are meant to map directly to graphics controller hardware layers. |
| | GFX_FrameBuffer | A frame buffer is a wrapper around a pixel buffer construct that is used by display drivers to manage frame buffers. |
| | GFX_Layer | Layers describe basic display drawing areas and are meant to map directly to graphics controller hardware layers. |

### Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Layer and buffer management.

### File Name

gfx_layer.h

### Company

Microchip Technology Inc.

## *gfx_math.h*

Contains some general purpose math functions

### Functions

| | Name | Description |
|---|---|---|
| | GFX_AbsoluteValue | Calculates the absolute value of a signed integer. |
| | GFX_Clampf | Clamps a float between a min and max |
| | GFX_Clampi | Clamps an integer between a min and max |
| | GFX_DivideRounding | This is function GFX_DivideRounding. |
| | GFX_Lerp | Performs a linear interpolation of an integer based on a percentage between two signed points. |
| | GFX_Maxf | Returns the larger of two floats. |
| | GFX_Maxi | Returns the larger of two integers. |

| | | |
|---|---|---|
| | GFX_Minf | Returns the smaller of two floats. |
| | GFX_Mini | Returns the smaller of two integers. |
| | GFX_Percent | Calculates the percentage of one number when applied to another. Integer based. Accuracy for higher numbers is not guaranteed.<br>The result is the decimal percentage multiplied by 100. |
| | GFX_PercentOf | Calculates the percentage of a number. Returns a whole number with no decimal component. |
| | GFX_PercentOfDec | This is function GFX_PercentOfDec. |
| | GFX_PercentWholeRounded | Calculates the percentage of one number when applied to another. Integer based. Accuracy for higher numbers is not guaranteed.<br>The difference between this and GFX_Percent is that the decimal portion of the whole number is rounded off. |
| | GFX_ScaleInteger | Scales an integer from one number range of 0 -> n0 to another range 0 -> n1 based on percentages. |
| | GFX_ScaleIntegerSigned | Scales a signed integer from one number range of 0 -> n0 to another range 0 -> n1 based on percentages. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Math support functions.

## File Name

gfx_layer.h

## Company

Microchip Technology Inc.

### *gfx_pixel_buffer.h*

Defines a general purpose pixel buffer construct.

## Functions

| | Name | Description |
|---|---|---|
| | GFX_PixelBufferAreaFill | Fills an area of a pixel buffer with a solid color. Caller is responsible for ensuring that the color is the same color format as the destination buffer. |
| | GFX_PixelBufferAreaFill_Unsafe | Fills an area of a pixel buffer with a solid color. Caller is responsible for ensuring that the color is the same color format as the destination buffer. Like GFX_PixelBufferAreaFill but performs no bounds checking. |
| | GFX_PixelBufferAreaGet | Extracts a rectanglar section of pixels from a pixel buffer. |
| | GFX_PixelBufferAreaGet_Unsafe | Extracts a rectanglar section of pixels from a pixel buffer. Like GFX_PixelBufferAreaGet but performs no clipping between the rectangles of the extract area and the source buffer. |
| | GFX_PixelBufferAreaSet | Copies an area of pixels from a source buffer to a destination buffer. If the source buffer format does not match the destination format the data will be converted to match during the copy operation. |
| | GFX_PixelBufferAreaSet_Unsafe | Copies an area of pixels from a source buffer to a destination buffer. If the source buffer format does not match the destination format the data will be converted to match during the copy operation. Like GFX_PixelBufferAreaSet but performs no bounds checking. |
| | GFX_PixelBufferClipRect | Clips a rectangle against a pixel buffer. The result is guaranteed to fit inside the buffer's area. |
| | GFX_PixelBufferConvert | Duplicates a pixel buffer and converts the copy to another color mode. |
| | GFX_PixelBufferCopy | Creates a copy of the input buffer. If the input buffer's pixel data is not null then the data will be duplicated for the result buffer. |
| | GFX_PixelBufferCreate | Initializes a pixel buffer struct. Does not actually allocate any memory. |
| | GFX_PixelBufferDestroy | Destroys a pixel buffer construct. If the buffer's pixels pointer is not zero this function will attempt to free it using the provided GFX_MemoryIntf memory interface. |
| | GFX_PixelBufferGet | Gets the value of the pixel that resides at the provided point in the given buffer. |
| | GFX_PixelBufferGet_Unsafe | Gets the value of the pixel that resides at the provided point in the given buffer. Like GFX_PixelBufferGet but performs no bounds checking. |
| | GFX_PixelBufferGetIndex | Interprets the pixel buffer as a table of indices and looks up a specific index at position 'idx'. Indices may be 1bpp, 4bpp, or 8bpp in size and are indicated by the color mode of the pixel buffer. |
| | GFX_PixelBufferOffsetGet | Gets the offset address of the pixel that resides at the provided point in the given buffer. |
| | GFX_PixelBufferOffsetGet_Unsafe | Gets the offset address of the pixel that resides at the provided point in the given buffer. Similar to GFX_PixelBufferOffsetGet but performs no bounds checking. |

| | | |
|---|---|---|
| | GFX_PixelBufferSet | Sets a pixel in a pixel buffer at a point to a specified color. Caller is responsible for ensuring that the input color is in the same color format as the pixel buffer. |
| | GFX_PixelBufferSet_Unsafe | Sets a pixel in a pixel buffer at a point to a specified color. Caller is responsible for ensuring that the input color is in the same color format as the pixel buffer. Like GFX_PixelBufferSet but performs no bounds checking. |

## Structures

| | Name | Description |
|---|---|---|
| | GFX_PixelBuffer_t | A pixel buffer is a wrapper around a basic data pointer. A pixel buffer has a color mode, a pixel count, a rectangular dimension, a pixel count, and a lenght in bytes. |
| | GFX_PixelBuffer | A pixel buffer is a wrapper around a basic data pointer. A pixel buffer has a color mode, a pixel count, a rectangular dimension, a pixel count, and a lenght in bytes. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Pixel buffer generation and management functions.

## File Name

gfx_pixel_buffer.h

## Company

Microchip Technology Inc.

### *gfx_rect.h*

Defines general purposes rectangle functions.

## Functions

| | Name | Description |
|---|---|---|
| | GFX_RectClip | Clips a rectangle to the space of another rectangle. The result rectangle is a rectangle that will fit inside both of the given rectangles. |
| | GFX_RectClipAdj | This is function GFX_RectClipAdj. |
| | GFX_RectCombine | Combines the area of two rectangles into a single rectangle. |
| | GFX_RectCompare | This is function GFX_RectCompare. |
| | GFX_RectContainsPoint | Determines if a point is inside a rectangle. |
| | GFX_RectContainsRect | Determines if a rectangle is completely inside another rectangle. Still returns true if the edges are touching. |
| | GFX_RectFromPoints | This is function GFX_RectFromPoints. |
| | GFX_RectIntersects | Determines if two rectangles are intersecting |
| | GFX_RectsAreSimilar | This is function GFX_RectsAreSimilar. |
| | GFX_RectSplit | This is function GFX_RectSplit. |
| | GFX_RectToPoints | This is function GFX_RectToPoints. |

## Variables

| | Name | Description |
|---|---|---|
| | GFX_Rect_Zero | This is variable GFX_Rect_Zero. |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Rectangle management functions.

## File Name

gfx_rect.h

## Company

Microchip Technology Inc.

### *gfx_util.h*

Utility functions for the Hardware Abstraction Layer

## Functions

| | Name | Description |
|---|---|---|
| | GFX_UtilMirrorPoint | Reorients a point to a given mirrored orientation. |
| | GFX_UtilOrientPoint | Reorients a point to a given orthogonal orientation. |
| | GFX_UtilPointFromOrientedSpace | Transforms a point from an oriented rectangle space to an outer space given a display orientation and a mirroring setting. |
| | GFX_UtilPointToOrientedSpace | Transforms a point to an oriented rectangle space to an outer space given a display orientation and a mirroring setting. |
| | GFX_UtilSizeFromOrientedSpace | Transforms a size tuple from oriented space to screen space |
| | GFX_UtilSizeToOrientedSpace | Transforms a size tuple from screen space to oriented space |
| | GFX_UtilSortPointsX | Sorts two points in the X axis |
| | GFX_UtilSortPointsY | Sorts two points in the Y axis |

## Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Layer and point utility functions.

## File Name

gfx_util.h

## Company

Microchip Technology Inc.

# Aria HAL Driver Examples

# *ILI9488 Display Controller Driver Library*

Provides information on the ILI9488 Display Controller Driver Library

## Description

This driver library provides an API interface to configure and use an external LCD module with an ILI9488 controller on a Microchip MCU. The driver is designed to work with the MPLAB Harmony Graphics Library.

## Introduction

Introduces the driver library.

## Description

The ILI9488 Display Controller Driver is a example implementation of a MPLAB Harmony Aria HAL driver, which supports DBI Type C 3-Line Serial Interface for MCUs with SPI peripheral, or DBI Type B 16-/8-bit parallel interface for MCUs with a Static Memory Controller (SMC) peripheral.

On these interfaces, the driver library provides APIs to:

- Send write and read commands to configure the ILI9488 Display Controller
- Write and read pixel(s) in the ILI9488's Graphics RAM (GRAM)

## Using the Library

This topic describes the basic architecture of the ILI9488 Display Controller Driver Library and provides information and examples on how to use it.

## Description

**Interface Header File:** `drv_gfx_ili9488.h`

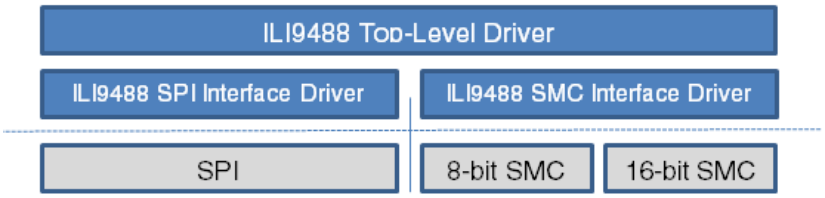
The top-level interface to the ILI9488 Display Controller Driver Library is defined in the `drv_gfx_ili9488.h` header file. Any C language source (`.c`), particularly in the graphics framework, file that uses the ILI9488 Display Controller Driver Library should include `drv_gfx_ili9488.h`.

### Abstraction Model

This section describes how the ILI9488 Display Controller Driver Library provides the APIs that abstracts the interface-specific implementation from the application layer.

## Description

The following figure shows a high-level block diagram that describes the structure of the display driver.




The top-level driver does the following:

- Glues the interface driver to the MPLAB Harmony Graphics Library's hardware abstraction layer
- Contains the ILI9488 setup routines and initialization commands, and calls to the interface driver layer
- Manages the buffer that contains pixel data that is sent to the ILI9488 Controller for display

The interface drivers provide the basic APIs for sending read/write commands, and reading/writing pixel data. The APIs contain corresponding calls to the interface's peripheral drivers (SPI or SMC) for communicating with the ILI9488 Controller.

### LIbrary Overview

Refer to the Driver Library Overview section for information on how the driver operates in a system.

The library interface routines are divided into various sub-sections, which address one of the blocks or the overall operation of the ILI9488 Display Controller Driver.

| Library Interface Section | Description |
|---|---|
| Top-level Driver Functions | Functions that glues to the MPLAB Harmony graphics library and contains APIs for the graphics library to draw pixels to the ILI9488 GRAM. |
| Interface Driver Functions | Interface-specific APIs for sending write/read commands and pixel data to the ILI9488. These functions perform the necessary peripheral calls based on the selected interface (SPI or SMC). |

### How the Library Works

This section describes how the ILI9488 Display Controller Driver Library works .

## Description

The library provides interfaces to support

- Initialization and setup of the ILI9488 Display Controller
- Sending read/write commands to the ILI9488 Display Controller using either the DBI-B parallel or DBI-C 4-Line SPI interface
- Reading/writing pixel data in the ILI9488 graphics RAM

## Initialization

Before the driver can be used, it needs to be initialized. In ILI9488_ContextInitialize, the top-level driver registers itself to the MPLAB Harmony Graphics HAL, so that its initialization routine is automatically called when the graphics library initializes.

The initialization process involves allocating the pixel buffer and opening the interface to the ILI9488 Display Controller module. The top-level driver calls ILI9488_Intf_Open to open the peripheral interface, issues a hardware reset and sends the setup commands to the ILI9488. initCmdParm contains a list of the basic commands needed to set up the ILI9488 and these commands are sent through the interface driver using the ILI9488_Intf_WriteCmd API.

## Displaying Frames

The top-level driver can be configured to write pixel data to the ILI9488 GRAM per pixel or per line. Per frame writes is also supported, but only in 16-bit DBI-B parallel mode. Per frame writes provides the fastest update rate, but requires more memory to allocate a full frame buffer. Writing per line has less memory space requirements, but requires an overhead to read the current pixel line data before it is updated and written back. This is done because the graphics library may update individual pixels only and not the whole line, thus the rest of the pixel buffer data must contain valid data before it is written back to the ILI9488 GRAM. Otherwise, the pixel buffer will contain pixel data from other lines and cause display artifacts.

To read and write pixel data, the top-level driver calls ILI9488_Intf_ReadPixels and ILI9488_Intf_WritePixels, respectively.

## Library Interface

### a) Functions

| | Name | Description |
|---|---|---|
| | ILI9488_Intf_Close | Closes the HW interface to the ILI9488 device |
| | ILI9488_Intf_Open | Opens the specified port to the ILI9488 device |
| | ILI9488_Intf_ReadCmd | Sends read command and reads response from ILI9488 |
| | ILI9488_Intf_ReadPixels | Read pixel data from specified position in ILI9488 GRAM |
| | ILI9488_Intf_WriteCmd | Sends write command and parameters to the ILI9488 device |
| | ILI9488_Intf_WritePixels | Writes pixel data to ILI9488 GRAM from specified position |

### b) Data Types and Constants

| | Name | Description |
|---|---|---|
| | ILI9488_CMD_PARAM | Structure contains command and parameter information |
| | ILI9488_DRV_STATE | Enum of ILI9488 driver states |
| | ILI9488_DRV | Structure contains driver-specific data and ops pointers |
| | ILI9488_Backlight_Off | This is macro ILI9488_Backlight_Off. |
| | ILI9488_Backlight_On | This is macro ILI9488_Backlight_On. |
| | ILI9488_Reset_Assert | This is macro ILI9488_Reset_Assert. |
| | ILI9488_Reset_Deassert | This is macro ILI9488_Reset_Deassert. |
| | ILI9488_SPI_DCX_Command | This is macro ILI9488_SPI_DCX_Command. |
| | ILI9488_SPI_DCX_Data | This is macro ILI9488_SPI_DCX_Data. |
| | ILI9488_SPI_SS_Assert | This is macro ILI9488_SPI_SS_Assert. |
| | ILI9488_SPI_SS_Deassert | This is macro ILI9488_SPI_SS_Deassert. |

### Description

The interface described in the following sections are provided as an example of the Aria User Library Interface Hardware Abstraction Layer. This example can be used as a starting point for developing your own Aria HAL driver.

#### a) Functions

#### ILI9488_Intf_Close Function

Closes the HW interface to the ILI9488 device

### File

help_drv_gfx_ili9488_common.h

### C

```
void ILI9488_Intf_Close(ILI9488_DRV * drv);
```

### Returns

None.

### Description

This function will close the specified interface, free the port-specific data structures and unset the port operation handler functions.

### Parameters

| Parameters | Description |
|---|---|
| drv | ILI9488 driver handle |

### Function

void ILI9488_Intf_Close( ILI9488_DRV *drv)

#### ILI9488_Intf_Open Function

Opens the specified port to the ILI9488 device

### File

help_drv_gfx_ili9488_common.h

## C

```
GFX_Result ILI9488_Intf_Open(ILI9488_DRV * drv, unsigned int index);
```

## Returns

- GFX_SUCCESS - Operation successful
- GFX_FAILURE - Operation failed

## Description

In SPI mode, this function will open the SPI port, allocate the port-specific data structures and set the port operation handler functions. When done using the port, ILI9488_Intf_Close must be called to free up the data structures and close the port.

## Parameters

| Parameters | Description |
|---|---|
| drv | ILI9488 driver handle |
| index | Port index |

## Function

GFX_Result ILI9488_Intf_Open(ILI9488_DRV *drv, unsigned int index)

### ILI9488_Intf_ReadCmd Function

Sends read command and reads response from ILI9488

## File

help_drv_gfx_ili9488_common.h

## C

```
GFX_Result ILI9488_Intf_ReadCmd(struct ILI9488_DRV * drv, uint8_t cmd, uint8_t * data, int bytes);
```

## Returns

- GFX_SUCCESS Operation successful
- GFX_FAILURE Operation failed

## Description

This function will fist write the the read command and then read back the response from the ILI9488 GRAM.

## Remarks

This function only supports 8-, 24- or 32-bit reads. This function performs multiple full-blocking write/read calls to the SPI port and won't return until the SPI transaction completes.

## Parameters

| Parameters | Description |
|---|---|
| drv | ILI9488 driver handle |
| cmd | Read command |
| data | Buffer to store the read data to |
| bytes | Number of bytes to read |

## Function

GFX_Result ILI9488_Intf_ReadCmd(struct ILI9488_DRV *drv,

uint8_t cmd,

uint8_t *data,

int bytes);

### ILI9488_Intf_ReadPixels Function

Read pixel data from specified position in ILI9488 GRAM

## File

help_drv_gfx_ili9488_common.h

## C

```
GFX_Result ILI9488_Intf_ReadPixels(struct ILI9488_DRV * drv, uint32_t x, uint32_t y, uint8_t * value,
unsigned int num_pixels);
```

## Returns

- GFX_SUCCESS - Operation successful
- GFX_FAILURE - Operation failed

## Description

This function will first write the start column, page information, then read the pixel data from the ILI9488 GRAM.

## Remarks

For SPI mode, this function performs multiple full-blocking write/read calls to the SPI port and won't return until the SPI transaction completes.

## Parameters

| Parameters | Description |
|---|---|
| drv | ILI9488 driver handle |
| x | Column position |
| y | Page position |
| value | Value to store the read pixel color (8-bit/pixel RGB) |
| num_pixels | Number of pixels to read |

## Function

GFX_Result ILI9488_Intf_ReadPixels(struct ILI9488_DRV *drv,
uint32_t x,
uint32_t y,
uint16_t *value,
unsigned int num_pixels)

### ILI9488_Intf_WriteCmd Function

Sends write command and parameters to the ILI9488 device

## File

help_drv_gfx_ili9488_common.h

## C

```
GFX_Result ILI9488_Intf_WriteCmd(struct ILI9488_DRV * drv, uint8_t cmd, uint8_t * parms, int num_parms);
```

## Returns

- GFX_SUCCESS - Operation successful
- GFX_FAILURE - Operation failed

## Description

This function will do a write operation to send the write command and its parameters to the ILI9488.

## Remarks

In SPI mode, this is a full-blocking call and waits for the SPI transaction to complete.

## Parameters

| Parameters | Description |
|---|---|
| drv | ILI9488 driver handle |
| cmd | Write command |
| parms | Pointer to array of 8-bit parameters |
| bytes | number of command parameters |

## Function

GFX_Result ILI9488_Intf_WriteCmd(struct ILI9488_DRV *drv,
uint8_t cmd,
uint8_t *parms,
int num_parms)

### ILI9488_Intf_WritePixels Function

Writes pixel data to ILI9488 GRAM from specified position

## File

help_drv_gfx_ili9488_common.h

## C

```
GFX_Result ILI9488_Intf_WritePixels(struct ILI9488_DRV * drv, uint32_t start_x, uint32_t start_y, uint8_t *
data, unsigned int num_pixels);
```

## Returns

- GFX_SUCCESS - Operation successful
- GFX_FAILURE - Operation failed

## Description

This function will fist write the start column, page information, then write the pixel data to the ILI9488 GRAM.

## Remarks

In SPI mode, this function performs multiple full-blocking write calls to the SPI port and won't return until the SPI transaction completes.

## Parameters

| Parameters | Description |
|---|---|
| drv | ILI9488 driver handle |
| start_x | Start column position |
| start_y | Start page position |
| data | Array of 8-bit pixel data (8-bit/pixel RGB) |
| num_pixels | Number of pixels |

## Function

GFX_Result ILI9488_Intf_WritePixels(struct ILI9488_DRV *drv,

uint32_t start_x,

uint32_t start_y,

uint8_t *data,

unsigned int num_pixels)

### b) Data Types and Constants

### ILI9488_CMD_PARAM Structure

Structure contains command and parameter information

## File

help_drv_gfx_ili9488_common.h

## C

```
typedef struct {
  uint8_t cmd;
  uint8_t parmCount;
  uint8_t parms[4];
} ILI9488_CMD_PARAM;
```

## Members

| Members | Description |
|---|---|
| uint8_t cmd; | Command |
| uint8_t parmCount; | Number of command parameters |
| uint8_t parms[4]; | Command parameters, max of 4 |

## Description

- ILI9488_CMD_PARAM

### ILI9488_DRV_STATE Enumeration

Enum of ILI9488 driver states

## File

help_drv_gfx_ili9488_common.h

## C

```
typedef enum {
  INIT = 0,
  RUN
} ILI9488_DRV_STATE;
```

## Description

- ILI9488_DRV_STATE

### ILI9488_DRV Structure

Structure contains driver-specific data and ops pointers

## File

help_drv_gfx_ili9488_common.h

## C

```
struct ILI9488_DRV {
  GFX_Context* gfx;
  ILI9488_DRV_STATE state;
  uint8_t * pixelBuffer;
  int currentLine;
  int lineX_Start;
  int lineX_End;
  GFX_Bool linePending;
  unsigned int bytesPerPixelBuffer;
  void * port_priv;
};
```

## Members

| Members | Description |
|---|---|
| GFX_Context* gfx; | GFX context pointer |
| ILI9488_DRV_STATE state; | Driver state |
| uint8_t * pixelBuffer; | Line buffer information |
| unsigned int bytesPerPixelBuffer; | bytes per pixel buffer |
| void * port_priv; | Port-specific private data |

## Description

- ILI9488_DRV

### ILI9488_Backlight_Off Macro

## File

help_drv_gfx_ili9488_common.h

## C

```
#define ILI9488_Backlight_Off BSP_DisplayBacklightStateSet(0)
```

## Description

This is macro ILI9488_Backlight_Off.

### ILI9488_Backlight_On Macro

## File

help_drv_gfx_ili9488_common.h

## C

```
#define ILI9488_Backlight_On BSP_DisplayBacklightStateSet(1)
```

## Description

This is macro ILI9488_Backlight_On.

**ILI9488_Reset_Assert Macro**

**File**

help_drv_gfx_ili9488_common.h

**C**

```
#define ILI9488_Reset_Assert BSP_DisplayResetStateSet(0)
```

**Description**

This is macro ILI9488_Reset_Assert.

**ILI9488_Reset_Deassert Macro**

**File**

help_drv_gfx_ili9488_common.h

**C**

```
#define ILI9488_Reset_Deassert BSP_DisplayResetStateSet(1)
```

**Description**

This is macro ILI9488_Reset_Deassert.

**ILI9488_SPI_DCX_Command Macro**

**File**

help_drv_gfx_ili9488_common.h

**C**

```
#define ILI9488_SPI_DCX_Command BSP_ILI9488_SPI_DCXStateSet(0)
```

**Description**

This is macro ILI9488_SPI_DCX_Command.

**ILI9488_SPI_DCX_Data Macro**

**File**

help_drv_gfx_ili9488_common.h

**C**

```
#define ILI9488_SPI_DCX_Data BSP_ILI9488_SPI_DCXStateSet(1)
```

**Description**

This is macro ILI9488_SPI_DCX_Data.

**ILI9488_SPI_SS_Assert Macro**

**File**

help_drv_gfx_ili9488_common.h

**C**

```
#define ILI9488_SPI_SS_Assert BSP_ILI9488_SPI_CSXStateSet(0)
```

**Description**

This is macro ILI9488_SPI_SS_Assert.

**ILI9488_SPI_SS_Deassert Macro**

**File**

help_drv_gfx_ili9488_common.h

**C**

```
#define ILI9488_SPI_SS_Deassert BSP_ILI9488_SPI_CSXStateSet(1)
```

**Description**

This is macro ILI9488_SPI_SS_Deassert.

## *Nano2D Graphics Processing Unit (GPU) Driver Library*

Provides information on the Nano 2-Dimensional (Nano2D) Driver Library.

### Description

Nano 2-Dimensional (Nano2D) is a driver library for rendering 2-Dimensional computer graphics. It is the means for hardware-accelerated graphics on PIC32MZ microcontrollers containing the 2-Dimensional Graphics Processing Unit (2-D GPU).

## Introduction

This topic provides the introduction to the Nano2D Driver Library.

### Description

The Nano2D Library API provides full functionality of the PIC32MZ 2-D GPU module, which includes lines, rectangles, bit block transfers (blits), transparency, and binary Raster Operations (ROP2). These features are employed in GFX application demonstrations compiled with Nano2D on 2-D GPU enabled microcontrollers. Blits can be used to quickly transfer pre-rendered images directly to the display's frame buffer. (Pre-rendering images converts image pixels from JPEG or some other format into raw RGB or RGBA bits.)

In Harmony 2.06 there is a new tool, the DDR Organizer, that assists in managing buffers, raw images, and other memory resources in the DDR memory of DA devices. Images can quickly and easily be organized for pre-rendering at boot-up so that later image draws will use the device's Nano2D GPU.

The library provides low-level 2D primitives while containing no facilities for GUI development, therefore, it can be used as a stand-alone API. If GUI development is desired, it can be achieved using a higher level widget library such as, libAria which is embedded in the MPLAB Harmony Graphics Suite.

**Note:** Nano2D requires MPLAB Harmony v2.02b or later.

MPLAB Harmony Graphics Suite has two memory rendering options:

- Aria – Uses Hardware-independent software rasterizer – software fallback
- Nano2D – Uses 2-D GPU hardware accelerated peripheral

In MPLAB Harmony v2.03 and later, the MPLAB Harmony Graphics Suite uses Nano2D as its default memory buffer rendering interface when developing for PIC32 2D-GPU enabled devices. This default can be overridden in MHC by user deselection of Nano2D Graphics Processor.

Nano2D provides the API for drawing accelerated raster graphics onto memory buffers. The actual drawing happens in the 2-D GPU peripheral. Nano2D can be a better option over libAria's hardware-independent graphics primitives because Nano2D uses little to no CPU resources. The use of the Nano2D library requires that the DA's built-in 2D graphics processor be enabled. Under *Harmony Framework Configuration* > *Graphics Stack* > *Graphics Processor*, select the NANO 2D processor.

## Using the Library

This topic describes how to use the Nano2D Driver Library.

### Description

**Interface Header File**: `libnano2d.h`

The interface to the Nano2D Library is defined in the `libnano2d.h` header file. Any C language source (`.c`) file that uses the Nano2D Library should include `libnano2d.h`. The header file can be found within the following MPLAB Harmony directory.

`<install_dir>/framework/gfx/driver/processor/nano2d/libnano2D.a`

**Library File**: `libnano2d.a`

The Nano2D Library archive (libnano2d.a) file is installed with MPLAB Harmony. The header file can be found within the following MPLAB Harmony directory.

`<install_dir>/framework/bin/driver/processor/nano2d/libnano2D.a`

Please refer to What is MPLAB Harmony? for how the Nano2D Library interacts with the framework.

`<install_dir>/framework/gfx/driver/processor/nano2d/libnano2D.h`

### Abstraction Model

Provides information on the abstraction model for the library.

