



MPLAB Harmony Graphics Library Help

MPLAB Harmony Integrated Software Framework

Graphics Overview

This topic provides a brief overview of graphics and MPLAB Harmony support for graphics.

Description

This distribution package contains a variety of graphics-related firmware projects that demonstrate the capabilities of the MPLAB Harmony Graphics.



The **gfx** repository ONLY contains the files for MPLAB Harmony Graphics quickstart applications, drivers and templates.

Important!

Due to distribution streamlining, the MPLAB Harmony Graphics applications are split between two repositories.

Quickstart applications, such as [aria_quickstart](#), can be found under the **apps folder** in Harmony **gfx** repository. Non-quickstart graphics applications reside under the **apps folder** in the Harmony **gfx_apps** repository.

Although each application can run standalone from within their respective repository, the use of MPLAB Harmony Configurator and its code regeneration feature require the presence of other Harmony repositories.

For information on Harmony repository dependencies, please refer to the Software Requirements section of Graphics Release Notes.

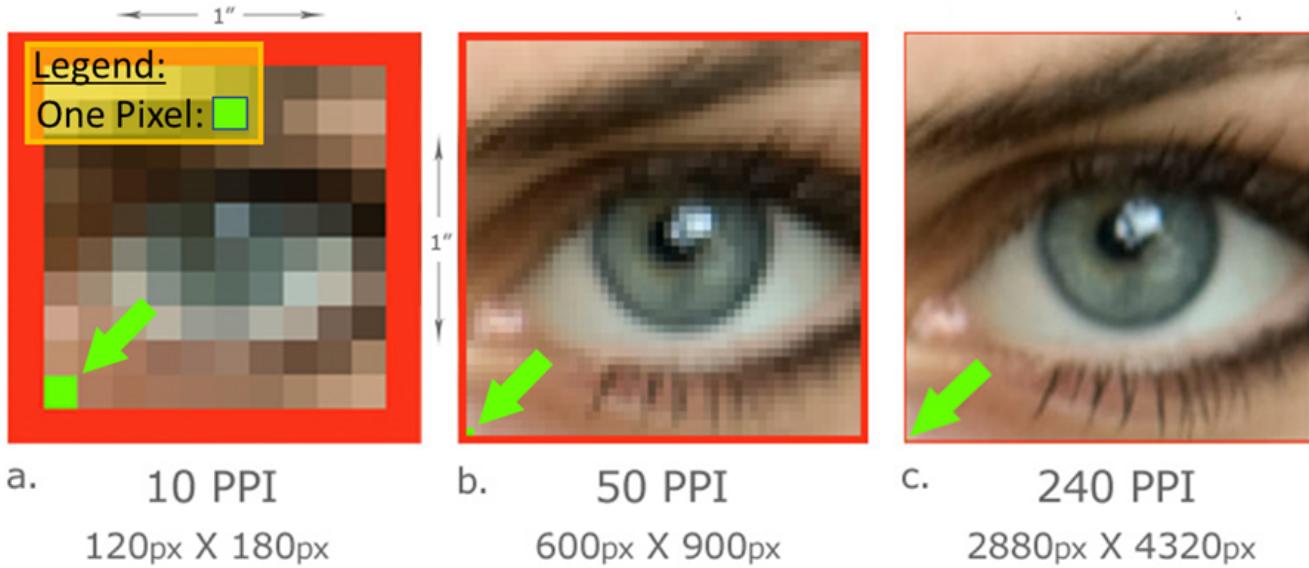
Graphics Basics

This topic covers key concepts in graphics.

Description

For someone new to graphics, this section provides definitions of basic concepts found in most discussions of graphics.

- Pixel:** A pixel is a physical point in a raster image, or the smallest addressable element in an all points addressable display device. The number of pixels per inch (PPI) is a metric for how finely the image is displayed:

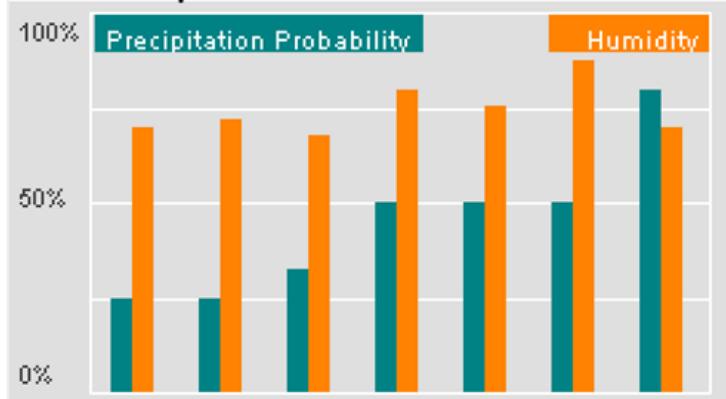


- Primitive:** A primitive is a basic building block for drawing to the screen: e.g. point, line, arc, circle, bar. Widgets are created by assembling multiple primitives.
- Widget:** A widget is a graphical object that is used to convey information or provide user input to the system.

Examples of Widgets in MPLAB Harmony Graphics are:

Button: 

Bar Graph:



Circular Gauge:

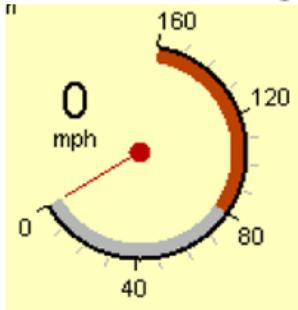


Image:



- **Font:** A set of type or characters of one style and size, for example:

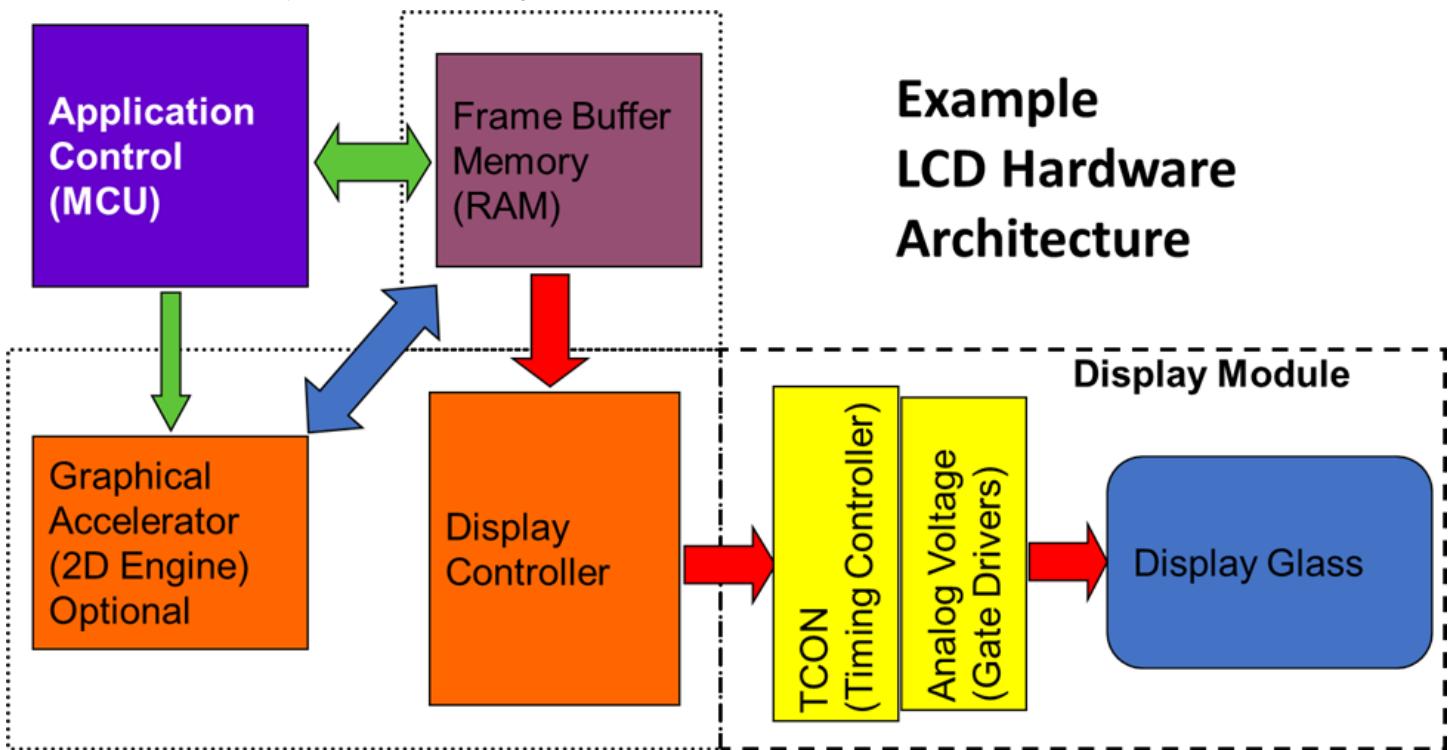
Arial-Bold 16pt, Courier New 12pt, Times New Roman 16pt

- **Glyph:** A pixel representation of an individual character in a given typeface, such as Times New Roman or Arial. On a computer a Font is a digital collection of glyphs for a given typeface.
- **BLIT:** Bit Block Transfer, a data operation that combines two or more bitmaps using Boolean functions. An example BLIT would be to add an animated object (sprite) on top of a background image while ignoring the background (black) pixels in the sprite's bitmap.
- **Layer:** The objects of a graphics design can be organized into a hierarchy with child objects inheriting properties from a parent object, which can also be a child of another parent object higher in the hierarchy. Every design in MPLAB Harmony Graphics has a Layer0. Additional layers can be added in the Tree View panel to provide overlapping layers of objects which are all children of their parent layer.
- **Sprite:** A small two-dimensional bit map that is added to a larger scene. Typically, sprites are used to simulate movement by quickly erasing a prior drawing and redrawing the sprite in a slightly different position.
- **Display Resolution or Resolution:** The number of distinct pixels in each dimension that can be displayed. It is usually quoted in width x height in units of pixels.
- **Alpha Blending:** A graphical technique which blends pixels of overlapping widgets together instead of overwriting one with another. Alpha blending ranges from fully transparent (invisible) to fully visible (blocking all beneath).
- **Color Depth:** The number of bits required to represent the color of a single pixel. It is expressed as bits per pixel (bpp). Typical bits per pixel are 8, 16, 24, and 32.
- **Color Model, RGB:** The RGB Color Model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. Colors are expressed as a triplet (RGB) (Source: Wikipedia).

Examples of color models are:

- **RGB 565** (16 bpp, Red: 5 bits, Green: 6 bits, Blue: 5 bits)
- **RGBA 8888** (32 bpp, Red: 8 bits, Green 8 bits, Blue: 8 bits, Alpha: 8 Bits)
- **Graphics Controller**: The peripheral of a processor or separate device responsible for driving the pixel data to the display.
- **Frame Buffer**: The block of memory that stores the pixel data to be transferred to the screen. A Frame Buffer can be thought of as a copy of the display's pixel values.
- **Graphics Processor**: A dedicated hardware within the microcontroller designed to accelerate updating the frame buffer for primitives such as lines and rectangles and for bit block transfers (blits).

Here is an example Display Hardware Block Diagram:



Why Graphics?

A brief discussion of why graphics has become increasing important to modern embedded systems.

Description

The arrival of smartphones, with feature-rich graphics, easy-to-use touch interface, and an intuitive interface has changed user expectations for all manner of products, from coffee makers to industrial control panels. Users expect an intuitive self-explanatory touch-enabled user interface with rich colors and crisp industrial design esthetics even in the simplest of products. Simple gestures like swiping and the pinch zoom are all part of the touch interactivity lexicon. When compared to as recent as a decade ago, modern embedded graphics is on another level of sophistication.

To standout in a crowded market, adding graphics to a full-color display with touch has become a necessity.

Every Graphical User Interface (GUI) involves some combination of three elements: images, text and geometric shapes. The use of fonts allows customization of text. Image formats such as JPEG or PNG may be used in-lieu of raw uncompressed pixels images, depending on the demands of the application. The capability to draw simple geometric shapes like circles and rectangles by the graphics library trades CPU cycles for storage and memory space.

To manage all three elements, a modern graphics library uses parameterized data structures called widgets. Widgets may also be touch-interactive elements such as buttons, slider bars, or list wheel. Widgets are grouped into screens. Whether widgets are drawn in front or behind (the z-order) within the screen must be managed as well. (A goal of any well designed and organized GUI is to minimize pixel redraws that can occur when objects overlap).

To build screens using widgets, a what-you-see-is-what-you-get (WYSIWIG) design tool is essential. The GUI design tool should provide a pixel-exact representation of screen objects and provide quick iterations of object edits.

There are many dimensions to a good Graphical User Interface Design:

- **Language** – The GUI should support the user's language.
- **Graphics** – The GUI should have a modern, fresh look and be simple, complete, easy to understand.
- **Motion** – When motion is used in the design, what does it signify? Is it merely for entertainment? Or does it provide additional information to the user?
- **Information Design** – What information is the GUI communicating to the user? Is this information easy-to-understand and use?
- **Interface Design** – How can the user manipulate the interface to accomplish useful tasks?
- **Interaction Design** – How does the underlying software/firmware respond to user inputs? Do these responses improve or detract from the user's experience?
- **Programming** – How is the GUI implemented in the application? Is the implementation robust and error free?

There are several trends in today's User Interface designs that can help keep a GUI design modern and fresh looking:

- **Seamless Interface** – Minimize screen transitions. Bring up needed content without changing the application's display or showing a lot of redraws. (Supported by double buffering, layers, mask colors.)
- **Typography** – Big font headlines in combination with smooth animation is trending. It provides an eye-catching design without using additional content. (Supported by flexible font management tools which minimize the memory footprint of fonts and strings.)
- **Gradient Transitions** – Smooth transitions from one gradient background to another to signify a specific action provides a very engaging user experience. (Gradient widget available.)
- **Custom Illustrations** – Quality illustrations (graphics) in a unique style brings a better user experience and can keep the user coming back for more. If the illustrations look interesting, then users will expect to find even more of interest behind the interface. (Flexible support for images in multiple formats. Image compression supported.)
- **Video** – Videos in a format that complements the look and feel of the GUI adds interest and can communicate complex ideas to the user. (Harmony 2 video demonstration examples. To be ported in future Harmony 3 release.)
- **Strong Focus on the User Experience (UX)** – Combine ease of use with a clean and convenient user interface. Don't trade effectiveness for beauty. (Design iterations are fast and easy. User feedback can easily be added to any design.)
- **Interaction** – Each interactive element in the design should provide immediate and smooth feedback to the user. (Event Management tool supports design of effective user interface.)
- **Soothing Colors** – Give preference to a GUI palette with a calm pastel colors and avoid bright colors. (Flexible color schemes can be customized for a desired look and feel.)

Why use MPLAB Harmony Graphics?

Why use MPLAB Harmony Graphics?

Description

Here are the prime reasons in using the MPLAB Harmony Graphics solution:

- **It's Good**
- **It's Fast**
- **It's Free.**

It's Good

MPLAB Harmony Graphics is a good graphics development environment.

Description

The **Screen Designer tool** provides an exact representation of the graphical design, eliminating the Tweak/Generate/Build/Load/Run iterative cycle needed when the physical display was the only way to see the design.

It has the best **multi-language** font and string support of any available graphics suite. It is essential in today's world-wide market that adding **localization features** be straight forward and quick. Strings are stored as an array of pointers to a glyph look up table rather than an array of glyphs, greatly reducing the cost of multi-language support using non-ASCII encoded fonts, such as necessary to support Chinese or other languages with non-Roman alphabets.

A full set of internal tools is provided to manage all the assets needed to build a GUI, including fonts, strings, and images. Images can be imported into the application in one format and stored in an alternate format. Images can be compressed to trade off smaller memory footprint for more image processing.

The memory footprint of all assets can be optimized using the **Memory Configuration tool**. For example, this tool enables applying image compression or format translation to the image assets that use up the most memory. The **Heap Estimator tool** estimates heap usage for all the widgets and image decoders used the application, thereby minimizing wasted memory from a heap allocation that is too big.

Support for multiple graphics **layers** and **double buffering** is built-in to the tool suite and can be enabled using a few mouse clicks. **Animation** using "sprites" is easily accomplished using double buffering. **Video** is also supported. Support for both on-chip and off-chip **display drivers** is provided.

It supports color encoding schemes from 8 bits per pixel using an 8-bit global look up table (LUT) to 32 bits per pixel (32-bit RGBA (Red/Green/Blue/Alpha Blending)). Using the global LUT replaces every pixel color value in frame buffers and images with just one byte that addresses a location in the global palette look up table. The **Global Palette tool** optimizes LUT values based on the images and color schemes used in the design to minimize "posterization". Using the Global Palette LUT allows double buffering on small memory devices, thereby greatly improving graphics performance.

Multiple widget **Color Schemes** are supporting, allowing easy and low overhead customization of the GUI's look.

Critical to building an effective GUI is managing application events, such as touches or swipes. The Event Manager tool supports events within the graphics stack and between the graphics stack and the application layer. In many cases, especially for events from one graphics widget to another, the **Event Manager tool** eliminates the need to write any code. In most other cases an event code template is built by the Event Manager that can be customized to fully implement the required GUI behavior.

MPLAB Harmony Graphics provides a full set of graphical design elements (aka Widgets):

Widget	Description	Application Example
Arc	A graphical object in the shape of an arc.	aria_showcase_reloaded
Bar Graph	A graphing widget that shows data in categories using rectangular bars.	aria_showcase_reloaded, aria_weather_forecast
Button	A selection box with Checked and Unchecked states, plus associated events.	aria_benchmark, aria_showcase, aria_quickstart, aria_weather_forecast
Check Box	A selection box with Checked and Unchecked states, plus associated events.	aria_showcase_reloaded
Circle	A graphical object in the shape of a circle.	
Circular Gauge	A circular widget that operates like a gauge, where the hand/needle position indicates a value.	aria_showcase_reloaded
Circular Slider	A circular widget that can change values based on external input like touch.	aria_showcase_reloaded
Draw Surface	A container with a callback from its paint loop. A Draw Surface lets the application have a chance to make draw calls directly to the HAL during LibAria's paint loop.	
Gradient	A draw window that can be associated with a gradient color scheme. This allows for color variation on the window.	aria_showcase (background)
Group Box	A container with a border and a text title. With respect to functionality, a group box is similar to a window.	
Image	Allows an image to be displayed on screen.	aria_benchmark, aria_showcase, aria_quickstart, aria_weather_forecast
Image Plus	An improved Image widget. The image can be resized (aspect ratio lock is optional). The widget can be set to accept two-finger touch input.	
Image Sequence	A special widget that allows image display on screen to be scheduled and sequenced using a timer.	aria_showcase
Key Pad	A key entry widget that can be designed for the number of entries divided as specified number of rows and column entries. The widget has a key click event that can be customized.	aria_showcase

Label	A text display widget. A Text Field widget should be used instead when user input is needed.	aria_benchmark, aria_showcase, aria_quickstart, aria_weather_forecast
Line	The name says it all.	
Line Graph	A graphing widget that shows data in categories using points and lines.	aria_showcase_reloaded, aria_weather_forecast
List Wheel	Allows multiple radial selections that were usually touch-based selections and browsing.	aria_showcase
List	Allows making lists of text and image items. The list contents, number of items, and the sequence can be managed through its properties.	
Panel	A container widget that is a simpler alternative to DrawSurface as it does not have the DrawSurface callback feature.	aria_benchmark, aria_showcase
Pie Chart	A graphing widget that shows data entries as sectors in a circle.	aria_showcase_reloaded
Progress Bar	Displays the progress pointer for an event being monitored through the "Value Changed" event in the Properties Editor.	
Radial Menu	A set of button widgets grouped together. Only one button in a group can be active.	aria_showcase
Radio Button	A set of button widgets grouped together. Only one button in a group can be active.	aria_showcase
Rectangle	The name says it all.	
Scroll Bar	Intended to be used with another relevant widget such as the List Wheel to scroll up and down. It has a callback each time the value is changed. The callback allows users to trigger actions to be handled on the scroll value change event.	
Slider	Can change values with an external input such as touch. Event callbacks on value change are also available through the Properties Editor.	aria_showcase
Text Field	Text input can be accepted into the text field from an external input or from a widget such as keypad.	aria_showcase
Touch Test	Allows tracking of touch inputs. Each new touch input is added to the list of displayed touch coordinates.	aria_showcase
Window	A container widget like the Panel but has the customizable title bar.	

It's Fast

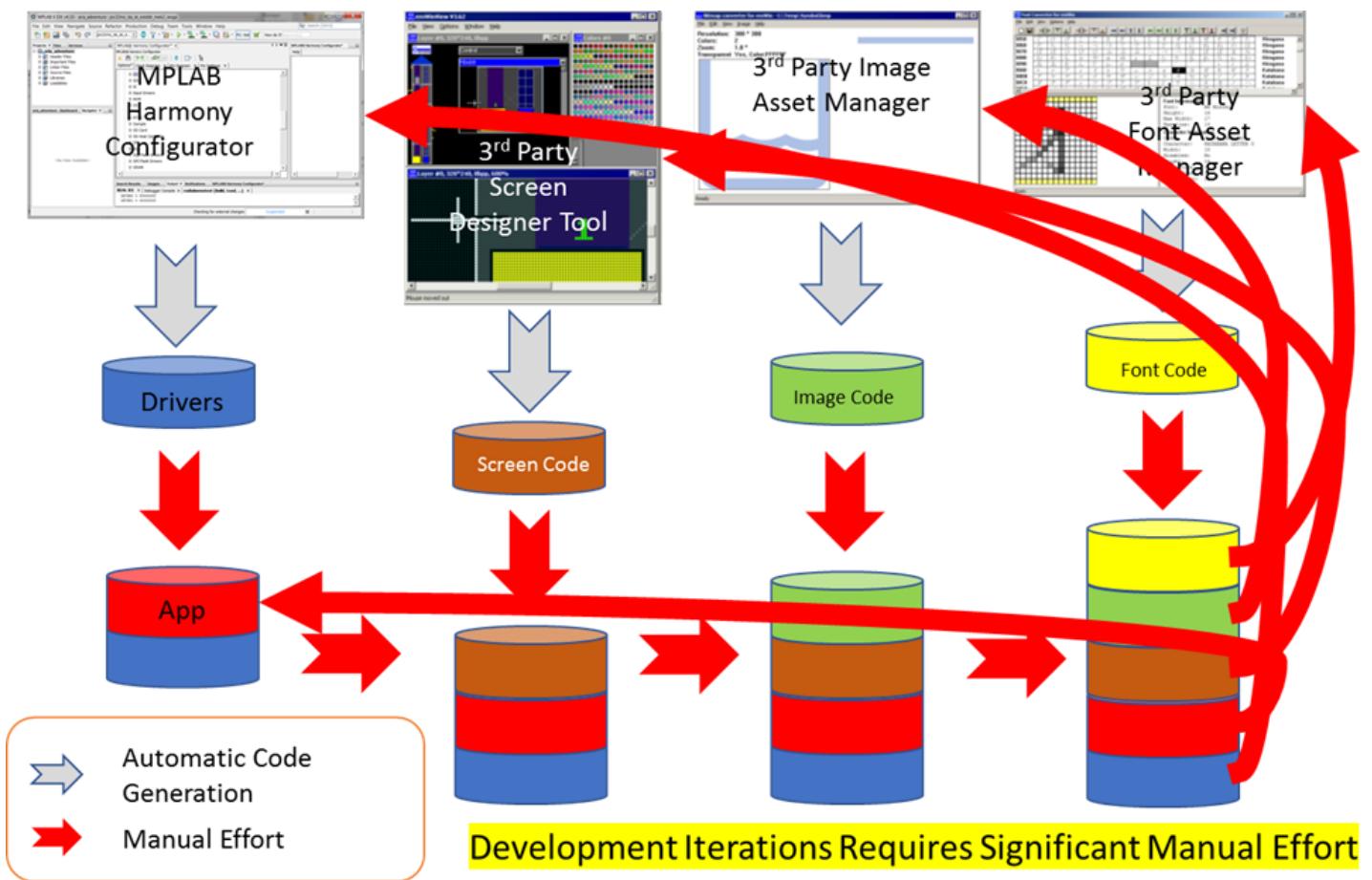
It's fast to market, minimizing development time. It produces very fast graphics on processors.

Description

It produces fast graphics on target (PIC32 or SAM) processors, minimizing the processor resources and processing bandwidth dedicated to graphics. This provides smaller footprint designs and lowers BOM costs.

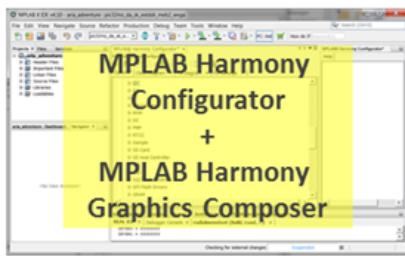
Unlike 3rd party graphics tools, it is deeply embedded and tightly coupled with the existing development tools (MPLAB X, Harmony, MPLAB Harmony Configurator).

With 3rd party tools each step in the graphics development is outside of the development environment (e.g. MPLAB X) and the code must be assembled manually:



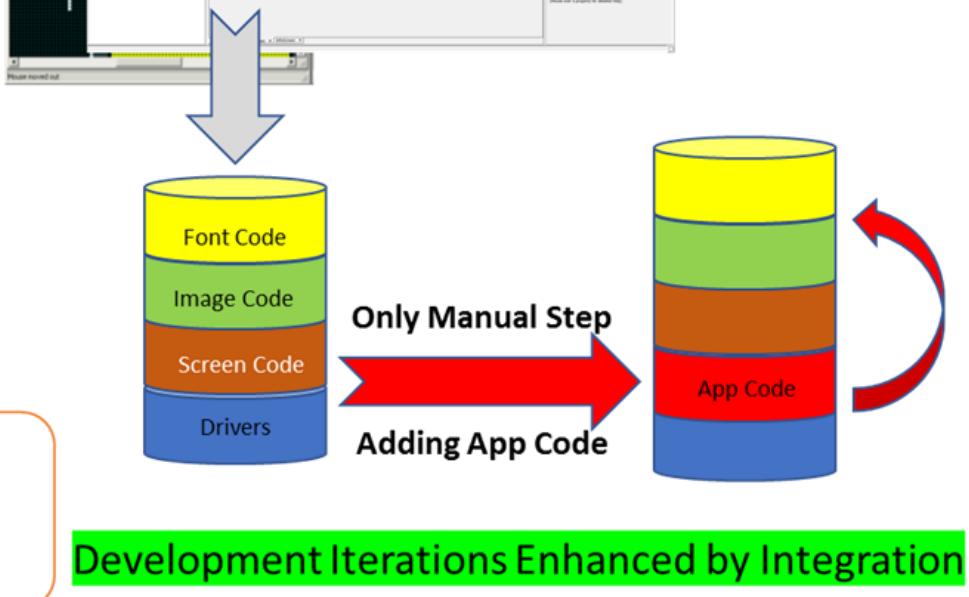
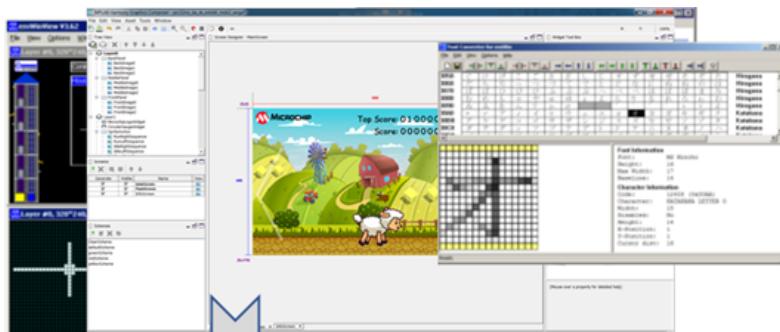
This process takes more time and is subject to user error because of the manual assembly of all the code developed by the tools outside of MPLAB X. Typically, the development of a graphical user interface (GUI) is a highly iterative process, requiring many loops through the process shown above before converging to the final GUI design. The faster the loop time of each iteration the faster the final design.

All MPLAB Harmony Graphics tools are tightly integrated into the MPLAB X/Harmony/MPLAB Harmony Configurator tool suite. The only place for manual software changes is in the application code of the project, everything else is automated and free from user error:



- ✓ Drivers
- ✓ System Services
- ✓ Graphics Framework
- ✓ Image Resources
- ✓ Font Resources
- ✓ Input Events
- ✓ External Events

- Automatic Code Generation
 Manual Development



The Aria Graphics Library is optimized to exploit MPLAB Harmony Graphics capabilities. It's not a one-design-fits-all processor library, it is focused solely on PIC32 or SAM. Thus, it can take full advantage of PIC32 or SAM features, such as the 2D Graphics Processing Unit (2D GPU) and built-in display controller when these features are available in the target processor.

The available graphics application examples provide optimized graphics designs that wring the maximum graphics performance out of each target processor. **Advanced Topics** in the documentation discuss additional performance-enhancing techniques, such as "Draw Pipeline Options" and "Improving Touch Performance with Phantom Buttons".

It's Free

No up front development costs or royalties!

Description

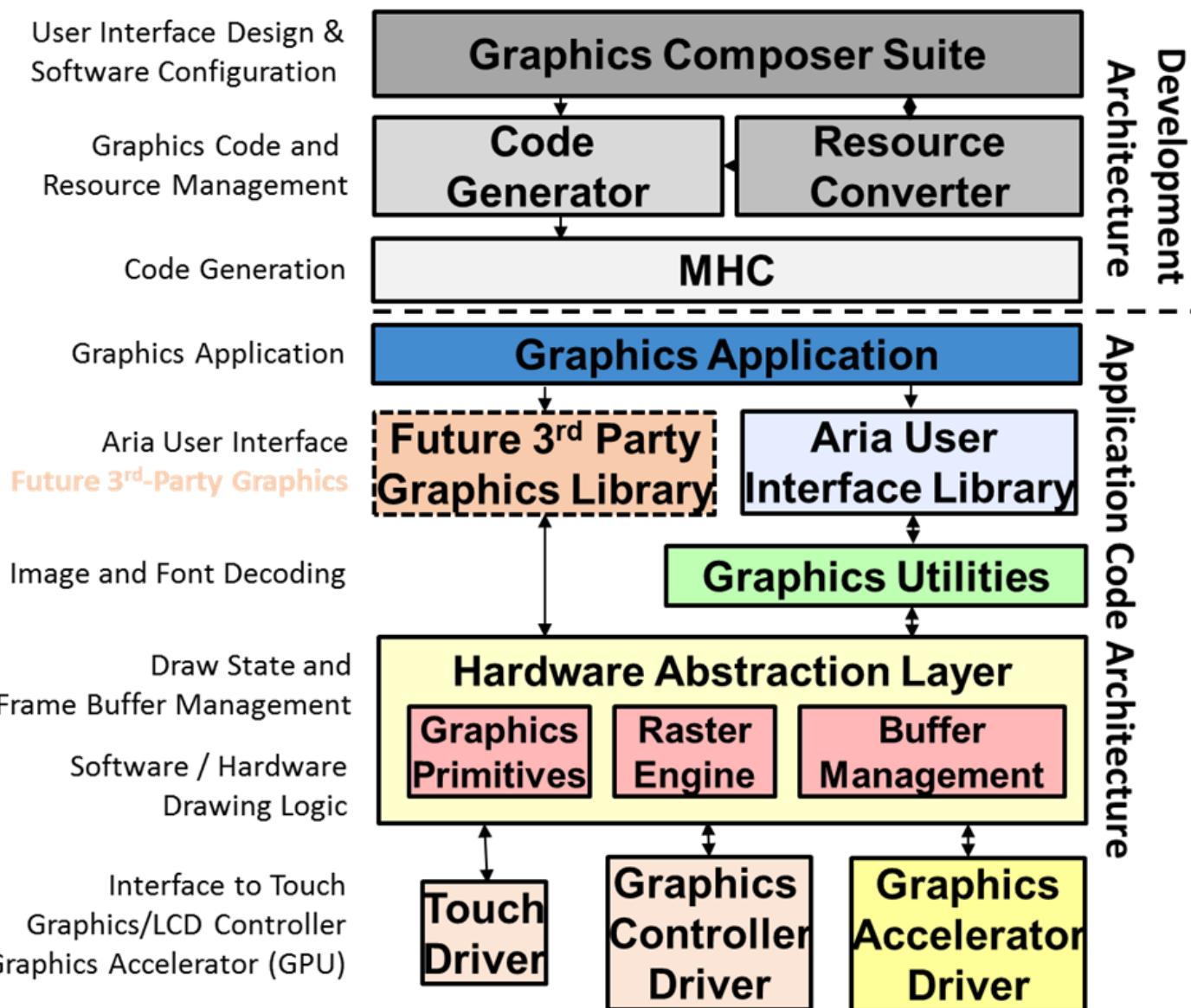
There are no up-front costs and MPLAB Harmony Graphics is royalty-free for use in all PIC32 products. Full source code is available in each release of MPLAB Harmony. There are no restrictions on user customization of this source code. Finally, it works with the free version of the XC32 compiler so no additional investment in compiler upgrades is needed.

How Does MPLAB Harmony Graphics Work?

A brief description of how MPLAB Harmony Graphics works.

Description

Conceptually, MPLAB Harmony graphics can be partitioned into two parts: the development architecture and the application code architecture:



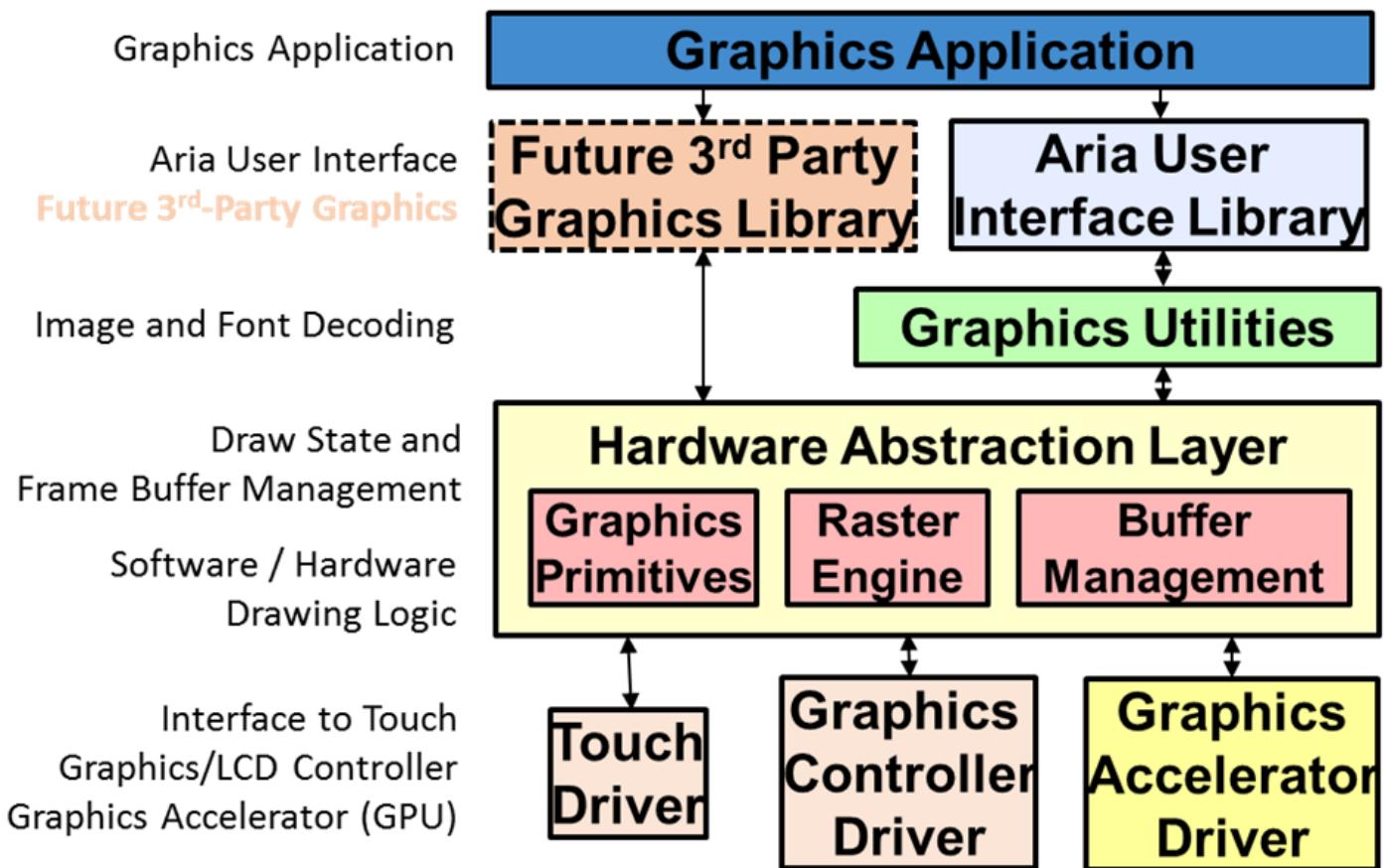
During graphics development the **MPLAB Harmony Graphics Composer** (MHGC) suite is launched from the MPLAB Harmony Configurator (MHC). MHC supports the basic setup of the application's code, including system clock setup, selection and configuration of drivers, and display setup (displays and display drivers). MHGC supports the design of the application's graphics, including asset (fonts, strings, images) management. The setup of color schemes and widgets is supported by MHGC.

The MPLAB Harmony Graphics Composer (MHGC) supports designing a graphical user interface (GUI) using a graphical drag and drop interface in the Screen Designer tool. MHGC configures all the code needed to initialize, configure, and manage an Aria library context. Then MHC can generate the code.

MHGC tools include:

- **WYSIWYG GUI editor** – Enables drag and drop capability to visualize the design.
- **Tree Manager** – Enables the user to select the drawing priority and establish parent / child relationships so that objects (widgets) can be grouped as desired.
- **Event Manager** – Enables the user to customize the experience of touch and logical (application) events and to interact with graphical attributes.
- **String and Font Managers** – Used to input strings in multiple languages for potential reuse, and optimization of fonts and memory requirements.
- **Image Manager** - 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's Flash memory.

Application Code Architecture



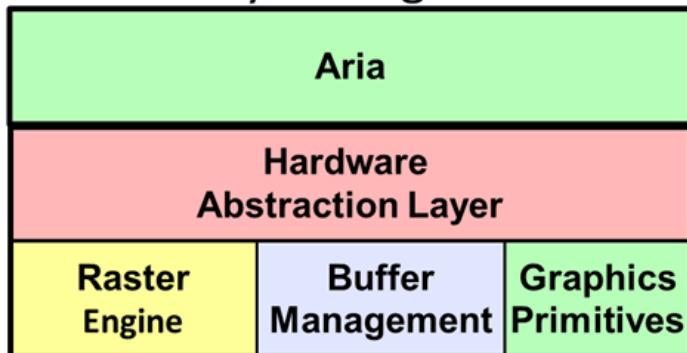
Aria User Interface Library – This library provides the capability for interface generation, management, and interaction. It provides the building blocks for constructing a user interface in the form of “Widgets” or user interface elements. These consist of things like buttons, check-boxes, images, etc. This library also handles user interaction events for things like touch actions.

For more information, see MPLAB Harmony Composer Suite > Aria User Interface Library.

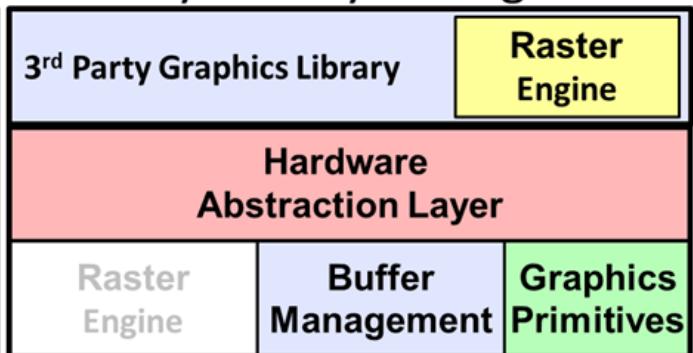
Graphics Utilities Library – This library is primarily responsible for managing and decoding assets such as images, fonts, and strings. It provides the means for interacting with asset data, complex data decoding, data decompression, and string asset look-up. It also abstractly handles accessing external memory sources during asset decoding.

Hardware Abstraction Layer (HAL) – Provides a standard software interface between the Aria User Interface Library (or a future 3rd Party Graphics Library) and graphics hardware (Display Driver and Graphics Processing Unit (GPU)). The HAL supports graphics buffer management, the graphics render pipeline, and graphics primitive algorithms.

Aria Library Configuration



3rd Party Library Configuration



The HAL supports a seamless integration of a Graphics Processing Unit (GPU) by via a standardized application interface that can switch between using a GPU for a hardware implementation of graphics primitives and a software implementation of the same primitives.

For more information, see [MPLAB Harmony Graphics Composer Suite >Hardware Abstraction Layer \(HAL\)](#).

Third Party Graphics Library (Future Upgrade) – 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.

Touch Driver – Supports capacitive or resistive touch interface on the display.

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. (*If available.*)

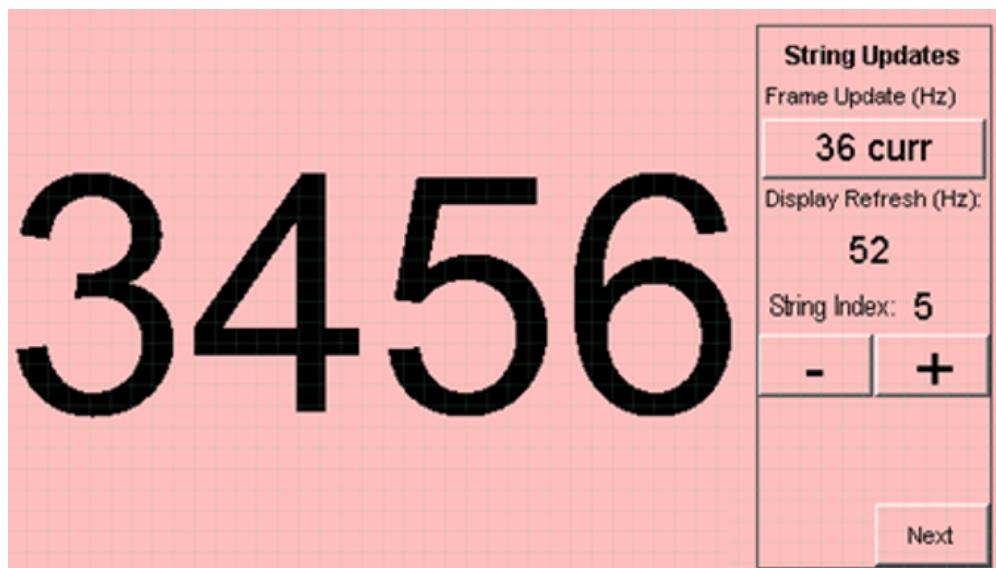
Example Graphics Projects

Example projects showing MPLAB Harmony Graphics in action.

Description

The aria_benchmark Demonstration

This application presents frame update rate metrics on the various rendering operations in the Harmony graphics library.



The aria_showcase Demonstration

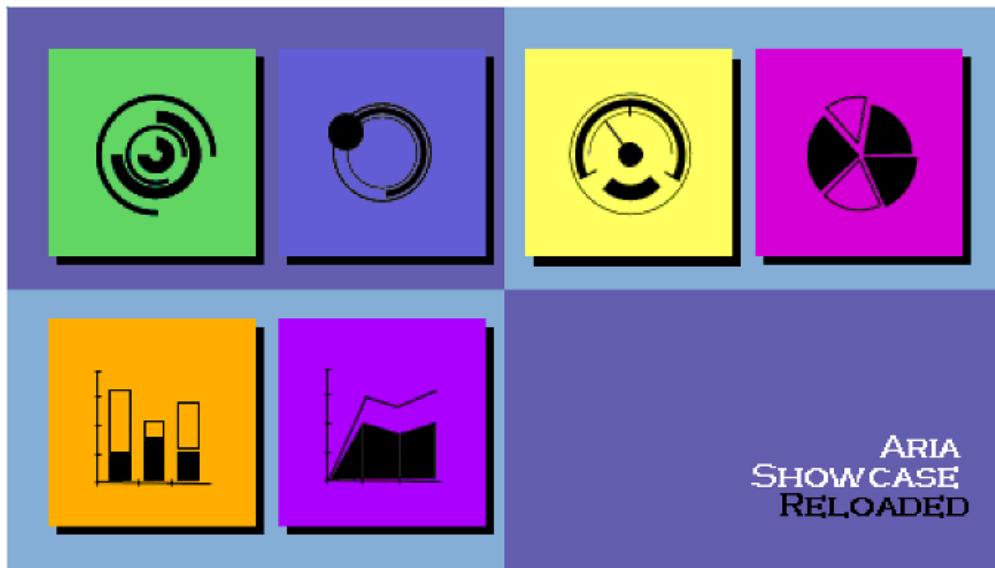
This demonstration provides a subset of capabilities offered by the Aria Graphics Library using Low-Cost Controllerless (LCC)

driver features with touch screen capabilities.



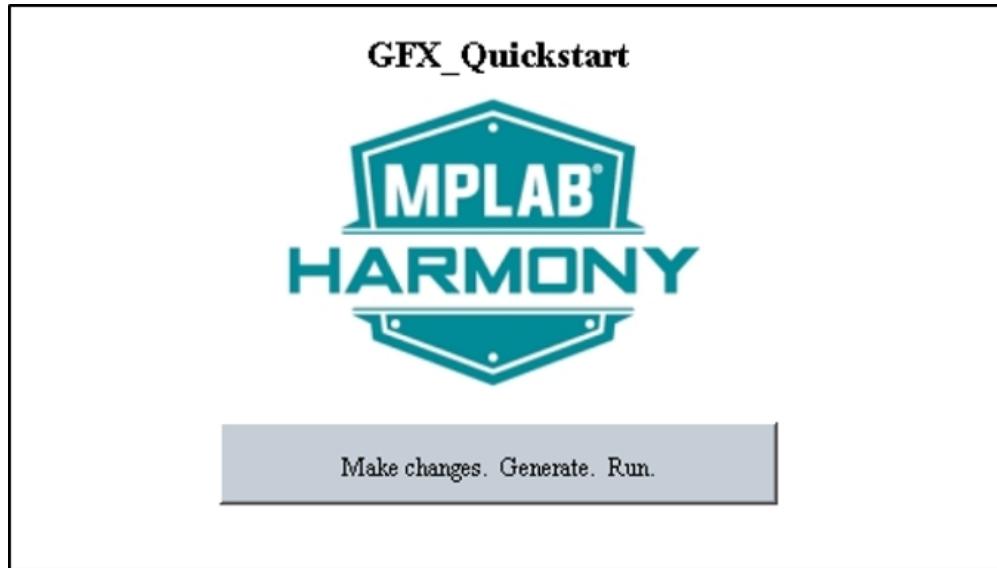
The [aria_showcase_reloaded](#) Demonstration

This application showcases the circular and graphing widgets – arc, circular slider, circular gauge, pie chart, bar graph, line graph. These widgets are available and ready-to-use using the Harmony Graphics Composer suite.



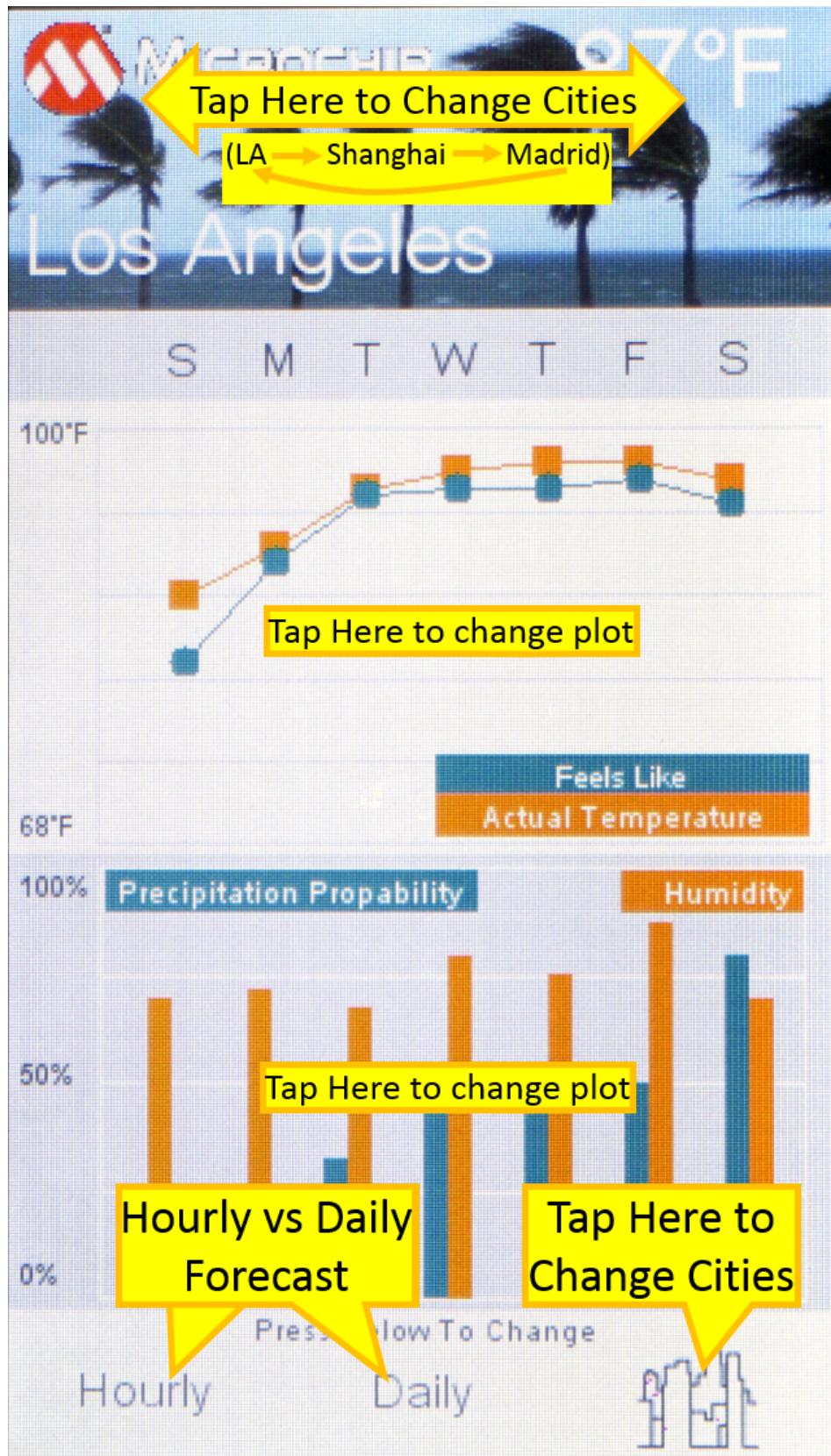
The [aria_quickstart](#) Demonstration

This demonstration provides a touch-enabled starting point for Aria Graphics development.



The [aria_weather_forecast](#) Demonstration

This demonstration provides a practical single-layered single-buffered application using the Aria User Interface Library.



Next Steps

Recommendations for what to do next.

Description

Here are some recommendations for what to do next:

- Learn how to create a Harmony 3 compliant project from scratch. (See [MPLAB Harmony Core Help > Creating Your First Project in Harmony](#))
- Learn how to create a Harmony 3 compliant graphics project from scratch. (See [MPLAB Harmony Graphics Library Help > Quick Start Guides > Creating New Graphics Applications](#))

Quick Start Guides

This topic provides brief user material to enable users to get started quickly on a variety of topics.

Creating New Graphics Applications

Covers the basic steps to create a graphics-enabled application from scratch, starting with a blank MPLAB project and finishing with a graphics application equivalent to aria_quickstart.

Description

This tutorial provides information on how to create a new touch input-enabled graphics project using the MPLAB Harmony Graphics Composer (MHGC). The following hardware setups are supported:

- SAM E70 Xplained Ultra board (DM320113) with High-Performance 4.3" WQVGA Display Module with maXTouch (AC320005-4) Display (Plus ICD4 In-Circuit Debugger (dv164045), ICD4/PICKit Adapter Board (ac102015))

Getting Started

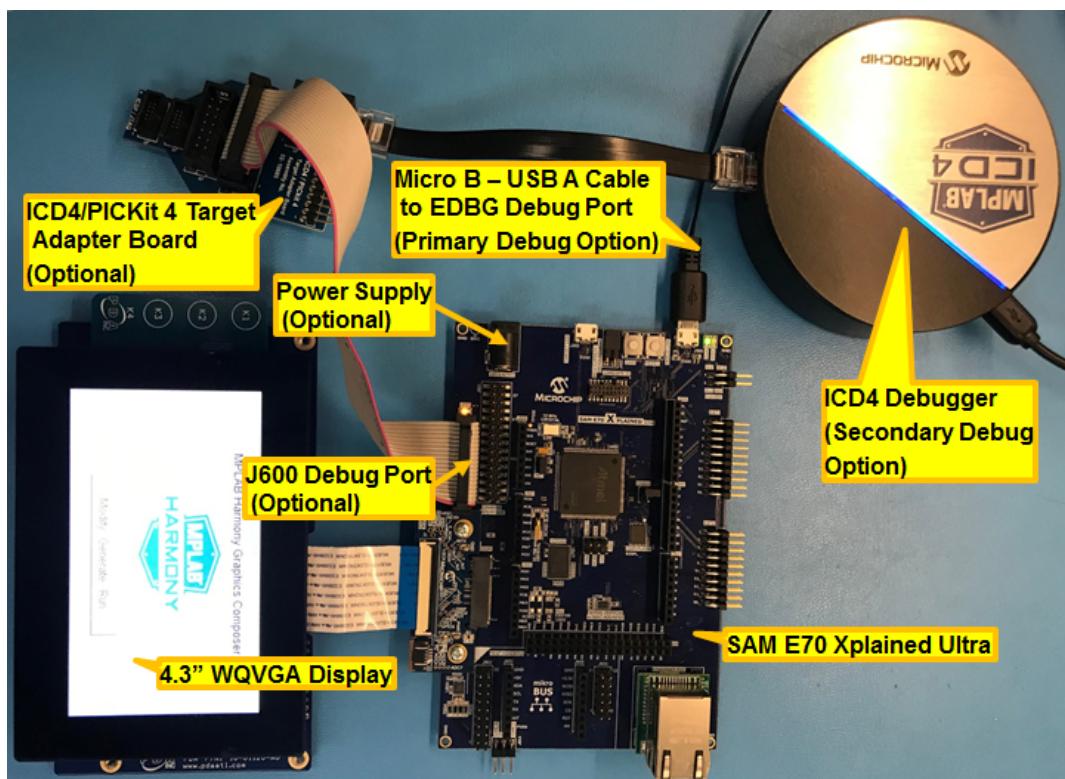
Before beginning this tutorial, ensure that the MPLAB X IDE is installed and necessary language tools as described in Getting Started With MPLAB Harmony. In addition, ensure that the MPLAB Harmony framework is installed on a local hard drive and that the correct version of MHC plug-in is installed within the MPLAB X IDE.

Follow the tutorial found in MPLAB Harmony Core Library Help > Creating Your First Project to first learn about the basics of software development using MPLAB Harmony. Successfully completing the tutorial can serve as a confirmation of the software setup.

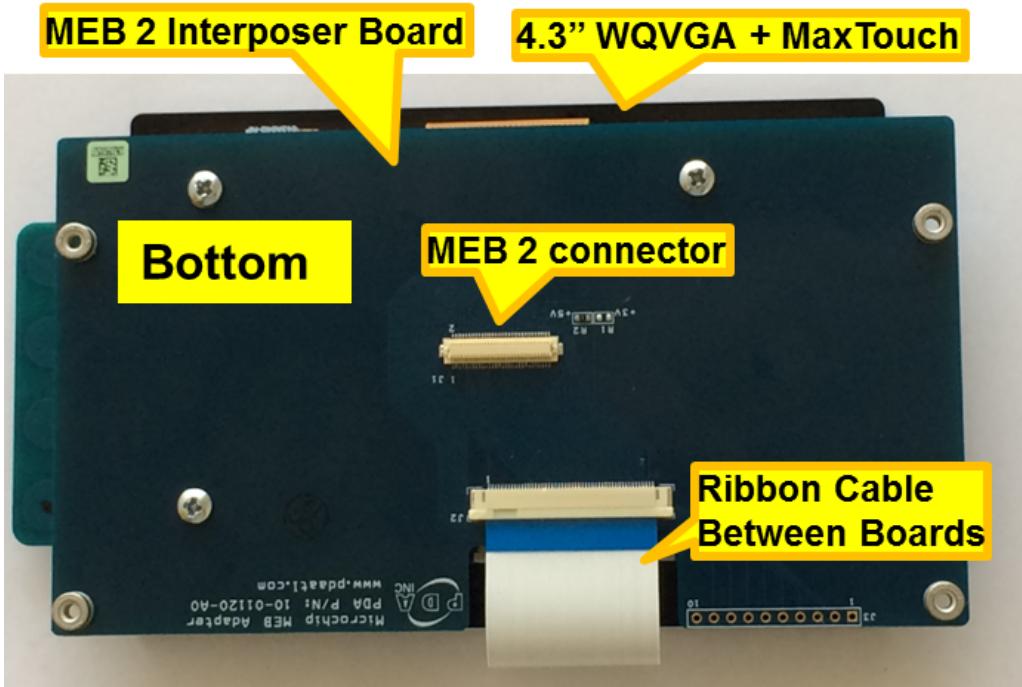
Configuring the Hardware

SAM E70 Xplained Ultra board with 4.3" WQVGA Display

There are two programming options. The primary option is the Micro B Embedded Debugger (EDBG) port. Power to the board can also be supplied via this connection. The secondary option is via the ICD4 debugger with the ICD4/PICKit 4 Target Adapter Board (Power has to be provided from power supply in this alternative setup). The image below shows both the EDBG and the ICD4 connection options:

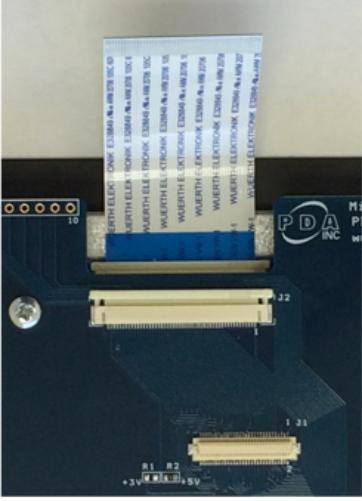


Configuring the 4.3" WQVGA Display requires disconnecting the ribbon cable that connects the display to the interposer board.



First, release the ribbon cable from the interposer board. Next, release the black clamp on the E70's J2 connector and turn the display over. Finally, insert the ribbon cable into J2 and close the clamp.

Step 1: Release Ribbon Cable on Interposer Board



Step 2: Release clamp on J2 and turn display over



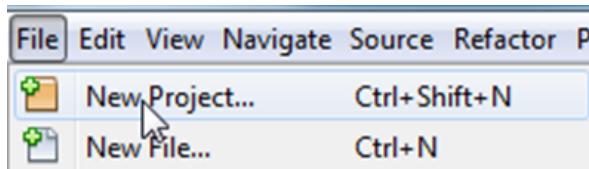
Step 3: Insert Cable, close clamp



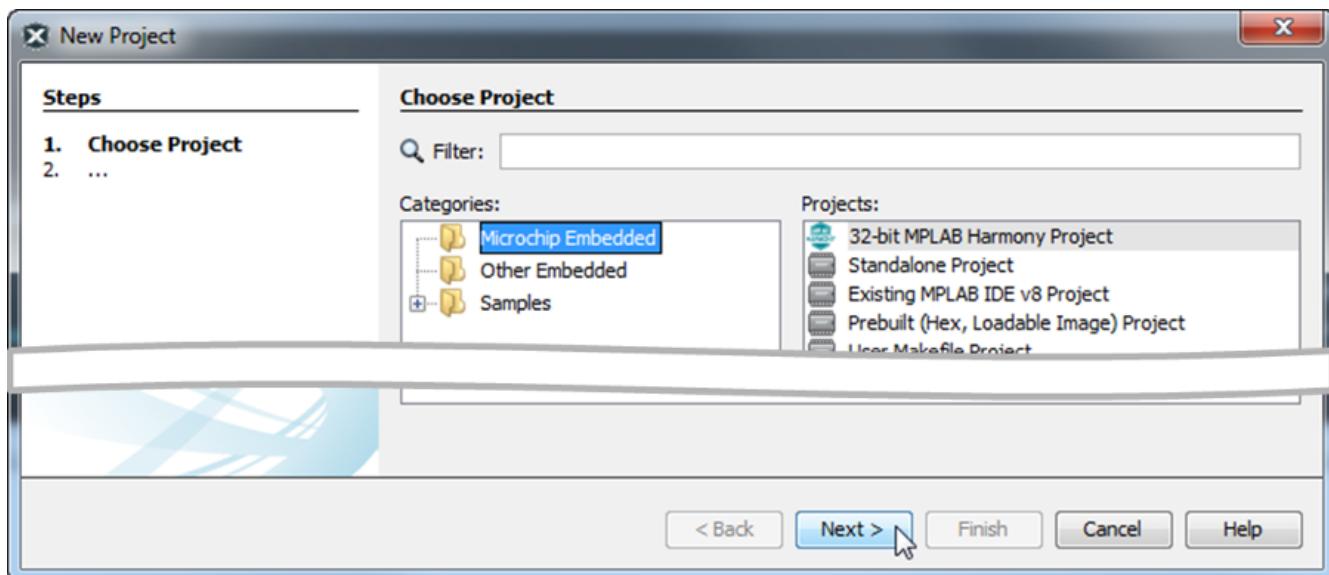
The board and display are powered by a Micro B – USB A cable from PC to the "Debug USB" port on the E70 board. The ICD4 Debugger and ICD4/PICKit4 Adapter Board are connected as shown above.

Tutorial Steps

1. Launch the MPLAB X IDE. From the File pull-down menu, select New Project. (This will bring up the New Project dialog window.)



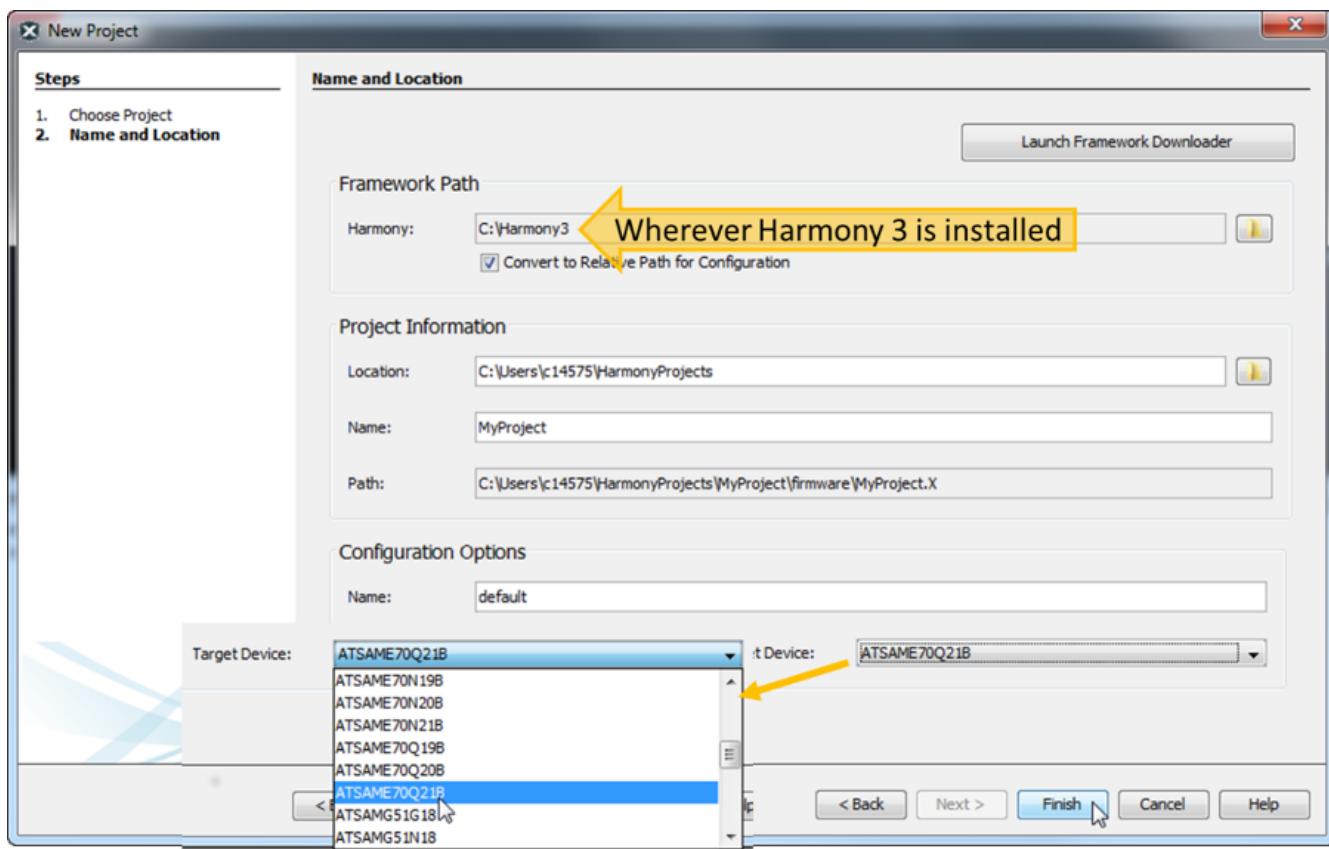
2. On the New Project dialog window, be sure the project type is 32-bit MPLAB Harmony Project, then hit the **Next >** button.



3. In the New Project dialog window, fill in or select the information needed as follows:

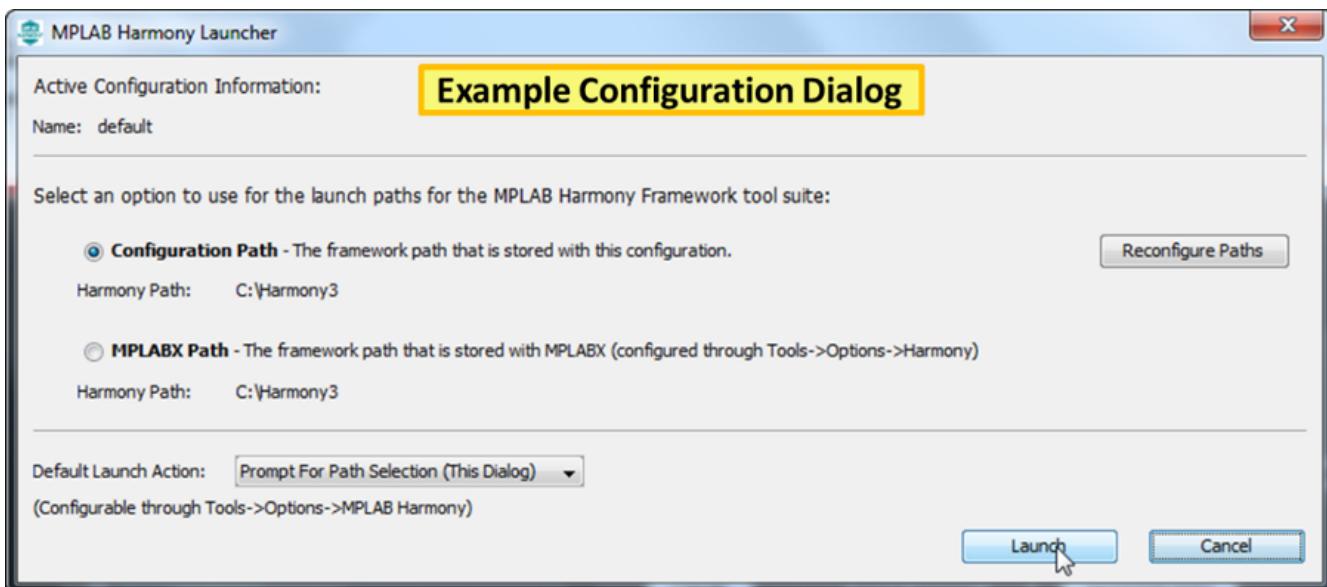
- **Harmony Path** - be sure it is pointing to the MPLAB Harmony framework installation.
- **Project Name** - Type in MyProject or some other name.
- For the SAM E70 Xplained Ultra board, select the ATSAME70Q21B as the target device.