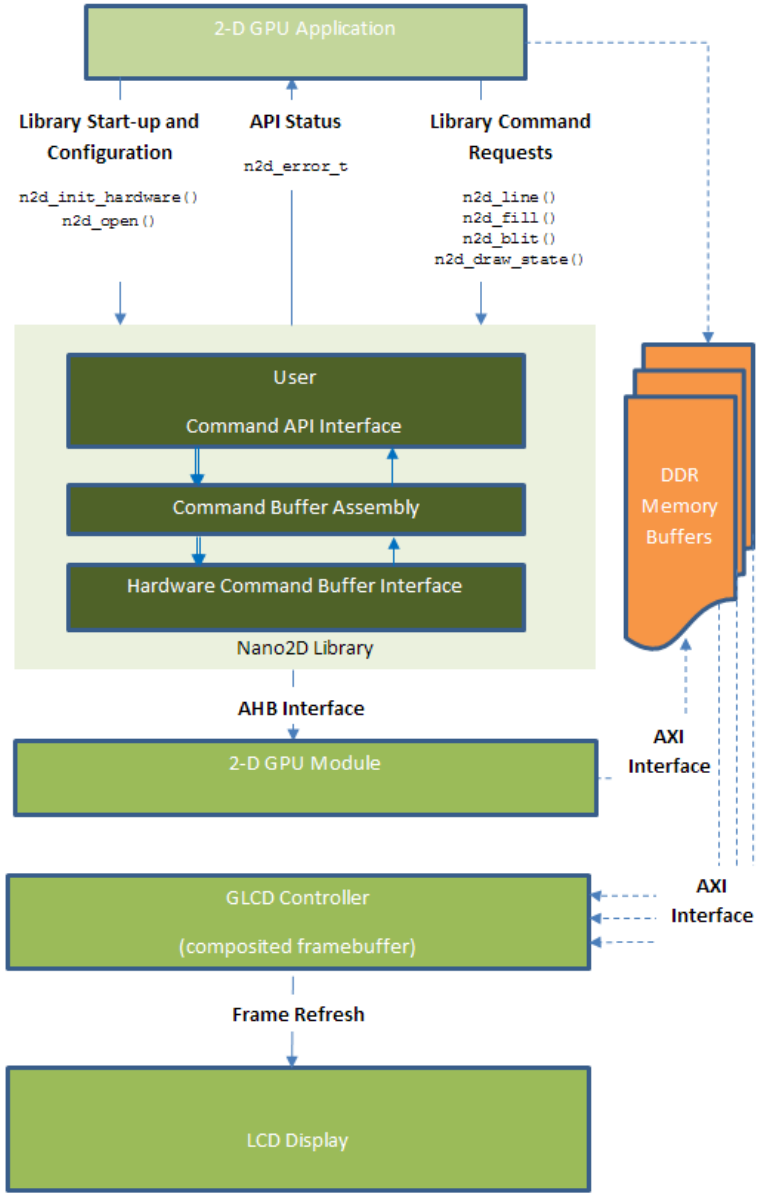
### Description

The Nano2D Driver Library is the single interface to the 2-D GPU Module. The 2-D GPU does not have Special Function Register (SFR) access like many PIC32 peripherals. Alternatively, Nano2D uses a command buffer, shared between it and the 2-D GPU, to communicate request. Nano2D Driver Library builds the command buffer content based on each API request and makes it available at a physical address known to the GPU. The address of the command buffer address location is established at startup before any request arrives.

The communication between application and Nano2D is C synchronous function calls. Each function returns on completion of requested. This is typically between 0 -1ms. The communication between Nano2D and 2-GPU is the command buffer and internal well-known status registers. These registers are not exposed to the application.

The Nano2D Driver Library commits a complex command buffer protocol for each GPU request. The Nano2D Driver Library removes the overhead of command buffer assembly and status from the application. This leaves the application with an easy to use, synchronous, return code interface.

**Nano2D Software Abstraction Block Diagram**




### Library Overview

Refer to the Driver Library Overview section for information on how the driver operates in a system.

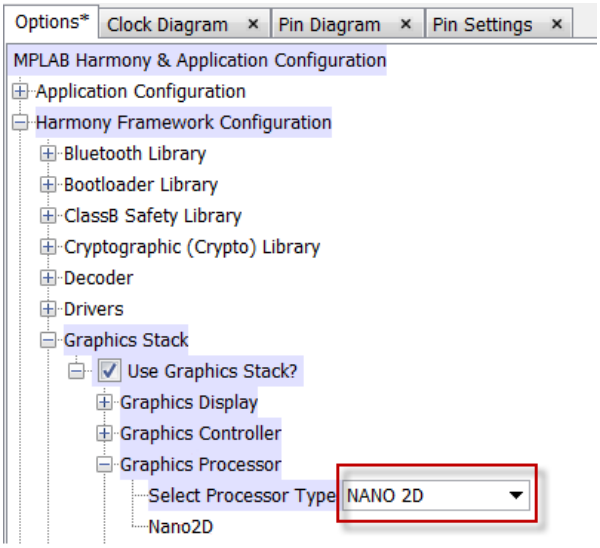
The library interface routines are divided into various sub-sections, which address one of the blocks or the overall operation of the Nano2D Driver.

### How the Library Works

This topic provides information on how the library works.

## Description

The Nano2D Driver Library service provides general APIs for graphics application use. To enable Nano2D, the user is required to select the Nano2D as the graphics processor using MHC within the “Options*” tab. Upon generation, the `libnano2d.a` library, Nano2D initialization code, and header file will be added to the project. There is no additional configuration required.




The following app.c code example shows typical usage of this Nano2D Library.
[Code Example]

```
void APP_Tasks ( void )
{
/* Check the application's current state. */
switch ( appData.state )
{
/* Application's initial state. */
case APP_STATE_INIT:
{
bool appInitialized = true;

if (appInitialized)
{
appData.state = APP_STATE_GPU_FILL;
}
break;
}

case APP_STATE_GPU_FILL:
{
n2d_rectangle_t rect;
n2d_color_t redColor = 0x00ff000000;
n2d_color_t greenColor = 0x0000ff0000;

/* create GPU buffer for GLCD layer 0 (dest) */
layer0.format = N2D_RGBA8888;
layer0.gpu = KVA_TO_PA(0xA8000000); //GLCD display layer 0
layer0.memory = (void*)0xA8000000;
layer0.width = 480; // width of buffer same as display width
layer0.height = 272; // height of buffer same as display height
layer0.orientation = N2D_0;
layer0.stride = layer0.width * 32 / 8;

/* fill entire buffer 480x272 */
n2d_fill(layer0, N2D_NULL, redColor, N2D_BLEND_NONE);

/* create a clipping rectangle for layer0 */
rect.x = 0; rect.y = 0; rect.width = 100; rect.height = 100;

/* fill 100x100 rectangle with green ADDITIVE blend to
make color brown */
n2d_fill(layer0, &rect, greenColor, N2D_BLEND_ADDITIVE);

appData.state = APP_STATE_IDLE;

}
}
```

## Drawing

Nano2D provides basic drawing primitives: n2n_line and n2n_rect. These functions support integer only coordinates. If more complex graphics is required, the user will need to use a higher level graphics library, such as Aria Graphics Library which is one part of the MPLAB Harmony Graphics Suite. Aria can be configured to call Nano2D as its 2D primitive memory interface renderer for acceleration.

The Line operation, n2n_line, draws a line. Two points are given: start point and end point. The end point is not drawn for point to point figure drawing is required. Lines are rendered using the Breshenham algorithm. Clipping is supported for lines on a per pixel basis. To draw a line, use the following statement:

```
n2d_line(destination, start, end, clip, color, blend);
```

The Rectangle operation, n2n_rect, draws a rectangle. It fills or draws a rectangle area with a given color. A rectangle is given the top left coordination and bottom right coordinate of the fill region, the fill color, clipping region, and blending mode which is applied to each pixel. To draw a rectangle, use the following statement:

```
n2d_fill(destination, rectangle, color, blend);
```

There are 7 types of blend modes:

- N2D_BLEND_NONE - S, i.e. no blending
- N2D_BLEND_SRC_OVER - S + (1 - Sa) * D
- N2D_BLEND_DST_OVER - (1 - Da) * S + D
- N2D_BLEND_SRC_IN - Da * S
- N2D_BLEND_DST_IN - Sa * D
- N2D_BLEND_ADDITIVE - S + D
- N2D_BLEND_SUBTRACT - D * (1 - S)

Where, Sa and Da represent the source and destination alpha channels.

## Block Transfers of Pixels (Blitting)

Nano2D provides the following blit operations: Blit, Stretch/Shrink Blit, Mask, Blit, and automatic Filter blit during stretch and shrink.

The Blit operation transfers data from one area of a memory source to another area of a memory destination. The source and destination can be from the same or from different memory locations. Both source and destination must be described by a rectangle. Blitting supports automatic behavior:

- Stretch/Shrink – If source and destination rectangles are different sizes the operation becomes a stretch or a shrink blit
- Mask – use of ROPs for transparent pixels
- Monochrome – using ROPs for monochrome images

Stretch blit is not allowed to overlap, that is no part of the source and destination can share the same portion of memory. Non-stretch blits can overlap.

Blits supports the 7 blending modes discussed previously which are applied to each pixel.

To blit, use the following statement:

```
n2d_blit(dst, dst_rect, src, src_rect, blend);
```

## Transparency

Nano2D provides a means of controlling transparency applied to each pixel for subsequent draw commands. Transparency is also synonymous to masking and operation. Nano2D uses Binary Raster Operations (ROP2) to affect action on each pixel during blitting. The user can set a transparency mode or turn off transparency.

The Draw State operation, n2n_draw_state, sets the drawing operation for subsequent n2d_blit() calls.

To change draw state, use the following statement:

```
n2d_draw_state (transparency, color, src, foreground_rop, background_rop);
```

The statement is executed like the C condition “?” ternary operator that takes three values. It reads: If color in transparency_mode is true then perform foreground_rop on matching color otherwise perform background_rop on all other non-matching colors. Color equates to a pixel.

The following standard Binary ROPs are supported:

| ROP | Formula | Description |
|---|---|---|
| 0x0 | `0` | Set all destination bits to 0. |
| 0x1 | `~(D\|S)` | Inverse of merge source and destination. |
| 0x2 | `D&~S` | Inverse of merge source and destination. |
| 0x3 | `~S` | Inverse of merge source and destination. |
| 0x4 | `S&~D` | Mask source and inverse of destination. |
| 0x5 | `~D` | Invert destination. |
| 0x6 | `D^S` | Exclusive or of source and destination. |
| 0x7 | `~(D&S)` | Inverse of mask source and destination. |

| | | |
|---|---|---|
| 0x8 | D&S | Inverse of mask source and destination. |
| 0x9 | ~(D^S) | Inverse of mask source and destination. |
| 0xA | D | Copy destination. |
| 0xB | D\|~S | Merge inverse of source and destination. |
| 0xC | S | Copy source. |
| 0xD | S\|~D | Merge source and inverse of destination. |
| 0xE | D\|S | Merge source and destination. |
| 0xF | 1 | Set all destination bits to 1. |

## Clipping

Nano2D provides a clipping rectangle for line, rectangles and blits. Clipping is performed on a per pixel basis. For all functions, the clipping area is defined by n2d_rectangle_t.

## Memory Buffers

Nano2D provides an abstraction over a memory buffer region. Each function defined in `libnano2D.h` uses the n2d_buffer_t to establish the portion of shared memory used as a source or destination buffer. The structure contains all the information the libnano2D APIs is required to complete a GPU render command request. See "Data Types and Constants" in the Library Interface section.

The Nano2D library supports four buffer and their alpha swizzle formats. They are: RGBA8888, RGB565, RGB4444, and A8. When using Nano2D through Aria Library, only supports the two major RGB formats RGB8888 and RGB565.

RGBA8888 (24bit true-color) memory buffer establishes green blue red channels for color space with an extra alpha channel for color blending. Requiring 32bits per-pixel, it will consume a large buffer space to hold display content.

RGB565 (reduced color) memory buffer uses half the color space, has no alpha channel, and, with CPU processing, proves to be a higher performer than RGBA8888.

A8 is an all-alpha buffer. It is can be used to produced a greyscale (dimming) of an existing RBGA8888 buffer.

The buffer supports the following:

- `width` - Width of the buffer in pixels
- `height` - Height of the buffer in pixels
- `stride` - Stride of the buffer in bytes
- `format` - Pixel format of the buffer
- `orientation` - Buffer's orientation: 0, 90, 180, 270
- `memory` - Logical pointer to the buffer's memory for the CPU
- `gpu` - Physical address of the buffer's memory the hardware can access

## Detailed Use

The 2-D GPU is made available through the Nano2D Library Module. In future releases, MHGC will be its higher level access. The remainder of these sections will describe how to command the 2-D GPU directly without using MHGC using customized C code.

Unlike most controller peripherals, 2-D GPU command register is not available to application developers. There are no Special Function Registers (SFRs). As a replacement to SFR access, the Nano2D Library provides a C interface API implemented within the libnano2D.a object file. When linked to the application, these entities provide command access to 2-D GPU.

Nano2D Library and 2-D GPU communicate through a shared buffer region residing in physical memory. This memory location is provided by default in Harmony. For rendering, Nano2D uses user supplied source and destination buffers. These buffers communicate the necessary details of the memory region which includes size, color depth, location, and orientation. One or more of these buffers are required for 2-D GPU rendering APIs using `libnano2D.a`.

## Creating a Pixel Buffer

An n2d_buffer_t data structure maintains the context of the rendering memory buffer. A memory buffer can point to a scratch buffer or to the active framebuffer. In these examples, we will draw to the active framebuffer as well as scratch memory buffers.

The characteristics of the currently displayed framebuffer must be understood. These include the physical address of the framebuffer, its size, color depth, and orientation. An n2d_buffer_t structure must be created to contain this information. If the n2d_buffer_t is different than the framebuffer, rendered graphics will have unexpected behavior. After generation, the user will need to view the generated `system_config.h` file and `libnano2d.h`. Use the following steps to create an n2d_buffer_t structure that points to the active framebuffer.

```
n2d_buffer_t layer0;

layer0.width = GFX_GLCD_LAYER0_RES_X; // see system_config.h
layer0.height = GFX_GLCD_LAYER0_RES_Y;
layer0.stride = layer0.width * 32 /8; // 32bits/8 == 4 bytes
layer0.format = N2D_RGBA8888; // Red Green Blue Alpha 32bbp
layer0.orientation = N2D_0; // 0 degree orientation
```

```
layer0.handle = GFX_NULL; // handle is unused
layer0.memory = (void*)GFX_GLCD_LAYER0_BASEADDR;
layer0.gpu = KVA_TO_PA(GFX_GLCD_LAYER0_BASEADDR);
```

`layer0.gpu` is the starting location in DDR memory from which 2-D GPU will read or write. This is a physical memory address. If the application must modify data at this location, it must use layer0.memory which maintains the virtual (accessible) address.

## Drawing a Grid of Lines

Lines are primitive graphic items necessary to higher level widgets. To render a line directly to the framebuffer, the user will need to know the location of the framebuffer, the start and end of the line, the color, and blend factors of the line.

**Note:** The color parameter is in ARGB format and is not aligned with the buffer format. The following code example demonstrates line drawing by creating a perpendicular grid.

```
n2d_buffer_t * buffer = layer0;
n2d_point_t start, end;
n2d_color_t color;
n2d_int32_t i;

color = 0xff00ffff;

n2d_fill(buffer, N2D_NULL, color, N2D_BLEND_NONE);

/* Draw vertical line. */
start.x = 0;
start.y = 5;
end.x = 0;
end.y = buffer->height - 5;

color = 0xffff0000;

for (i = 0; i < buffer->width / 10; i++)
{
n2d_line(buffer, start, end, N2D_NULL, color, N2D_BLEND_NONE);

start.x += 10;
end.x += 10;
}

/* Draw horizontal line. */
start.x = 5;
start.y = 0;
end.x = buffer->width - 5;
end.y = 0;

for (i = 0; i < buffer->height / 10; i++)
{
n2d_line(buffer, start, end, N2D_NULL, color, N2D_BLEND_NONE);

start.y += 10;
end.y += 10;
}
```

## Drawing Cascading Blended Rectangles

Rectangles are primitive graphic items necessary to higher level widgets. The 2-D GPU can render these quickly with alpha-blending and orientation settings. To render cascading rectangles directly to the framebuffer, the user will need to know the location of the framebuffer, the top, left, width, height (rect), the color, and blend factors of the rect. The following code example integrates rectangle drawing by creating a rectangles in a cascading order using different colors.

```
n2d_buffer_t * buffer = layer0;
n2d_point_t start, end;
n2d_color_t color;
n2d_int32_t i;

/* Clear background color to black. */
color = 0xff000000;
n2d_fill(buffer, N2D_NULL, color, N2D_BLEND_NONE);

rect.x = 0;
```

```
rect.y = 0;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;
n2d_fill(buffer, &rect, 0xffff0000, N2D_BLEND_SRC_OVER);

rect.x = buffer->width / 8;
rect.y = buffer->height / 8;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;

n2d_fill(buffer, &rect, 0x7f00ff00, N2D_BLEND_SRC_OVER);

rect.x = buffer->width / 4;
rect.y = buffer->height / 4;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;
n2d_fill(buffer, &rect, 0xffffffff, N2D_BLEND_SRC_OVER);

rect.x = buffer->width * 3 / 8;
rect.y = buffer->height * 3 / 8;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;
n2d_fill(buffer, &rect, 0x700000ff, N2D_BLEND_SRC_OVER);

rect.x = buffer->width * 4 / 8;
rect.y = buffer->height * 4 / 8;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;
n2d_fill(buffer, &rect, 0xffffff00, N2D_BLEND_SRC_OVER);

rect.x = buffer->width * 5 / 8;
rect.y = buffer->height * 5 / 8;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;
n2d_fill(buffer, &rect, 0x7fff00ff, N2D_BLEND_SRC_OVER);

rect.x = buffer->width * 6 / 8;
rect.y = buffer->height * 6 / 8;
rect.width = buffer->width / 4;
rect.height = buffer->height / 4;
n2d_fill(buffer, &rect, 0xff00ffff, N2D_BLEND_SRC_OVER);
```

## Image Source and Destination Rotation

A copy of one buffer source region to another is called blitting. The example below, blits an image rendered to a source buffer onto a destination buffer. The destination buffer points to the framebuffer. For this example, a staging src buffer to hold the image is established. Also a small set of GFX Hal APIs are used to decode the image to the src buffer.

```
n2d_buffer_t src0;
src0.format = N2D_RGBA8888;
src0.gpu = KVA_TO_PA(0xA8465000);
src0.memory = (void*)0xA8465000;
src0.width = 256;
src0.height = 256;
src0.orientation = N2D_0;
src0.stride = src0.width * 32 / 8;

/* use gfx hal to render image to source buffer 0 */
GFX_PixelBufferCreate(256,
256,
GFX_COLOR_MODE_RGBA_8888,
(uint32_t*)src0.memory,
&pixelBuffer);

GFX_Begin();
GFX_Set(GFXF_DRAW_TARGET, &pixelBuffer);
GFXU_DrawImage(&image0,
0,
0,
256,
```

```
256,
0,
0,
NULL,
NULL);
GFX_Set(GFXF_DRAW_TARGET, NULL);
GFX_End();

/* use gfx hal to render image to source buffer 1 */
GFX_PixelBufferCreate(256,
256,
GFX_COLOR_MODE_RGBA_8888,
(uint32_t*)src1.memory,
&pixelBuffer);

GFX_Begin();
GFX_Set(GFXF_DRAW_TARGET, &pixelBuffer);
GFXU_DrawImage(&image1,
0,
0,
256,
256,
0,
0,
NULL,
NULL);
GFX_Set(GFXF_DRAW_TARGET, NULL);
GFX_End();
```

## Raster Operations

The processing of source pixels onto destination pixels is called a raster operation. The 2-D GPU through the libnano2D library support two input (binary) Raster Operations (ROP2). These are industry standard bitwise logical operations defined by 16 possible functions listed above in the Transparency section.

In order to set the operation for subsequent draw functions, n2d_buffer_t is used. The function can be continuously used to set mask/filter operations or turn transparency off.

Its arguments are similar to C conditional “?” statements. The first argument determines the operation mode. The operation mode determines whether color is to be applied to N2D_TRANSPARENCY_NONE (no pixels) or N2D_TRANSPARENCY_SOURCE or N2D_TRANSPARENCY_DESTINATION buffer. If the color matches, then the foreground operation is applied, otherwise the background operation is applied.

Consider the following example statement:

```
n2d_draw_state(N2D_TRANSPARENCY_SOURCE, 0xff0000, 0xe, 0xc);
```

The statement will inform the GPU to look for the color red in the source the source buffer during n2d_blit. If the pixel is found, the final pixel will become a merge of the original source and destination pixels, otherwise the pixel will become a copy of the original source pixel.

The following code example applies all 16 ROP functions on green (source) and blue (destination) pixels.

```
n2d_buffer_t src, *buffer;
n2d_uint8_t rop = 0;
n2d_uint32_t x, y;
n2d_int32_t deltaX, deltaY;
n2d_rectangle_t rect;

buffer = &layer0;

deltaX = buffer->width >> 2;
deltaY = buffer->height >> 2;

/* Fill the source buffer with green color. */
n2d_fill(&src, N2D_NULL, 0xff00, N2D_BLEND_NONE);

/* Fill the dst buffer with blue color. */
n2d_fill(buffer, N2D_NULL, 0xff, N2D_BLEND_NONE);

/* Loop all rop values. */
for (y = 0; y < 4; y++)
{
for (x = 0; x < 4; x++)
{
rect.x = x * deltaX;
```

```
rect.y = y * deltaY;
rect.width = deltaX;
rect.height = deltaY;

/* Set rop. */
n2d_draw_state(N2D_TRANSPARENCY_NONE, 0x0, rop, rop);

n2d_blit(buffer, &rect, &src, &rect, N2D_BLEND_NONE);
rop++;
}
}

n2d_draw_state(N2D_TRANSPARENCY_NONE, 0x0, 0xc, 0xc);
Alpha Greyscale -<< Red Color "section"
/* alpha buffer for blending */
alpha.format = N2D_A8;
alpha.gpu = KVA_TO_PA(0xA85DC000);
alpha.memory = (void*)0xA85DC000;
alpha.width = appData.display_info->rect.width;
alpha.height = appData.display_info->rect.height;
alpha.orientation = orientation;
alpha.stride = alpha.width * 8 / 8;
/* dim pixels within a rectangle area */
/* Init the alpha buffer with an alpha channel. */
memset(alpha.memory, 0x07, alpha.stride * alpha.height);
/* Blit – subtract alpha value on all pixels of destination */
 n2d_blit(&layer0, N2D_NULL, &alpha, N2D_NULL, N2D_BLEND_SUBTRACT);
```

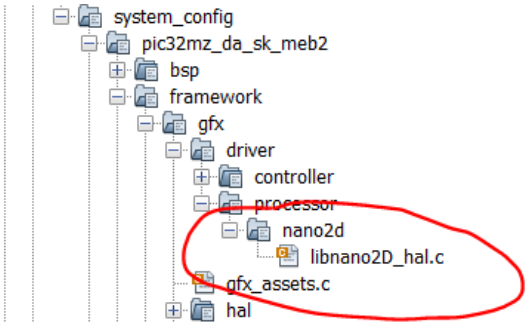
## Configuring the Library

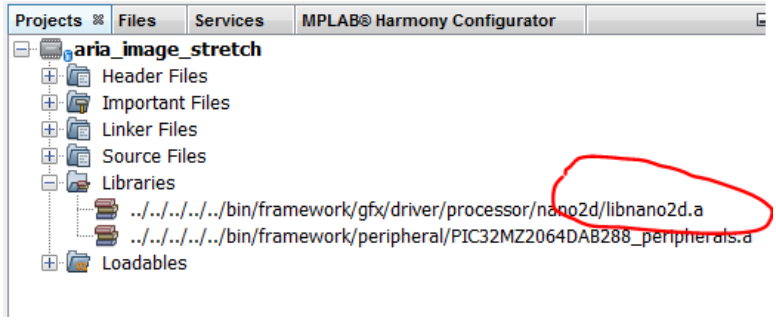
This topic describes how to configure the Nano2D Driver Library.

## Description

The configuration of the Nano2D module is based on Nano2D processor selections of MHC. There are no additional configurations available for Nano2D. To enable Nano2D, the user is required to select the Nano2D as the graphics processor using MHC under “Options*”. Upon generation, the `libnano2d.a` library, Nano2D initialization code, and header file will be added to the project.

Upon generation, the `libnano2d.a` library, Nano2D initialization code, Nano2D thin adapter file, and header file will be added to the project.



## Building the Library

This topic explains building the Nano2D Driver Library.

### Description

The library is provided in binary form only, and comes prebuilt when you install MPLAB Harmony. In addition, the library is added to your project when Nano2D is selected and the project is generated.

The following three tables list and describe the header (`.h`) and source (`.c`) files that implement this library. The parent folder for these files is `<install-dir>/framework/gfx/driver/processor/nano2d`.

**Interface File(s)**

This table lists and describes the header files that must be included (i.e., using `#include`) by any code that uses this library.

| Source File Name | Description |
|---|---|
| `/libnano2D.h` | Header file that exports the driver API. |

**Library File(s)**

MHC ***All of the required files listed in the following table are automatically added into the MPLAB X IDE project by the MHC when the library is selected for use.***

This table lists and describes the source and header files that must *always* be included in the MPLAB X IDE project to build this library.

| Source File Name | Description |
|---|---|
| `/libnano2D.a` | The static library that contains the implementation of `libnano2D.h`. |

**Optional File(s)**

This table lists and describes the source and header files that may optionally be included if required for the desired implementation.

| Source File Name | Description |
|---|---|
| N/A | No optional files are available for this library. |

**Module Dependencies**

The Nano2D Driver Library depends on the following modules:

- GLCD Graphics Display Controller (currently, the Nano2D module is initialized within the GLCD display controller)

## Library Interface

This section describes the Application Programming Interface (API) functions of the Nano2D module.

### a) Functions

| | Name | Description |
|---|---|---|
| | n2d_blit | Copy a source buffer to the the destination buffer |
| | n2d_draw_state | Set the drawing state for any following Nano2D API draw call |
| | n2d_fill | Fill a (partial) buffer with a specified color |
| | n2d_init_hardware | Initializes the n2d driver and peripheral hardware |
| | n2d_line | Draw a line |
| | n2d_open | Open Nano2d context |
| | n2d_dither | Enable or disable dithering |

### b) Data Types and Constants

| | Name | Description |
|---|---|---|
| | n2d_blend | List of blending modes |
| | n2d_buffer | A wrapper structure for any image or render target |
| | n2d_buffer_format | List of blending modes |
| | n2d_error | Error codes that the Nano2D functions can return |
| | n2d_module_parameters | GPU peripheral Initialization parameters |
| | n2d_orientation | List of blending modes |

| | | |
|---|---|---|
| | n2d_point | A position on a pixel |
| | n2d_rectangle | A rectangle |
| | n2d_transparency | Transparency modes |
| | n2d_blend_t | List of blending modes |
| | n2d_buffer_format_t | List of blending modes |
| | n2d_buffer_t | A wrapper structure for any image or render target |
| | n2d_color_t | Identifies a specific pixel color |
| | n2d_error_t | Error codes that the Nano2D functions can return |
| | n2d_module_parameters_t | GPU peripheral Initialization parameters |
| | n2d_orientation_t | List of blending modes |
| | n2d_point_t | A position on a pixel |
| | n2d_rectangle_t | A rectangle |
| | n2d_transparency_t | Transparency modes |
| | n2d_bool_t | This is type n2d_bool_t. |
| | n2d_float_t | This is type n2d_float_t. |
| | n2d_int16_t | This is type n2d_int16_t. |
| | n2d_int32_t | This is type n2d_int32_t. |
| | n2d_size_t | This is type n2d_size_t. |
| | n2d_uint16_t | This is type n2d_uint16_t. |
| | n2d_uint32_t | This is type n2d_uint32_t. |
| | n2d_uint64_t | This is type n2d_uint64_t. |
| | n2d_uint8_t | This is type n2d_uint8_t. |
| | __gcmALIGN | This is macro __gcmALIGN. |
| | __gcmEND | This is macro __gcmEND. |
| | __gcmGETSIZE | This is macro __gcmGETSIZE. |
| | __gcmMASK | This is macro __gcmMASK. |
| | __gcmSTART | This is macro __gcmSTART. |
| | _nano2D_types_h__ | This is macro _nano2D_types_h__. |
| | gcmALIGN | This is macro gcmALIGN. |
| | gcmCOUNTOF | This is macro gcmCOUNTOF. |
| | gcmGETFIELD | This is macro gcmGETFIELD. |
| | gcmINT2PTR | This is macro gcmINT2PTR. |
| | gcmMAX | This is macro gcmMAX. |
| | gcmMIN | This is macro gcmMIN. |
| | gcmPTR2INT | This is macro gcmPTR2INT. |
| | gcmSETFIELD | This is macro gcmSETFIELD. |
| | gcmSETFIELDVALUE | This is macro gcmSETFIELDVALUE. |
| | gcmSETMASKEDFIELD | This is macro gcmSETMASKEDFIELD. |
| | gcmSETMASKEDFIELDVALUE | This is macro gcmSETMASKEDFIELDVALUE. |
| | gcmVERIFYFIELDVALUE | This is macro gcmVERIFYFIELDVALUE. |
| | IN | This is macro IN. |
| | N2D_FALSE | This is macro N2D_FALSE. |
| | N2D_INFINITE | This is macro N2D_INFINITE. |
| | N2D_IS_ERROR | This is macro N2D_IS_ERROR. |
| | N2D_IS_SUCCESS | This is macro N2D_IS_SUCCESS. |
| | N2D_NULL | This is macro N2D_NULL. |
| | N2D_ON_ERROR | This is macro N2D_ON_ERROR. |
| | N2D_TRUE | This is macro N2D_TRUE. |
| | OUT | This is macro OUT. |

## Description

Refer to each section for a detailed description.

### a) Functions

### n2d_blit Function

Copy a source buffer to the the destination buffer

## File

libnano2D.h

## C

```
n2d_error_t n2d_blit(n2d_buffer_t * destination, n2d_rectangle_t * destination_rectangle, n2d_buffer_t *
source, n2d_rectangle_t * source_rectangle, n2d_blend_t blend);
```

## Returns

Returns the status as defined by n2d_error_t

## Description

The specified region of the source buffer is copied to the specified region of the destination buffer. If the regions are different in size, simple low-quality scaling will automatically be performed.

An optional blend mode can be specified that defines the blending of the source onto the destination.

## Remarks

This function will wait until the hardware is complete, i.e. it is synchronous.

## Parameters

| Parameters | Description |
|---|---|
| destination | Pointer to a n2d_buffer_t structure that describes the destination of the blit |
| destination_rectangle | Optional pointer to the rectangle that defines the region inside the destination buffer. If this rectangle is not specified, the entire destination buffer is used as the destination region |
| source | Pointer to a n2d_buffer_t structure that describes the source of the blit |
| source_rectangle | Optional pointer to the rectangle that defines the region inside the source buffer. If this rectangle is not specified, the entire source buffer is used as the source region |
| blend | Optional blending mode to be applied to each pixel. If no blending is required, set this value to N2D_BLEND_NONE (0) |

## Function

n2d_error_t n2d_blit()

### n2d_draw_state Function

Set the drawing state for any following Nano2D API draw call

## File

libnano2D.h

## C

```
n2d_error_t n2d_draw_state(n2d_transparency_t transparency, n2d_color_t color, n2d_uint8_t foreground_rop,
n2d_uint8_t background_rop);
```

## Returns

Returns the status as defined by n2d_error_t

## Description

In order to setup transparency for the n2d_blit function, this function needs to be called. Note that this function is static, so set once, all draw commands that follow this function will take this transparency into effect. Call this function again with different parameters to set another transparency mode or turn transparency off.

It will return N2D_INVALID_ARGUMENT if the source defines the transparency but the rop has nothing to do with the source buffer.

The default transparency mode for any newly opened context is N2D_TRANSPARENCY_NONE, using a foreground_rop of 0xC (copy source).

Binary ROPs supported in both foreground and background operations: ROP Formula Description 0x0 0 Set all destination bits to 0. 0x1 ~(D|S) Inverse of merge source and destination. 0x2 D&~S Mask inverse of source and destination. 0x3 ~S Copy inverse of source. 0x4 S&~D Mask source and inverse of destination. 0x5 ~D Invert destination. 0x6 D^S Exclusive or of source and destination. 0x7 ~(D&S) Inverse of mask source and destination. 0x8 D&S Mask source and destination. 0x9 ~(D^S) Inverse of exclusive or of source and destination. 0xA D Copy destination. 0xB D|~S Merge inverse of source and destination. 0xC S Copy source. 0xD S|~D Merge source and inverse of destination. 0xE D|S Merge source and destination. 0xF 1 Set all destination bits to 1.

## Remarks

When using a source buffer with the A8 pixel format, transparency must be enabled to N2D_TRANSPARENCY_SOURCE and the alpha channel of color will be used to check for transparency. If the pixel is not transparent, the RGB channels of color value will be used as the color for the pixel.

## Parameters

| Parameters | Description |
|---|---|
| transparency | The transparency mode applied to each pixel. See n2d_transparency_t for a list of all supported transparency modes |
| color | If transparency is not N2D_TRANSPARENCY_NONE, this color value specifies if a pixel is a foreground or a background pixel. If the color matches, it is a background pixel, otherwise it is a foreground pixel |
| foreground_rop | A Binary ROP (ROP2) code that gets executed by the hardware for each foreground pixel |
| background_rop | A Binary ROP (ROP2) code that gets executed by the hardware for each background pixel |

## Function

n2d_error_t n2d_draw_state()

### n2d_fill Function

Fill a (partial) buffer with a specified color

## File

libnano2D.h

## C

```
n2d_error_t n2d_fill(n2d_buffer_t * destination, n2d_rectangle_t * rectangle, n2d_color_t color,
n2d_blend_t blend);
```

## Returns

Returns the status as defined by n2d_error_t

## Description

Draws and fills a rectangle with a specific color onto destination buffer.

An optional blend mode can be specified that defines the blending of the color onto the destination.

## Remarks

This function will wait until the hardware is complete, i.e. it is synchronous

## Parameters

| Parameters | Description |
|---|---|
| destination | Pointer to a n2d_buffer_t structure that describes the buffer to be filled |
| rectangle | Pointer to a rectangle that specifies the area to be filled. If rectangle is NULL, the entire buffer will be filled with the specified color |
| color | The color value to use for filling the buffer |
| blend | The blending mode to be applied to each pixel. If no blending is required, set this value to N2D_BLEND_NONE (0) |

## Function

n2d_error_t n2d_fill()

### n2d_init_hardware Function

Initializes the n2d driver and peripheral hardware

## File

libnano2D.h

## C

```
n2d_error_t n2d_init_hardware(n2d_module_parameters_t * params);
```

## Returns

Returns the status as defined by n2d_error_t

## Description

The initializes the memory region, sets base address, establishes the irq and connects the hardware to application.

## Remarks

For PIC32MZ DA, registerMemBase2D is 0xBF8EB000 and baseAddress is 0

## Parameters

| Parameters | Description |
|---|---|
| params | Initialization parameters. See n2d_module_parameters_t |

## Function

n2d_error_t n2d_init_hardware()

### n2d_line Function

Draw a line

## File

libnano2D.h

## C

```
n2d_error_t n2d_line(n2d_buffer_t * destination, n2d_point_t start, n2d_point_t end, n2d_rectangle_t *
clip, n2d_color_t color, n2d_blend_t blend);
```

## Returns

Returns the status as defined by n2d_error_t

## Description

Draw a line with a specific color. The last pixel of the line will not be drawn.

An optional blend mode can be specified that defines the blending of the color onto the destination.

## Remarks

This function will wait until the hardware is complete, i.e. it is synchronous

## Parameters

| Parameters | Description |
|---|---|
| destination | Pointer to a n2d_buffer_t structure that describes the buffer to be used to draw the line into. |
| start | The starting point of the line, given in destination coordinates. |
| end | The ending point of the line, given in destination coordinates. The last pixel will not be drawn. |
| clip | Optional pointer to a rectangle that specifies the clipping region of the destination. If clip is NULL, the clipping region will be the entire destination buffer. |
| color | The color value to use for drawing the line. |
| blend | The blending mode to be applied to each pixel on the line. If no blending is required, set this value to N2D_BLEND_NONE (0). |

## Function

n2d_error_t n2d_line()

### n2d_open Function

Open Nano2d context

## File

libnano2D.h

## C

```
n2d_error_t n2d_open();
```

## Returns

Returns the status as defined by n2d_error_t.

## Description

The n2d_line, n2d_fill, n2d_blit, and n2d_draw_state functions require a Nano2D context to be opened. This function is the first interface that

accesses the hardware. The hardware will be turned on and initialized.

## Remarks

There is only one Nano2d context per application, so this function must be called once in your application.

## Function

n2d_error_t n2d_open()

### n2d_dither Function

Enable or disable dithering

## File

libnano2D.h

## C

```
n2d_error_t n2d_dither(n2d_bool_t enable);
```

## Returns

Returns the status as defined by n2d_error_t

## Description

Sets the capability to scatter or approximate colors when using less than 32bpp or 16bpp color depth. Dither attempts to improve the overall appearance of low resolution images. Dithering is on when enable is true, otherwise, dithering is off.

## Remarks

This function will wait until the hardware is complete, i.e. it is synchronous.

## Parameters

| Parameters | Description |
|---|---|
| enable | defines whether dither is set on or off. |

## Function

n2d_error_t n2d_dither()

### b) Data Types and Constants

### n2d_blend_t Enumeration

List of blending modes

## File

libnano2D.h

## C

```
typedef enum n2d_blend {
  N2D_BLEND_NONE,
  N2D_BLEND_SRC_OVER,
  N2D_BLEND_DST_OVER,
  N2D_BLEND_SRC_IN,
  N2D_BLEND_DST_IN,
  N2D_BLEND_ADDITIVE,
  N2D_BLEND_SUBTRACT
} n2d_blend_t;
```

## Description

Structure: n2d_blend

N2D_BLEND_NONE - S, i.e. no blending N2D_BLEND_SRC_OVER - S + (1 - Sa) * D N2D_BLEND_DST_OVER - (1 - Da) * S + D N2D_BLEND_SRC_IN - Da * S N2D_BLEND_DST_IN - Sa * D N2D_BLEND_ADDITIVE - S + D N2D_BLEND_SUBTRACT - D * (1 - S)

## Remarks

Some of the Nano2D API functions calls support blending. S and D represent source and destination color channels and Sa and Da represent the source and destination alpha channels

### n2d_buffer_format_t Enumeration

List of blending modes

## File

libnano2D.h

## C

```
typedef enum n2d_buffer_format {
  N2D_RGBA8888,
  N2D_BGRA8888,
  N2D_RGB565,
  N2D_BGR565,
  N2D_RGBA4444,
  N2D_BGRA4444,
  N2D_A8
} n2d_buffer_format_t;
```

## Members

| Members | Description |
|---|---|
| N2D_RGBA4444 | currently not available in MPLAB Harmony HAL |
| N2D_BGRA4444 | currently not available in MPLAB Harmony HAL |

## Description

Structure: n2d_buffer_format

N2D_RGBA8888 - 32-bit RGBA format with 8 bits per color channel. Red is in bits 7:0, green in bits 15:8, blue in bits 23:16, and the alpha channel is in bits 31:24 N2D_BGRA8888 - 32-bit RGBA format with 8 bits per color channel. Red is in bits 23:16, green in bits 15:8, blue in bits 7:0, and the alpha channel is in bits 31:24 N2D_RGB565 - 16-bit RGB format with 5 and 6 bits per color channel. Red is in bits 4:0, green in bits 10:5, and the blue color channel is in bits 15:11 N2D_BGR565 - 16-bit RGB format with 5 and 6 bits per color channel. Red is in bits 15:11, green in bits 10:5, and the blue color channel is in bits 4:0 N2D_RGBA4444 - 16-bit RGBA format with 4 bits per color channel. Red is in bits 3:0, green in bits 7:4, blue in bits 11:8 and the alpha channel is in bits 15:12. Note: currently not available in HAL N2D_BGRA4444 - 16-bit RGBA format with 4 bits per color channel. Red is in bits 11:8, green in bits 7:4, blue in bits 3:0 and the alpha channel is in bits 15:12. Note: currently not available in HAL N2D_A8 - 8-bit alpha format. There are no RGB values.

## Remarks

The pixel type for a n2d_buffer_t structure

### n2d_buffer_t Structure

A wrapper structure for any image or render target

## File

libnano2D.h

## C

```
typedef struct n2d_buffer {
  n2d_int32_t width;
  n2d_int32_t height;
  n2d_int32_t stride;
  n2d_buffer_format_t format;
  n2d_orientation_t orientation;
  void * handle;
  void * memory;
  n2d_uint32_t gpu;
} n2d_buffer_t;
```

## Description

Structure: n2d_buffer

width - Width of the buffer in pixels height - Height of the buffer in pixels stride - Stride of the buffer in bytes format - Pixel format of the buffer orientation - Buffer's orientation memory - Logical pointer to the buffer's memory for the CPU gpu - Physical address of the buffer's memory the hardware can access

## Remarks

Each piece of memory, whether it is an image used as a source or a buffer used as a destination, requires a structure to define it. This structure contains all the information the Nano2D API requires to access the buffer's memory by the hardware

### n2d_color_t Type

Identifies a specific pixel color

## File

libnano2D.h

## C

```
typedef n2d_int32_t n2d_color_t;
```

## Description

Color

Color container

### n2d_error_t Enumeration

Error codes that the Nano2D functions can return

## File

libnano2D.h

## C

```
typedef enum n2d_error {
  N2D_SUCCESS = 0,
  N2D_INVALID_ARGUMENT,
  N2D_OUT_OF_MEMORY,
  N2D_NO_CONTEXT,
  N2D_TIMEOUT,
  N2D_OUT_OF_RESOURCES,
  N2D_GENERIC_IO,
  N2D_NOT_SUPPORTED
} n2d_error_t;
```

## Description

Structure: n2d_error

N2D_SUCCESS - Success N2D_INVALID_ARGUMENT - An invalid argument was specified N2D_OUT_OF_MEMORY - Out of memory N2D_NO_CONTEXT - No open context is present N2D_TIMEOUT - A timeout has accored during a wait N2D_OUT_OF_RESOURCES - Out of system resources N2D_GENERIC_IO - Cannot communicate with the kernel driver N2D_NOT_SUPPORTED - The request is not supported

## Remarks

All API functions return a status code. On success, N2D_SUCCESS will be returned when a function is successful. This value is set to zero, so if any function returns a non-zero value, an error has occured

### n2d_module_parameters_t Structure

GPU peripheral Initialization parameters

## File

libnano2D.h

## C

```
typedef struct n2d_module_parameters {
  n2d_int32_t irqLine2D;
  n2d_uint32_t registerMemBase2D;
  n2d_uint32_t registerMemSize2D;
  n2d_uint32_t contiguousSize;
  n2d_uint32_t contiguousBase;
  n2d_uint32_t baseAddress;
} n2d_module_parameters_t;
```

## Description

Structure: n2d_module_parameters

irqLine2D - command completion interrupt pin registerMemBase2D - base address of gpu (physical address) registerMemsize2D - size of gpu address space contiguousSize - size of memory pool contiguousBase - start address of memory (virtual address) baseAddress - base address display buffer

## Remarks

None

### n2d_orientation_t Enumeration

List of blending modes

## File

libnano2D.h

## C

```
typedef enum n2d_orientation {
  N2D_0,
  N2D_90,
  N2D_180,
  N2D_270
} n2d_orientation_t;
```

## Description

Structure: n2d_orientation

N2D_0 - Buffer is 0 degrees rotated. N2D_90 - Buffer is 90 degrees rotated. N2D_180 - Buffer is 180 degrees rotated. N2D_270 - Buffer is 270 degrees rotated.

## Remarks

Orientation is orthogonal. Rotation which is not parallel to the x or y axis is not supported.

### n2d_point_t Structure

A position on a pixel

## File

libnano2D.h

## C

```
typedef struct n2d_point {
  n2d_int32_t x;
  n2d_int32_t y;
} n2d_point_t;
```

## Description

Structure: n2d_point

Defines a position on the screen

x - horizontal coordinate of the point y - vertical coordinate of the point

### n2d_rectangle_t Structure

A rectangle

## File

libnano2D.h

## C

```
typedef struct n2d_rectangle {
  n2d_int32_t x;
  n2d_int32_t y;
  n2d_int32_t width;
  n2d_int32_t height;
} n2d_rectangle_t;
```

## Description

Structure: n2d_rectangle

Defines a rectangular shape area of the screen

x - Left coordinate of the rectangle y - Top coordinate of the rectangle width - Width of the rectangle height - Height of the rectangle

### n2d_transparency_t Enumeration

Transparency modes

## File

libnano2D.h

## C

```
typedef enum n2d_transparency {
  N2D_TRANSPARENCY_NONE,
```

```
    N2D_TRANSPARENCY_SOURCE,
    N2D_TRANSPARENCY_DESTINATION
} n2d_transparency_t;
```

## Description

Structure: n2d_transparency

N2D_TRANSPARENCY_NONE - No transparency N2D_TRANSPARENCY_SOURCE - The source defines the transparency N2D_TRANSPARENCY_DESTINATION - The destination defines the transparency

## Remarks

The Nano2D hardware can be programmed to use transparency, extracted from either the source or the destination

### n2d_bool_t Type

## File

libnano2D_types.h

## C

```
typedef int n2d_bool_t;
```

## Description

This is type n2d_bool_t.

### n2d_float_t Type

## File

libnano2D_types.h

## C

```
typedef float n2d_float_t;
```

## Description

This is type n2d_float_t.

### n2d_int16_t Type

## File

libnano2D_types.h

## C

```
typedef short n2d_int16_t;
```

## Description

This is type n2d_int16_t.

### n2d_int32_t Type

## File

libnano2D_types.h

## C

```
typedef int n2d_int32_t;
```

## Description

This is type n2d_int32_t.

### n2d_size_t Type

## File

libnano2D_types.h

## C

```
typedef unsigned int n2d_size_t;
```

## Description

This is type n2d_size_t.

### n2d_uint16_t Type

#### File

libnano2D_types.h

#### C

```
typedef unsigned short n2d_uint16_t;
```

#### Description

This is type n2d_uint16_t.

### n2d_uint32_t Type

#### File

libnano2D_types.h

#### C

```
typedef unsigned int n2d_uint32_t;
```

#### Description

This is type n2d_uint32_t.

### n2d_uint64_t Type

#### File

libnano2D_types.h

#### C

```
typedef unsigned long long n2d_uint64_t;
```

#### Description

This is type n2d_uint64_t.

### n2d_uint8_t Type

#### File

libnano2D_types.h

#### C

```
typedef unsigned char n2d_uint8_t;
```

#### Description

This is type n2d_uint8_t.

### __gcmALIGN Macro

#### File

libnano2D_types.h

#### C

```
#define __gcmALIGN(data, reg_field) \
    (((n2d_uint32_t) (data)) << __gcmSTART(reg_field))
```

#### Description

This is macro __gcmALIGN.

### __gcmEND Macro

#### File

libnano2D_types.h

#### C

```
#define __gcmEND(reg_field) \
    (1 ? reg_field)
```

## Description

This is macro __gcmEND.

### __gcmGETSIZE Macro

## File

libnano2D_types.h

## C

```
#define __gcmGETSIZE(reg_field) \
    (__gcmEND(reg_field) - __gcmSTART(reg_field) + 1)
```

## Description

This is macro __gcmGETSIZE.

### __gcmMASK Macro

## File

libnano2D_types.h

## C

```
#define __gcmMASK(reg_field) \
    ((n2d_uint32_t) ((__gcmGETSIZE(reg_field) == 32) \
        ?  ~0 \
        : (~(~0 << __gcmGETSIZE(reg_field)))))
```

## Description

This is macro __gcmMASK.

### __gcmSTART Macro

## File

libnano2D_types.h

## C

```
#define __gcmSTART(reg_field) \
    (0 ? reg_field)
```

## Description

This is macro __gcmSTART.

### _nano2D_types_h__ Macro

## File

libnano2D_types.h

## C

```
#define _nano2D_types_h__
```

## Description

This is macro _nano2D_types_h__.

### gcmALIGN Macro

## File

libnano2D_types.h

## C

```
#define gcmALIGN(n, align) \
( \
    ((n) + ((align) - 1)) & ~((align) - 1) \
)
```

## Description

This is macro gcmALIGN.

### gcmCOUNTOF Macro

#### File

libnano2D_types.h

#### C

```
#define gcmCOUNTOF(array) \
        (sizeof(array) / sizeof(array[0]))
```

#### Description

This is macro gcmCOUNTOF.

### gcmGETFIELD Macro

#### File

libnano2D_types.h

#### C

```
#define gcmGETFIELD(data, reg, field) \
( \
    ((((n2d_uint32_t) (data)) >> __gcmSTART(reg##_##field)) \
        & __gcmMASK(reg##_##field)) \
)
```

#### Description

This is macro gcmGETFIELD.

### gcmINT2PTR Macro

#### File

libnano2D_types.h

#### C

```
#define gcmINT2PTR(i) \
( \
    (void *)(n2d_uint32_t)(i) \
)
```

#### Description

This is macro gcmINT2PTR.

### gcmMAX Macro

#### File

libnano2D_types.h

#### C

```
#define gcmMAX(x, y) \
( \
    ((x) >= (y)) \
        ? (x) \
        : (y) \
)
```

#### Description

This is macro gcmMAX.

### gcmMIN Macro

#### File

libnano2D_types.h

#### C

```
#define gcmMIN(x, y) \
( \
    ((x) <= (y)) \
        ? (x) \
```

```
        : (y) \
)
```

## Description

This is macro gcmMIN.

### gcmPTR2INT Macro

## File

libnano2D_types.h

## C

```
#define gcmPTR2INT(p) \
( \
    (n2d_uint32_t)(p) \
)
```

## Description

This is macro gcmPTR2INT.

### gcmSETFIELD Macro

## File

libnano2D_types.h

## C

```
#define gcmSETFIELD(data, reg, field, value) \
( \
    (((n2d_uint32_t) (data)) \
        & ~__gcmALIGN(__gcmMASK(reg##_##field), reg##_##field)) \
        |  __gcmALIGN((n2d_uint32_t) (value) \
            & __gcmMASK(reg##_##field), reg##_##field) \
)
```

## Description

This is macro gcmSETFIELD.

### gcmSETFIELDVALUE Macro

## File

libnano2D_types.h

## C

```
#define gcmSETFIELDVALUE(data, reg, field, value) \
( \
    (((n2d_uint32_t) (data)) \
        & ~__gcmALIGN(__gcmMASK(reg##_##field), reg##_##field)) \
        |  __gcmALIGN(reg##_##field##_##value \
            & __gcmMASK(reg##_##field), reg##_##field) \
)
```

## Description

This is macro gcmSETFIELDVALUE.

### gcmSETMASKEDFIELD Macro

## File

libnano2D_types.h

## C

```
#define gcmSETMASKEDFIELD(reg, field, value) \
( \
    gcmSETFIELD     (~0, reg,          field, value) & \
    gcmSETFIELDVALUE(~0, reg, MASK_ ## field, ENABLED) \
)
```

## Description

This is macro gcmSETMASKEDFIELD.

### gcmSETMASKEDFIELDVALUE Macro

#### File

libnano2D_types.h

#### C

```
#define gcmSETMASKEDFIELDVALUE(reg, field, value) \
( \
    gcmSETFIELDVALUE(~0, reg,          field, value) & \
    gcmSETFIELDVALUE(~0, reg, MASK_ ## field, ENABLED) \
)
```

#### Description

This is macro gcmSETMASKEDFIELDVALUE.

### gcmVERIFYFIELDVALUE Macro

#### File

libnano2D_types.h

#### C

```
#define gcmVERIFYFIELDVALUE(data, reg, field, value) \
( \
    (((n2d_uint32_t) (data)) >> __gcmSTART(reg##_##field) & \
                            __gcmMASK(reg##_##field)) \
        == \
    (reg##_##field##_##value & __gcmMASK(reg##_##field)) \
)
```

#### Description

This is macro gcmVERIFYFIELDVALUE.

### IN Macro

#### File

libnano2D_types.h

#### C

```
#define IN
```

#### Description

This is macro IN.

### N2D_FALSE Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_FALSE 0
```

#### Description

This is macro N2D_FALSE.

### N2D_INFINITE Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_INFINITE ((n2d_uint32_t) ~0U)
```

#### Description

This is macro N2D_INFINITE.

### N2D_IS_ERROR Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_IS_ERROR(error) (error != N2D_SUCCESS)
```

#### Description

This is macro N2D_IS_ERROR.

### N2D_IS_SUCCESS Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_IS_SUCCESS(error) (error == N2D_SUCCESS)
```

#### Description

This is macro N2D_IS_SUCCESS.

### N2D_NULL Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_NULL ((void *) 0)
```

#### Description

This is macro N2D_NULL.

### N2D_ON_ERROR Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_ON_ERROR(func) \
    do \
    { \
        error = func; \
        if (N2D_IS_ERROR(error)) \
        { \
            goto on_error; \
        } \
    } \
    while (0)
```

#### Description

This is macro N2D_ON_ERROR.

### N2D_TRUE Macro

#### File

libnano2D_types.h

#### C

```
#define N2D_TRUE 1
```

#### Description

This is macro N2D_TRUE.

### OUT Macro

## File

libnano2D_types.h

## C

```
#define OUT
```

## Description

This is macro OUT.

## Files

This section describes the Application Programming Interface (API) functions of the Nano2D module.

## Files

| Name | Description |
|---|---|
| libnano2D.h | Main header file for MPLAB Harmony Graphics Driver libnano2D GPU functions |
| libnano2D_types.h | Defines libnano2D data types and constants |

## Description

| Source File Name | Description |
|---|---|
| `libnano2D.h` | Header file that includes all of the nano2d APIs |
| `libnano2D_types.h` | Header file that includes all the nano2d data types |
| `libnano2D.a` | Static library file which must be linked to the application. This library implements all Nano2D feature APIs |

### libnano2D.h

Main header file for MPLAB Harmony Graphics Driver libnano2D GPU functions

## Enumerations

| | Name | Description |
|---|---|---|
| | n2d_blend | List of blending modes |
| | n2d_buffer_format | List of blending modes |
| | n2d_error | Error codes that the Nano2D functions can return |
| | n2d_orientation | List of blending modes |
| | n2d_transparency | Transparency modes |
| | n2d_blend_t | List of blending modes |
| | n2d_buffer_format_t | List of blending modes |
| | n2d_error_t | Error codes that the Nano2D functions can return |
| | n2d_orientation_t | List of blending modes |
| | n2d_transparency_t | Transparency modes |

## Functions

| | Name | Description |
|---|---|---|
| | n2d_blit | Copy a source buffer to the the destination buffer |
| | n2d_dither | Enable or disable dithering |
| | n2d_draw_state | Set the drawing state for any following Nano2D API draw call |
| | n2d_fill | Fill a (partial) buffer with a specified color |
| | n2d_init_hardware | Initializes the n2d driver and peripheral hardware |
| | n2d_line | Draw a line |
| | n2d_open | Open Nano2d context |

## Structures

| | Name | Description |
|---|---|---|
| | n2d_buffer | A wrapper structure for any image or render target |
| | n2d_module_parameters | GPU peripheral Initialization parameters |

| | | |
|---|---|---|
| | n2d_point | A position on a pixel |
| | n2d_rectangle | A rectangle |
| | n2d_buffer_t | A wrapper structure for any image or render target |
| | n2d_module_parameters_t | GPU peripheral Initialization parameters |
| | n2d_point_t | A position on a pixel |
| | n2d_rectangle_t | A rectangle |

## Types

| | Name | Description |
|---|---|---|
| | n2d_color_t | Identifies a specific pixel color |

## Description

Module for Microchip Graphics Library - Graphics Driver Layer

The API functions to be used for the Nano2D graphics accelerator.

## File Name

libnano2D.h

## Company

Microchip Technology Inc.

### libnano2D_types.h

Defines libnano2D data types and constants

## Macros

| | Name | Description |
|---|---|---|
| | __gcmALIGN | This is macro __gcmALIGN. |
| | __gcmEND | This is macro __gcmEND. |
| | __gcmGETSIZE | This is macro __gcmGETSIZE. |
| | __gcmMASK | This is macro __gcmMASK. |
| | __gcmSTART | This is macro __gcmSTART. |
| | _nano2D_types_h__ | This is macro _nano2D_types_h__. |
| | gcmALIGN | This is macro gcmALIGN. |
| | gcmCOUNTOF | This is macro gcmCOUNTOF. |
| | gcmGETFIELD | This is macro gcmGETFIELD. |
| | gcmINT2PTR | This is macro gcmINT2PTR. |
| | gcmMAX | This is macro gcmMAX. |
| | gcmMIN | This is macro gcmMIN. |
| | gcmPTR2INT | This is macro gcmPTR2INT. |
| | gcmSETFIELD | This is macro gcmSETFIELD. |
| | gcmSETFIELDVALUE | This is macro gcmSETFIELDVALUE. |
| | gcmSETMASKEDFIELD | This is macro gcmSETMASKEDFIELD. |
| | gcmSETMASKEDFIELDVALUE | This is macro gcmSETMASKEDFIELDVALUE. |
| | gcmVERIFYFIELDVALUE | This is macro gcmVERIFYFIELDVALUE. |
| | IN | This is macro IN. |
| | N2D_FALSE | This is macro N2D_FALSE. |
| | N2D_INFINITE | This is macro N2D_INFINITE. |
| | N2D_IS_ERROR | This is macro N2D_IS_ERROR. |
| | N2D_IS_SUCCESS | This is macro N2D_IS_SUCCESS. |
| | N2D_NULL | This is macro N2D_NULL. |
| | N2D_ON_ERROR | This is macro N2D_ON_ERROR. |
| | N2D_TRUE | This is macro N2D_TRUE. |
| | OUT | This is macro OUT. |

## Types

| | Name | Description |
|---|---|---|
| | n2d_bool_t | This is type n2d_bool_t. |

| | | |
|---|---|---|
| | n2d_float_t | This is type n2d_float_t. |
| | n2d_int16_t | This is type n2d_int16_t. |
| | n2d_int32_t | This is type n2d_int32_t. |
| | n2d_size_t | This is type n2d_size_t. |
| | n2d_uint16_t | This is type n2d_uint16_t. |
| | n2d_uint32_t | This is type n2d_uint32_t. |
| | n2d_uint64_t | This is type n2d_uint64_t. |
| | n2d_uint8_t | This is type n2d_uint8_t. |

## Description

Module for Microchip Graphics Library - Graphics Driver Layer

This is a definition file for libnano2d types and constants used in the library and applications.