The New Project dialog should show the following:



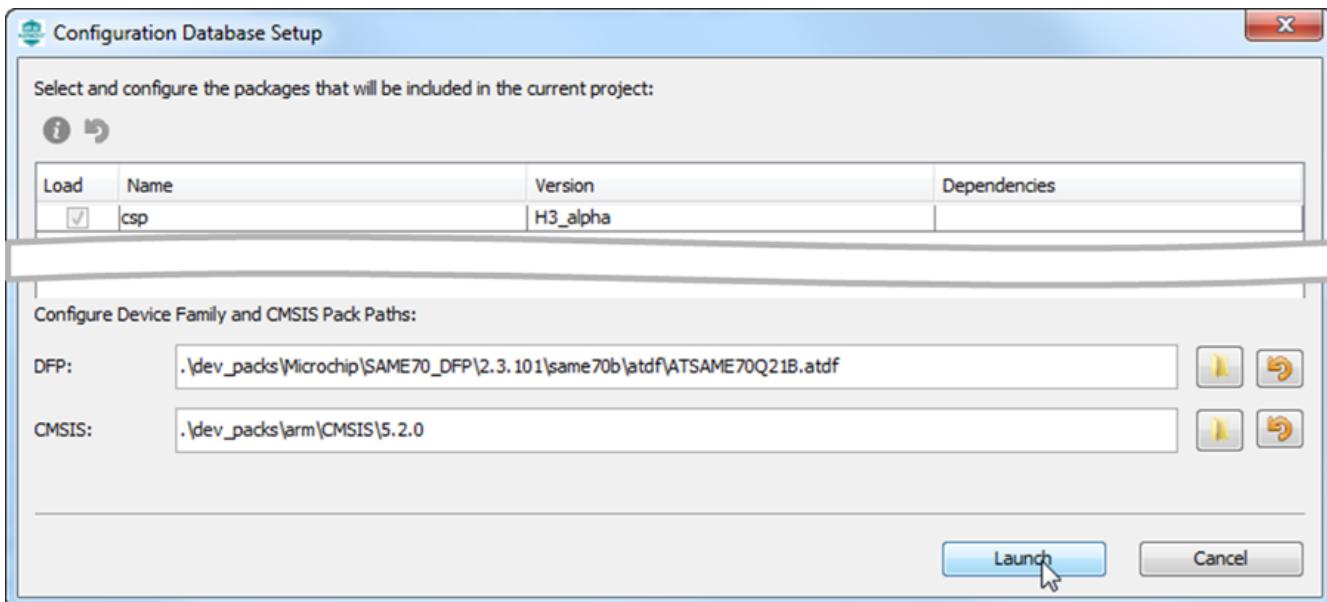
Select **Finish** when done.

4. In the MPLAB Harmony Launcher dialog window select the Launch Path that corresponds to the Harmony installation to be used. In the example below, both MPLAB X IDE's default configuration and the project's path point to the same MPLAB Harmony installation:

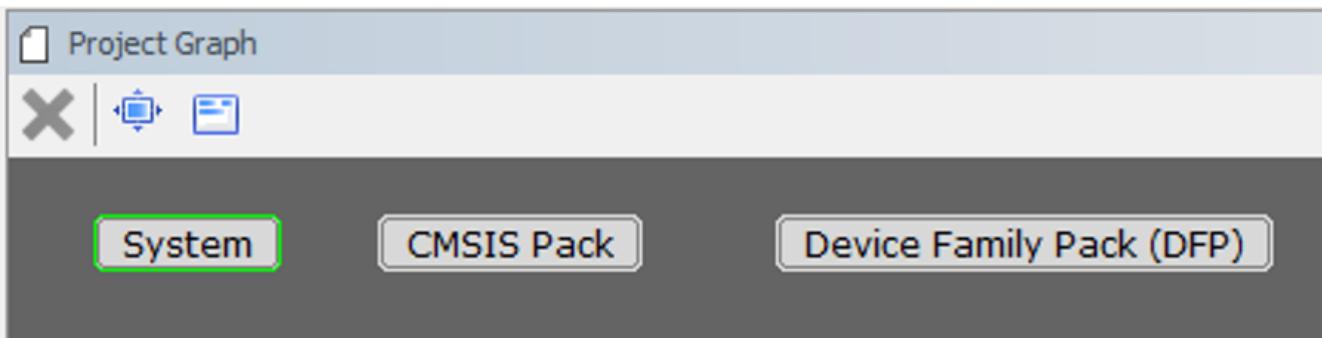


The select the **Launch** button.

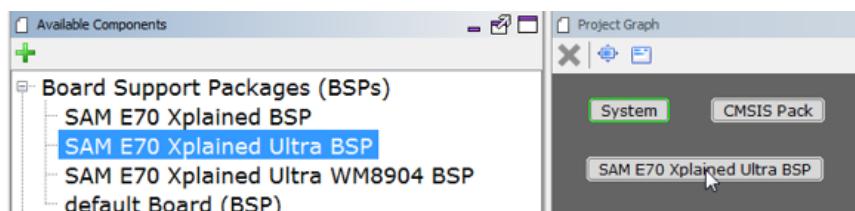
When the "Configuration Database Setup" dialog window appears, just select **Launch**:



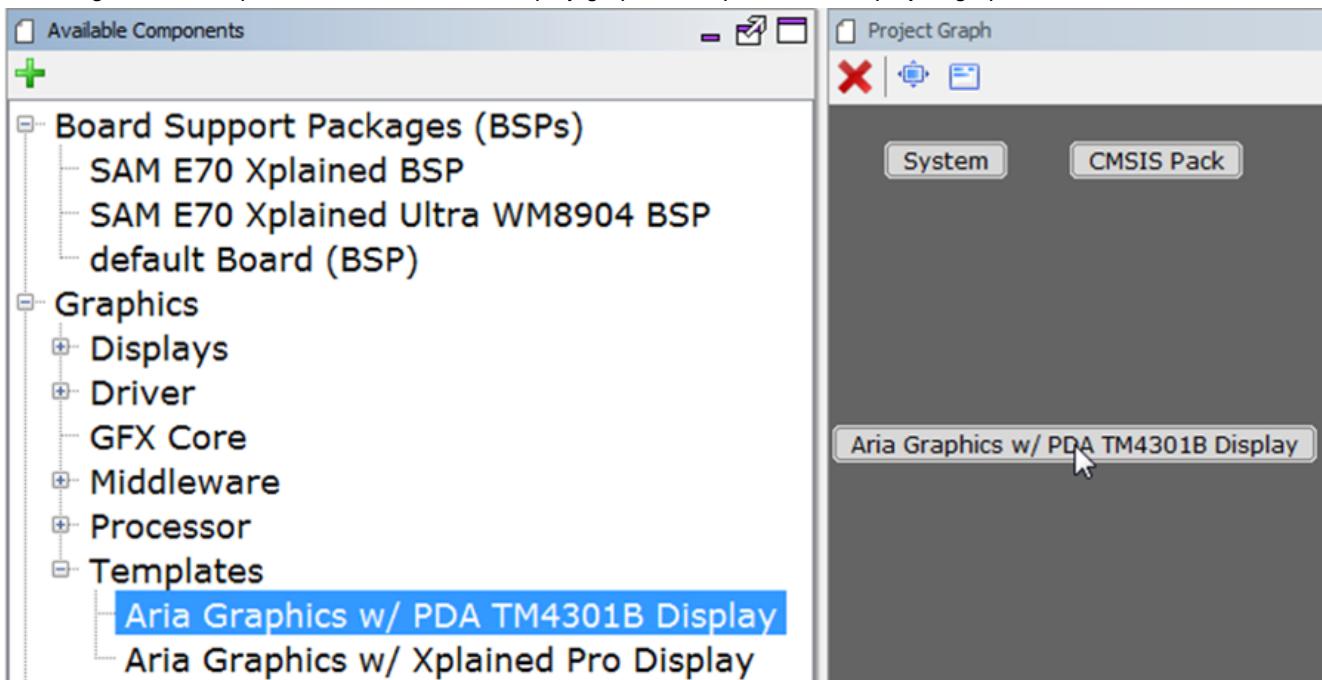
MHC's component graph should show:



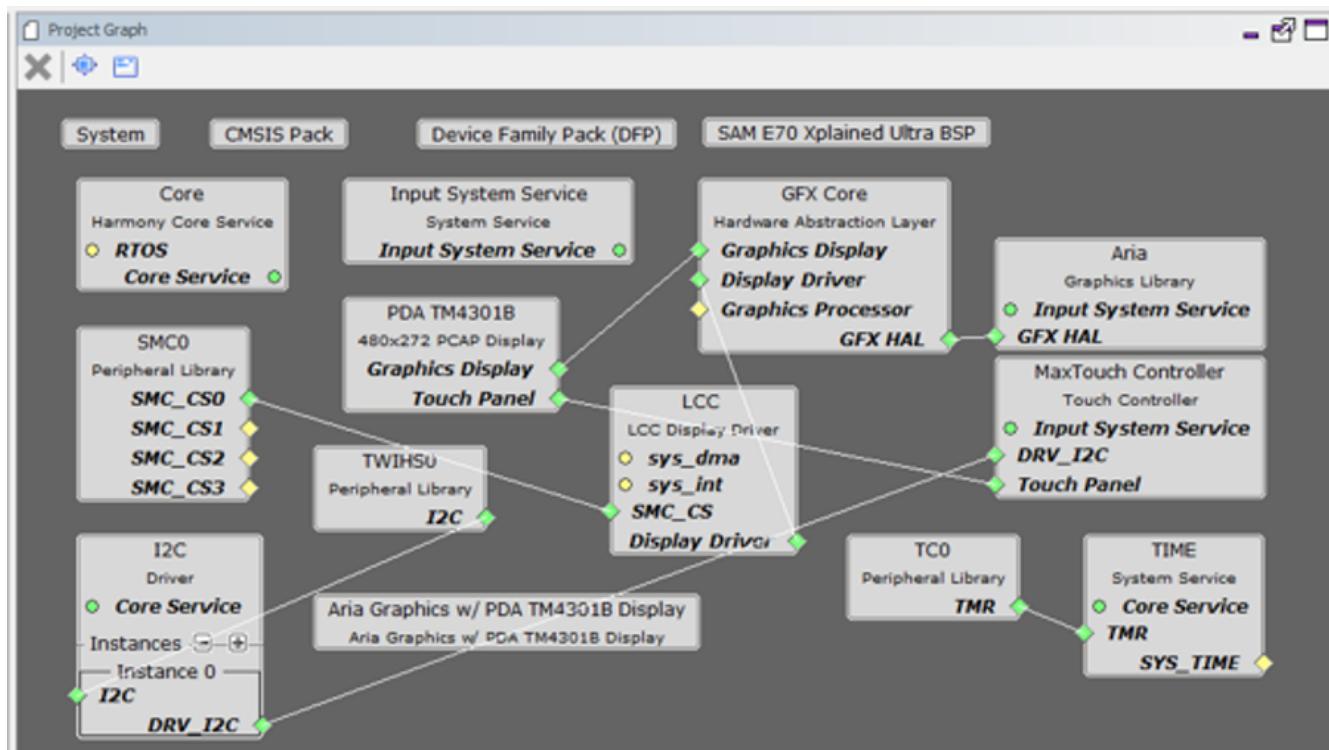
5. In the MHC's Available Components panel drag and drop the SAM E70 Xplained Ultra BSP onto the project's Component Graph



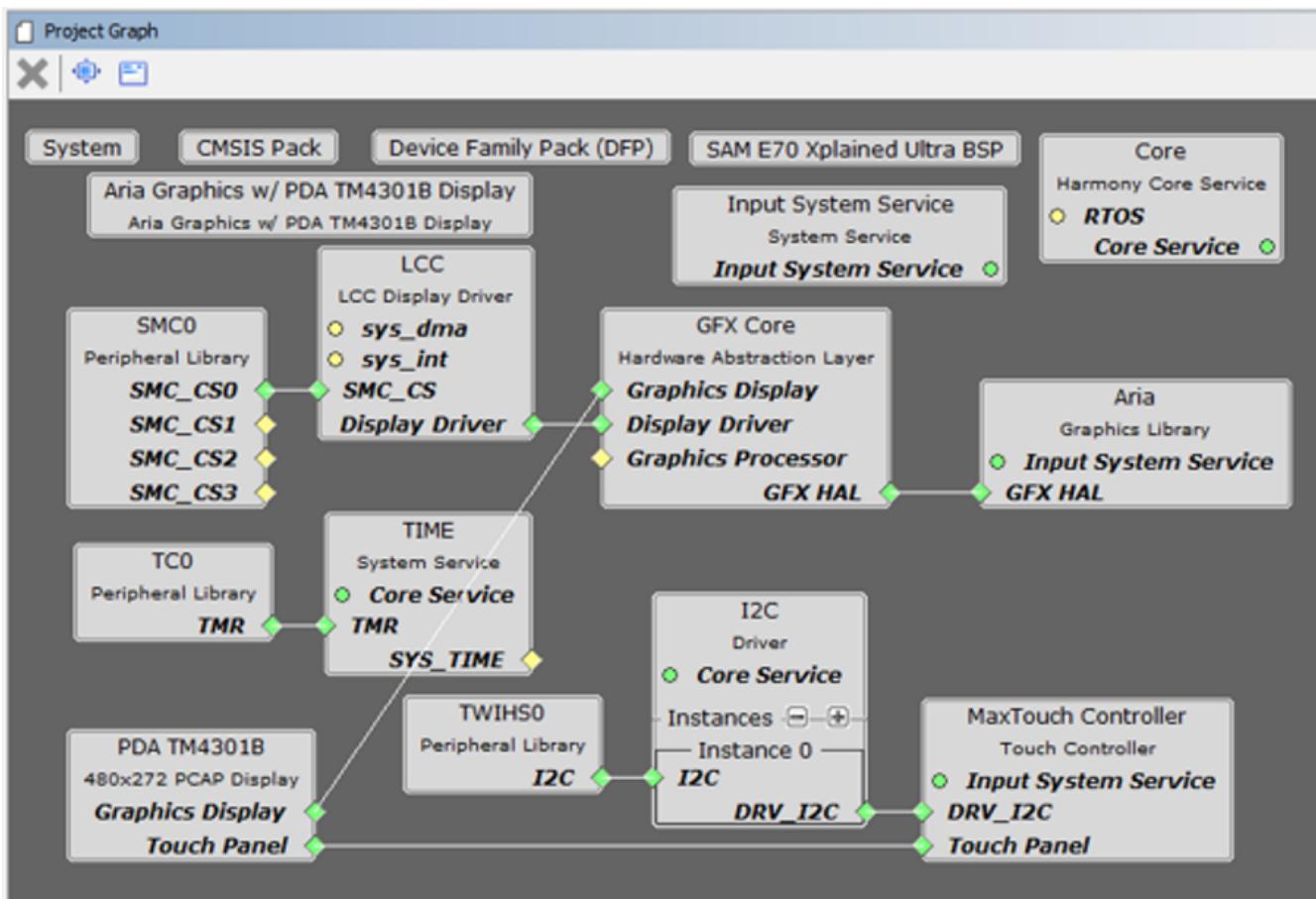
6. Next drag the Aria Graphics w/ PDA TM4301B Display graphics template onto the project graph:



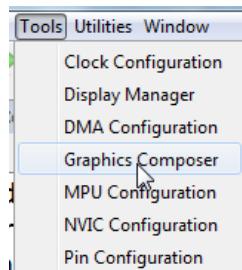
Then answer **Yes** to all the dialogs that follow. The Project Graph should show:



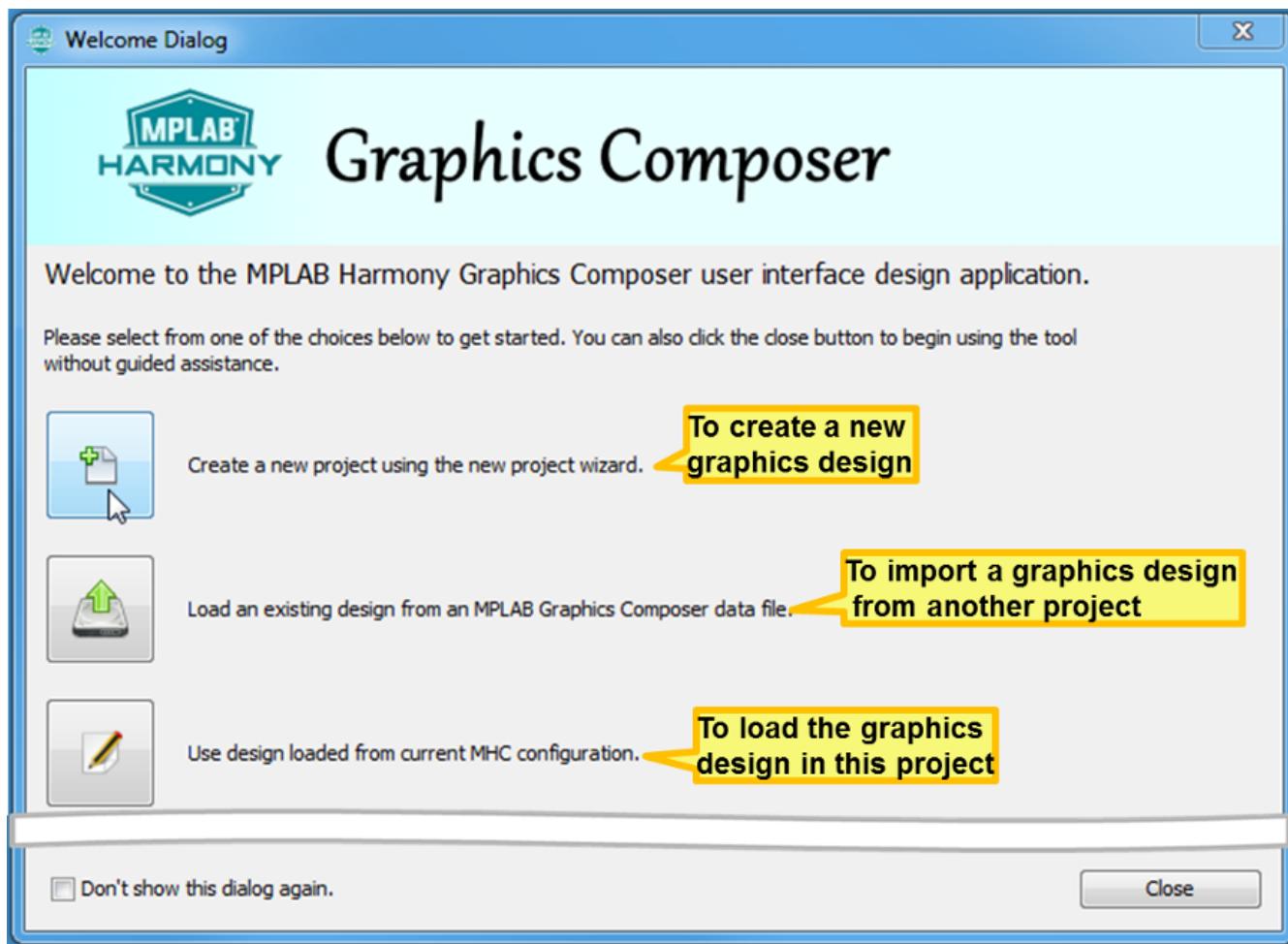
After rearranging the components:



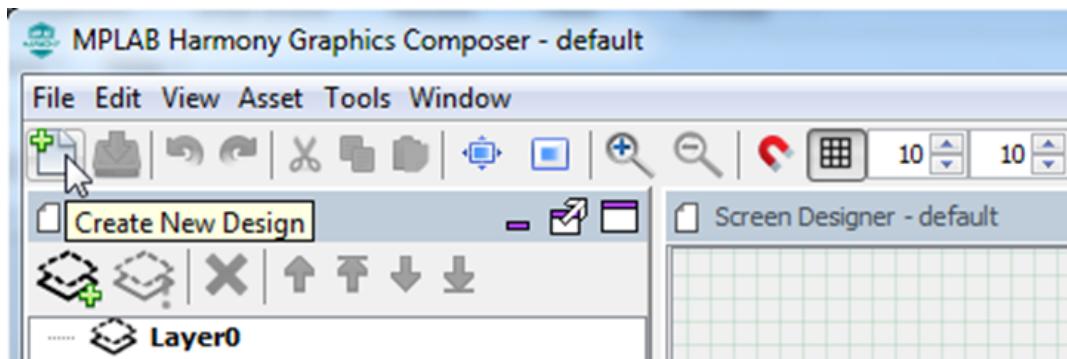
7. Launch the Graphics Composer from the MHC Tools Menu:

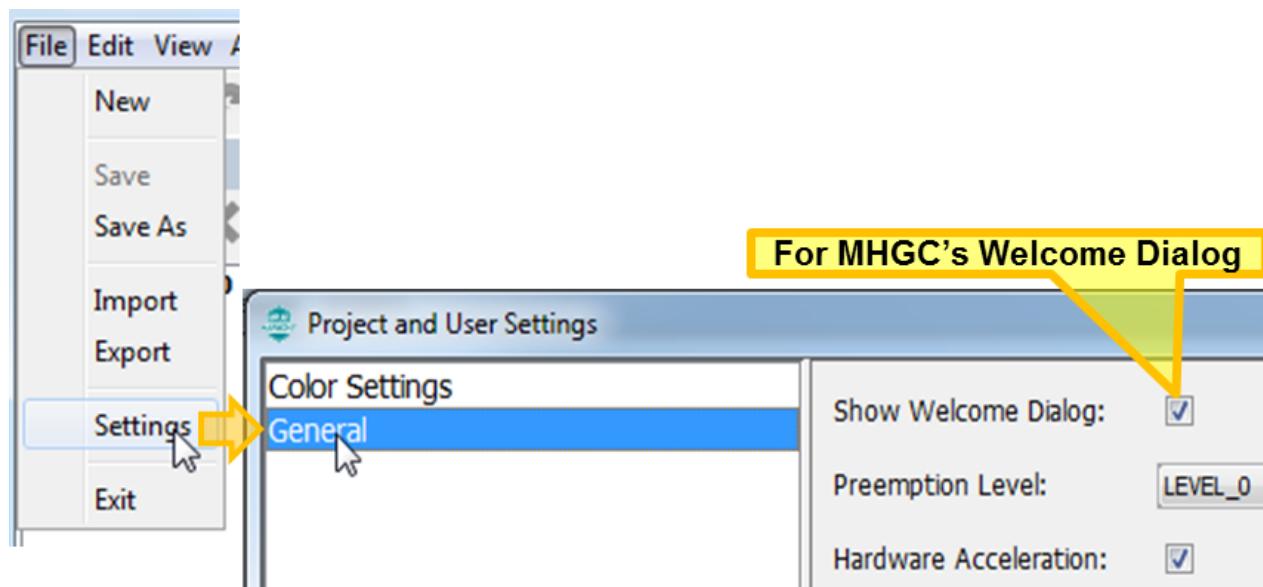


Next, the MHGC's Welcome Dialog should appear. Click the first icon to create a new graphics design.

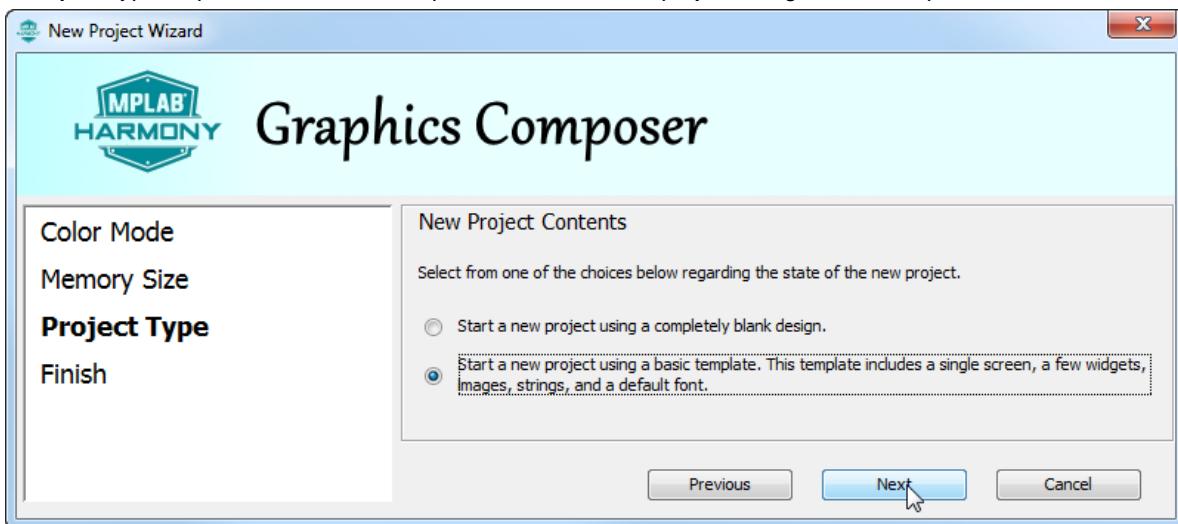


If the Welcome Dialog does not appear, it is because it had been disabled previously. In the MPLAB Harmony Graphics Composer (MHGC) screen use the left-most icon to create a new graphics design.

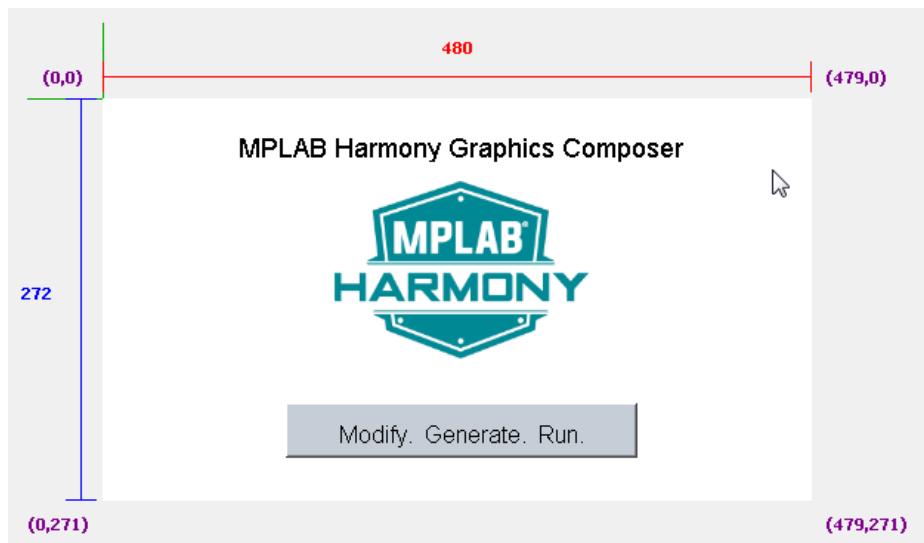




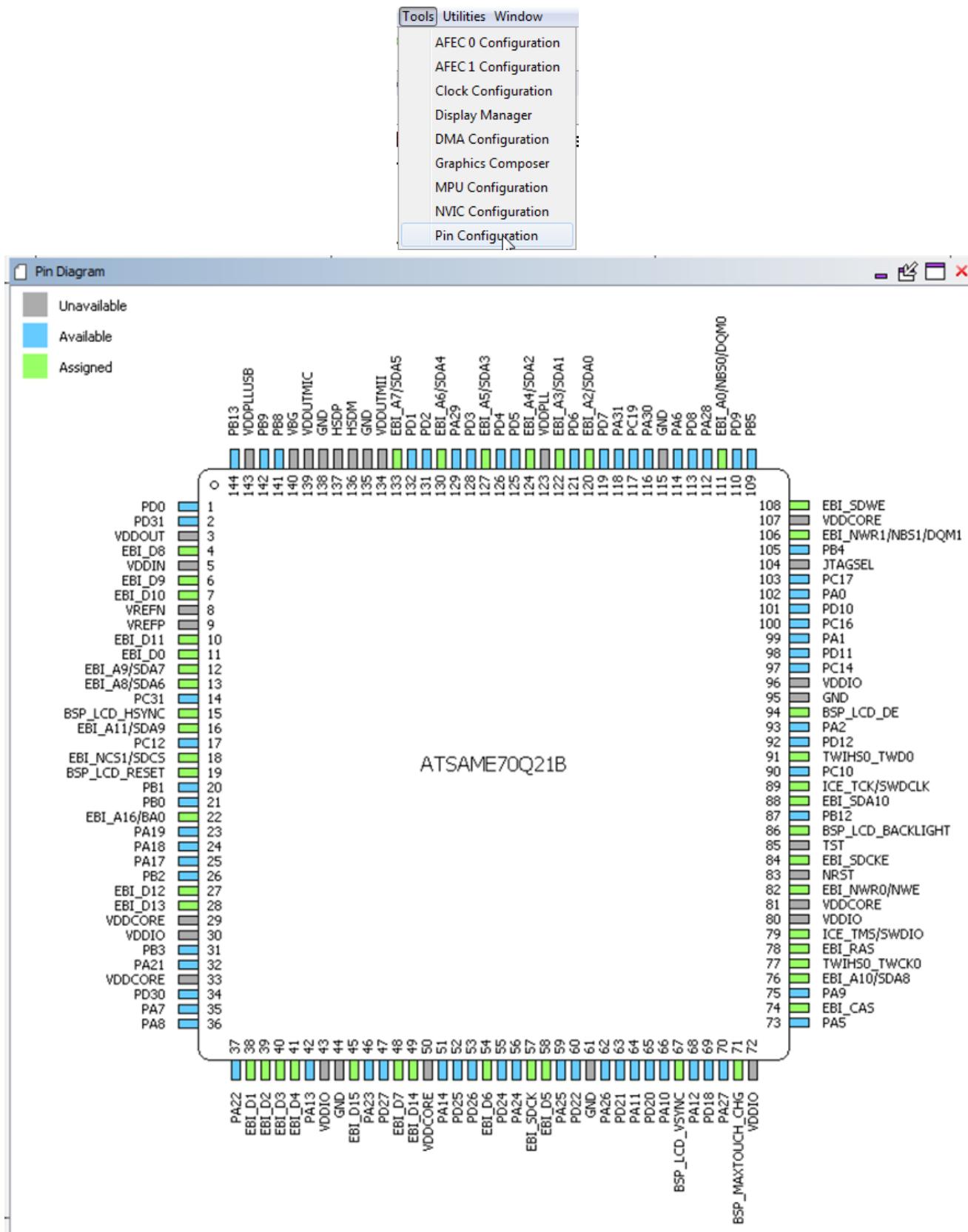
8. In the New Project Wizard, for the Color Mode step, select RGB_565 and hit **NEXT**.
9. For the Memory Size step, accept the default Flash Memory Size and hit **NEXT**.
10. For the Project Type step, chose the second option to create a new project using a basic template:



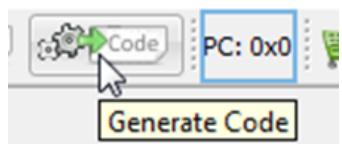
The MHGC window should now show:



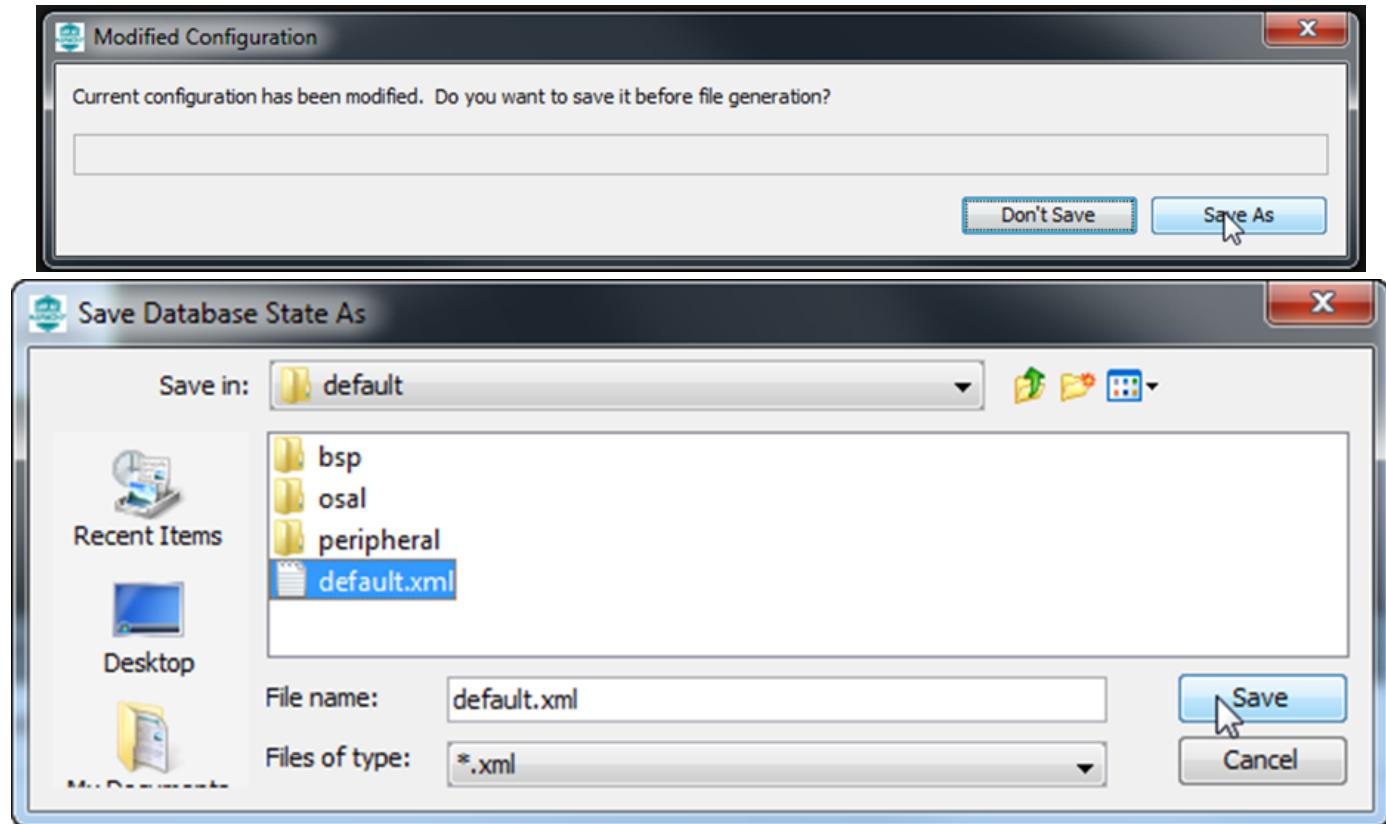
11. Launch the Pin Configuration tool and verify that the display is correctly setup:



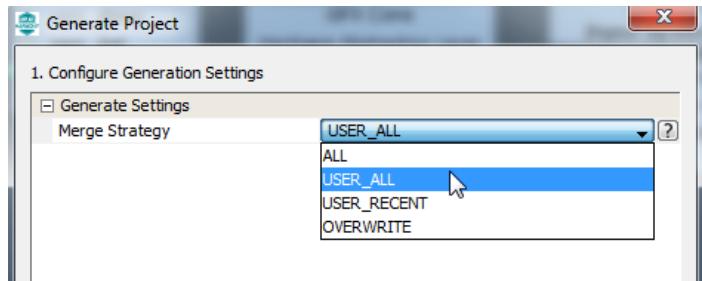
12. Generate the application's code for the first time. Select the Generate Code button of MHC's window:



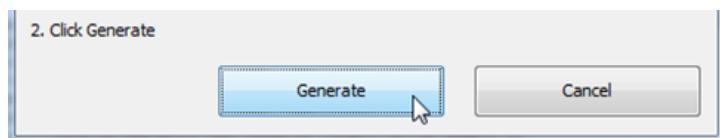
Save the project's configuration (any name will do for the .xml file):



Select USER_ALL as the Merge Strategy :



Click Generate:

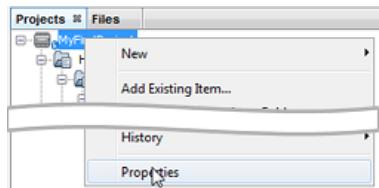


Now the project's initial software has been configured.

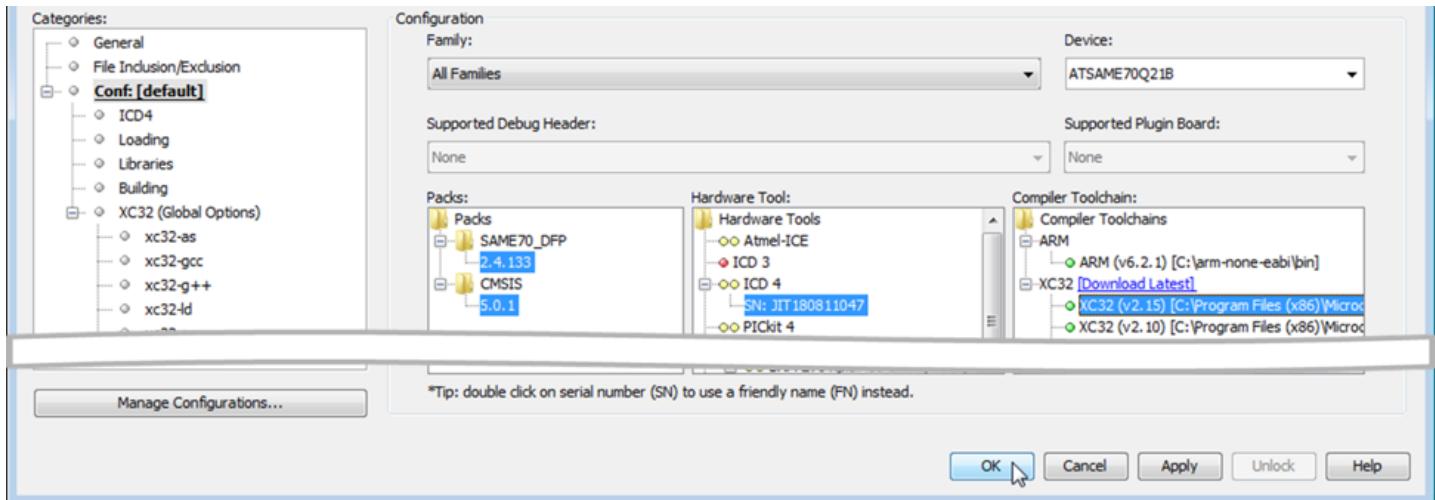
NOTE: Here is a brief explanation of the different merge strategies that are available:

- **ALL:** The user will be prompted with a merge window for all generated files. This includes files that have no user modifications but are changed because of changes in MHC configuration or component updates. (This choice is always the safest.)
- **USER_ALL:** The user will always be prompted with a merge window for all generated files that contain user modifications.
- **USER_RECENT:** The user will be prompted with a merge window for all generated files that contain user modifications.
- **OVERWRITE:** All generated file content will be replaced by the contents of this generate operation. All user changes will be overwritten.

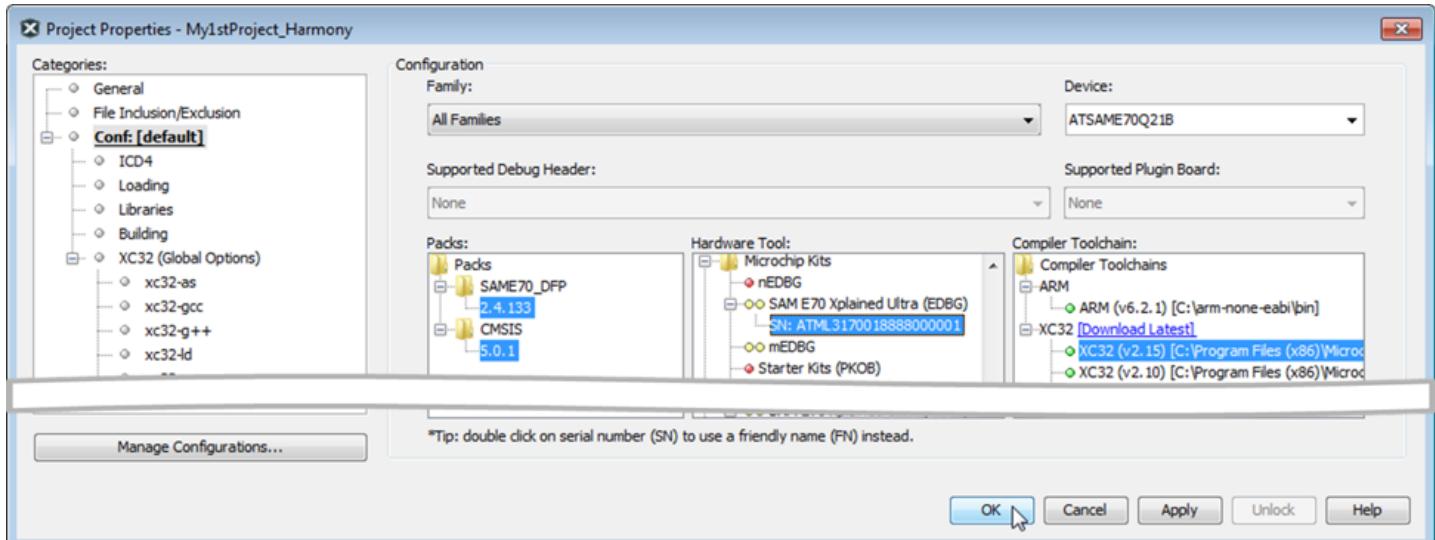
13. Right mouse click on the project's name and bring up the Project Properties dialog:



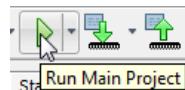
Select the debugger (here ICD 4) and the XC32 compiler (here V2.15). Then hit OK.



If you are using the Embedded Debugger (EDBG) instead:



14. Build and run the project:



15. The follow should appear on the display:



Next Steps

- Learn how to add events to the project: *MPLAB Harmony Graphics Library Help > Quick Start Guides > Graphics and Touch Quick Start Guides > Adding an Event to the Aria Quickstart Demonstration*
- Learn how to use phantom buttons to improve touch performance: *MPLAB Harmony Graphics Library Help > MPLAB Harmony Graphics Composer User's Guide > Advanced Topics > Improved Touch Performance with Phantom Buttons*

Adding an Event to the Aria Quickstart Demonstration

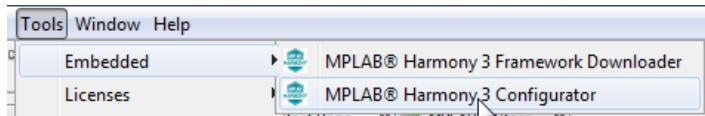
Provides a step-by-step guide to adding simple events to the Aria Quickstart demonstration project.

Description

This Quick Start tutorial shows how to add a simple event to the Aria Quickstart demonstration's button. For details on how to load, build, program, and run this project, see *MPLAB Harmony Graphics Library Help > Graphics Demonstrations > Demonstrations > aria_quickstart*.

The steps are as follows:

1. Launch the MPLAB Harmony Configurator:

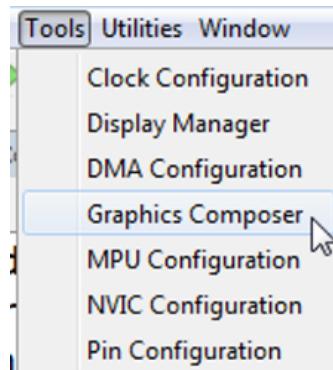


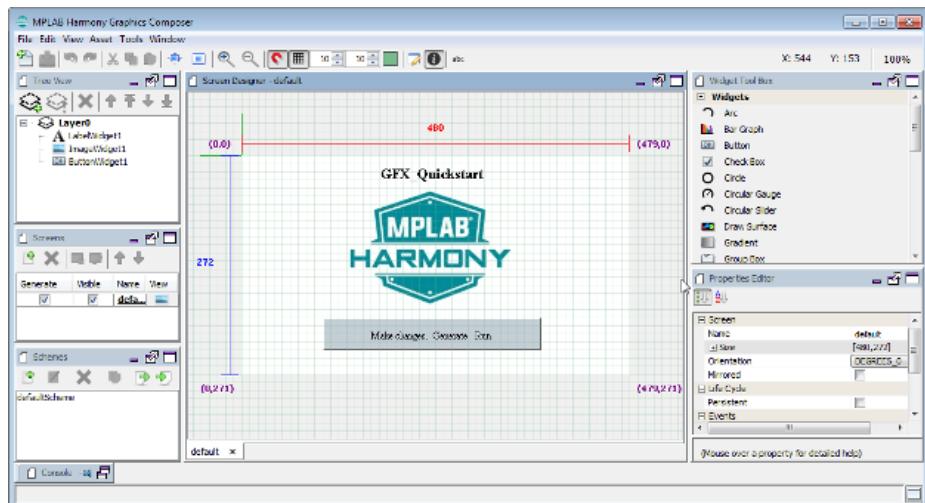
2. Open the project's default saved state:



When the Configuration Database Setup dialog appears, just hit **Launch**.

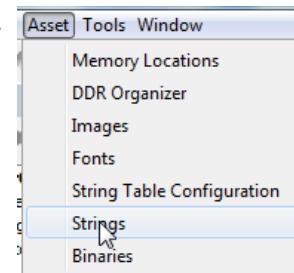
3. Select the Graphics Composer from the MHC Pull-Down Menu which will open a new window for the MPLAB Harmony Graphics Composer:





4. In step 8, below, we will change the behavior of the screen's button by adding **Pressed** and **Released** events.

The **Pressed** event will change the text of the button. First, we must add the needed text. Select the **Asset** menu and click **Strings**.



5. Click the Add icon, , and name the new string "Ouch_String" and then select **Create**.

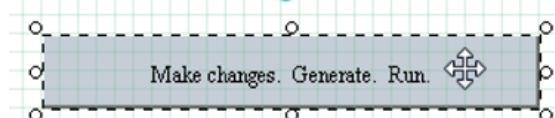


6. Enter the string value of **Ouch! Ouch! Ouch!**. Next, Select **TimesNewRoman12** as the font . On completion, the screen should show:

Ouch_String	"Ouch! Ouch! Ouch!"	
default	Ouch! Ouch! Ouch!	TimesNewRoman12

When finished, close the String Asset dialog.

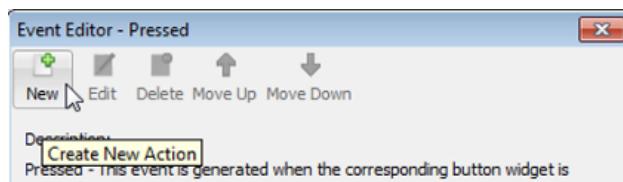
7. Select the Composer Screen Designer panel and select the **ButtonWidget1** ("Make changes. Generate. Run").



8. Under the **Properties Editor** panel on the right side of the screen, the properties of **ButtonWidget1** should appear. Enable the **Pressed** and **Released** events:

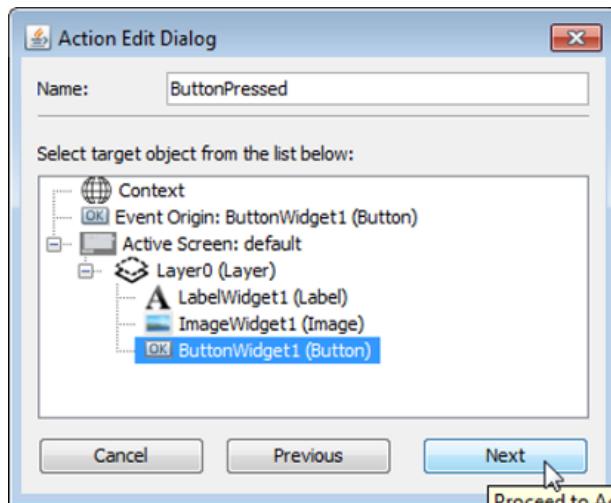


9. Click the  button on the right side to bring up the Events Editor for the **Pressed** event and create a new action.

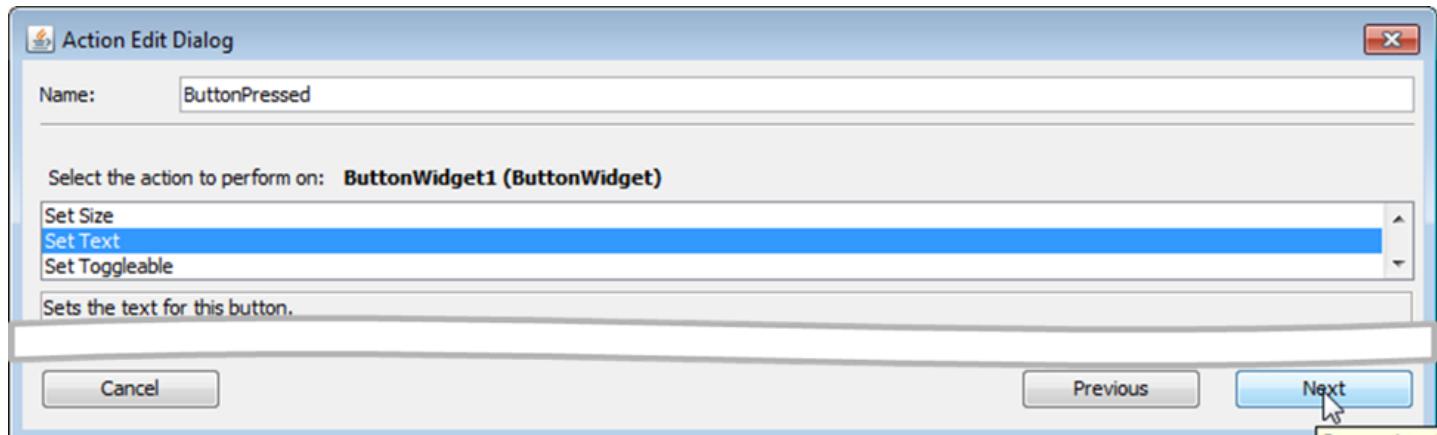


When the **Action Edit** dialog appears. Enter the name **ButtonPressed** and click **Next**.

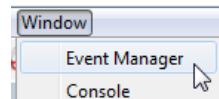
- Now select which widget the ButtonPressed event of ButtonWidget1 affects. This event can affect any of the display's widgets. However, the objective is for the ButtonPressed event to change the text of its own widget. Therefore, select **ButtonWidget1** and click **Next**.



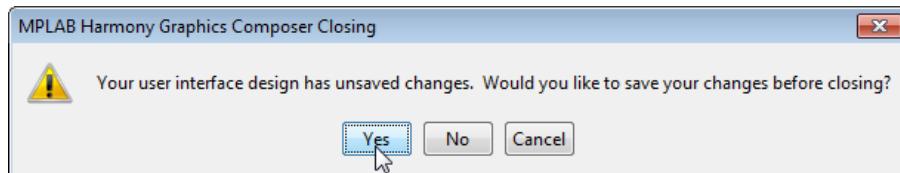
- The next dialog window provides instructions to select the action associated with this event. There are many choices, from **Adjust Position** to **Set Y Position**, all of which are related to properties of the button widget. Select **SetText** as the action, and then click **Next**.



- In the next **Action Edit Dialog** window select the **Ouch_String** from steps 5 and 6, and then click **Finish**. This action resurfaces the **Event Editor - Pressed** dialog. Click **OK** to close this dialog.
- Follow the same steps for the **Released** event, but instead create a new event named **ButtonReleased**. Use the same **SetText** action as before, but use the **Instructions** string instead.
- Examine the events that have been created by the Window:Event Manager menu. The Event Manager dialog is an alternative way of creating and managing graphics events for the application.



- Close the Graphics Composer window and save the new design.



16. In the MPLAB Harmony Configurator (MHC) window save the new configuration, since it has been modified. Generate code

for this new project configuration by selecting the Generate icon (). Select **Overwrite** as the Merging Strategy, since we don't need to review all the changes made to the graphics design. Then select the **Generate** button to generate the new code that implements the events that have just been added.

17. Select the Run Main Project icon () to build, load, and run this new configuration.

18. After the project has loaded, click the "Make changes. Generate. Run." button. The button's text should change to **Ouch!** **Ouch! Ouch!** when pressed and revert back to the original text when released.

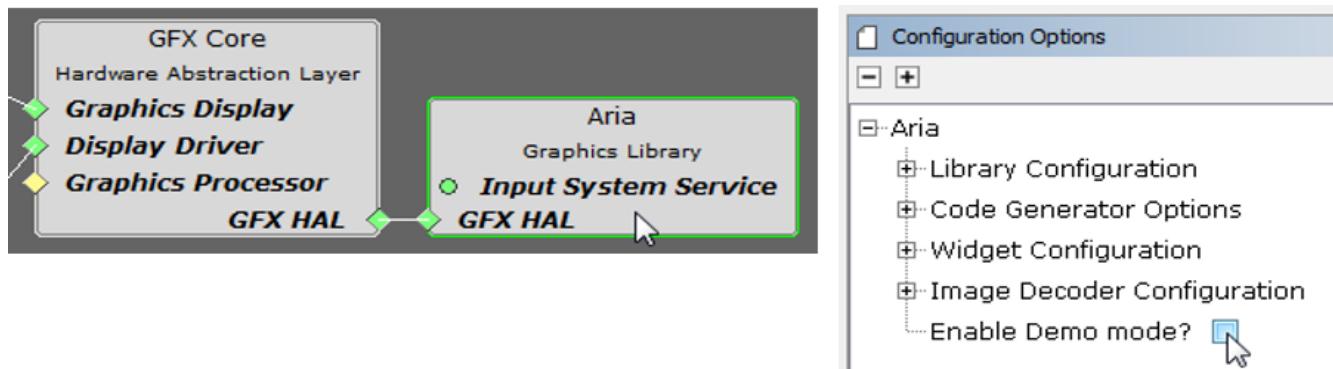
Enabling Demo Mode in a MPLAB Harmony Graphics Application

This section provides a step-by-step guide on how to enable Demo Mode in a MPLAB Harmony Graphics Application.

Description

Demo Mode is a feature in the MPLAB Harmony Graphics Library that enables an application to automatically replay touch input events after a specified period of idle time. This feature is useful for the application to automatically show simulated user activity, and show various screens and widgets without user input.

Demo Mode is enabled by clicking on the Aria Graphic Library component in MHC's Project Graph. Then under the component's Configuration Options panel, select "Enable Demo Mode?":



The following table lists the Demo Mode options and their functions:

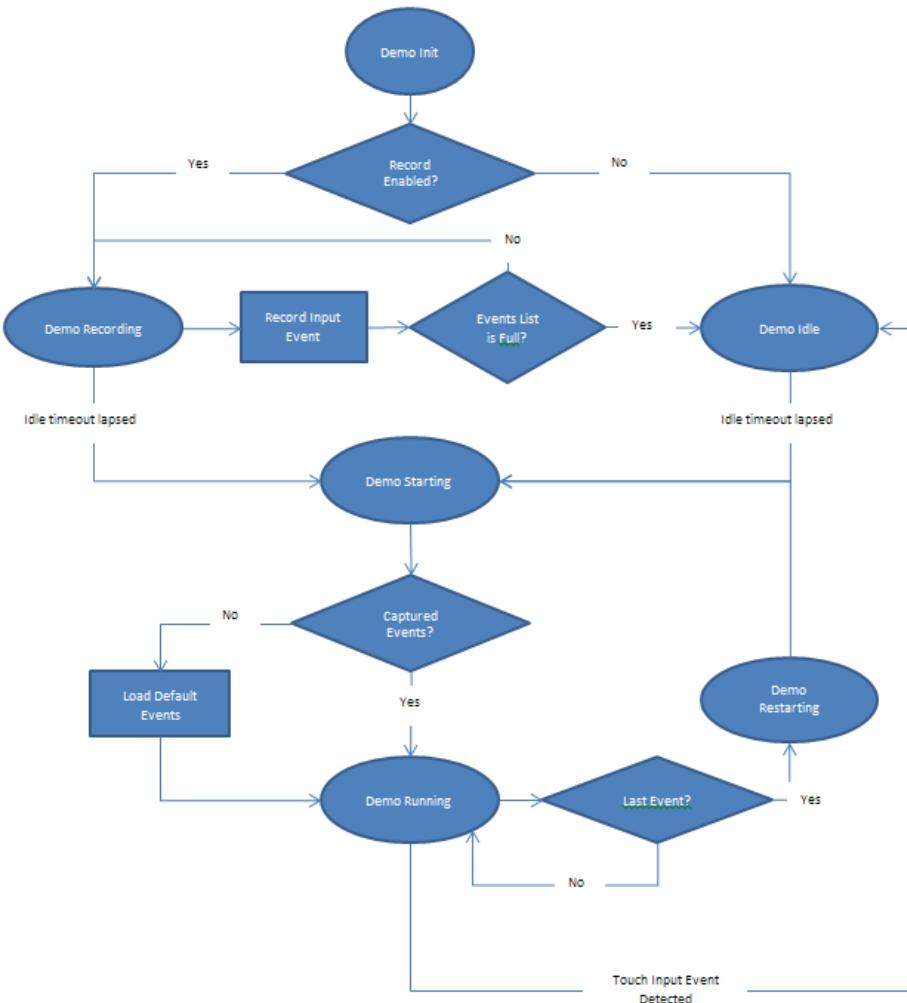
MHC Option	Function
Record Input Events	Enables input recording (touch input events are recorded and replayed).
Record Tick Period	Minimum time stamp interval between recorded input events.
Maximum number of events	Size of the events list (statically allocated).
Idle Timeout (seconds)	Period of inactivity before Demo mode starts. The idle time-out counter resets when a touch input event occurs.
Replay Delay (seconds)	Delay period in after Demo mode ends and restarts.

The Demo Mode feature can record input touch events, or load a default (programmed) list of events and replay them after the idle time-out period lapses.

To configure Demo Mode to record touch events, select the Record Input Events option in Demo Mode in MHC. This will start capturing touch input events by recording the event type and the interval between events. Intervals will be a multiple of the Record Tick Period time. The first touch event should occur before the Idle Timeout period lapses; otherwise, it will stop recording and use a default list of events, if available.

To program a default list of touch events, go to LibAria_LoadDefaultEvents(void) in libaria_demo_mode.c and call LibAria_DemoModeAddInputEvents with the desired touch parameters. This list of touch events will be loaded when Demo Mode starts.

The following figure shows the Demo Mode states and the actions in each state.



The following table lists and describes each state.

State	Description
Demo Init	Initializes data structures, launches record and idle timeout timers.
Demo Recording	If recording is enabled, touch events are captured to a list. Idle timeout timer is reset when a touch event occurs.
Demo Idle	Waits for idle timeout timer to trigger. If a touch event occurs, idle timeout timer is reset.
Demo Starting	Loads captured or default events list, reset application to first screen.
Demo Running	Replays events list. If a touch event occurs, demo is stopped and goes back to idle state. If all events have been replayed, starts replay delay timer.
Demo Restarting	Restarts Demo Mode after the delay timer lapses.

Graphics Demonstrations

This section provides descriptions of the Harmony Graphics applications.

Description

Introduction

This topic provides help for MPLAB Harmony Graphics applications.

Description

This distribution package contains a variety of Graphics-related firmware projects that demonstrate the capabilities of the MPLAB Harmony Graphics library. This help file describes the hardware requirements, hardware setups, and procedures to run these demonstration projects on Microchip graphics boards.



Important!

Due to distribution streamlining, the MPLAB Harmony Graphics applications are split between two repositories.

Quickstart applications, such as [aria_quickstart](#), can be found under the **apps folder** in Harmony **gfx** repository.

Non-quickstart graphics applications reside under the **apps folder** in the Harmony **gfx_apps** repository.

Although each application can run standalone from within their respective repository, the use of MPLAB Harmony Configurator and code regeneration require the presence of other Harmony repositories.

For information on Harmony repository dependencies, please refer to the Software Requirements section of Graphics Release Notes.

Prior to using these demonstrations, it is recommended to review the **MPLAB Harmony Graphics Release Notes** for any known issues. A PDF copy of the release notes is provided in the <install-dir>/doc folder of the installation.

To learn how to create graphics applications within MPLAB Harmony see [Quick Start Guides > Creating New Graphics Applications](#). To know more about MPLAB Harmony Graphics, configuring the library, and the APIs provided; see [Graphics Library Help](#).

Demonstrations

This topic provides information on how to run the MPLAB Harmony Graphics applications included in this release.

aria_benchmark

This application presents frame update rate metrics on the various rendering operations in the Harmony graphics library.

Description

This application shows the frame update rates for various operations in the Harmony graphics library, including string rendering, area fills, and image decode and rendering. The benchmarks can be configured for different text sizes, number and size of discrete area fills and different image formats. The instantaneous and averaged frame update rates are shown at real-time on the display.

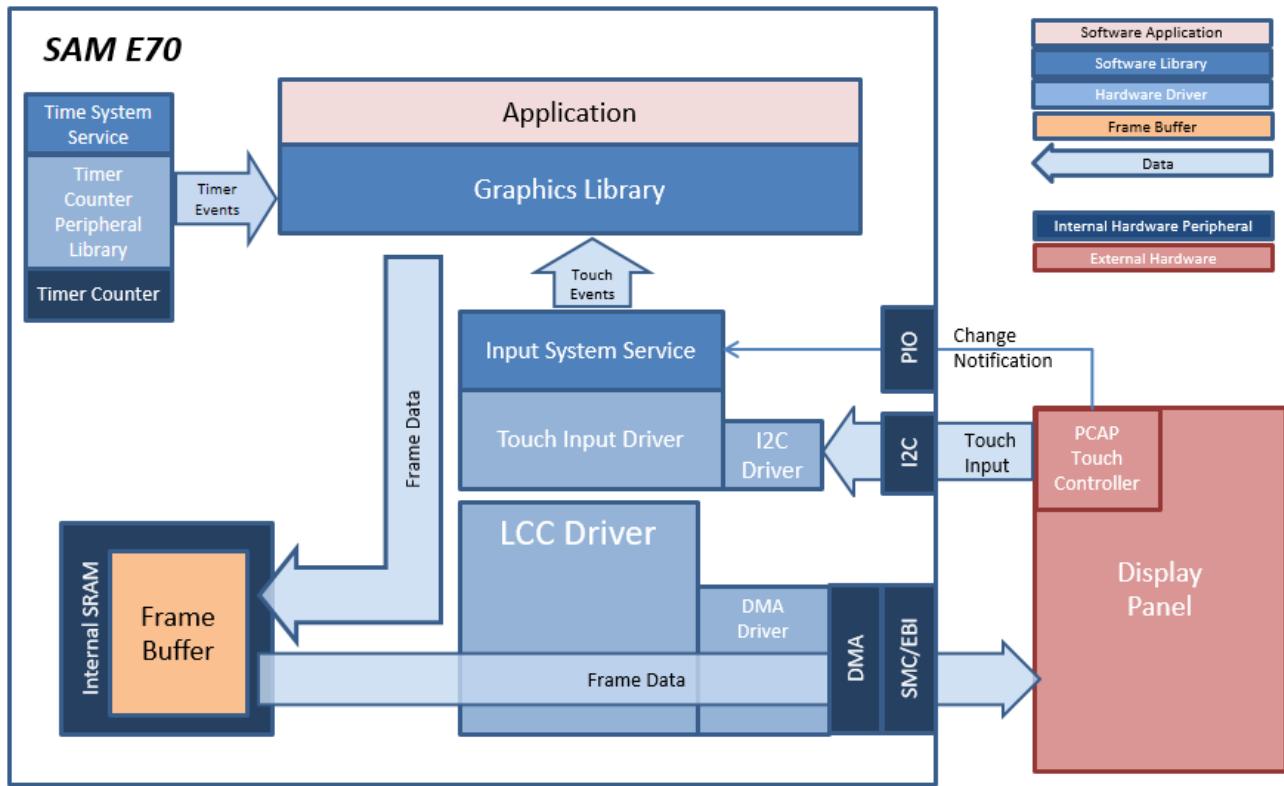
Architecture

The application continuously uses the graphics library to render text, fill areas, and draw images to the screen. Once a layer is completely rendered to, the graphics library increments a layer swap counter. The application periodically (at 1 second intervals) samples the layer swap counter and calculates the difference from the previous sample. This difference is shown as the Frame Update Rate (Hz).

The following diagrams show the various software and hardware blocks used in this application:

aria_bm_e70_xu_tm4301b

In this configuration, a 16-bit RGB565 frame buffer is stored in internal SRAM. Due to the limited size of the internal SRAM, only a single frame buffer can be used which can cause tearing to be visible as the GFX library draws on the screen. These configurations use the Low-Cost Controller-less (LCC) display driver to manage the DMA that transfers the framebuffer contents to the display.



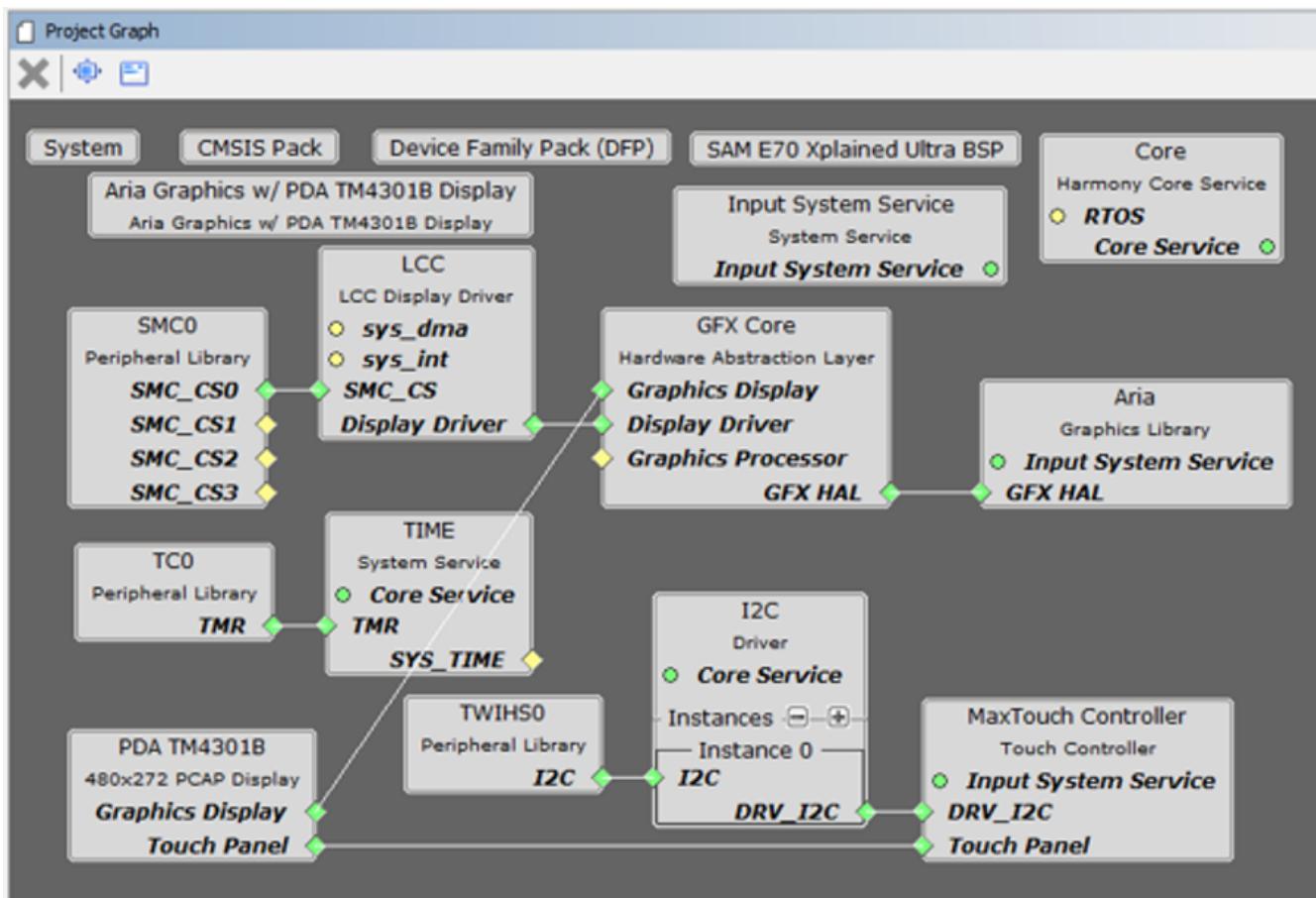
Demonstration Features

- Input system service and driver
- Time system service, timer-counter peripheral library and driver
- DMA System Service
- Low-Cost Controllerless (LCC) graphics driver
- I²C driver
- 16-bit RGB565 color depth support (65535 unique colors)
- JPEG and PNG images stored in internal flash

MPLAB Harmony Configurator Setup

The Project Graph diagram below shows the Harmony components that are included in this application. Lines between components are drawn to satisfy components that depend on a capability that another component provides.

Adding the “SAM E70 XPlained Ultra BSP” and the “Aria Graphics w/ PDA TMA4301B Display” components into the project graph will automatically add the components needed for a graphics project and resolve their dependencies. It will also configure the pins needed to drive the external peripherals like the display and the touch controller.



Building the Application

This section identifies the MPLAB X IDE project name and location and lists and describes the available configurations for the Aria Showcase Reloaded demonstration.

Description

The parent directory for this application is gfx_apps/apps/aria_benchmark. To build this application, open the gfx_apps/apps/aria_benchmark/firmware/*.X project file in MPLAB X IDE and then select the desired configuration for the board.

MPLAB X IDE Project Configurations

The following table lists and describes the supported configurations:

Project Name	BSP Used	Graphics Template Used	Description
aria_bm_e70_xu_tm4301b.X	sam_e70_xplained_ultra	Aria Graphics w/ PDA TM4301B Display	SAM E70 Xplained Ultra board with PDA TM4301B 480x272 (WQVGA) Display



Important! This application may contain custom code that is marked by the comments // START OF CUSTOM CODE ... and // END OF CUSTOM CODE. When using the MPLAB Harmony Configurator to regenerate the application code, use the "ALL" merging strategy and do not remove or replace the custom code.

Configuring the Hardware

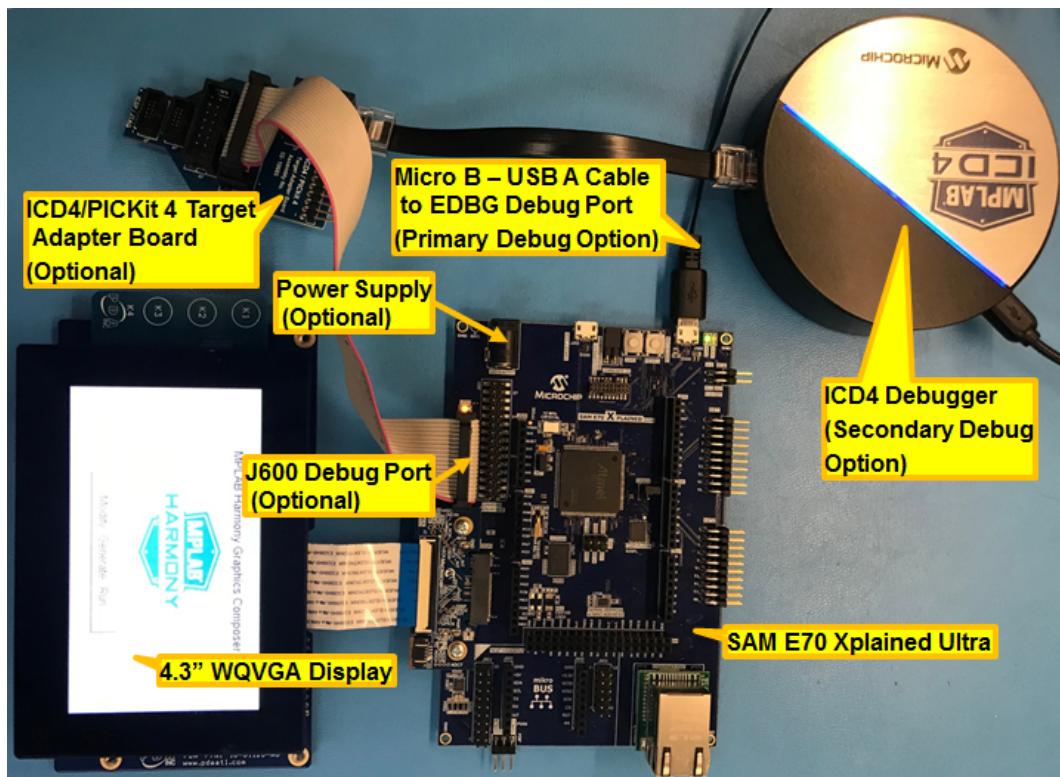
This section describes how to configure the supported hardware.

Description

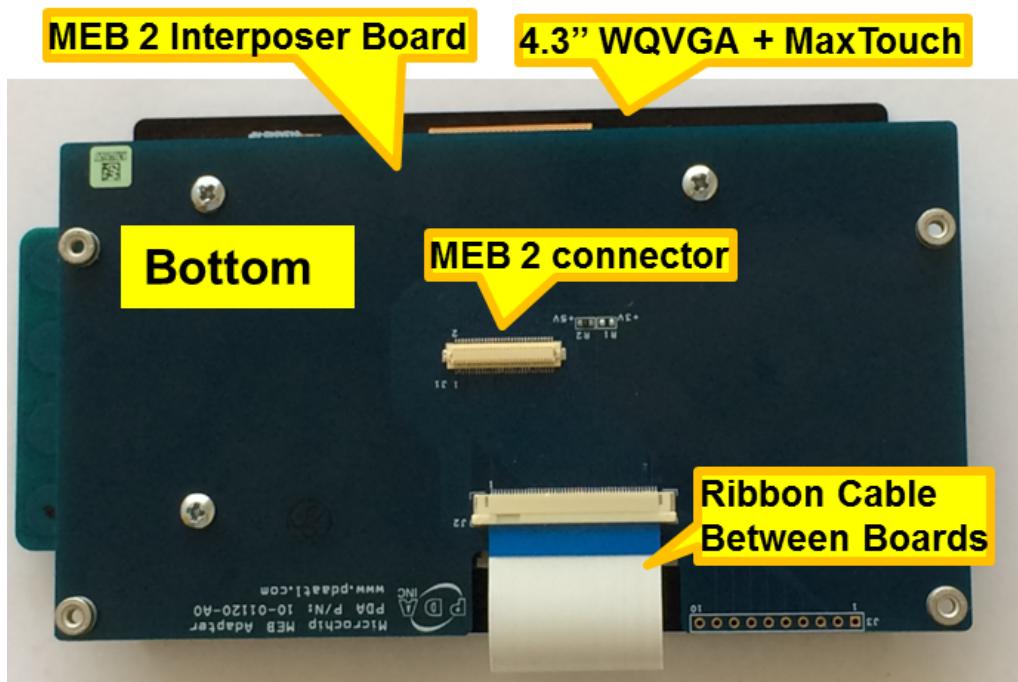
SAM E70 Xplained Ultra board with 4.3" WQVGA Display

There are two programming options. The primary option is the Micro B Embedded Debugger (EDBG) port. Power to the board can also be supplied via this connection. The secondary option is via the ICD4 debugger with the ICD4/PICKit 4 Target Adapter Board. (Power can be supplied by one of the USB Micro-B ports or via a bench supply.)

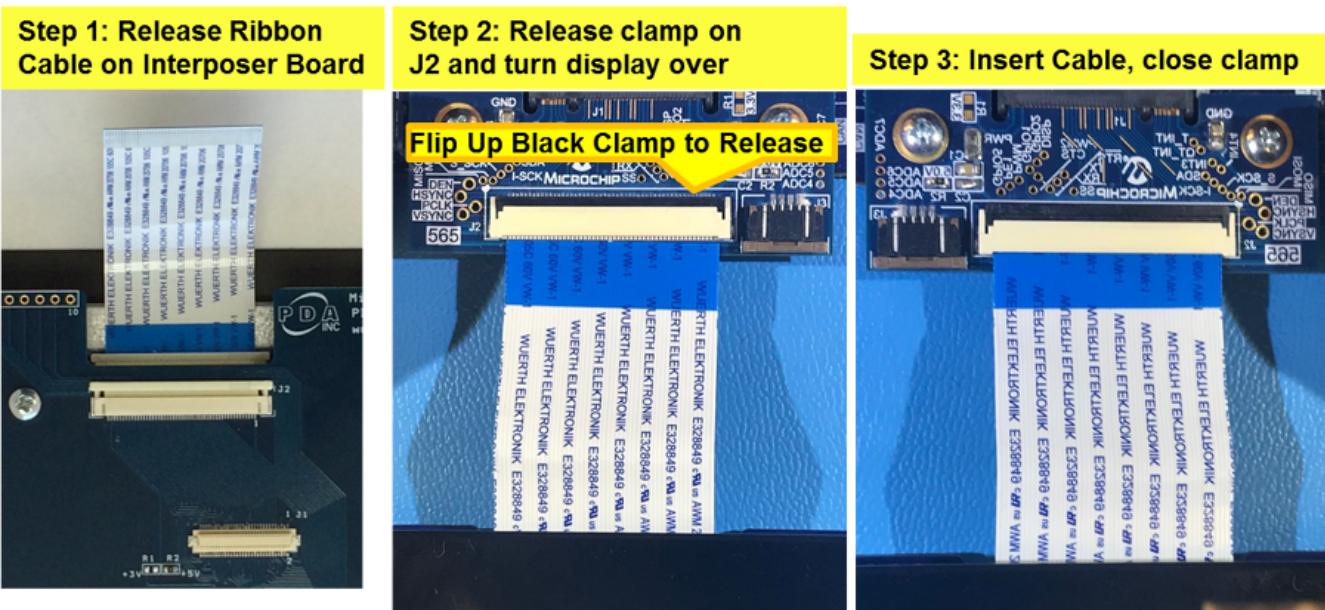
The image below shows both the EDBG and the ICD4 connection options:



Configuring the 4.3" WQVGA Display requires disconnecting the ribbon cable that connects the display to the interposer board.



First, release the ribbon cable from the interposer board. Next, release the black clamp on the E70's J2 connector and turn the display over. Finally, insert the ribbon cable into J2 and close the clamp.



The board and display are powered by a Micro B – USB A cable from PC to the “Debug USB” port on the E70 board. The ICD4 Debugger and ICD4/PICKit4 Adapter Board are connected as shown above.

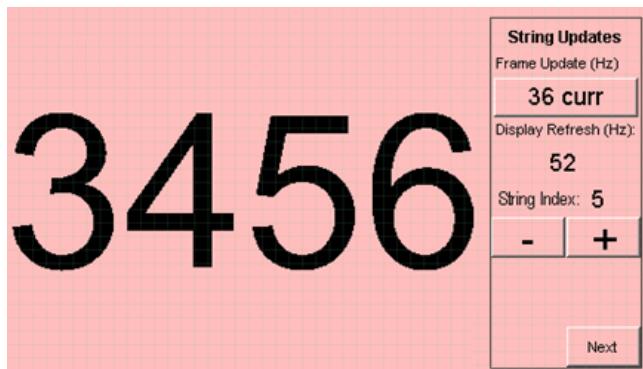
Running the Demonstration

This section provides information on how to run and use the application.

Description

On start-up, the application will display a splash screen.

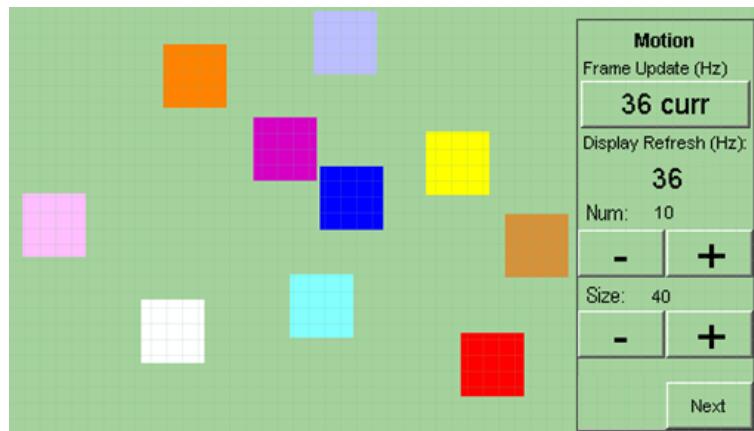
After the splash-screen completes, the String Update benchmark screen is shown. In this screen, a counter is incremented at every application cycle. The screen demonstrates the rate at which the graphics library renders a string on the screen. This involves a fill operation that clears the background, lookup of the glyphs from the string library, and the drawing of the glyphs on the frame buffer.



The “Frame Update (Hz)” field shows the current or instantaneous rate at which the graphics library updates the label widget that shows the counter value. Touching the Frame Update value switches between the current value (curr) and the average (avg) value across 10 samples.

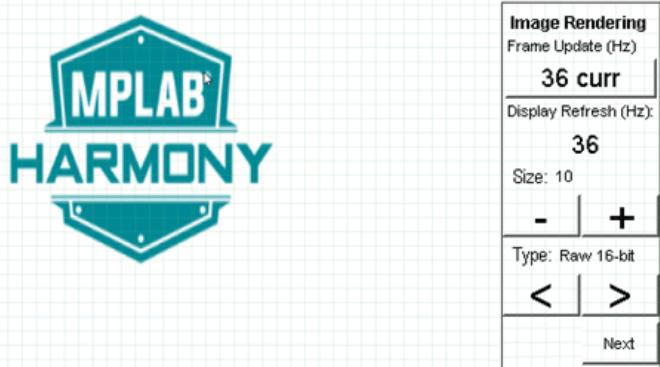
Touching the “+” and “-“ buttons increases and decreases the size of the string, respectively.

Touching the “Next” button switches to the Motion and Fill benchmark screen. In this screen, squares are showing moving across the screen. The Frame Update value is the rate at which the graphics library is able to render all the squares on the screen at their new positions. This involves a fill operation of the background color at the old location of the squares and a fill of the squares’ colors at the new position.



The number and size of the squares can be increased and decreased using the “+” and “-” buttons. If the maximum or minimum size is reached, touching “+” or “-”, respectively, will switch to a full screen fill of alternating colors.

Touching the “Next” button transitions to the Image Decode and Rendering screen. In this screen, two images of the same size are alternately rendered between application cycles. This involves a fill of the background color, decode and conversion of the image to the frame buffer format, and the drawing of the image to the frame buffer. The Frame Update value is the rate at which the graphics library is able to render an image on the screen.



The size of the images can be increased and decreased using the “+” and “-” buttons.

Touching the “<” and “>” buttons switches between the various image formats. The formats that are supported are PNG, RAW RLE 16-bit, RAW 16-bit and JPG 24-bit. RAW 32-bit pre-decoded in DDR is also supported in the PIC32MZ DA configurations with DDR memory.

aria_quickstart

This demonstration provides a touch-enabled starting point for the Aria Graphics.

Description

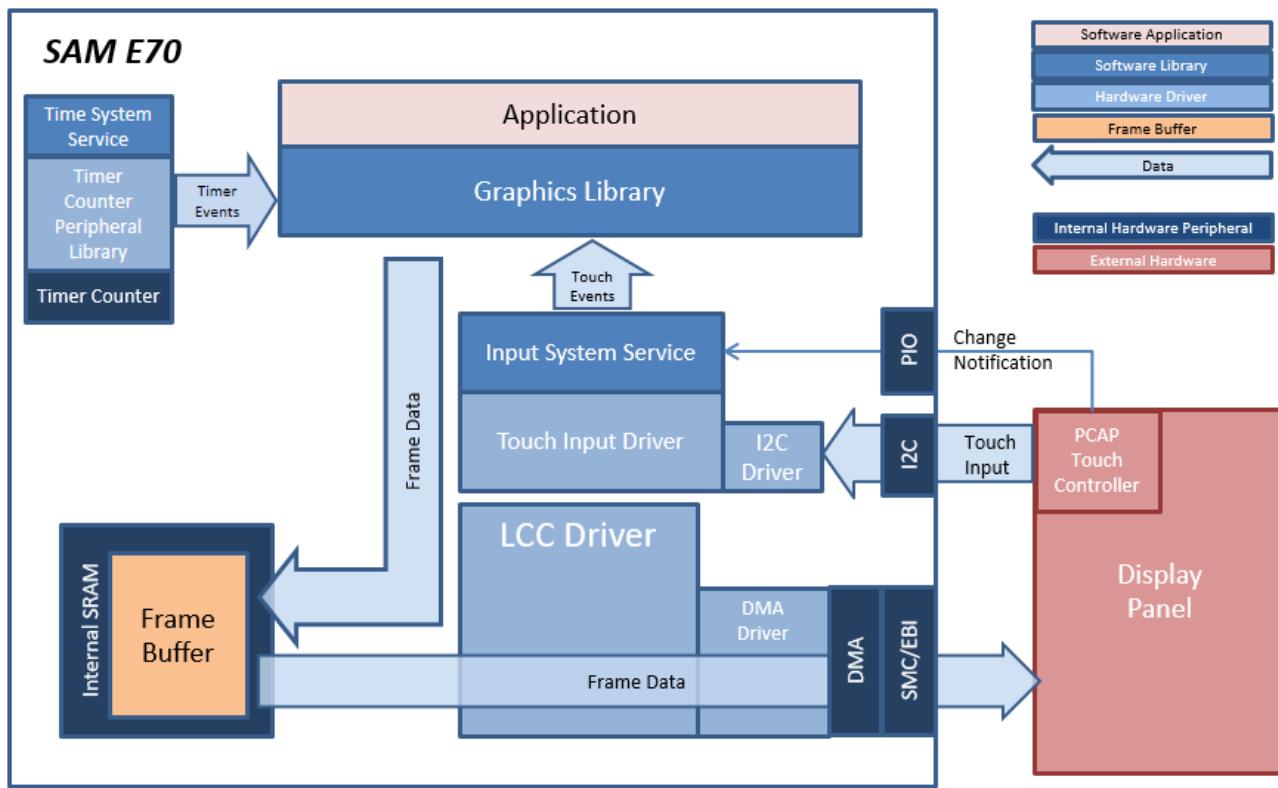
This demonstration serves as a preconfigured starting point for a touch-enabled application powered by the Aria User Interface Library. The design contains an image widget, a label widget and a button widget. To create this project from scratch, see *MPLAB Harmony Graphics Library Help > Quick Start Guides > Creating New Graphics Applications*.

Architecture

aria_qs_e70_xu_tm4301b

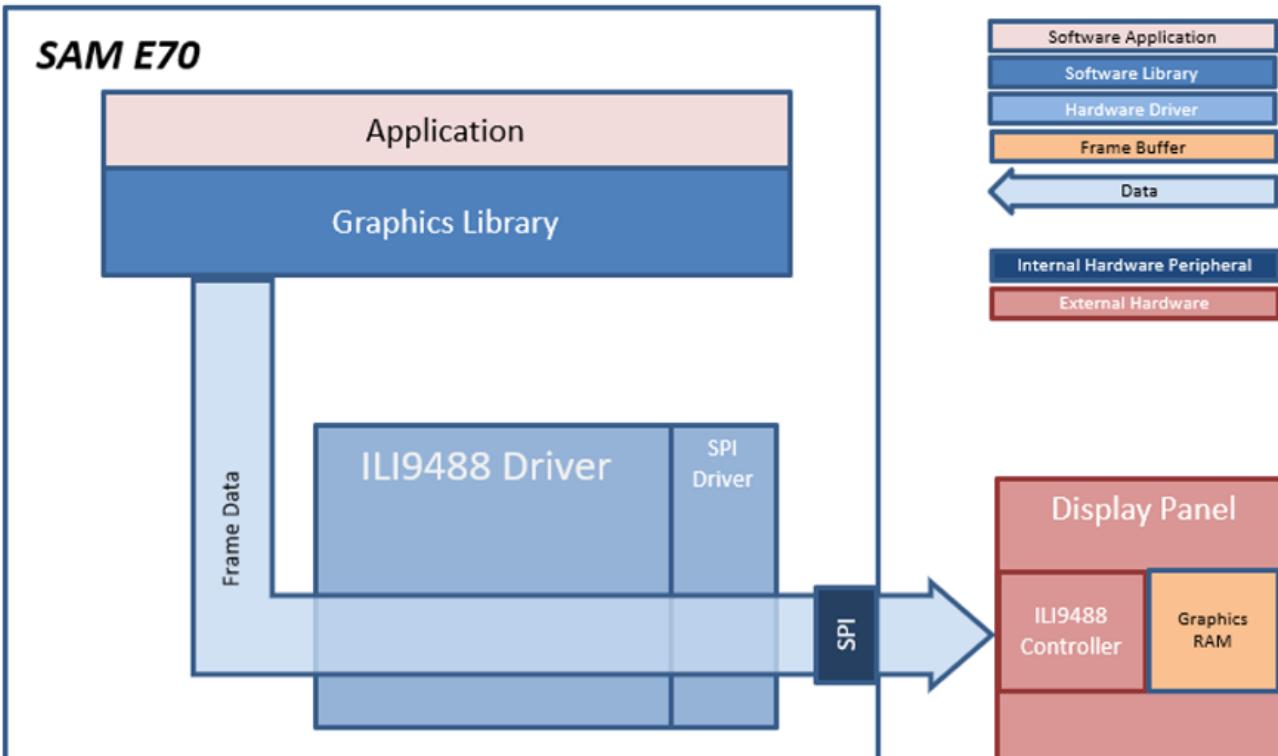
In this configuration, a 16-bit RGB565 frame buffer is stored in the internal SRAM. These configurations use the Low Cost Controller-less (LCC) display driver to manage the DMA that transfers the framebuffer contents to the display.

User touch input on the display panel is received thru the PCAP capacitive touch controller, which sends a notification to the Touch Input Driver. The Touch Input Driver reads the touch information over I2C and sends the touch event to the Graphics Library thru the Input System Service.



aria_quickstart_e70_xult_xpro_spi

This aria_quickstart project uses a maXTouch Xplained Pro display panel that is connected to the E70 Xplained Ultra board. The display panel has an ILI9488 display controller that communicates to the E70 using the SPI peripheral. The ILI9488 contains a graphics memory (GRAM) that is used as frame buffer and the Aria graphics library uses the ILI9488 SPI driver to write pixel data to the ILI9488 GRAM. The maXTouch Xplained Pro board has a touch controller, but that is not supported in this project.



Demonstration Features

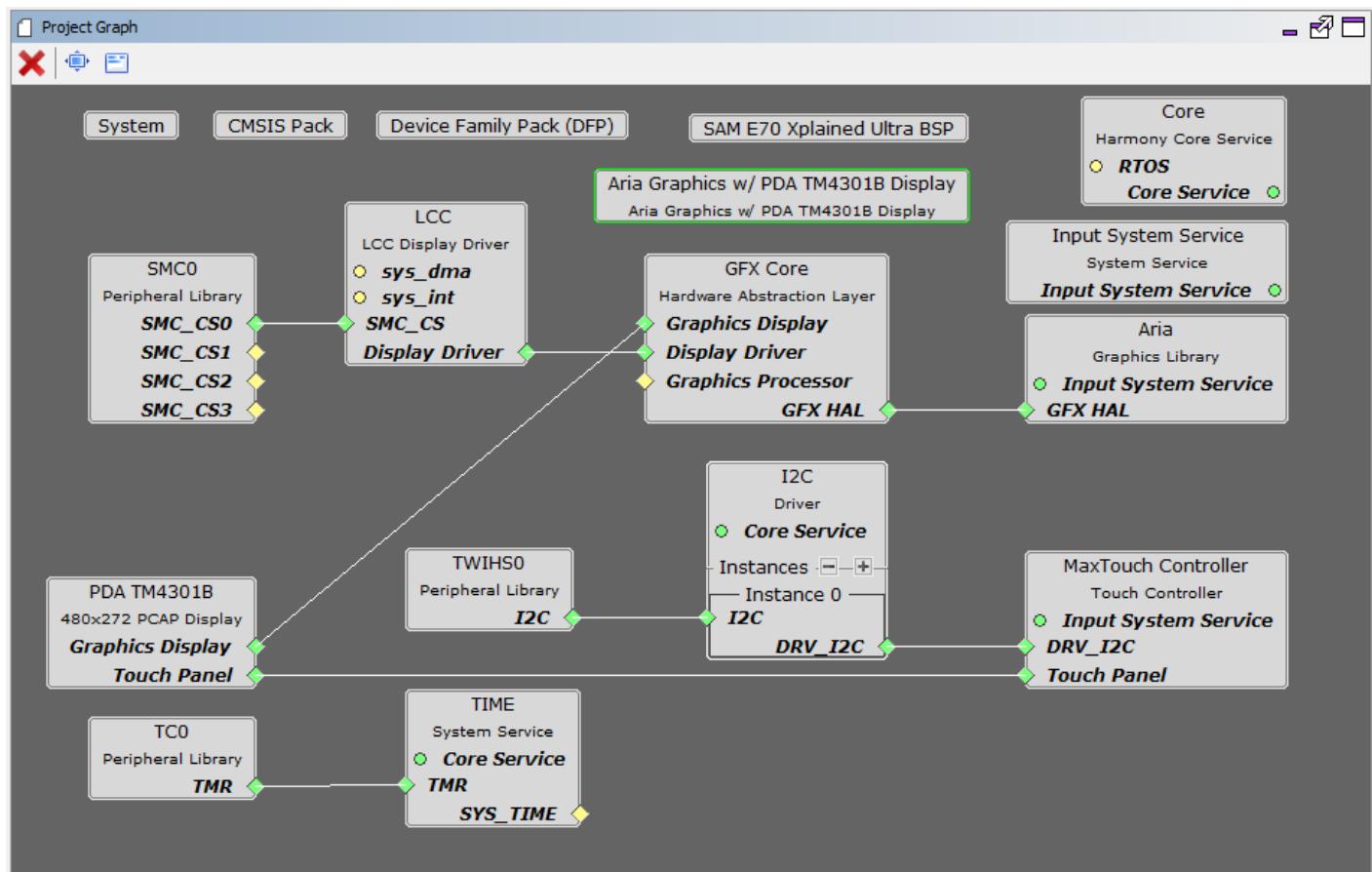
- Input system service and driver
- Time system service, timer-counter peripheral library and driver
- DMA System Service
- Low-Cost Controllerless (LCC) graphics driver
- I2C driver
- 16-bit RGB565 color depth support (65535 unique colors)
- JPEG image stored in internal flash

MPLAB Harmony Configurator Setup

aria_qs_e70_xu_tm4301b

The Project Graph diagram below shows the Harmony components that are included in this application. Lines between components are drawn to satisfy components that depend on a capability that another component provides.

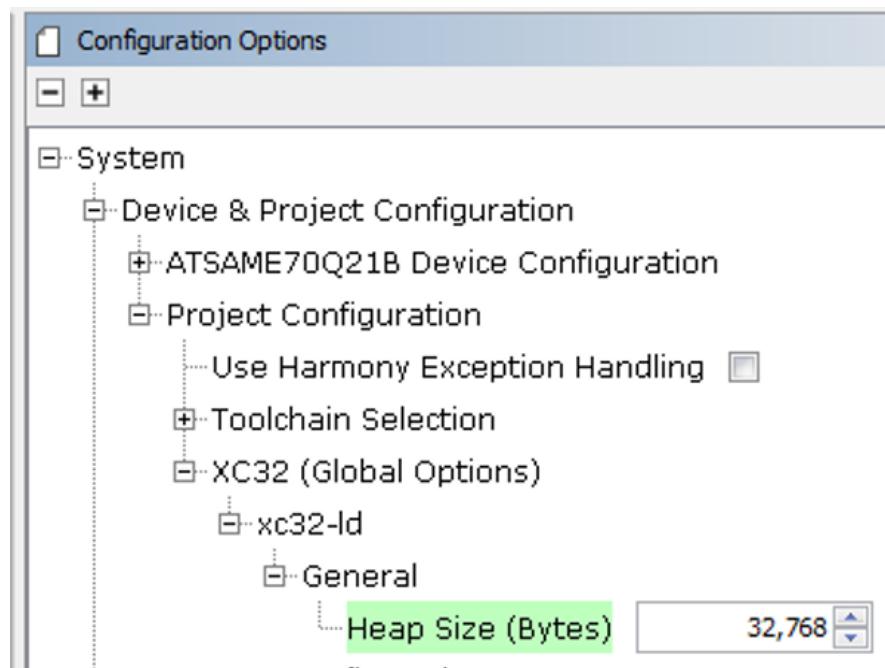
Adding the “SAM E70 XPlained Ultra BSP” and “Aria Graphics w/ PDA TM4301B Display” Graphics Template component into the project graph will automatically add the components needed for a graphics project and resolve their dependencies. It will also configure the pins needed to drive the external peripherals like the display and the touch controller.



The heap size is set to 32768 bytes. This is done by setting the Device & Project Configuration > Project Configuration > XC32

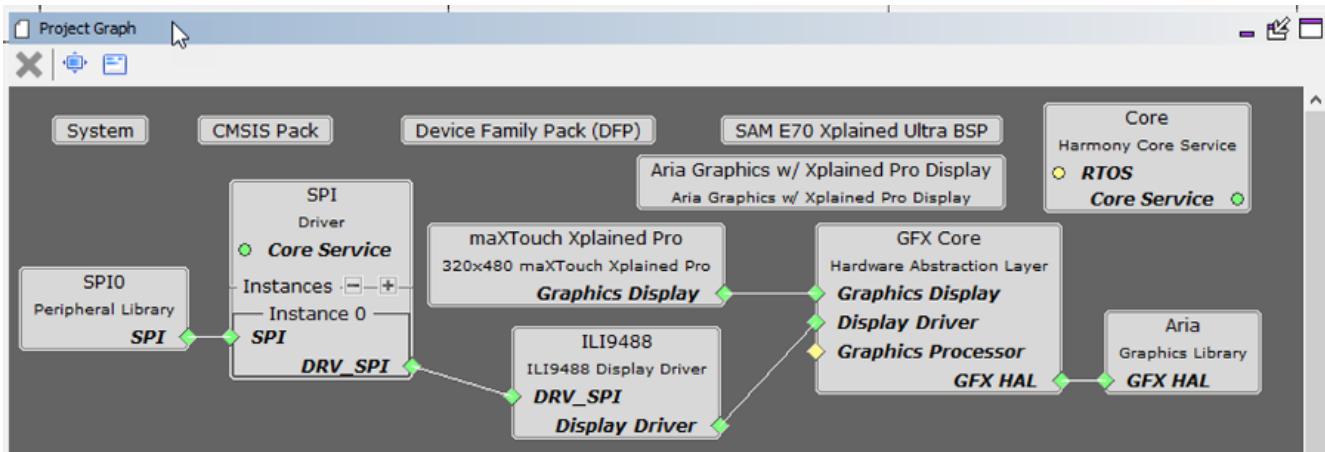
(Global Options) xc32-ld > General > Heap Size option for the “System” component.



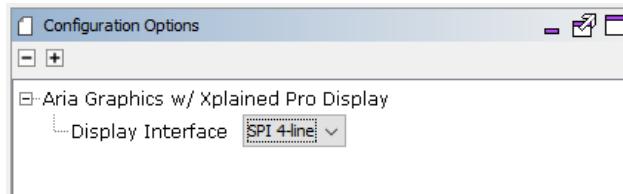


aria_quickstart_e70_xult_xpro_spi

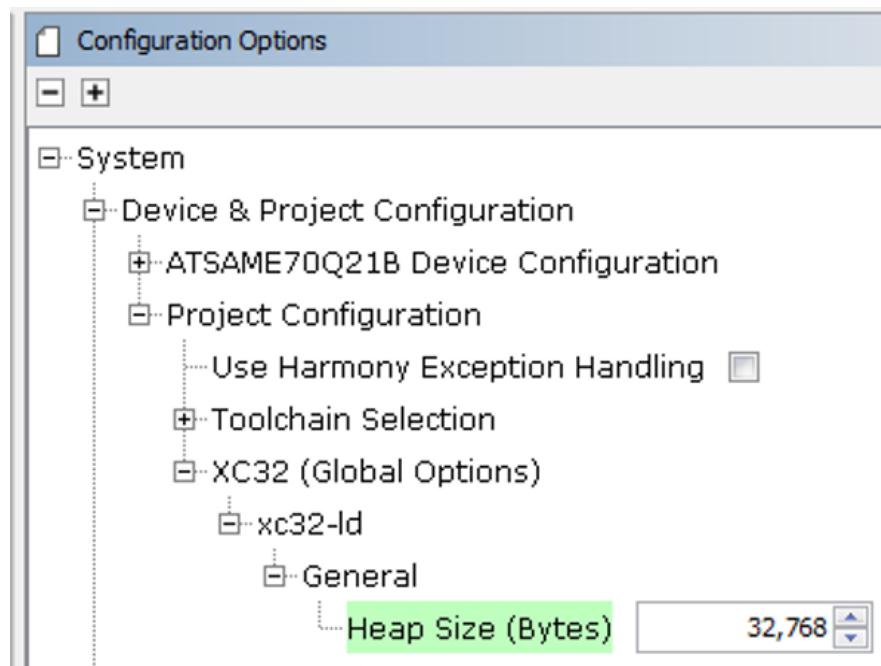
The Project Graph diagram below shows the Harmony components that are included in this application. Lines between components are drawn to satisfy components that depend on a capability that another component provides. Adding the "SAM E70 XPlained Ultra BSP" and "Aria Graphics w/ Xplained Pro Display" Graphics Template component into the project graph will automatically add the components needed for a graphics project and resolve their dependencies. It will also configure the pins needed to drive the external peripherals like the display and the touch controller.



Under the configuration options for the "Aria Graphics w/ Xplained Pro Display" GFX template component, the "Display Interface" is set to "SPI 4-line":



The heap size is set to 32768 bytes. This is done by setting the Device & Project Configuration > Project Configuration > XC32 (Global Options) xc32-Id > General > Heap Size option for the "System" component.



Building the Application

This section identifies the MPLAB X IDE project name and location and lists and describes the available configurations for the Aria Quickstart demonstration.

Description

The parent directory for this application is gfx/apps/aria_quickstart. To build this application, open the gfx/apps/aria_quickstart/firmware/*.X project file in MPLABX IDE that corresponds to the hardware configuration.

MPLAB X IDE Project Configurations

The following table lists and describes the supported configurations:

Project Name	BSP Used	Graphics Template Used	Description
aria_qs_e70_xu_tm4301b.X	sam_e70_xplained_ultra	Aria Graphics w/ PDA TM4301B Display	SAM E70 Xplained Ultra board with PDA TM4301B 480x272 (WQVGA) Display
aria_qs_e70_xu_xpro_spi.X	sam_e70_xplained_ultra	Aria Graphics w/ Xplained Pro Display	Aria GFX on SAM E70 Xplained Ultra Board and Xplained Pro Display with ILI9488 SPI display driver



Important! This application may contain custom code that is marked by the comments // START OF CUSTOM CODE ... and // END OF CUSTOM CODE. When using the MPLAB Harmony Configurator to regenerate the application code, use the "ALL" merging strategy and *do not* remove or replace the custom code.

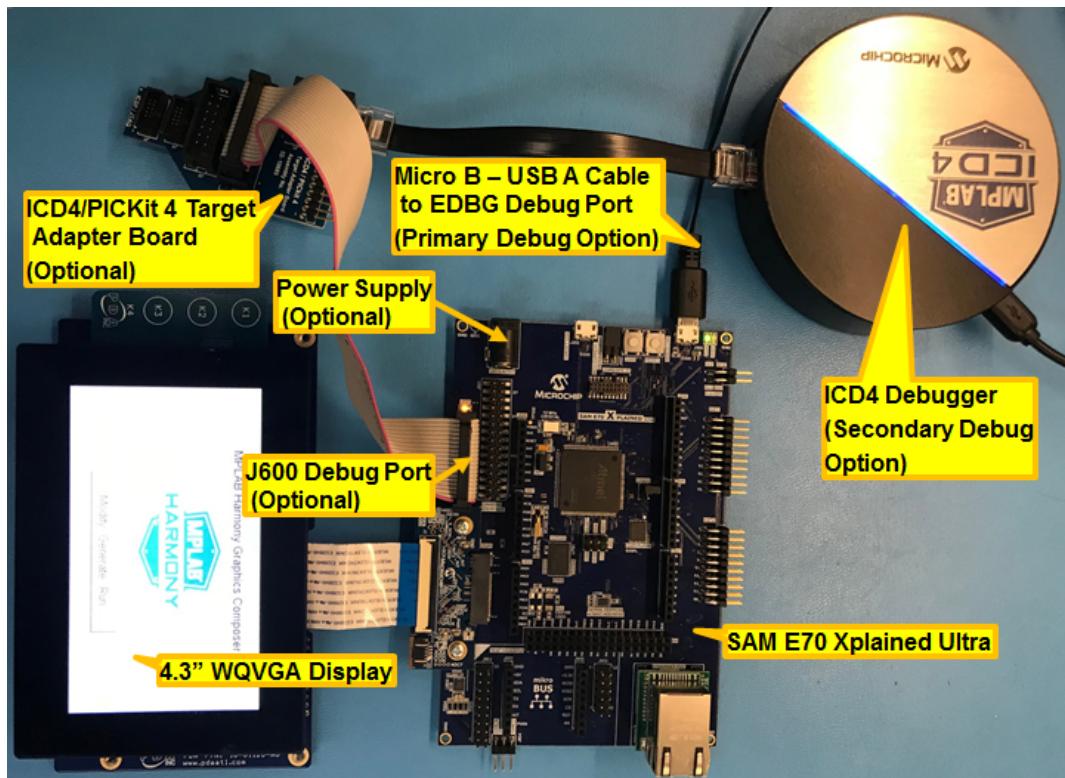
Configuring the Hardware

This section describes how to configure the supported hardware.

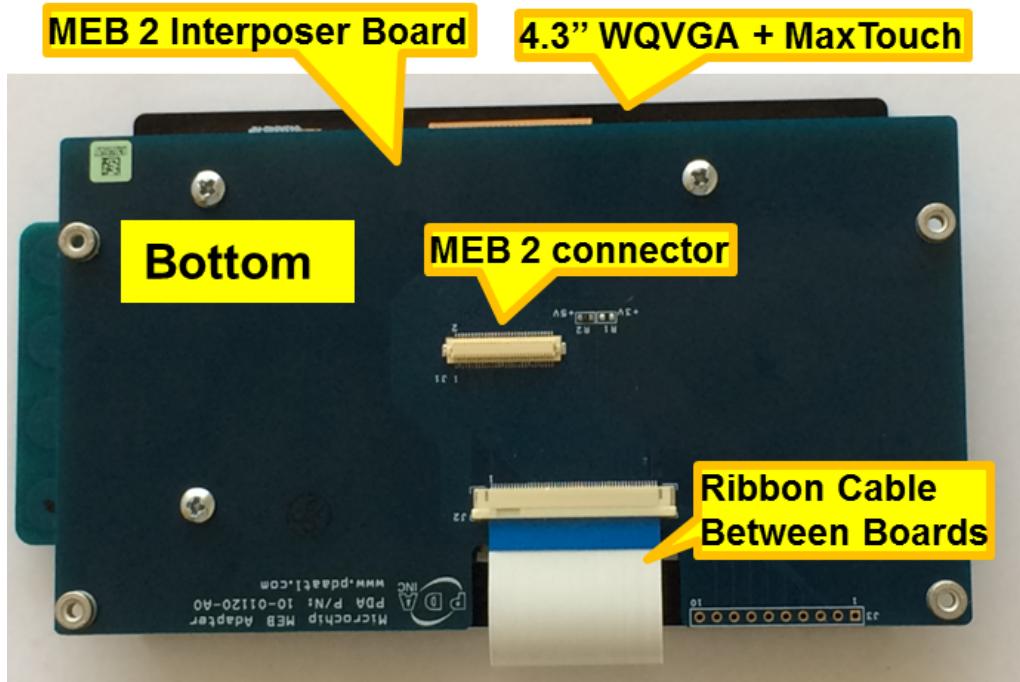
Description

SAM E70 Xplained Ultra board with 4.3" WQVGA Display

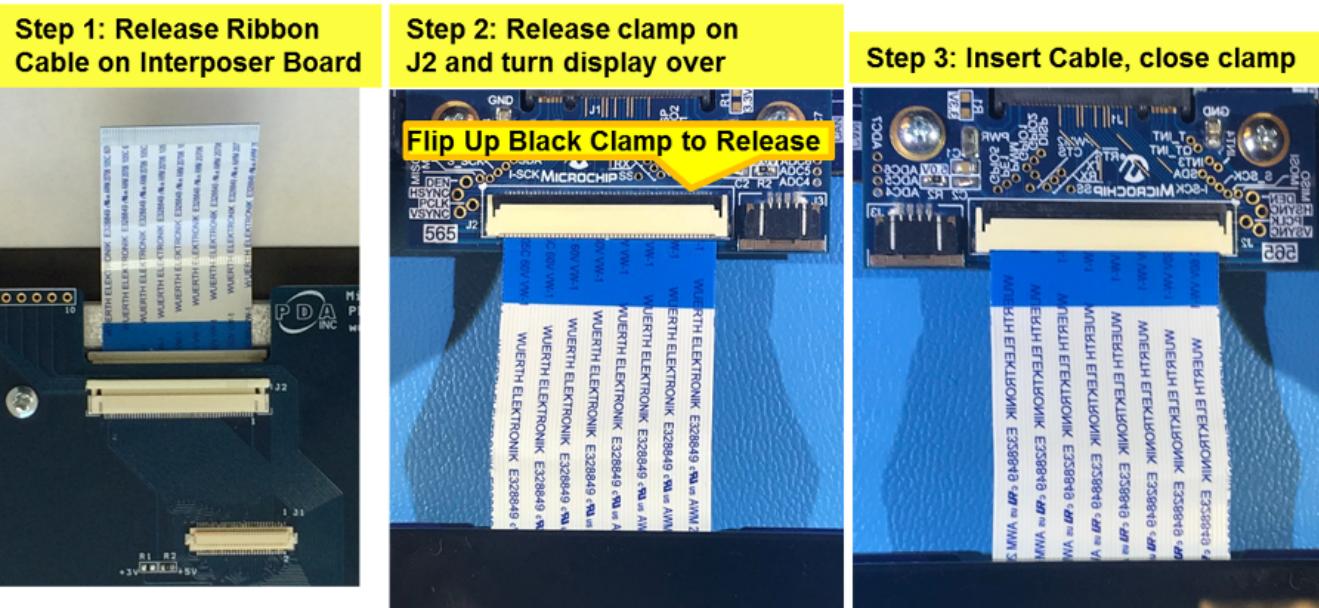
The final setup should be:



Configuring the 4.3" WQVGA Display requires disconnecting the ribbon cable that connects the display to the interposer board.



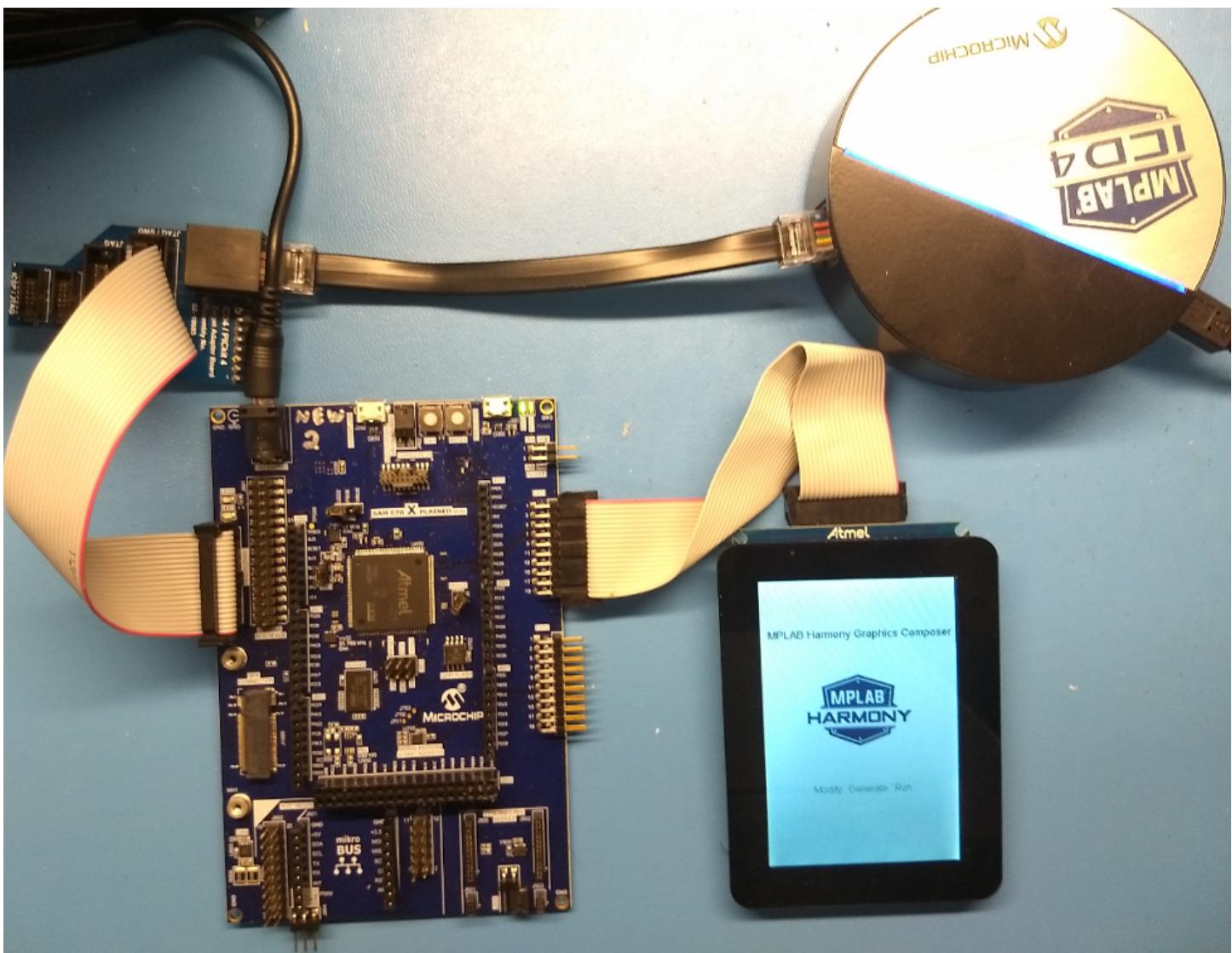
First, release the ribbon cable from the interposer board. Next, release the black clamp on the E70's J2 connector and turn the display over. Finally, insert the ribbon cable into J2 and close the clamp.



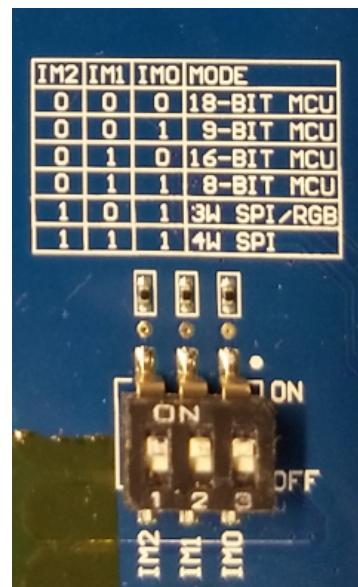
The board and display are powered by a Micro B – USB A cable from PC to the “Debug USB” port on the E70 board. The ICD4 Debugger and ICD4/PICKit4 Adapter Board are connected as shown above.

SAM E70 Xplained Ultra board with Xplained Pro Display

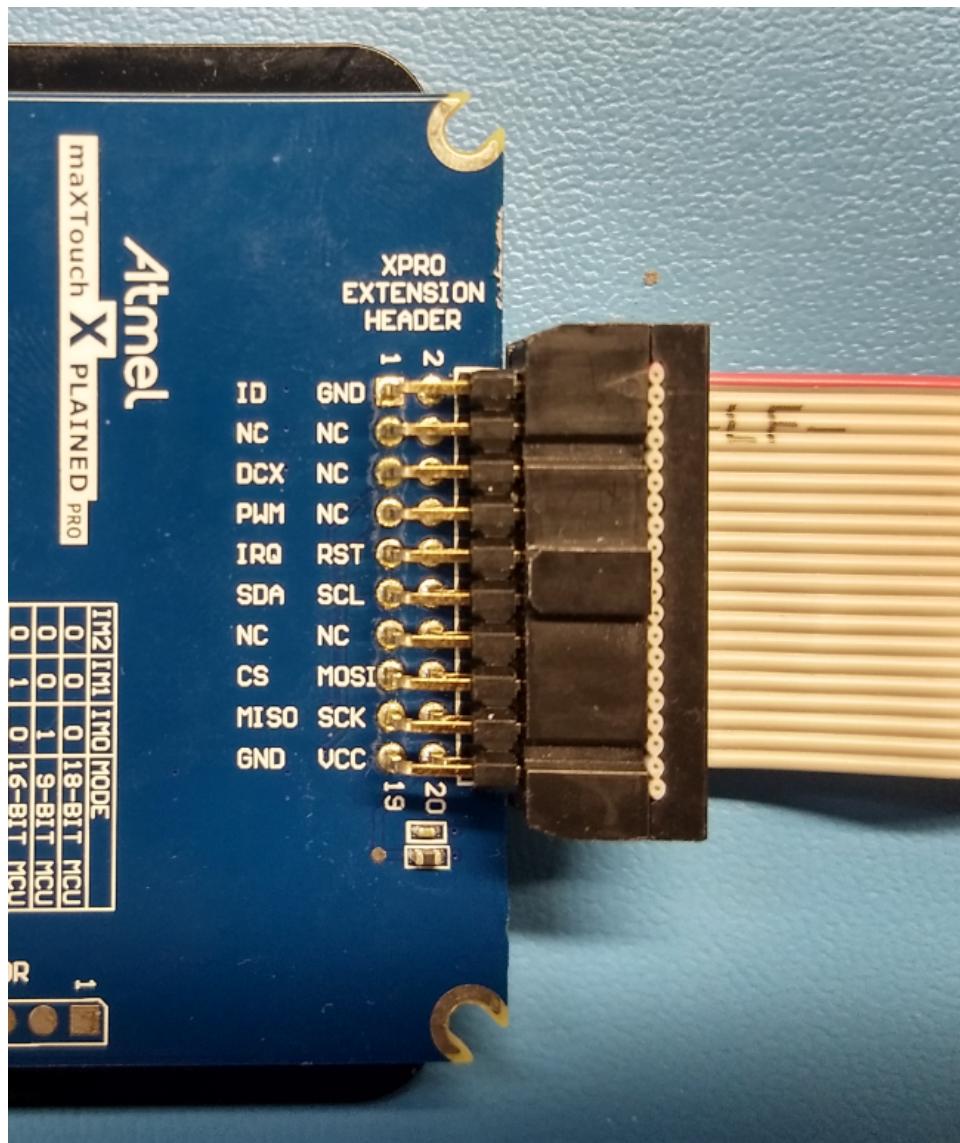
The final setup should be:



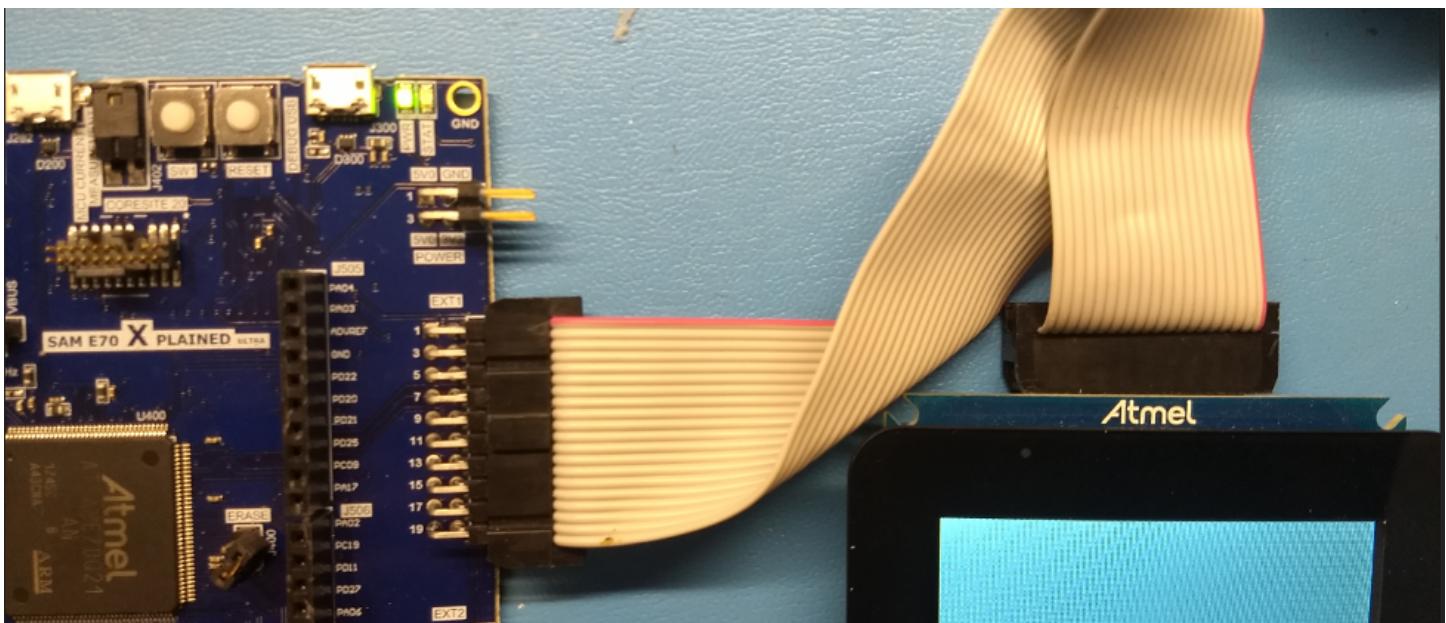
Set the IM switches on the maXTouch Xplained Pro Display to IM[2:0] = 111b:



Connect the Cable to the XPRO EXTENSION HEADER on the maXTouch Xplained Pro Display:



Connect the other end of the cable to the EXT1 header on the SAM E70 Xplained Ultra board:



Running the Demonstration

This section provides instructions about how to build and run the Aria Quick Start demonstration.

Description

When power-on is successful, the demonstration will display a similar menu to that shown in the following figure (different configurations may have slight variation in the screen aspect ratio):



When **Make changes. Generate. Run.** is touched, the button will toggle with each individual touch. (Note: The aria_qs_e70_xu_xpro_spi configuration does not support touch input.)



aria_showcase

This demonstration provides a subset of capabilities offered by the Aria Graphics Library using Low-Cost Controllerless features with touch screen capabilities.

Description

This application showcases the various widgets and advanced capabilities offered by the Aria User Interface Library. The application features an interactive menu screen where the user can access the different widget and graphics demonstrations. As a showcase application, it also features a 'Demo mode', where it autonomously runs the demonstrations after a specified period of idle time (typically 5 to 20 seconds). After idle time-out, the application will replay a series of prerecorded touch events, repeating the replay after a delay (typically 5 seconds). Event replay can be terminated by user initiation of a touch event and will not restart until another idle time out.

The application has Demo Mode enabled and will run autonomously if no touch input is detected after 20 seconds.

Architecture

The aria_showcase application uses the MPLAB Harmony Graphics Library to render graphics to the display. The graphics library draws the widgets and images to a frame buffer. The contents of the frame buffer are continuously transferred to refresh the

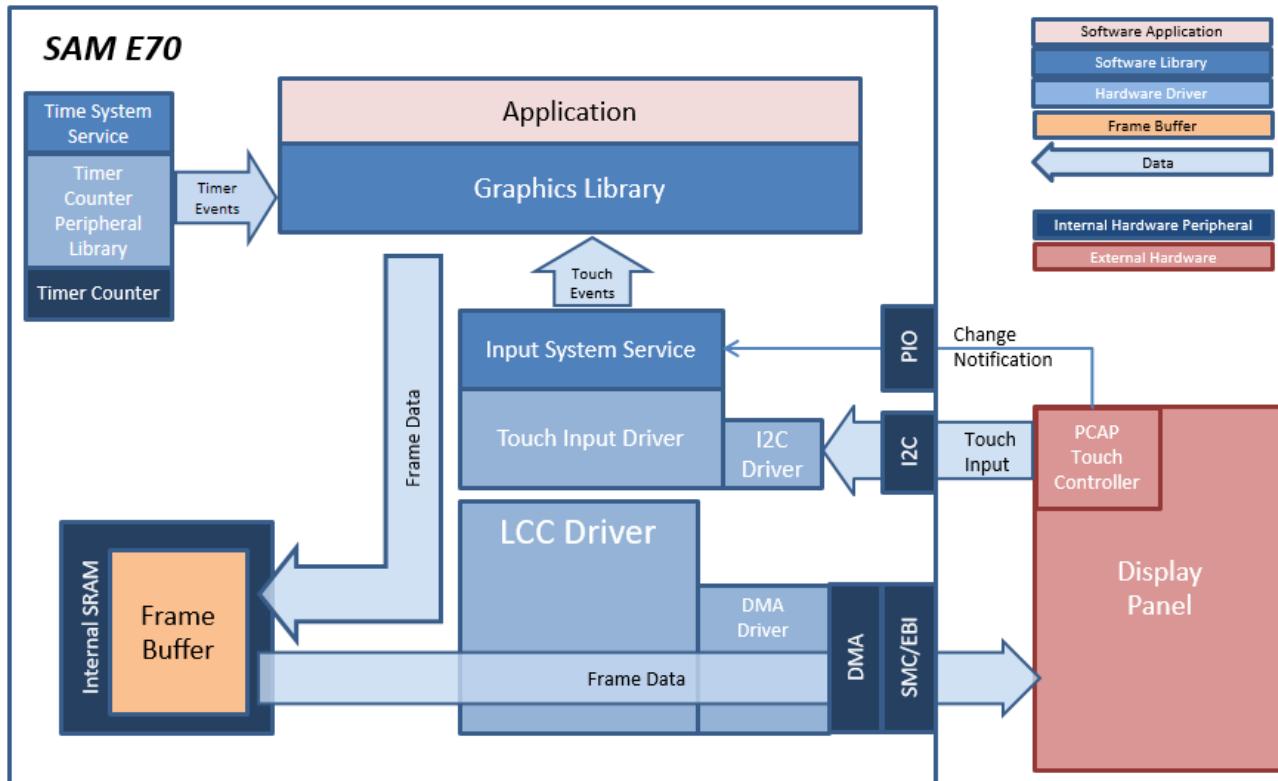
contents of the LCD display. The application also features user touch input through the integrated touch screen on the display panel. Touch input from the touch controller goes through the I2C port, and the Input System Service acquires the touch input information from the Touch and I2C Drivers. The Input System Service sends touch events to the Graphics Library, which processes these events and updates the frame data accordingly.

Finally, the application uses the Timer System Service and driver to send timer events for Demo mode and for transitioning images in the Slideshow Demo at specified intervals.

The block diagram(s) below show the various software and hardware blocks used in this application.

aria_sc_e70_xu_tm4301b

In this configuration, a 16-bit RGB565 frame buffer is stored in internal SRAM. Due to the limited size of the internal SRAM, only a single frame buffer can be used which can cause tearing to be visible as the GFX library draws on the screen. These configurations use the Low-Cost Controller-less (LCC) display driver to manage the DMA that transfers the framebuffer contents to the display.



Demonstration Features

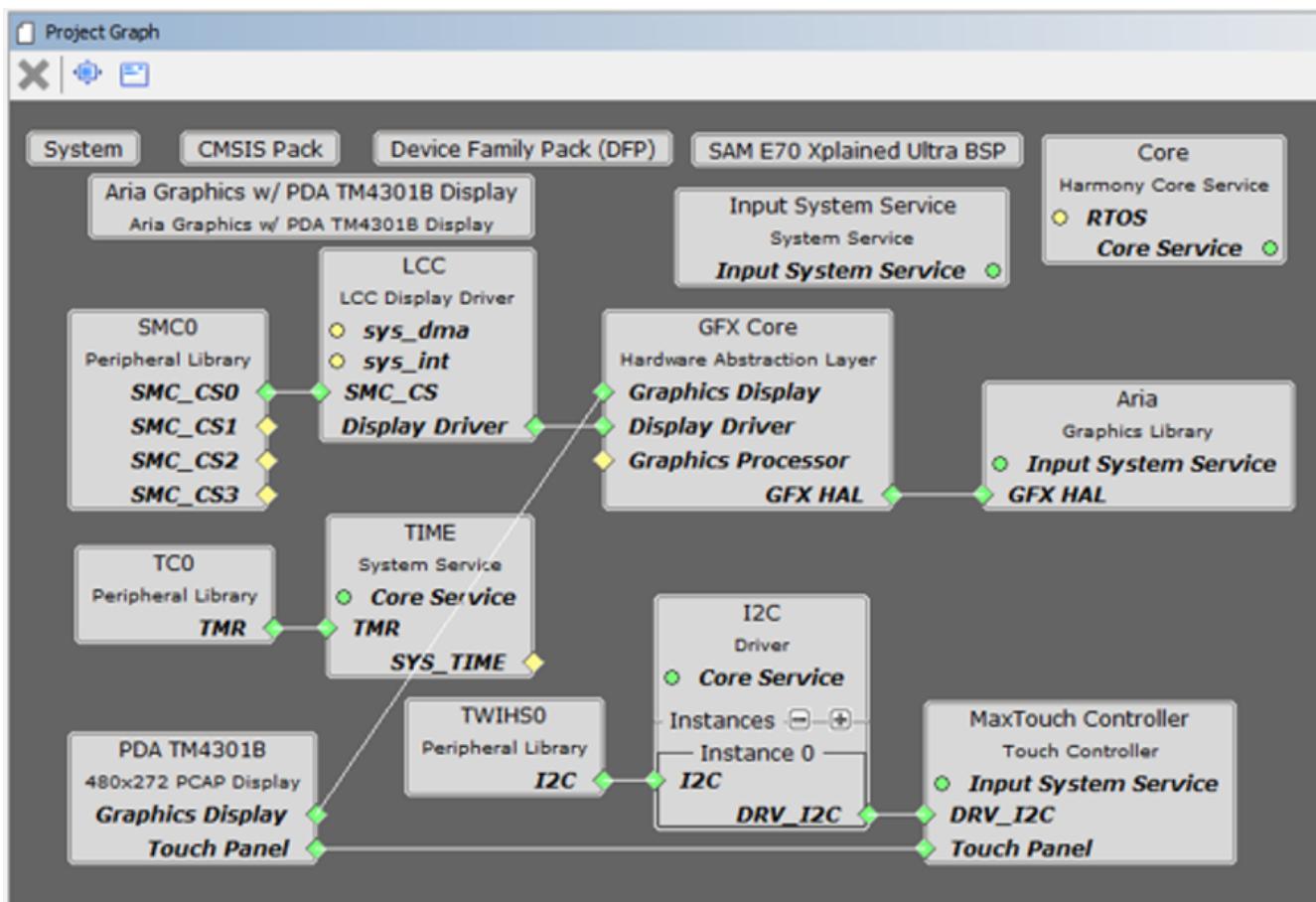
- Input system service and driver
- Time system service, timer-counter peripheral library and driver
- DMA System Service
- Low-Cost Controllerless (LCC) graphics driver
- I2C driver
- 16-bit RGB565 color depth support (65535 unique colors)
- JPEG and PNG images stored in internal flash
- UTF-8 and UTF-16 character font support
- Full multi-lingual font and localization (Chinese and English)
- Graphics Demo mode
- Graphics widgets: List wheel, Touch Test, Keypad, Slideshow
- Alpha-blending

MPLAB Harmony Configurator Setup

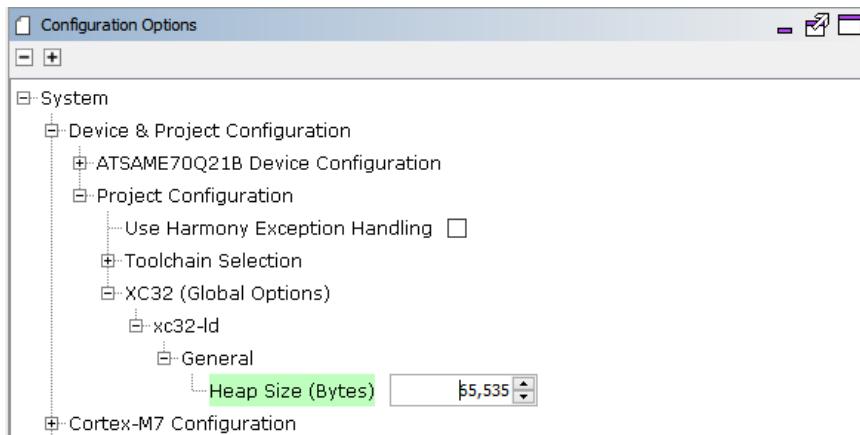
The Project Graph diagram below shows the Harmony components that are included in this application. Lines between

components are drawn to satisfy components that depend on a capability that another component provides.

Adding the “SAM E70 XPlained Ultra BSP” and the “Aria Graphics w/ PDA TM4301B Display” components into the project graph will automatically add the components needed for a graphics project and resolve their dependencies. It will also configure the pins needed to drive the external peripherals like the display and the touch controller.