## File Name

libnano2D_types.h

## Company

Microchip Technology Inc.

# *Graphics Utilities Library*

This section provides information about the Graphics Utilities Library within the MPLAB Harmony framework for graphics applications.

## Description

This library is primarily responsible for managing and decoding assets such as images, fonts, and strings. It provides the means for interacting with asset data, complex data decoding, data decompression, and string asset lookup. The library also abstractly handles the accessing of external memory sources during asset decoding.

## Definitions

- Anti-aliasing – Uses transparency to achieve a less jagged look when rasterizing pixel information.
- ASCII – The standard 8-bit-per-character text representation.
- Asset – A generic term for any resource that consists of blocks of binary data. Can be images, raw files, fonts, strings, and so on.
- Binary Asset – A chunk of raw data.
- Codepoint – The numerical ID of a glyph, typically four bytes in size
- External Asset – An asset that is stored on an external storage peripheral not directly accessible from the CPU.
- Font – A collection of images that represent linguistic characters. Each font has a distinct look and feel.
- Glyph – A linguistic character.
- Image Asset – A collection of pixel data that, when rendered, forms a visual image.
- Index Map – An image that is stored as a series of lookup table indices, rather than raw pixel data.
- Palette Asset – A form of image compression. Palettes are lookup tables of color information.
- Run-length Encoding (RLE) – A simple form of data compression that indexes long strings of duplicate bytes into a single value with an associated length value. Data blocks with long runs of duplicate characters are good candidates for RLE compression. Poor candidates will see the data size increase rather than decrease.
- String – A series of characters often used to form linguistic sentences.
- UTF8 – The encoding format for Unicode characters that favors space over decoding speed.
- UTF16 – The encoding format for Unicode characters that favors decoding speed over space.

## Graphics Utilities Library Objective

The library serves four main purposes:

- Provides a common definition for asset description.
- Provides APIs for asset indexing, decoding, and rendering.
- Provides an abstract API for asset decoders.
- Provides state machines for accessing assets located in external memory locations.

## Asset Common Definition

All assets share a common header “GFXU_AssetHeader”. This header is defined as follows:

```
typedef struct GFXU_AssetHeader_t
{
 uint32_t type;
 uint32_t dataLocation;
```

```
 void* dataAddress;
 uint32_t dataSize;
} GFXU_AssetHeader;
```

- type – the type of an asset
- dataLocation – The location ID for the asset. A location of “0” always indicates internal flash memory.
- dataAddress – The address of the asset. Depending on memory location, this address may not be accessible from the CPU.
- dataSize – The size of the asset in bytes.

## Image Decoding and Rendering

Image Asset Descriptor

The image asset descriptor is defined as follows:

```
typedef struct GFXU_ImageAsset_t
{
 GFXU_AssetHeader header;
 GFXU_ImageFormat format;
 uint32_t width;
 uint32_t height;
 GFX_ColorMode colorMode;
 GFXU_ImageCompressionType compType;
 GFX_Bool useMask;
 GFX_Color mask;
 GFXU_PaletteAsset* palette;
} GFXU_ImageAsset;
```

- header – The common asset header.
- format – The format of the image data.
- width – The width of the image.
- height – The height of the image.
- colorMode – The format of the image's pixel data.
- compType – If compressed, indicates the compression type.
- useMask – Indicates whether the image specifies a pixel transparency mask.
- mask – The value of the image transparency mask.
- palette – If the color mode is an index format, then this is the address of the lookup table to reference.

Image decoding with the GFX Utilities library is meant to be simple and straightforward. The library provides a single API that clients can use to render image assets.

```
GFX_Result GFXU_DrawImage(GFXU_ImageAsset* img,
 int32_t src_x,
 int32_t src_y,
 int32_t src_width,
 int32_t src_height,
 int32_t dest_x,
 int32_t dest_y,
 GFXU_MemoryIntf* read_cb,
 GFXU_ExternalAssetReader** reader);
```

This function accepts a pointer to an image header and rendering dimensions for rendering sub-sections of the image. The last two arguments refer to external memory access and will be described later. If the type of an image specifies an image decoder, then that decoder is automatically invoked by the library. If an image is stored as an index map, then its associated palette is referenced during decoding.

**Image Palette Asset**

Palette assets are color lookup tables that can be referenced when decoding indexed images. The index format can be 1bpp, 4bpp, or 8bpp large, resulting in a maximum of 1, 16, or 256 colors, respectively.

The palette descriptor is defined as follows:

```
typedef struct GFXU_PaletteAsset_t
{
 GFXU_AssetHeader header;
 uint32_t colorCount;
 GFX_ColorMode colorMode;
} GFXU_PaletteAsset;
```

- header – The common asset header.
- colorCount – The number of colors in this palette.
- colorMode – The color format of this palette.

**Font Assets**

Fonts are chunks of data that contain color information for drawing individual linguistic characters. Font glyph data can either be stored by using a 1bpp or 8bpp format. The larger format is for storing transparency information for use in drawing anti-aliased characters.

The font descriptor is as follows:

```
typedef struct GFXU_FontAsset_t
{
 GFXU_AssetHeader header;
 uint32_t height;
 uint32_t ascent;
 uint32_t descent;
 uint32_t baseline;
 GFXU_FontAssetBPP bpp;
 GFXU_FontGlyphIndexTable* indexTable;
} GFXU_FontAsset;
```

- header – The common asset header.
- height – The height of the font.
- ascent – The ascent of the font.
- descent – The descent of the font.
- baseline – The baseline of the font.
- bpp – The size of the per-pixel font data.
- indexTable – The pointer to the font index table.

**Font Index Table**

Each font provides an index table for quickly locating pixel data inside the font data chunk. The index table specifies the number of individual ranges or series of glyphs that it provides, followed by the actual range data.

A typical font range table entry is as follows:

```
typedef struct GFXU_FontGlyphRange_t
{
 uint32_t glyphCount;
 uint32_t startID;
 uint32_t endID;
 uint8_t* lookupTable;
} GFXU_FontGlyphRange;
```

- glyphCount – The number of glyphs in this range.
- startID – The starting glyph codepoint.
- endID – The ending glyph codepoint.
- lookupTable – The pointer to the lookup table for this range.

**Font Lookup Table**

The font lookup table contains location and size data for referencing glyphs in the font data chunk. This table provides with values to allow geometric analysis of font glyphs without having to render any data.

The data in a font lookup table is defined as follows:

- Byte 1 – The size of the offset values in the lookup table. 1-4 bytes possible depending on the max size of the font data chunk.
- Byte 2 – The size of the width values in the lookup table. This depends on the font size. The maximum size is 0xFFFF.

Repeating 'glyphCount' number of times:

- Offset value – Read offset value of 1-4 bytes, depending on the header.
- Width value – Read width value of 1-2 bytes, depending on the header

**Font Glyph Raster Data**

Font raster data is stored in a single chunk of memory and is referenced through the previously mentioned lookup tables. When rendering a font, the decoder uses the code point to find the appropriate lookup table and then uses that table to get the offset of the glyph. The width value says how large the glyph is in pixels, which could be 1bpp or 8bpp, depending on the anti-alias setting. The decoder then reads "width" number of pixels starting at "offset". The pixel data offsets are always byte-aligned.

**String Table**

The graphics utilities library defines a special asset called the "String Table". This table is a predefined lookup table of strings, languages, and their associated fonts. This construct makes runtime localization possible for user interface libraries.

## Graphics Utilities Interface

### a) Functions

| | Name | Description |
|---|---|---|
| | convertColorAndSetDraw | internal use only |
| | getDiscreteValueAtIndex | internal use only |
| | getOffsetFromIndexAndBPP | internal use only |
| | getRLEDataAtIndex | internal use only |
| | GFXU_CalculateCharStringWidth | Gets the width of a string buffer in pixels. |

| | | |
|---|---|---|
| | GFXU_CalculatePartialCharStringWidth | Gets the width of a partial string buffer in pixels. |
| | GFXU_CalculatePartialStringWidth | Gets the partial width, starting from the left, of a string contained in the string table. |
| | GFXU_CalculateStringWidth | Gets the width of a string contained in the string table. |
| | GFXU_CompareString | Compares a string table entry and a GFXU_CHAR type string. |
| | GFXU_DecodeCodePoint | internal use only |
| | GFXU_DecodeUTF16 | internal use only |
| | GFXU_DecodeUTF8 | internal use only |
| | GFXU_DrawCharString | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawImage | Draws a portion of the given image at the specified coordinates. |
| | GFXU_DrawString | Draws a string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_ExtractString | Extracts a string from the string table into a local character buffer. The local buffer must have already been allocated. Will attempt to write 'size - 1' number of characters into the buffer with the last being a zero terminator. |
| | GFXU_FontGetGlyphInfo | internal use only |
| | GFXU_FontGetLookupTableEntry | internal use only |
| | GFXU_GetCharAt | Gets a character of a string contained in the string table. |
| | GFXU_GetCharWidth | Gets the width of a character for a given font. |
| | GFXU_GetStringHeight | Gets the height of a string contained in the string table. |
| | GFXU_GetStringLength | Gets the length of a string contained in a string table. |
| | GFXU_GetStringRect | Gets the bounding rectangle for a string contained in the string table. |
| | GFXU_GetStringSizeInBytes | Gets the size of a string contained in a string table, in bytes. |
| | GFXU_PaletteGetColor | Gets a color from a palette asset given an index value. |
| | GFXU_StringFontIndexLookup | internal use only |
| | GFXU_StringIndexLookup | internal use only |
| | GFXU_StringLookup | internal use only |
| | GFXU_DecodeAndDrawString | internal use only |
| | GFXU_DrawCharStringClipped | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawGlyph | internal use only |
| | GFXU_DrawGlyphRow | internal use only |
| | GFXU_DrawStringClipped | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawUnknownGlyph | internal use only |
| | GFXU_GetGlyphRowDataSize | internal use only |
| | GFXU_PreprocessImage | Preprocesses an image to a specified memory address. |
| | GFXU_DecodeAndDrawSubString | internal use only |
| | GFXU_DrawCharSubStringClipped | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawSubStringClipped | Draws a sub-string from a string asset in a clipped rectangle at the given coordinates. Strings are drawn from the top down. |
| | GFXU_GetCharStringLineRect | Gets the bounding rectangle for a line in a character string. |
| | GFXU_GetStringLineRect | Gets the bounding rectangle for a line in a string asset. |

## b) Data Types and Constants

| | Name | Description |
|---|---|---|
| | GFXU_AssetHeader_t | Defines a common header for all assets supported by the generic decoder interface. |
| | GFXU_AssetType_t | Enumerates known asset types. |
| | GFXU_BinaryAsset_t | A binary asset type. Generic data that can be of any type |
| | GFXU_ExternalAssetReader_t | Defines the base description of an external asset reader. Specific decoder implementations will build on this foundation for each decoder type. |
| | GFXU_ExternalAssetReaderStatus_t | Enumerates external reader state machine states.<br>Invalid - state machine hasn't been initialized Read - state machine is ready to begin processing but hasn't been run yet Waiting - state machine is waiting for a memory operation to complete Drawing - state machine is waiting for a drawing operation to complete Finished - state machine has finished drawing the entire asset Aborted - state machine encountered an error and has aborted |

| | | |
|---|---|---|
| | GFXU_FontAsset_t | Describes a font asset. A font asset is a series of raster images that represet linguistic characters. These characters are referenced by an index called a 'code point'. This code point is 1-4 bytes in length. Code points may be encoded to save space. Fonts also contain kerning data that describes character positioning data.<br>header - standard asset header height - font height in pixels ascent - font ascent in pixels descent - font descent in pixels baseline - font baseline in pixels bpp - font pixel size, either 1 or 8. 8 is for anti-aliased font data indexTable -... more |
| | GFXU_FontAssetBPP | Indicates the bits per pixel mode of a font |
| | GFXU_FontGlyphIndexTable_t | Describes a font glyph index table. Essentially a series of glyph look up tables where each non-consecutive range of glyphs occupies one range entry<br>count - number of ranges in the index table ranges - the glyph range array |
| | GFXU_FontGlyphRange_t | Describes a range of glyphs for a font. All IDs are in raw code points and are not encoded in any way<br>glyphCount - number of glyphs in the range startID - the starting glyph id endID - the ending glyph id lookupTable - the corresponding look up table to find the glyph raster data |
| | GFXU_ImageAsset_t | Describes an image asset.<br>header - standard asset header format - the format of the image. this directly affects which decoder is invoked when rendering the image width - the width of the image in pixels height - the height of the image in pixels colorMode - the color mode of the image compType - the compression mode of the image useMask - indicates of the mask field is used mask - may contain a masking color for the image. blit operations may reference this value and reject image pixels that match this value palette - will contain a valid... more |
| | GFXU_ImageCompressionType_t | Indicates the image compression type, only applies to RAW types |
| | GFXU_ImageFormat_t | Indicates the image encoding format |
| | GFXU_MemoryIntf_t | Defines a memory interface for all memory operations. Essentially wraps a GFX_MemoryIntf with the notable addition lf a GFXU_MemoryReadRequest_FnPtr.<br>The 'read' function pointer will be invoked when a decoder attempts to access an asset that has a location that is greater than zero. The application must then look at the asset's location id and determine which peripheral to communicate with. |
| | GFXU_PaletteAsset_t | Describes a palette asset. A palette is a lookup table for unique colors. Given in an index, a color can be retrieved.<br>header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image |
| | GFXU_StringEncodingMode_t | Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16 |
| | GFXU_StringTableAsset_t | Describes a string table asset. There is typically only ever one of these defined at any one time.<br>header - standard asset header languageCount - the number of languages in the string table stringCount - the number of strings in the string table stringIndexTable - the pointer to the string index table. the string index table is a table that contains all of the unique strings defined in the string table. fontTable - the font table contains an array of pointers to all defined font assets that the string table references fontIndexTable - the font index table is a table... more |
| | GFXU_AssetHeader | Defines a common header for all assets supported by the generic decoder interface. |
| | GFXU_AssetType | Enumerates known asset types. |
| | GFXU_BinaryAsset | A binary asset type. Generic data that can be of any type |
| | GFXU_CHAR | strings are defined as having 32bit characters due to the need to support international code points and unicode strings encoded using these types will not be compatible with standard library string functions like strlen or strcat |
| | GFXU_ExternalAssetReader | Defines the base description of an external asset reader. Specific decoder implementations will build on this foundation for each decoder type. |
| | GFXU_ExternalAssetReaderRun_FnPtr | This function pointer represents a function that maintains the state of an external reader state machine.<br>The argument is the state machine to process. |
| | GFXU_ExternalAssetReaderStatus | Enumerates external reader state machine states.<br>Invalid - state machine hasn't been initialized Read - state machine is ready to begin processing but hasn't been run yet Waiting - state machine is waiting for a memory operation to complete Drawing - state machine is waiting for a drawing operation to complete Finished - state machine has finished drawing the entire asset Aborted - state machine encountered an error and has aborted |

| | | |
|---|---|---|
| | GFXU_FontAsset | Describes a font asset. A font asset is a series of raster images that represet linguistic characters. These characters are referenced by an index called a 'code point'. This code point is 1-4 bytes in length. Code points may be encoded to save space. Fonts also contain kerning data that describes character positioning data.<br>header - standard asset header height - font height in pixels ascent - font ascent in pixels descent - font descent in pixels baseline - font baseline in pixels bpp - font pixel size, either 1 or 8. 8 is for anti-aliased font data indexTable -... more |
| | GFXU_FontGlyphIndexTable | Describes a font glyph index table. Essentially a series of glyph look up tables where each non-consecutive range of glyphs occupies one range entry<br>count - number of ranges in the index table ranges - the glyph range array |
| | GFXU_FontGlyphRange | Describes a range of glyphs for a font. All IDs are in raw code points and are not encoded in any way<br>glyphCount - number of glyphs in the range startID - the starting glyph id endID - the ending glyph id lookupTable - the corresponding look up table to find the glyph raster data |
| | GFXU_ImageAsset | Describes an image asset.<br>header - standard asset header format - the format of the image. this directly affects which decoder is invoked when rendering the image width - the width of the image in pixels height - the height of the image in pixels colorMode - the color mode of the image compType - the compression mode of the image useMask - indicates of the mask field is used mask - may contain a masking color for the image. blit operations may reference this value and reject image pixels that match this value palette - will contain a valid... more |
| | GFXU_ImageCompressionType | Indicates the image compression type, only applies to RAW types |
| | GFXU_ImageFormat | Indicates the image encoding format |
| | GFXU_MemoryIntf | Defines a memory interface for all memory operations. Essentially wraps a GFX_MemoryIntf with the notable addition lf a GFXU_MemoryReadRequest_FnPtr.<br>The 'read' function pointer will be invoked when a decoder attempts to access an asset that has a location that is greater than zero. The application must then look at the asset's location id and determine which peripheral to communicate with. |
| | GFXU_PaletteAsset | Describes a palette asset. A palette is a lookup table for unique colors. Given in an index, a color can be retrieved.<br>header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image |
| | GFXU_StringEncodingMode | Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16 |
| | GFXU_ASSET_LOCATION_INTERNAL | This is macro GFXU_ASSET_LOCATION_INTERNAL. |
| | GFXU_STRING_ARRAY_SIZE | defines meta data sizes for the string table, don't change! |
| | GFXU_STRING_ENTRY_SIZE | This is macro GFXU_STRING_ENTRY_SIZE. |
| | GFXU_STRING_MAX_CHAR_WIDTH | This is macro GFXU_STRING_MAX_CHAR_WIDTH. |
| | GFXU_MediaCloseRequest_FnPtr | A callback that indicates that a media decoder is finished with a given media location and that the application can close if it.<br>The argument is the asset that was being read. |
| | GFXU_MediaOpenRequest_FnPtr | A callback that indicates that a media decoder wishes to read from an external media source and that the application should prepare that source.<br>The argument is the asset that needs to be read.<br>If the result is false then the decoder will abort. |
| | GFXU_MediaReadRequest_FnPtr | callback |
| | GFXU_MediaReadRequestCallback_FnPtr | A callback that signifies that a media read request has been fulfilled. Often signals a state machine to continue processing a decode operation.<br>The argument is the reader that requested the memory. |

## Description

This section describes the interface for the Graphics Utilities Library.

### *a) Functions*

**convertColorAndSetDraw Function**

## File

gfxu_image_utils.h

### C

```
GFX_Result convertColorAndSetDraw(uint32_t color, GFX_ColorMode mode);
```

### Description

internal use only

## getDiscreteValueAtIndex Function

### File

gfxu_image_utils.h

### C

```
uint32_t getDiscreteValueAtIndex(uint32_t index, uint32_t value, GFX_ColorMode mode);
```

### Description

internal use only

## getOffsetFromIndexAndBPP Function

### File

gfxu_image_utils.h

### C

```
uint32_t getOffsetFromIndexAndBPP(uint32_t index, GFX_BitsPerPixel bpp);
```

### Description

internal use only

## getRLEDataAtIndex Function

### File

gfxu_image_utils.h

### C

```
uint32_t getRLEDataAtIndex(uint8_t* data, uint32_t max, uint32_t idx, uint32_t* startBlock, uint32_t*
startOffset);
```

### Description

internal use only

## GFXU_CalculateCharStringWidth Function

Gets the width of a string buffer in pixels.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_CalculateCharStringWidth(GFXU_CHAR* str, GFXU_FontAsset* fnt);
```

### Returns

uint32_t - the width of the string buffer

### Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR* str | the string buffer, terimated with a zero value |
| GFXU_FontAsset* fnt | the font asset to reference |

### Function

uint32_t GFXU_CalculateCharStringWidth( GFXU_CHAR* str,
GFXU_FontAsset* fnt)

## GFXU_CalculatePartialCharStringWidth Function

Gets the width of a partial string buffer in pixels.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_CalculatePartialCharStringWidth(GFXU_CHAR* str, GFXU_FontAsset* fnt, uint32_t
length);
```

### Returns

uint32_t - the width of the partial string buffer

### Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR* str | the string buffer, terimated with a zero value |
| GFXU_FontAsset* fnt | the font asset to reference |
| uint32_t length | the number of characters to include |

### Function

uint32_t GFXU_CalculatePartialCharStringWidth( GFXU_CHAR* str,
GFXU_FontAsset* fnt,
uint32_t length)

## GFXU_CalculatePartialStringWidth Function

Gets the partial width, starting from the left, of a string contained in the string table.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_CalculatePartialStringWidth(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t
lang, uint32_t length);
```

### Returns

uint32_t - the width of the sub-string

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| uint32_t length | the number of characters to include in the sub-string |

### Function

uint32_t GFXU_CalculatePartialStringWidth( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
uint32_t length)

## GFXU_CalculateStringWidth Function

Gets the width of a string contained in the string table.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_CalculateStringWidth(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang);
```

### Returns

uint32_t - the width of the string

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |

### Function

uint32_t GFXU_CalculateStringWidth( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang)

## GFXU_CompareString Function

Compares a string table entry and a GFXU_CHAR type string.

### File

gfxu_string.h

### C

```
LIB_EXPORT int32_t GFXU_CompareString(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang, GFXU_CHAR*
buffer);
```

### Returns

int32_t - the compare result, should be identical to strcmp()

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| GFXU_CHAR* buffer | char buffer to compare |

### Function

int32_t GFXU_CompareString( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
GFXU_CHAR* buffer)

## GFXU_DecodeCodePoint Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_DecodeCodePoint(uint32_t encoding, uint8_t* data, uint32_t max, uint32_t* codePoint,
uint32_t* offset);
```

### Description

internal use only

## GFXU_DecodeUTF16 Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_DecodeUTF16(uint8_t* val, uint32_t max, uint32_t* codePoint, uint32_t* size);
```

### Description

internal use only

## GFXU_DecodeUTF8 Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_DecodeUTF8(uint8_t* val, uint32_t max, uint32_t* codePoint, uint32_t* size);
```

### Description

internal use only

## GFXU_DrawCharString Function

Draws a clipped string at the given coordinates. Strings are drawn from the top down.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawCharString(GFXU_CHAR* str, GFXU_FontAsset* fnt, int32_t x, int32_t y,
GFXU_MemoryIntf* memoryInterface, GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR* str | the string buffer to draw |
| int32_t x | the x position to which to draw the string |
| int32_t y | the y position to which to draw the string |
| GFXU_MemoryIntf* memoryInterface | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

```
GFX_Result GFXU_DrawCharString(GFXU_CHAR* str,
                               GFXU_FontAsset* fnt
int32_t x,
int32_t y,
                               GFXU_MemoryIntf* memoryInterface,
                               GFXU_ExternalAssetReader** reader)
```

## GFXU_DrawImage Function

Draws a portion of the given image at the specified coordinates.

### File

gfxu_image.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawImage(GFXU_ImageAsset* img, int32_t src_x, int32_t src_y, int32_t src_width,
int32_t src_height, int32_t dest_x, int32_t dest_y, GFXU_MemoryIntf* read_cb, GFXU_ExternalAssetReader**
reader);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_ImageAsset* img | pointer to the image asset to draw |
| int32_t src_x | the x position of the source image to draw (0 if whole image) |
| int32_t src_y | the y position of the source image to draw (0 if whole image) |
| int32_t src_width | the width of the source rectangle to draw (source width if whole image) the height of the source rectangle to draw (source height if whole image) |
| int32_t dest_x | the x position to draw to |
| int32_t dest_y | the y position to draw to |
| GFXU_MemoryIntf* read_cb | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

GFX_Result GFXU_DrawImage(void);

## GFXU_DrawString Function

Draws a string at the given coordinates. Strings are drawn from the top down.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawString(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang, int32_t x,
int32_t y, GFXU_MemoryIntf* memoryInterface, GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| int32_t x | the x position to which to draw the string |
| int32_t y | the y position to which to draw the string |
| GFXU_MemoryIntf* memoryInterface | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

GFX_Result GFXU_DrawString(GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
int32_t x,
int32_t y,

GFXU_MemoryIntf* memoryInterface,
GFXU_ExternalAssetReader** reader)

## GFXU_ExtractString Function

Extracts a string from the string table into a local character buffer. The local buffer must have already been allocated. Will attempt to write 'size - 1' number of characters into the buffer with the last being a zero terminator.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_ExtractString(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang, GFXU_CHAR*
buffer, uint32_t size, uint32_t offset);
```

### Returns

uint32_t - the number of characters written

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to reference |
| uint32_t lang | the id of the language to reference |
| GFXU_CHAR* buffer | the pointer to the buffer to write to |
| uint32_t size | the maximum size of the buffer to write to |
| uint32_t offset | the buffer write offset if any |

### Function

uint32_t GFXU_ExtractString( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
GFXU_CHAR* buffer,
uint32_t size,
uint32_t offset)

## GFXU_FontGetGlyphInfo Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_FontGetGlyphInfo(GFXU_FontAsset* fnt, uint32_t glyph, uint32_t* offset, uint32_t* width);
```

### Description

internal use only

## GFXU_FontGetLookupTableEntry Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_FontGetLookupTableEntry(uint8_t* table, uint32_t index, uint32_t max, uint32_t* offset,
uint32_t* width);
```

### Description

internal use only

## GFXU_GetCharAt Function

Gets a character of a string contained in the string table.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFXU_CHAR GFXU_GetCharAt(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang, uint32_t idx);
```

### Returns

GFXU_CHAR - the code point of the character

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| uint32_t idx | the index of the characer |

### Function

GFXU_CHAR GFXU_GetCharAt(GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
uint32_t idx)

## GFXU_GetCharWidth Function

Gets the width of a character for a given font.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_GetCharWidth(GFXU_CHAR chr, GFXU_FontAsset* fnt);
```

### Returns

uint32_t - the width of the character or zero if the font doesn't contain that character

### Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR chr | the code point of the character |
| GFXU_FontAsset* fnt | the font to reference |

### Function

uint32_t GFXU_GetCharWidth( GFXU_CHAR chr, GFXU_FontAsset* fnt)

## GFXU_GetStringHeight Function

Gets the height of a string contained in the string table.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_GetStringHeight(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang);
```

### Returns

uint32_t - the height of the string

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |

### Function

uint32_t GFXU_GetStringHeight( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang)

## GFXU_GetStringLength Function

Gets the length of a string contained in a string table.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_GetStringLength(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang);
```

### Returns

uint32_t - the length of the string entry

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |

### Function

uint32_t GFXU_GetStringLength( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang)

## GFXU_GetStringRect Function

Gets the bounding rectangle for a string contained in the string table.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_GetStringRect(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang, GFX_Rect*
rect);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| GFX_Rect* rect | the resultant rectangle of the string |

### Function

GFX_Result GFXU_GetStringRect(GFXU_StringTableAsset* tbl,
uint32_t id,

uint32_t lang,
GFX_Rect* rect)

## GFXU_GetStringSizeInBytes Function

Gets the size of a string contained in a string table, in bytes.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_GetStringSizeInBytes(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang);
```

### Returns

uint32_t - the size of the string entry in bytes

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |

### Function

uint32_t GFXU_GetStringSizeInBytes( GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang)

## GFXU_PaletteGetColor Function

Gets a color from a palette asset given an index value.

### File

gfxu_palette.h

### C

```
GFX_Color GFXU_PaletteGetColor(GFXU_PaletteAsset* pal, uint32_t idx, GFXU_MemoryIntf* read_cb,
GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Color - the color that was retrieved