The heap size is set to 65536 bytes. This is done by setting the Device & Project Configuration > Project Configuration > XC32 (Global Options) xc32-ld > General > Heap Size option for the “System” component:



Building the Application

This section identifies the MPLAB X IDE project name and location and lists and describes the available configurations for the Aria Showcase Reloaded demonstration.

Description

The parent directory for this application is gfx_apps/apps/aria_showcase. To build this application, open the gfx_apps/apps/aria_showcase/firmware/*.X project file in MPLABX IDE that corresponds to the hardware configuration.

MPLAB X IDE Project Configurations

The following table lists and describes the supported configurations:

Project Name	BSP Used	Graphics Template Used	Description
aria_sc_e70_xu_tm4301b.X	sam_e70_xplained_ultra	Aria Graphics w/ PDA TM4301B Display	SAM E70 Xplained Ultra board with PDA TM4301B 480x272 (WQVGA) Display



Important! This application may contain custom code that is marked by the comments // START OF CUSTOM CODE ... and // END OF CUSTOM CODE. When using the MPLAB Harmony Configurator to regenerate the application code, use the "ALL" merging strategy and *do not* remove or replace the custom code.

Configuring the Hardware

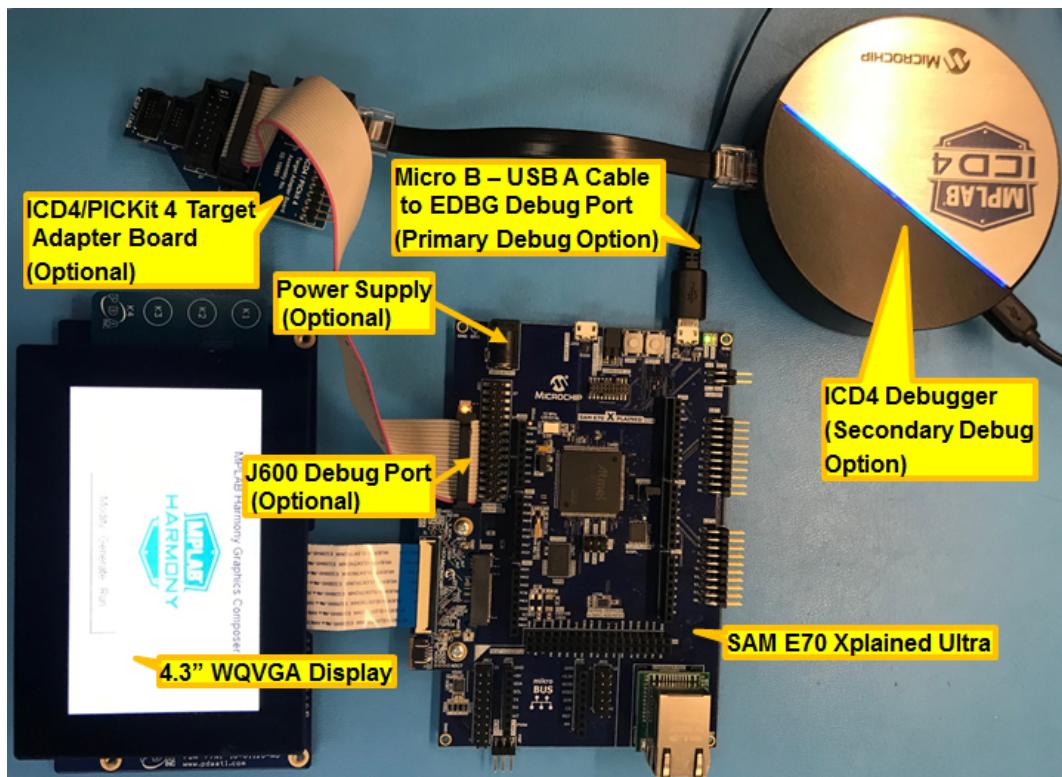
This section describes how to configure the supported hardware.

Description

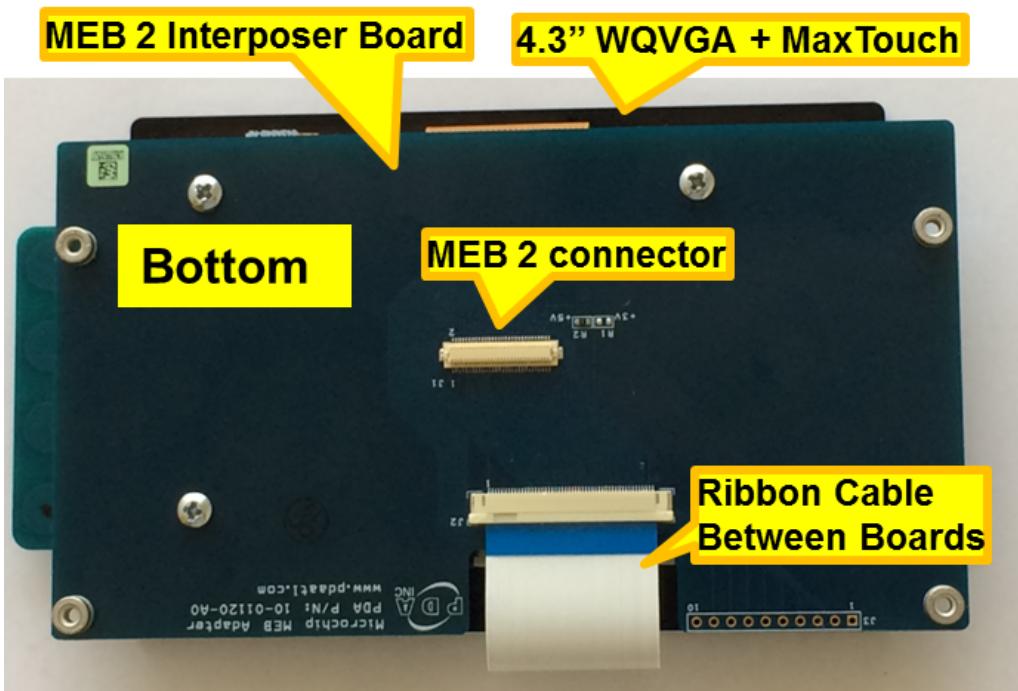
SAM E70 Xplained Ultra board with 4.3" WQVGA Display

There are two programming options. The primary option is the Micro B Embedded Debugger (EDBG) port. Power to the board can also be supplied via this connection. The secondary option is via the ICD4 debugger with the ICD4/PICKit 4 Target Adapter Board (Power has to be provided from power supply in this alternative setup).

The image below shows both the EDBG and the ICD4 connection options:

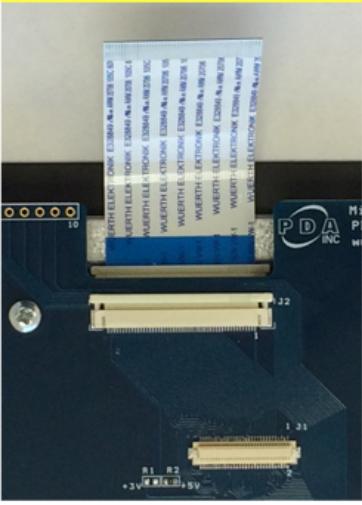


Configuring the 4.3" WQVGA Display requires disconnecting the ribbon cable that connects the display to the interposer board.



First, release the ribbon cable from the interposer board. Next, release the black clamp on the E70's J2 connector and turn the display over. Finally, insert the ribbon cable into J2 and close the clamp.

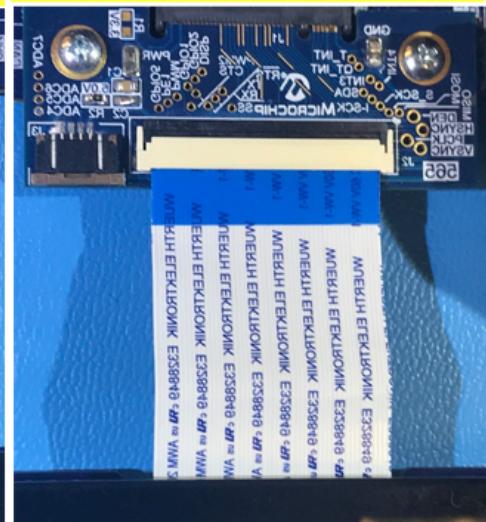
Step 1: Release Ribbon Cable on Interposer Board



Step 2: Release clamp on J2 and turn display over



Step 3: Insert Cable, close clamp



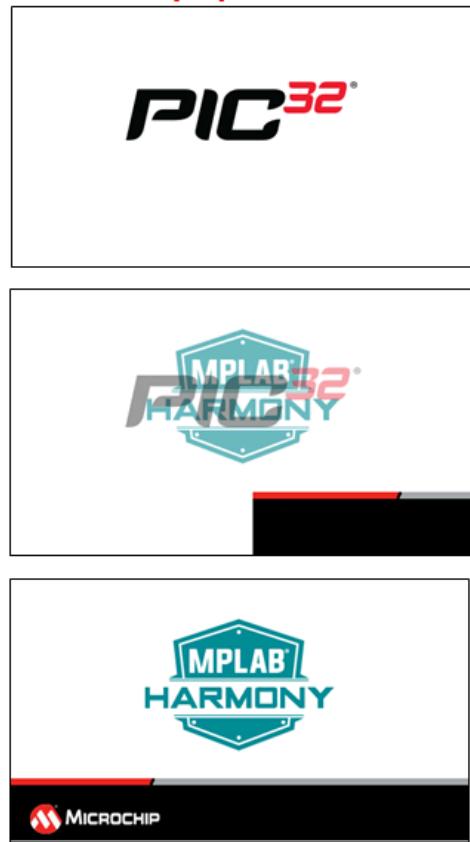
The board and display are powered by a Micro B – USB A cable from PC to the "Debug USB" port on the E70 board. The ICD4 Debugger and ICD4/PICKit4 Adapter Board are connected as shown above.

Running the Demonstration

This section provides instructions on how to use the Aria Showcase demonstration.

Description

Upon boot-up, the application displays the animation Splash Screen, which shows an animated splash bar while blending in the MPLAB Harmony and Microchip PIC32 logos, as shown in the following figure.

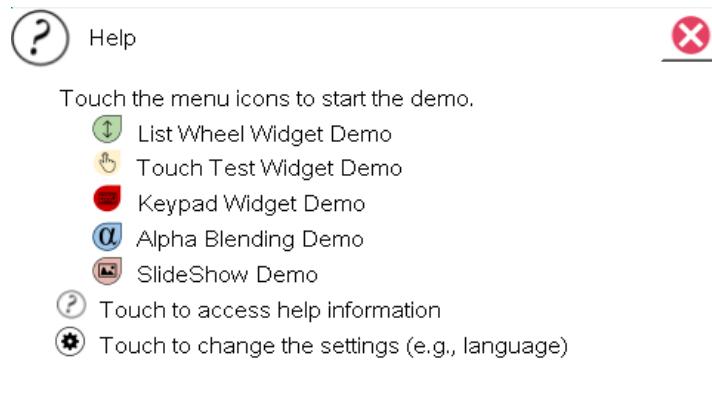


When power-on is successful, the application Home Screen will be displayed.



On the Home Screen, touching the large icons will open various screens that demonstrate the functionality of the widgets in the Aria graphics library.

Touching the Help (?) button on the lower left corner of each screen will show help information similar to the following figure.

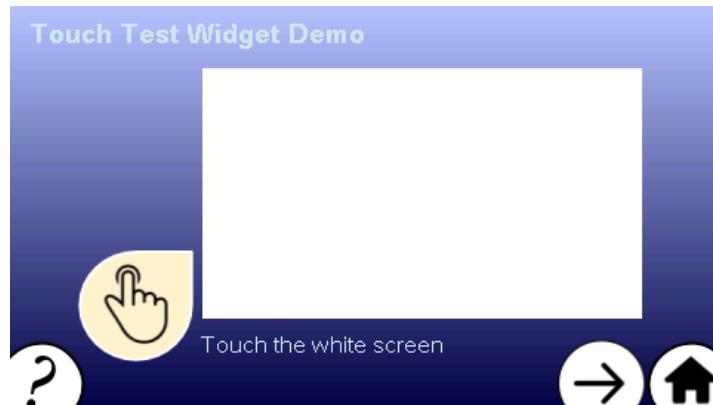


The Settings Screen shows an option to switch the language between English and Simplified Chinese and demonstrates the Aria string library's capability to support multi-lingual fonts and symbols.

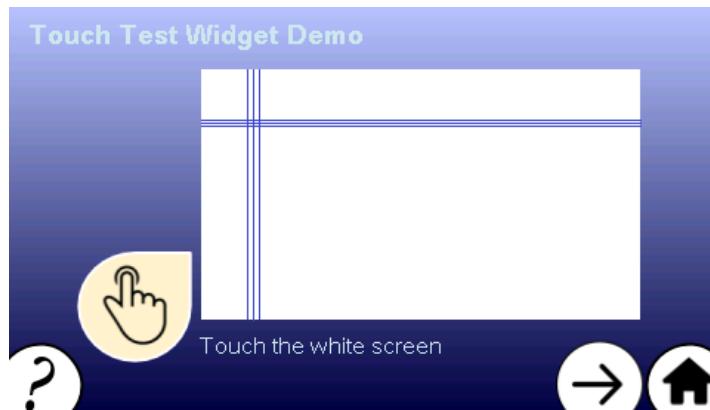


The List Wheel Widget Demo screen shows an example on how the list wheel widget can be used to provide UI controls for changing the time. The following figures show the List Wheel Widget Demo screen on PIC32MZ DA and PIC32MZ EF, respectively.

The Touch Test Widget Demo screen shows the touch screen functionality.



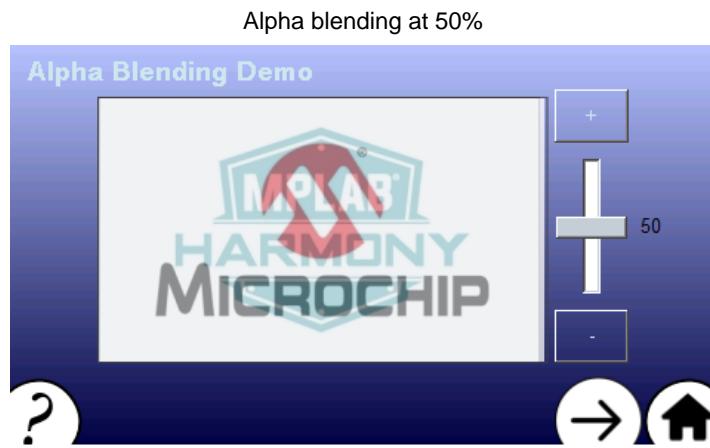
When the white active area of the Touch Test Widget is touched, intersecting horizontal and vertical lines indicate the touch points.



The Keypad Widget Demo Screen shows how the keypad and text entry widgets can be used to provide an interface for obtaining user text input. Touching the Text Field widget will show the Keypad for typing the text input.



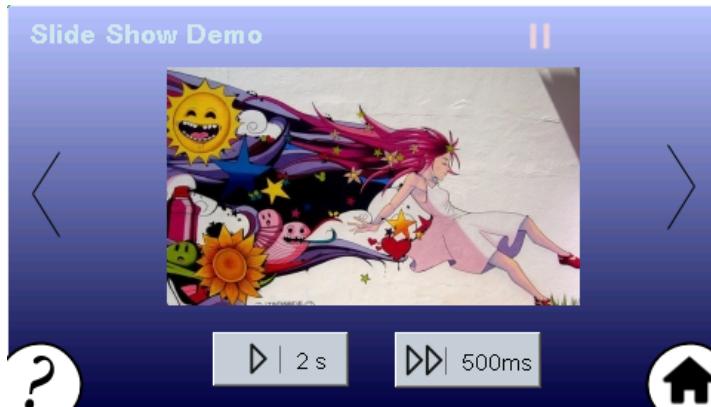
The Alpha Blending Demo screen shows the alpha blending capabilities of the Aria User Interface Library. The demonstration features two JPEG images that are alpha blended on top of each other. The (+) and (-) buttons and the slider widget on the right side of the images provides a way to change the alpha amounts.



Alpha blending at 100%

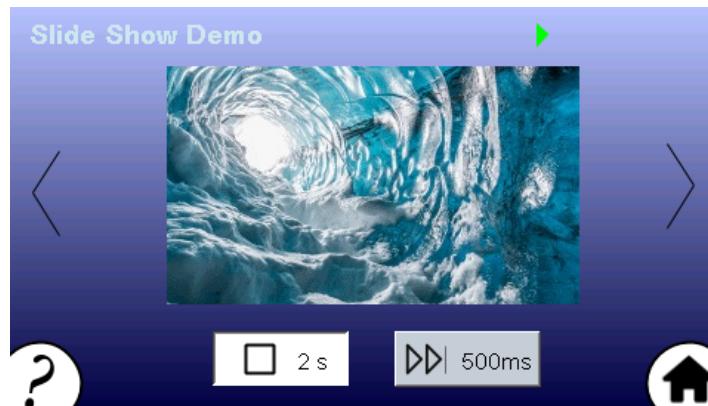


The Slide Show Demo Screen features the slide show widget being used to transition between several JPEG images using the available controls.

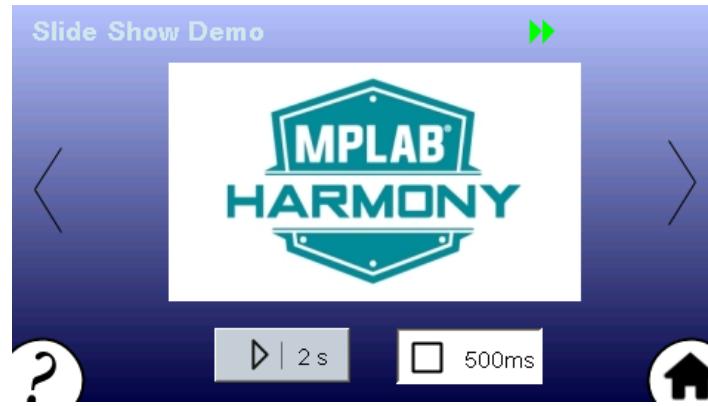


Touching the LEFT (<) and RIGHT (>) buttons manually transitions through the list of images. Touching the PLAY (>) and FAST-FORWARD (>>) buttons will automatically transition the images at 2s and 500ms intervals, respectively. The transitions are triggered using events from the Timer System Service.

The following figure shows Slide Show widget in Play mode.



The following figure shows the Slide Show widget in FAST-FORWARD mode.



Help information for the demonstration screens can be accessed by touching the (?) button on the lower left corner of each screen. Touching the HOME button on the lower right corner takes the application back to the Home Screen.

aria_showcase_reloaded

This application showcases the circular and graphing widgets – arc, circular slider, circular gauge, pie chart, bar graph, line graph. These widgets are available and ready-to-use using the Harmony Graphics Composer suite.

Description

This application showcases the circular and graphing widgets – arc, circular slider, circular gauge, pie chart, bar graph, line graph.

- **Arc** – a primitive drawing widget that can be used to draw filled arcs or circles.
- **Circular Slider** – a circular widget that takes user input to set/show a value within a specified range.
- **Circular Gauge** – a circular widget that shows a value within a range using a needle and tick marks.
- **Pie Chart** – a graphing widget that shows data values as sectors in a circle.
- **Bar Graph** – a graphing widget that shows data values in categories using rectangular bars.
- **Line Graph** – a graphing widget that shows data values in categories using points and lines.

Each widget is demonstrated on a separate application screen. Touching the screen or an applicable widget button will show each widget change in form or value.

The application has Demo Mode enabled, and will run autonomously if no touch input is detected after 20 seconds.

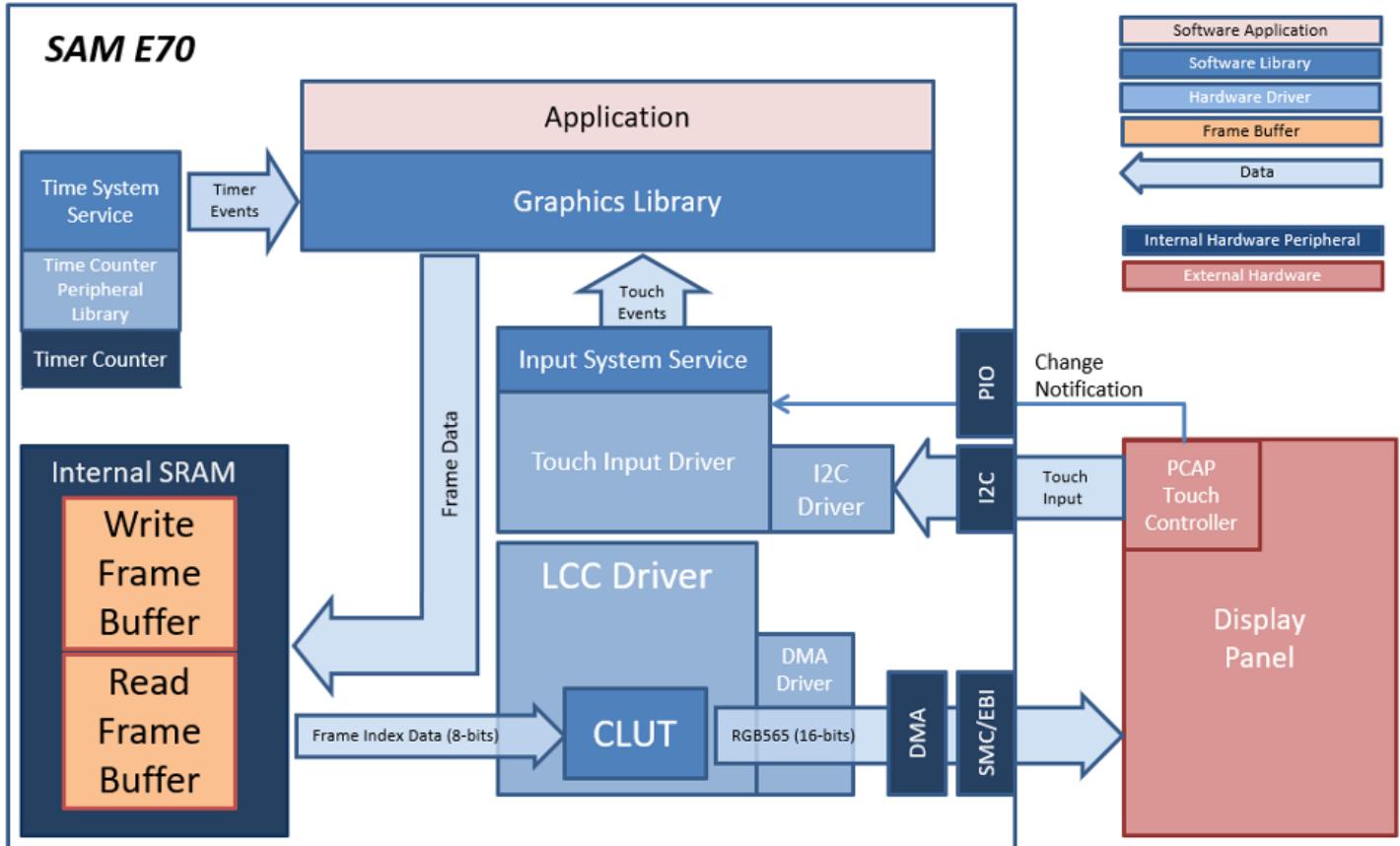
Architecture

The block diagrams below show the various software and hardware blocks used in this application.

In all configurations, the graphics library renders the widgets to the frame buffer(s), which is periodically refreshed to the display. The graphics library also processes user input via the touch screen and sends events to the application code which updates the widgets appropriately.

aria_scr_e70_xu_tm4301b_lut8

Since two 16-bit RGB565 frame buffers cannot fit into the internal device SRAM, this configuration uses two 8-bit buffers to support double-buffering. The 8-bit buffers contain the indices of 16-bit colors in a palette lookup table (CLUT). The graphics library renders the widgets by writing the index of a pixel color into the buffer. During a display line refresh, the LCC (low-cost controllerless) driver performs a palette lookup to convert the 8-bit indices to their 16-bit color equivalent for each line and then writes the line data to the display via DMA. Using 8-bit palettized buffers allows for double-buffering and eliminates tearing during rendering. On the other hand, the lookup table conversion requires extra processing time and reduces performance.

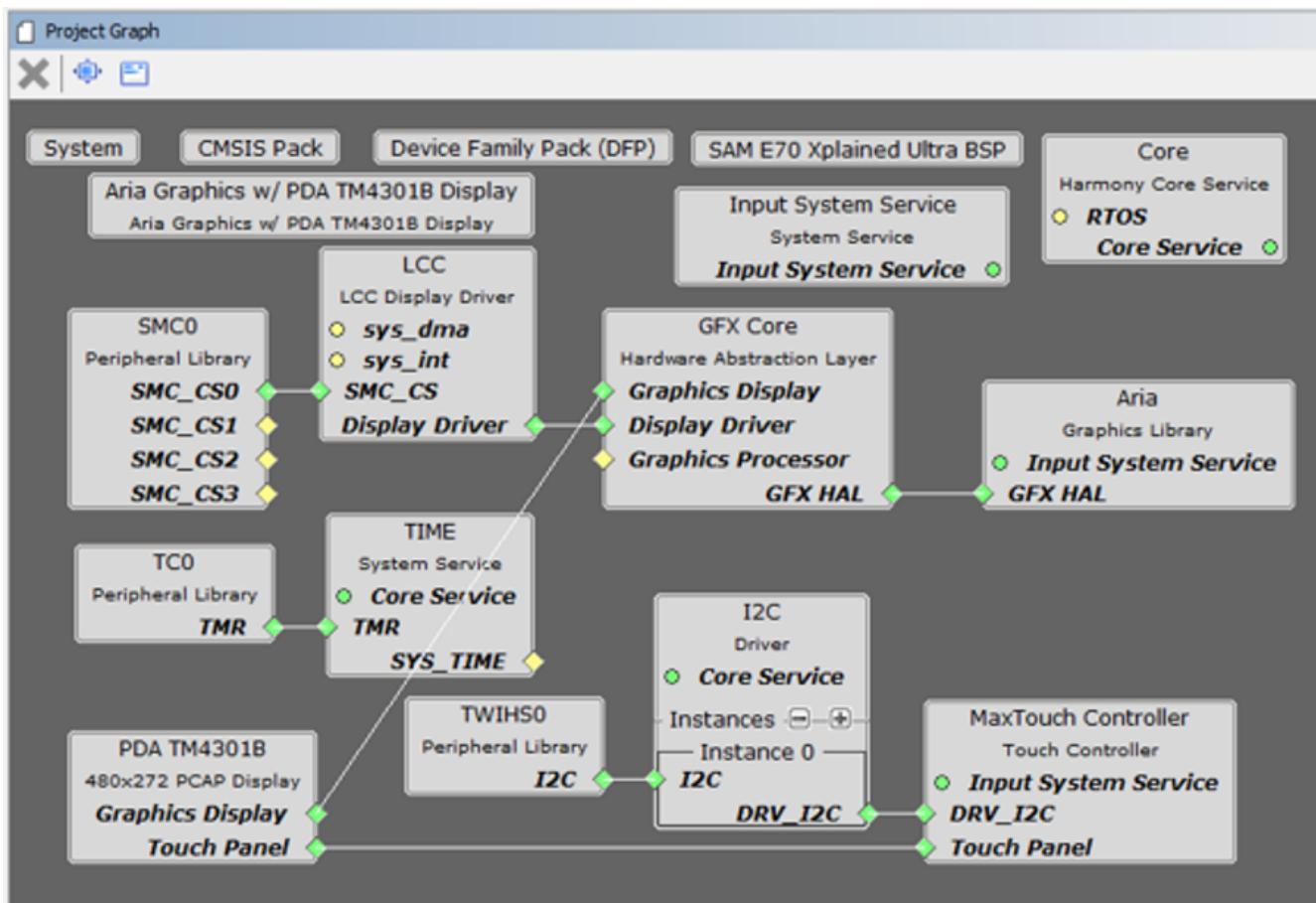
**Demonstration Features**

- Circular widgets – arc, circular slider, circular gauge
- Graphing widgets – pie chart, bar graph, line graph
- Timer HW and Timer System Service
- Image compression techniques using Run-Length Encoding
- Double-buffering using LUT (lookup-table) based frame buffers

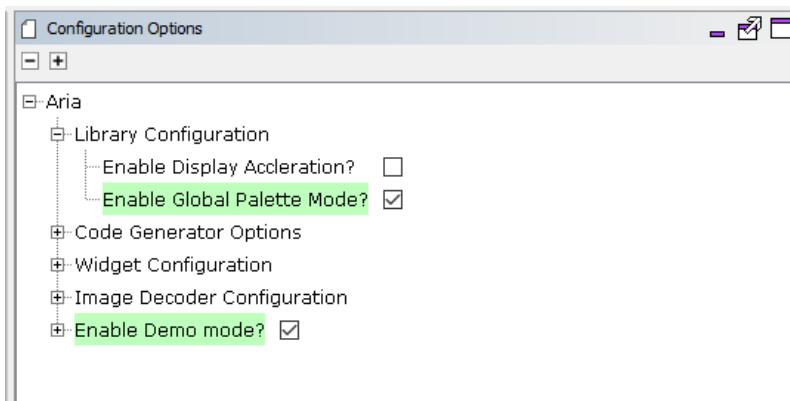
MPLAB Harmony Configurator Setup

The Project Graph diagram below shows the Harmony components that are included in this application. Lines between components are drawn to satisfy components that depend on a capability that another component provides.

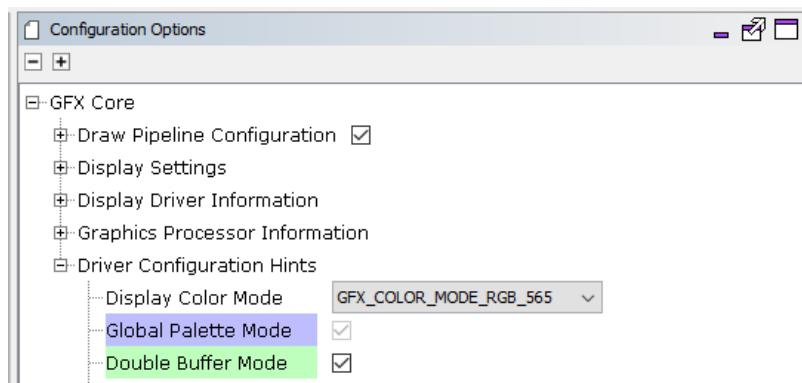
Adding the “SAM E70 XPlained Ultra BSP” and the “Aria Graphics w/ PDA TMA4301B Display” components into the project graph will automatically add the components needed for a graphics project and resolve their dependencies. It will also configure the pins needed to drive the external peripherals like the display and the touch controller.



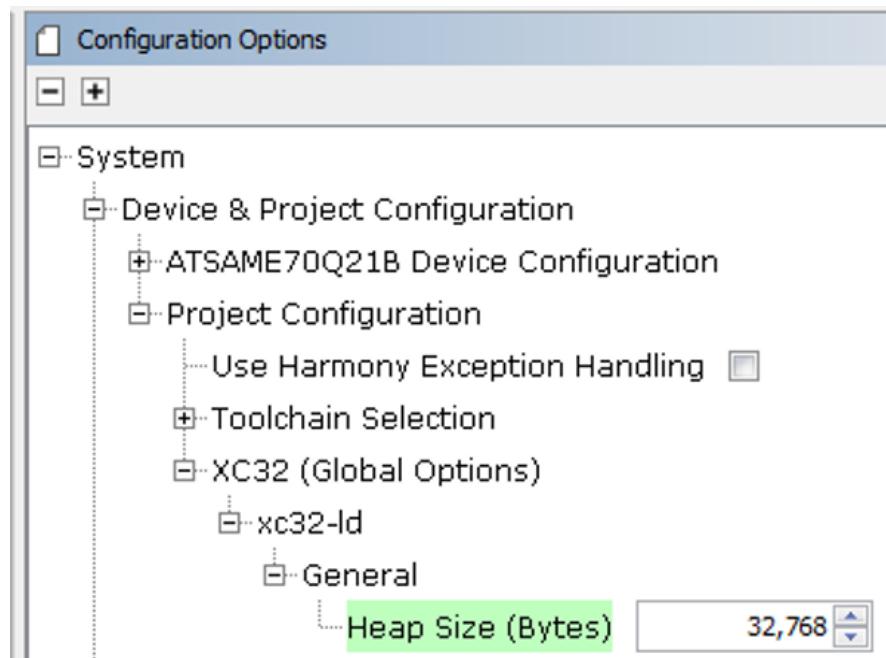
Enable the global 8-bit color palette by selecting the Aria Graphics Library component:



For the GFX Core component enable double buffering. (Global Palette Mode is enabled and grayed out since it is inherited from the Aria Graphics Library component.)



The heap size is set to 32768 bytes. This is done by setting the Device & Project Configuration > Project Configuration > XC32 (Global Options) xc32-ld > General > Heap Size option for the "System" component:



Building the Application

This section identifies the MPLAB X IDE project name and location and lists and describes the available configurations for the Aria Showcase Reloaded demonstration.

Description

The parent directory for this application is gfx_apps/apps/aria_showcase_reloaded. To build this application, open the gfx_apps/apps/aria_showcase_reloaded/firmware/*.X project file in MPLABX IDE that corresponds to the hardware configuration.

MPLAB X IDE Project Configurations

The following table lists and describes the supported configurations:

Project Name	BSP Used	Graphics Template Used	Description
aria_scr_e70_xu_tm4301b_lut8.X	sam_e70_xplained_ultra	Aria Graphics w/ PDA TM4301B Display	SAM E70 Xplained Ultra board with PDA TM4301B 480x272 (WQVGA) Display



Important! This application may contain custom code that is marked by the comments // START OF CUSTOM CODE ... and // END OF CUSTOM CODE. When using the MPLAB Harmony Configurator to regenerate the application code, use the "ALL" merging strategy and *do not* remove or replace the custom code.

Configuring the Hardware

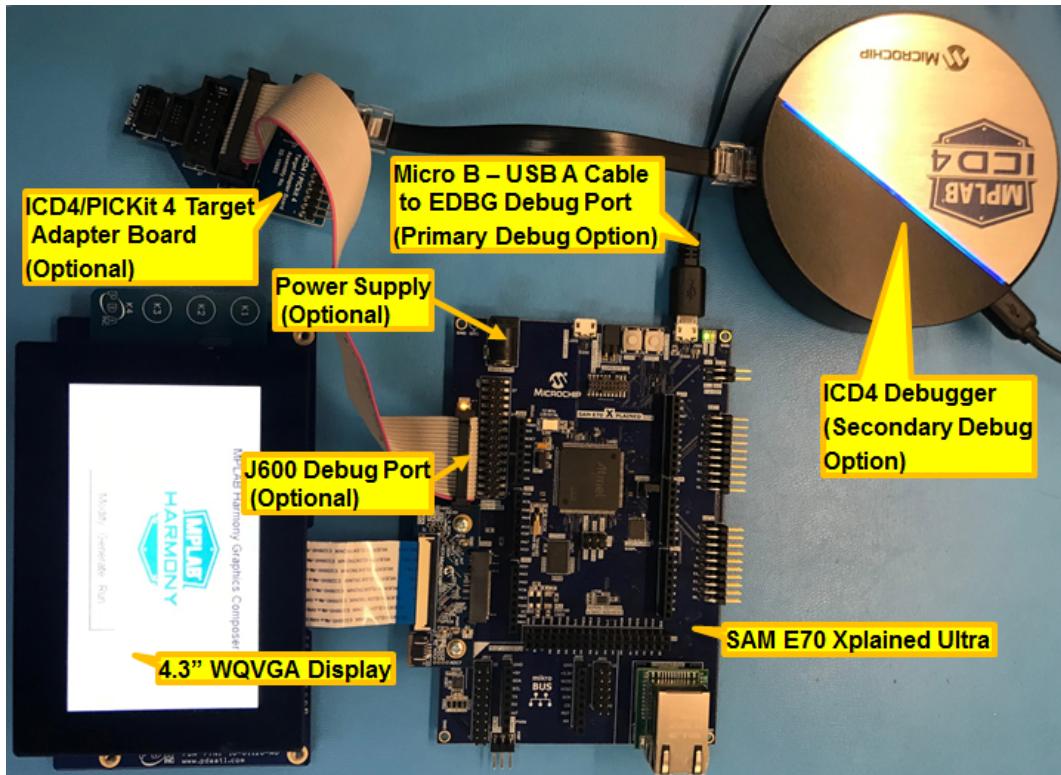
This section describes how to configure the supported hardware.

Description

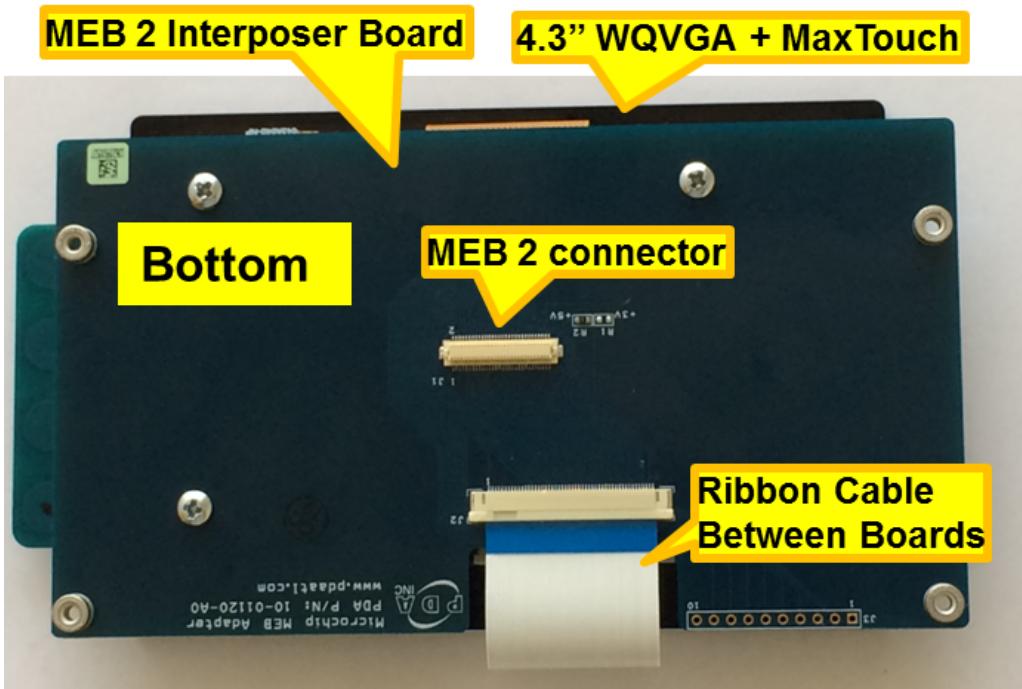
SAM E70 Xplained Ultra board with 4.3" WQVGA Display

There are two programming options. The primary option is the Micro B Embedded Debugger (EDBG) port. Power to the board can also be supplied via this connection. The secondary option is via the ICD4 debugger with the ICD4/PICKit 4 Target Adapter Board (Power has to be provided from power supply in this alternative setup).

The image below shows both the EDBG and the ICD4 connection options:

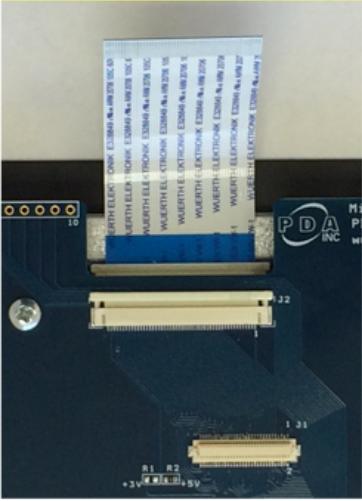


Configuring the 4.3" WQVGA Display requires disconnecting the ribbon cable that connects the display to the interposer board.



First, release the ribbon cable from the interposer board. Next, release the black clamp on the E70's J2 connector and turn the display over. Finally, insert the ribbon cable into J2 and close the clamp.

Step 1: Release Ribbon Cable on Interposer Board



Step 2: Release clamp on J2 and turn display over



Step 3: Insert Cable, close clamp



The board and display are powered by a Micro B – USB A cable from PC to the "Debug USB" port on the E70 board. The ICD4 Debugger and ICD4/PICKit4 Adapter Board are connected as shown above.

Running the Demonstration

This section provides information on how to run and use the application.

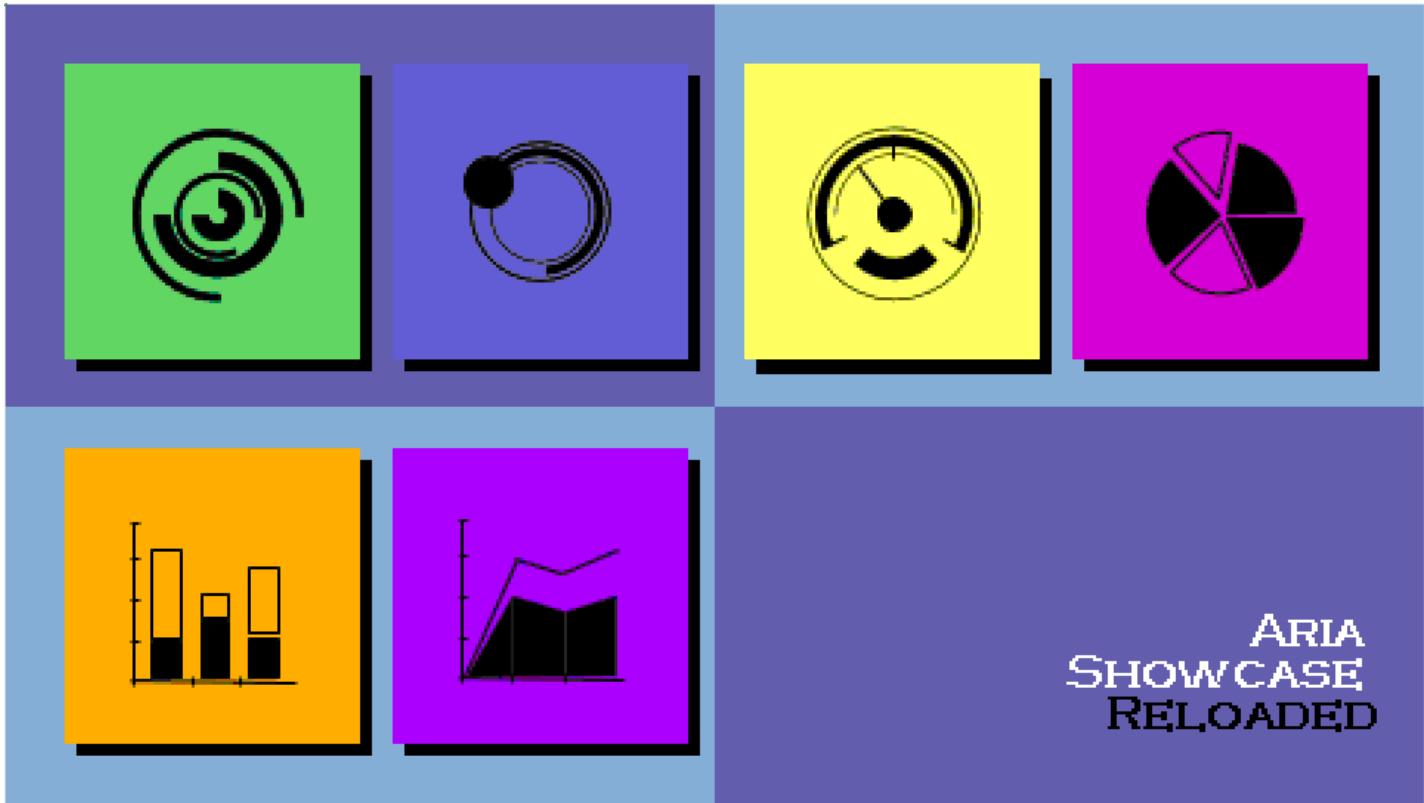
Description

On start-up, the application will display a splash screen.

PIC³²[®]

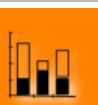


After the splash screen completes, the Main Menu will display.



The Main Menu buttons operate as follows:

Button	Description
A green square containing a black circular arrow icon.	Arc Demo Button – Opens the Arc Widget Demonstration Screen.

	Circular Slider Demo Button – Opens the Circular Slider Demonstration Screen.
	Circular Gauge Demo Button – Opens the Circular Gauge Demonstration Screen.
	Pie Chart Demo Button – Opens the Pie Chart Demonstration Screen.
	Bar Graph Demo Button – Opens the Bar Graph Demonstration Screen
	Line Graph Demo Button – Opens the Line Graph Demonstration Screen

Touching one of the buttons above opens the corresponding widget demonstration screen. Each demonstration screen will have the following buttons:

Button	Description
	Home Button – Opens the Main Menu Screen
	Next Button – Opens the Next Demonstration Screen

Touching the Arc Demo Button on the Main Menu Screen brings up the Arc Widget Demonstration Screen, as shown in the following figure. Touching the screen will show animated, rotating concentric arcs and circles. This demonstrates the graphics library's ability to draw arcs of varying thickness, angles and colors.

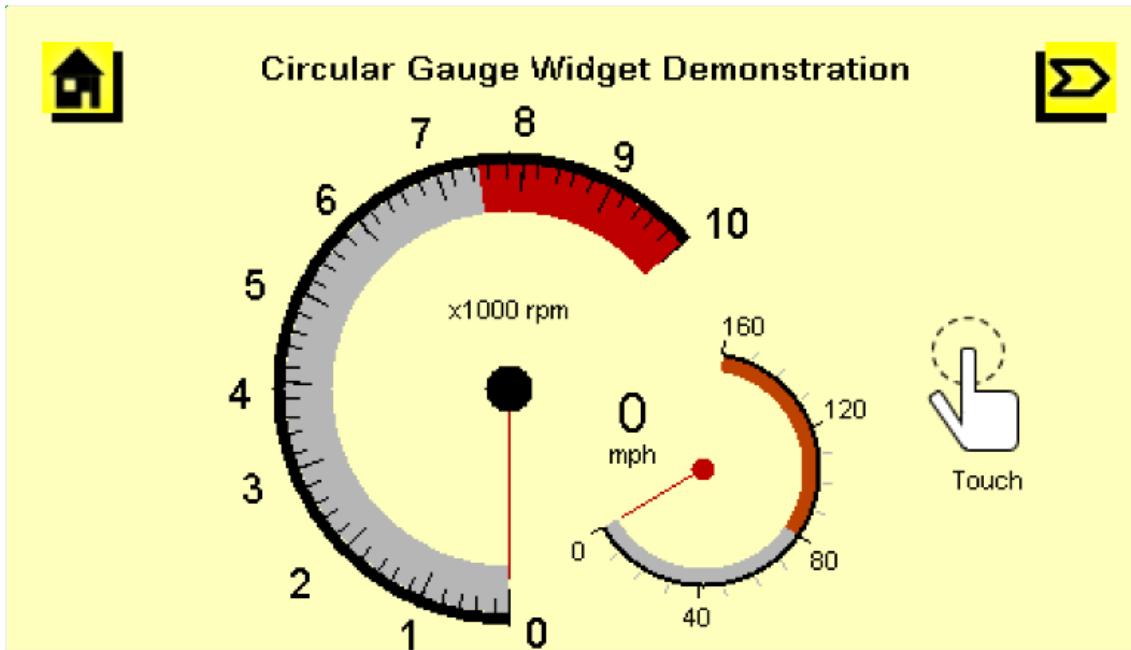


Touching the Circular Slider Button on the Main Menu or the Next Button on the Arc Widget Demonstration Screen will open the Circular Slider Demonstration Screen, as shown below. The screen contains a Circular Slider widget on the right, and Circular Progress Bar widget on the left. The Circular Progress Bar widget is a form of the Circular Slider widget with no button and does not respond to touch input.

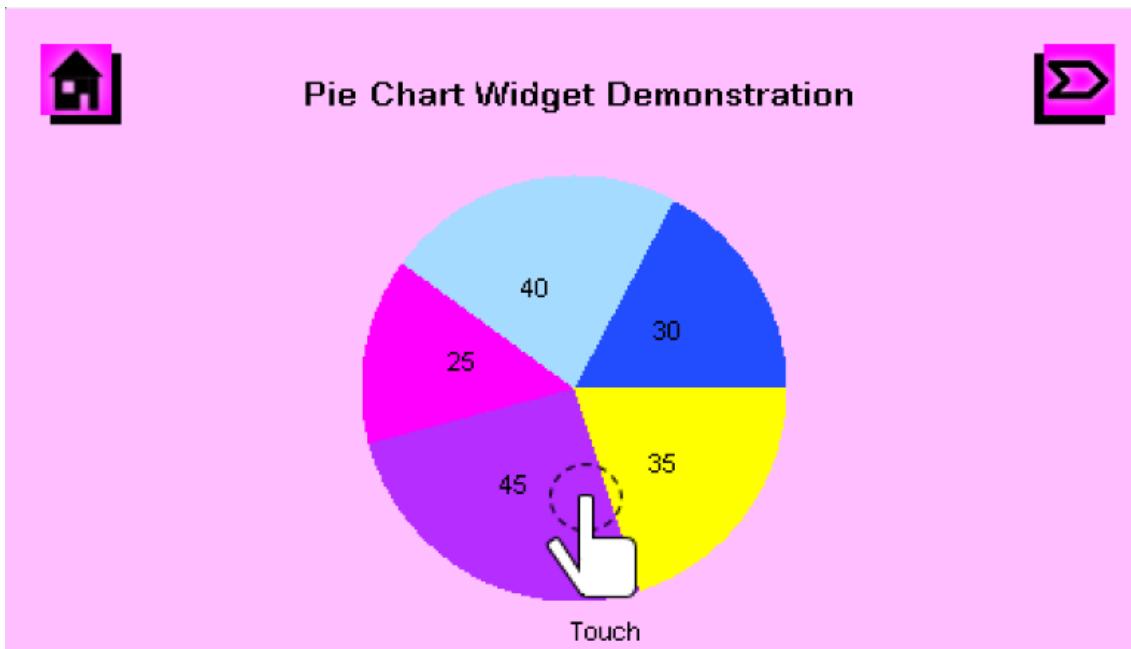
Touching and moving the button on the Circular Slider widget clockwise or counter-clockwise, will decrease or increase its value. This value is assigned to the Circular Progress Bar widget, which updates based on the set value, and to a label widget inside it to show the numeric value.



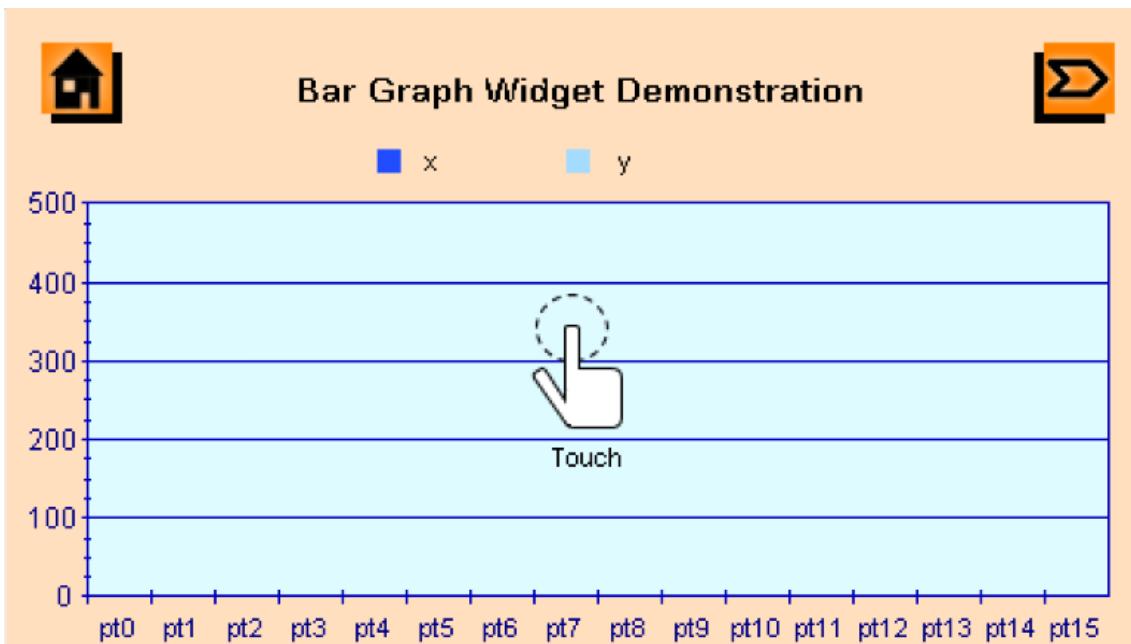
Touching the Circular Gauge button on the Main Menu screen or the Next button on the Circular Slider Demonstration screen opens the Circular Gauge Demo Screen, as shown in the following figure. This simulates a vehicle dashboard using two Circular Gauge widgets as tachometer and speedometer. Touching the screen increases the value of the two gauges.



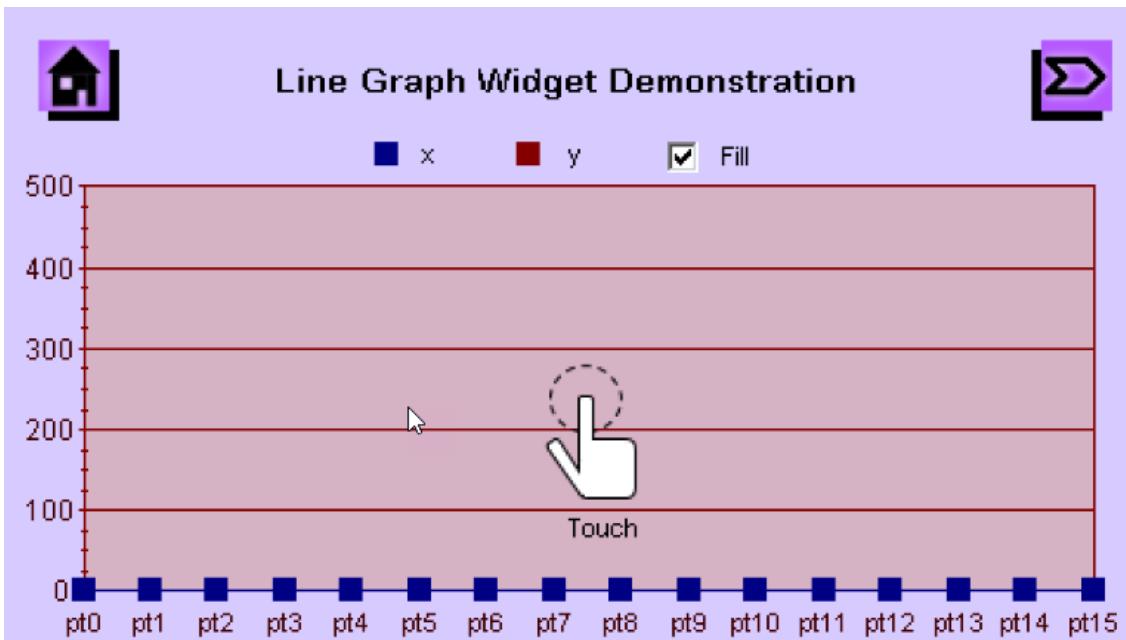
Touching the Pie Chart button on the Main Menu or the Next button on the Circular Gauge Widget Demonstration screen opens the Pie Chart Demonstration Screen, as shown in the following figure. Touching a slice on the Pie Chart widget will change the slice's radius and offset from the center.



Touching the Bar Graph Button on the Main Menu, or the Next Button on the Pie Chart Demonstration Screen will open the Bar Graph Demonstration Screen, as shown in the following figure. Touching the screen will plot the x and y coordinates of the touch point in the bar graph widget.



Touching the Line Graph Button on the Main Menu or the Next Button on the Bar Graph Demonstration Screen will open the Line Graph Demonstration Screen, as shown in the following figure. Touching the screen will plot the x and y coordinates of the touch points in the line graph widget.



aria_weather_forecast

This demonstration provides a practical single-layered single-buffered application using the Aria User Interface Library.

Description

This application showcases an example of how a graphics controller with single graphical layer support can be achieved using the Aria Graphics Library. It also highlights how tightly integrated the MPLAB Harmony Graphics Composer Suite can be in supporting a rich feature set in an application.

Various Aria Graphics Library features such as multi-language localization support and tree-based architecture are highlighted in this demonstration, which is disguised as a user-interface for weather forecast application.

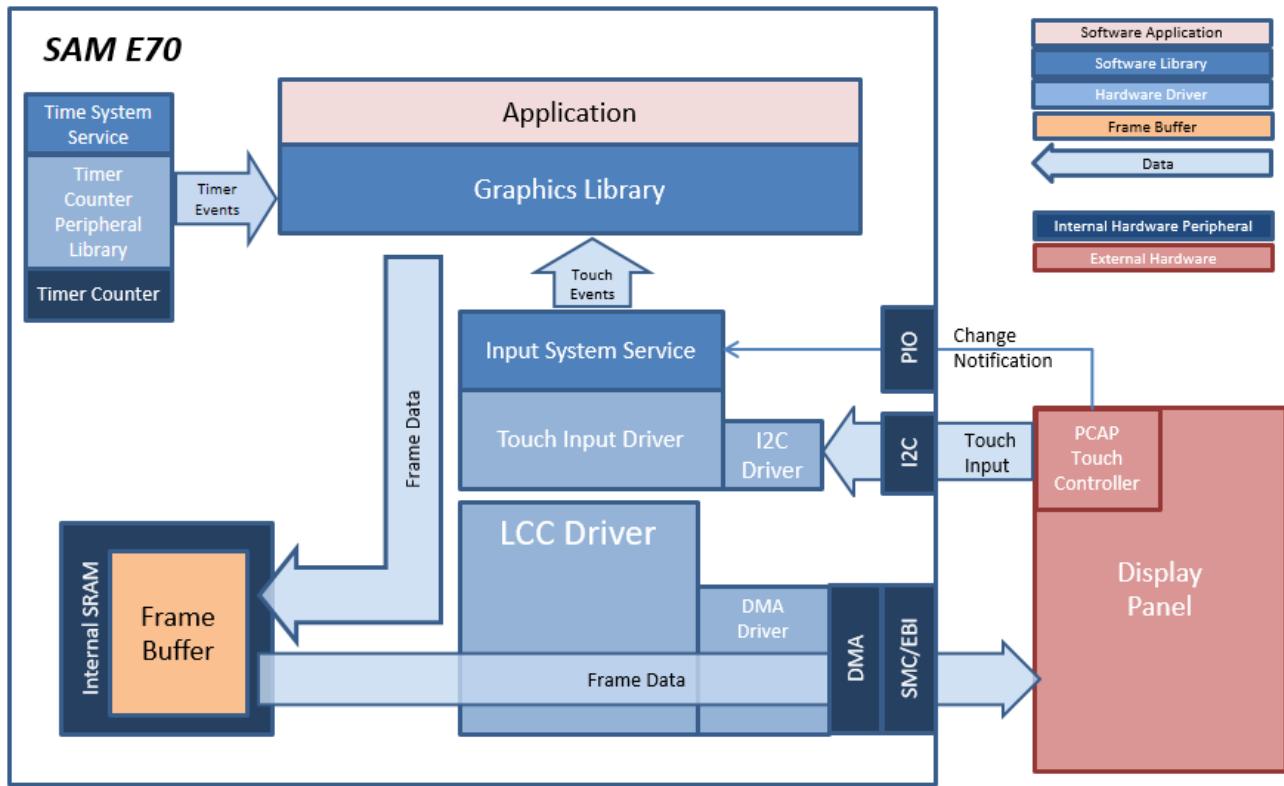
The demonstration launches with a splash screen highlighting basic motion capability supported by Aria Graphics Library. When running the application, the user can interface with it via capacitive single-fingered touch.

Architecture

The following diagrams show the various software and hardware components for each configuration.

aria_wf_e70_xu_tm4301b

In this configuration, a 16-bit RGB565 frame buffer is stored in internal SRAM. Due to the limited size of the internal SRAM, only a single frame buffer can be used which can cause tearing to be visible as the GFX library draws on the screen. These configurations use the Low-Cost Controller-less (LCC) display driver to manage the DMA that transfers the framebuffer contents to the display.



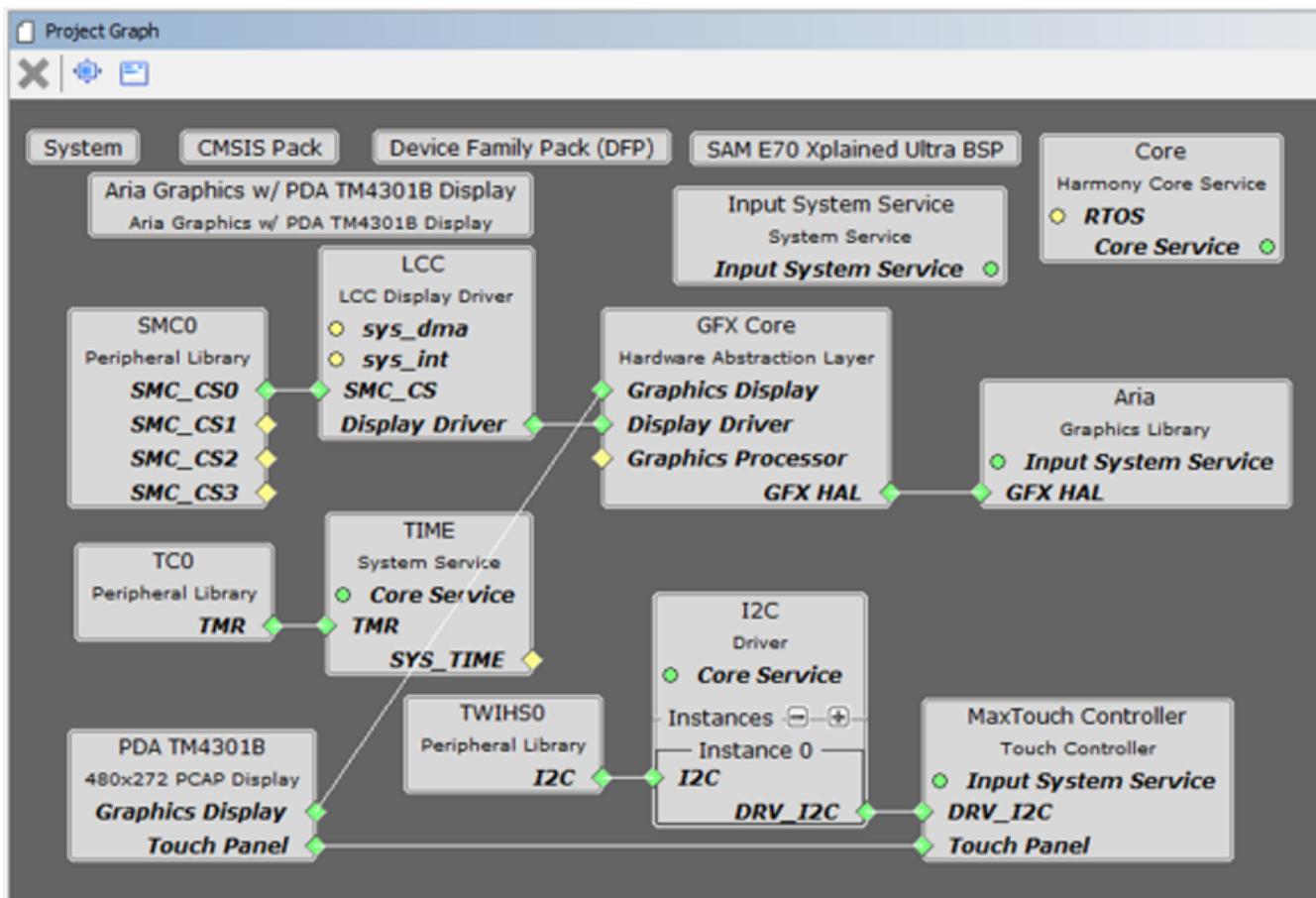
Demonstration Features

- Low-Cost Controllerless (LCC) Graphics driver on the SAM E70 device
- Language localization in English, Chinese Simplified, and Spanish [add link to (see *MPLAB Harmony Graphics Library Help > MPLAB Harmony Graphics Composer User's Guide > Graphics Composer Window User Interface > Widget Tool Box Panel*)]
- Image compression techniques using Run-Length Encoding and JPEG (see [Enabling Demo Mode in a MPLAB Harmony Graphics Application](#))
- Run-time JPEG decoding
- UTF-8 character font support (see *MPLAB Harmony Graphics Library Help > MPLAB Harmony Graphics Composer User's Guide > Graphics Composer Window User Interface > Font Assets*)
- Button Widget
- Image Widget
- Bar Graph Widget
- Line Graph Widget

MPLAB Harmony Configurator Setup

The Project Graph diagram below shows the Harmony components that are included in this application. Lines between components are drawn to satisfy components that depend on a capability that another component provides.

Adding the “SAM E70 XPlained Ultra BSP” and the “Aria Graphics w/ PDA TMA4301B Display” components into the project graph will automatically add the components needed for a graphics project and resolve their dependencies. It will also configure the pins needed to drive the external peripherals like the display and the touch controller.



Building the Application

This section identifies the MPLAB X IDE project name and location and lists and describes the available configurations for the Aria Showcase Reloaded demonstration.

Description

The parent directory for this application is gfx_apps/apps/aria_weather_forecast. To build this application, open the gfx_apps/apps/aria_weather_forecast/firmware/*.X project file in MPLABX IDE that corresponds to the hardware configuration.

MPLAB X IDE Project Configurations

The following table lists and describes the supported configurations:

Project Name	BSP Used	Graphics Template Used	Description
aria_wf_e70_xu_tm4301b.X	sam_e70_xplained_ultra	Aria Graphics w/ PDA TM4301B Display	SAM E70 Xplained Ultra board with PDA TM4301B 480x272 (WQVGA) Display



Important! This application may contain custom code that is marked by the comments // START OF CUSTOM CODE ... and // END OF CUSTOM CODE. If you use the MPLAB Harmony Configurator to regenerate the application code, use the "ALL" merging strategy and do not remove or replace the custom code.

Configuring the Hardware

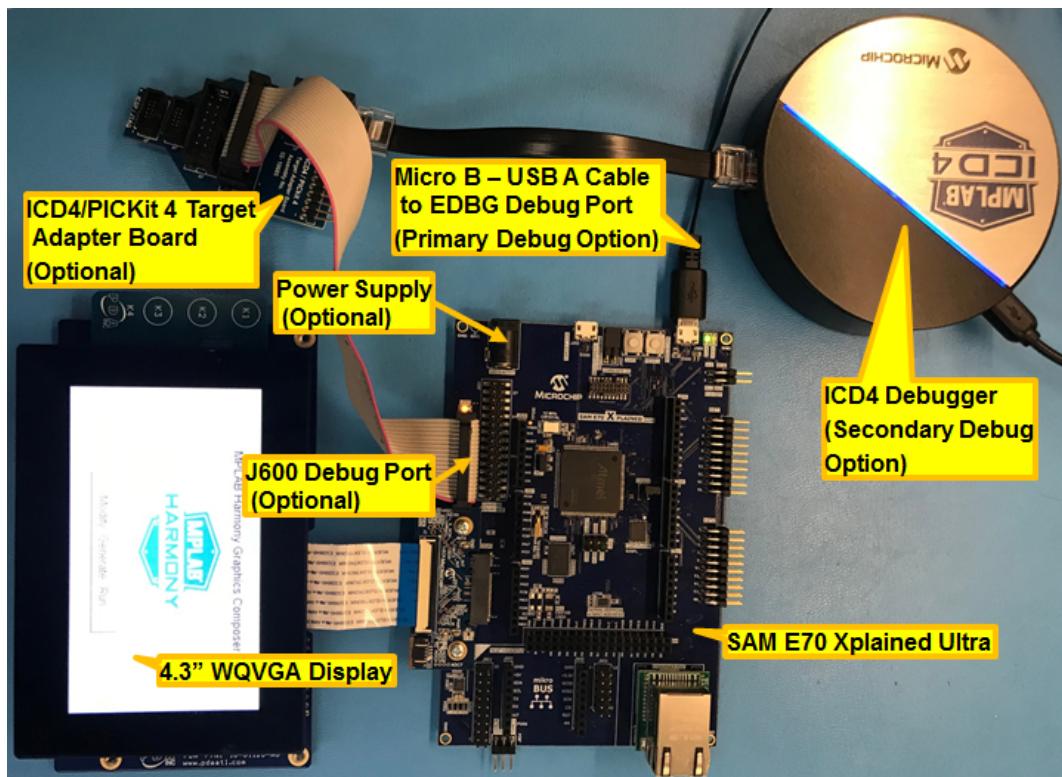
This section describes how to configure the supported hardware.

Description

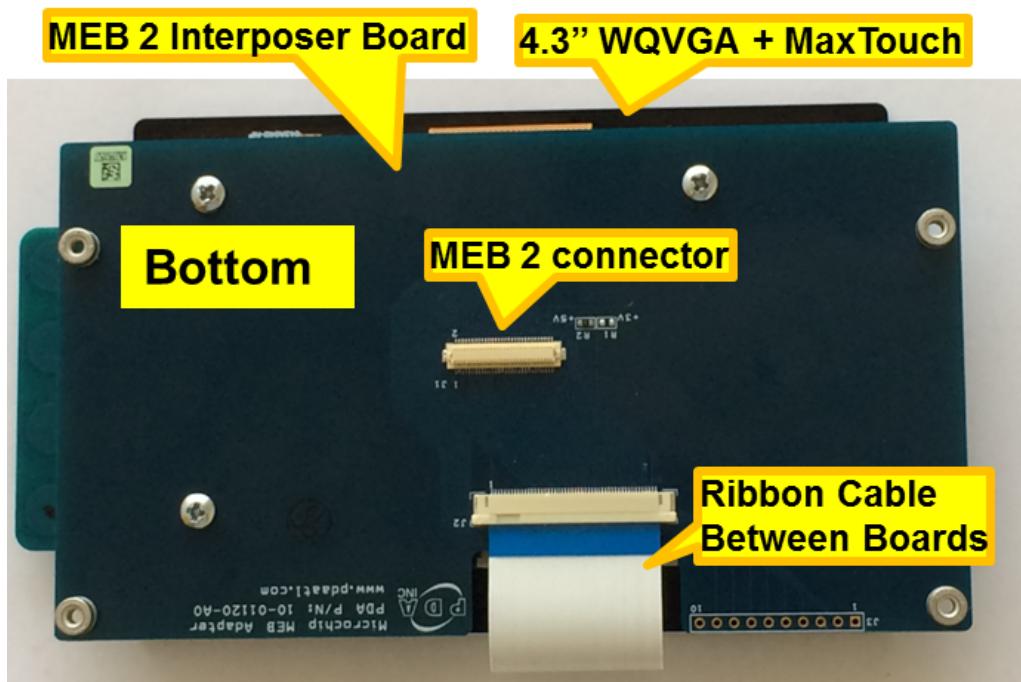
SAM E70 Xplained Ultra board with 4.3" WQVGA Display

There are two programming options. The primary option is the Micro B Embedded Debugger (EDBG) port. Power to the board can also be supplied via this connection. The secondary option is via the ICD4 debugger with the ICD4/PICKit 4 Target Adapter Board (Power has to be provided from power supply in this alternative setup).

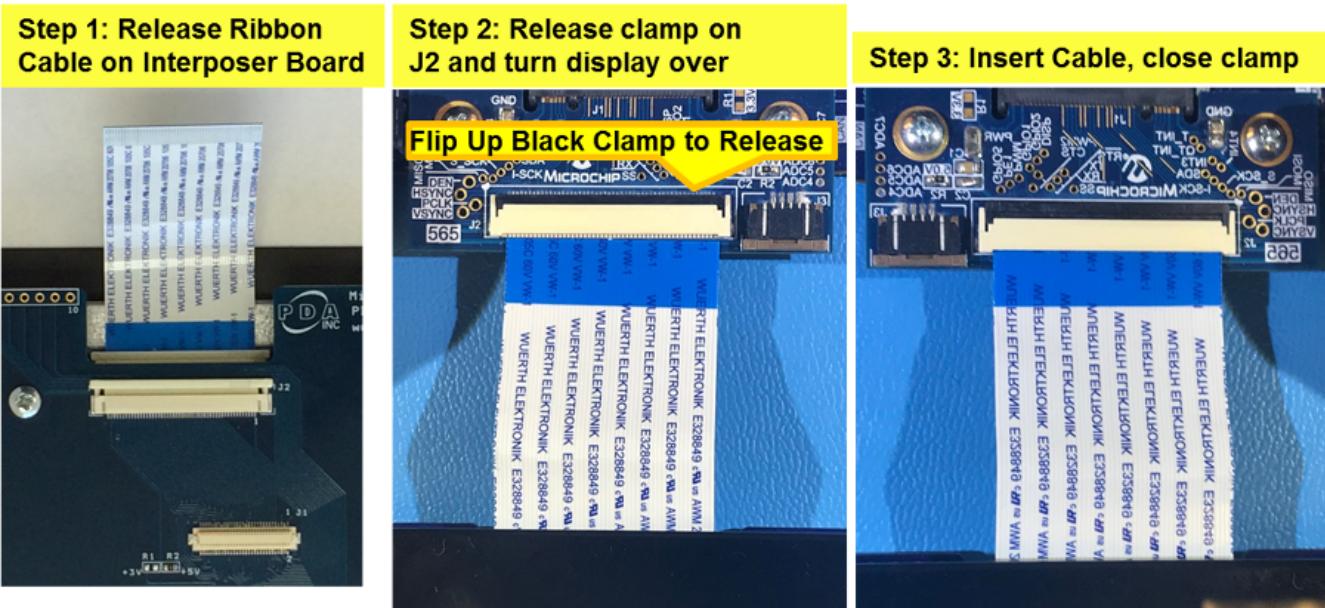
The image below shows both the EDBG and the ICD4 connection options:



Configuring the 4.3" WQVGA Display requires disconnecting the ribbon cable that connects the display to the interposer board.



First, release the ribbon cable from the interposer board. Next, release the black clamp on the E70's J2 connector and turn the display over. Finally, insert the ribbon cable into J2 and close the clamp.



The board and display are powered by a Micro B – USB A cable from PC to the “Debug USB” port on the E70 board. The ICD4 Debugger and ICD4/PICKit4 Adapter Board are connected as shown above.

Running the Demonstration

This section provides instructions about how to build and run the Aria Weather Forecast demonstration.

Description

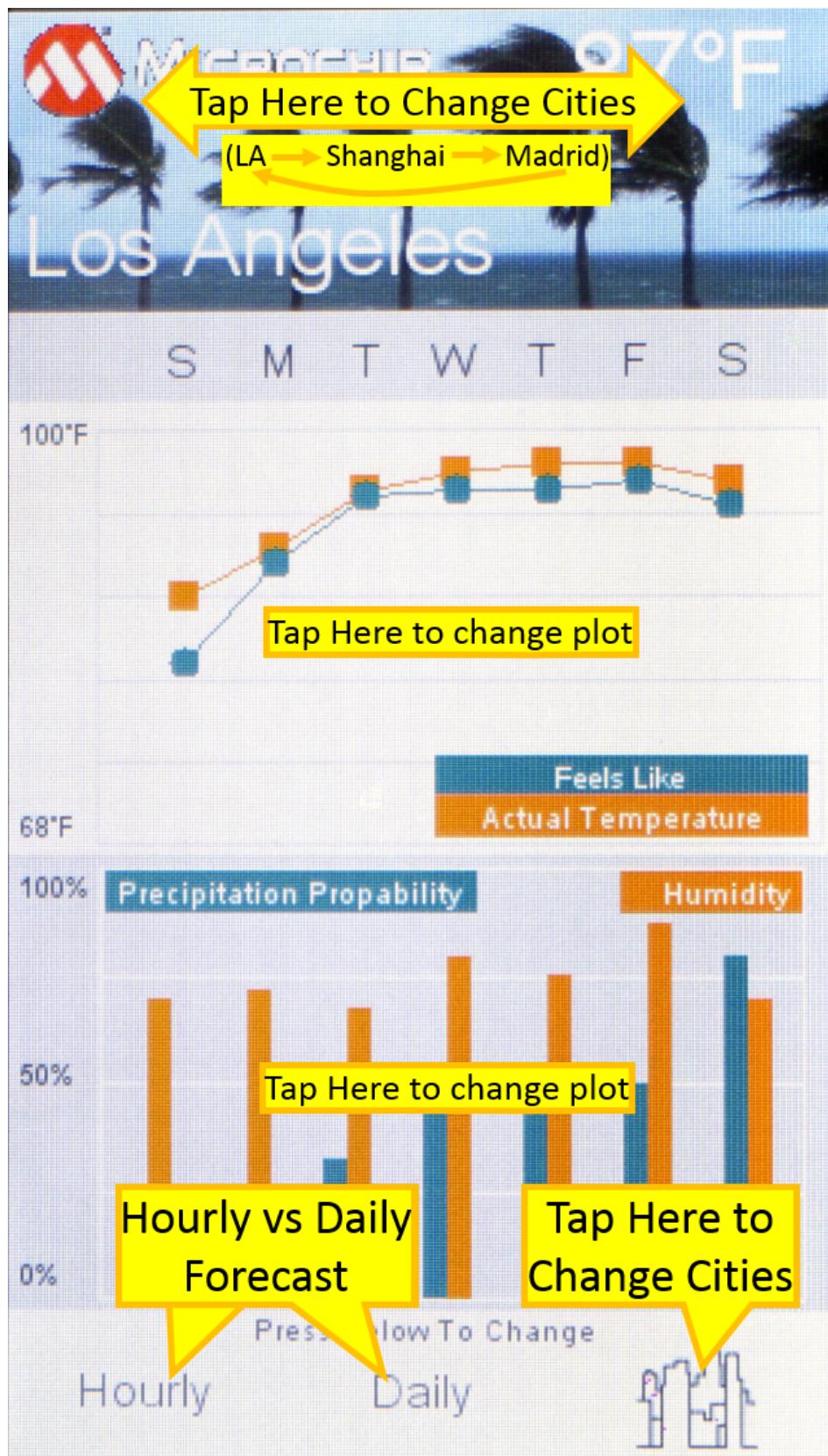
Splash Screen

When power-on is successful, the demonstration will display an animated splash screen:

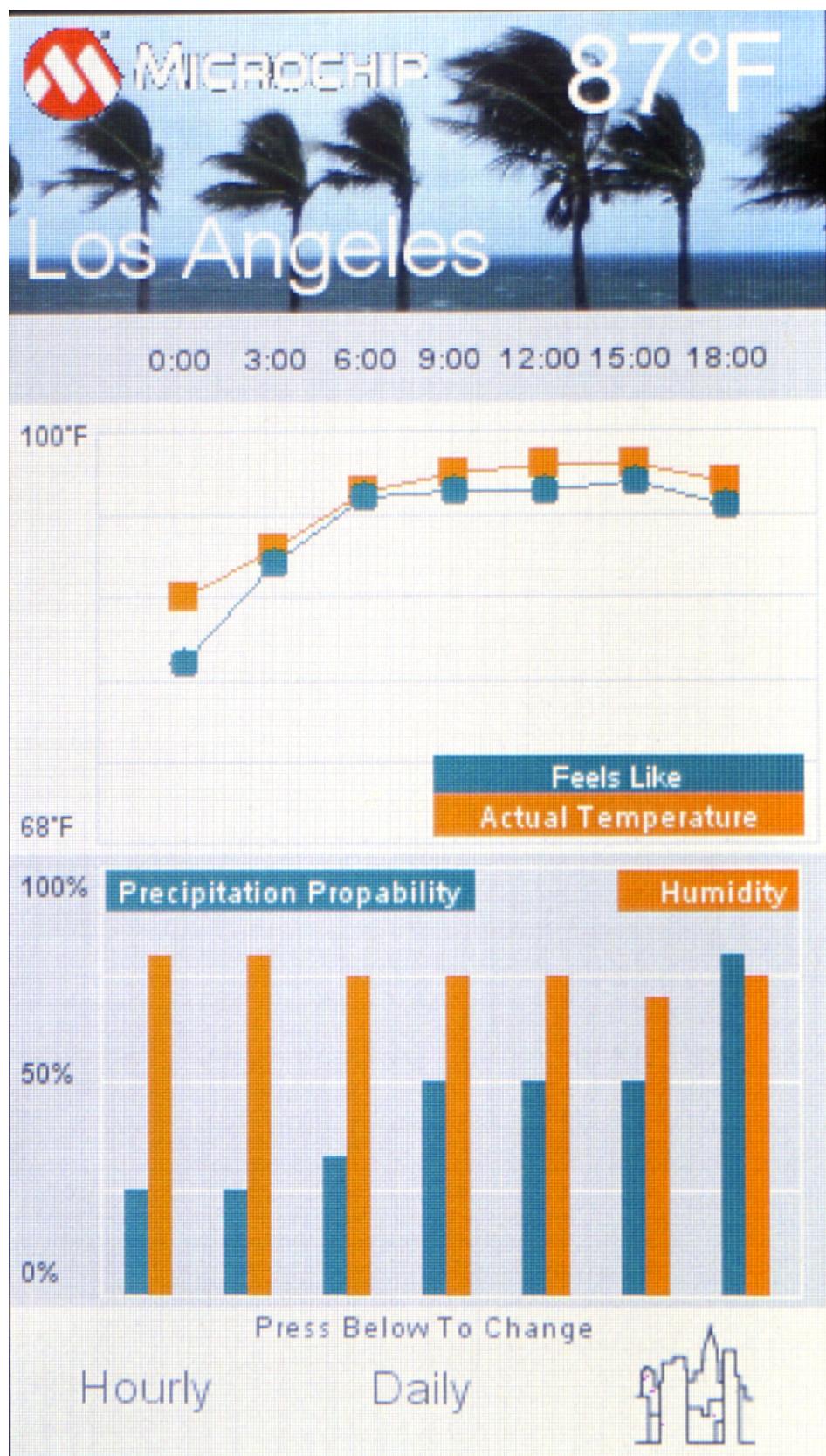


Main Screen

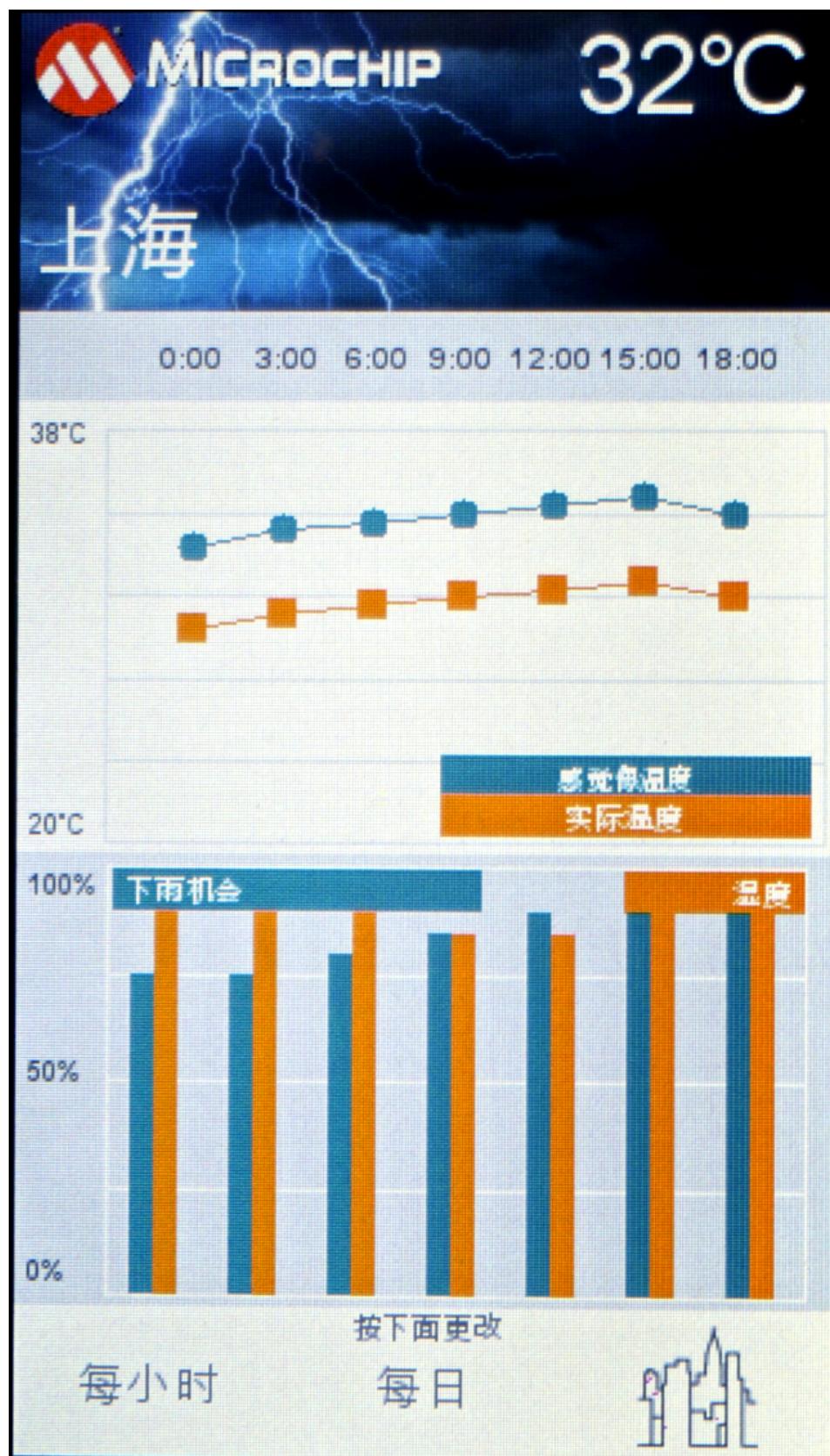
The next screen after boot-up shows Los Angeles (LA) daily weather:



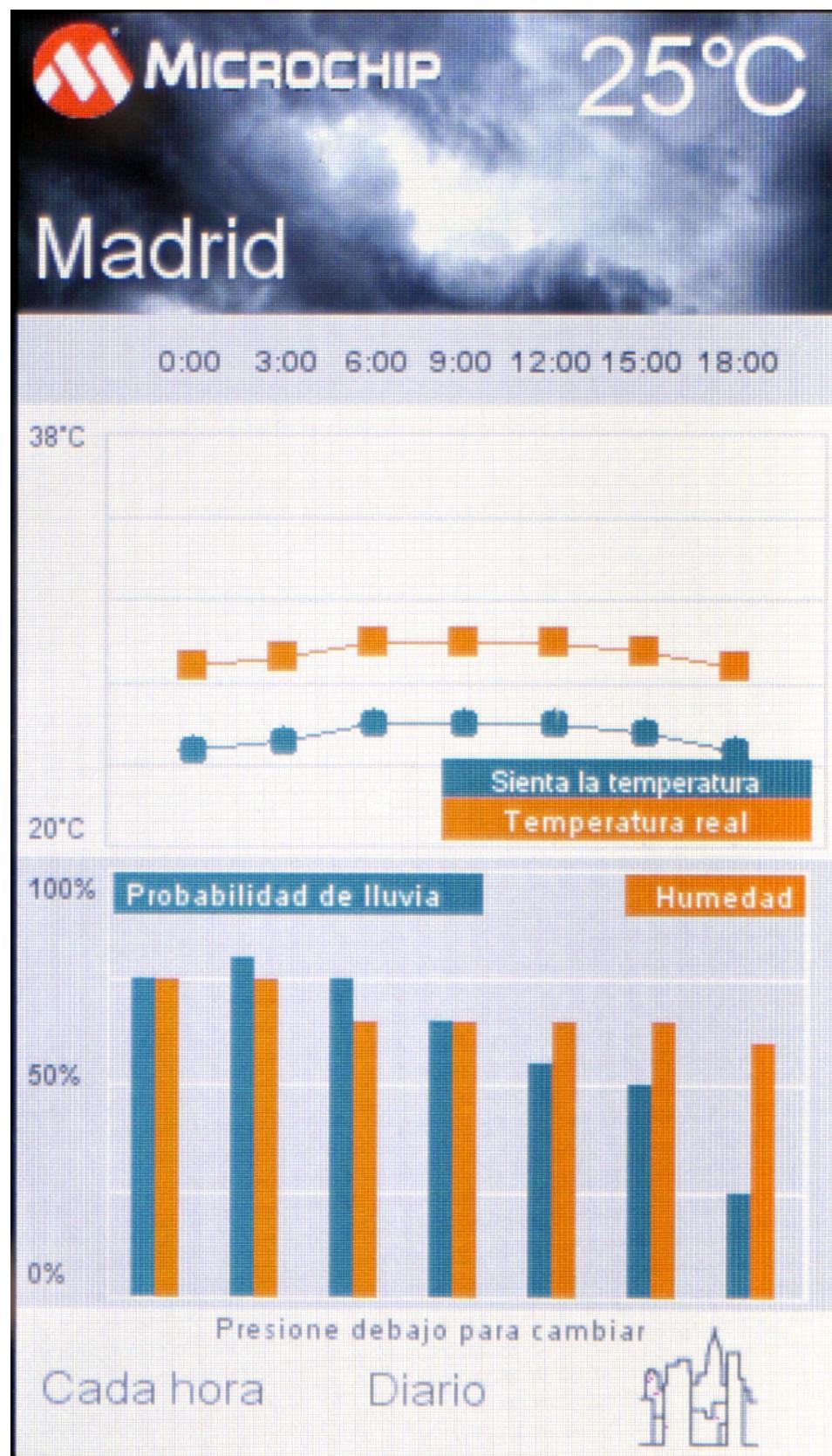
A tap at the top or at the lower right button (city graphic) changes from LA to China to Madrid and back to LA with each touch. Buttons on the bottom of the display changes the forecast between daily and hourly. The same change can be accomplished by tapping the center of the plots. Here is the LA hourly forecast display:



The Shanghai forecast display is in Chinese:



The Madrid forecast is in Spanish:



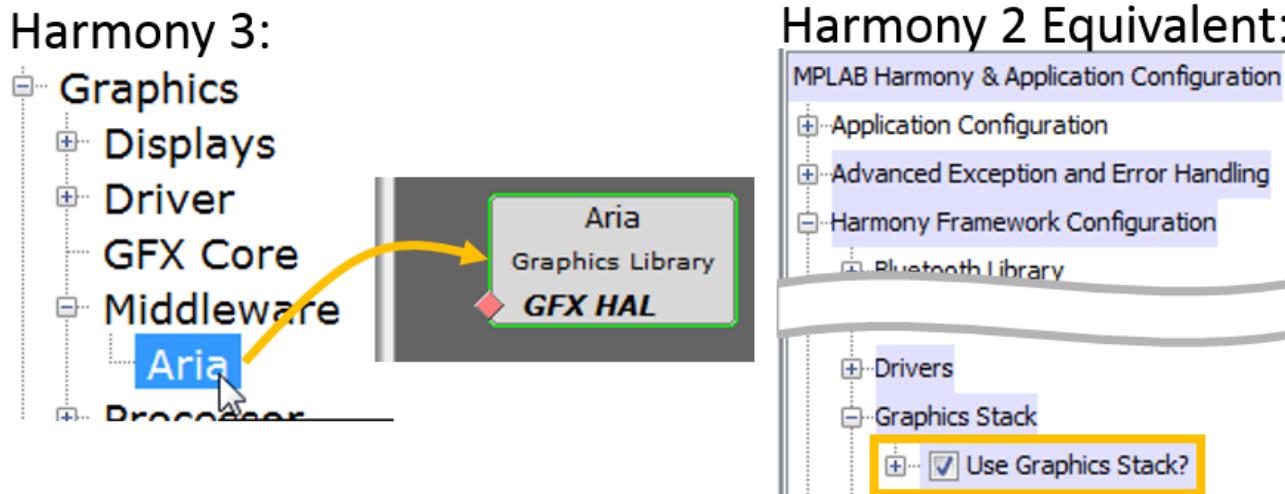
MHC GFX Components

Introduction

This section provides an overview of Graphics components used in MPLAB Harmony Configurator (MHC) to assemble graphics-enabled applications.

Description

In Harmony 2 applications are built by selecting options in the Options tab of MPLAB Harmony Configurator under the MPLAB Harmony & Application Configuration menu. For example, to add the Graphics Stack to an application the option “Use Graphics Stack?” is enabled under Harmony Framework Configuration > Graphics Stack:



In Harmony 3 project components are selected from the **Available Components** panel within MHC. The Harmony 3 equivalent to the “User Graphics Stack?” checkbox in Harmony 2 is to double click (or drag) the Aria Graphics Library component into the Project Graph.

To better understand how to configure graphics-enabled applications in Harmony 3 it helps to focus on an existing application, such as aria_quickstart's .aria_qs_e70_xu_tm4301b configuration.

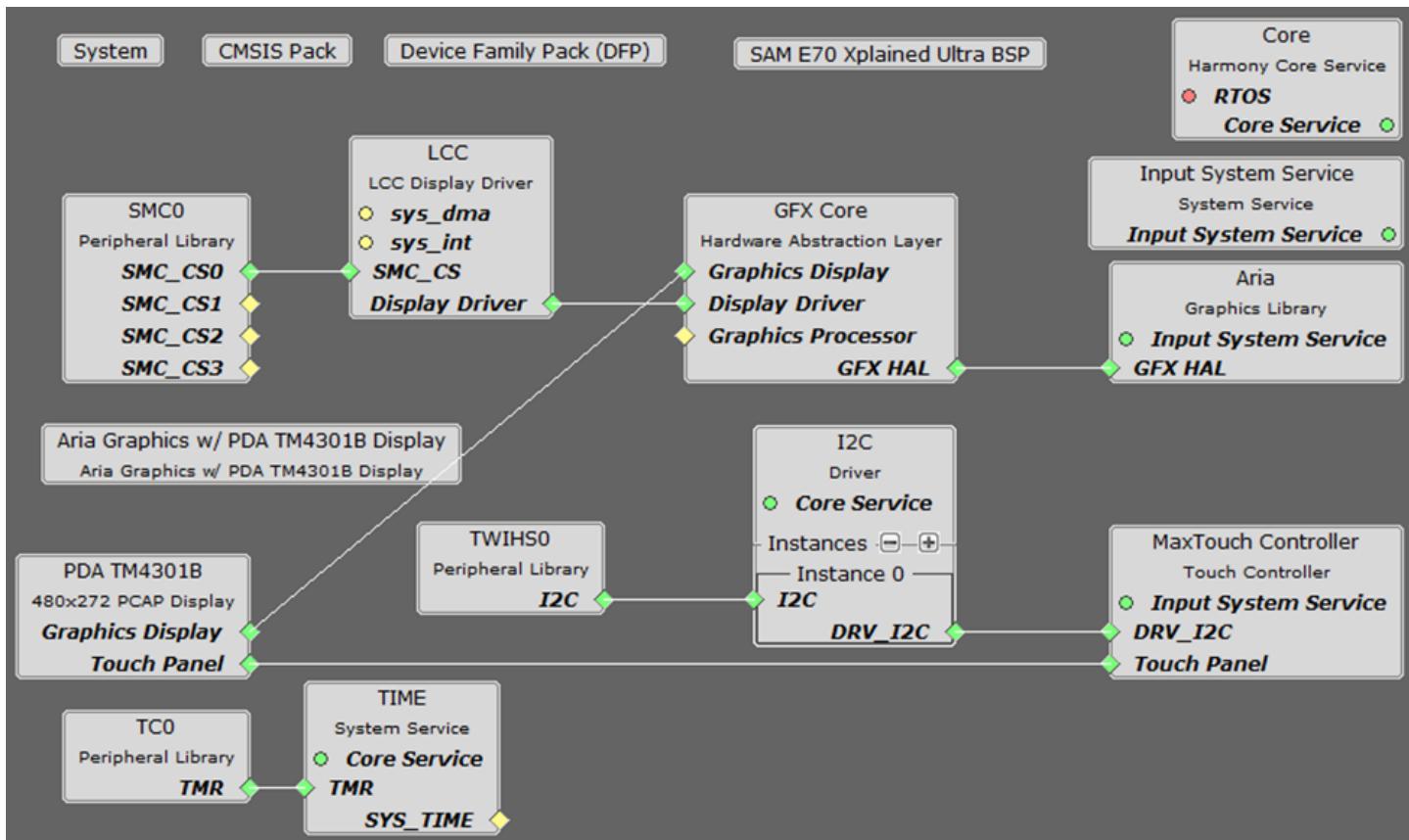
Exploring Aria Quickstart

The Project Graph of aria_quickstart is used to explore the components available in support of graphics.

Description

The Project Graph for aria_quickstart is typical of a graphics-enabled application. Details may vary for other graphics applications, primarily with the display, display driver, and touch capabilities. The basic graphics structure will be the same for all graphics-enabled applications.

The Project Graph for aria_quickstart shows:

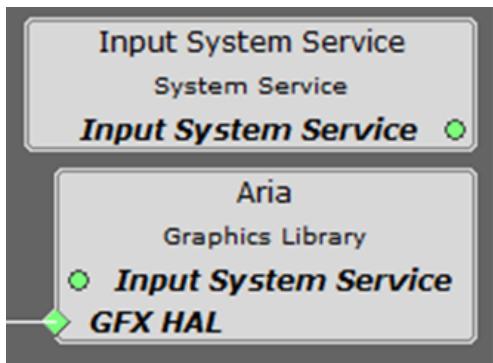


Components typically have connections to “Satisfiers” (aka services) on the left side and “Consumers” (aka clients) on the right side:

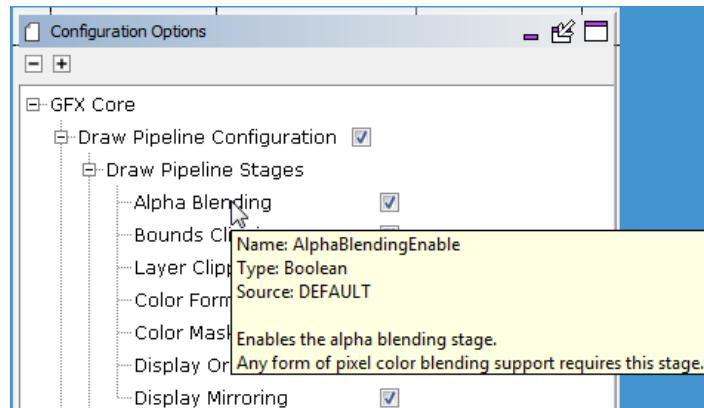
Typical Component:



Diamonds are for direct connections between components, as shown above. Circles are for implied connections, as is seen between the Aria Graphics Library and Input System Service components. Red shows that a connection is required and green shows that a connection exists. Yellow shows that a direct connection is optional:



Clicking on a component in the Project Graph brings up the configuration options in the MHC Configuration Options panel. A mouse-over above any of the listed configuration options provides a brief explanation of the option:



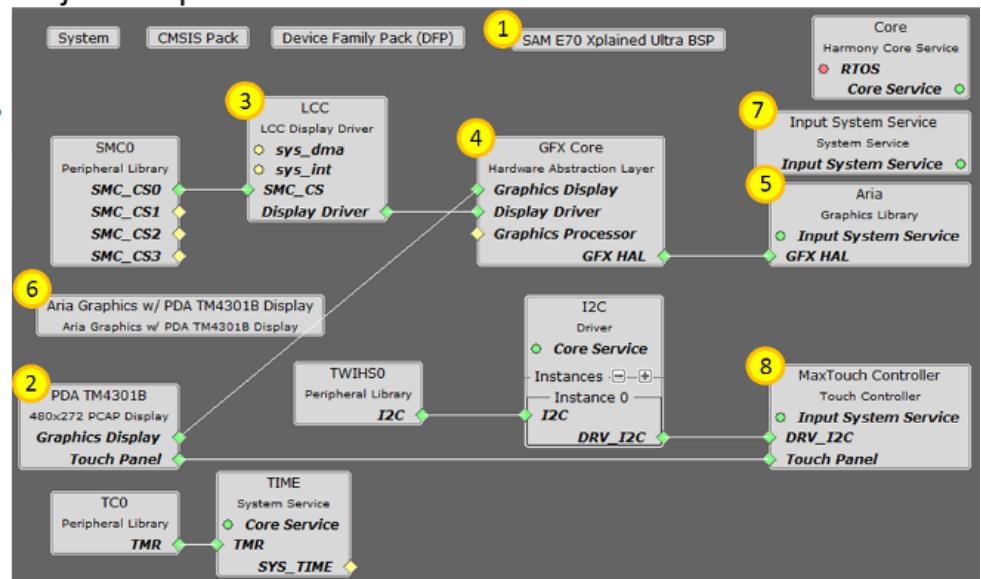
Components to Graph Map

Here is a map between the graphics components available and the components of aria_quickstart:

Components:

- Board Support Packages (BSPs)
 - SAM E70 Xplained BSP
 - SAM E70 Xplained Ultra BSP **1**
 - SAM E70 Xplained Ultra WM8904 BSP
 - default Board (BSP)
- Graphics
 - Displays
 - PDA TM4301B **2**
 - maXTouch Xplained Pro
 - Driver
 - Generic Controller
 - ILI9488
 - LCC **3**
 - GFX Core **4**
 - Middleware
 - Aria **5**
 - Processor
 - Generic GPU
 - Templates
 - Aria Graphics w/ PDA TM4301B Display **6**
 - Aria Graphics w/ Xplained Pro Display
- Input
 - Service
 - Input System Service **7**
 - Touch
 - Generic Touch Controller
 - MaxTouch Controller **8**

Project Graph:

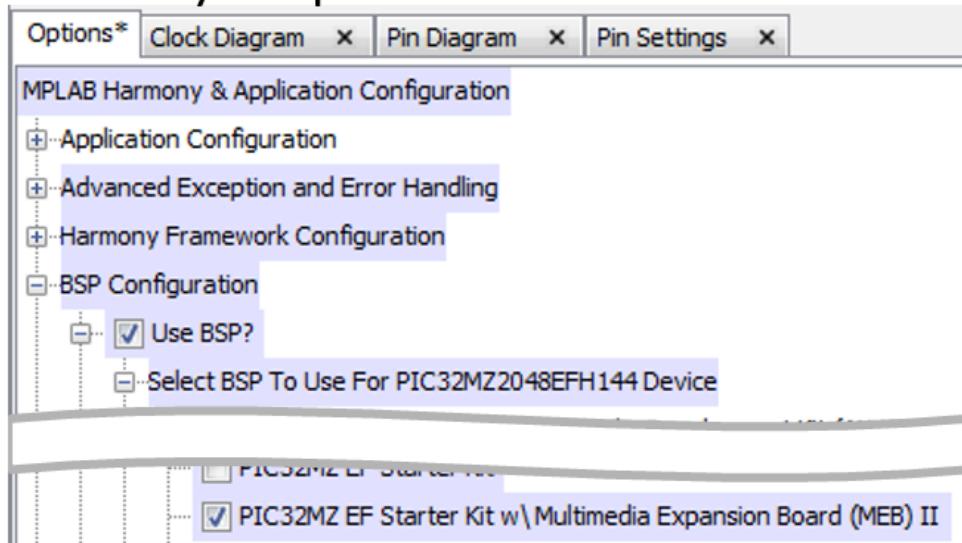


The circled numbers link components in the Components panel and the components in the Project Graph.

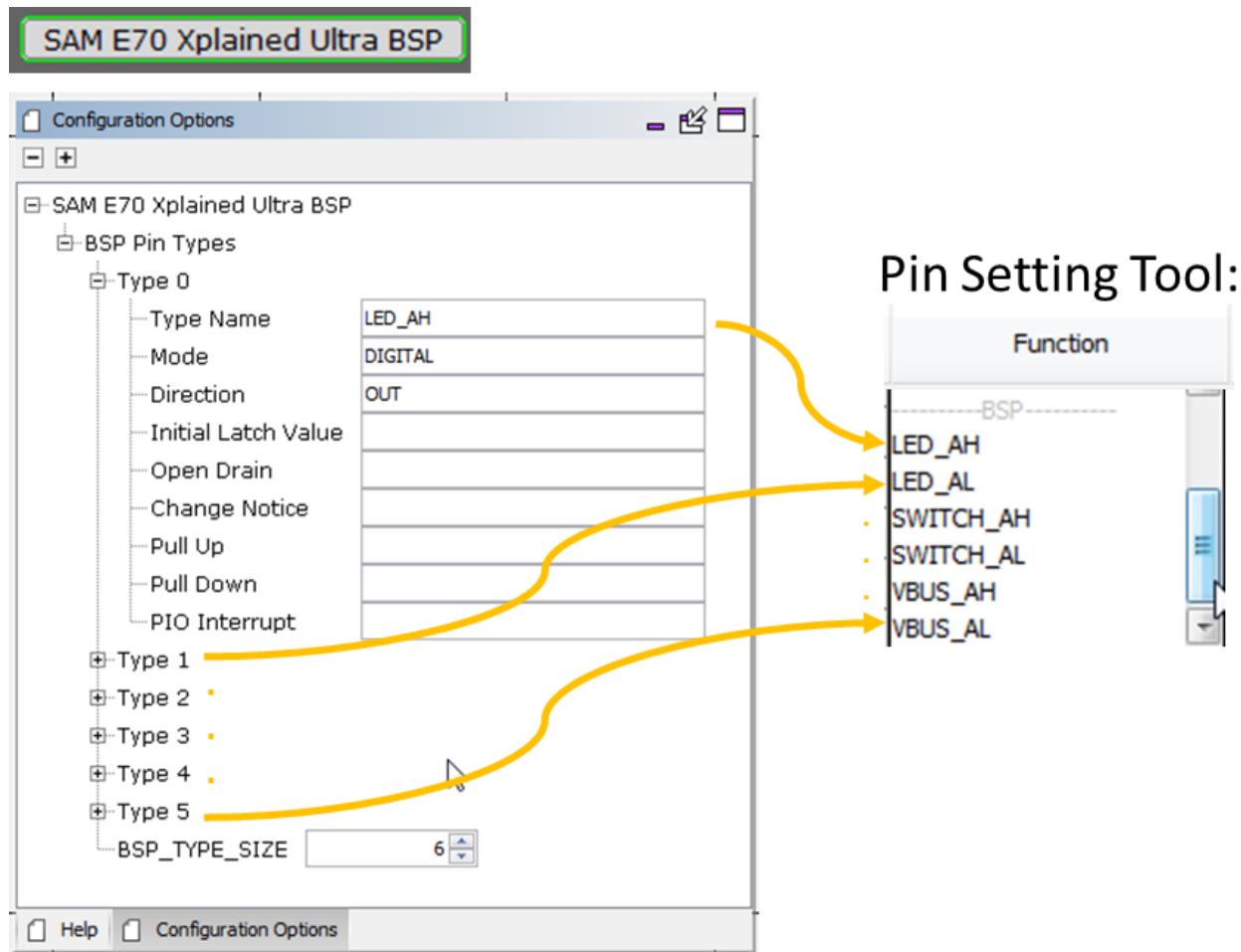
(1) Board Support Package

In Harmony 3 a Board Support Package (BSP) is selected from the Available Components panel. The equivalent selection in Harmony 2 is accomplished by a check box in the MPLAB Harmony & Application Configuration Menu:

Harmony 2 Equivalent in MHC Menu:

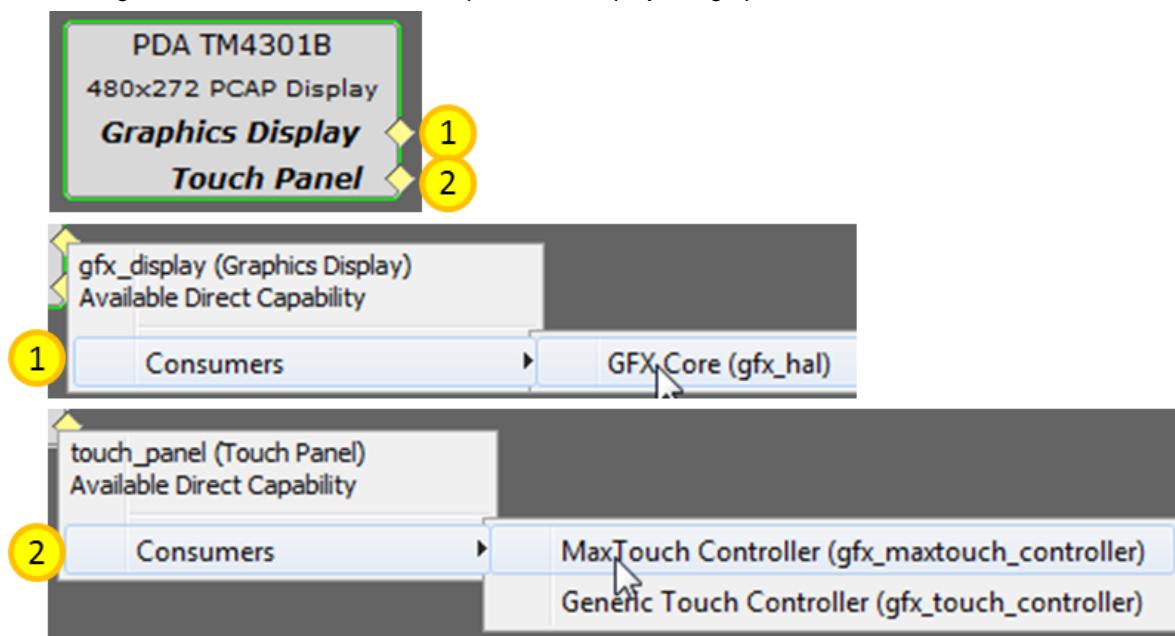


Clicking on the SAM E70 Xplained Ultra BSP brings up the Configuration Options for this component. The BSP Pin Types defined are available for use in MHC's Pin Setting Tool:

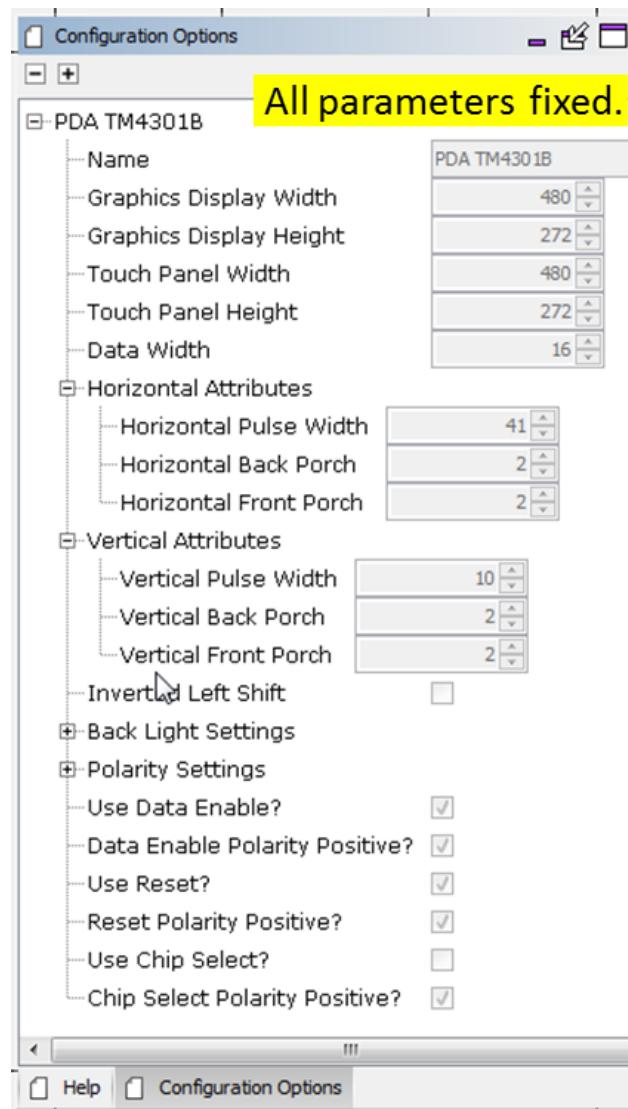


(2) Display Component

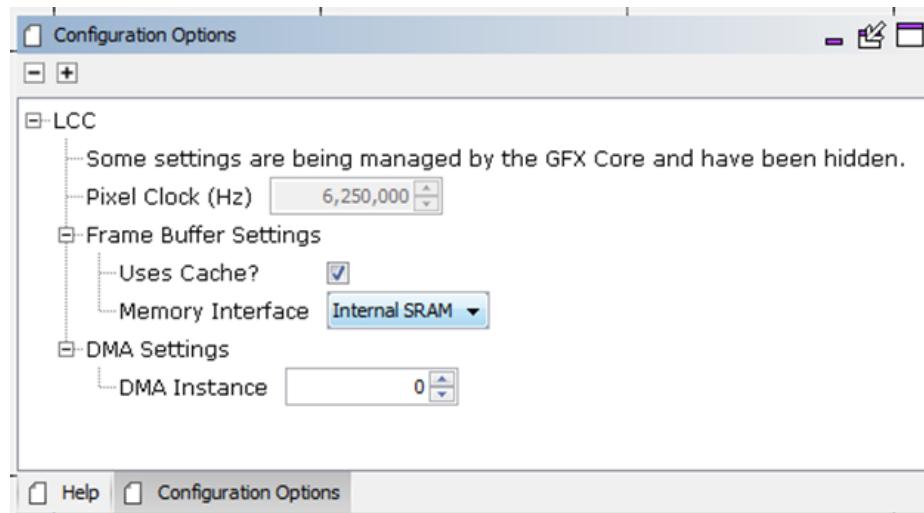
The PDA TM4301B component has two “Consumer” connections. A right click on a connection will bring up a list of the available consumers. Selecting one will add the consumer component to the project’s graph:



A click on the display component will bring up its configuration options in the Configuration Options panel. In this case all options are fixed (grayed out):

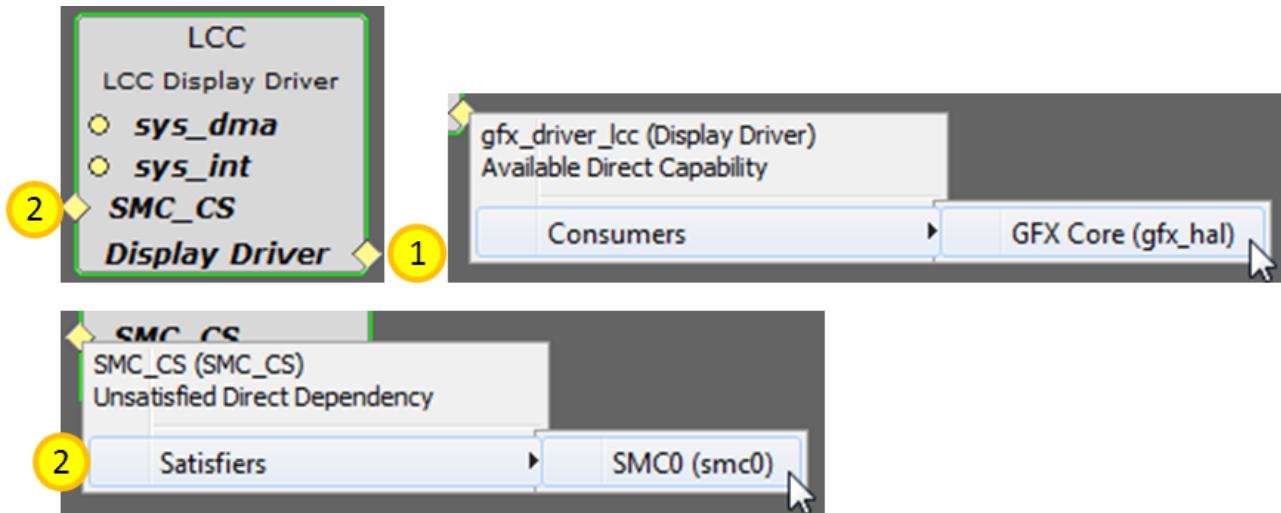


When connected to the GFX Core component, most of these options are handed off to the GFX Core:

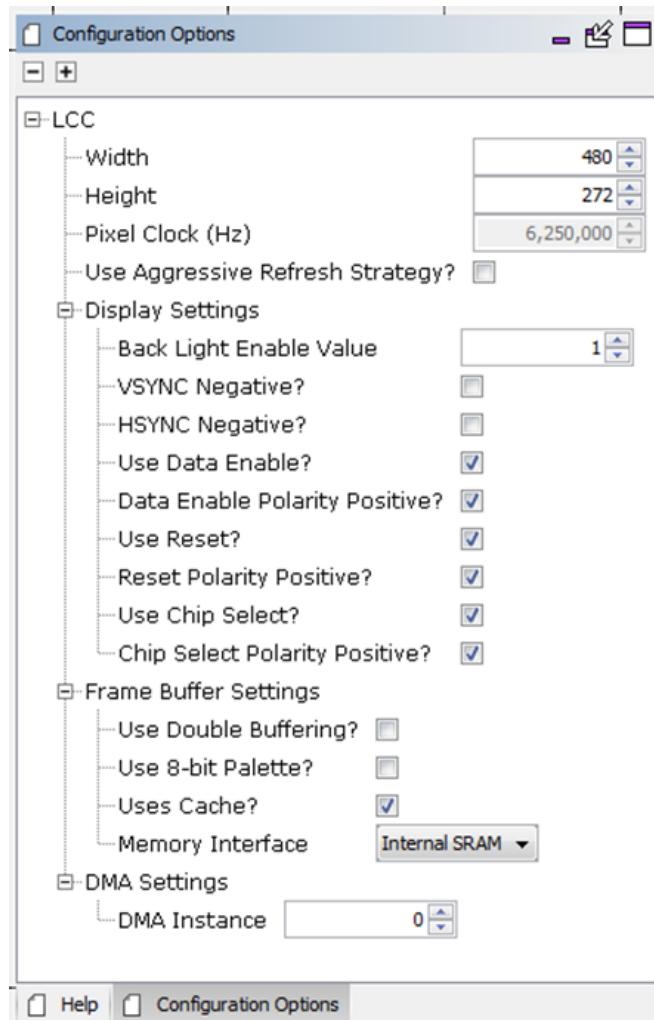


(3) Display Driver Component

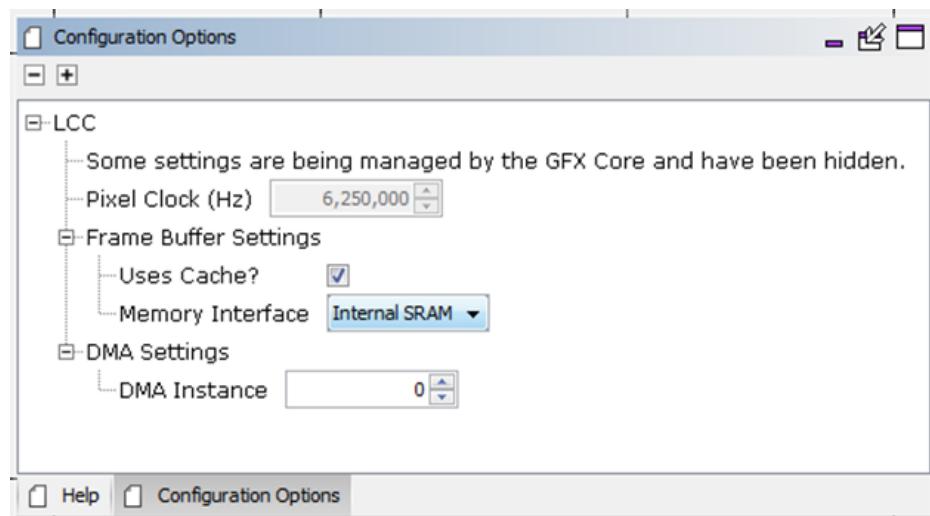
Since the SAM E70 lacks a on-chip graphics display driver, the Low-Cost Controllerless (LCC) driver is used to drive the display. The LCC component needs an SMC driver as a client and is connected to the GFX Core as a server:



When not connected to the GFX Core the LCC Display Driver component has the following configuration options:

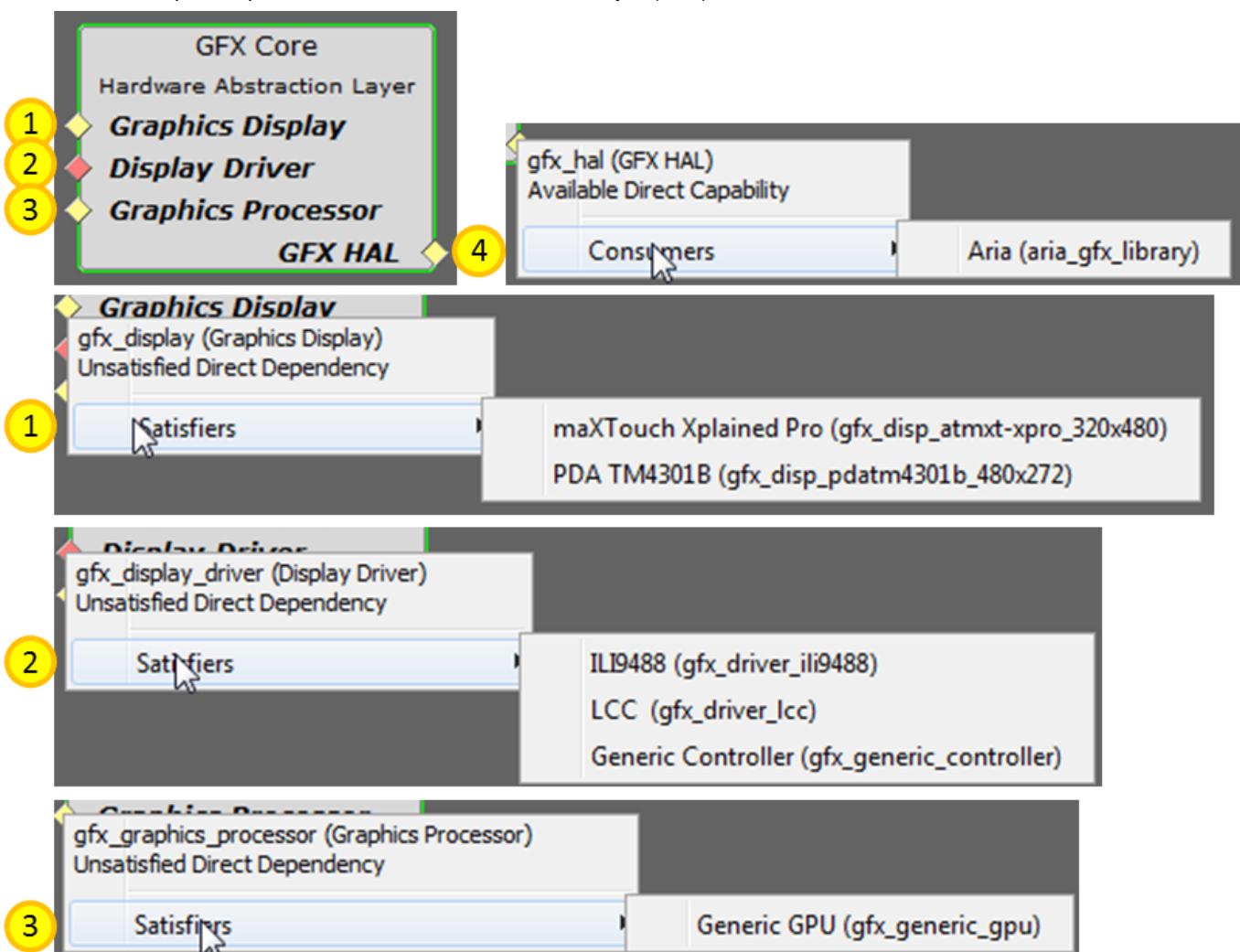


When connected to the GFX Core component, most of these options are handed off to the GFX Core:



(4) GFX Core Component

The GFX Core component provides the Hardware Abstraction Layer (HAL):



Only the Display Driver connection is required (it is shown in red above).

The GFX Core component has many options:

The screenshot shows the 'Configuration Options' window with the 'GFX Core' tab selected. On the left, a tree view lists various configuration categories. A yellow arrow points from the 'Draw Pipeline Configuration' node in the tree to its corresponding checked checkbox in the main panel. Below the tree, there are two tabs: 'Help' and 'Configuration Options'. The 'Configuration Options' tab is active.

GFX Core

- Draw Pipeline Configuration
- Display Settings
- Display Driver Information
- Graphics Processor Information
- Driver Configuration Hints

Display Settings

- Name
- Width
- Height
- Display Orientation
- Data Width
- Inverted Left Shift
- + Back Light Settings
- + Polarity Settings
- Use Data Enable?
- Data Enable Polarity Negative?
- Use Reset?
- Reset Polarity Positive?
- Use Chip Select?
- Chip Select Polarity Positive?

Custom Display

480
272
0
16

Back Light Settings

- Back Light Disable Value
- Back Light Enable Value

Polarity Settings

- VSYNC Negative?
- HSYNC Negative?

Display Driver Information

- Driver Info Function Name **Provided By Connected Display Driver**
- Driver Init Function Name

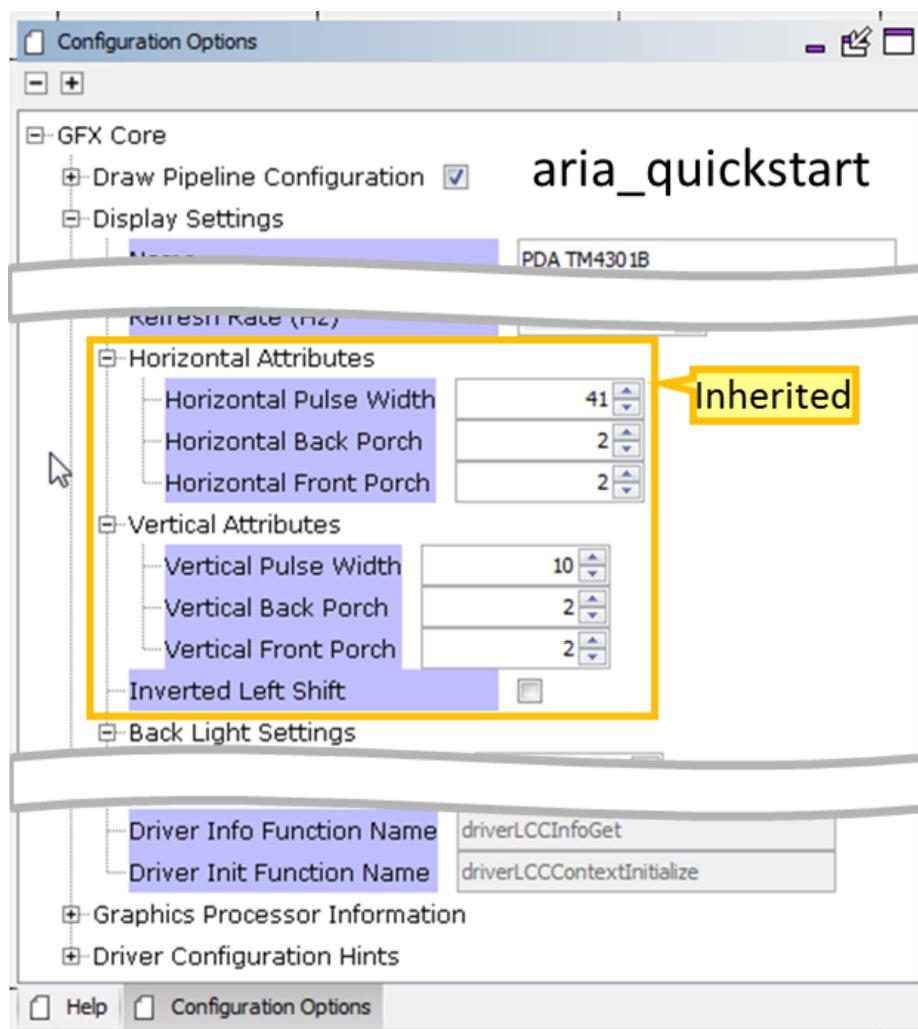
Graphics Processor Information

- Processor Info Function Name **Provided By Connected Graphics Processor**
- Processor Init Function Name

Driver Configuration Hints

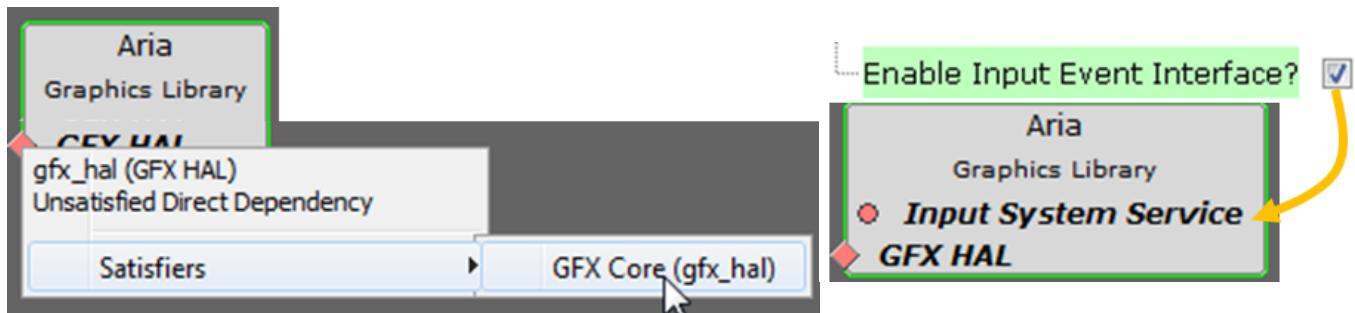
- Display Color Mode **GFX_COLOR_MODE_RGB_565**
- Global Palette Mode
- Double Buffer Mode
- Aggressive Refresh
- Hardware Layer Count **1**

When connected to a display the Display Settings part of the GFX Core's configuration options inherits timing information from the display:

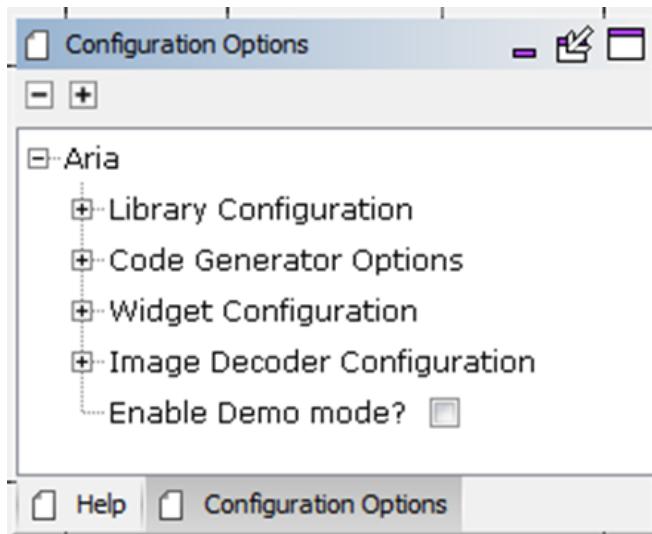


(5) Aria Graphics Library Component

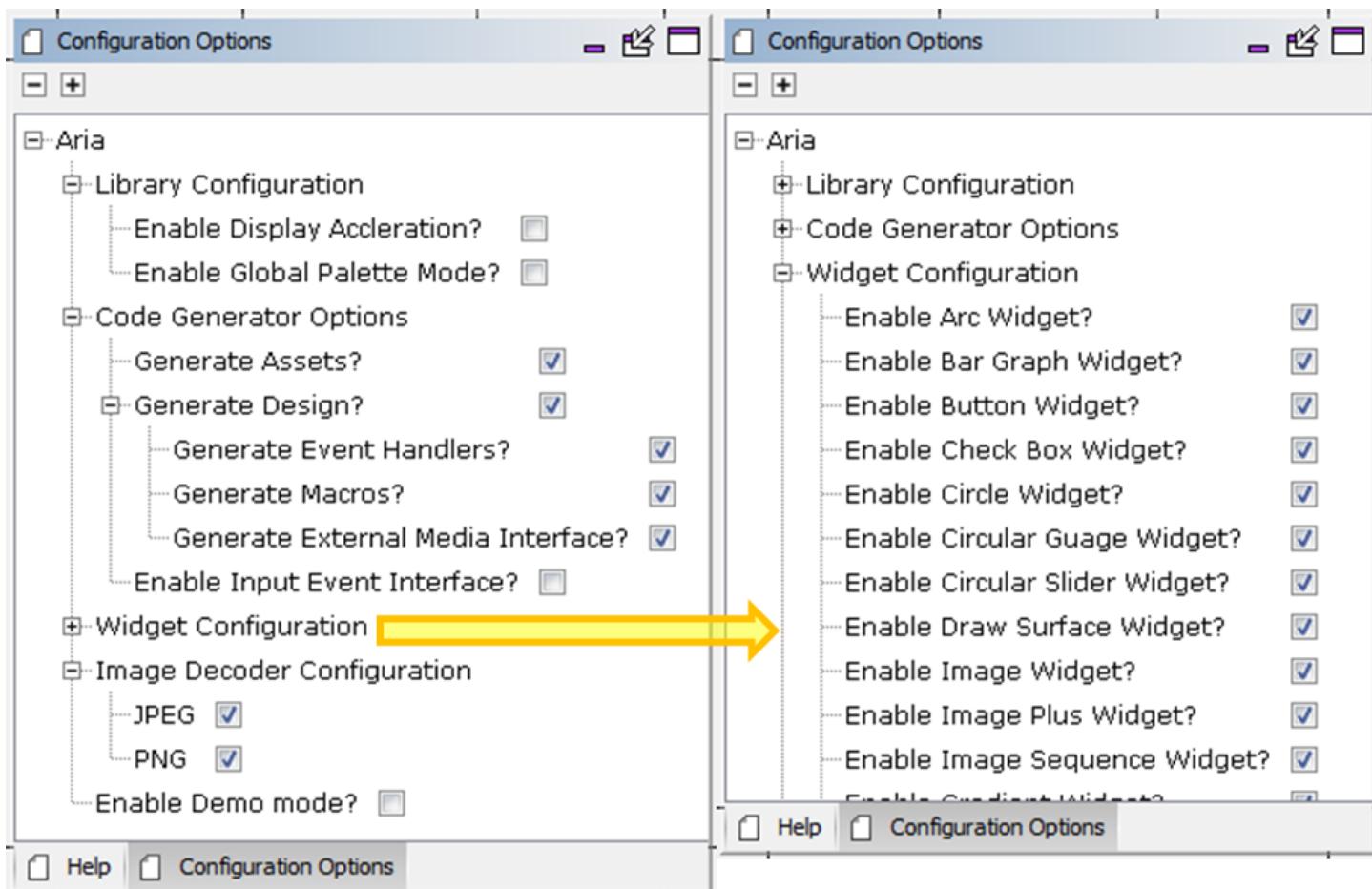
The Aria Graphics Library component requires a connection to the GFX Core component to provide Hardware Abstraction Layer (HAL) support. If the “Enable Input Event Interface?” configuration is enabled, then it will also need an implied connection to the Input System Service component:



The configuration options for this component are:

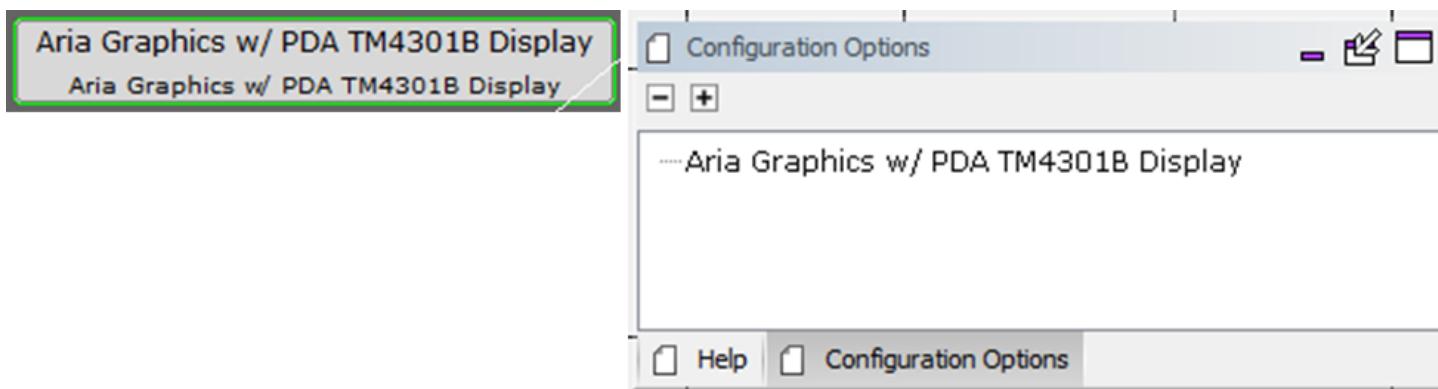


The Widget Configuration sub-menu allows disabling graphics widgets that are not used to reduce code size:

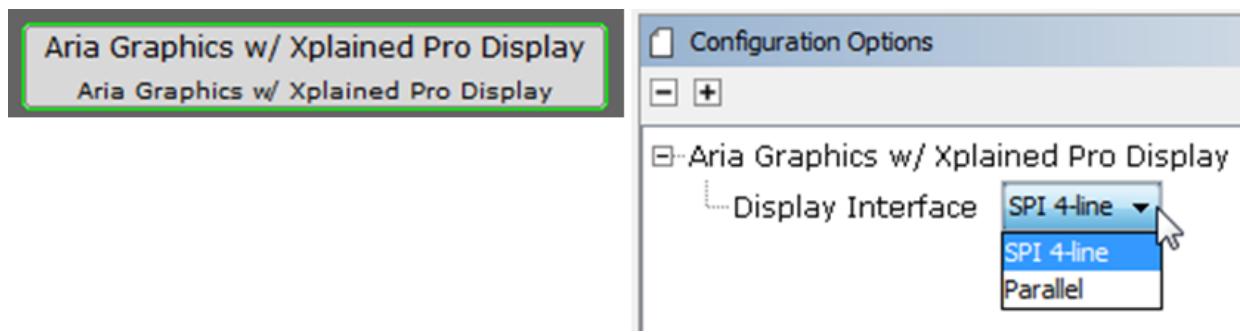


(6) Graphics Application Template

Templates package graphics components so that building a graphics-enabled application can be done in a single step by selecting the right template. The Aria Graphics w/ PDA TM4301b Display Template has no configuration options:

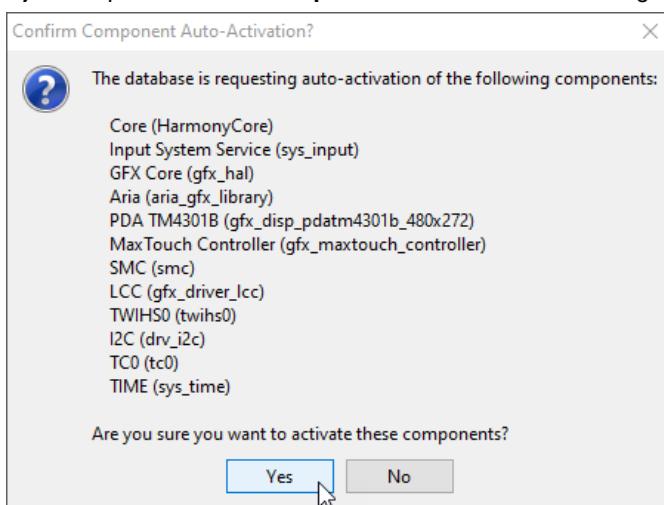


The Aria Graphics w/ Xplained Pro Display has one option, related to the display's interface:

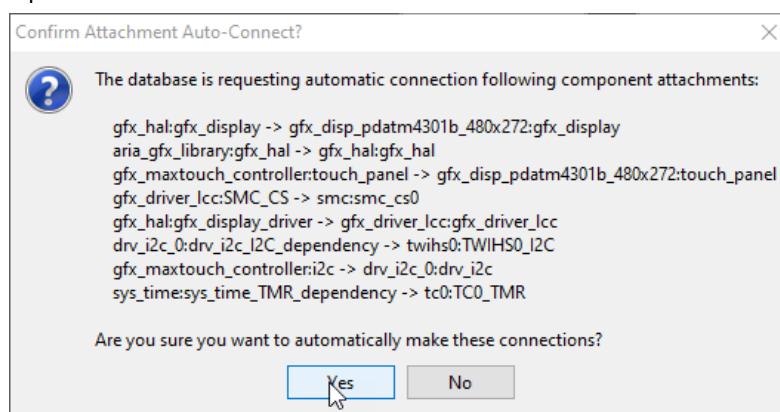


To properly use a Graphics Template, a supported Board Support Package (BSP) component must first be added. Having a BSP lets the Template know what the part and board the project is supposed to run on. Based on the BSP the template knows what drivers and pin settings need to be configured to run the desired graphics library and display. Otherwise, the template will only add and configure the Graphics Software components like the HAL and the Graphics Library, but it will not add the MCU or hardware-specific properties like the display drivers or the pin settings.