### Parameters

| Parameters | Description |
|---|---|
| GFXU_PaletteAsset* pal | pointer to the palette to read |
| uint32_t idx | the index of the color to look up |
| GFXU_MemoryIntf* read_cb | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the palette asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

GFX_Result GFXU_PaletteGetColor(void);

## GFXU_StringFontIndexLookup Function

### File

gfxu_string_utils.h

### C

```
LIB_EXPORT GFXU_FontAsset* GFXU_StringFontIndexLookup(GFXU_StringTableAsset* table, uint32_t stringID,
uint32_t languageID);
```

### Description

internal use only

## GFXU_StringIndexLookup Function

### File

gfxu_string_utils.h

### C

```
uint16_t GFXU_StringIndexLookup(GFXU_StringTableAsset* table, uint32_t stringID, uint32_t languageID);
```

### Description

internal use only

## GFXU_StringLookup Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_StringLookup(GFXU_StringTableAsset* table, uint32_t stringIndex, uint8_t** stringAddress,
uint32_t* stringSize);
```

### Description

internal use only

## GFXU_DecodeAndDrawString Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_DecodeAndDrawString(uint8_t* string, uint32_t length, GFXU_StringEncodingMode mode,
GFXU_FontAsset* fnt, int32_t clipX, int32_t clipY, int32_t clipWidth, int32_t clipHeight, int32_t x,
int32_t y);
```

### Description

internal use only

## GFXU_DrawCharStringClipped Function

Draws a clipped string at the given coordinates. Strings are drawn from the top down.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawCharStringClipped(GFXU_CHAR* str, GFXU_FontAsset* fnt, int32_t clipX,
int32_t clipY, int32_t clipWidth, int32_t clipHeight, int32_t x, int32_t y, GFXU_MemoryIntf*
memoryInterface, GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR* str | the string buffer to draw |
| int32_t clipX | clipped x position |
| int32_t clipY | clipped y position |

| | |
|---|---|
| int32_t clipWidth | clipped rectangle width |
| int32_t clipHeight | clipped rectangle height |
| int32_t x | the x position to which to draw the string |
| int32_t y | the y position to which to draw the string |
| GFXU_MemoryIntf* memoryInterface | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

GFX_Result GFXU_DrawCharStringClipped(GFXU_CHAR* str,
GFXU_FontAsset* fnt
int32_t clipX,
int32_t clipY,
int32_t clipWidth,
int32_t clipHeight,
int32_t x,
int32_t y,
GFXU_MemoryIntf* memoryInterface,
GFXU_ExternalAssetReader** reader)

## GFXU_DrawGlyph Function

### File

gfxu_string_utils.h

### C

```
int32_t GFXU_DrawGlyph(GFXU_FontAsset* fnt, uint32_t glyph, int32_t clipX, int32_t clipY, int32_t
clipWidth, int32_t clipHeight, int32_t x, int32_t y);
```

### Description

internal use only

## GFXU_DrawGlyphRow Function

### File

gfxu_string_utils.h

### C

```
void GFXU_DrawGlyphRow(GFXU_FontAssetBPP bpp, uint8_t* data, int32_t width, int32_t x, int32_t y, int32_t
clipXStart, int32_t clipXEnd);
```

### Description

internal use only

## GFXU_DrawStringClipped Function

Draws a clipped string at the given coordinates. Strings are drawn from the top down.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawStringClipped(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang,
int32_t clipX, int32_t clipY, int32_t clipWidth, int32_t clipHeight, int32_t x, int32_t y, GFXU_MemoryIntf*
memoryInterface, GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Result

## Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| int32_t clipX | clipped x position |
| int32_t clipY | clipped y position |
| int32_t clipWidth | clipped rectangle width |
| int32_t clipHeight | clipped rectangle height |
| int32_t x | the x position to which to draw the string |
| int32_t y | the y position to which to draw the string |
| GFXU_MemoryIntf* memoryInterface | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

## Function

GFX_Result GFXU_DrawStringClipped(GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
int32_t clipX,
int32_t clipY,
int32_t clipWidth,
int32_t clipHeight,
int32_t x,
int32_t y,
GFXU_MemoryIntf* memoryInterface,
GFXU_ExternalAssetReader** reader)

### GFXU_DrawUnknownGlyph Function

## File

gfxu_string_utils.h

## C

```
int32_t GFXU_DrawUnknownGlyph(int32_t x, int32_t y, int32_t height, int32_t clipX, int32_t clipY, int32_t clipWidth, int32_t clipHeight);
```

## Description

internal use only

### GFXU_GetGlyphRowDataSize Function

## File

gfxu_string_utils.h

## C

```
int32_t GFXU_GetGlyphRowDataSize(GFXU_FontAssetBPP bpp, int32_t width);
```

## Description

internal use only

### GFXU_PreprocessImage Function

Preprocesses an image to a specified memory address.

### File

gfxu_image.h

### C

```
LIB_EXPORT GFX_Result GFXU_PreprocessImage(GFXU_ImageAsset* img, uint32_t destAddress, GFX_ColorMode
destMode, GFX_Bool padBuffer);
```

### Returns

GFX_Result - the result of the operation

### Description

This function preprocesses an image asset through the HAL pipeline and renders it to a given address, in a given color mode, and can pad the image buffer dimensions to be powers of 2 as required by some graphics accelerators.

This function is also useful for pre-staging images into run-time memory locations.

The caller is required to ensure that the destination address is capable of containing the result. The size can be calculated by using the method:

GFX_ColorInfo[destMode].size * img->width * img->height

This function only works with images that are located in a core accessible memory location like SRAM or DDR. If the image is located in an external source then GFXU_DrawImage should be called directly. The caller will then need to service the media streaing state machine. Once finished the image asset descriptor must be changed manually. This function can be used as a reference on how to accomplish this.

### Parameters

| Parameters | Description |
|---|---|
| GFXU_ImageAsset* img | pointer to the image asset to draw |
| uint32_t destAddress | the address to render the image to |
| GFX_ColorMode destMode | the desired output mode of the image |
| GFX_Bool padBuffer | indicates that the image buffer dimensions should be padded to equal powers of 2 (required by some GPUs) |
| GFXU_MemoryIntf* read_cb | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

GFX_Result GFXU_PreprocessImage(GFXU_ImageAsset* img,
uint32_t destAddress,
GFX_ColorMode destMode,
GFX_Bool padBuffer);

## GFXU_DecodeAndDrawSubString Function

### File

gfxu_string_utils.h

### C

```
GFX_Result GFXU_DecodeAndDrawSubString(uint8_t* string, uint32_t length, GFXU_StringEncodingMode mode,
GFXU_FontAsset* fnt, uint32_t start, uint32_t end, int32_t clipX, int32_t clipY, int32_t clipWidth, int32_t
clipHeight, int32_t x, int32_t y);
```

### Description

internal use only

## GFXU_DrawCharSubStringClipped Function

Draws a clipped string at the given coordinates. Strings are drawn from the top down.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawCharSubStringClipped(GFXU_CHAR* str, GFXU_FontAsset* fnt, uint32_t start,
```

```
uint32_t end, int32_t clipX, int32_t clipY, int32_t clipWidth, int32_t clipHeight, int32_t x, int32_t y,
GFXU_MemoryIntf* memoryInterface, GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR* str | the string buffer to draw |
| GFXU_FontAsset* fnt | font asset to use |
| uint32_t start | start offset of substring |
| uint32_t end | end offset of substring |
| int32_t clipX | clipped x position |
| int32_t clipY | clipped y position |
| int32_t clipWidth | clipped rectangle width |
| int32_t clipHeight | clipped rectangle height |
| int32_t x | the x position to which to draw the string |
| int32_t y | the y position to which to draw the string |
| GFXU_MemoryIntf* memoryInterface | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

### Function

GFX_Result GFXU_DrawCharSubStringClipped(GFXU_CHAR* str,
GFXU_FontAsset* fnt
uint32_t start,
uint32_t end,
int32_t clipX,
int32_t clipY,
int32_t clipWidth,
int32_t clipHeight,
int32_t x,
int32_t y,
GFXU_MemoryIntf* memoryInterface,
GFXU_ExternalAssetReader** reader)

## GFXU_DrawSubStringClipped Function

Draws a sub-string from a string asset in a clipped rectangle at the given coordinates. Strings are drawn from the top down.

### File

gfxu_string.h

### C

```
LIB_EXPORT GFX_Result GFXU_DrawSubStringClipped(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang,
uint32_t start, uint32_t end, int32_t clipX, int32_t clipY, int32_t clipWidth, int32_t clipHeight, int32_t
x, int32_t y, GFXU_MemoryIntf* memoryInterface, GFXU_ExternalAssetReader** reader);
```

### Returns

GFX_Result

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| uint32_t start | offset of first character of substring |
| uint32_t end | offset of last character of substring |

| | |
|---|---|
| int32_t clipX | clipped x position |
| int32_t clipY | clipped y position |
| int32_t clipWidth | clipped rectangle width |
| int32_t clipHeight | clipped rectangle height |
| int32_t x | the x position to which to draw the string |
| int32_t y | the y position to which to draw the string |
| GFXU_MemoryIntf* memoryInterface | a pointer to a memory interface to use for memory operations, not needed for internal assets |
| GFXU_ExternalAssetReader** reader | will return as a valid pointer if the image asset is located in an external source. If this is the case then the caller is responsible for servicing the external asset reader state machine until completion. The caller is then responsible for freeing the reader's memory. |

## Function

GFX_Result GFXU_DrawSubStringClipped(GFXU_StringTableAsset* tbl,
uint32_t id,
uint32_t lang,
uint32_t start,
uint32_t end,
int32_t clipX,
int32_t clipY,
int32_t clipWidth,
int32_t clipHeight,
int32_t x,
int32_t y,
GFXU_MemoryIntf* memoryInterface,
GFXU_ExternalAssetReader** reader)

### GFXU_GetCharStringLineRect Function

Gets the bounding rectangle for a line in a character string.

## File

gfxu_string.h

## C

```
LIB_EXPORT uint32_t GFXU_GetCharStringLineRect(GFXU_CHAR* str, GFXU_FontAsset* font, uint32_t offset,
GFX_Rect* rect);
```

## Returns

The offset of end of line (including line feed or end of string)

## Parameters

| Parameters | Description |
|---|---|
| GFXU_CHAR* str | the string buffer, terimated with a zero value |
| GFXU_FontAsset* fnt | the font asset to reference |
| uint32_t offset | the start offset of the first character in the line |
| uint32_t length | the number of characters to include |

## Function

uint32_t GFXU_GetCharStringLineRect( GFXU_CHAR* str,
GFXU_FontAsset* font,
uint32_t offset,
GFX_Rect* rect)

### GFXU_GetStringLineRect Function

Gets the bounding rectangle for a line in a string asset.

### File

gfxu_string.h

### C

```
LIB_EXPORT uint32_t GFXU_GetStringLineRect(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang, uint32_t
offset, GFX_Rect* rect);
```

### Returns

The offset of end of line (including line feed or end of string)

### Parameters

| Parameters | Description |
|---|---|
| GFXU_StringTableAsset* tbl | the string table to reference |
| uint32_t id | the id of the string to look up |
| uint32_t lang | the language entry to look up |
| uint32_t offset | the start offset of the first character in the line |
| GFX_Rect* rect | the resultant rectangle of the string |

### Function

uint32_t GFXU_GetStringLineRect( GFXU_StringTableAsset* tbl,

uint32_t id,

uint32_t lang,

uint32_t offset,

GFX_Rect* rect)

## *b) Data Types and Constants*

### GFXU_FontAsset_t Structure

Describes a font asset. A font asset is a series of raster images that represet linguistic characters. These characters are referenced by an index called a 'code point'. This code point is 1-4 bytes in length. Code points may be encoded to save space. Fonts also contain kerning data that describes character positioning data.

header - standard asset header height - font height in pixels ascent - font ascent in pixels descent - font descent in pixels baseline - font baseline in pixels bpp - font pixel size, either 1 or 8. 8 is for anti-aliased font data indexTable - pointer to the corresponding glyph index table. this table is used to reference code points to glyph data.

### File

gfxu_font.h

### C

```
struct GFXU_FontAsset_t {
  GFXU_AssetHeader header;
  uint32_t height;
  uint32_t ascent;
  uint32_t descent;
  uint32_t baseline;
  GFXU_FontAssetBPP bpp;
  GFXU_FontGlyphIndexTable* indexTable;
};
```

### Description

Structure: GFXU_FontAsset_t

### GFXU_FontAssetBPP Enumeration

Indicates the bits per pixel mode of a font

### File

gfxu_font.h

### C

```
enum GFXU_FontAssetBPP {
  GFXU_FONT_BPP_1,
  GFXU_FONT_BPP_8
};
```

### Members

| Members | Description |
|---|---|
| GFXU_FONT_BPP_1 | standard |
| GFXU_FONT_BPP_8 | antialiased |

### Description

Enumeration: GFXU_FontAssetBPP

## GFXU_AssetHeader Structure

Defines a common header for all assets supported by the generic decoder interface.

### File

gfxu_global.h

### C

```
typedef struct GFXU_AssetHeader_t {
  uint32_t type;
  uint32_t dataLocation;
  void* dataAddress;
  uint32_t dataSize;
} GFXU_AssetHeader;
```

### Description

Structure: GFXU_AssetHeader_t

type - GFXU_AssetType - indicates the type of the asset dataLocation - indicates the location of the asset data. 0 is always internal flash. any other number must be understood by the application dataAddress - the address at which the data resides. may be a local pointer or a location in some external storage location not in the local memory map. dataSize - the size of the asset data in bytes

## GFXU_AssetType Enumeration

Enumerates known asset types.

### File

gfxu_global.h

### C

```
typedef enum GFXU_AssetType_t {
  GFXU_ASSET_TYPE_IMAGE = 0,
  GFXU_ASSET_TYPE_PALETTE,
  GFXU_ASSET_TYPE_FONT,
  GFXU_ASSET_TYPE_BINARY,
  GFXU_ASSET_TYPE_STRINGTABLE
} GFXU_AssetType;
```

### Description

enum: GFXU_AssetType_t

## GFXU_BinaryAsset Structure

A binary asset type. Generic data that can be of any type

### File

gfxu_binary.h

### C

```
typedef struct GFXU_BinaryAsset_t {
```

```
  GFXU_AssetHeader header;
} GFXU_BinaryAsset;
```

### Members

| Members | Description |
|---|---|
| GFXU_AssetHeader header; | generic asset header |

### Description

Structure: GFXU_BinaryAsset_t

## GFXU_CHAR Type

### File

gfxu_string.h

### C

```
typedef uint32_t GFXU_CHAR;
```

### Description

strings are defined as having 32bit characters due to the need to support international code points and unicode strings encoded using these types will not be compatible with standard library string functions like strlen or strcat

## GFXU_ExternalAssetReader Structure

Defines the base description of an external asset reader. Specific decoder implementations will build on this foundation for each decoder type.

### File

gfxu_global.h

### C

```
typedef struct GFXU_ExternalAssetReader_t {
  GFXU_ExternalAssetReaderStatus status;
  uint32_t state;
  int32_t result;
  GFXU_MemoryIntf* memIntf;
  GFXU_ExternalAssetReaderRun_FnPtr run;
  void* userData;
} GFXU_ExternalAssetReader;
```

### Description

Structure: GFXU_MemoryIntf_t

status - the overall status of the decoder state machine state - mostly for decoder internal use result - can be used for an overall result of the state of an operation memIntf - the memory interface the decoder is using for memory operations run - the run pointer that must be called periodically to allow the state machine to process to completion.

## GFXU_ExternalAssetReaderRun_FnPtr Type

This function pointer represents a function that maintains the state of an external reader state machine.

The argument is the state machine to process.

### File

gfxu_global.h

### C

```
typedef GFX_Result (* GFXU_ExternalAssetReaderRun_FnPtr)(GFXU_ExternalAssetReader*);
```

### Description

typedef: GFXU_ExternalAssetReaderRun_FnPtr

## GFXU_ExternalAssetReaderStatus Enumeration

Enumerates external reader state machine states.

Invalid - state machine hasn't been initialized Read - state machine is ready to begin processing but hasn't been run yet Waiting - state machine is waiting for a memory operation to complete Drawing - state machine is waiting for a drawing operation to complete Finished - state machine has finished drawing the entire asset Aborted - state machine encountered an error and has aborted

### File

gfxu_global.h

### C

```
typedef enum GFXU_ExternalAssetReaderStatus_t {
  GFXU_READER_STATUS_INVALID = 0,
  GFXU_READER_STATUS_READY,
  GFXU_READER_STATUS_WAITING,
  GFXU_READER_STATUS_DRAWING,
  GFXU_READER_STATUS_FINISHED,
  GFXU_READER_STATUS_ABORTED
} GFXU_ExternalAssetReaderStatus;
```

### Description

enum: GFXU_ExternalAssetReaderStatus_t

## GFXU_FontAsset Type

Describes a font asset. A font asset is a series of raster images that represet linguistic characters. These characters are referenced by an index called a 'code point'. This code point is 1-4 bytes in length. Code points may be encoded to save space. Fonts also contain kerning data that describes character positioning data.

header - standard asset header height - font height in pixels ascent - font ascent in pixels descent - font descent in pixels baseline - font baseline in pixels bpp - font pixel size, either 1 or 8. 8 is for anti-aliased font data indexTable - pointer to the corresponding glyph index table. this table is used to reference code points to glyph data.

### File

gfxu_string.h

### C

```
typedef struct GFXU_FontAsset_t GFXU_FontAsset;
```

### Description

Structure: GFXU_FontAsset_t

## GFXU_FontGlyphIndexTable Structure

Describes a font glyph index table. Essentially a series of glyph look up tables where each non-consecutive range of glyphs occupies one range entry

count - number of ranges in the index table ranges - the glyph range array

### File

gfxu_font.h

### C

```
typedef struct GFXU_FontGlyphIndexTable_t {
  uint32_t count;
  GFXU_FontGlyphRange ranges[];
} GFXU_FontGlyphIndexTable;
```

### Description

Structure: GFXU_FontGlyphIndexTable_t

## GFXU_FontGlyphRange Structure

Describes a range of glyphs for a font. All IDs are in raw code points and are not encoded in any way

glyphCount - number of glyphs in the range startID - the starting glyph id endID - the ending glyph id lookupTable - the corresponding look up table to find the glyph raster data

### File

gfxu_font.h

### C

```
typedef struct GFXU_FontGlyphRange_t {
  uint32_t glyphCount;
  uint32_t startID;
  uint32_t endID;
  uint8_t* lookupTable;
} GFXU_FontGlyphRange;
```

### Description

Structure: GFXU_FontGlyphRange_t

## GFXU_ImageAsset Structure

Describes an image asset.

header - standard asset header format - the format of the image. this directly affects which decoder is invoked when rendering the image width - the width of the image in pixels height - the height of the image in pixels colorMode - the color mode of the image compType - the compression mode of the image useMask - indicates of the mask field is used mask - may contain a masking color for the image. blit operations may reference this value and reject image pixels that match this value palette - will contain a valid pointer to a palette asset if thie image is an index map instead of a color image

### File

gfxu_image.h

### C

```
typedef struct GFXU_ImageAsset_t {
  GFXU_AssetHeader header;
  GFXU_ImageFormat format;
  uint32_t width;
  uint32_t height;
  uint32_t bufferWidth;
  uint32_t bufferHeight;
  GFX_ColorMode colorMode;
  GFXU_ImageCompressionType compType;
  GFXU_ImageFlags flags;
  GFX_Color mask;
  GFXU_PaletteAsset* palette;
} GFXU_ImageAsset;
```

### Description

Structure: GFXU_ImageAsset_t

## GFXU_ImageCompressionType Enumeration

Indicates the image compression type, only applies to RAW types

### File

gfxu_image.h

### C

```
typedef enum GFXU_ImageCompressionType_t {
  GFXU_IMAGE_COMPRESSION_NONE = 0,
  GFXU_IMAGE_COMPRESSION_RLE
} GFXU_ImageCompressionType;
```

### Description

Enumeration: GFXU_ImageCompressionType_t

## GFXU_ImageFormat Enumeration

Indicates the image encoding format

### File

gfxu_image.h

### C

```
typedef enum GFXU_ImageFormat_t {
  GFXU_IMAGE_FORMAT_RAW = 0,
  GFXU_IMAGE_FORMAT_JPEG,
  GFXU_IMAGE_FORMAT_PNG
} GFXU_ImageFormat;
```

### Description

Enumeration: GFXU_ImageFormat_t

## GFXU_MemoryIntf Structure

Defines a memory interface for all memory operations. Essentially wraps a GFX_MemoryIntf with the notable addition If a GFXU_MemoryReadRequest_FnPtr.

The 'read' function pointer will be invoked when a decoder attempts to access an asset that has a location that is greater than zero. The application must then look at the asset's location id and determine which peripheral to communicate with.

### File

gfxu_global.h

### C

```
typedef struct GFXU_MemoryIntf_t {
  GFX_MemoryIntf heap;
  GFXU_MediaOpenRequest_FnPtr open;
  GFXU_MediaReadRequest_FnPtr read;
  GFXU_MediaCloseRequest_FnPtr close;
} GFXU_MemoryIntf;
```

### Description

Structure: GFXU_MemoryIntf_t

heap - function pointer for memory operations read - function pointer to use for memory read requests

## GFXU_PaletteAsset Structure

Describes a palette asset. A palette is a lookup table for unique colors. Given in an index, a color can be retrieved.

header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image

### File

gfxu_palette.h

### C

```
typedef struct GFXU_PaletteAsset_t {
  GFXU_AssetHeader header;
  uint32_t colorCount;
  GFX_ColorMode colorMode;
} GFXU_PaletteAsset;
```

### Description

Structure: GFXU_PaletteAsset_t

## GFXU_StringEncodingMode Enumeration

Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16

### File

gfxu_string.h

### C

```
typedef enum GFXU_StringEncodingMode_t {
  GFXU_STRING_ENCODING_ASCII,
  GFXU_STRING_ENCODING_UTF8,
  GFXU_STRING_ENCODING_UTF16
} GFXU_StringEncodingMode;
```

### Description

Enumeration: GFXU_StringEncodingMode_t

## GFXU_ASSET_LOCATION_INTERNAL Macro

### File

gfxu_global.h

### C

```
#define GFXU_ASSET_LOCATION_INTERNAL 0
```

### Description

This is macro GFXU_ASSET_LOCATION_INTERNAL.

## GFXU_STRING_ARRAY_SIZE Macro

### File

gfxu_string.h

### C

```
#define GFXU_STRING_ARRAY_SIZE 4
```

### Description

defines meta data sizes for the string table, don't change!

## GFXU_STRING_ENTRY_SIZE Macro

### File

gfxu_string.h

### C

```
#define GFXU_STRING_ENTRY_SIZE 2
```

### Description

This is macro GFXU_STRING_ENTRY_SIZE.

## GFXU_STRING_MAX_CHAR_WIDTH Macro

### File

gfxu_string.h

### C

```
#define GFXU_STRING_MAX_CHAR_WIDTH 6
```

### Description

This is macro GFXU_STRING_MAX_CHAR_WIDTH.

## GFXU_MediaCloseRequest_FnPtr Type

A callback that indicates that a media decoder is finished with a given media location and that the application can close if it.
The argument is the asset that was being read.

### File

gfxu_global.h

### C

```
typedef void (* GFXU_MediaCloseRequest_FnPtr)(GFXU_AssetHeader* ast);
```

## Description

typedef: GFXU_MediaCloseRequest_FnPtr

### GFXU_MediaOpenRequest_FnPtr Type

A callback that indicates that a media decoder wishes to read from an external media source and that the application should prepare that source.

The argument is the asset that needs to be read.

If the result is false then the decoder will abort.

## File

gfxu_global.h

## C

```
typedef GFX_Result (* GFXU_MediaOpenRequest_FnPtr)(GFXU_AssetHeader* ast);
```

## Description

typedef: GFXU_MediaOpenRequest_FnPtr

### GFXU_MediaReadRequest_FnPtr Type

## File

gfxu_global.h

## C

```
typedef GFX_Result (* GFXU_MediaReadRequest_FnPtr)(GFXU_ExternalAssetReader* reader, GFXU_AssetHeader*
asset, void*, uint32_t, uint8_t*, GFXU_MediaReadRequestCallback_FnPtr);
```

## Description

callback

### GFXU_MediaReadRequestCallback_FnPtr Type

A callback that signifies that a media read request has been fulfilled. Often signals a state machine to continue processing a decode operation.

The argument is the reader that requested the memory.

## File

gfxu_global.h

## C

```
typedef void (* GFXU_MediaReadRequestCallback_FnPtr)(GFXU_ExternalAssetReader*);
```

## Description

typedef: GFXU_MediaReadRequestCallback_FnPtr

### *Files*

## Files

| Name | Description |
|---|---|
| gfxu_binary.h | Defines binary asset type. |
| gfxu_font.h | Describes font assets |
| gfxu_global.h | Global defines for graphics utility library. |
| gfxu_image.h | Defines image assets |
| gfxu_image_utils.h | Image return utilities |
| gfxu_palette.h | Defines palette assets |
| gfxu_string.h | Defines string table struct, string / character functions |
| gfxu_string_utils.h | Contains definitions for various internal string utility functions. |

## Description

### gfxu_binary.h

Defines binary asset type.

## Structures

| | Name | Description |
|---|---|---|
| | GFXU_BinaryAsset_t | A binary asset type. Generic data that can be of any type |
| | GFXU_BinaryAsset | A binary asset type. Generic data that can be of any type |

## Description

Module for Microchip Graphics Library - Graphics Utilities Library

Type definitions.

## File Name

gfxu_binary.h

## Company

Microchip Technology Inc.

### gfxu_font.h

Describes font assets

## Enumerations

| | Name | Description |
|---|---|---|
| | GFXU_FontAssetBPP | Indicates the bits per pixel mode of a font |

## Structures

| | Name | Description |
|---|---|---|
| | GFXU_FontAsset_t | Describes a font asset. A font asset is a series of raster images that represet linguistic characters. These characters are referenced by an index called a 'code point'. This code point is 1-4 bytes in length. Code points may be encoded to save space. Fonts also contain kerning data that describes character positioning data.<br>header - standard asset header height - font height in pixels ascent - font ascent in pixels descent - font descent in pixels baseline - font baseline in pixels bpp - font pixel size, either 1 or 8. 8 is for anti-aliased font data indexTable -... more |
| | GFXU_FontGlyphIndexTable_t | Describes a font glyph index table. Essentially a series of glyph look up tables where each non-consecutive range of glyphs occupies one range entry<br>count - number of ranges in the index table ranges - the glyph range array |
| | GFXU_FontGlyphRange_t | Describes a range of glyphs for a font. All IDs are in raw code points and are not encoded in any way<br>glyphCount - number of glyphs in the range startID - the starting glyph id endID - the ending glyph id lookupTable - the corresponding look up table to find the glyph raster data |
| | GFXU_FontGlyphIndexTable | Describes a font glyph index table. Essentially a series of glyph look up tables where each non-consecutive range of glyphs occupies one range entry<br>count - number of ranges in the index table ranges - the glyph range array |
| | GFXU_FontGlyphRange | Describes a range of glyphs for a font. All IDs are in raw code points and are not encoded in any way<br>glyphCount - number of glyphs in the range startID - the starting glyph id endID - the ending glyph id lookupTable - the corresponding look up table to find the glyph raster data |

## Description

Module for Microchip Graphics Library - Graphics Utilities Library

Type definitions.

## File Name

gfx_font.h

## Company

Microchip Technology Inc.

## gfxu_global.h

Global defines for graphics utility library.