After adding the template to the Project Graph a **Confirm Component Auto-Activation** dialog appears:



Select the **Yes** button to proceed. Next a **Confirm Attachment Auto Connect** dialog appears to connect the newly installed components based on their dependencies:

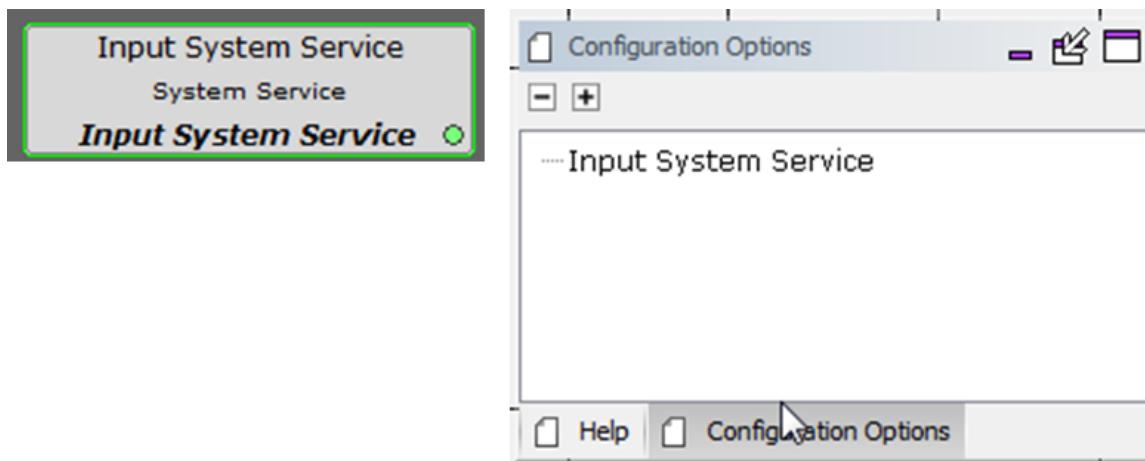


Select the **Yes** button to complete the task. Rearrange the project graph to clean up the component locations.

(7) Input System Service Component

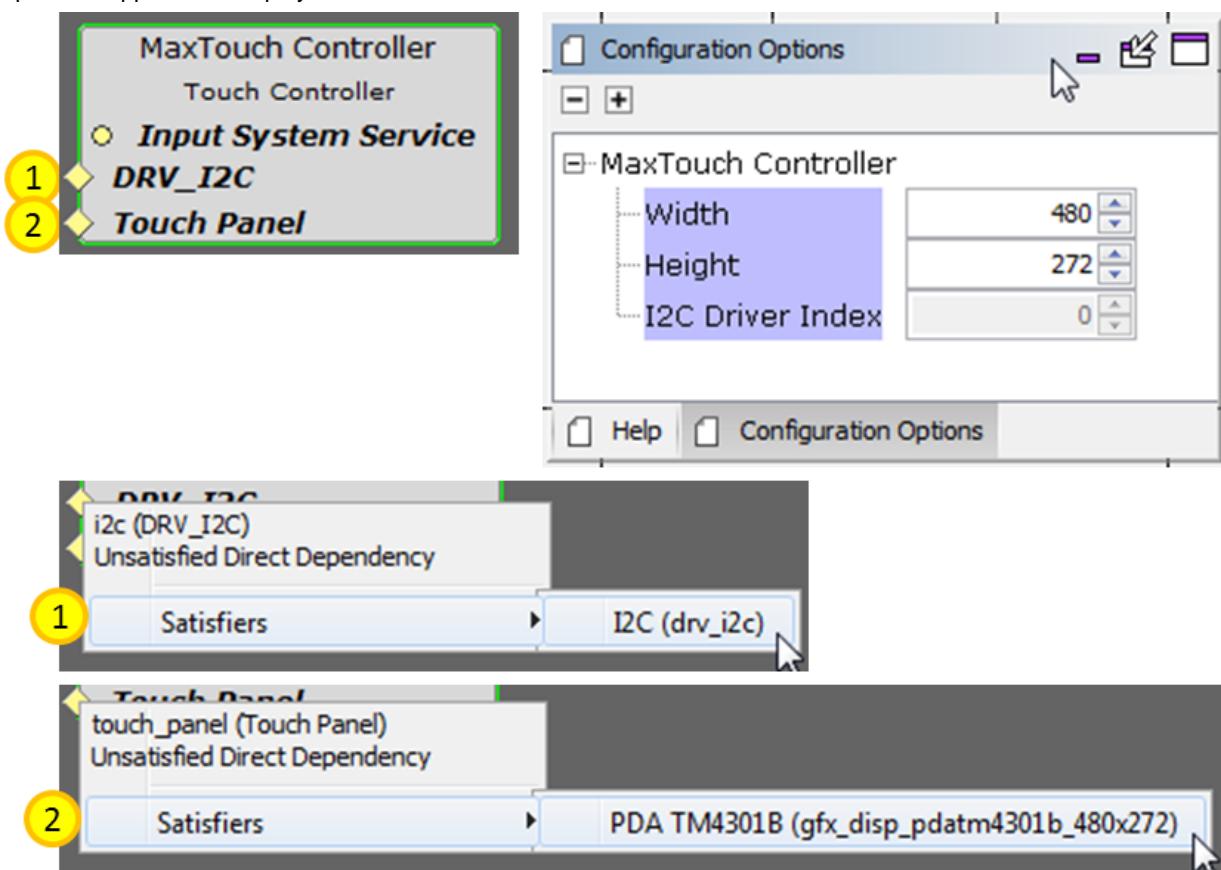
This component is needed to support touch event processing by the Aria Graphics Library component.

It has no configuration options:



(8) MaxTouch Controller Component

This component supports the display's MaxTouch touch controller:



The display width and height are adjustable and can be reduced from the display's actual size.

MPLAB Harmony Graphics Composer User's Guide

This section provides user information about using the MPLAB Harmony Graphics Composer (MHGC).

Introduction

This user's guide provides information on the MPLAB Harmony Graphics Composer (MHGC), also referred to as the graphics composer, which is included in your installation of MPLAB Harmony. MHGC is tightly coupled with the Aria User Interface Library to facilitate rapid prototyping and optimization of the application's graphical user interface (GUI).

Description

The MPLAB Harmony Graphics Composer (MHGC), also referred to as the graphics composer, is a graphics user interface design tool that is integrated as part of the MPLAB Harmony Configurator (MHC). MHGC is tightly coupled with the Aria User Interface Library to facilitate rapid prototyping and optimization of the application's graphical user interface (GUI). The tool provides a "What you see is what you get" (WSYWIG) environment for users to design the graphics user interface for their application. Refer to *MPLAB Harmony Graphics Library Help > Graphics Library Help > Aria User Interface Library* for more information.

The MPLAB Harmony Graphics Composer (MHGC) Tool Suite and the Aria User Interface Library provide the following benefits to developers:

- **Enhanced User Experience** – Libraries and tools are easy to learn and use.
- **Intuitive MHGC Window Tool** – Flexible window docking/undocking. Undo/Redo and Copy/Paste support. Tree-based design model. Display design canvas control including zooming.
- **Tight Integration Experience** – Graphics design & code generator tools are tightly integrated, providing rapid prototyping and optimization of look and feel
- **Powerful User Interface (UI) Library** – Provides graphics objects and touch support
- **Multi-Layer UI design** – Supported in the MHGC tool and Aria Library
- **Complete Code Generation** – Can generate code for library initialization, library management, touch integration, color schemes and event handling with a single click
- **Supports Performance and Resource Optimization** – Draw order, background caching, and advanced color mode support improve performance
- **Resource Optimization** – Measures Flash memory usage and can direct resources to external memory if needed. Global 8-bit color look-up table (LUT) supports reduced memory footprint. Heap Estimator tool, which helps to manage the SRAM memory footprint.
- **Text Localization** – Easily integrate international language characters into a design and seamlessly change between defined languages at run-time
- **Easy to Use Asset Management** – Tools provide intuitive management of all graphics assets (fonts, images, text strings)
- **Image Optimization** – Supports cropping, resizing, and color mode tuning of images
- **Expanded Color Mode Support** – The graphics stack can manage frame buffers using 8-bit to 32-bit color
- **Powerful Asset Converter** – Inputs several image formats, auto converts from input format to several popular internal asset formats, performs auto palette generation for image compression, supports run-length encoding. Supports automatic font character inclusion & rasterization.
- **Event Management** – Wizard-based event configuration. Tight coupling to enable touch user events and external logical events to change the graphics state machine and graphics properties.
- **Abstract Hardware Support** – Graphics controllers and accelerators can be added or removed without any change to the application.

Glossary of Terms

Throughout this user's guide the following terms are used:

Acronym or Term	Description
Action	A specific task to perform when an event occurs.
Asset	An image, font, or binary data blob that is used by a user interface.
Event	A notification that a specific occurrence has taken place.
Resolution	The size of the target device screen in pixels.
Screen	A discrete presentation of organized objects.
Tool	An interface used to create objects.

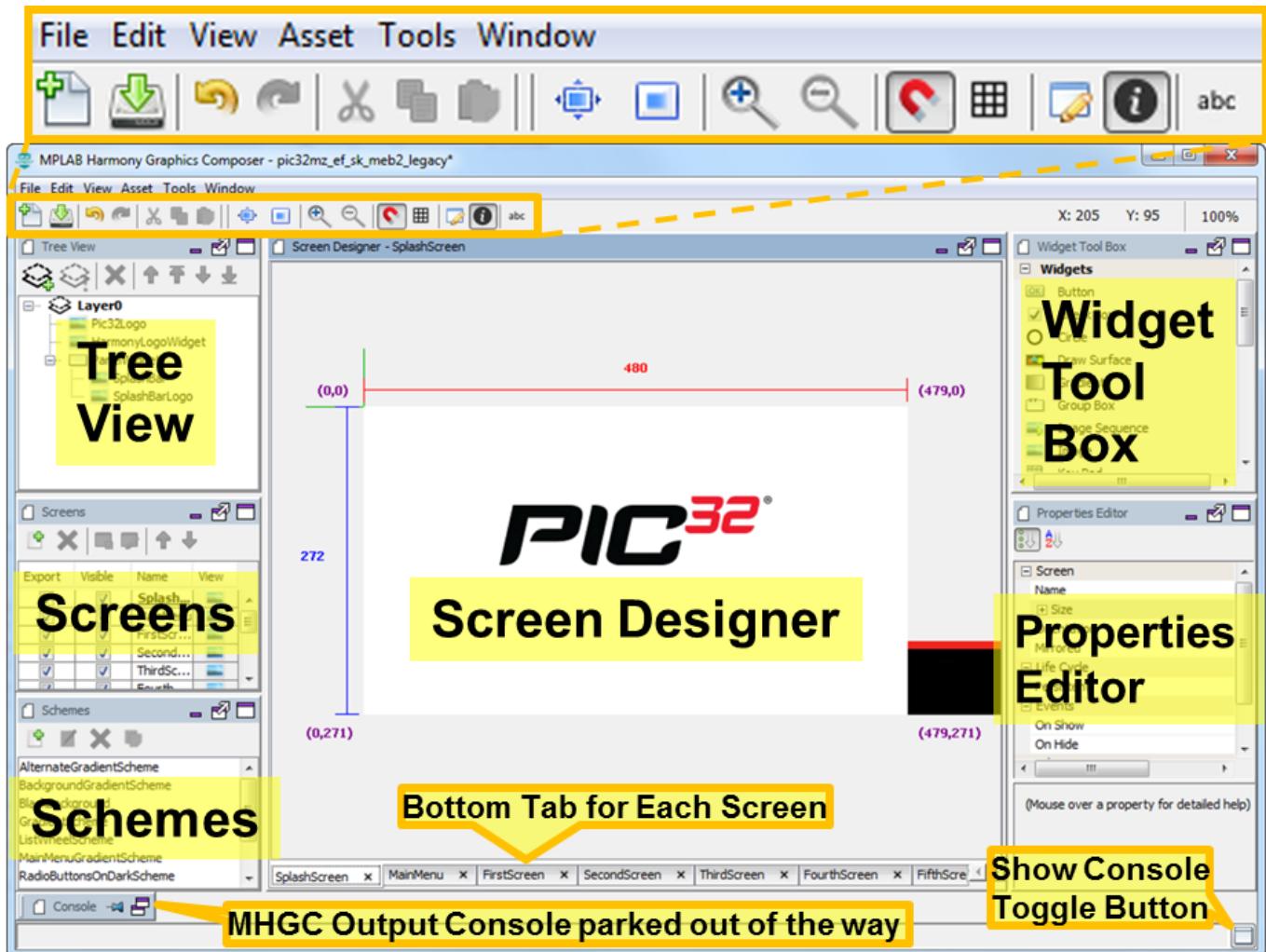
UI	Abbreviation for User Interface.
Widget	A graphical object that resides on the user interface screen.

Graphics Composer Window User Interface

This section describes the layout of the different windows and tool panels available through MHGC.

Description

MPLAB Harmony Graphics Composer (MHGC) is launched from the MPLAB Harmony Configurator (MHC) Tools menu. Launching the Graphics Composer creates a new screen. Shown below is the MHGC screen for the Aria Showcase demonstration. (If you don't see this screen layout, reset the screen by selecting *Window > Reset Dock Areas* from the window's menus.)



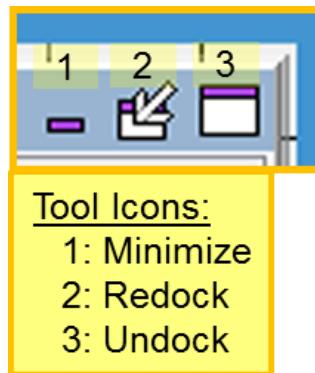
Panels

By default, there are five active panels and one minimize panel on this screen:

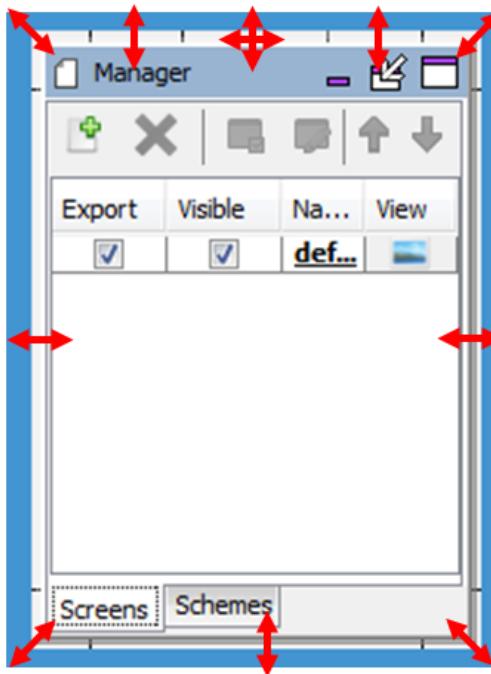
- **Screen Designer** – Shows the screen design for the selected screen. Tabs on the bottom of the Screen Designer panel show the available screens.
- **Tree View** – Shows the layer and widget hierarchy for the current screen.
- **Screens** – Manages screens in the application.
- **Schemes** – Manages coloring schemes in the application.
- **Widget Tool Box** – Available graphics widgets are shown on this panel. Widgets are added to the screen by selecting an icon and dragging or clicking. Widget properties are discussed in the Widget Properties section below.
- **Properties Editor** – All properties for the currently selected object are shown in this panel.
- The MHGC Output console is parked at the bottom of the Screen Designer window. This console panel can be used to debug

problems when the Graphics Composer boots up or during its operation.

Each of the panels has a window tool icon at the upper right corner. Minimizing a panel parks it on the screen just like the Output Console. Undocking the panel creates a new, free floating window. Redocking returns a previously undocked window to its original location on the Screen Designer window.

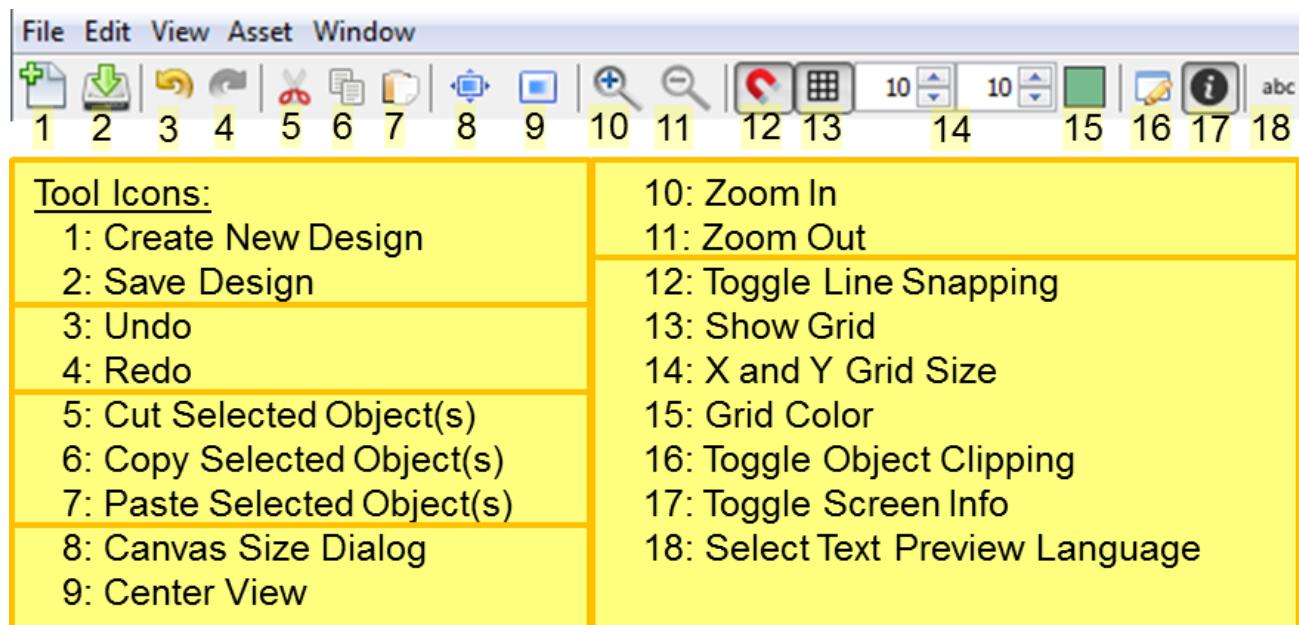


When a panel is undocked, its edges become active and support moving or manipulating the panel as an independent window.



Tool Bar

There are 18 tool bar icons on the Screen Designer Window, as described in the following figure.



Create New Design brings up a New Project Wizard dialog that allows you to select anew the screen size, color mode, memory size, and project type. This will erase the currently displayed design.

Save Design saves the current graphics design.



Note: The target configuration's `configuration.xml` will not be updated to reflect these changes in the graphics design until one of the following events happens:

1. The application is regenerated in MHC,
2. The target configurations are changed in the MPLAB X IDE,
3. MPLAB X IDE is exited.

In items 2 and 3 you will be prompted to save the new configuration.

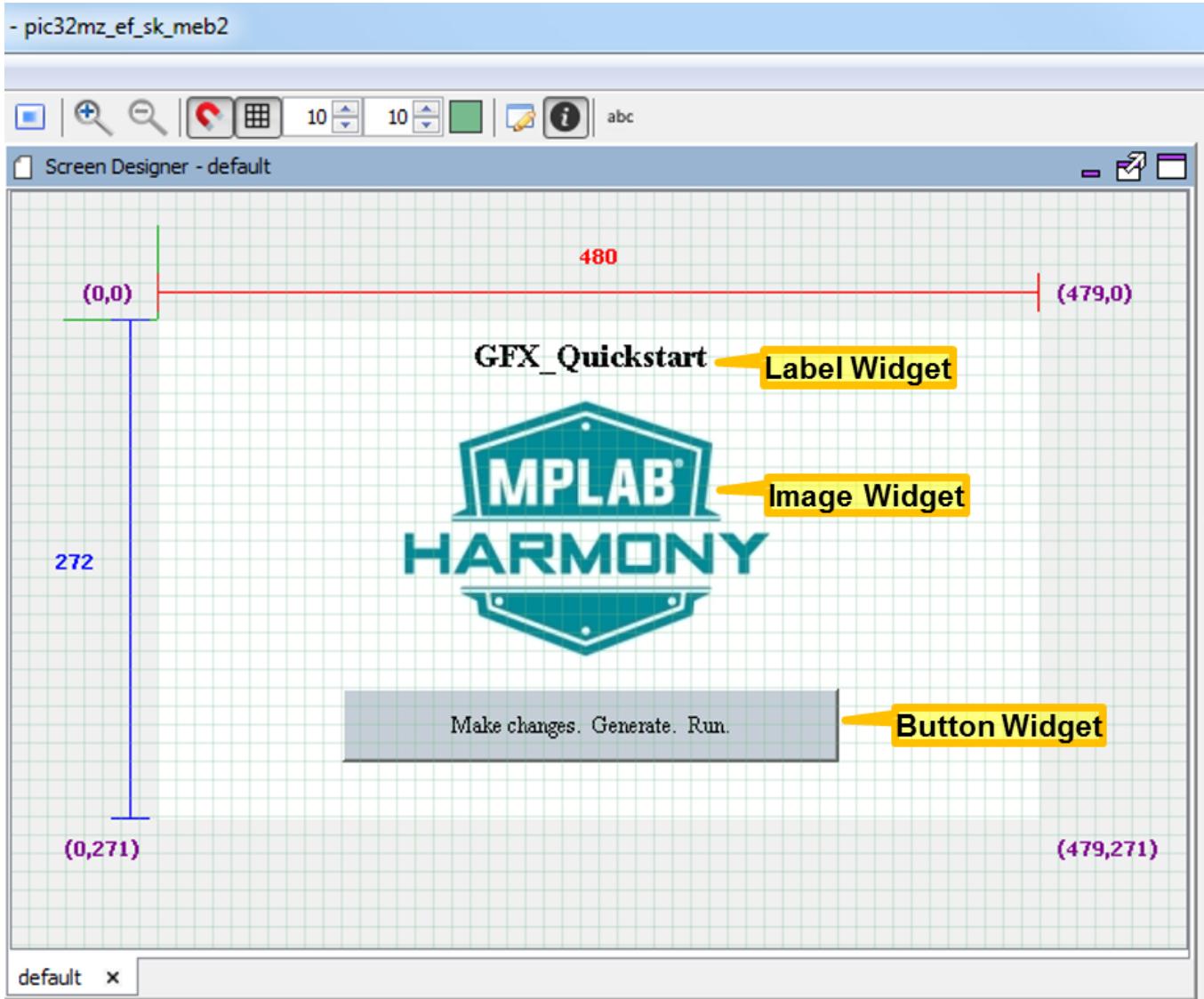
- **Undo** and **Redo** manipulate changes in the screen design into internal MHC memory.
- **Cut/Copy/Paste** support the manipulation of graphics objects (widgets).
- **Canvas Size Dialog** brings up a dialog window allowing changes in the pixel width and height of the Screen Designer panel. (Note: Dimensions smaller than the display's dimensions are ignored).
- **Center View** centers the panel's view of the screen.
- **Zoom In** and **Zoom Out** allow you to change the scale of the Screen Designer's display of the current window. Currently this only supports coarse zooming (powers of two zooms in and out).
- **Toggle Line Snapping** enables/disables line snapping when moving objects (widgets).
- **Show Grid** turns the Screen Designer pixel grid on/off.
- **X and Y Grid Size** adjust the pixel grid.
- **Grid Color** selects the pixel grid color.
- **Toggle Object Clipping** turns object clipping on/off.
- **Toggle Screen Info** turns the display of screen information (X and Y axes) on/off.
- **Select Text Preview Language** changes the language used on all text strings shown, when the application supports more than one language.

Screen Designer Window

Most of the work of the MPLAB Harmony Graphics Composer is done using the Screen Designer. This section covers the basics of how a graphical user interface is designed using the screen designer.

Description

The following figure shows the Screen Designer window for the [Aria Quickstart](#) demonstration, with the sam_e70_xplained_ultra_wqvga configuration selected.

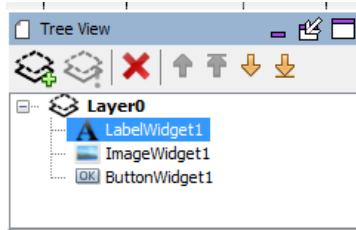


The pixel dimensions of the display (480x272) are determined by the MHC Display Manager's Display Settings panel. Other configurations in Aria Quickstart may have different size displays (such as: 220x176, 320x24, or 800x480).

This demonstration has three widgets: a label containing the title string at the top, an image of the MPLAB Harmony logo in the middle, and a button containing the text string "Make changes. Generate. Run." at the bottom. The label widget's text string was first created using the String Assets window before it was assigned to the label widget. The image assigned to the image widget was first imported using the Image Assets. The string embedded in the button widget was also created using the String Assets window before it was assigned to the button widget.

The Tree View panel organizes the display's widgets into groups using layers. Every display has at least one layer and complex designs can have many more. Within the tree view, the order of layers and the order of widgets within a layer determine the draw order. Draw order goes from top to bottom. Top-most layers and widgets are drawn first and bottom-most are drawn last.

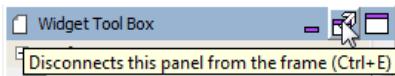
Controlling draw order is one of the ways to improve graphics performance by minimizing redrawing.



Since the location of every widget within a layer is relative to the layer, you can move a layer's worth of widgets by simply moving the layer. Layers also provide inheritance of certain properties from the layer to all the layer's widgets.

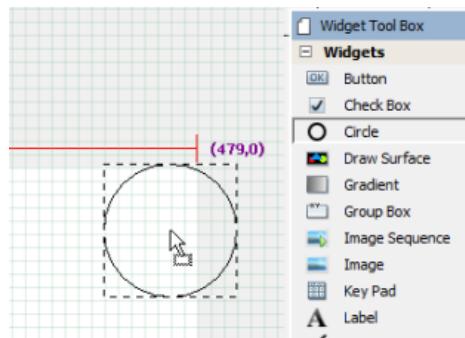
Exploring the Screen Designer Window

Another widget can be added to this screen by launching the Widget Tool Box panel into a separate window.

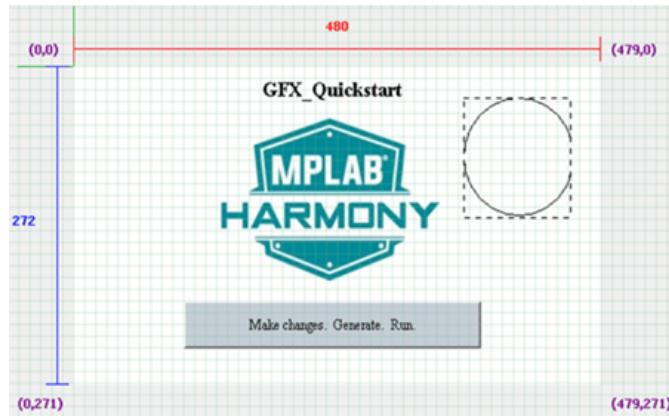


Next, drag a circle from the tool box onto the display. Find a place on the display for this new widget.

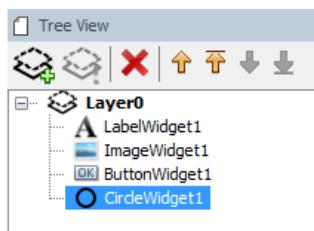
Besides dragging widgets onto the display, click on a widget in the Widget Tool Box, converting the cursor into that widget, and then click on the screen to drop the widget in place.



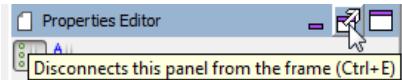
The display should now look appear like the following figure.



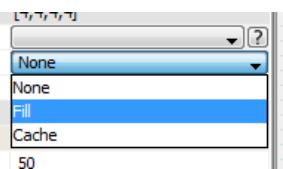
Note how the Tree View panel now shows the widget that was just added.



Launch the Properties Editor for the circle.

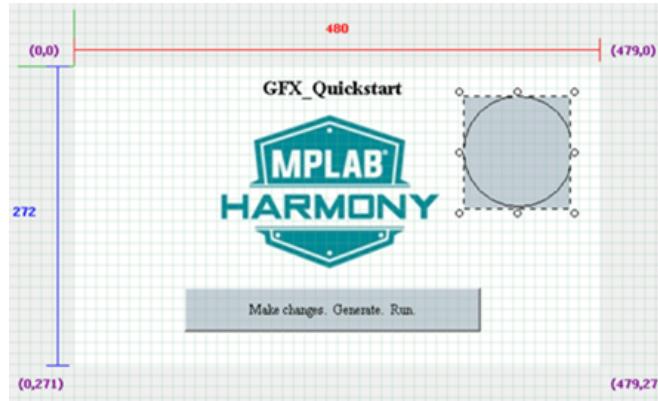


Next, change the fill property on the circle from "None" to "Fill".

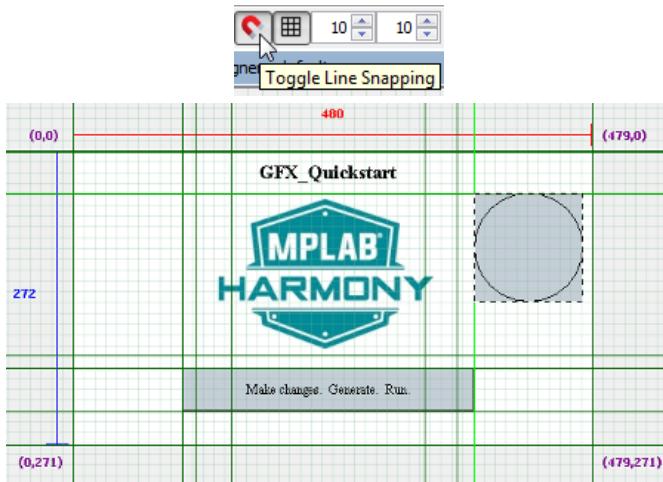


 **Note:** If the properties in the Properties Editor shown are not for CircleWidget1, click on the circle widget to change the focus of the Properties Window.

When done, the screen should now appear, as follows.

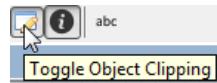


Turn on Line Snapping, which enables drawing guides to assist in aligning widgets on the display.

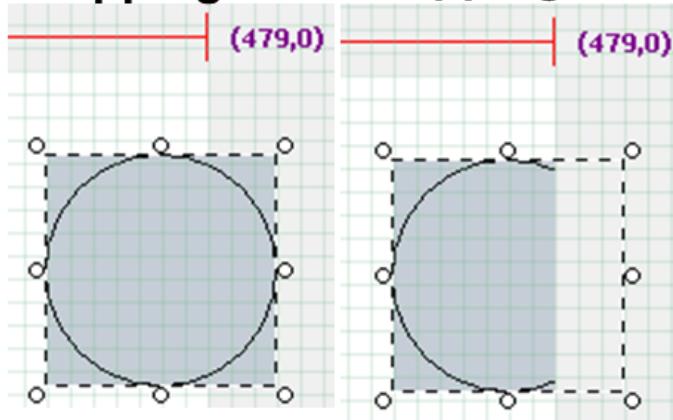


Next, turn on Object Clipping, which shows how the widgets are clipped by the boundaries of the layer that contains them.

Note: Clipping applies to layers, which can be smaller than the display.



Clipping Off Clipping On



To delete a widget, select the widget and press Delete on the keyboard or use the delete icon () on the [Tree View](#) panel.

For more hands-on exploration of graphics using the Aria Quickstart demonstration, see *MPLAB Harmony Graphics Library Help > Quick Start Guides > Graphics and Touch Quick Start Guides > Adding an Event to the Aria Quickstart Demonstration*.

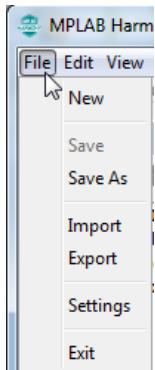
The steps to create a new MPLAB Harmony project with touch input on a SAM E70 Xplained Ultra board with 4.3" WQVGA display can be found in *MPLAB Harmony Graphics Library Help > Quick Start Guides > Graphics and Touch Quick Start Guides > Creating New Graphics Applications*.

Menus

This section provides information on the menus for the MPLAB Harmony Graphics Composer screen.

Description

File Menu



- **New** – Same as the Create New Design tool icon.
- **Save** – Same as the Save Design tool icon.
- **Save As** – Supports exporting the design under a new name. By default, the name is `composer_export.xml`. See [Importing and Exporting Graphics Data](#) for more information.
- **Import** - Reads in (imports) a previously exported design or a `./framework/src/system_config/{board_config}/configuration.xml` file that contains the graphics design to be imported. See [Importing and Exporting Graphics Data](#) for more information.
- **Export** – Same as Save As. See [Importing and Exporting Graphics Data](#) for more information.
- **Settings** – Brings up Project and User Settings dialog, including:
 - Color Settings > Project Color Mode - How colors are managed
 - Color Settings > Using a Global Palette
 - General > Show Welcome Dialog

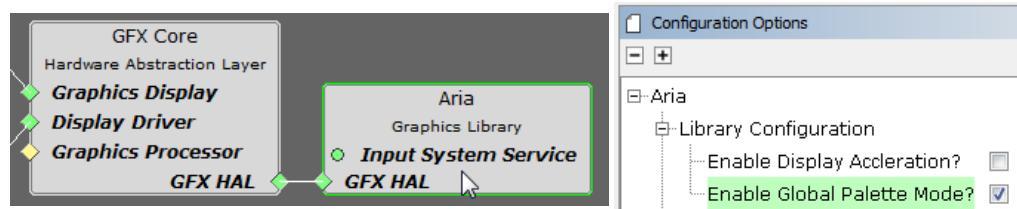
- General > Pre-emption Level – Allows for sharing of the device's cycles with other parts of the application
- General > Hardware Acceleration – Is graphics hardware accelerator enabled in software?
- **Exit** – Closes the MHGC window and exits

The choices for Project Color Mode are:

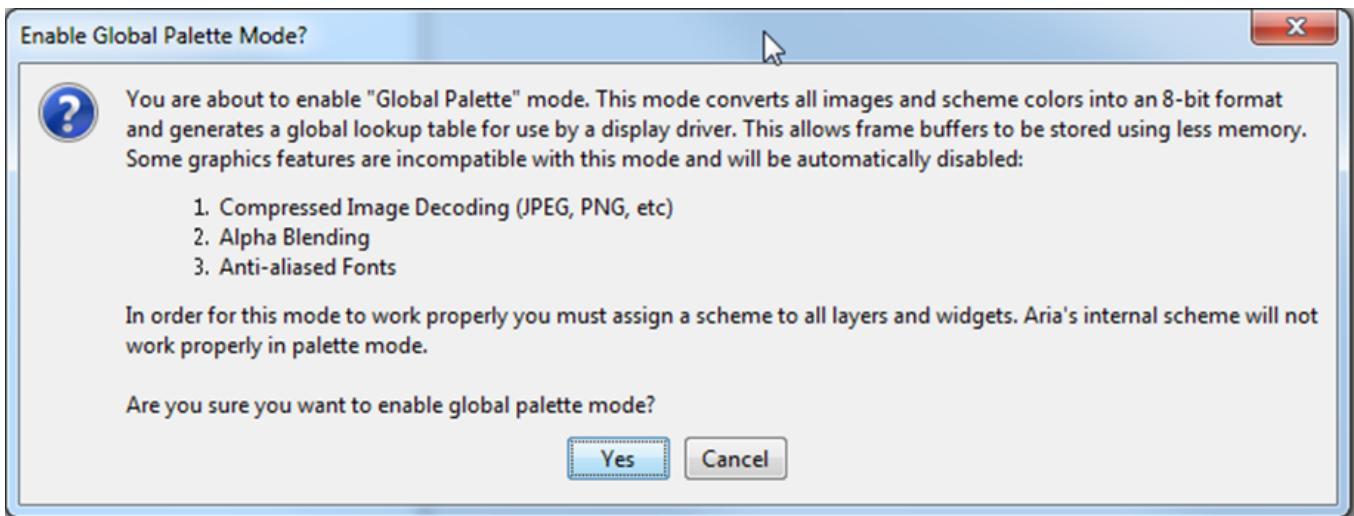
- **GS_8** - 8-bit gray scale
- **RGB_332** - Red/Green/Blue, 3 bits Red/Green, 2 bits Blue
- **RGB_565** - Red/Green/Blue, 5 bits Red, 6 bits Green, 5 bits Blue
- **RGBA_5551** - Red/Green/Blue/Alpha, 5 bits Red/ Green/Blue, 1 bit for Alpha Blending
- **RGB_888** - Red/Green/Blue, 8 bits Red/Green/Blue
- **RGBA_8888** - Red/Green/Blue/Alpha, 8 bits Red/Green/Blue/Alpha Blending
- **ARGB_8888** - Alpha/Red/Green/Blue, 8 bits Alpha Blending/Red/Green/Blue

Ensure that the Project Color Mode chosen is compatible with the display hardware being used; otherwise, the colors shown on the display will not match those shown on the Graphics Composer Screen Designer.

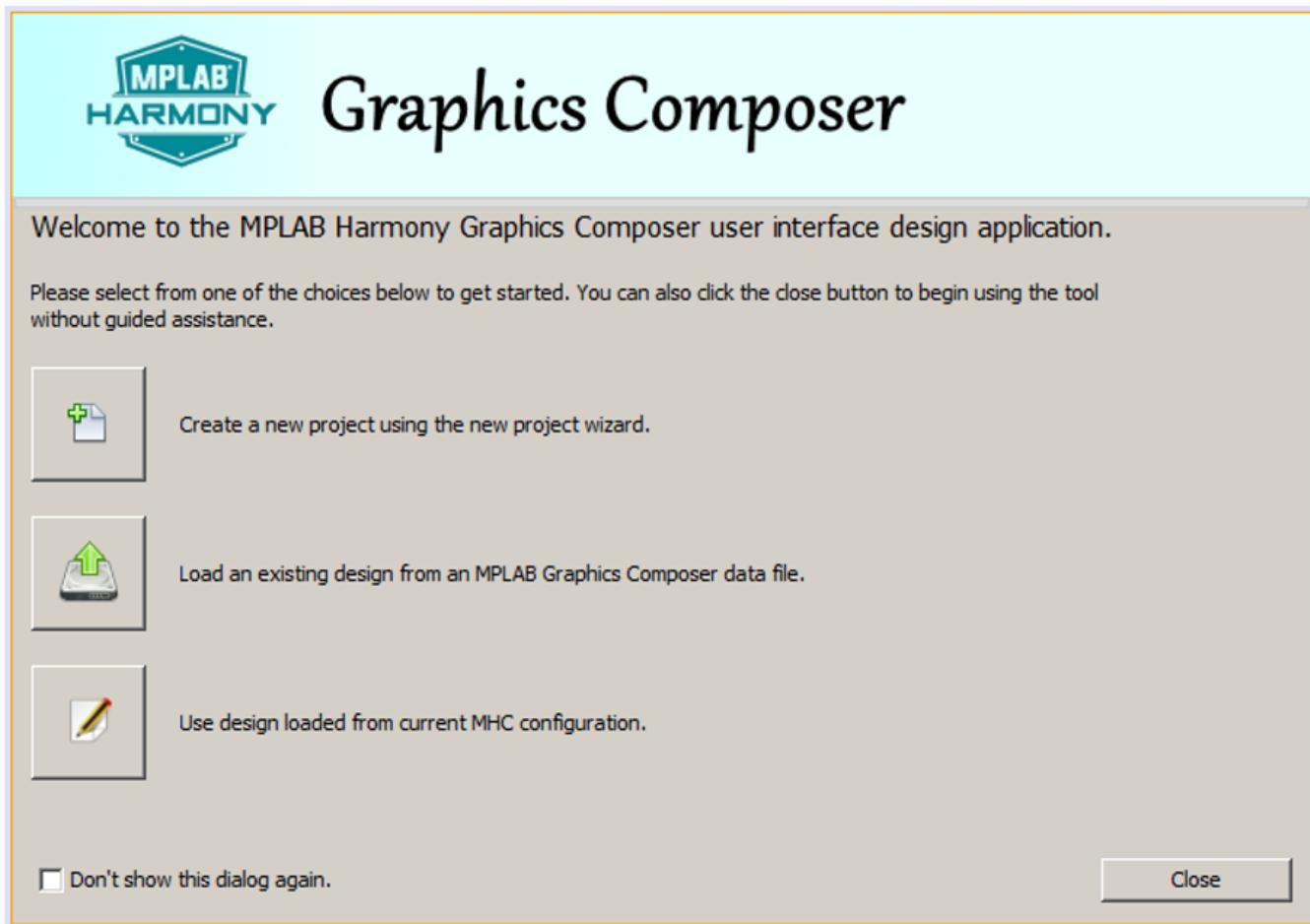
Using a Global Palette enables frame buffer compression for applications using the Low-Cost Controllerless (LCC) Graphics Controller. If the global palette is enabled here, make sure to check the "Enable Global Palette Mode?" in the Configuration Option for the Aria Graphics Library component in MHC's Project Graph:



If **Using a Global Palette** is enabled, the following warning appears.



If **Show Welcome Dialog** is enabled, the following welcome screen appears when launching MHGC.



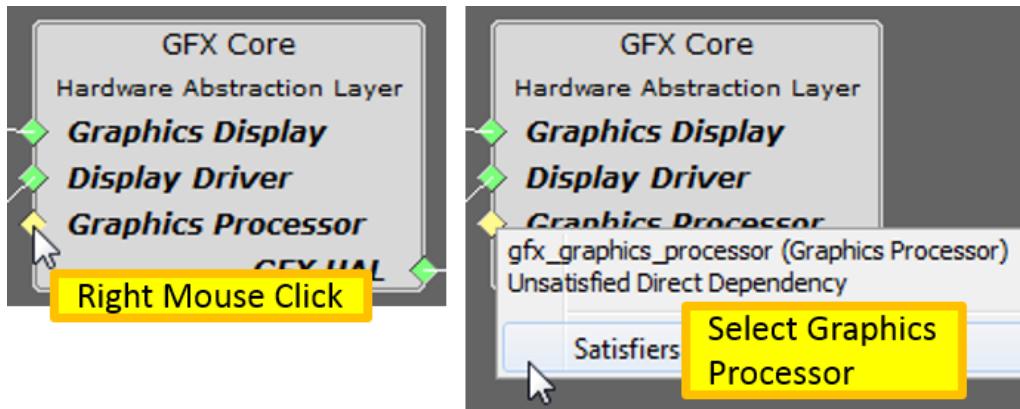
Note: This window's sole purpose is to support creating a new MHGC project.

When the **Preemption Level** is set to zero, all dirty graphics objects are refreshed before the graphics process relinquishes control of the device. (Dirty means needing a redraw.) With the level set to two, graphics provides maximum sharing with the rest of the application, at the cost of slower display refreshes. A level of one provides an intermediate level of sharing.

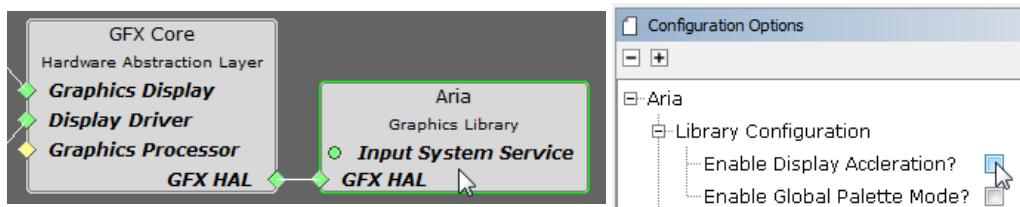
The **Hardware Acceleration** check box determines whether the device's built-in graphics hardware accelerator is used.



Note: Optionally, a graphics hardware accelerator may also be installed in the MHC's Project Graph. For the GFX Core component, do a right mouse click on the Graphics Processor component and then select among the available Graphics Processors:

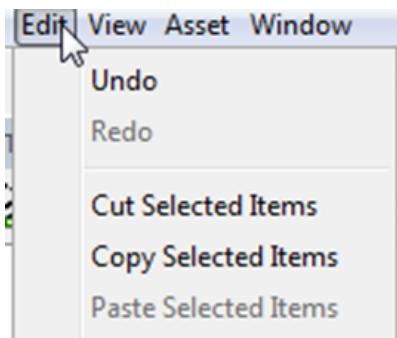


Next, click on the Aria Graphics Library component and in its Configuration Panel enable display acceleration:



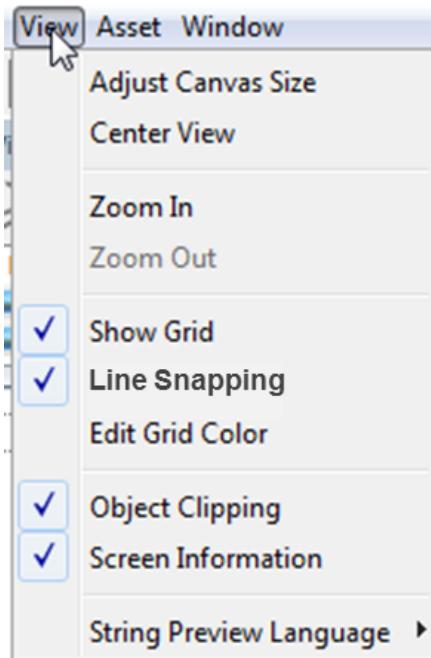
Edit Menu

This menu implements the same functions as the first seven tool icons.



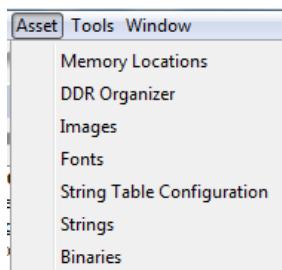
View Menu

This implements the same functions as the remaining tool icons.



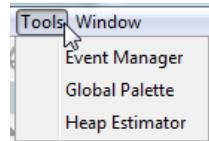
Asset Menu

These menu features are discussed in [Graphics Composer Asset Management](#).



Tools Menu

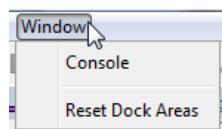
The Event Manager, Global Palette, and Heap Estimator are discussed in [MHGC Tools](#).



Window Menu

Selecting **Console** opens the Output Console for the Graphics Composer. This console panel can be used to debug problems when the Graphics Composer boots up or during its operation.

Selecting **Reset Dock Areas** restores the MHGC panel configuration to the default setup by redocking all of the panels that have been undocked into separate windows.



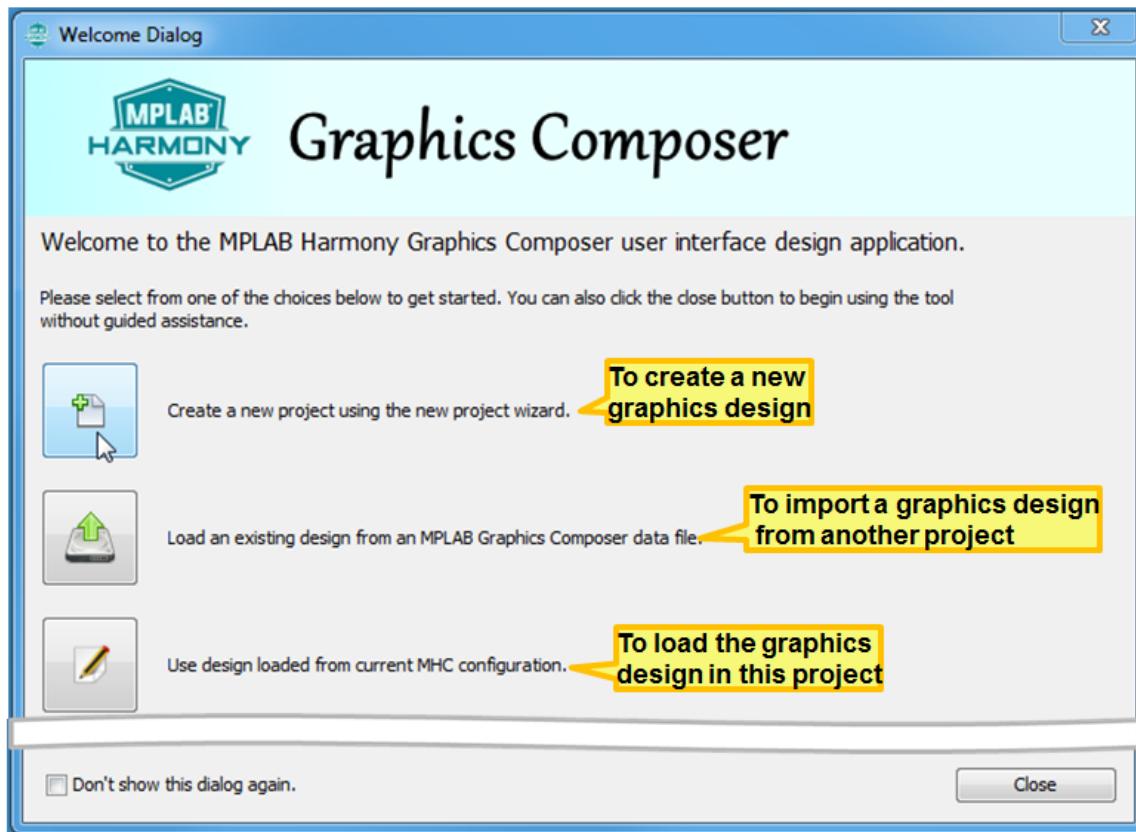
New Project Wizard

The New Project Wizard is launched from the Welcome dialog of the MPLAB Harmony Graphics Composer (MHGC), which supports the creation of a new graphics design, or the importing of an existing graphics design.

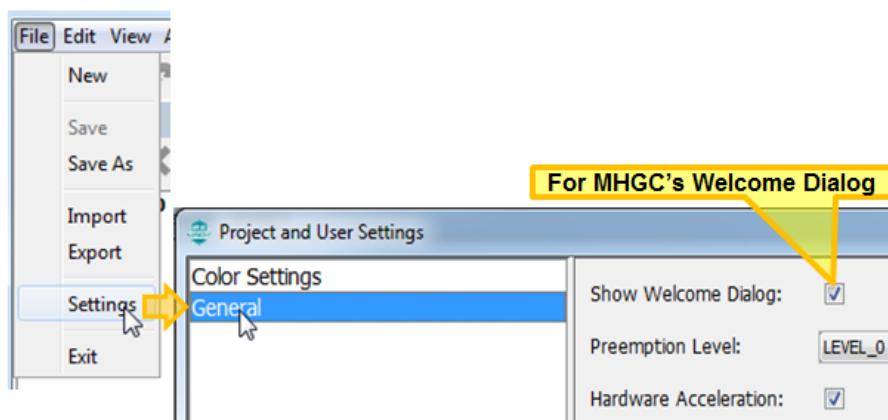
Description

Welcome Dialog window

The Welcome dialog is launched when the Graphics Composer is chosen from the MPLAB Harmony Configurator (MHC) Tools menu. The window has three options:



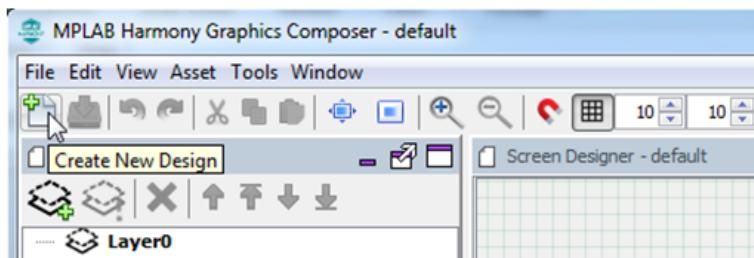
If this window does not appear, it can be re-enabled from MHGC's *File > Settings > General* menu.



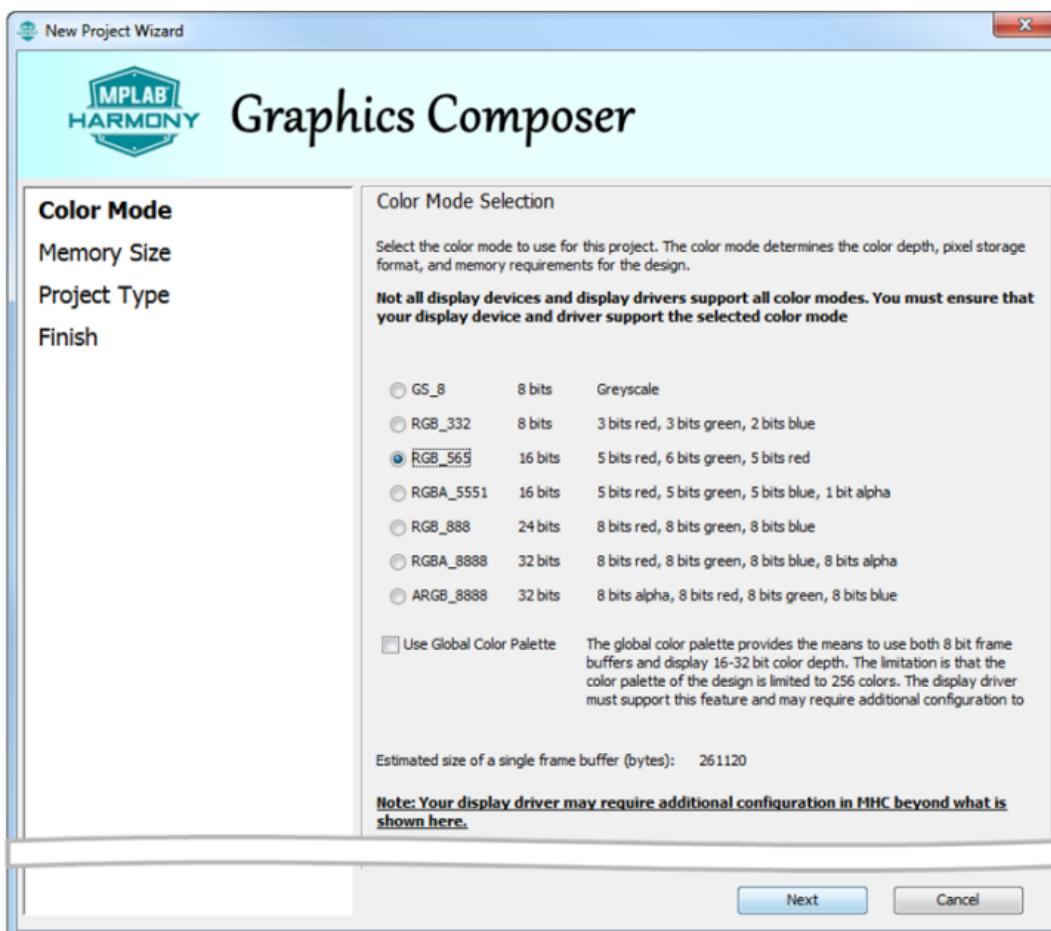
New Project Wizard Windows

Selecting the first icon in the Welcome dialog launches the New Project Wizard. There are four stages in the New Project Wizard: Color Mode, Memory Size, Project Type, and Finish.

The New Project Wizard can also be launched from the first icon (Create New Design) of MHGC's tool bar:



In the Color Mode stage, choose the Display Color Mode for the new graphics design:

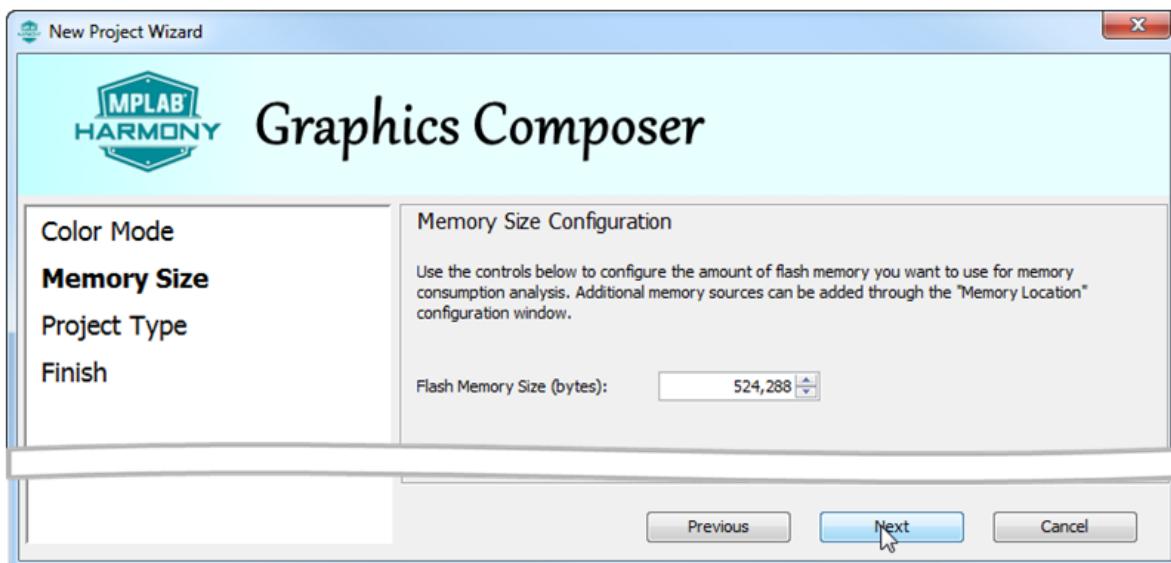


Next

Cancel

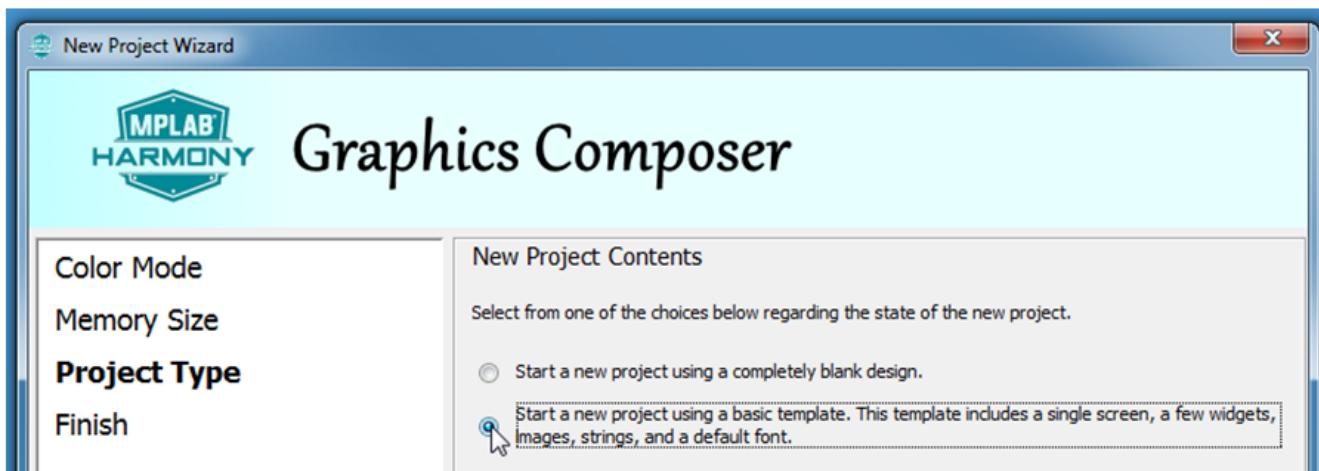
This choice must be supported by the graphics controller defined in the board support package of the project configuration. (If a mistake is made, it can be corrected using MHGC's *File > Settings > Color Settings > Project Color Mode*.) Click **Next** moves the wizard on to the next stage.

The Memory Size stage configures the Program Flash allocated to memory use. This value is only used by the Graphics Composer's Asset menu Memory Configuration tool. The value used in the Memory Size stage can be updated using the Configuration sub-tab of the Memory Configuration tool window.

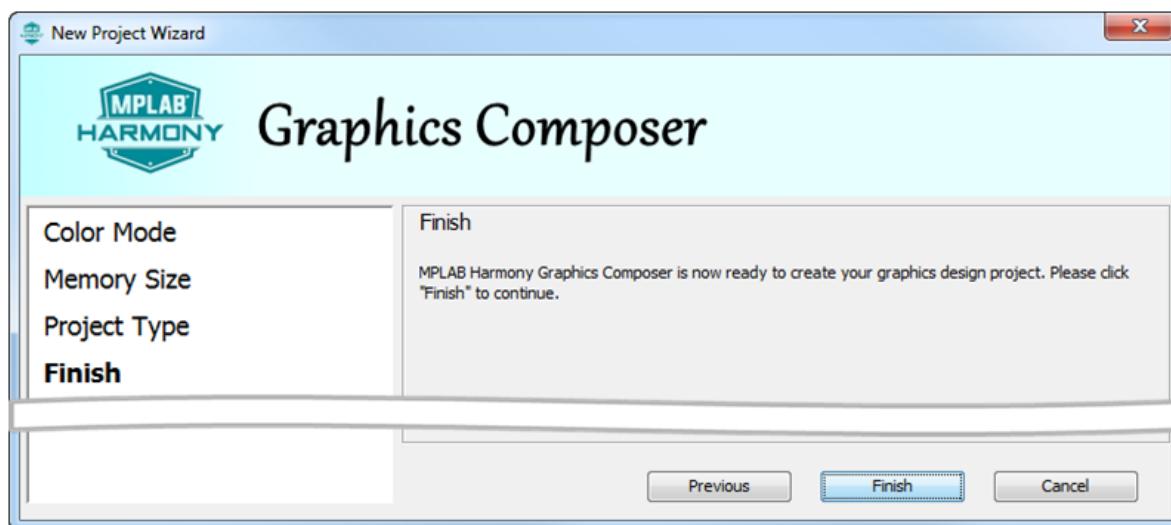


Clicking **Previous** returns to the Color Mode stage and clicking **Next** moves the wizard to the Project Type stage.

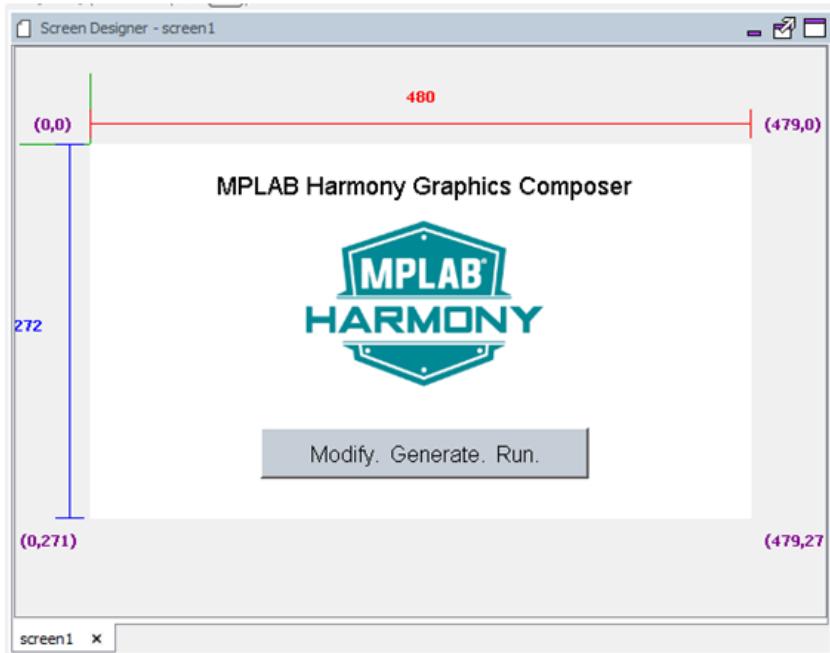
There are two choices at the Project Type stage: A completely blank design, and a template design with a few predefined widgets.



Clicking **Previous** returns to the Memory Size stage, and clicking **Next** moves the wizard to the Finish stage.



If the "Template" project type was chosen, MHGC's Screen Designer will show:



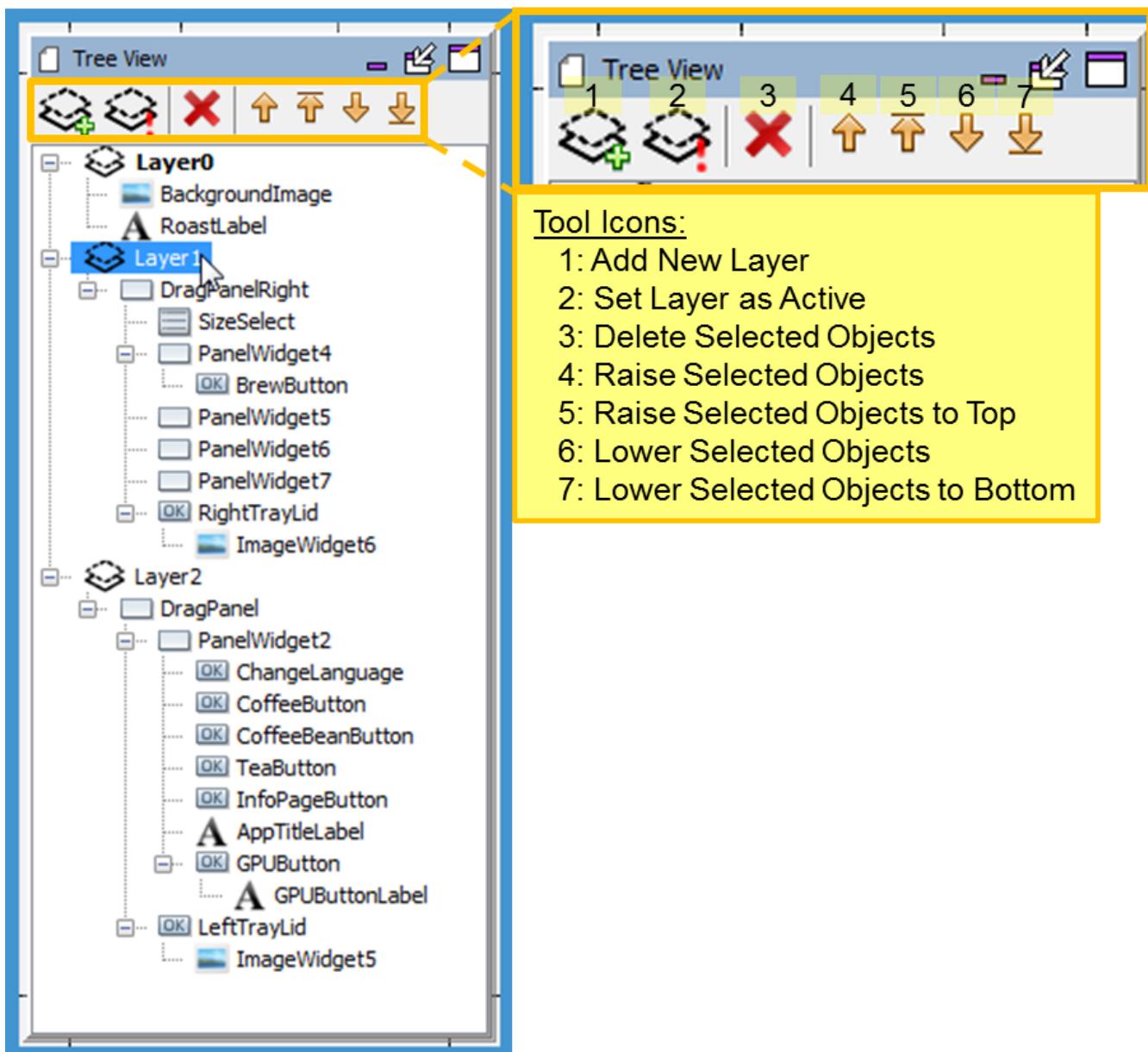
Tree View Panel

The organization of application widgets and layers, including draw order, is managed using this panel.

Description

Example Tree View

The following Tree View (from main screen of the Aria Coffee Maker demonstration shows the tree structure for a screen with three layers.



The tool icons for this panel support layers and managing screen objects (layers/widgets).

Drawing Order and Parent/Child Relationships

The Graphics Composer Tree View panel allows you to organize the widgets per screen in the desired drawing order (z-order). It also allows for the user to organize the widgets into parent – child hierarchies to allow for the paint algorithm to draw the groups together in event of motion or re-draw. Please note that this does not associate or group the widgets by functionality. (Example: a group of radio buttons might not belong to a common parent on the screen.) This parent-child relationship is limited to the widgets location on the screen, motion on the screen and the drawing order on the screen. (Exceptions to this general rule are the Editor > Hidden, Alpha Blending properties, and layer single versus double buffering. These apply to the parent and all the parent's children.)

The tree is traversed depth-first. This means that the z-order goes background (bottom of z-order) to foreground (top of z-order) as we go from top to bottom in the list of widgets, i.e., ImageWidget1, is the widget at the bottom of the z-order and the PanelWidget1 is the topmost widget on the z-order. The tree structure can be arranged and modified by dragging the widgets and releasing it under the desired parent/child. Also, the list can be modified by using the up/down arrows provided at the header of the Composer Widget tree window to traverse the tree.

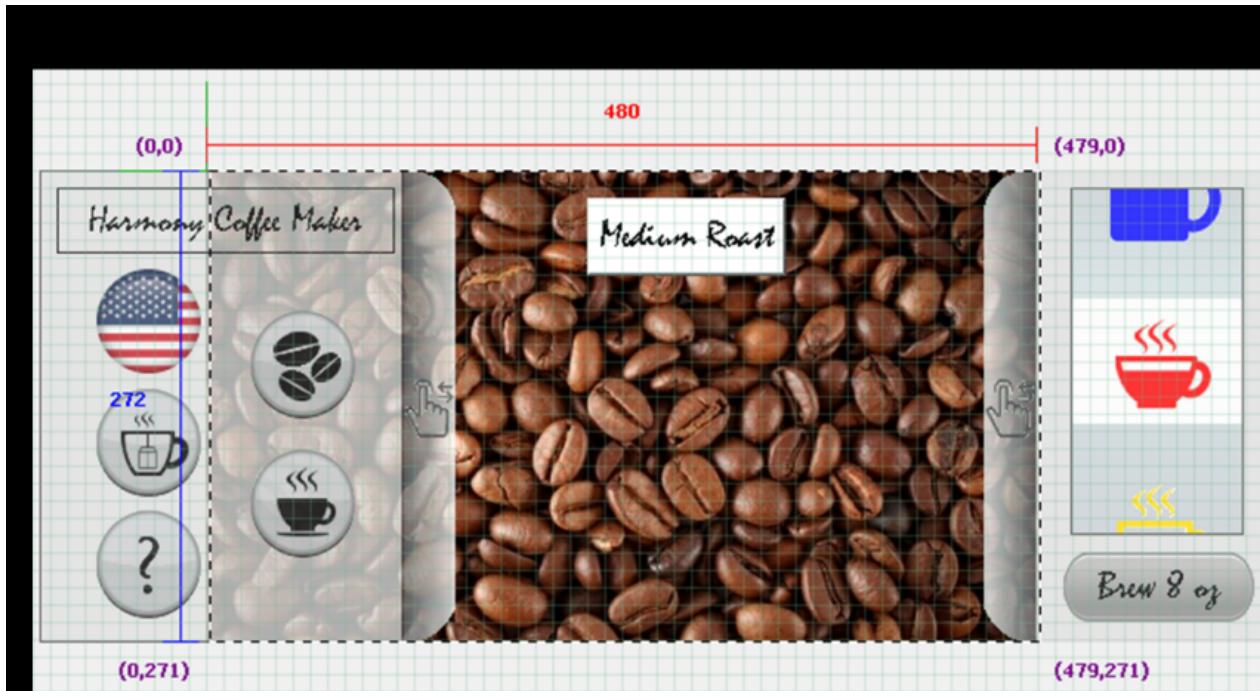
Editor > Hidden Property for Layers

Setting *Editor > Hidden* hides the layer and all its children from the Graphics Composer Screen Designer but does not affect how the layer and its children are displayed when the application is running. This can be useful when designing complex screens with overlapping layers.

Alpha Blending Property for Layers

Enabling Alpha Blending provides the ability to control the transparency of a layer and all its children. The Aria Coffee Maker demonstration provides an example of Alpha Blending. There are three layers (Layer0, Layer1, Layer2) in this demonstration. Layer1 (the drag panel on the right) and Layer2 (the drag panel on the left) have Alpha Blending enabled with Alpha Amount = 225. Setting the Alpha Amount to 255 is the same as disabling Alpha Blending (255 = no transparency). Setting the Alpha Amount to 0 makes the layer invisible (0 = full transparency, i.e., invisible).

The following figure shows the main screen with Alpha Blending = 225.

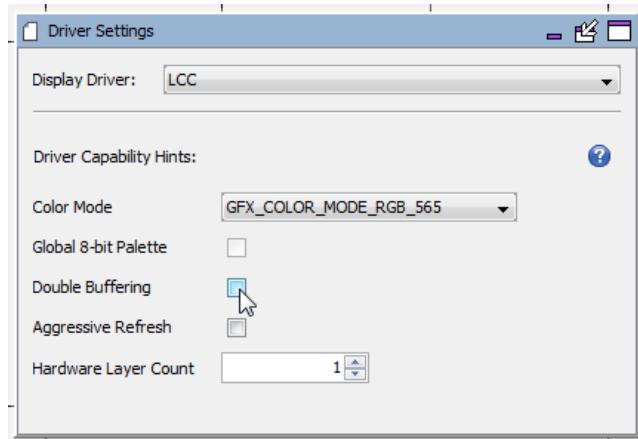


The following figure shows the main screen with Layer 2's Alpha Blending = 255.

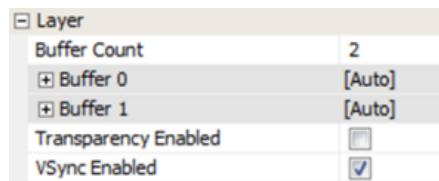


Double Buffering for Layers

Graphics double buffering for the LCC driver is enabled in the Display Manager's Driver Setting screen:



Increasing the Buffer Count of a layer from 1 to 2 enables double buffering for the layer and all its child widgets. To prevent tearing on the display when switching from one buffer to the other, VSync Enabled should also be selected.

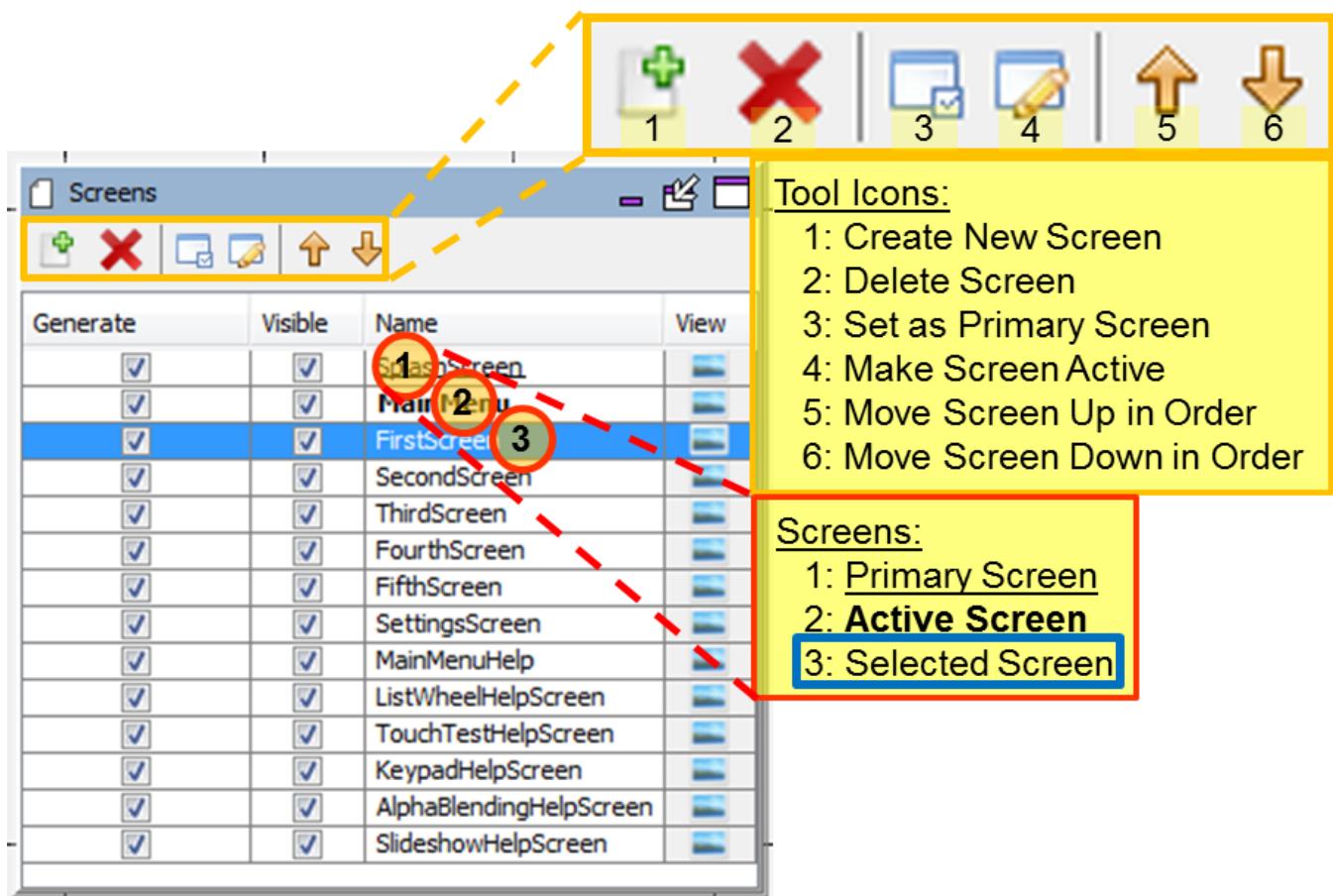


Screens Panel

Application screens are managed using the Screens Panel.

Description

The Screens panel tab manages all the application's screens, as shown in the following figure.



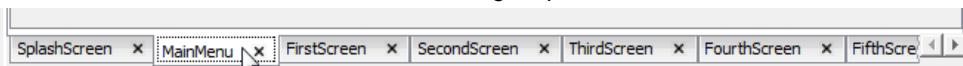
Note: These screens are examples from the [Aria Showcase](#) demonstration project

The underlined screen name identifies the primary screen (in this case, SplashScreen.) The **bold** screen name identifies the currently active screen in the Graphics Composer Screen Designer window (in this case MainMenu.) The blue background identifies the selected screen (i.e., the screen that is manipulated by the tool icons), in this case FirstScreen.

Window Toolbar

The window's tools icons support:

- Create New Screen** – Create a new screen. The user would be prompted for the name of the new screen, which will appear at the bottom of the Screens list.
- Delete Screen** – Delete the selected screen. This removes the selected screen from the application.
- Set as Primary Screen** – Sets the selected screen as the default screen displayed by the application at boot-up.
- Make Screen Active** – This selected screen is displayed in the Screen Designer panel. The active screen can also be selected by clicking on the screen's tab at the bottom of the Screen Designer panel.



- Move Screen Up in Order** – Moves the selected screen up in the list of screens, which is useful in organizing a large list of screens, but has no other significance.
 - Move Screen Down in Order** – Moves the selected screen down in the list of screens.
- Useful in organizing a large list of screens, but has no other significance.

Window Columns

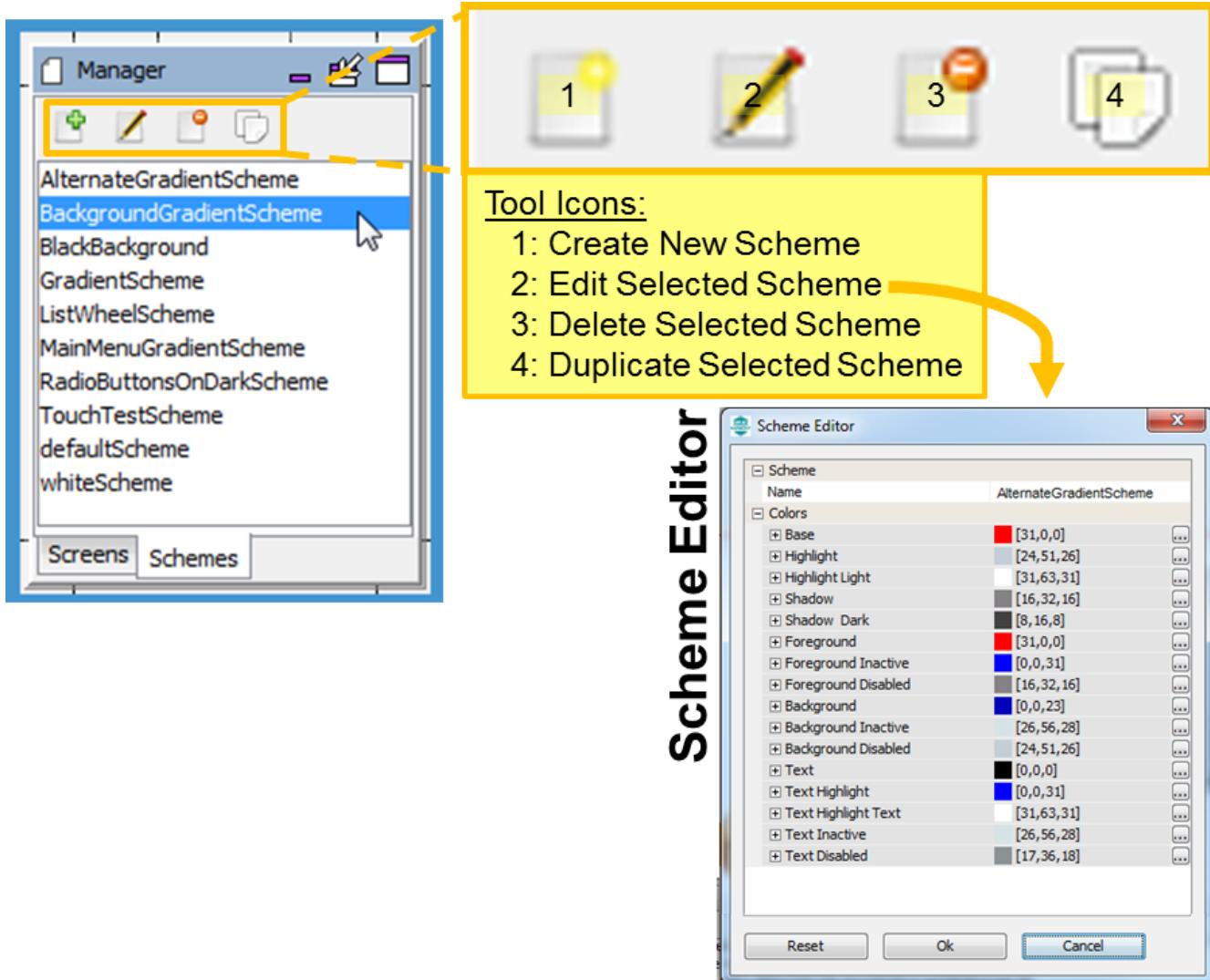
The **Generate** check box is used in selecting those screens that will be included in the application when MPLAB Harmony Configurator (MHC) generates/regenerates the application. (This, along with the Enabled check box for languages, allows customization of the application's build to support different end uses from the same project.) The **Visible** check box can be cleared to hide a screen from the sub-tabs located at the bottom of the Screen Designer. The **View** column provides a mouse-over preview of the screen.

Schemes Panel

Application color schemes are managed using the Schemes Panel.

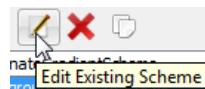
Description

Color schemes for the application's graphics are managed using the Schemes sub-tab.



Editing a Scheme

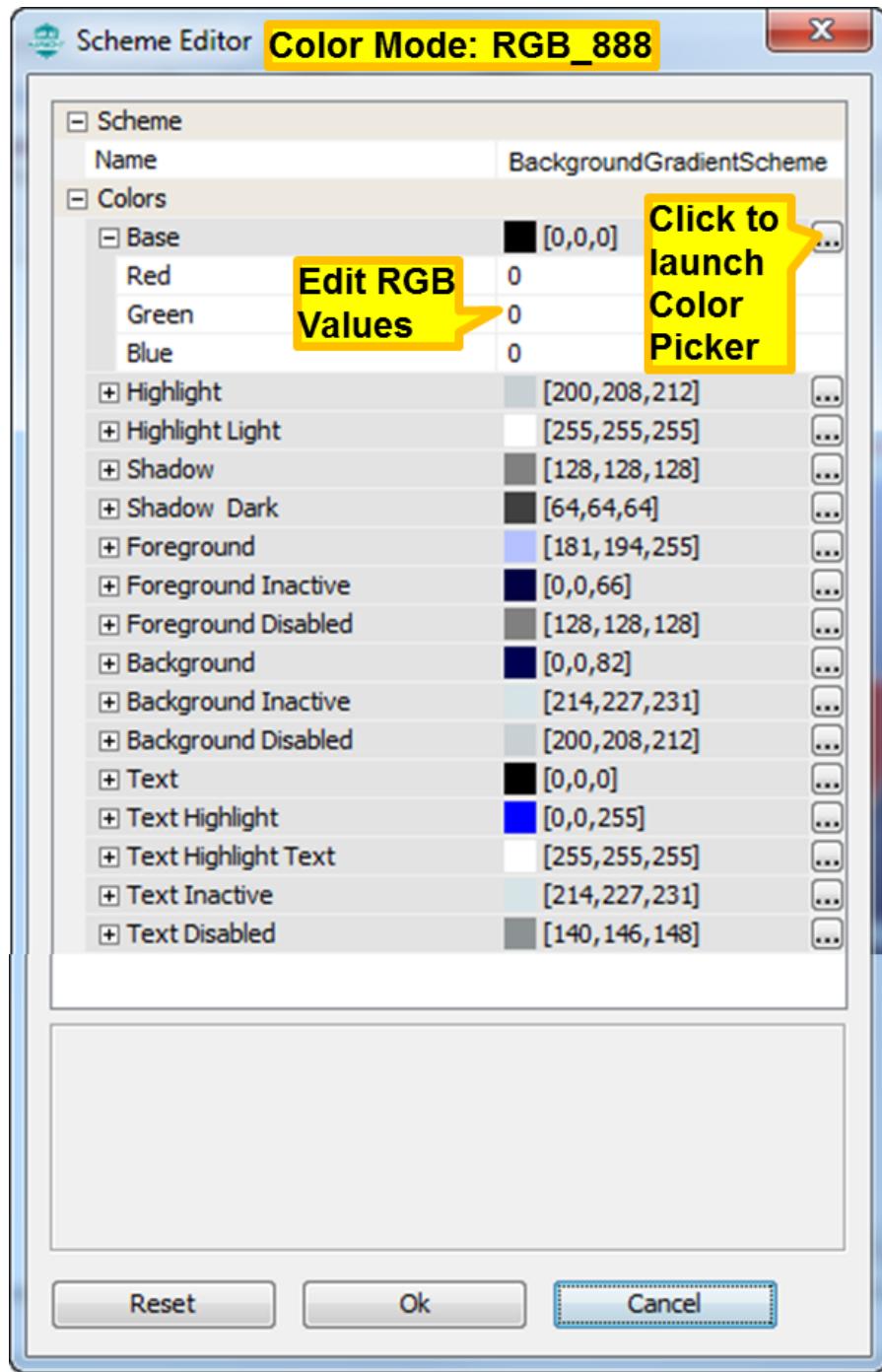
To edit an existing scheme, select the scheme from the list and click **Edit**.



The Scheme Editor dialog appears, which allows change to be made to the colors associated with this display scheme.

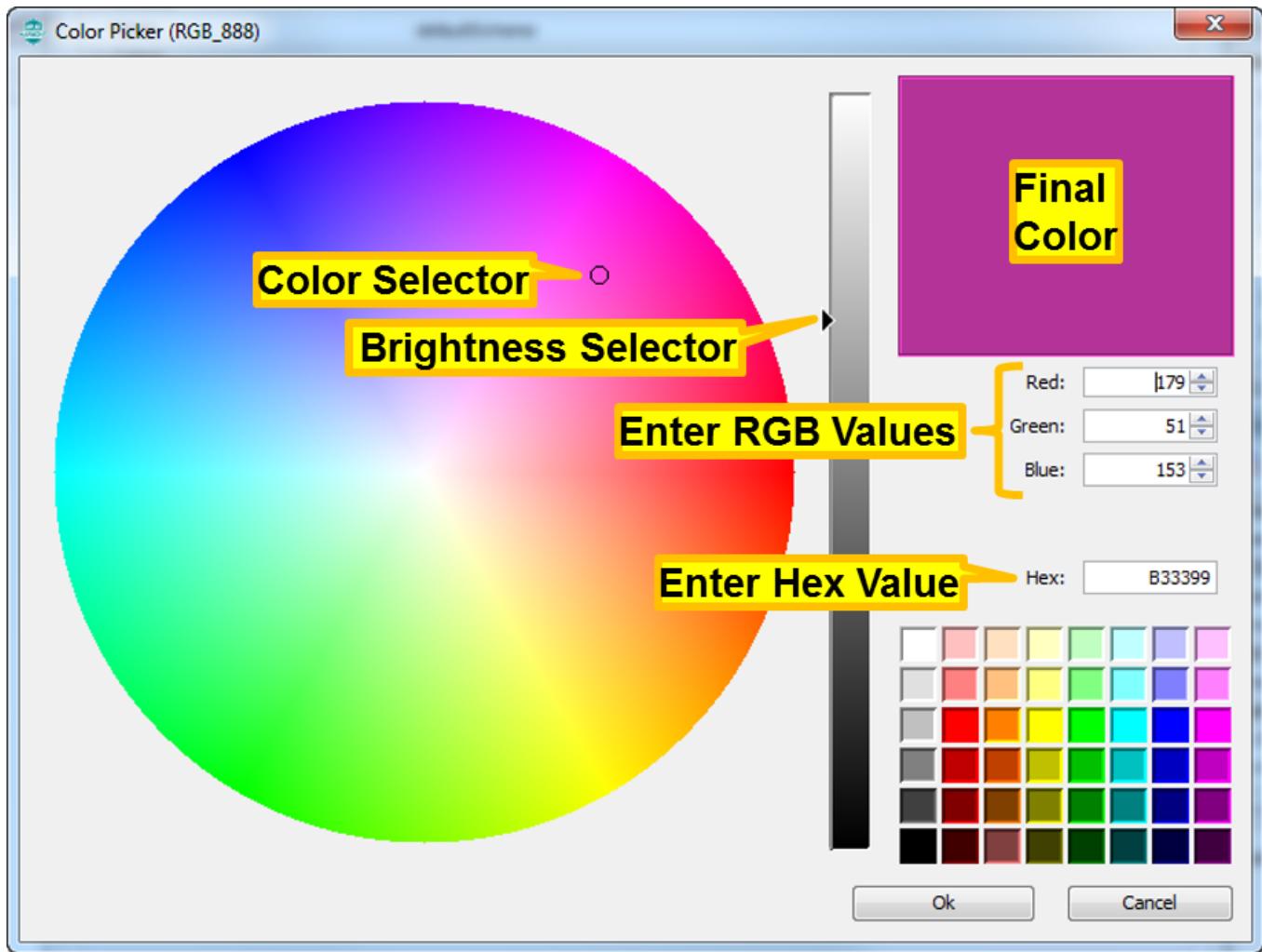
Scheme Editor

The Scheme Editor window supports editing the individual colors of a color scheme. Clicking the ellipsis (...) opens the Color Picker window.



Color Picker

The Color Picker window allows the user to easily select a color by providing a color wheel, brightness gauge, and some common predefined color choices. The user can change the individual color values or input a number in Hexadecimal format. The end result is displayed in the top right corner.



Options

Provides information on the defeatured Options window.

Description

In v2.03b, MPLAB Harmony Graphics Composer user interface provided a third window along with Screens and Schemes, named **Options**. Beginning with v2.04b of MPLAB Harmony, these options are now located within the *File > Settings* menu (see [Menus](#) for details).

Widget Tool Box Panel

The Widget Tool Box panel is the interface by which users add widgets into the screen representation.

Description

All the available graphics widgets are shown in the Widget Tool Box:

MPLAB Harmony Graphics Composer provides automatic code optimization by keeping track of the widgets that are currently being used. When MHC generates or regenerates the application, only the Graphics Library code necessary for the user-created design is included in the project.

There are two primary methods for creating new widget objects: clicking and dragging. To add a new layer to a screen use the Screens sub-tab.

Click Method

The following actions can be performed by using the Click method:

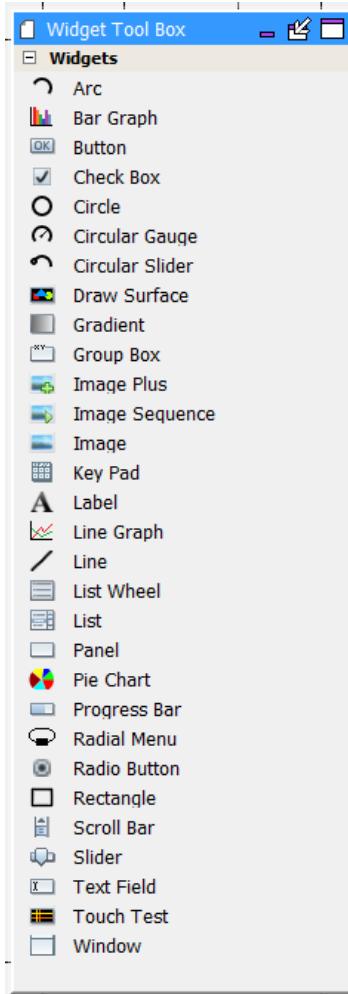
- Clicking an item selects it as active. Users can then move the cursor into the screen window and view a representation of the object about to be added.
- Left-clicking confirms the placement of the new object
- Right-clicking aborts object creation
- Clicking the active item again deactivates it

Drag Method

Dragging and dropping a tool item into the Screen Designer Window creates a new instance of an object. When dragging a tool item, releasing the cursor outside of the Screen Designer Window cancels the drag operation.

Widget List

The Graphics Composer Tool Box is the interface by which users add widgets into the screen representation.



Widget	Example Application
Arc	aria_showcase_reloaded
Bar Graph	aria_showcase_reloaded, aria_weather_forecast
Button	aria_benchmark, aria_showcase, aria_quickstart, aria_weather_forecast
Check Box	aria_showcase_reloaded
Circle	None
Circular Gauge	aria_showcase_reloaded
Circular Slider	aria_showcase_reloaded

Draw Surface	None
Gradient	aria_showcase (background)
Group Box	None
Image	aria_quickstart
Image Plus	None
Image Sequence	aria_showcase
Key Pad	aria_showcase
Label	aria_quickstart
Line	None
Line Graph	aria_showcase_reloaded, aria_weather_forecast
List Wheel	aria_showcase
List	None
Panel	aria_benchmark, aria_showcase
Pie Chart	aria_showcase_reloaded
Progress Bar	aria_flash
Radial Menu	aria_showcase_reloaded
Radio Button	aria_showcase
Rectangle	aria_benchmark
Scroll Bar	None
Slider	aria_showcase
Text Field	aria_showcase
Touch Test	aria_showcase
Window	None

Click Method

The following actions can be performed by using the Click method:

- Clicking an item selects it as active. Users can then move the cursor into the screen window and view a representation of the object about to be added.
- Left-clicking confirms the placement of the new object
- Right-clicking aborts object creation
- Clicking the active item again deactivates it.

Drag Method

Dragging and dropping a tool item into the Screen Designer Window creates a new instance of an object. When dragging a tool item, releasing the cursor outside of the Screen Designer Window cancels the drag operation.

Automatic Code Optimization

MPLAB Harmony Graphics Composer keeps track of the types of widgets that are used and updates the MHGC Tree View panel constantly to ensure that only the Graphics Library code necessary for your design is included in the project.

Widgets

Widgets can be configured by using the [Properties Editor](#) on the right side of the MHGC interface. Each widget has multiple properties to manage their appearance as well as their functioning. Most properties related to appearance are common between widgets, though some widgets require specific property entries.

- **Arc** – A graphical object in the shape of an arc. The arc thickness can be set and filled.
- **Bar Graph** – A graphing widget that shows data in categories using rectangular bars.
- **Button** - A binary On and Off control with events generation for Press and Release state.
- **Check Box** - A selection box with Checked and Unchecked states, and associated events.
- **Circle** - A graphical object in the shape of a circle.
- **Circular Gauge** – A circular widget that operates like a gauge, where the hand/needle position indicates a value.

- **Circular Slider** – A circular widget that can change values based on external input like touch. The slider is filled based on the value of the widget relative to the maximum value.
- **Draw Surface** - A container with a callback from its paint loop. A draw surface lets the application have a chance to make draw calls directly to the HAL during LibAria's paint loop.
- **Gradient** - A draw window that can be associated with a gradient color scheme. This allows for color variation on the window.
- **Group Box** - A container with a border and a text title. With respect to functionality, a group box is similar to a window.
- **Image Sequence** - A special widget that allows image display on screen to be scheduled and sequenced. Select the images to be displayed, and the order for display. A timer to trigger the transitions must be created by calling the image sequence APIs to show the next image from the timer callback function.
- **Image** - Allows an image to be displayed on screen. The size and shape of the widget decides the visible part of the image, as scaling is not enabled for images at this time.
- **Image Plus** - Allows an image to be displayed on screen. The image can be resized (aspect ratio lock is optional). The widget can be set to accept two-finger touch input.
- **Key Pad** - A key entry widget that can be designed for the number of entries divided as specified number of rows and column entries. The widget has a key click event that can be customized.
- **Label** - A text display widget. This does not have any input at runtime capability. A Text Field widget should be used instead when user input is needed.
- **Line** - A graphical object in the shape of a line.
- **Line Graph** – A graphing widget that shows data in categories using points and lines.
- **List Wheel** - Allows multiple radial selections that were usually touch-based selections and browsing.
- **List** - Allows making lists of text and image items. The list contents, number of items, and the sequence can be managed through a List Configuration dialog box in the Properties box.
- **Panel** - A container widget that is a simpler alternative to DrawSurface as it does not have the DrawSurface callback feature.
- **Pie Chart** – A graphing widget that shows data entries as sectors in a circle.
- **Progress Bar** - Displays the progress pointer for an event being monitored through the "Value Changed" event in the [Properties Editor](#).
- **Radial Menu** - A widget that groups any number of images into an elliptical carousel. It can be configured as a touch interactive image carousel or interface menu.
- **Radio Button** - A set of button widgets that are selected out of the group one at a time. The group is specified by the Group property in the [Properties Editor](#).



Note: The radio buttons in the same group must have the same group number specified in their properties.

- **Rectangle** - A graphical object in the shape of a rectangle.
- **Scroll Bar** - Intended to be used with another relevant widget such as the List Wheel to scroll up and down. It has a callback each time the value is changed. The callback allows users to trigger actions to be handled on the scroll value change event.
- **Slider** - Can change values with an external input such as touch. Event callbacks on value change are also available through the [Properties Editor](#).
- **Text Field** - Text input can be accepted into the text field from an external input or from a widget such as keypad. The Event 'Text Changed' in the [Properties Editor](#) is used for accepting the input. This widget can also generate an event when the widget is touched (gains focus) by the user or is no longer touched (loses focus).
- **Touch Test** - Allows tracking of touch inputs. Each new touch input is added to the list of displayed touch coordinates. The input is accepted through the 'Point Added' event callback in the [Properties Editor](#).
- **Window** - A container widget similar to the Panel but has the customizable title bar.

Properties Editor Panel

The properties for all layers and widgets are managed using this panel.

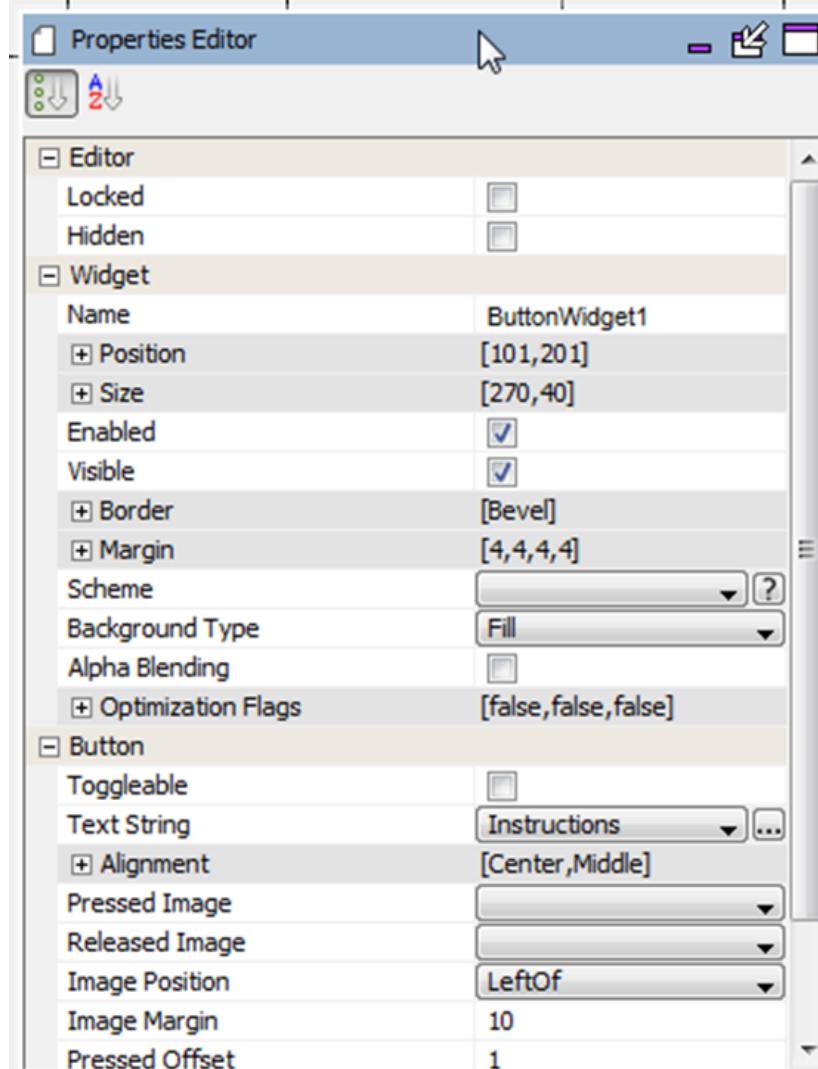
Description

The Properties Editor displays options for the currently-selected object (layer or widget), or the options for the active screen if no objects are selected. To edit an option: left-click the value in the right column and then change the value. Some values have an ellipsis that will provide additional options. In the previous case, the ellipsis button will display the Color Picker dialog.

Some properties, like the screen width and height, are locked and cannot be edited. Other properties offer check boxes and combo-type drop-down box choices. Some properties are grouped together like the Position and Size entries. Individual values of the group can be edited by expanding the group using the plus symbol. For example, the following figure shows properties for a

Button Widget.

A new support feature is the ? icon to the right of the Scheme pull-down, which brings up an “Scheme Helper” for the widget showing how it is colored when using a Bevel border. For a more complete description of widget coloring, see [Widget Colors](#).



Object Properties

Provides information on widget, layer, and screen properties.

Description

Object Properties and Event Actions

Each widget has a structured tree of properties, visible under the MPLAB Harmony Configurator window on the right of the standard window setup within MPLAB X IDE. Most widget properties have a Related Event action that can be used in an event or macro to change or set a property from the application.

Each widget has 3-4 property sets:

Editor – Controls the behavior of layers and widgets under the MPLAB Harmony Graphics Composer Suite Editor.

Property Name	Type	Description	Related Event Actions
Locked	Boolean	Locks the object (widget), preventing changes by the designer. Only affects the object (widget) in the editor.	N/A
Hidden	Boolean	Hides the widget and its children in the designer window. Only affects the appearance of the widget in the editor.	N/A

Active	Boolean	For layers only. Sets the layer as active. Any objects (widgets) added to the screen will be added to this layer.	N/A
Locked to Screen Size	Boolean	For layers only. Locks the layer size to the size of the display's screen.	N/A

Widget – Controls the behavior of screens, layers, and widgets on the display.

Property Name	Type	Description	Related Event Actions
Name	String	Editable name for each object. By default, widgets are named NameWidget1, ..., NameWidgetN. For example: ButtonWidget1, ButtonWidget2,	N/A
Position	[X,Y] Pair of Integers	Location on the layer of the upper left corner of the widget or the location on the display of the upper left corner of the layer. Measured in display pixels. X is measured from left-to-right and Y is measured from up-to-down from the upper left corner of the parent object (typically a Layer or Panel).	Adjust Position, Set X Position, Set Y Position
Size	[X,Y] Pair of Integers	X: Width, Y: Height of object, in display pixels.	Adjust Size, Set Size, Set Width, Set Height
Enabled	Boolean	Is the object enabled? Disabled objects are not built into the display's firmware.	Set Enabled
Visible	Boolean	Is the object visible by default? Object visibility can be manipulated in firmware using laWidget_GetVisible and laWidget_SetVisible .	Set Visible
Border	Widget Border	Choices are: { None Line Bevel }.	Set Border Type
Margin	Integer	Four integers ([Left,Top,Right,Bottom]) defining the widget's margins on the display, in display pixels.	Set Margins
Scheme	-	Color scheme assigned to the layer or widget. Blank implies the default color scheme.	Set Scheme
Background Type	-	Sets the background of the layer or widget. Choices are { None Fill Cache }. In MPLAB Harmony v2.03, this type was Boolean. Now, Off = None, On = Fill. With Fill selected, the widget's background is one solid color. With Cache selected, a copy (cache) of the framebuffer is created before the widget is drawn and this cache is used to fill the background of the widget. This supports transparent widgets in front of complex widgets, such as JPEG images. Instead of rerendering the JPEG image, it is just drawn from the cache.	Set Draw Background
Alpha Blending	Boolean	Is alpha blending enabled for this layer or widget and all of its children? If enabled, specify the amount of alpha blending as an 8-bit integer. Zero makes the object invisible, whereas 255 makes the background invisible.	N/A

Widget Advanced – Advanced control of layers and widgets

Optimization Sub-Property Name	Type	Description	Related Event Actions
Draw Once	Boolean	Indicates that the widget should draw once per screen Show Event. All other attempts to invalidate or paint the widget will be rejected.	N/A
Force Opaque	Boolean	Provides a hint to the renderer that the entire area for this widget is opaque. Useful for widgets that may use something like an opaque image to fill the entire widget rectangle despite having fill mode set to None. This can help reduce unnecessary drawing.	N/A

Local Redraw	Boolean	Provides a "hint" to the widget's renderer that the widget is responsible for removing old pixel data. This can avoid unnecessary redrawing.	N/A
--------------	---------	--	-----



Use Local Redraw only if you know what you're doing!

Important!

Widget Name (e.g., Button Check Box, Circle, etc.) – Optional properties tied to each widget. See [Dedicated Widget Properties and Event Actions](#).

Events – Associates widget events with event call-backs. For example, you can enable and specify a button pressed event and button release event for the Button widget.

For each event you specify:

- Enabled/Disabled Check box – To enable or disable (default) the event.
- Event Callback – Selected from the Event Editor Action List.

There are additional Event actions that do not correspond to any specific property:

- Set Parent – Set the parent of the object, including no parent.

Dedicated Widget Properties and Event Actions

Arc Widget

Property Name	Type	Description	Related Event Actions
Radius	Integer	The outside radius of the arc.	Set Radius
Start Angle	Integer	The starting angle of the arc in degrees.	Set Start Angle
Center Angle	Integer	The center angle of the arc in degrees. A positive angle draws the arc counter-clockwise from the start angle. A negative angle draws clockwise.	Set Center Angle
Thickness	Integer	The thickness of the arc fill, measured from the radius to center. (radius – thickness) determines the inside radius.	Set Thickness
Round Edge	Boolean	Draws round arc edge.	Set Round Edge

Bar Graph Widget

Property Name	Type	Description	Related Event Actions
Stacked	Boolean	Stacks the bars for the entries in a category	Set Stacked Bars
Tick Length	Integer	The length, in pixels, of the ticks on each axis	Set Tick Length
Fill Graph Area	Boolean	Fills the graph area with scheme base color	Fill Graph Area

Value Axis Configuration	Axis Integer	Configures the value (Y) axis The maximum value of the axis The minimum value of the axis The intervals between major ticks The interval between minor ticks Show/Hide the major ticks Position of major ticks on the value axis. Choices are: {Inside Center Outside} Show/Hide the tick labels Show/Hide the minor ticks Position of minor ticks on the value axis. Choices are: {Inside Center Outside} Show/Hide the gridlines The string asset containing the numeric characters for the tick labels. The asset must contain the characters for numbers 0 to 9.	Set Max Value Set Min Value Set Tick Interval Set Subtick Interval Show Value Axis Ticks Set Value Axis Ticks Position Show Value Axis Labels Show Value Axis Subticks Set Value Axis Subticks Position Show Value Axis Gridlines Set Labels String
Category Axis Configuration	Axis Boolean	Configures the category (X) axis Show/Hide the ticks Show/Hide the category labels Position of the ticks on the category axis. Choices are: {Inside Center Outside}	Show Category Axis Ticks Show Category Axis Labels Set Category Axis Ticks Position
Category Configuration Dialog	(See Description)	The Category Configuration Dialog lets users add categories to the line graph. The following properties can be set: <ul style="list-style-type: none">Label – String Asset. The label to show for each category	None
Data Configuration Dialog	(See Description)	The Data Configuration Dialog lets users add and configure data series to the line graph. The following properties can be set: <ul style="list-style-type: none">Scheme – Scheme. The color scheme of the data seriesCategory Values – Integer. Values in series for each category	None

Button

Property Name	Type	Description	Related Event Actions
Toggleable	Boolean	Is button toggle enabled?	Set Toggleable
Pressed	Boolean	If Toggleable is enabled, provide default state of the button. This can be used to see the colors of an asserted button.	Set Press State
Text String	-	Select widget's text string from the Select String Dialog.	Set Text
Alignment:	-	Text string alignment within the button object. Horizontal alignment. Choices are: { Left Center Right }. Vertical alignment. Choices are: { Top Middle Bottom }.	Set Horizontal Alignment Set Vertical Alignment
Pressed Image	-	Select image used for pressed state. Default: no image.	Set Pressed Image
Released Image	-	Select image used for released state. Default: no image.	Set Released Image
Image Position	-	Position of image relative to button text. Choices are: { LeftOf Above RightOf Below Bottom }.	Set Image Position
Pressed Offset	Integer	Offset of button contents when pressed. In Pixels. The X and Y position of the button contents is offset by this amount.	Set Pressed Offset

Check Box

Property Name	Type	Description	Related Event Actions
Text String	-	Select widget's text string from the Select String Dialog.	Set Text
Alignment:	-	Text string alignment within the button object. Horizontal alignment. Choices are: { Left Center Right }. Vertical alignment. Choices are: { Top Middle Bottom }.	Set Horizontal Alignment Set Vertical Alignment
Checked	Boolean	Default state of the check box.	Set Check State
Unchecked Image	-	Select image used for widget's unchecked state. Default: no image.	Set Unchecked Image
Checked Image	-	Select image used for the widget's checked state. Default: no image.	Set Checked Image
Image Position	-	Position of image relative to check box text. Choices are: { LeftOf Above RightOf Below Bottom }.	Set Image Position
Image Margin	Integer	Space between image and text. In Pixels.	Set Image Margin

Circle

Property Name	Type	Description	Related Event Actions
X	Integer	X offset of circle's center, from widget's upper left hand corner, in pixels.	N/A
Y	Integer	Y offset of circle's center, from widget's upper left hand corner, in pixels.	N/A
Radius	Integer	Circle's radius, in pixels.	Set Radius
Thickness	Integer	Thickness of circle, in pixels, from the outside edge inwards	N/A
Filled	Boolean	Specifies if the center area of the circle is filled with the palette's background color	N/A

Circular Gauge Widget

Property Name	Type	Description	Related Event Actions
Radius	Integer	The outside radius of circular gauge.	Set Radius
Start Angle	Integer	The starting angle of the circular gauge in degrees.	Set Start Angle
Center Angle	Integer	The center angle of the circular gauge in degrees. A positive value draws the gauge counter-clockwise. Clockwise if negative.	Set Center Angle
Start Value	Integer	The start value of the circular gauge.	Set Start Value
End Value	Integer	The end value of the circular gauge.	Set End Value
Value	Integer	The value of the circular gauge.	Set Value
String Set	String Asset	The string asset containing the numeric characters for the tick labels. The asset must contain the characters for numbers 0 to 9.	-
Major Ticks Configuration		Configures the major ticks. Shows/Hides the major ticks.	Show/Hide Ticks
• Ticks Visible	Boolean	The length of ticks in pixels.	Set Tick Length
• Tick Length	Integer	The interval between ticks.	Set Tick Value
• Tick Value	Integer	Shows/Hides the major tick labels.	Show/Hide Tick Labels
• Tick Labels Visible	Boolean		

Hand Configuration • Hand Visible • Hand Radius • Center Circle Visible • Center Circle Radius • Center Circle Thickness	Boolean Integer Integer Integer Integer	Configures the gauge hand/needle. Shows/Hides the gauge hand/needle. Sets the length of the hand in pixels Shows/Hides the hand center circle. Sets the radius of the center circle in pixels Sets the thickness of the center circle in pixels.	Show/Hide Hand Set Hand Radius/Length Show/Hide Center Circle Set Center Circle Radius Set Center Circle Thickness
Advanced Configuration	-	Additional widget configuration options for adding minor ticks, labels and arcs.	-
Minor Ticks Configuration Dialog	(See Description)	The Minor Ticks configuration lets users add minor ticks to the widget. The following properties can be set: <ul style="list-style-type: none">• Start Value – Integer. The value where the first tick starts• End Value – Integer. The value where the last tick ends• Interval – Integer. The interval between ticks• Radius – The radius in pixels where the ticks will be drawn from• Length – The length of the ticks in pixels, drawn from the radius towards the center• Scheme – The color scheme for the ticks	None
Minor Tick Labels Configuration Dialog	(See Description)	The Minor Ticks configuration lets users add minor tick labels to the widget. The following properties can be set: <ul style="list-style-type: none">• Start Value – Integer. The value where the first tick label is drawn• End Value – Integer. The value where the last tick ends• Interval – Integer. The interval between ticks• Radius – Integer. The radius, in pixels, where the tick labels will be drawn from• Position – Enum, choices are {Outside Inside}. Position of the label relative to the radius• Scheme – The color scheme for the ticks	None
Arcs Configuration Dialog	(See Description)	The Arcs configuration lets users draw arcs in the gauge widget. The arcs can be used to colorize regions or range of values in the gauge. The following properties can be set for each arc: <ul style="list-style-type: none">• Type – Enum, choices are {VALUE ANGLE}. A value type arc is drawn relative to the values in the gauge. An angle type arc is drawn based on the angles and is not affected by the values in the gauge.• Start – Integer. The start value or angle of the arc• End – Integer. The end value or angle of the arc• Thickness – Integer. The thickness of the arc in pixels, filled inward from the radius towards the center• Radius – Integer. The radius of the arc in pixels• Scheme. The color scheme of the arc	None

Circular Slider Widget

Property Name	Type	Description	Related Event Actions
Radius	Integer	The outside radius of circular slider.	Set Radius
Start Angle	Integer	The start angle of the circular slider, in degrees.	Set Start Angle
Start value	Integer	The start value of the circular slider.	Set Start Value
End Value	Integer	The end value of the circular slider.	Set End Value

Value	Type	Description	Set Value
Border Circle Configuration <ul style="list-style-type: none">• Show Outside Circle• Outside Circle Thickness• Show Inside Circle• Inner Circle Thickness	Boolean Integer Boolean Integer	Configures the border circle. Shows/Hides the outside circle border. The thickness of the outside circle border in pixels. Shows/Hides the inside circle border. The thickness of the inside circle border in pixels.	Show/Hide Outside Border Set Outside Border Thickness Show/Hide Inside Border Set Inside Border Thickness
Active Area Configuration <ul style="list-style-type: none">• Fill Active Slider Area• Round Edges• Active Slider Area Thickness• Inner Circle Thickness	Boolean Boolean Integer Integer	Configures the slider active area. Fills the active slider area. Draws a round edge for the active area. The thickness of the slider active area in pixels. The thickness of the inside circle border in pixels.	Show/Hide Active Arc Area Set Round Edges Set Active Arc Area Thickness Show/Hide Inactive Arc Area
Button Configuration <ul style="list-style-type: none">• Show Circular Button• Sticky Button• Touch on Button Only• Circular Button Radius• Circular Button Thickness	Boolean Boolean Boolean Integer Integer	Configures the slider button. Shows/Hides the circular slider button. If set, the button sticks when it reaches the start/end values. If set, the widget responds to touches within the button area only. The radius of the circular button in pixels. The thickness of the of the circular button border in pixels.	Show/Hide Circular Button Set Sticky Button None Set Circular Button Radius Set Circular Button Thickness

Draw Surface – No additional properties.

Gradient

Property Name	Type	Description	Related Event Actions
Direction	-	Gradient draw direction. Choices are: { Right Down Left Up }.	Set Direction

Group Box

Property Name	Type	Description	Related Event Actions
Text String	-	Select widget's text string from the Select String Dialog.	Set Text
Alignment	-	Text string alignment within the widget. Choices are: { Left Center Right }.	Set Alignment

Image Sequence

Property Name	Type	Description	Related Event Actions
Sequence Configuration Dialog	-	Specify image sequence by using the Image Sequence Configuration Dialog window.	Set Entry Image, Set Entry Horizontal Alignment, Set Entry Vertical Alignment, Set Entry Duration, Set Image Count
Starting Image	Integer	Selects the first image to be shown.	Set Active Image
Play By Default	Boolean	Will image sequence play automatically?	N/A
Repeat	Boolean	Should the image sequence repeat?	Set Repeat Additional related event actions: , Show Next, Start Playing, Stop Playing.

Image Widget

Property Name	Type	Description	Related Event Actions
Image	-	Select image used.	Set Image
Alignment:	-	Image alignment within the image object. Horizontal alignment. Choices are: { Left Center Right }. Vertical alignment. Choices are: { Top Middle Bottom }.	Set Horizontal Alignment Set Vertical Alignment

Image Plus Widget

Property Name	Type	Description	Related Event Actions
Image	-	Select Image used	Set Image
Resize Fit	To Boolean	Resize the image to fill the size of the widget area	Toggles option to best fit the image to the widget area
Interactive	Boolean	Makes the widget interactive, allowing the image to be translated, stretched and zoomed	Toggles option to permit two-finger gestures to interact with the widget

Key Pad

Property Name	Type	Description	Related Event Actions
Row Count	Integer	Number of key pad rows.	None.
Column Count	Integer	Number of key pad columns.	None.
Key Pad Configuration Dialog	(see Description)	The Key Pad dialog window has the following: <ul style="list-style-type: none">• Width – Integer. Width of each key, in pixels.• Height – Integer. Height of each key, in pixels.• Rows – Integer. Number of key rows. A duplicate of Row Count.• Columns – Integer. Number of key columns. A duplicate of Column Count.	None. None. None. None.
-	-	Selecting one of the keys on the key pad diagram displays the Cell Properties for that key: <ul style="list-style-type: none">• Enabled – Boolean. Disabled cells (keys) are made invisible.• Text String – Select key's text string from the Select String Dialog.• Pressed Image – Select image used for pressed state. Default: no image.• Released Image – Select image used for released state. Default: no image.• Image Position – Position of image relative to key text. Choices are: { LeftOf Above RightOf Below Behind }.• Image Margin – Integer. Space between image and text. In Pixels.• Draw Background – Boolean. Controls whether the key should fill its background rectangle.• Editor Action – Select the generic editor action that fires when the key is clicked. Choices are: { None Accept Append Editor Value String } Other Key Event Actions:	Set Key Enabled Set Key Text Set Key Pressed Image Set Key Released Image Set Key Image position Set Key Image Margin None. Set Key Action Set Key Value Set Key Background Type

Label

Property Name	Type	Description	Related Event Actions
Text String	-	Select widget's text string from the Select String Dialog.	Set Text
Alignment:	-	Text string alignment within the widget. Horizontal alignment. Choices are: { Left Center Right }. Vertical alignment. Choices are: { Top Middle Bottom }.	Set Horizontal Alignment Set Vertical Alignment
Custom Spacing	Boolean	Enables custom spacing between lines	N/A
Line Spacing (pixels)	Integer	A positive value will adjust the spacing between lines	N/A

Line

Property Name	Type	Description	Related Event Actions
Start X	Integer	X start of line, in pixels, from upper left hand corner of the widget.	Set Start Point Position
Start Y	Integer	Y start of line, in pixels, from upper left hand corner of the widget.	Set Start Point Position
End X	Integer	X end of line, in pixels, from upper left hand corner of the widget.	Set End Point Position.
End Y	Integer	Y end of line, in pixels, from upper left hand corner of the widget.	Set End Point Position.

Line Graph Widget

Property Name	Type	Description	Related Event Actions
Stacked	Boolean	Stacks the values of the entries in a category	Set Stacked Points
Tick Length	Integer	The length of the ticks on each axis	Set Tick Length
Fill Graph Area	Boolean	Fills the graph area with scheme base color	Fill Graph Area
Fill Series Area	Boolean	Fills the series area with series scheme base color	Fill Series Area
Value Axis Configuration			
• Maximum Value	Integer	Configures the value (Y) axis	Set Max Value
• Minimum Value	Integer	The maximum value of the axis.	Set Min Value
• Tick Interval	Integer	The minimum value of the axis.	Set Tick Interval
• Subtick Interval	Integer	The intervals between major ticks.	Set Subtick Interval
• Show Ticks	Boolean	The interval between minor ticks.	Show Value Axis Ticks
• Tick Position	Enum	Show/Hide the major ticks.	Set Value Axis Ticks Position
• Show Tick Labels	Boolean	Position of major ticks on the value axis. Choices are: {Inside Center Outside}.	Show Value Axis Labels
• Show Subticks	Boolean	Show/Hide the tick labels.	Show Value Axis Subticks
• Subtick Position	Enum	Show/Hide the minor ticks.	Show Value Axis Subticks Position
• Show Gridlines	Boolean	Position of minor ticks on the value axis. Choices are: {Inside Center Outside}.	Show Value Axis Gridlines
• String Set	String Asset	Show/Hide the gridlines.	Set Value Axis Subticks Position
		The string asset containing the numeric characters for the tick labels. The asset must contain the characters for numbers 0 to 9.	Set Labels String

Category Axis Configuration		Configures the category (X) axis Show/Hide the ticks Show/Hide the category labels Position of the ticks on the category axis. Choices are: {Inside Center Outside}	Show Category Axis Ticks Show Category Axis Labels Set Category Axis Ticks Position
Category Configuration Dialog	(See Description)	The Category Configuration Dialog lets users add categories to the line graph. The following properties can be set: <ul style="list-style-type: none">Label – String Asset. The label to show for each category	None
Data Configuration Dialog	(See Description)	The Data Configuration Dialog lets users add and configure data series to the line graph. The following properties can be set: <ul style="list-style-type: none">Scheme – Scheme. The color scheme of the data seriesPoint Type – Enum. The point indicator to use for the series. Choices are: {None Circle Square}Fill Points – Boolean. Fills the points with series scheme foreground colorDraw Lines – Boolean. Draws lines between points in the series using series scheme foreground colorCategory Values – Integer. Values in series for each category	None

List

Property Name	Type	Description	Related Event Actions
Selection Mode	-	Select list selection mode. Choices are: {Single Multiple Contiguous}.	Set Selection Mode
Allow Empty Selection	Boolean	Is a list selection allowed to be empty?	Set Allow Empty Selection
Alignment	-	Horizontal text alignment. Choices are: { Left Center Right }.	Set Item Alignment
Icon Position	-	Position of list icons relative to list text. Choices are: { LeftOf RightOf }.	Set Icon Position
Icon Margin	-	Space between icon and text, in pixels.	Set Icon Margin
List Configuration Dialog	-	Defines the string and icon image for each entry in the list.	Set Item Icon, Set Item Icon (actually sets item text). Additional Related Event Actions: Deselect All Items, Insert Item, Remove All Items, Remove Item, Select All Items, Set Item Selected, Toggle Item Select(ed).

List Wheel

Property Name	Type	Description	Related Event Actions
Alignment	-	Sets horizontal text alignment. Choices are: { Left Center Right }.	Set Item Alignment
Icon Position	-	Position of icons relative to text. Choices are: { LeftOf RightOf }.	Set Icon Position
Icon Margin	Integer	Sets the space between icon and text. In pixels.	Set Icon Margin
Selected Index	Integer	Selects the default list item.	Set Selected Index

List Configuration Dialog	-	Defines the image/text for each entry in the list.	Set Item Icon, Set Item Icon (actually sets item text) Additional Related Event Actions: Append Item, Insert Item, Remove All Items, Remove Item, Select Next Item, Select Previous Item.
---------------------------	---	--	--

Panel – No additional properties.

Pie Chart Widget

Property Name	Type	Description	Related Event Actions
Start Angle	Integer	The starting angle of the pie chart in degrees.	Set Start Angle
Center Angle	Integer	The center angle of the pie chart in degrees. A positive value draws the chart counter-clockwise. Clockwise if negative.	Set Center Angle
Labels Visible	Boolean	Shows/Hides the labels for each data	Show/Hide Labels
Labels Offset	Integer	The position of the labels relative to the center of the pie chart, in pixels.	Set Label Offset
String Set	String Asset	The string asset containing the numeric characters for the tick labels. The asset must contain the characters for numbers 0 to 9.	Set Label String ID
Data Configuration Dialog	(See Description)	The Data Configuration Dialog lets users add data entries to the pie chart. The following properties can be set: <ul style="list-style-type: none">• Value – Integer. The value of the entry• Radius – Integer. The radius, in pixels, of the pie for the entry• Offset – Integer. The offset, in pixels, of the pie from the center• Scheme – The color scheme for the ticks	None

Progress Bar

Property Name	Type	Description	Related Event Actions
Direction	-	Direction of progress bar. Choices are: { Right Down Left Up }.	Set Direction
Value	-	Default value of the progress bar. The primitives <code>laProgressBarWidget_GetValue</code> and <code>laProgressBarWidget_SetValue</code> can be used to manipulate the widget's value during run time.	Set Value

Radial Menu Widget

Property Name	Type	Description	Related Event actions
Ellipse Visible	Boolean	Show the elliptical track of the widget	Elliptical track gets draw in Harmony Composer simulation and at runtime.
Highlight Prominent	Boolean	Highlights the prominent item when the widget rotation has completed its reset to the static, selectable position by drawing a rectangle behind the prominent item.	-

Ellipse Type	Enum	Selects the type of elliptical track Default – an elliptical track that best fits the widget area based on the size of the tallest and widest images with the size scale settings factored-in. Orbital – a “flatter” elliptical track that is best used with the Theta setting for a tilted look Rolodex – a vertical track with Theta setting locked at 90 degrees	Locks Theta to 90 degrees when Rolodex is selected
Theta	Integer	The angle (in degrees) of tilt relative to the y-axis of the ellipse. The number range is 0 to 90 degrees.	This field is only valid for Default and Orbital Ellipse Type setting. It is locked at 90 when Rolodex is selected.
a	Integer	This is the half-length (in pixels) of the 0-180 axis of ellipse. It is auto-calculated based on the widget size, the tallest image's height, the ellipse type and scale settings.	-
b	Integer	This is the half-length (in pixels) of the 90-270 axis of ellipse. It is auto-calculated based on the widget size, the widest image's width, the ellipse type and scale settings.	-
Size Scale Configuration • Size Scale	Enum	Off – all images displays at its original size Gradual – images in the very back are scale to the Minimum Size Modifier setting, the scale is gradually increased, with the prominent front item scaled to the Maximum Size Modifier setting Prominent – the image that is at the front, prominent location is scaled based on the Maximum Size Modifier, all other images are scaled to the Minimum Size Modifier setting The value (in percent) for the widget to resize the image to. When Size Scale is set to Gradual, this value represents the lowest scale for the item in the back. When Size Scale is set to Prominent, this value represents the scaling value for every image in the widget except for the prominent item. This value is equal to or less than the Maximum Size Modifier value	-
* Minimum Size Modifier	Integer	The value (in percent) for the widget to resize the image to. When Size Scale is set to Gradual, this value represents the largest scale for the item in the front (prominent position). When Size Scale is set to Prominent, this value represents the scaling value for the prominent item. This value is equal to or greater than the Minimum Size Modifier value	-
* Maximum Size Modifier	Integer		-

Item List Configuration • Total Number of Items Shown	Integer	The number images visible on the radial menu. This number does not may be less than or equal to the total images in the widget.	The widget automatically space-out the images along the elliptical track base on this value.
* Total Number of Widget Items	Integer (See Description)	The total number of images the widget contains. The Widget Items Configuration Dialog lets users add images to the widget. The follow properties can be set: <ul style="list-style-type: none">• Image – Image Asset. The image to show for the widget item	If this number is greater than Total Number of Items Shown, some of the images will be hidden in a FIFO queue in the back
* Widget Items Configuration Dialog			-
Touch Area Configuration • Show Touch Area	Boolean	Show visually in Harmony Graphics composer the rectangular area that permits touch interaction.	This setting is for preview in Harmony Graphics composer only. The touch area is not rendered at runtime.
* Touch Area X Offset	Integer	The X-coordinate in local space of the touch-allowed area for the widget. This is auto-calculated based on the Touch Area Width Percent.	-
* Touch Area Y Offset	Integer	The Y-coordinate in local space of the touch-allowed area for the widget. This is auto-calculated based on the Touch Area Height Percent.	-
* Touch Area Width Percent	Integer	The percentage of the width of the touch-allowed area as compared to the entire widget area.	-
* Touch Area Height Percent	Integer	The percentage of the height of the touch-allowed area as compared to the entire widget area. The default value is 50.	If this value is less than 100 percent, the area is horizontally centered.