## Enumerations

| | Name | Description |
|---|---|---|
| | GFXU_AssetType_t | Enumerates known asset types. |
| | GFXU_ExternalAssetReaderStatus_t | Enumerates external reader state machine states.<br>Invalid - state machine hasn't been initialized Read - state machine is ready to begin processing but hasn't been run yet Waiting - state machine is waiting for a memory operation to complete Drawing - state machine is waiting for a drawing operation to complete Finished - state machine has finished drawing the entire asset Aborted - state machine encountered an error and has aborted |
| | GFXU_AssetType | Enumerates known asset types. |
| | GFXU_ExternalAssetReaderStatus | Enumerates external reader state machine states.<br>Invalid - state machine hasn't been initialized Read - state machine is ready to begin processing but hasn't been run yet Waiting - state machine is waiting for a memory operation to complete Drawing - state machine is waiting for a drawing operation to complete Finished - state machine has finished drawing the entire asset Aborted - state machine encountered an error and has aborted |

## Macros

| | Name | Description |
|---|---|---|
| | GFXU_ASSET_LOCATION_INTERNAL | This is macro GFXU_ASSET_LOCATION_INTERNAL. |

## Structures

| | Name | Description |
|---|---|---|
| | GFXU_AssetHeader_t | Defines a common header for all assets supported by the generic decoder interface. |
| | GFXU_ExternalAssetReader_t | Defines the base description of an external asset reader. Specific decoder implementations will build on this foundation for each decoder type. |
| | GFXU_MemoryIntf_t | Defines a memory interface for all memory operations. Essentially wraps a GFX_MemoryIntf with the notable addition If a GFXU_MemoryReadRequest_FnPtr.<br>The 'read' function pointer will be invoked when a decoder attempts to access an asset that has a location that is greater than zero. The application must then look at the asset's location id and determine which peripheral to communicate with. |
| | GFXU_AssetHeader | Defines a common header for all assets supported by the generic decoder interface. |
| | GFXU_ExternalAssetReader | Defines the base description of an external asset reader. Specific decoder implementations will build on this foundation for each decoder type. |
| | GFXU_MemoryIntf | Defines a memory interface for all memory operations. Essentially wraps a GFX_MemoryIntf with the notable addition If a GFXU_MemoryReadRequest_FnPtr.<br>The 'read' function pointer will be invoked when a decoder attempts to access an asset that has a location that is greater than zero. The application must then look at the asset's location id and determine which peripheral to communicate with. |

## Types

| | Name | Description |
|---|---|---|
| | GFXU_ExternalAssetReaderRun_FnPtr | This function pointer represents a function that maintains the state of an external reader state machine.<br>The argument is the state machine to process. |
| | GFXU_MediaCloseRequest_FnPtr | A callback that indicates that a media decoder is finished with a given media location and that the application can close if it.<br>The argument is the asset that was being read. |
| | GFXU_MediaOpenRequest_FnPtr | A callback that indicates that a media decoder wishes to read from an external media source and that the application should prepare that source.<br>The argument is the asset that needs to be read.<br>If the result is false then the decoder will abort. |
| | GFXU_MediaReadRequest_FnPtr | callback |

| | | |
|---|---|---|
| | GFXU_MediaReadRequestCallback_FnPtr | A callback that signifies that a media read request has been fulfilled. Often signals a state machine to continue processing a decode operation.<br>The argument is the reader that requested the memory. |

## Description

Module for Microchip Graphics Library - Graphics Utilities Library

Type definitions.

## File Name

gfxu_global.h

## Company

Microchip Technology Inc.

# gfxu_image.h

Defines image assets

## Enumerations

| | Name | Description |
|---|---|---|
| | GFXU_ImageCompressionType_t | Indicates the image compression type, only applies to RAW types |
| | GFXU_ImageFlags_t | A list of flags describing an image asset |
| | GFXU_ImageFormat_t | Indicates the image encoding format |
| | GFXU_ImageCompressionType | Indicates the image compression type, only applies to RAW types |
| | GFXU_ImageFlags | A list of flags describing an image asset |
| | GFXU_ImageFormat | Indicates the image encoding format |

## Functions

| | Name | Description |
|---|---|---|
| | GFXU_DrawImage | Draws a portion of the given image at the specified coordinates. |
| | GFXU_PreprocessImage | Preprocesses an image to a specified memory address. |

## Structures

| | Name | Description |
|---|---|---|
| | GFXU_ImageAsset_t | Describes an image asset.<br>header - standard asset header format - the format of the image. this directly affects which decoder is invoked when rendering the image width - the width of the image in pixels height - the height of the image in pixels colorMode - the color mode of the image compType - the compression mode of the image useMask - indicates of the mask field is used mask - may contain a masking color for the image. blit operations may reference this value and reject image pixels that match this value palette - will contain a valid... more |
| | GFXU_ImageAsset | Describes an image asset.<br>header - standard asset header format - the format of the image. this directly affects which decoder is invoked when rendering the image width - the width of the image in pixels height - the height of the image in pixels colorMode - the color mode of the image compType - the compression mode of the image useMask - indicates of the mask field is used mask - may contain a masking color for the image. blit operations may reference this value and reject image pixels that match this value palette - will contain a valid... more |

## Description

Module for Microchip Graphics Library - Graphics Utilities Library

Image drawing at specified coordinates

## File Name

gfx_image.h

## Company

Microchip Technology Inc.

## gfxu_image_utils.h

Image return utilities

### Functions

| | Name | Description |
|---|---|---|
| | convertColorAndSetDraw | internal use only |
| | createDefaultMemIntf | internal use only |
| | getDiscreteValueAtIndex | internal use only |
| | getOffsetFromIndexAndBPP | internal use only |
| | getRLEDataAtIndex | internal use only |

### Description

Module for Microchip Graphics Library - Graphics Utilities Library

Internal library use only

### File Name

gfx_image_utils.h

### Company

Microchip Technology Inc.

## gfxu_palette.h

Defines palette assets

### Functions

| | Name | Description |
|---|---|---|
| | GFXU_PaletteGetColor | Gets a color from a palette asset given an index value. |

### Structures

| | Name | Description |
|---|---|---|
| | GFXU_PaletteAsset_t | Describes a palette asset. A palette is a lookup table for unique colors. Given in an index, a color can be retrieved.<br>header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image |
| | GFXU_PaletteAsset | Describes a palette asset. A palette is a lookup table for unique colors. Given in an index, a color can be retrieved.<br>header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image |

### Description

Module for Microchip Graphics Library - Graphics Utilities Library

Get palette color

### File Name

gfx_palette.h

### Company

Microchip Technology Inc.

## gfxu_string.h

Defines string table struct, string / character functions

### Enumerations

| | Name | Description |
|---|---|---|
| | GFXU_StringEncodingMode_t | Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16 |
| | GFXU_StringEncodingMode | Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16 |

## Functions

| | Name | Description |
|---|---|---|
| | GFXU_CalculateCharStringWidth | Gets the width of a string buffer in pixels. |
| | GFXU_CalculatePartialCharStringWidth | Gets the width of a partial string buffer in pixels. |
| | GFXU_CalculatePartialStringWidth | Gets the partial width, starting from the left, of a string contained in the string table. |
| | GFXU_CalculateStringWidth | Gets the width of a string contained in the string table. |
| | GFXU_CompareString | Compares a string table entry and a GFXU_CHAR type string. |
| | GFXU_DrawCharString | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawCharStringClipped | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawCharSubStringClipped | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawString | Draws a string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawStringClipped | Draws a clipped string at the given coordinates. Strings are drawn from the top down. |
| | GFXU_DrawSubStringClipped | Draws a sub-string from a string asset in a clipped rectangle at the given coordinates. Strings are drawn from the top down. |
| | GFXU_ExtractString | Extracts a string from the string table into a local character buffer. The local buffer must have already been allocated. Will attempt to write 'size - 1' number of characters into the buffer with the last being a zero terminator. |
| | GFXU_GetCharAt | Gets a character of a string contained in the string table. |
| | GFXU_GetCharStringLineRect | Gets the bounding rectangle for a line in a character string. |
| | GFXU_GetCharWidth | Gets the width of a character for a given font. |
| | GFXU_GetStringHeight | Gets the height of a string contained in the string table. |
| | GFXU_GetStringLength | Gets the length of a string contained in a string table. |
| | GFXU_GetStringLineRect | Gets the bounding rectangle for a line in a string asset. |
| | GFXU_GetStringRect | Gets the bounding rectangle for a string contained in the string table. |
| | GFXU_GetStringSizeInBytes | Gets the size of a string contained in a string table, in bytes. |

## Macros

| | Name | Description |
|---|---|---|
| | GFXU_STRING_ARRAY_SIZE | defines meta data sizes for the string table, don't change! |
| | GFXU_STRING_ENTRY_SIZE | This is macro GFXU_STRING_ENTRY_SIZE. |
| | GFXU_STRING_MAX_CHAR_WIDTH | This is macro GFXU_STRING_MAX_CHAR_WIDTH. |

## Structures

| | Name | Description |
|---|---|---|
| | GFXU_StringTableAsset_t | Describes a string table asset. There is typically only ever one of these defined at any one time.<br>header - standard asset header languageCount - the number of languages in the string table stringCount - the number of strings in the string table stringIndexTable - the pointer to the string index table. the string index table is a table that contains all of the unique strings defined in the string table. fontTable - the font table contains an array of pointers to all defined font assets that the string table references fontIndexTable - the font index table is a table... more |
| | GFXU_StringTableAsset | Describes a string table asset. There is typically only ever one of these defined at any one time.<br>header - standard asset header languageCount - the number of languages in the string table stringCount - the number of strings in the string table stringIndexTable - the pointer to the string index table. the string index table is a table that contains all of the unique strings defined in the string table. fontTable - the font table contains an array of pointers to all defined font assets that the string table references fontIndexTable - the font index table is a table... more |

## Types

| | Name | Description |
|---|---|---|
| | GFXU_CHAR | strings are defined as having 32bit characters due to the need to support international code points and unicode strings encoded using these types will not be compatible with standard library string functions like strlen or strcat |

| | | |
|---|---|---|
| | GFXU_FontAsset | Describes a font asset. A font asset is a series of raster images that represet linguistic characters. These characters are referenced by an index called a 'code point'. This code point is 1-4 bytes in length. Code points may be encoded to save space. Fonts also contain kerning data that describes character positioning data.<br>header - standard asset header height - font height in pixels ascent - font ascent in pixels descent - font descent in pixels baseline - font baseline in pixels bpp - font pixel size, either 1 or 8. 8 is for anti-aliased font data indexTable -... more |

## Description

Module for Microchip Graphics Library - Graphics Utilities Library

String statistics and drawing utilities

## File Name

gfx_string.h

## Company

Microchip Technology Inc.

### gfxu_string_utils.h

Contains definitions for various internal string utility functions.

## Functions

| | Name | Description |
|---|---|---|
| | GFXU_DecodeAndDrawString | internal use only |
| | GFXU_DecodeAndDrawSubString | internal use only |
| | GFXU_DecodeCodePoint | internal use only |
| | GFXU_DecodeUTF16 | internal use only |
| | GFXU_DecodeUTF8 | internal use only |
| | GFXU_DrawGlyph | internal use only |
| | GFXU_DrawGlyphRow | internal use only |
| | GFXU_DrawUnknownGlyph | internal use only |
| | GFXU_FontGetGlyphInfo | internal use only |
| | GFXU_FontGetLookupTableEntry | internal use only |
| | GFXU_GetGlyphRowDataSize | internal use only |
| | GFXU_StringFontIndexLookup | internal use only |
| | GFXU_StringIndexLookup | internal use only |
| | GFXU_StringLookup | internal use only |

## Description

Module for Microchip Graphics Library - Graphics Utilities Library

Internal use only

## File Name

gfx_string_utils.h

## Company

Microchip Technology Inc.

# SEGGER emWin Graphics Library

This topic is a reference for the third-party SEGGER Microcontroller GmbH & Co. KG emWin software graphics library.

## Description

In addition to the standard Graphics Library, the MPLAB Harmony Integrated Software Framework also offers a third-party graphics library, emWin, from SEGGER Microcontroller GmbH & Co. KG. emWin is a software graphics library that provides an efficient, processor and LCD controller-independent Graphical User Interface (GUI) for applications that operate with a graphical LCD. This library package, which includes the binary library, headers, and utility tools, is free to use as part of development using 32-bit and 16-bit products from Microchip.

The choice of using the standard MPLAB Harmony Graphics Library and/or the SEGGER emWin Graphics Library can be made during application development through the use of the MPLAB Harmony Configurator (MHC) in MPLAB Harmony.

For details about using the emWin graphics library, refer to SEGGER emWin Graphics Library.

A detailed description of the architecture and capabilities of the SEGGER emWin Graphics Library is available on the official SEGGER website at: https://www.segger.com/emwin.html https://www.segger.com/emwin.html

# Index


_

__gcmALIGN macro 418
__gcmEND macro 418
__gcmGETSIZE macro 419
__gcmMASK macro 419
__gcmSTART macro 419
_nano2D_types_h__ macro 419

**A**

Abstraction Model 391, 399
AGBA_8888_ALPHA_MASK macro 365
AGBA_8888_BLUE_MASK macro 365
AGBA_8888_GREEN_MASK macro 366
AGBA_8888_RED_MASK macro 366
Aria HAL Driver Examples 391
Aria User Interface Library 6
Aria User Interface Library Interface 12

**B**

begin_FnPtr type 338
blendColor_FnPtr type 374
blendGetPoint_FnPtr type 374
boolGet_FnPtr type 338
boolSet_FnPtr type 338
brightnessGet_FnPtr type 338
brightnessRangeGet_FnPtr type 339
brightnessSet_FnPtr type 339
Building the Library 408
BUTTON_LAST enumeration member 243
BUTTON_LEFT enumeration member 243
BUTTON_MIDDLE enumeration member 243
BUTTON_NONE enumeration member 243
BUTTON_RIGHT enumeration member 243
BUTTON_WHEEL_DOWN enumeration member 243
BUTTON_WHEEL_UP enumeration member 243

**C**

colorModeGet_FnPtr type 339
colorModeSet_FnPtr type 339
Configuring the Library 407
convertColorAndSetDraw function 431
createDefaultMemIntf function 323

**D**

DEFAULT_BORDER_MARGIN macro 261
destroy_FnPtr type 339
drawAlphaValueGet_FnPtr type 340
drawAlphaValueSet_FnPtr type 340
drawBlendModeGet_FnPtr type 340
drawBlendModeSet_FnPtr type 340
drawBlit_FnPtr type 340
drawCircle_FnPtr type 341
drawClipRectGet_FnPtr type 341
drawClipRectSet_FnPtr type 341
drawColorGet_FnPtr type 341
drawColorSet_FnPtr type 341
drawGradientColorGet_FnPtr type 342
drawGradientColorSet_FnPtr type 342
drawLine_FnPtr type 342
drawLock_FnPtr type 342
drawMaskValueGet_FnPtr type 342
drawMaskValueSet_FnPtr type 343
drawModeGet_FnPtr type 343
drawModeSet_FnPtr type 343
drawPaletteGet_FnPtr type 343
drawPaletteSet_FnPtr type 343
drawPipelineModeGet_FnPtr type 374
drawPipelineModeSet_FnPtr type 375
drawPixel_FnPtr type 344
drawRect_FnPtr type 344
drawResizeModeGet_FnPtr type 375
drawResizeModeSet_FnPtr type 375
drawStretchBlit_FnPtr type 375
drawTargetGet_FnPtr type 375
drawTargetSet_FnPtr type 376
drawThicknessGet_FnPtr type 344
drawThicknessSet_FnPtr type 344
drawUnlock_FnPtr type 344

**E**

end_FnPtr type 358

**F**

Files 265, 380, 424, 454

**G**

gcmALIGN macro 419
gcmCOUNTOF macro 420
gcmGETFIELD macro 420
gcmINT2PTR macro 420
gcmMAX macro 420
gcmMIN macro 420
gcmPTR2INT macro 421
gcmSETFIELD macro 421
gcmSETFIELDVALUE macro 421
gcmSETMASKEDFIELD macro 421
gcmSETMASKEDFIELDVALUE macro 422
gcmVERIFYFIELDVALUE macro 422
getDiscreteValueAtIndex function 432
getOffsetFromIndexAndBPP function 432
getRLEDataAtIndex function 432
GFX_AbsoluteValue function 305
GFX_ActiveContext function 306
GFX_ANTIALIAS_MODE_COUNT macro 366
GFX_ANTIALIAS_OFF enumeration member 344
GFX_ANTIALIAS_ON enumeration member 344
GFX_AntialiasMode enumeration 344
GFX_AntialiasMode_t enumeration 344
GFX_ASSERT macro 379
GFX_BitsPerPixel enumeration 345
GFX_BitsPerPixel_t enumeration 345
GFX_BLEND_ALL enumeration member 345
GFX_BLEND_CHANNEL enumeration member 345
GFX_BLEND_GLOBAL enumeration member 345
GFX_BLEND_NONE enumeration member 345
GFX_BlendMode enumeration 345

GFX_BlendMode_t enumeration 345
GFX_Bool type 345
GFX_BPP1 enumeration member 345
GFX_BPP16 enumeration member 345
GFX_BPP24 enumeration member 345
GFX_BPP32 enumeration member 345
GFX_BPP4 enumeration member 345
GFX_BPP8 enumeration member 345
GFX_BS_ADDRESS enumeration member 346
GFX_BS_MALLOC enumeration member 346
GFX_BS_MANAGED enumeration member 346
GFX_BS_NONE enumeration member 346
GFX_Buffer type 346
GFX_BUFFER_READ enumeration member 346
GFX_BUFFER_WRITE enumeration member 346
GFX_BufferSelection enumeration 346
GFX_BufferSelection_t enumeration 346
GFX_BufferState enumeration 346
GFX_BufferState_t enumeration 346
GFX_Calloc_FnPtr type 346
GFX_Clampf function 306
GFX_Clampi function 306
GFX_Color type 347
gfx_color.h 382
GFX_COLOR_BLACK enumeration member 348
GFX_COLOR_BLUE enumeration member 348
GFX_COLOR_CYAN enumeration member 348
GFX_COLOR_DARKGRAY enumeration member 348
GFX_COLOR_GRAY enumeration member 348
GFX_COLOR_GREEN enumeration member 348
GFX_COLOR_LAST enumeration member 348
GFX_COLOR_LIGHTGRAY enumeration member 348
GFX_COLOR_LIME enumeration member 348
GFX_COLOR_MAGENTA enumeration member 348
GFX_COLOR_MAROON enumeration member 348
GFX_COLOR_MASK_ALL enumeration member 347
GFX_COLOR_MASK_ARGB_8888 enumeration member 347
GFX_COLOR_MASK_GS_8 enumeration member 347
GFX_COLOR_MASK_RGB_332 enumeration member 347
GFX_COLOR_MASK_RGB_565 enumeration member 347
GFX_COLOR_MASK_RGB_888 enumeration member 347
GFX_COLOR_MASK_RGBA_5551 enumeration member 347
GFX_COLOR_MASK_RGBA_8888 enumeration member 347
GFX_COLOR_MASK_YUV enumeration member 347
GFX_COLOR_MAX_SIZE macro 366
GFX_COLOR_MODE_ARGB_8888 enumeration member 337
GFX_COLOR_MODE_COUNT macro 366
GFX_COLOR_MODE_GS_8 enumeration member 337
GFX_COLOR_MODE_INDEX_1 enumeration member 337
GFX_COLOR_MODE_INDEX_4 enumeration member 337
GFX_COLOR_MODE_INDEX_8 enumeration member 337
GFX_COLOR_MODE_IS_ALPHA macro 367
GFX_COLOR_MODE_IS_INDEX macro 367
GFX_COLOR_MODE_IS_PIXEL macro 367
GFX_COLOR_MODE_LAST enumeration member 337
GFX_COLOR_MODE_LAST_COLOR macro 367
GFX_COLOR_MODE_RGB_332 enumeration member 337
GFX_COLOR_MODE_RGB_565 enumeration member 337
GFX_COLOR_MODE_RGB_888 enumeration member 337
GFX_COLOR_MODE_RGBA_5551 enumeration member 337
GFX_COLOR_MODE_RGBA_8888 enumeration member 337
GFX_COLOR_MODE_YUV enumeration member 337
GFX_COLOR_NAVY enumeration member 348
GFX_COLOR_OLIVE enumeration member 348
GFX_COLOR_PURPLE enumeration member 348
GFX_COLOR_RED enumeration member 348
GFX_COLOR_SILVER enumeration member 348
GFX_COLOR_TEAL enumeration member 348
GFX_COLOR_WHITE enumeration member 348
GFX_COLOR_YELLOW enumeration member 348
GFX_ColorBilerp function 323
GFX_ColorBlend_RGBA_8888 function 324
GFX_ColorChannelAlpha function 307
GFX_ColorChannelBlue function 324
GFX_ColorChannelGreen function 307
GFX_ColorChannelRed function 307
GFX_ColorConvert function 308
GFX_ColorInfo variable 365
GFX_ColorLerp function 308
GFX_ColorMask enumeration 347
GFX_ColorMask_t enumeration 347
GFX_ColorMode type 347
GFX_ColorMode_t enumeration 337
GFX_ColorModeInfo structure 347
GFX_ColorModeInfo_t structure 347
GFX_ColorModeInfoGet function 309
GFX_ColorName enumeration 348
GFX_ColorName_t enumeration 348
GFX_ColorValue function 309
gfx_common.h 380
GFX_Context structure 348
gfx_context.h 383
GFX_Context_t structure 348
GFX_ContextActiveSet function 309
gfx_default_impl.h 384
GFX_DEPRECATED macro 380
GFX_Display type 349
gfx_display.h 384
GFX_DisplayInfo structure 349
GFX_DisplayInfo_t structure 349
GFX_DivideRounding function 325
gfx_draw.h 384
gfx_draw_blit.h 385
gfx_draw_circle.h 385
GFX_DRAW_FILL enumeration member 350
GFX_DRAW_GRADIENT_LEFT_RIGHT enumeration member 350
GFX_DRAW_GRADIENT_TOP_BOTTOM enumeration member 350
GFX_DRAW_LINE enumeration member 350
gfx_draw_line.h 385
GFX_DRAW_MODE_COUNT macro 367
gfx_draw_pixel.h 385
gfx_draw_rect.h 385
GFX_DrawBlit function 325
GFX_DrawCircle function 325
GFX_DrawDirectBlit function 335
GFX_DrawLine function 326

GFX_DrawMode enumeration 350
GFX_DrawMode_t enumeration 350
GFX_DrawPipeline structure 376
GFX_DrawPipeline_t structure 376
GFX_DrawPixel function 326
GFX_DrawRect function 327
GFX_DrawState structure 350
GFX_DrawState_t structure 350
GFX_DrawStretchBlit function 327
GFX_Driver type 351
gfx_driver_interface.h 385
GFX_DriverInfo type 351
GFX_DriverInfo_t structure 337
GFX_FAILURE macro 368
GFX_FALSE macro 368
GFX_Flag enumeration 352
GFX_Flag_t enumeration 352
GFX_FrameBuffer structure 352
GFX_FrameBuffer_t structure 352
GFX_Free_FnPtr type 353
GFX_GLOBAL_PALETTE_SIZE macro 379
GFX_GlobalPalette type 379
GFX_HAL structure 353
gfx_hal.h 386
GFX_HAL_t structure 353
GFX_Handle type 355
gfx_interface.h 387
GFX_Layer structure 355
gfx_layer.h 388
GFX_Layer_t structure 355
GFX_LayerFromOrientedSpace function 328
GFX_LayerPointFromOrientedSpace function 328
GFX_LayerPointToOrientedSpace function 329
GFX_LayerReadBuffer function 309
GFX_LayerRectFromOrientedSpace function 329
GFX_LayerRectToOrientedSpace function 330
GFX_LayerRotate function 310
GFX_LayerSwap function 310
GFX_LayerToOrientedSpace function 330
GFX_LayerWriteBuffer function 310
GFX_Lerp function 311
GFX_Malloc_FnPtr type 356
gfx_math.h 388
GFX_MAX_BUFFER_COUNT macro 368
GFX_Maxf function 311
GFX_Maxi function 311
GFX_Memcpy_FnPtr type 356
GFX_MemoryIntf structure 356
GFX_MemoryIntf_t structure 356
GFX_Memset_FnPtr type 357
GFX_Minf function 312
GFX_Mini function 312
GFX_NULL macro 368
GFX_NUM_FLAGS macro 368
GFX_Orientation enumeration 357
GFX_ORIENTATION_0 enumeration member 357
GFX_ORIENTATION_180 enumeration member 357
GFX_ORIENTATION_270 enumeration member 357
GFX_ORIENTATION_90 enumeration member 357
GFX_Orientation_t enumeration 357
GFX_Percent function 312
GFX_PercentOf function 313
GFX_PercentOfDec function 331
GFX_PercentWholeRounded function 313
GFX_PIPELINE_GCU enumeration member 376
GFX_PIPELINE_GCUGPU enumeration member 376
GFX_PIPELINE_GPU enumeration member 376
GFX_PIPELINE_MODE_COUNT macro 378
GFX_PIPELINE_SOFTWARE enumeration member 376
GFX_PipelineMode enumeration 376
GFX_PipelineMode_t enumeration 376
gfx_pixel_buffer.h 389
GFX_PixelBuffer structure 357
GFX_PixelBuffer_t structure 357
GFX_PixelBufferAreaFill function 313
GFX_PixelBufferAreaFill_Unsafe function 314
GFX_PixelBufferAreaGet function 314
GFX_PixelBufferAreaGet_Unsafe function 315
GFX_PixelBufferAreaSet function 315
GFX_PixelBufferAreaSet_Unsafe function 316
GFX_PixelBufferClipRect function 316
GFX_PixelBufferConvert function 317
GFX_PixelBufferCopy function 317
GFX_PixelBufferCreate function 318
GFX_PixelBufferDestroy function 318
GFX_PixelBufferGet function 318
GFX_PixelBufferGet_Unsafe function 319
GFX_PixelBufferGetIndex function 319
GFX_PixelBufferOffsetGet function 320
GFX_PixelBufferOffsetGet_Unsafe function 320
GFX_PixelBufferSet function 320
GFX_PixelBufferSet_Unsafe function 321
GFX_Point type 224
GFX_Point_t structure 337
GFX_Processor type 358
GFX_Realloc_FnPtr type 358
GFX_Rect type 225
gfx_rect.h 390
GFX_Rect_t structure 338
GFX_Rect_Zero variable 365
GFX_RectClip function 321
GFX_RectClipAdj function 331
GFX_RectCombine function 335
GFX_RectCompare function 336
GFX_RectContainsPoint function 322
GFX_RectContainsRect function 322
GFX_RectFromPoints function 331
GFX_RectIntersects function 322
GFX_RectsAreSimilar function 336
GFX_RectSplit function 331
GFX_RectToPoints function 331
GFX_RESIZE_BILINEAR enumeration member 377
GFX_RESIZE_MODE_COUNT macro 378
GFX_RESIZE_NEARESTNEIGHBOR enumeration member 377
GFX_ResizeMode enumeration 377
GFX_ResizeMode_t enumeration 377

GFX_Result type 359
GFX_ScaleInteger function 323
GFX_ScaleIntegerSigned function 336
GFX_Size structure 359
GFX_Size_t structure 359
GFX_SUCCESS macro 369
GFX_SyncCallback_FnPtr type 359
GFX_TRUE macro 369
GFX_UNSUPPORTED macro 369
gfx_util.h 390
GFX_UtilMirrorPoint function 332
GFX_UtilOrientPoint function 332
GFX_UtilPointFromOrientedSpace function 332
GFX_UtilPointToOrientedSpace function 333
GFX_UtilSizeFromOrientedSpace function 333
GFX_UtilSizeToOrientedSpace function 334
GFX_UtilSortPointsX function 334
GFX_UtilSortPointsY function 334
GFXF_BRIGHTNESS enumeration member 352
GFXF_BRIGHTNESS_RANGE enumeration member 352
GFXF_COLOR_MODE enumeration member 352
GFXF_DISPLAY_COUNT enumeration member 352
GFXF_DISPLAY_INFO enumeration member 352
GFXF_DRAW_ALPHA_ENABLE enumeration member 352
GFXF_DRAW_ALPHA_VALUE enumeration member 352
GFXF_DRAW_BLEND_MODE enumeration member 352
GFXF_DRAW_CLIP_ENABLE enumeration member 352
GFXF_DRAW_CLIP_RECT enumeration member 352
GFXF_DRAW_COLOR enumeration member 352
GFXF_DRAW_GRADIENT_COLOR enumeration member 352
GFXF_DRAW_MASK_ENABLE enumeration member 352
GFXF_DRAW_MASK_VALUE enumeration member 352
GFXF_DRAW_MODE enumeration member 352
GFXF_DRAW_PALETTE enumeration member 352
GFXF_DRAW_PIPELINE_MODE enumeration member 352
GFXF_DRAW_RESIZE_MODE enumeration member 352
GFXF_DRAW_TARGET enumeration member 352
GFXF_DRAW_THICKNESS enumeration member 352
GFXF_DRIVER_COUNT enumeration member 352
GFXF_DRIVER_INFO enumeration member 352
GFXF_GLOBAL_PALETTE enumeration member 352
GFXF_HSYNC_CALLBACK enumeration member 352
GFXF_LAST_FLAG enumeration member 352
GFXF_LAYER_ACTIVE enumeration member 352
GFXF_LAYER_ALPHA_AMOUNT enumeration member 352
GFXF_LAYER_ALPHA_ENABLE enumeration member 352
GFXF_LAYER_BUFFER_ADDRESS enumeration member 352
GFXF_LAYER_BUFFER_ALLOCATE enumeration member 352
GFXF_LAYER_BUFFER_COHERENT enumeration member 352
GFXF_LAYER_BUFFER_COUNT enumeration member 352
GFXF_LAYER_BUFFER_FREE enumeration member 352
GFXF_LAYER_COUNT enumeration member 352
GFXF_LAYER_ENABLED enumeration member 352
GFXF_LAYER_INVALID enumeration member 352
GFXF_LAYER_MASK_COLOR enumeration member 352
GFXF_LAYER_MASK_ENABLE enumeration member 352
GFXF_LAYER_POSITION enumeration member 352
GFXF_LAYER_SIZE enumeration member 352
GFXF_LAYER_SWAP enumeration member 352
GFXF_LAYER_SWAP_SYNC enumeration member 352
GFXF_LAYER_VISIBLE enumeration member 352
GFXF_LAYER_VSYNC enumeration member 352
GFXF_MIRRORED enumeration member 352
GFXF_NONE enumeration member 352
GFXF_ORIENTATION enumeration member 352
GFXF_VSYNC_CALLBACK enumeration member 352
GFXU_ASSET_LOCATION_INTERNAL macro 453
GFXU_ASSET_TYPE_BINARY enumeration member 448
GFXU_ASSET_TYPE_FONT enumeration member 448
GFXU_ASSET_TYPE_IMAGE enumeration member 448
GFXU_ASSET_TYPE_PALETTE enumeration member 448
GFXU_ASSET_TYPE_STRINGTABLE enumeration member 448
GFXU_AssetHeader structure 448
GFXU_AssetHeader_t structure 448
GFXU_AssetType enumeration 448
GFXU_AssetType_t enumeration 448
gfxu_binary.h 455
GFXU_BinaryAsset structure 448
GFXU_BinaryAsset_t structure 448
GFXU_CalculateCharStringWidth function 432
GFXU_CalculatePartialCharStringWidth function 433
GFXU_CalculatePartialStringWidth function 433
GFXU_CalculateStringWidth function 433
GFXU_CHAR type 449
GFXU_CompareString function 434
GFXU_DecodeAndDrawString function 441
GFXU_DecodeAndDrawSubString function 444
GFXU_DecodeCodePoint function 434
GFXU_DecodeUTF16 function 435
GFXU_DecodeUTF8 function 435
GFXU_DrawCharString function 435
GFXU_DrawCharStringClipped function 441
GFXU_DrawCharSubStringClipped function 444
GFXU_DrawGlyph function 442
GFXU_DrawGlyphRow function 442
GFXU_DrawImage function 435
GFXU_DrawString function 436
GFXU_DrawStringClipped function 442
GFXU_DrawSubStringClipped function 445
GFXU_DrawUnknownGlyph function 443
GFXU_ExternalAssetReader structure 449
GFXU_ExternalAssetReader_t structure 449
GFXU_ExternalAssetReaderRun_FnPtr type 449
GFXU_ExternalAssetReaderStatus enumeration 449
GFXU_ExternalAssetReaderStatus_t enumeration 449
GFXU_ExtractString function 437
gfxu_font.h 455
GFXU_FONT_BPP_1 enumeration member 447
GFXU_FONT_BPP_8 enumeration member 447
GFXU_FontAsset type 450
GFXU_FontAsset_t structure 447
GFXU_FontAssetBPP enumeration 447
GFXU_FontGetGlyphInfo function 437
GFXU_FontGetLookupTableEntry function 437
GFXU_FontGlyphIndexTable structure 450
GFXU_FontGlyphIndexTable_t structure 450

GFXU_FontGlyphRange structure 450
GFXU_FontGlyphRange_t structure 450
GFXU_GetCharAt function 438
GFXU_GetCharStringLineRect function 446
GFXU_GetCharWidth function 438
GFXU_GetGlyphRowDataSize function 443
GFXU_GetStringHeight function 438
GFXU_GetStringLength function 439
GFXU_GetStringLineRect function 446
GFXU_GetStringRect function 439
GFXU_GetStringSizeInBytes function 440
gfxu_global.h 456
gfxu_image.h 457
GFXU_IMAGE_COMPRESSION_NONE enumeration member 451
GFXU_IMAGE_COMPRESSION_RLE enumeration member 451
GFXU_IMAGE_DIRECT_BLIT enumeration member 377
GFXU_IMAGE_FORMAT_JPEG enumeration member 451
GFXU_IMAGE_FORMAT_PNG enumeration member 451
GFXU_IMAGE_FORMAT_RAW enumeration member 451
GFXU_IMAGE_SUPPORTS_CLIPPING enumeration member 377
GFXU_IMAGE_USE_MASK enumeration member 377
gfxu_image_utils.h 458
GFXU_ImageAsset structure 451
GFXU_ImageAsset_t structure 451
GFXU_ImageCompressionType enumeration 451
GFXU_ImageCompressionType_t enumeration 451
GFXU_ImageFlags enumeration 377
GFXU_ImageFlags_t enumeration 377
GFXU_ImageFormat enumeration 451
GFXU_ImageFormat_t enumeration 451
GFXU_MediaCloseRequest_FnPtr type 453
GFXU_MediaOpenRequest_FnPtr type 454
GFXU_MediaReadRequest_FnPtr type 454
GFXU_MediaReadRequestCallback_FnPtr type 454
GFXU_MemoryIntf structure 452
GFXU_MemoryIntf_t structure 452
gfxu_palette.h 458
GFXU_PaletteAsset structure 452
GFXU_PaletteAsset_t structure 452
GFXU_PaletteGetColor function 440
GFXU_PreprocessImage function 443
GFXU_READER_STATUS_ABORTED enumeration member 449
GFXU_READER_STATUS_DRAWING enumeration member 449
GFXU_READER_STATUS_FINISHED enumeration member 449
GFXU_READER_STATUS_INVALID enumeration member 449
GFXU_READER_STATUS_READY enumeration member 449
GFXU_READER_STATUS_WAITING enumeration member 449
gfxu_string.h 458
GFXU_STRING_ARRAY_SIZE macro 453
GFXU_STRING_ENCODING_ASCII enumeration member 452
GFXU_STRING_ENCODING_UTF16 enumeration member 452
GFXU_STRING_ENCODING_UTF8 enumeration member 452
GFXU_STRING_ENTRY_SIZE macro 453
GFXU_STRING_MAX_CHAR_WIDTH macro 453
gfxu_string_utils.h 460
GFXU_StringEncodingMode enumeration 452
GFXU_StringEncodingMode_t enumeration 452
GFXU_StringFontIndexLookup function 440
GFXU_StringIndexLookup function 441
GFXU_StringLookup function 441
GFXU_StringTableAsset structure 216
GFXU_StringTableAsset_t structure 216
globalPaletteGet_FnPtr type 379
globalPaletteSet_FnPtr type 379
Graphics Composer Porting 6
Graphics Composer Suite Goals 5
Graphics Composer Suite Salient Features 5
Graphics Libraries Help 3
Graphics Stack Architecture 3
Graphics Utilities Interface 428
Graphics Utilities Library 426

## H

Hardware Abstraction Layer (HAL) 294
How the Library Works 392, 400

## I

ILI9488 Display Controller Driver Library 391
ILI9488_Backlight_Off macro 397
ILI9488_Backlight_On macro 397
ILI9488_CMD_PARAM structure 396
ILI9488_DRV structure 397
ILI9488_DRV_STATE enumeration 396
ILI9488_Intf_Close function 393
ILI9488_Intf_Open function 393
ILI9488_Intf_ReadCmd function 394
ILI9488_Intf_ReadPixels function 394
ILI9488_Intf_WriteCmd function 395
ILI9488_Intf_WritePixels function 395
ILI9488_Reset_Assert macro 398
ILI9488_Reset_Deassert macro 398
ILI9488_SPI_DCX_Command macro 398
ILI9488_SPI_DCX_Data macro 398
ILI9488_SPI_SS_Assert macro 398
ILI9488_SPI_SS_Deassert macro 398
IN macro 422
initialize_FnPtr type 374
interrupt_FnPtr type 374
Introduction 3, 6, 294, 391, 399

## K

KEY_0 enumeration member 234
KEY_1 enumeration member 234
KEY_2 enumeration member 234
KEY_3 enumeration member 234
KEY_4 enumeration member 234
KEY_5 enumeration member 234
KEY_6 enumeration member 234
KEY_7 enumeration member 234
KEY_8 enumeration member 234
KEY_9 enumeration member 234
KEY_A enumeration member 234
KEY_B enumeration member 234
KEY_BACKQUOTE enumeration member 234
KEY_BACKSLASH enumeration member 234
KEY_BACKSPACE enumeration member 234
KEY_BRACKET_LEFT enumeration member 234

KEY_BRACKET_RIGHT enumeration member 234
KEY_C enumeration member 234
KEY_CAPSLOCK enumeration member 234
KEY_COMMA enumeration member 234
KEY_D enumeration member 234
KEY_DOWN enumeration member 234
KEY_E enumeration member 234
KEY_END enumeration member 234
KEY_ENTER enumeration member 234
KEY_EQUALS enumeration member 234
KEY_ESCAPE enumeration member 234
KEY_F enumeration member 234
KEY_F1 enumeration member 234
KEY_F10 enumeration member 234
KEY_F11 enumeration member 234
KEY_F12 enumeration member 234
KEY_F2 enumeration member 234
KEY_F3 enumeration member 234
KEY_F4 enumeration member 234
KEY_F5 enumeration member 234
KEY_F6 enumeration member 234
KEY_F7 enumeration member 234
KEY_F8 enumeration member 234
KEY_F9 enumeration member 234
KEY_G enumeration member 234
KEY_H enumeration member 234
KEY_HOME enumeration member 234
KEY_I enumeration member 234
KEY_INSERT enumeration member 234
KEY_J enumeration member 234
KEY_K enumeration member 234
KEY_KP_0 enumeration member 234
KEY_KP_1 enumeration member 234
KEY_KP_2 enumeration member 234
KEY_KP_3 enumeration member 234
KEY_KP_4 enumeration member 234
KEY_KP_5 enumeration member 234
KEY_KP_6 enumeration member 234
KEY_KP_7 enumeration member 234
KEY_KP_8 enumeration member 234
KEY_KP_9 enumeration member 234
KEY_KP_DIVIDE enumeration member 234
KEY_KP_ENTER enumeration member 234
KEY_KP_MINUS enumeration member 234
KEY_KP_MULTIPLY enumeration member 234
KEY_KP_PERIOD enumeration member 234
KEY_KP_PLUS enumeration member 234
KEY_L enumeration member 234
KEY_LALT enumeration member 234
KEY_LAST enumeration member 234
KEY_LCTRL enumeration member 234
KEY_LEFT enumeration member 234
KEY_LMETA enumeration member 234
KEY_LSHIFT enumeration member 234
KEY_M enumeration member 234
KEY_MINUS enumeration member 234
KEY_N enumeration member 234
KEY_NULL enumeration member 234
KEY_NUMLOCK enumeration member 234
KEY_O enumeration member 234
KEY_P enumeration member 234
KEY_PAGEDOWN enumeration member 234
KEY_PAGEUP enumeration member 234
KEY_PAUSE enumeration member 234
KEY_PERIOD enumeration member 234
KEY_PRINTSCREEN enumeration member 234
KEY_Q enumeration member 234
KEY_QUOTE enumeration member 234
KEY_R enumeration member 234
KEY_RALT enumeration member 234
KEY_RCTRL enumeration member 234
KEY_RIGHT enumeration member 234
KEY_RMETA enumeration member 234
KEY_RSHIFT enumeration member 234
KEY_S enumeration member 234
KEY_SCROLLLOCK enumeration member 234
KEY_SEMICOLON enumeration member 234
KEY_SLASH enumeration member 234
KEY_SPACE enumeration member 234
KEY_T enumeration member 234
KEY_TAB enumeration member 234
KEY_U enumeration member 234
KEY_UP enumeration member 234
KEY_V enumeration member 234
KEY_W enumeration member 234
KEY_X enumeration member 234
KEY_Y enumeration member 234
KEY_Z enumeration member 234

## L

LA_BUFFER_TYPE_ADDRESS enumeration member 239
LA_BUFFER_TYPE_AUTO enumeration member 239
LA_BUTTON_STATE_DOWN enumeration member 225
LA_BUTTON_STATE_TOGGLED enumeration member 225
LA_BUTTON_STATE_UP enumeration member 225
LA_CONTEXT_FRAME_DRAWING enumeration member 264
LA_CONTEXT_FRAME_POSTLAYER enumeration member 264
LA_CONTEXT_FRAME_PREFRAME enumeration member 264
LA_CONTEXT_FRAME_PRELAYER enumeration member 264
LA_CONTEXT_FRAME_READY enumeration member 264
LA_CONTEXT_UPDATE_DONE enumeration member 264
LA_CONTEXT_UPDATE_PENDING enumeration member 264
LA_DEFAULT_SCHEME_COLOR_MODE macro 260
LA_EVENT_DEFERRED enumeration member 263
LA_EVENT_HANDLED enumeration member 263
LA_EVENT_NONE enumeration member 229
LA_EVENT_RESET_QUEUE enumeration member 263
LA_EVENT_SCREEN_CHANGE enumeration member 229
LA_EVENT_TOUCH_DOWN enumeration member 229
LA_EVENT_TOUCH_MOVED enumeration member 229
LA_EVENT_TOUCH_UP enumeration member 229
LA_FAILURE enumeration member 220
LA_FALSE enumeration member 216
LA_GESTURE_NONE enumeration member 230
LA_GRADIENT_DIRECTION_DOWN enumeration member 230
LA_GRADIENT_DIRECTION_LEFT enumeration member 230

LA_GRADIENT_DIRECTION_RIGHT enumeration member 230
LA_GRADIENT_DIRECTION_UP enumeration member 230
LA_IMAGESEQ_RESTART macro 261
LA_INPUT_PRIMARY_ID macro 261
LA_KEYPAD_CELL_ACTION_ACCEPT enumeration member 236
LA_KEYPAD_CELL_ACTION_APPEND enumeration member 236
LA_KEYPAD_CELL_ACTION_BACKSPACE enumeration member 236
LA_KEYPAD_CELL_ACTION_CLEAR enumeration member 236
LA_KEYPAD_CELL_ACTION_NONE enumeration member 236
LA_KEYPAD_CELL_ACTION_SET enumeration member 236
LA_LAYER_FRAME_COMPLETE enumeration member 263
LA_LAYER_FRAME_IN_PROGRESS enumeration member 263
LA_LAYER_FRAME_PREFRAME enumeration member 263
LA_LAYER_FRAME_READY enumeration member 263
LA_LIST_WIDGET_SELECTION_MODE_CONTIGUOUS enumeration member 243
LA_LIST_WIDGET_SELECTION_MODE_MULTIPLE enumeration member 243
LA_LIST_WIDGET_SELECTION_MODE_SINGLE enumeration member 243
LA_MAX_TOUCH_STATES macro 261
LA_PREEMPTION_LEVEL_0 enumeration member 215
LA_PREEMPTION_LEVEL_1 enumeration member 215
LA_PREEMPTION_LEVEL_2 enumeration member 215
LA_PROGRESSBAR_DIRECTION_DOWN enumeration member 244
LA_PROGRESSBAR_DIRECTION_LEFT enumeration member 244
LA_PROGRESSBAR_DIRECTION_RIGHT enumeration member 244
LA_PROGRESSBAR_DIRECTION_UP enumeration member 244
LA_RELATIVE_POSITION_ABOVE enumeration member 215
LA_RELATIVE_POSITION_BEHIND enumeration member 215
LA_RELATIVE_POSITION_BELOW enumeration member 215
LA_RELATIVE_POSITION_LEFTOF enumeration member 215
LA_RELATIVE_POSITION_RIGHTOF enumeration member 215
LA_SCREEN_ORIENTATION_0 enumeration member 248
LA_SCREEN_ORIENTATION_180 enumeration member 248
LA_SCREEN_ORIENTATION_270 enumeration member 248
LA_SCREEN_ORIENTATION_90 enumeration member 248
LA_SCROLLBAR_ORIENT_HORIZONTAL enumeration member 248
LA_SCROLLBAR_ORIENT_VERTICAL enumeration member 248
LA_SCROLLBAR_STATE_BOTTOM_INSIDE enumeration member 248
LA_SCROLLBAR_STATE_BOTTOM_PRESSED enumeration member 248
LA_SCROLLBAR_STATE_HANDLE_DOWN enumeration member 248
LA_SCROLLBAR_STATE_NONE enumeration member 248
LA_SCROLLBAR_STATE_TOP_INSIDE enumeration member 248
LA_SCROLLBAR_STATE_TOP_PRESSED enumeration member 248
LA_SLIDER_ORIENT_HORIZONTAL enumeration member 250
LA_SLIDER_ORIENT_VERTICAL enumeration member 250
LA_SLIDER_STATE_AREA_DOWN enumeration member 250
LA_SLIDER_STATE_HANDLE_DOWN enumeration member 250
LA_SLIDER_STATE_NONE enumeration member 250
LA_STRING_NULLIDX macro 261
LA_SUCCESS enumeration member 220
LA_TOUCHTEST_MEMORY_SIZE macro 262
LA_TOUCHTEST_STATE_DOWN enumeration member 252
LA_TOUCHTEST_STATE_UP enumeration member 252
LA_TRUE enumeration member 216
LA_WIDGET_ARC enumeration member 258
LA_WIDGET_BACKGROUND_CACHE enumeration member 259
LA_WIDGET_BACKGROUND_FILL enumeration member 259
LA_WIDGET_BACKGROUND_LAST enumeration member 259
LA_WIDGET_BACKGROUND_NONE enumeration member 259
LA_WIDGET_BAR_GRAPH enumeration member 258
LA_WIDGET_BORDER_BEVEL enumeration member 225
LA_WIDGET_BORDER_LAST enumeration member 225
LA_WIDGET_BORDER_LINE enumeration member 225
LA_WIDGET_BORDER_NONE enumeration member 225
LA_WIDGET_BUTTON enumeration member 258
LA_WIDGET_CHECKBOX enumeration member 258
LA_WIDGET_CIRCLE enumeration member 258
LA_WIDGET_CIRCULAR_GAUGE enumeration member 258
LA_WIDGET_CIRCULAR_SLIDER enumeration member 258
LA_WIDGET_DIRTY_STATE_CHILD enumeration member 257
LA_WIDGET_DIRTY_STATE_CLEAN enumeration member 257
LA_WIDGET_DIRTY_STATE_DIRTY enumeration member 257
LA_WIDGET_DRAW_STATE_DONE enumeration member 257
LA_WIDGET_DRAW_STATE_READY enumeration member 257
LA_WIDGET_DRAWSURFACE enumeration member 258
LA_WIDGET_GRADIENT enumeration member 258
LA_WIDGET_GROUPBOX enumeration member 258
LA_WIDGET_IMAGE enumeration member 258
LA_WIDGET_IMAGEPLUS enumeration member 258
LA_WIDGET_IMAGESEQUENCE enumeration member 258
LA_WIDGET_KEYPAD enumeration member 258
LA_WIDGET_LABEL enumeration member 258
LA_WIDGET_LAYER enumeration member 258
LA_WIDGET_LINE enumeration member 258
LA_WIDGET_LINE_GRAPH enumeration member 258
LA_WIDGET_LIST enumeration member 258
LA_WIDGET_LISTWHEEL enumeration member 258
LA_WIDGET_OPT_DRAW_ONCE enumeration member 260
LA_WIDGET_OPT_LOCAL_REDRAW enumeration member 260
LA_WIDGET_OPT_OPAQUE enumeration member 260
LA_WIDGET_PIE_CHART enumeration member 258
LA_WIDGET_PROGRESSBAR enumeration member 258
LA_WIDGET_RADIAL_MENU enumeration member 258
LA_WIDGET_RADIOBUTTON enumeration member 258
LA_WIDGET_RECTANGLE enumeration member 258
LA_WIDGET_SCROLLBAR enumeration member 258
LA_WIDGET_SLIDER enumeration member 258
LA_WIDGET_TEXTFIELD enumeration member 258
LA_WIDGET_TOUCHTEST enumeration member 258
LA_WIDGET_UPDATE_STATE_DONE enumeration member 265
LA_WIDGET_UPDATE_STATE_PENDING enumeration member 265
LA_WIDGET_WIDGET enumeration member 258
LA_WIDGET_WINDOW enumeration member 258
laBackgroundType enumeration 259
laBackgroundType_t enumeration 259
laBool enumeration 216
laBool_t enumeration 216
laBorderType enumeration 225
laBorderType_t enumeration 225
laButtonState enumeration 225
laButtonState_t enumeration 225
laButtonWidget type 225
laButtonWidget_GetHAlignment function 52
laButtonWidget_GetImageMargin function 53

laButtonWidget_GetImagePosition function 53
laButtonWidget_GetPressed function 53
laButtonWidget_GetPressedEventCallback function 54
laButtonWidget_GetPressedImage function 54
laButtonWidget_GetPressedOffset function 54
laButtonWidget_GetReleasedEventCallback function 55
laButtonWidget_GetReleasedImage function 55
laButtonWidget_GetText function 55
laButtonWidget_GetToggleable function 56
laButtonWidget_GetVAlignment function 56
laButtonWidget_New function 56
laButtonWidget_PressedEvent type 226
laButtonWidget_ReleasedEvent type 226
laButtonWidget_SetHAlignment function 57
laButtonWidget_SetImageMargin function 57
laButtonWidget_SetImagePosition function 57
laButtonWidget_SetPressed function 58
laButtonWidget_SetPressedEventCallback function 58
laButtonWidget_SetPressedImage function 58
laButtonWidget_SetPressedOffset function 59
laButtonWidget_SetReleasedEventCallback function 59
laButtonWidget_SetReleasedImage function 60
laButtonWidget_SetText function 60
laButtonWidget_SetToggleable function 60
laButtonWidget_SetVAlignment function 61
laButtonWidget_t structure 222
laCheckBoxWidget structure 226
laCheckBoxWidget_CheckedEvent type 227
laCheckBoxWidget_GetChecked function 61
laCheckBoxWidget_GetCheckedEventCallback function 61
laCheckBoxWidget_GetCheckedImage function 62
laCheckBoxWidget_GetHAlignment function 62
laCheckBoxWidget_GetImageMargin function 62
laCheckBoxWidget_GetImagePosition function 63
laCheckBoxWidget_GetText function 63
laCheckBoxWidget_GetUncheckedEventCallback function 63
laCheckBoxWidget_GetUncheckedImage function 64
laCheckBoxWidget_GetVAlignment function 64
laCheckBoxWidget_New function 64
laCheckBoxWidget_SetChecked function 65
laCheckBoxWidget_SetCheckedEventCallback function 65
laCheckBoxWidget_SetCheckedImage function 66
laCheckBoxWidget_SetHAlignment function 66
laCheckBoxWidget_SetImageMargin function 190
laCheckBoxWidget_SetImagePosition function 66
laCheckBoxWidget_SetText function 67
laCheckBoxWidget_SetUncheckedEventCallback function 67
laCheckBoxWidget_SetUncheckedImage function 67
laCheckBoxWidget_SetVAlignment function 68
laCheckBoxWidget_t structure 226
laCheckBoxWidget_UncheckedEvent type 227
laCircleWidget structure 227
laCircleWidget_GetOrigin function 68
laCircleWidget_GetRadius function 68
laCircleWidget_New function 69
laCircleWidget_SetOrigin function 69
laCircleWidget_SetRadius function 69
laCircleWidget_t structure 227
laContext type 216
laContext_ActiveScreenChangedCallback_FnPtr type 217
laContext_AddScreen function 27
laContext_Create function 28
laContext_Destroy function 28
laContext_GetActive function 28
laContext_GetActiveScreen function 29
laContext_GetActiveScreenIndex function 29
laContext_GetColorMode function 29
laContext_GetDefaultScheme function 30
laContext_GetEditWidget function 30
laContext_GetFocusWidget function 30
laContext_GetPremptionLevel function 31
laContext_GetScreenRect function 31
laContext_GetStringLanguage function 31
laContext_GetStringTable function 32
laContext_HideActiveScreen function 32
laContext_IsDrawing function 208
laContext_IsLayerDrawing function 208
laContext_LanguageChangedCallback_FnPtr type 217
laContext_RedrawAll function 32
laContext_RemoveScreen function 32
laContext_SetActive function 33
laContext_SetActiveScreen function 33
laContext_SetActiveScreenChangedCallback function 33
laContext_SetEditWidget function 34
laContext_SetFocusWidget function 34
laContext_SetLanguageChangedCallback function 34
laContext_SetPreemptionLevel function 191
laContext_SetStringLanguage function 35
laContext_SetStringTable function 35
laContext_t structure 214
laContext_Update function 35
laContextFrameState enumeration 264
laContextFrameState_t enumeration 264
laContextUpdateState enumeration 264
laContextUpdateState_t enumeration 264
laDraw_1x2BevelBorder function 70
laDraw_2x1BevelBorder function 70
laDraw_2x2BevelBorder function 71
laDraw_LineBorder function 71
laDrawSurfaceWidget structure 228
laDrawSurfaceWidget_DrawCallback type 228
laDrawSurfaceWidget_GetDrawCallback function 71
laDrawSurfaceWidget_New function 72
laDrawSurfaceWidget_SetDrawCallback function 72
laDrawSurfaceWidget_t structure 228
laEditWidget structure 217
laEditWidget_Accept function 36
laEditWidget_Accept_FnPtr type 218
laEditWidget_Append function 36
laEditWidget_Append_FnPtr type 218
laEditWidget_Backspace function 36
laEditWidget_Backspace_FnPtr type 218
laEditWidget_Clear function 36
laEditWidget_Clear_FnPtr type 218
laEditWidget_EndEdit function 36
laEditWidget_EndEdit_FnPtr type 218

laEditWidget_Set function 37
laEditWidget_Set_FnPtr type 219
laEditWidget_StartEdit function 37
laEditWidget_StartEdit_FnPtr type 219
laEditWidget_t structure 217
laEvent structure 228
laEvent_AddEvent function 72
laEvent_ClearList function 73
laEvent_FilterEvent type 229
laEvent_GetCount function 73
laEvent_ProcessEvents function 73
laEvent_SetFilter function 74
laEvent_t structure 228
laEventID enumeration 229
laEventID_t enumeration 229
laEventResult enumeration 263
laEventResult_t enumeration 263
laEventState structure 229
laEventState_t structure 229
laGestureID enumeration 230
laGestureID_t enumeration 230
laGradientWidget structure 230
laGradientWidget_GetDirection function 74
laGradientWidget_New function 74
laGradientWidget_SetDirection function 75
laGradientWidget_t structure 230
laGradientWidgetDirection enumeration 230
laGradientWidgetDirection_t enumeration 230
laGroupBoxWidget structure 231
laGroupBoxWidget_GetAlignment function 75
laGroupBoxWidget_GetText function 75
laGroupBoxWidget_New function 76
laGroupBoxWidget_SetAlignment function 76
laGroupBoxWidget_SetText function 76
laGroupBoxWidget_t structure 231
laHAlignment enumeration 219
laImageSequenceEntry structure 231
laImageSequenceEntry_t structure 231
laImageSequenceImageChangedEvent_FnPtr type 232
laImageSequenceWidget structure 232
laImageSequenceWidget_GetImage function 77
laImageSequenceWidget_GetImageChangedEventCallback function 77
laImageSequenceWidget_GetImageCount function 78
laImageSequenceWidget_GetImageDelay function 78
laImageSequenceWidget_GetImageHAlignment function 78
laImageSequenceWidget_GetImageVAlignment function 79
laImageSequenceWidget_GetRepeat function 79
laImageSequenceWidget_IsPlaying function 79
laImageSequenceWidget_New function 80
laImageSequenceWidget_Play function 80
laImageSequenceWidget_Rewind function 80
laImageSequenceWidget_SetImage function 81
laImageSequenceWidget_SetImageChangedEventCallback function 81
laImageSequenceWidget_SetImageCount function 82
laImageSequenceWidget_SetImageDelay function 82
laImageSequenceWidget_SetImageHAlignment function 82
laImageSequenceWidget_SetImageVAlignment function 83
laImageSequenceWidget_SetRepeat function 83
laImageSequenceWidget_ShowImage function 84
laImageSequenceWidget_ShowNextImage function 84
laImageSequenceWidget_ShowPreviousImage function 84
laImageSequenceWidget_Stop function 85
laImageSequenceWidget_t structure 232
laImageWidget structure 232
laImageWidget_DrawEventCallback type 264
laImageWidget_GetHAlignment function 85
laImageWidget_GetImage function 85
laImageWidget_GetVAlignment function 86
laImageWidget_New function 86
laImageWidget_SetCallBackEnd function 212
laImageWidget_SetCallBackStart function 212
laImageWidget_SetHAlignment function 86
laImageWidget_SetImage function 87
laImageWidget_SetVAlignment function 87
laImageWidget_t structure 232
laInput_GetEnabled function 87
laInput_InjectTouchDown function 88
laInput_InjectTouchMoved function 88
laInput_InjectTouchUp function 88
laInput_SetEnabled function 89
laInput_TouchDownEvent structure 233
laInput_TouchDownEvent_t structure 233
laInput_TouchMovedEvent structure 233
laInput_TouchMovedEvent_t structure 233
laInput_TouchUpEvent structure 234
laInput_TouchUpEvent_t structure 234
laInputState structure 234
laInputState_t structure 234
laKey enumeration 234
laKey_t enumeration 234
laKeyPadCell structure 236
laKeyPadCell_t structure 236
laKeyPadCellAction enumeration 236
laKeyPadCellAction_t enumeration 236
laKeyPadWidget structure 237
laKeyPadWidget_GetKeyAction function 89
laKeyPadWidget_GetKeyClickEventCallback function 89
laKeyPadWidget_GetKeyDrawBackground function 90
laKeyPadWidget_GetKeyEnabled function 90
laKeyPadWidget_GetKeyImageMargin function 90
laKeyPadWidget_GetKeyImagePosition function 91
laKeyPadWidget_GetKeyPressedImage function 91
laKeyPadWidget_GetKeyReleasedImage function 92
laKeyPadWidget_GetKeyText function 92
laKeyPadWidget_GetKeyValue function 93
laKeyPadWidget_KeyClickEvent type 237
laKeyPadWidget_New function 93
laKeyPadWidget_SetKeyAction function 93
laKeyPadWidget_SetKeyBackgroundType function 191
laKeyPadWidget_SetKeyClickEventCallback function 94
laKeyPadWidget_SetKeyEnabled function 94
laKeyPadWidget_SetKeyImageMargin function 95
laKeyPadWidget_SetKeyImagePosition function 95
laKeyPadWidget_SetKeyPressedImage function 96
laKeyPadWidget_SetKeyReleasedImage function 96
laKeyPadWidget_SetKeyText function 97

laKeyPadWidget_SetKeyValue function 97
laKeyPadWidget_t structure 237
laLabelWidget structure 238
laLabelWidget_GetHAlignment function 98
laLabelWidget_GetText function 98
laLabelWidget_GetVAlignment function 99
laLabelWidget_New function 99
laLabelWidget_SetHAlignment function 99
laLabelWidget_SetText function 100
laLabelWidget_SetVAlignment function 100
laLabelWidget_t structure 238
laLayer type 238
laLayer_AddDamageRect function 262
laLayer_Delete function 100
laLayer_GetAllowInputPassThrough function 191
laLayer_GetAlphaAmount function 101
laLayer_GetAlphaEnable function 101
laLayer_GetBufferCount function 101
laLayer_GetEnabled function 192
laLayer_GetInputRect function 209
laLayer_GetInputRectLocked function 209
laLayer_GetMaskColor function 102
laLayer_GetMaskEnable function 102
laLayer_GetVSync function 102
laLayer_IsDrawing function 209
laLayer_New function 103
laLayer_SetAllowInputPassthrough function 192
laLayer_SetAlphaAmount function 103
laLayer_SetAlphaEnable function 104
laLayer_SetBufferCount function 104
laLayer_SetEnabled function 193
laLayer_SetInputRect function 210
laLayer_SetInputRectLocked function 210
laLayer_SetMaskColor function 104
laLayer_SetMaskEnable function 105
laLayer_SetVSync function 105
laLayer_t structure 223
laLayerBuffer structure 238
laLayerBuffer_t structure 238
laLayerBufferType enumeration 239
laLayerBufferType_t enumeration 239
laLayerFrameState enumeration 263
laLayerFrameState_t enumeration 263
laLineWidget structure 239
laLineWidget_GetEndPoint function 105
laLineWidget_GetStartPoint function 106
laLineWidget_New function 106
laLineWidget_SetEndPoint function 106
laLineWidget_SetStartPoint function 107
laLineWidget_t structure 239
laList type 219
laList_Assign function 37
laList_Clear function 37
laList_Copy function 38
laList_Create function 38
laList_Destroy function 38
laList_Find function 39
laList_Get function 39
laList_InsertAt function 39
laList_PopBack function 40
laList_PopFront function 40
laList_PushBack function 40
laList_PushFront function 41
laList_Remove function 41
laList_RemoveAt function 41
laList_t structure 215
laListItem structure 239
laListItem_t structure 239
laListNode structure 220
laListNode_t structure 220
laListWheelItem structure 240
laListWheelItem_t structure 240
laListWheelWidget structure 240
laListWheelWidget_AppendItem function 107
laListWheelWidget_GetAlignment function 107
laListWheelWidget_GetFlickInitSpeed function 193
laListWheelWidget_GetIconMargin function 108
laListWheelWidget_GetIconPosition function 108
laListWheelWidget_GetIndicatorArea function 193
laListWheelWidget_GetItemCount function 108
laListWheelWidget_GetItemIcon function 109
laListWheelWidget_GetItemText function 109
laListWheelWidget_GetMaxMomentum function 194
laListWheelWidget_GetMomentumFalloffRate function 194
laListWheelWidget_GetRotationUpdateRate function 194
laListWheelWidget_GetSelectedItem function 110
laListWheelWidget_GetSelectedItemChangedEventCallback function 110
laListWheelWidget_GetShaded function 195
laListWheelWidget_GetShowIndicators function 195
laListWheelWidget_GetVisibleItemCount function 195
laListWheelWidget_InsertItem function 110
laListWheelWidget_New function 111
laListWheelWidget_RemoveAllItems function 111
laListWheelWidget_RemoveItem function 111
laListWheelWidget_SelectedItemChangedEvent type 242
laListWheelWidget_SelectNextItem function 112
laListWheelWidget_SelectPreviousItem function 112
laListWheelWidget_SetAlignment function 112
laListWheelWidget_SetFlickInitSpeed function 196
laListWheelWidget_SetIconMargin function 113
laListWheelWidget_SetIconPosition function 113
laListWheelWidget_SetIndicatorArea function 196
laListWheelWidget_SetItemIcon function 114
laListWheelWidget_SetItemText function 114
laListWheelWidget_SetMaxMomentum function 197
laListWheelWidget_SetMomentumFalloffRate function 197
laListWheelWidget_SetRotationUpdateRate function 198
laListWheelWidget_SetSelectedItem function 114
laListWheelWidget_SetSelectedItemChangedEventCallback function 115
laListWheelWidget_SetShaded function 198
laListWheelWidget_SetShowIndicators function 198
laListWheelWidget_SetVisibleItemCount function 199
laListWheelWidget_t structure 240
laListWidget structure 242
laListWidget_AppendItem function 115
laListWidget_DeselectAll function 116

laListWidget_GetAlignment function 116
laListWidget_GetAllowEmptySelection function 116
laListWidget_GetFirstSelectedItem function 117
laListWidget_GetIconMargin function 117
laListWidget_GetIconPosition function 117
laListWidget_GetItemCount function 118
laListWidget_GetItemIcon function 118
laListWidget_GetItemSelected function 118
laListWidget_GetItemText function 119
laListWidget_GetLastSelectedItem function 119
laListWidget_GetSelectedItemChangedEventCallback function 119
laListWidget_GetSelectionCount function 120
laListWidget_GetSelectionMode function 120
laListWidget_InsertItem function 120
laListWidget_ItemSelectedChangedEvent type 243
laListWidget_New function 121
laListWidget_RemoveAllItems function 121
laListWidget_RemoveItem function 121
laListWidget_SelectAll function 122
laListWidget_SelectedItemChangedEvent type 243
laListWidget_SelectionMode enumeration 243
laListWidget_SelectionMode_t enumeration 243
laListWidget_SetAlignment function 122
laListWidget_SetAllowEmptySelection function 123
laListWidget_SetIconMargin function 123
laListWidget_SetIconPosition function 123
laListWidget_SetItemIcon function 124
laListWidget_SetItemSelected function 124
laListWidget_SetItemText function 125
laListWidget_SetItemVisible function 125
laListWidget_SetSelectedItemChangedEventCallback function 125
laListWidget_SetSelectionMode function 126
laListWidget_t structure 242
laListWidget_ToggleItemSelected function 126
laMargin structure 220
laMargin_t structure 220
laMouseButton enumeration 243
laMouseButton_t enumeration 243
laPreemptionLevel enumeration 215
laProgressBar type 244
laProgressBar_ValueChangedEventCallback type 244
laProgressBarDirection enumeration 244
laProgressBarDirection_t enumeration 244
laProgressBarWidget structure 244
laProgressBarWidget_GetDirection function 127
laProgressBarWidget_GetValue function 127
laProgressBarWidget_GetValueChangedEventCallback function 127
laProgressBarWidget_New function 128
laProgressBarWidget_SetDirection function 128
laProgressBarWidget_SetValue function 128
laProgressBarWidget_SetValueChangedCallback function 129
laProgressBarWidget_t structure 244
laRadioButtonGroup type 245
laRadioButtonGroup_AddButton function 129
laRadioButtonGroup_Create function 129
laRadioButtonGroup_Destroy function 130
laRadioButtonGroup_RemoveButton function 130
laRadioButtonGroup_SelectButton function 130
laRadioButtonGroup_t structure 224
laRadioButtonWidget structure 245
laRadioButtonWidget_DeselectedEvent type 246
laRadioButtonWidget_GetDeselectedEventCallback function 131
laRadioButtonWidget_GetGroup function 131
laRadioButtonWidget_GetHAlignment function 131
laRadioButtonWidget_GetImageMargin function 132
laRadioButtonWidget_GetImagePosition function 132
laRadioButtonWidget_GetSelected function 132
laRadioButtonWidget_GetSelectedEventCallback function 133
laRadioButtonWidget_GetSelectedImage function 133
laRadioButtonWidget_GetText function 133
laRadioButtonWidget_GetUnselectedImage function 134
laRadioButtonWidget_GetVAlignment function 134
laRadioButtonWidget_New function 134
laRadioButtonWidget_SelectedEvent type 246
laRadioButtonWidget_SetDeselectedEventCallback function 135
laRadioButtonWidget_SetHAlignment function 135
laRadioButtonWidget_SetImageMargin function 199
laRadioButtonWidget_SetImagePosition function 136
laRadioButtonWidget_SetSelected function 136
laRadioButtonWidget_SetSelectedEventCallback function 136
laRadioButtonWidget_SetSelectedImage function 137
laRadioButtonWidget_SetText function 137
laRadioButtonWidget_SetUnselectedImage function 138
laRadioButtonWidget_SetVAlignment function 138
laRadioButtonWidget_t structure 245
laRectangleWidget structure 246
laRectangleWidget_GetThickness function 138
laRectangleWidget_New function 139
laRectangleWidget_SetThickness function 205
laRectangleWidget_t structure 246
laRectArray type 263
laRelativePosition enumeration 215
laResult enumeration 220
laResult_t enumeration 220
laScheme structure 247
laScheme_Initialize function 139
laScheme_t structure 247
laScreen structure 221
laScreen_CreateCallback_FnPtr type 247
laScreen_Delete function 139
laScreen_GetHideEventCallback function 140
laScreen_GetLayerIndex function 140
laScreen_GetLayerSwapSync function 211
laScreen_GetMirrored function 199
laScreen_GetOrientation function 141
laScreen_GetShowEventCallback function 141
laScreen_Hide function 141
laScreen_New function 142
laScreen_SetHideEventCallback function 142
laScreen_SetLayer function 143
laScreen_SetLayerSwapSync function 211
laScreen_SetMirrored function 200
laScreen_SetOrientation function 143
laScreen_SetShowEventCallback function 144
laScreen_Show function 144
laScreen_ShowHideCallback_FnPtr type 248

laScreen_t structure 221
laScreenOrientation enumeration 248
laScreenOrientation_t enumeration 248
laScrollBarOrientation enumeration 248
laScrollBarOrientation_t enumeration 248
laScrollBarState enumeration 248
laScrollBarState_t enumeration 248
laScrollBarWidget structure 249
laScrollBarWidget_GetExtentValue function 144
laScrollBarWidget_GetMaxinumValue function 145
laScrollBarWidget_GetOrientation function 145
laScrollBarWidget_GetScrollPercentage function 145
laScrollBarWidget_GetScrollValue function 146
laScrollBarWidget_GetStepSize function 146
laScrollBarWidget_GetValueChangedEventCallback function 147
laScrollBarWidget_New function 147
laScrollBarWidget_SetExtentValue function 147
laScrollBarWidget_SetMaximumValue function 148
laScrollBarWidget_SetOrientation function 148
laScrollBarWidget_SetScrollPercentage function 148
laScrollBarWidget_SetScrollValue function 149
laScrollBarWidget_SetStepSize function 149
laScrollBarWidget_SetValueChangedEventCallback function 149
laScrollBarWidget_StepBackward function 150
laScrollBarWidget_StepForward function 150
laScrollBarWidget_t structure 249
laScrollBarWidget_ValueChangedEvent type 250
laSliderOrientation enumeration 250
laSliderOrientation_t enumeration 250
laSliderState enumeration 250
laSliderState_t enumeration 250
laSliderWidget structure 250
laSliderWidget_GetGripSize function 151
laSliderWidget_GetMaxinumValue function 151
laSliderWidget_GetMininumValue function 151
laSliderWidget_GetOrientation function 152
laSliderWidget_GetSliderPercentage function 152
laSliderWidget_GetSliderValue function 152
laSliderWidget_GetValueChangedEventCallback function 153
laSliderWidget_New function 153
laSliderWidget_SetGripSize function 153
laSliderWidget_SetMaximumValue function 154
laSliderWidget_SetMinimumValue function 154
laSliderWidget_SetOrientation function 154
laSliderWidget_SetSliderPercentage function 155
laSliderWidget_SetSliderValue function 155
laSliderWidget_SetValueChangedEventCallback function 155
laSliderWidget_Step function 156
laSliderWidget_t structure 250
laSliderWidget_ValueChangedEvent type 251
laString structure 221
laString_Allocate function 42
laString_Append function 42
laString_Capacity function 43
laString_CharAt function 43
laString_Clear function 43
laString_Compare function 44
laString_CompareBuffer function 44
laString_Copy function 44
laString_CreateFromBuffer function 45
laString_CreateFromCharBuffer function 45
laString_CreateFromID function 46
laString_Delete function 46
laString_Destroy function 46
laString_Draw function 47
laString_DrawClipped function 205
laString_DrawSubStringClipped function 212
laString_ExtractFromTable function 47
laString_GetCharIndexAtPoint function 47
laString_GetCharOffset function 48
laString_GetCharWidth function 48
laString_GetHeight function 48
laString_GetLineRect function 213
laString_GetMultiLineRect function 213
laString_GetRect function 49
laString_Initialize function 49
laString_Insert function 50
laString_IsEmpty function 206
laString_Length function 50
laString_New function 50
laString_ReduceLength function 51
laString_Remove function 200
laString_Set function 51
laString_SetCapacity function 51
laString_t structure 221
laString_ToCharBuffer function 52
laTextFieldWidget structure 251
laTextFieldWidget_GetAlignment function 156
laTextFieldWidget_GetCursorDelay function 157
laTextFieldWidget_GetCursorEnabled function 157
laTextFieldWidget_GetCursorPosition function 157
laTextFieldWidget_GetText function 158
laTextFieldWidget_GetTextChangedEventCallback function 158
laTextFieldWidget_New function 158
laTextFieldWidget_SetAlignment function 159
laTextFieldWidget_SetClearOnFirstEdit function 201
laTextFieldWidget_SetCursorDelay function 159
laTextFieldWidget_SetCursorEnabled function 159
laTextFieldWidget_SetCursorPosition function 160
laTextFieldWidget_SetText function 160
laTextFieldWidget_SetTextChangedEventCallback function 161
laTextFieldWidget_t structure 251
laTextFieldWidget_TextChangedCallback type 252
laTouchState structure 252
laTouchState_t structure 252
laTouchTest_AddPoint function 161
laTouchTest_ClearPoints function 161
laTouchTestState enumeration 252
laTouchTestState_t enumeration 252
laTouchTestWidget structure 253
laTouchTestWidget_GetPointAddedEventCallback function 162
laTouchTestWidget_New function 162
laTouchTestWidget_PointAddedEventCallback type 253
laTouchTestWidget_SetPointAddedEventCallback function 162
laTouchTestWidget_t structure 253
laUtils_ArrangeRectangle function 163

laUtils_ArrangeRectangleRelative function 164
laUtils_ChildIntersectsParent function 164
laUtils_ClipRectToParent function 165
laUtils_GetLayer function 165
laUtils_GetNextHighestWidget function 206
laUtils_ListOcclusionCullTest function 165
laUtils_OcclusionCullTest function 166
laUtils_Pick function 166
laUtils_PickFromLayer function 201
laUtils_PickRect function 167
laUtils_PointScreenToLocalSpace function 167
laUtils_PointToLayerSpace function 201
laUtils_RectFromLayerSpace function 206
laUtils_RectFromParentSpace function 167
laUtils_RectToLayerSpace function 168
laUtils_RectToParentSpace function 168
laUtils_RectToScreenSpace function 168
laUtils_ScreenToMirroredSpace function 202
laUtils_ScreenToOrientedSpace function 202
laUtils_WidgetIsOccluded function 207
laUtils_WidgetLayerRect function 207
laUtils_WidgetLocalRect function 203
laVAlignment enumeration 222
laWidget structure 253
laWidget_AddChild function 169
laWidget_Constructor_FnPtr type 255
laWidget_Delete function 169
laWidget_DeleteAllDescendants function 211
laWidget_Destructor_FnPtr type 255
laWidget_DrawFunction_FnPtr type 255
laWidget_Focus_FnPtr type 256
laWidget_GetAlphaAmount function 170
laWidget_GetAlphaEnable function 170
laWidget_GetBackgroundType function 203
laWidget_GetBorderType function 170
laWidget_GetChildAtIndex function 171
laWidget_GetChildCount function 171
laWidget_GetCumulativeAlphaAmount function 172
laWidget_GetCumulativeAlphaEnable function 172
laWidget_GetEnabled function 172
laWidget_GetHeight function 173
laWidget_GetIndexOfChild function 173
laWidget_GetMargin function 174
laWidget_GetOptimizationFlags function 203
laWidget_GetScheme function 174
laWidget_GetVisible function 174
laWidget_GetWidth function 175
laWidget_GetX function 175
laWidget_GetY function 176
laWidget_HasFocus function 176
laWidget_Invalidate function 176
laWidget_InvalidateBorderAreas_FnPtr type 263
laWidget_isOpaque function 177
laWidget_LanguageChangingEvent_FnPtr type 260
laWidget_Moved_FnPtr type 256
laWidget_New function 177
laWidget_OverrideTouchDownEvent function 178
laWidget_OverrideTouchMovedEvent function 178
laWidget_OverrideTouchUpEvent function 178
laWidget_Paint_FnPtr type 256
laWidget_RectToLayerSpace function 179
laWidget_RectToParentSpace function 179
laWidget_RectToScreenSpace function 180
laWidget_RemoveChild function 180
laWidget_Resize function 180
laWidget_Resized_FnPtr type 256
laWidget_SetAlphaAmount function 181
laWidget_SetAlphaEnable function 181
laWidget_SetBackgroundType function 204
laWidget_SetBorderType function 182
laWidget_SetEnabled function 182
laWidget_SetFocus function 183
laWidget_SetHeight function 183
laWidget_SetMargins function 183
laWidget_SetOptimizationFlags function 204
laWidget_SetParent function 184
laWidget_SetPosition function 184
laWidget_SetScheme function 185
laWidget_SetSize function 185
laWidget_SetVisible function 186
laWidget_SetWidth function 186
laWidget_SetX function 187
laWidget_SetY function 187
laWidget_t structure 253
laWidget_TouchDownEvent_FnPtr type 256
laWidget_TouchMovedEvent_FnPtr type 257
laWidget_TouchUpEvent_FnPtr type 257
laWidget_Translate function 187
laWidget_Update_FnPtr type 257
laWidgetDirtyState enumeration 257
laWidgetDirtyState_t enumeration 257
laWidgetDrawState enumeration 257
laWidgetDrawState_t enumeration 257
laWidgetEvent structure 258
laWidgetEvent_t structure 258
laWidgetOptimizationFlags enumeration 260
laWidgetOptimizationFlags_t enumeration 260
laWidgetType enumeration 258
laWidgetType_t enumeration 258
laWidgetUpdateState enumeration 265
laWidgetUpdateState_t enumeration 265
laWindowWidget structure 259
laWindowWidget_GetIcon function 188
laWindowWidget_GetIconMargin function 188
laWindowWidget_GetIconRect function 207
laWindowWidget_GetTextRect function 207
laWindowWidget_GetTitle function 189
laWindowWidget_GetTitleBarRect function 208
laWindowWidget_New function 189
laWindowWidget_SetIcon function 189
laWindowWidget_SetIconMargin function 190
laWindowWidget_SetTitle function 190
laWindowWidget_t structure 259
layerActiveGet_FnPtr type 359
layerActiveSet_FnPtr type 359
layerAlphaAmountGet_FnPtr type 360

layerAlphaAmountSet_FnPtr type 360
layerBufferAddressGet_FnPtr type 360
layerBufferAddressSet_FnPtr type 360
layerBufferAllocate_FnPtr type 360
layerBufferCoherentGet_FnPtr type 361
layerBufferCoherentSet_FnPtr type 361
layerBufferCountGet_FnPtr type 361
layerBufferCountSet_FnPtr type 361
layerBufferFree_FnPtr type 361
layerBufferIsAllocated_FnPtr type 362
layerEffectSet_FnPtr type 380
layerMaskColorGet_FnPtr type 362
layerMaskColorSet_FnPtr type 362
layerPositionGet_FnPtr type 362
layerPositionSet_FnPtr type 362
layerSizeGet_FnPtr type 363
layerSizeSet_FnPtr type 363
layerSwapped_FnPtr type 363
layerSwapPending_FnPtr type 377
libaria_common.h 266
libaria_context.h 266
libaria_draw.h 268
libaria_editwidget.h 268
libaria_event.h 269
libaria_global.h 269
libaria_input.h 270
libaria_layer.h 270
libaria_list.h 272
libaria_math.h 272
libaria_radiobutton_group.h 272
libaria_scheme.h 273
libaria_screen.h 273
libaria_string.h 274
libaria_utils.h 275
libaria_widget.h 276
libaria_widget_button.h 278
libaria_widget_checkbox.h 280
libaria_widget_circle.h 281
libaria_widget_drawsurface.h 281
libaria_widget_gradient.h 282
libaria_widget_groupbox.h 282
libaria_widget_image.h 283
libaria_widget_imagesequence.h 283
libaria_widget_keypad.h 284
libaria_widget_label.h 285
libaria_widget_line.h 286
libaria_widget_list.h 286
libaria_widget_listwheel.h 287
libaria_widget_progressbar.h 289
libaria_widget_radiobutton.h 289
libaria_widget_rectangle.h 290
libaria_widget_scrollbar.h 291
libaria_widget_slider.h 291
libaria_widget_textfield.h 292
libaria_widget_touchtest.h 293
libaria_widget_window.h 294
libnano2D.h 424
libnano2D_types.h 425
Library Interface 299, 393, 408
Library Overview 400
LIbrary Overview 392

## M

maskColor_FnPtr type 377
mirrorPoint_FnPtr type 378
MPLAB Harmony Graphics Composer (MHGC) Suite 3

## N

N2D_0 enumeration member 415
N2D_180 enumeration member 415
N2D_270 enumeration member 415
N2D_90 enumeration member 415
N2D_A8 enumeration member 413
N2D_BGR565 enumeration member 413
N2D_BGRA4444 enumeration member 413
N2D_BGRA8888 enumeration member 413
n2d_blend enumeration 413
N2D_BLEND_ADDITIVE enumeration member 413
N2D_BLEND_DST_IN enumeration member 413
N2D_BLEND_DST_OVER enumeration member 413
N2D_BLEND_NONE enumeration member 413
N2D_BLEND_SRC_IN enumeration member 413
N2D_BLEND_SRC_OVER enumeration member 413
N2D_BLEND_SUBTRACT enumeration member 413
n2d_blend_t enumeration 413
n2d_blit function 410
n2d_bool_t type 417
n2d_buffer structure 414
n2d_buffer_format enumeration 413
n2d_buffer_format_t enumeration 413
n2d_buffer_t structure 414
n2d_color_t type 414
n2d_dither function 413
n2d_draw_state function 410
n2d_error enumeration 415
n2d_error_t enumeration 415
N2D_FALSE macro 422
n2d_fill function 411
n2d_float_t type 417
N2D_GENERIC_IO enumeration member 415
N2D_INFINITE macro 422
n2d_init_hardware function 411
n2d_int16_t type 417
n2d_int32_t type 417
N2D_INVALID_ARGUMENT enumeration member 415
N2D_IS_ERROR macro 423
N2D_IS_SUCCESS macro 423
n2d_line function 412
n2d_module_parameters structure 415
n2d_module_parameters_t structure 415
N2D_NO_CONTEXT enumeration member 415
N2D_NOT_SUPPORTED enumeration member 415
N2D_NULL macro 423
N2D_ON_ERROR macro 423
n2d_open function 412
n2d_orientation enumeration 415

n2d_orientation_t enumeration 415
N2D_OUT_OF_MEMORY enumeration member 415
N2D_OUT_OF_RESOURCES enumeration member 415
n2d_point structure 416
n2d_point_t structure 416
n2d_rectangle structure 416
n2d_rectangle_t structure 416
N2D_RGB565 enumeration member 413
N2D_RGBA4444 enumeration member 413
N2D_RGBA8888 enumeration member 413
n2d_size_t type 417
N2D_SUCCESS enumeration member 415
N2D_TIMEOUT enumeration member 415
n2d_transparency enumeration 416
N2D_TRANSPARENCY_DESTINATION enumeration member 416
N2D_TRANSPARENCY_NONE enumeration member 416
N2D_TRANSPARENCY_SOURCE enumeration member 416
n2d_transparency_t enumeration 416
N2D_TRUE macro 423
n2d_uint16_t type 418
n2d_uint32_t type 418
n2d_uint64_t type 418
n2d_uint8_t type 418
Nano2D Graphics Processing Unit (GPU) Driver Library 399
NUM_BUTTONS macro 262
NUM_KEYS macro 262

## O

orientationGet_FnPtr type 363
orientationSet_FnPtr type 363
orientPoint_FnPtr type 378
OUT macro 424

## P

pixelGet_FnPtr type 364
pixelGetArray_FnPtr type 378
pixelSet_FnPtr type 364

## R

RGB_2_BITS macro 369
RGB_3_BITS macro 369
RGB_332_BLUE_MASK macro 370
RGB_332_GREEN_MASK macro 370
RGB_332_RED_MASK macro 370
RGB_5_BITS macro 370
RGB_565_BLUE_MASK macro 370
RGB_565_GREEN_MASK macro 371
RGB_565_RED_MASK macro 371
RGB_6_BITS macro 371
RGB_8_BITS macro 371
RGB_888_BLUE_MASK macro 371
RGB_888_GREEN_MASK macro 372
RGB_888_RED_MASK macro 372
RGBA_5551_ALPHA_MASK macro 372
RGBA_5551_BLUE_MASK macro 372
RGBA_5551_GREEN_MASK macro 372
RGBA_5551_RED_MASK macro 373
RGBA_8888_ALPHA_MASK macro 373
RGBA_8888_BLUE_MASK macro 373
RGBA_8888_GREEN_MASK macro 373
RGBA_8888_RED_MASK macro 373

## S

SEGGER emWin Graphics Library 460
syncCallbackGet_FnPtr type 364
syncCallbackSet_FnPtr type 364
syncCallbackSt_FnPtr type 364

## U

update_FnPtr type 365
Using the Library 391, 399

## V

Volume V: MPLAB Harmony Framework Reference 2