Radio Button

Property Name	Type	Description	Related Event Actions
Text String	-	Select widget's text string from the Select String Dialog.	Set Text
Alignment: <ul style="list-style-type: none">• Horizontal• Vertical	-	Text string alignment within the widget. Horizontal alignment. Choices are: { Left Center Right }. Vertical alignment. Choices are: { Top Middle Bottom }.	Set Horizontal Alignment Set Vertical Alignment

Group	Integer	Radio Button Group Number. Default is -1, indicating no group. Only one radio button in a group can have a default selected value of On. All others in the group are Off	N/A
Selected	Boolean	If selected, the button has a default value of On. All other buttons in the group have a Selected value of Off.	Select
Selected Image	-	Select image used for selected state. Default: no image.	Set Selected Image
Unselected Image	-	Select image used for unselected state. Default: no image.	Set Unselected Image
Image Position	-	Position of image relative to widget text. Choices are: { LeftOf Above RightOf Below Behind }.	Set Image Position
Image Margin	-	Space between radio button image and text, in pixels.	Set Image Margin
Circle Button Size	-	The diameter of the default circle button, in pixels	Set Circle Button Size

Rectangle

Property Name	Type	Description	Related Event Actions
Thickness	Integer	Line thickness in pixels.	Set Thickness

Scroll Bar

Property Name	Type	Description	Related Event Actions
Orientation	-	Scroll bar orientation. Choices are: { Vertical Horizontal }.	Set Orientation
Maximum	Integer	Maximum scroll value (minimum = 0.)	Set Maximum Value
Extent	Integer	Length of scroll bar slider, re scroll bar maximum value. Indicates the number of lines or size of window visible at each scroll setting.	Set Extent
Value	Integer	Initial scroll bar value.	Set Value, Set Value Percentage
Step Size	Integer	Step size value of scroll bar arrow buttons. (Min = 1, Max = 9999).	Set Step Size Additional Related Event Actions: Step Backward, Step Forward

Slider

Property Name	Type	Description	Related Event Actions
Orientation	-	Orientation of the slider. Choices are: { Vertical Horizontal }.	Set Orientation
Minimum	-	Minimum slider value.	Set Minimum Value
Maximum	-	Maximum slider value.	Set Maximum Value
Value	-	Initial slider value.	Set Value, Set Value Percentage
Grip Size	-	Grip size of slider, from 10 to 9999, in pixels.	Set Grip Size Additional Related Event Actions: Step

Text Field

Property Name	Type	Description	Related Event Actions
Text String	-	Select widget's text string from the Select String Dialog.	Clear Text followed by Append Text

Alignment	-	Horizontal alignment. Choices are: { Left Center Right }.	Set Alignment
Cursor Enable	-	Boolean. Show blinking cursor while editing.	Set Cursor Enabled
Cursor Delay	-	Cursor delay in milliseconds. From 1 to 999,999.	Set Cursor Delay Additional Related Event Actions: Accept Text, Append Text, Backspace, Clear Text, Start Editing.

Touch Test – No dedicated properties.

Window

Property Name	Type	Description	Related Event Actions
Title String	-	Select widget's title string from the Select String Dialog.	Set Title
Icon Image	-	Select image used. Default: no image.	Set Icon
Image Margin	Integer	Space between icon and title, in pixels.	N/A

Layer Properties and Event Actions

The property list for a graphic layer is close in look and feel to that of a widget. Each Layer has three property sets: Editor (see above), Widget (see above), and Layer (see below).

Layer Properties

Property Name	Type	Description	Related Event Actions
Transparency Enabled	Boolean	Automatically mask out pixels of with a specified color. If enabled Specify:	N/A
Mask Color	Integer	Red/Green/Blue or Red/Green/Blue/Alpha color value	N/A
All Input Passthrough	Boolean	Allow input events to pass through this layer to layers behind it.	N/A
VSync Enabled	Boolean	Layers should swap only during vertical syncs.	N/A
Buffer Count	Integer	Integer number of frame buffers associated with this layer, either 1 or 2.	N/A
Buffer N	-	For each buffer (N= 1 or 2) you specify:	-
Allocation Method	-	Buffer allocation method. Choices are: { Auto Address Variable Name } <ul style="list-style-type: none">• Auto – Automatically allocate frame buffer space• Address – Specify a memory address• Variable Name – Use variable name as buffer location	N/A
Memory Address	-	If Address is the allocation method, specify the raw (physical) memory address as a hexadecimal number.	N/A
Variable Name	String	If Variable name is the allocation method, specify the variable name as a string value.	N/A

Screen Properties and Events

The property list for a screen shares the Name and Size properties with Layers and Widgets but has these unique properties.

Screen Properties

Property Name	Type	Description	Related Event Actions
Orientation	-	Display orientation: 0, 90, 180, 270 Degrees. This can also be set using the Display Manager.	N/A
Mirrored	Boolean	Enables screen mirroring.	N/A
Layer Swap Sync	Boolean	Enables that all layer buffer swapping happen at the same time, delaying lower layers until higher layers are finished drawing as well. For example, assume you make changes to layer 0 and layer 1 and you want to see those changes show up on the screen at the same time. Without this option you'd see layer 0's changes as soon as it finishes when layer 1 has not yet started drawing. This option will hold layer 0's swap operation until layer 1 finishes as well.  Note: Currently, this property is only supported by the GLCD Graphics Controller Driver and is ignored by all other drivers.	N/A
Persistent	Boolean	Indicates that the screen should not free its widgets and memory when it is hidden. This results in faster load times and persistent data, but at the cost of higher memory consumption.	N/A
Export	Boolean	Includes this screen the application build. This can also be set using the Screens panel.	N/A
Primary	Boolean	Sets this screen as the primary screen. The primary screen is the first screen displayed when the application starts. This can also be done using the Screens Panel Generate check box.	N/A

Graphics Composer Asset Management

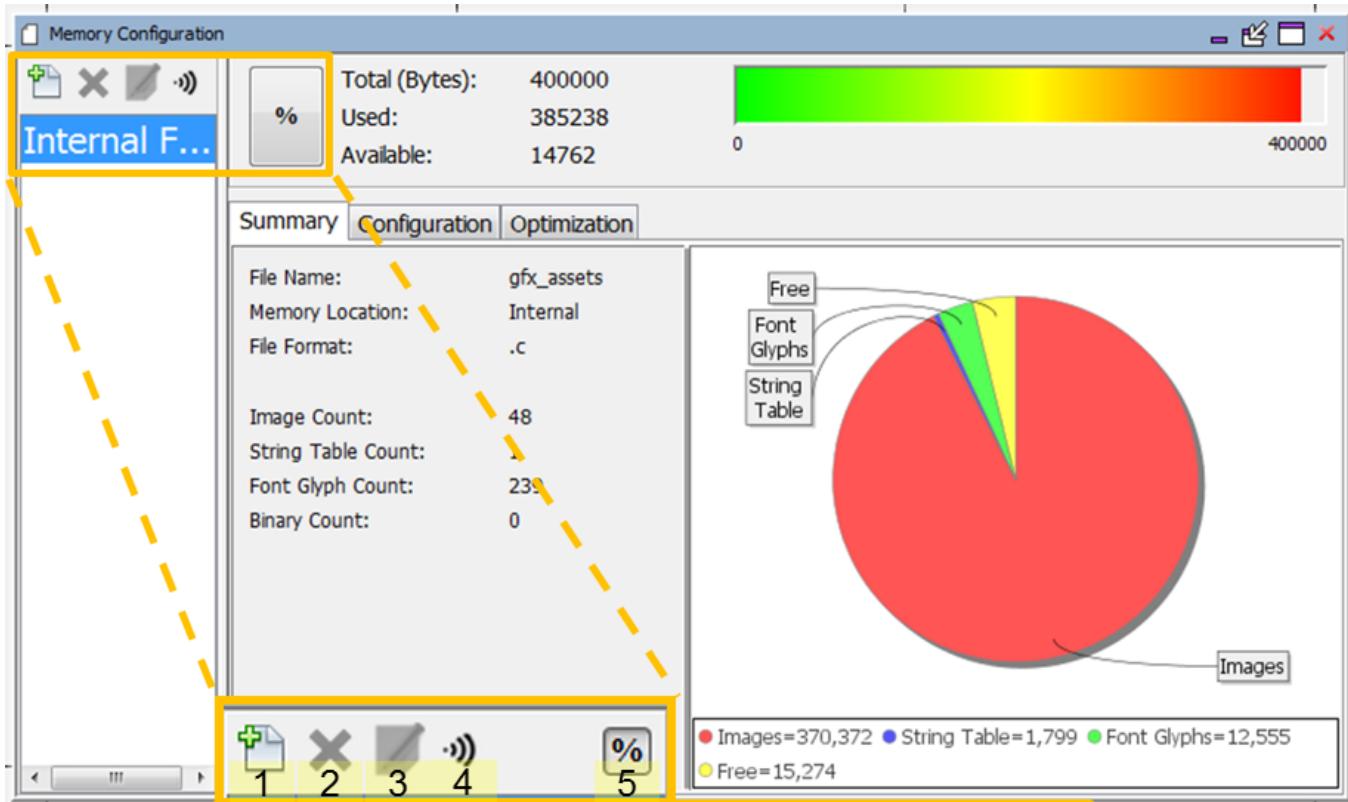
The Asset menu supports managing all graphical assets (memory, images, languages, fonts, strings, and binary data).

Memory Configuration

Provides information on configuring memory locations.

Description

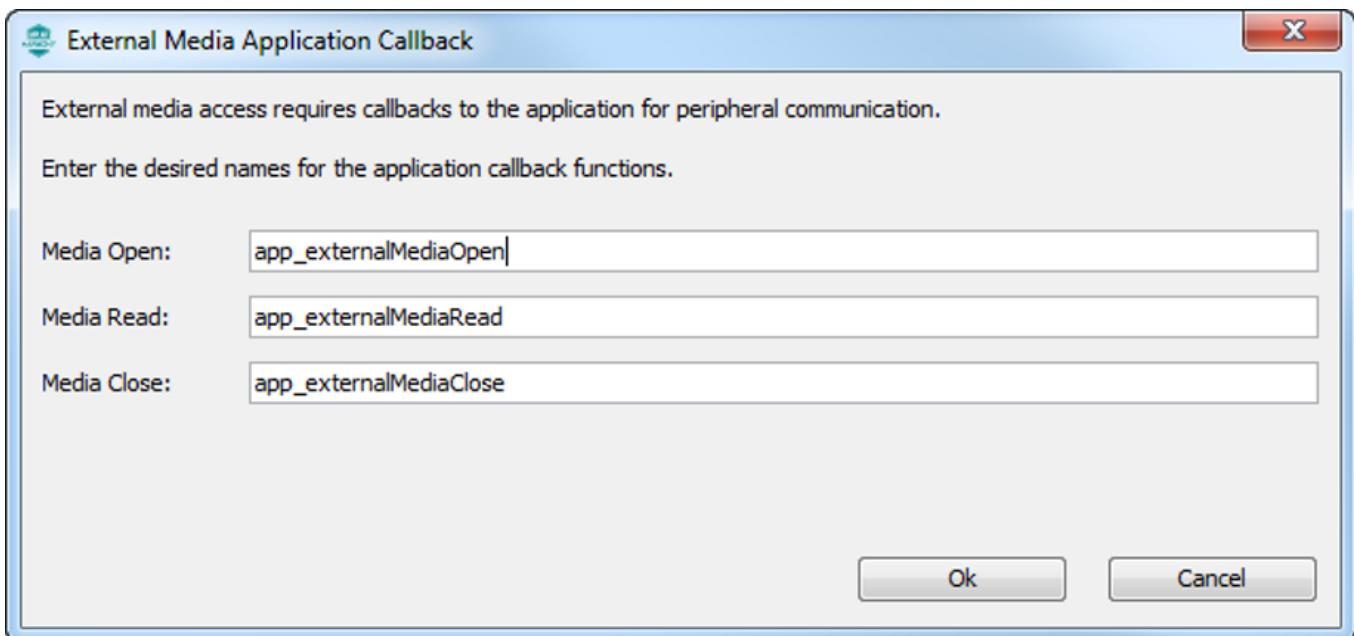
The Memory Locations window is launched from the Graphics Composer's Asset menu. Selecting Memory Locations this brings up a window with three sub-tabs (in this example, the [Aria Showcase](#) demonstration is referenced):



Window Toolbar

The window's tools icons support:

1. **Add New Memory Location** – This supports multiple external memory resources.
2. **Delete Selected Memory Location** – Removes a previously defined memory location.
3. **Rename Selected Memory Location** – Renames a previously defined memory location.
4. **Configure External Media Application Callback** – This allow definition of media callbacks, which must be provided in the project.



5. **Show Values as Percent** – Memory utilization on the bar graph can be in bytes or as a percent of the total internal flash memory assigned to support asset storage. (That memory allocation is set using the Configuration sub-tab.)

The APIs for the external media callback functions are as follows:

```
GFX_Result app_externalMediaOpen(GFXU_AssetHeader* asset);
GFX_Result app_externalMediaRead(GFXU_ExternalAssetReader* reader,
                                 GFXU_AssetHeader* asset,
                                 void* address,
                                 uint32_t readSize,
                                 uint8_t* destBuffer,
                                 GFXU_MediaReadRequestCallback_FnPtr cb);
void app_externalMediaClose(GFXU_AssetHeader* asset);
```

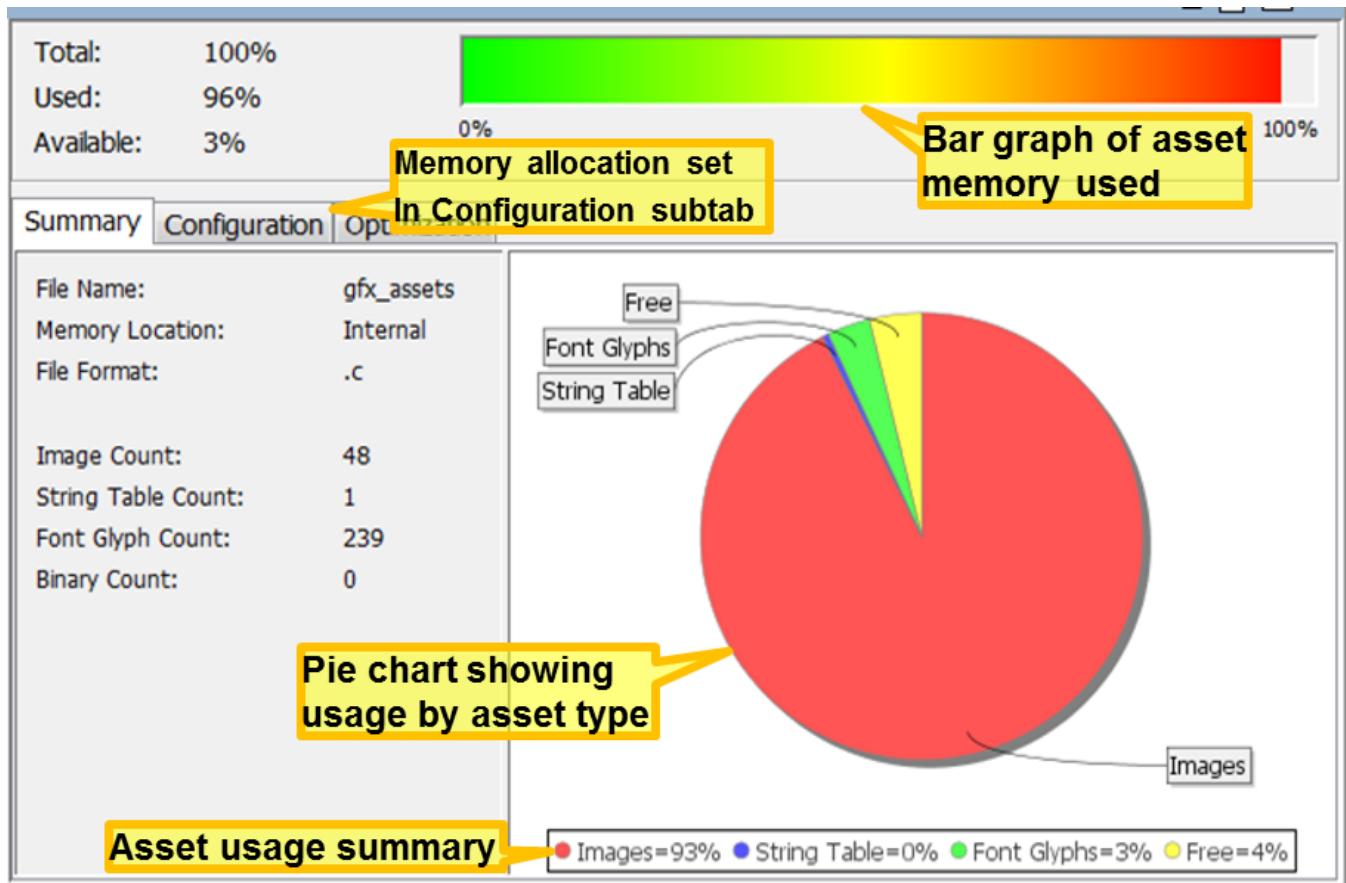
The graphics demonstration project, `aria_external_resources`, provides an example of how to write these callbacks. This demonstration supports three types of external memory: SPI External Memory, USB Binary, and USB with File System. Examples of these callbacks are found in the project's `app.c` file. The Aria demonstration projects `Aria External Resources` and `Aria Flash` provide more details on how to use external memory to store graphics assets.

Sub-tabs

There are three sub-tabs to this window.

Summary Sub-tab

This sub-tab summarizes program flash allocations for images, strings, and fonts.



The memory allocation shown for "Font Glyphs" measure the space that holds all the font glyphs used by the application, either by static strings or by glyph ranges defined in support of dynamic strings. Strings are defined by arrays of pointers to glyphs, so string memory usage measures the size of these arrays, not the actual font glyphs used. ("Glyph" is defined here.)

Note: The word "glyph" comes from the Greek for "carving", as seen in the word hieroglyph – Greek for "sacred writing". In modern usage, a glyph is an elemental or atomic symbol representing a readable character for purposes of communicating through writing.

Configuration Sub-tab

This sub-tab specifies the intended allocation of internal (program) flash memory to graphics assets (Total Size). (The default value is 1024 bytes.) It also names the graphics assets file name (here it will be `gfx_assets.c`). The allocation of flash is only used to scale the Total/Used/Available bar graph at the top of the display. Under sizing or oversizing this amount does not affect how the application is built.

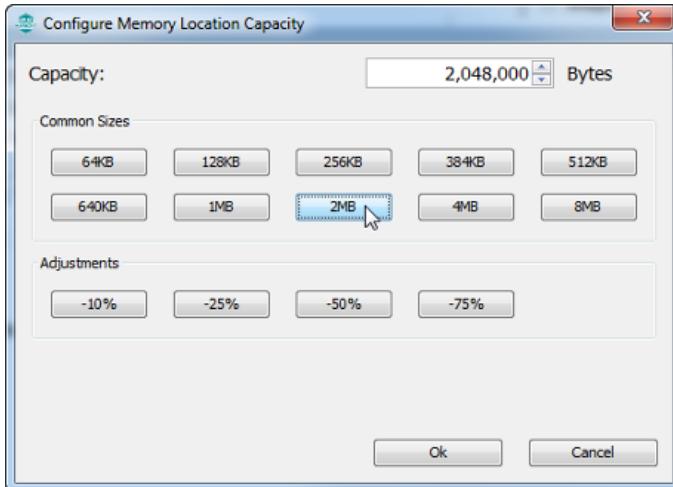
Capacity:	1,536,000 Bytes
<input type="button" value="Calculator"/>	
Output File Name:	gfx_assets

If the device has 1024 Kbytes (1048576 bytes) of flash, the user can assign 40% to asset storage and 60% to code. In that case the "Total Size" in the above sub-tab would be set to 419430 (= 40% of 1048576).

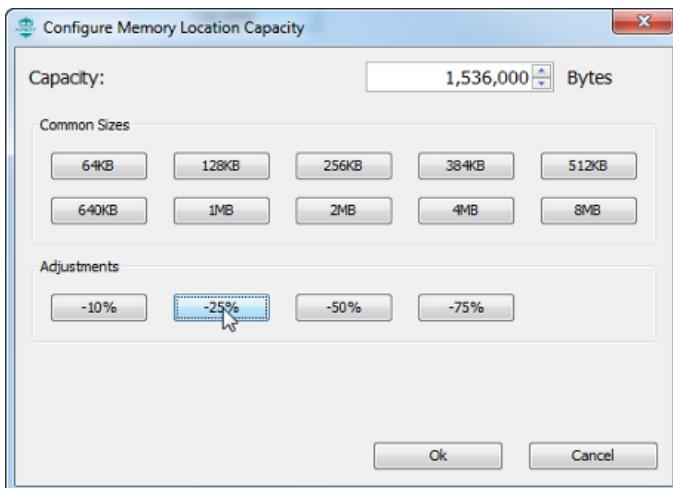
The Calculator button can assist in allocating internal flash. Click on it and then set the device flash capacity. Then the user can apply an adjustment to that value to assign that memory to asset storage.

Example:

If the device has 2 Mbytes of internal Flash, click **2MB**.



Then, to assign 75% of the 2 Mbytes to asset storage, click **-25%** to reduce the 2 MB by 25%, leaving 75%, and then click **OK** to finish. This will then assign 1,536,000 bytes to asset storage.

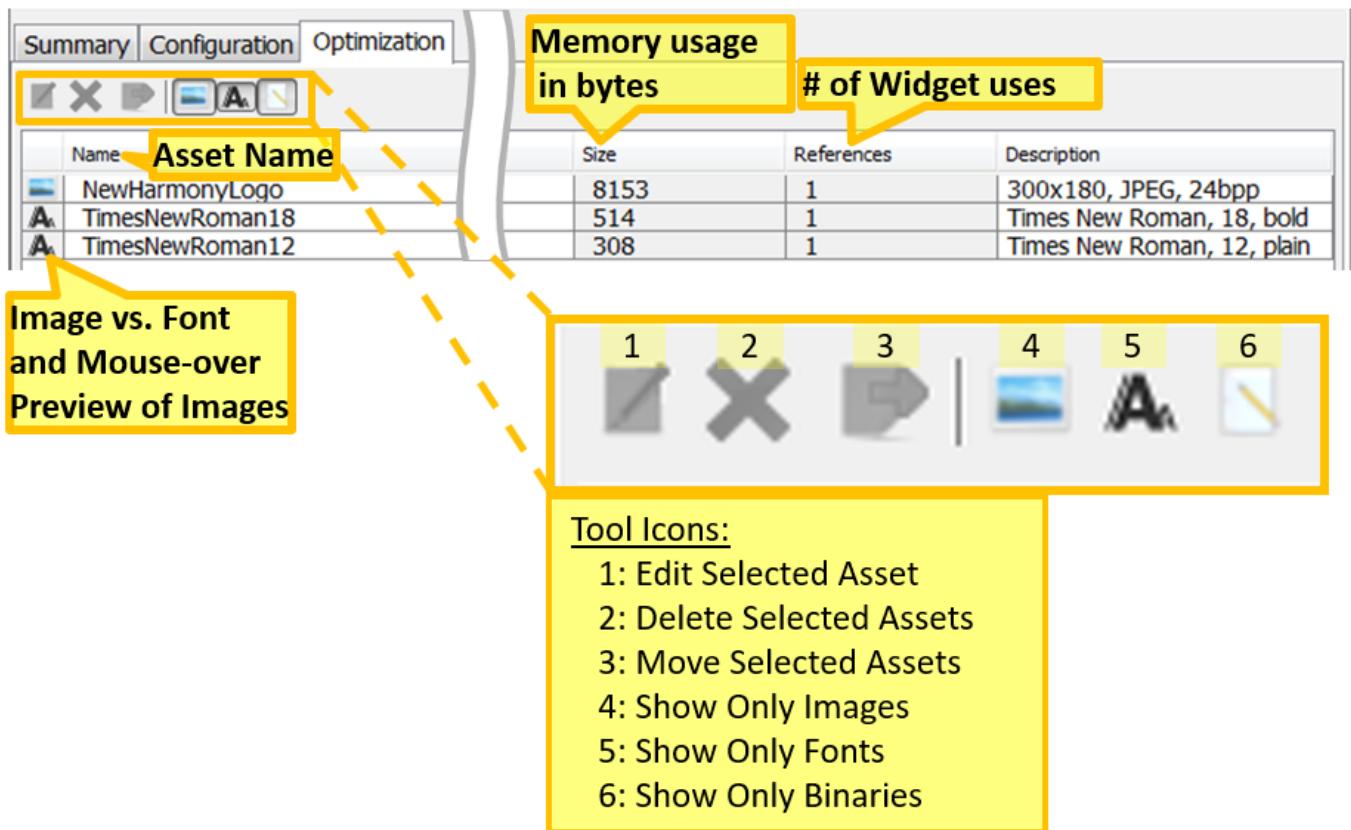


Internal (program) Flash is shared between the application's code and asset storage. If the application code and graphics assets (fonts, strings, images) won't fit into the available flash memory then the linker will be unable to build the application and an error will be generated in MPLAB X IDE.

The **Output File Name** must be compatible with the operating system hosting MPLAB X IDE. In most cases the default name (`gfx_asset.c`) will suffice, but this is provided for additional flexibility in building the application.

Optimization Sub-tab

The Optimization sub-tab for the Aria Quickstart demonstration is shown in the following figure.



The **Size** column shows the bytes allocated for storage in internal flash for the images, fonts, and binaries of the application.

The **References** column shows the number of known references for these assets by the application's widgets. A references count of zero suggests that the asset is not used by the application, but it could also mean that the asset is only used in real-time when it is dynamically assigned to a widget by the application. Clicking the title of a column (Name, Size, or References) sorts the lists of graphics assets by that column. Clicking the same column again reverses the sort order.

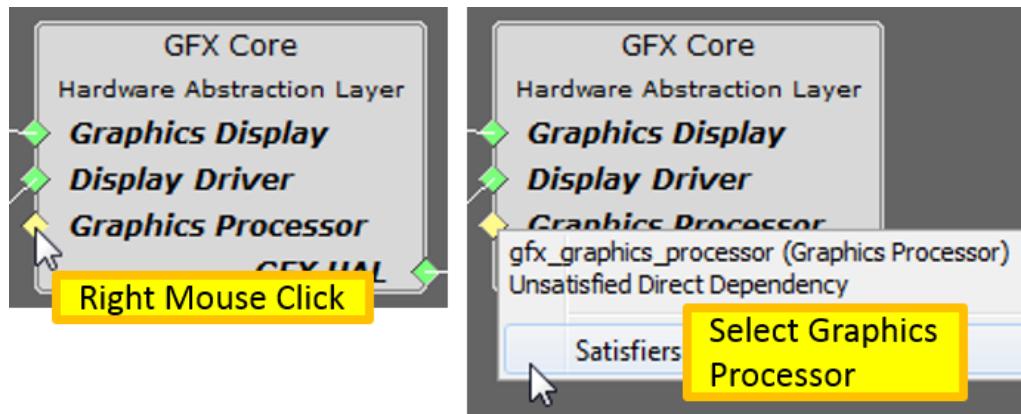
The window's tools icons support:

1. **Edit Selected Asset** – This brings up the edit dialog for the image, font, or binary chosen
2. **Delete Selected Assets** – Removes the selected assets
3. **Move Selected Assets** – Move assets from one location to another. This is useful for moving assets to/from internal memory from/to external memory.
4. **Show Only Images** – Show image assets toggle on/off
5. **Show Only Fonts** – Show font assets toggle on/off
6. **Show Only Binaries** – Show binary assets toggle on/off

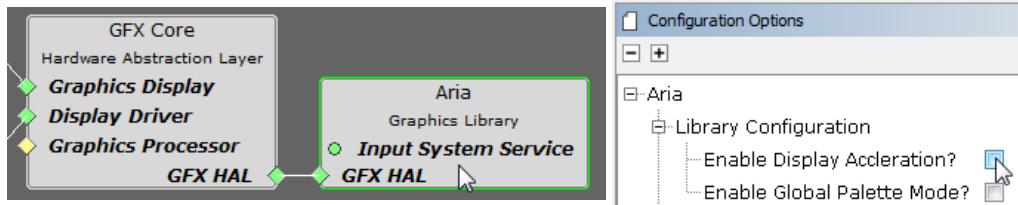
DDR Organizer

The DDR Organizer tool supports managing buffers, raw images, and other memory resources in the DDR memory of DA devices and only DDR-enabled DA devices.

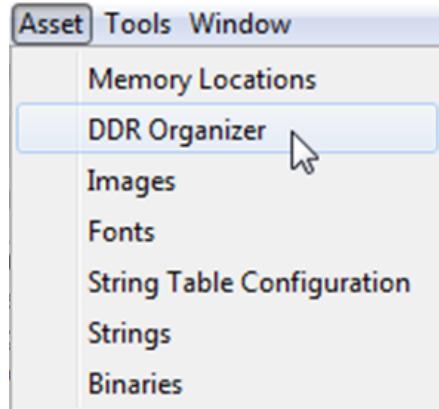
A Graphics Hardware Accelerator (Graphics Processor) must be installed in the MHC's Project Graph. For the GFX Core component, do a right mouse click on the Graphics Processor component and then select among the available Graphics Processors. Select the "Nano2D" processor from the Satisfiers list:



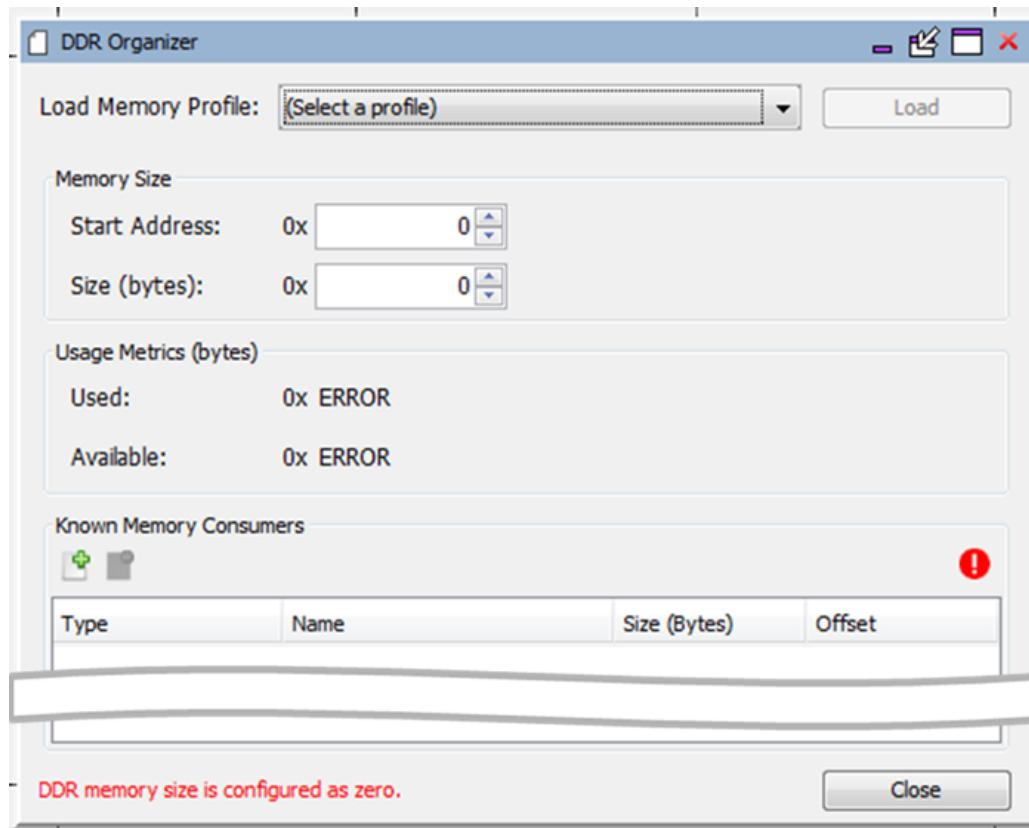
Next, click on the Aria Graphics Library component and in its Configuration Panel enable display acceleration:



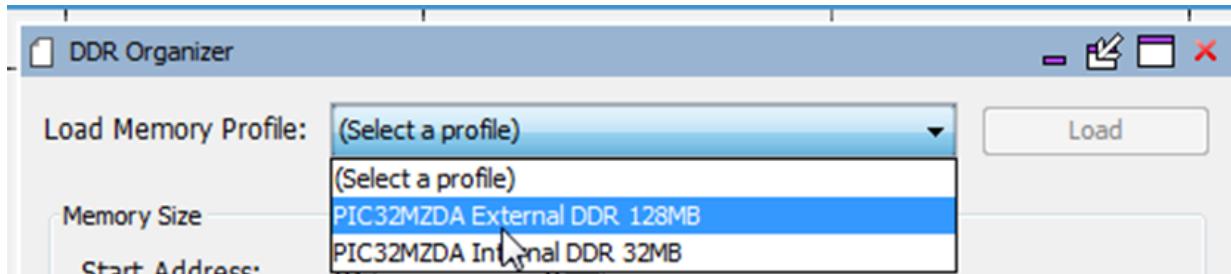
The DDR Organizer tool is launched from the Assets Management pull-down menu:



The following window will appear if the tool has not been used before for the active project target configuration:



Select the memory profile that corresponds to the target DDR-enabled DA device:



Then select the **Load** Button to load that memory configuration into the tool.

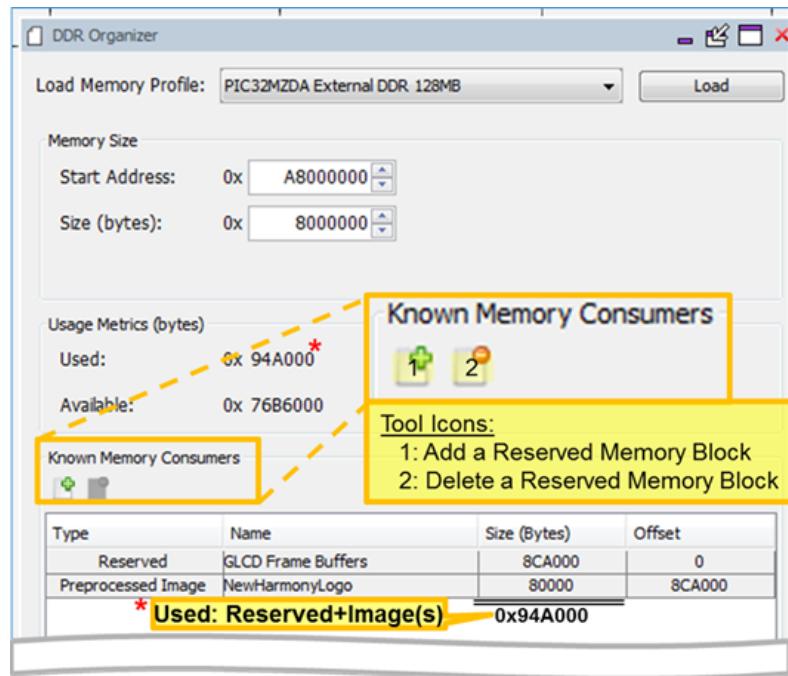
When Preprocessing is enabled for an image under the Image Assets tool:

Preprocessing Options:

- 1: Enable Image Preprocessing
- 2: Automatically Manage Memory Address
- 3: Preprocessing Output Mode
- 4: Enable Power of Two Image Padding
- 5: Expected Processed Image Size

Preprocessing	
1	Enabled
2	Managed
3	Output Mode RGBA_8888
4	Padding
5	Expected Size 524288

An entry for the image appears in the DDR Organizer window:



When the memory profile is loaded, the tool automatically reserves DDR memory for the GLCD Frame Buffers sufficient for three double-buffered layers, allocating 32 bits (4 bytes) for RGBA_8888 format for each pixel. This provides 384,000 pixels (800x480) per frame buffer.

The tool icons support adding non-image memory allocations to the DDR memory map. To add or remove the memory allocation belonging to an image, the Preprocessing enabled property for that image is enabled/disabled using the Image Asset tool.

Image Assets

Provides information on the Image Assets features.

Description

The Image Assets window is launched from the Graphics Composer's Asset menu.

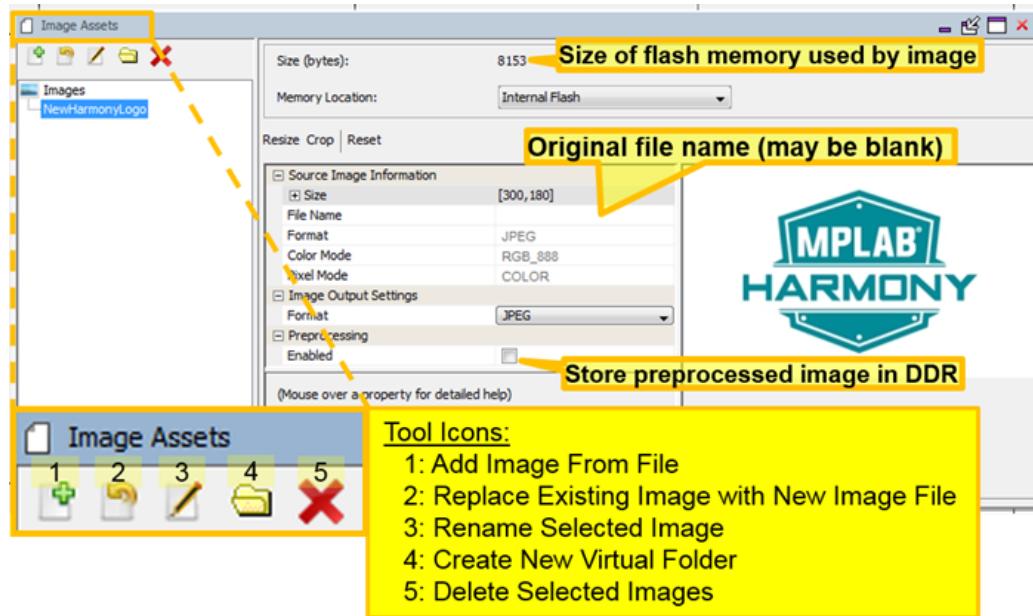
The Image Assets window supports import images, select different image formats/color modes for each image, select compression methods (for example, RLE) for each image, and displays the memory footprint of each. Images can be imported as a BMP, GIF, JPEG, and PNG (but not TIFF). Images can be stored as Raw (BMP, GIF), JPEG, and PNG.

Note: MHGC does not support image motion that can be found in GIF (.gif) files. GIF images are stored in the raw image format, meaning that there is no image header information stored with the image.

When an image is imported into MGHC, the Graphics Asset Converter (GAC) stores the input format and color mode along with any relevant header data. The image's pixel data is then promoted from its native format into a Java Image using 32 bits/pixel (8 bits for each color, RGB, and 8 bits for Alpha Blending). If the image contains Alpha Blending then this information is stored in the "A" of RGBA, otherwise the "A" is set to maximum opacity. When the application is built each image is stored in the image format and color mode selected. Images displayed in the Screen Designer are converted from Java Image format into the format/color mode selected so that the Screen Designer accurately represents what the application will show when running.

The images are decoded on the fly by the graphics library and rendered on the screen. This provides the designer with considerable flexibility to import using one format and store resources using another format, thus exploring and maximizing the best memory utilization for their application and hardware. This supports trading a smaller memory footprint at the cost of additional processing (for static or drawn-once) or reducing processing at the cost of a larger memory footprint (dynamic or drawn many times).

The following figure shows the Image Assets window for the [Aria Quickstart](#) demonstration.

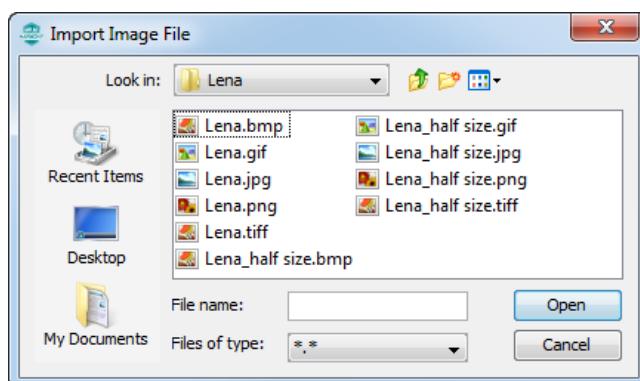


Window Toolbar

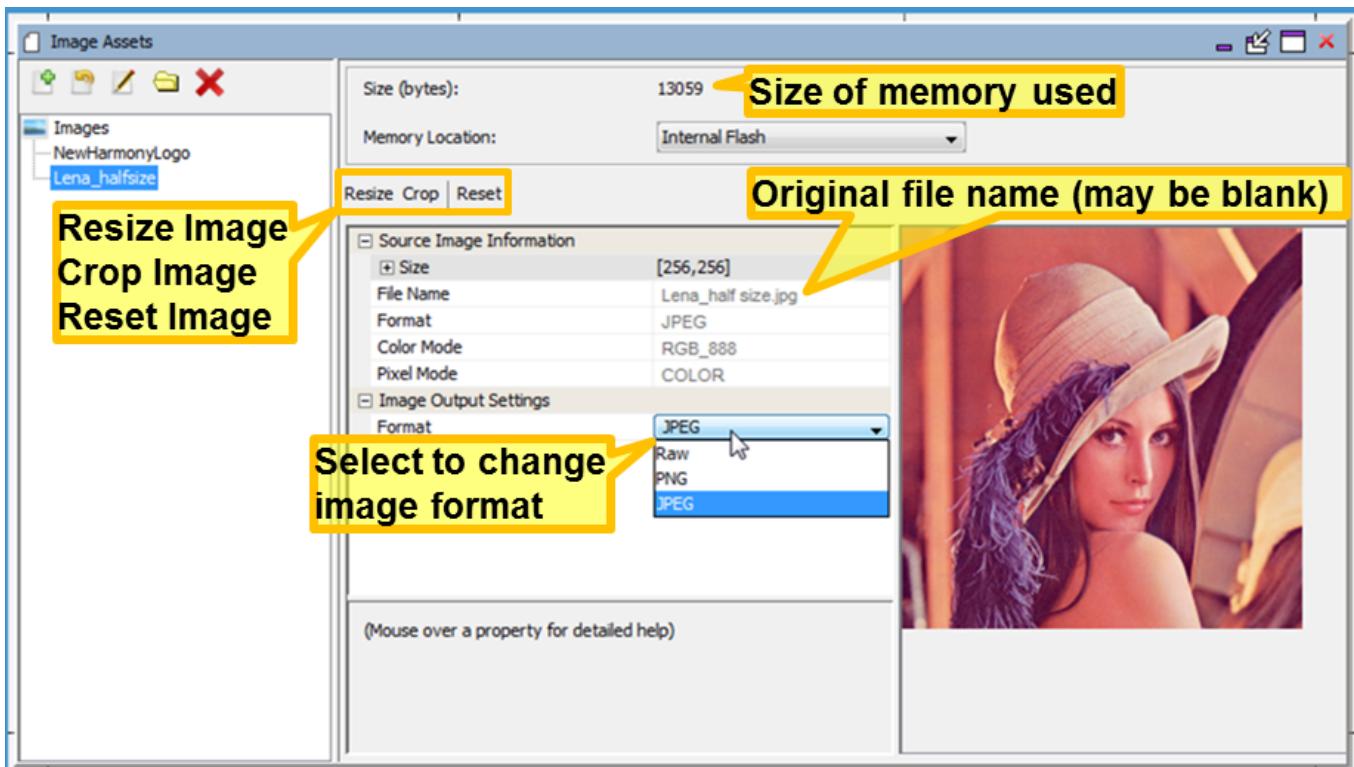
There are five icons on the toolbar below the Images tab:

1. **Add Image Asset** – Brings up “Import Image File” dialog window to select image file to add to the graphics application.
2. **Replace Existing Image with New Image File** – Brings up the same “Import Image File” dialog but instead of creating a new image, the file’s content replaces the currently selected image.
3. **Rename Selected Image** – Renames the selected image.
4. **Create New Virtual Folder** – Creates a new virtual folder, organizes images in a hierarchy.
5. **Delete Selected Images** – removes the selected images from the application.

Selecting the Add Image Asset or Replace Existing Image icon opens the Import Image File dialog that can be used to select and import an image.



After selecting the file and clicking **Open**, the Image Assets window opens.



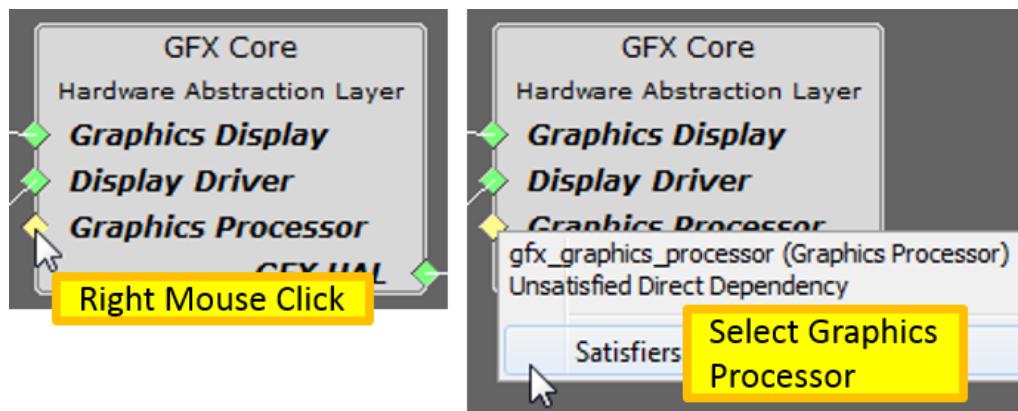
The size of the memory used for this image based on its color mode, format, compression, and global palette usage is shown by Size (bytes). See Image Format Options below for more details.

The File Name of the original source file is also shown, but may be blank if the image was imported under MPLAB Harmony v2.03b or earlier. The format and color mode of the stored image can be changed to reduce the image's memory footprint. (If using an LCC controller, you can also turn on the Global Palette, replacing each pixel in the image with just an 8 bit LUT index.)

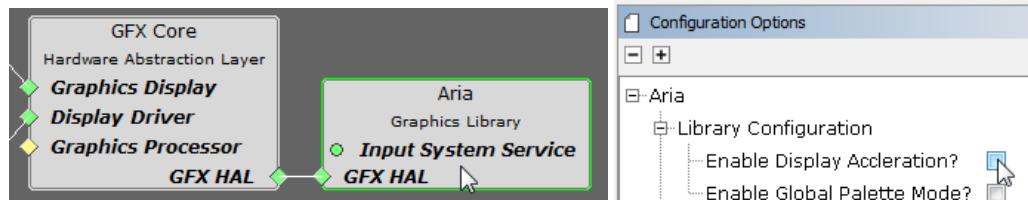
The three internal image formats are:

- **Raw** – binary bit map with no associated header information. GIF and BMP images are imported into this format.
- **PNG** – lossless image format with compression, 24 bits/pixel (RGB_888) or 32bits/pixel (RGBA_8888). A good choice for line drawings, text, and icons.
- **JPEG (JPG)** – loss compressed format, uses much less storage than the equivalent bit map (raw). Good for photos and realistic images.

New to Harmony 2.06 is the option to preprocess an image into raw pixels at boot-up, which will greatly improve image draw/redraw times though the use of the high performance 2-D graphics processing unit (GPU) that is available on DDR-enabled DA devices. Be sure that this feature is enabled in MPLAB Harmony Configurator. Install a graphics hardware accelerator in MHC's Project Graph. For the GFX Core component, do a right mouse click on the Graphics Processor component and then select among the available Graphics Processors. Select the "Nano2D" processor from the Satisfiers list:



Next, click on the Aria Graphics Library component and in its Configuration Panel enable display acceleration:



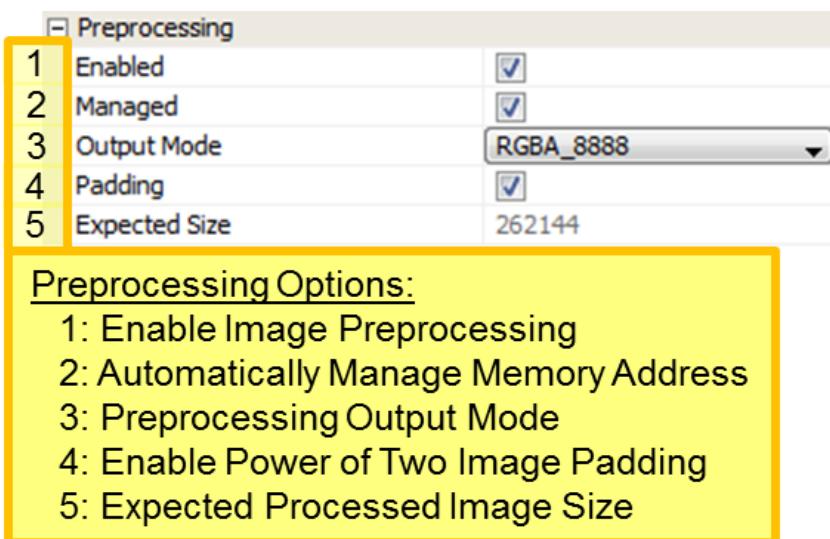
Note: Do not enable image preprocessing except on DDR-enabled DA devices with the 2D GPU graphics processor enabled (which uses the Nano 2D Driver).

To do so will produce an application that builds but does not run.

With Preprocessing of the image enabled, additional options become available:

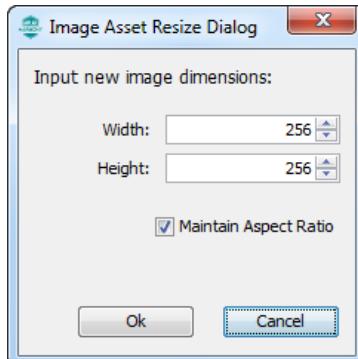
- DDR Memory allocation for the image is automatically handled when the **Managed** option is selected
- The **Output Mode** should be selected to match the GLCD's color mode, typically **RGBA_8888**
- The **Padding** option expands the image size to the nearest power of two. For example, a 480x212 image would be increased to 512x256 pixels.
- The expected size of the preprocessed image in DDR memory is shown in the **Expected Size** entry

For more information on how images are stored within DDR memory, see the section on the Asset Management DDR Memory tool above.

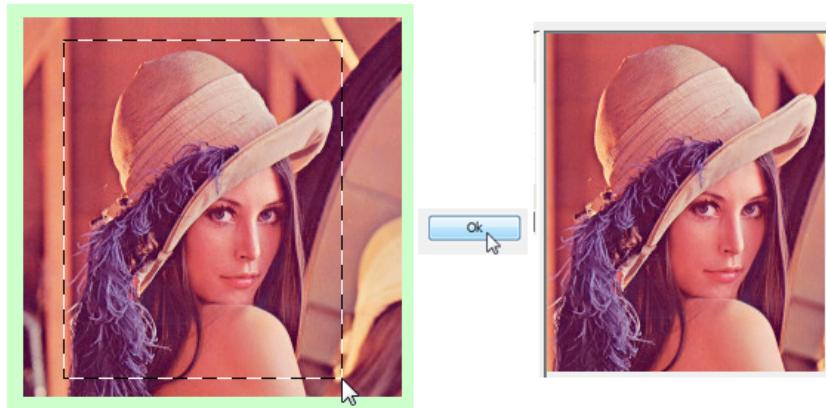


The Image Assets window supports resizing, cropping, or resetting an image:

- **Resize** – Brings up a dialog window to change the pixel dimensions of the image. The image is interpolated from the original pixel array into the new pixel array.



- **Crop** – Places a cropping rectangle on the image. Click and drag a rectangle across the image to select the new image. Then click **Ok** to crop the image.



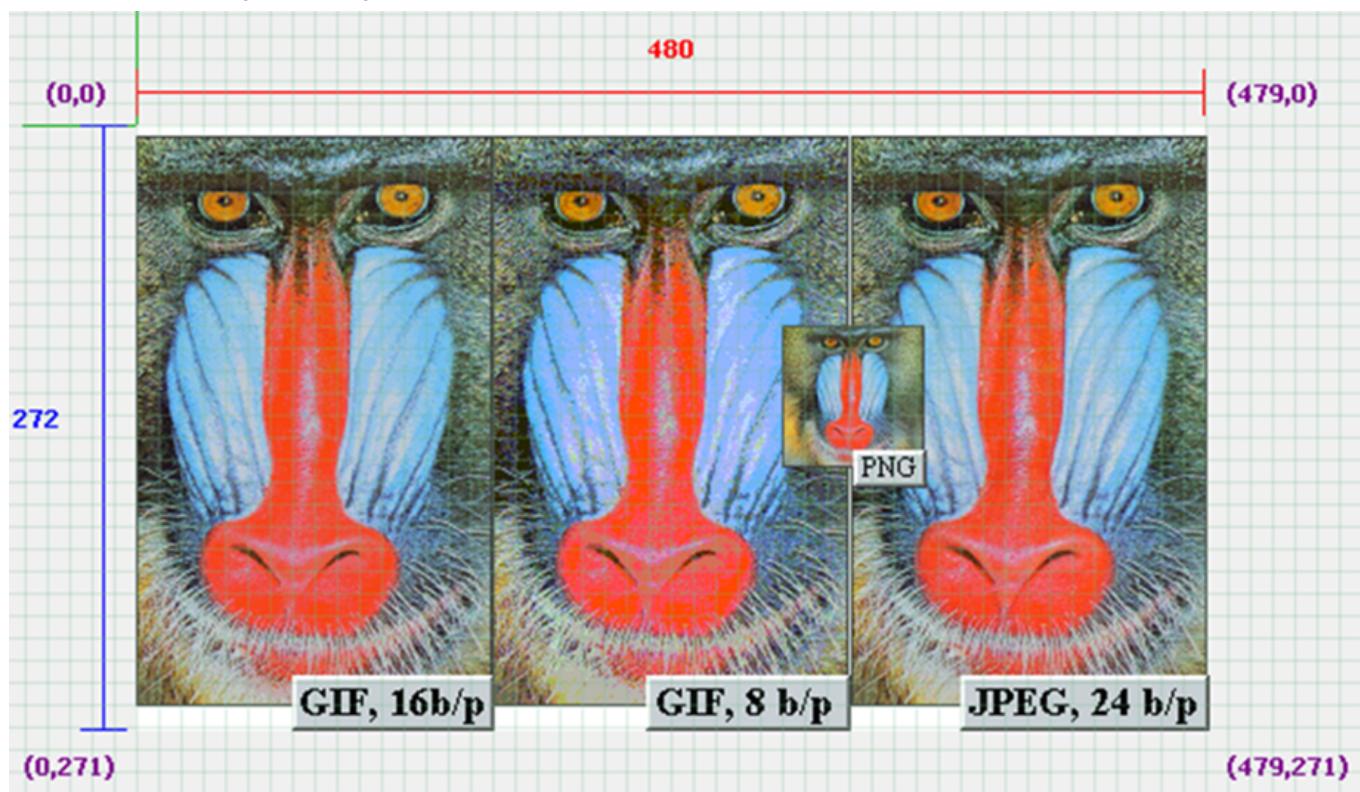
- **Reset** – Allows undoing of a resize or crop. The original image is always stored in the project, so a Reset is always available to return the image to its original state.

Original images are retained by MHGC by the superset Java Image format. So an image crop will change how the image is stored in the application but not how it is stored in MHGC. Reset will always restore the image back to the original pixels. (Reset is not an "undo".)

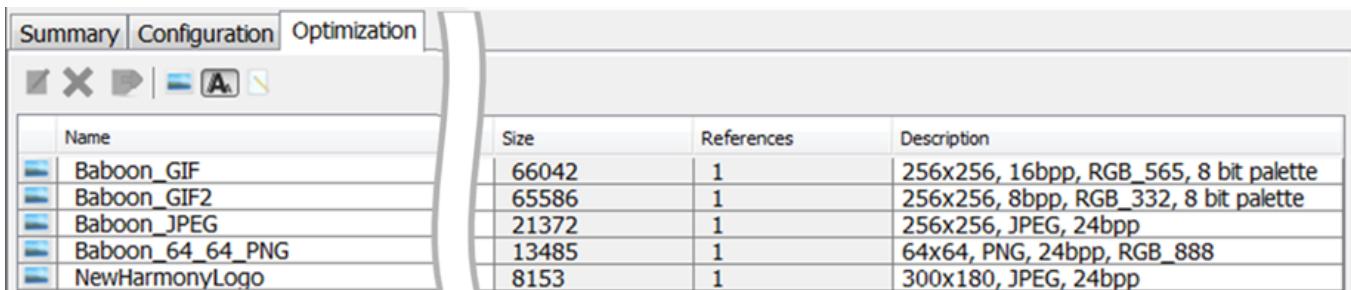
Example Images

Example images are available from many sites on the internet. One of the best sites is found at the USC-SIPI Image Database (<http://sipi.usc.edu/database/>). There are many canonical test images, such as Lena, The Mandrill (Baboon), and other favorites, all in the TIFF format. The TIFF format is not supported by the Graphics Composer, but it can be easily convert from TIFF to BMP, GIF, JPEG, or PNG using the export feature found in the GNU Image Manipulation Program (GIMP), which is available for free download at: <https://www.gimp.org>. GIMP also allows the user to change the pixel size of these images, usually 512x512, to something that will fit on the MEB II display (either 256x256 or smaller).

The following figure shows the Graphics Composer Screen Designer for the pic32mz_da_sk_meb2 configuration of the [Aria Quick Start](#) project after adding three images.



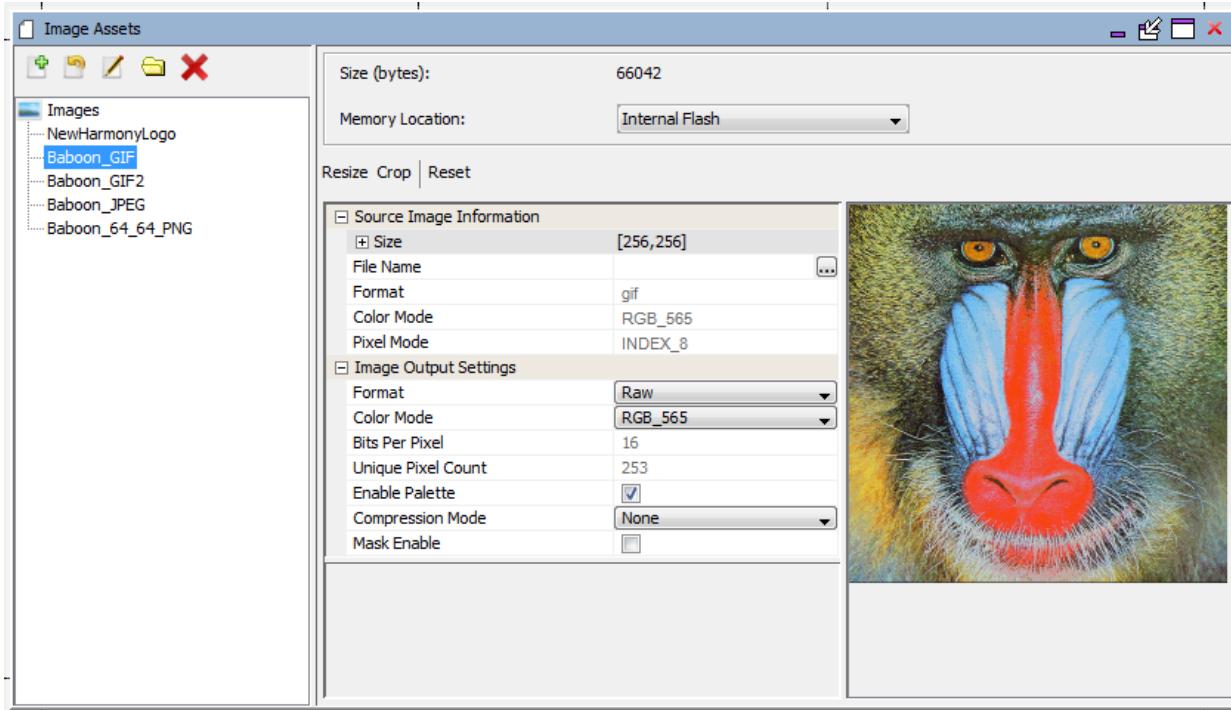
The following figure shows the Optimization Tab after adding these images.



The screenshot shows the 'Asset Management' tab of the Graphics Composer interface. It displays a table of assets with columns for Name, Size, References, and Description. The assets listed are Baboon_GIF, Baboon_GIF2, Baboon_JPEG, Baboon_64_64.PNG, and NewHarmonyLogo. The Baboon_GIF asset is selected.

Name	Size	References	Description
Baboon_GIF	66042	1	256x256, 16bpp, RGB_565, 8 bit palette
Baboon_GIF2	65586	1	256x256, 8bpp, RGB_332, 8 bit palette
Baboon_JPEG	21372	1	256x256, JPEG, 24bpp
Baboon_64_64.PNG	13485	1	64x64, PNG, 24bpp, RGB_888
NewHarmonyLogo	8153	1	300x180, JPEG, 24bpp

Selecting the Baboon_GIF image and the Edit Selected Asset icon () opens an Image Assets window, as shown in the following figure.

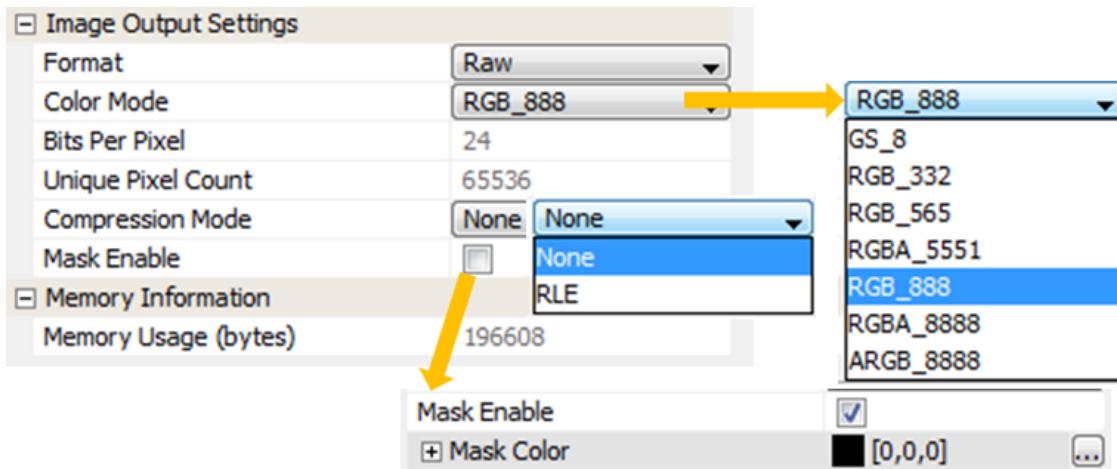


Because this image had only 253 unique pixel colors (Unique Pixel Count = 253) the Enable Palette option was automatically enabled. This feature, which works on an image by image basis, is separate from enabling a Global Palette. The image is stored using 8 bits of indexing into an image-specific lookup table (LUT). If the image has more than 256 unique colors then the Enable Palette option is not available and is not shown.

Image Format Options

Raw Format Images

Raw format images have the following options:



Regardless of the Color Mode of the imported the image, the stored image can be stored in a different color mode. For example, a JPEG image could be in 24 bits/pixel RGB format but stored in the application using RGB_565 or even RGB_332 to save space. The Project Color Mode (set through the *File > Settings* menu) is different from the Color Mode of images. This is determined by the capabilities of the projects graphics controller. The graphics library converts images from the stored color mode to the project's color mode before output.

If the image has 256 or less unique pixel colors an option to Enable Palette is set by default. If the image has more than 256 unique colors this option is not displayed. This replaces the palette pixels with 8-bit indices into the image's palette look up table (LUT). NOTE: Enabling the Global Palette disables this for all images and all image pixels are replaced by 8-bit indices into the global palette LUT.

The Compression Mode for a raw format image is either None (no compression) or RLE for run-length encoding.

Image masking is a form of cheap blending. For example, given the following image, you may want to show the image without having to match the lime green background. With image masking the user can specify that the lime green color as the "mask color", causing it to be ignored when drawing this image. The rasterizer will simply match a pixel to be drawn with the mask. If they match, the pixel is not rendered.



PNG Format Images

For PNG format images, the user can change the image format and the image color mode:



JPEG Format Images

For JPEG format images, the user can change from JPEG format to Raw or PNG:



Once changed from JPEG into another format, the new format will have other options.

Managing Complex Designs

The Image assets tool lists the images in the order of their creation. In a future version of MPLAB Harmony this will be sortable by image name. For now, it is recommended that the user use the Memory Locations asset tool, and use the Optimization sub-tab instead to manage a complex set of images. The Optimization sub-tab allows the user to sort graphics assets (fonts, images, binaries) by Name, Size, and number of widget References. This makes it much easier to find and edit an image by its name rather than order of creation.

Font Assets

Provides information on the Font Assets features.

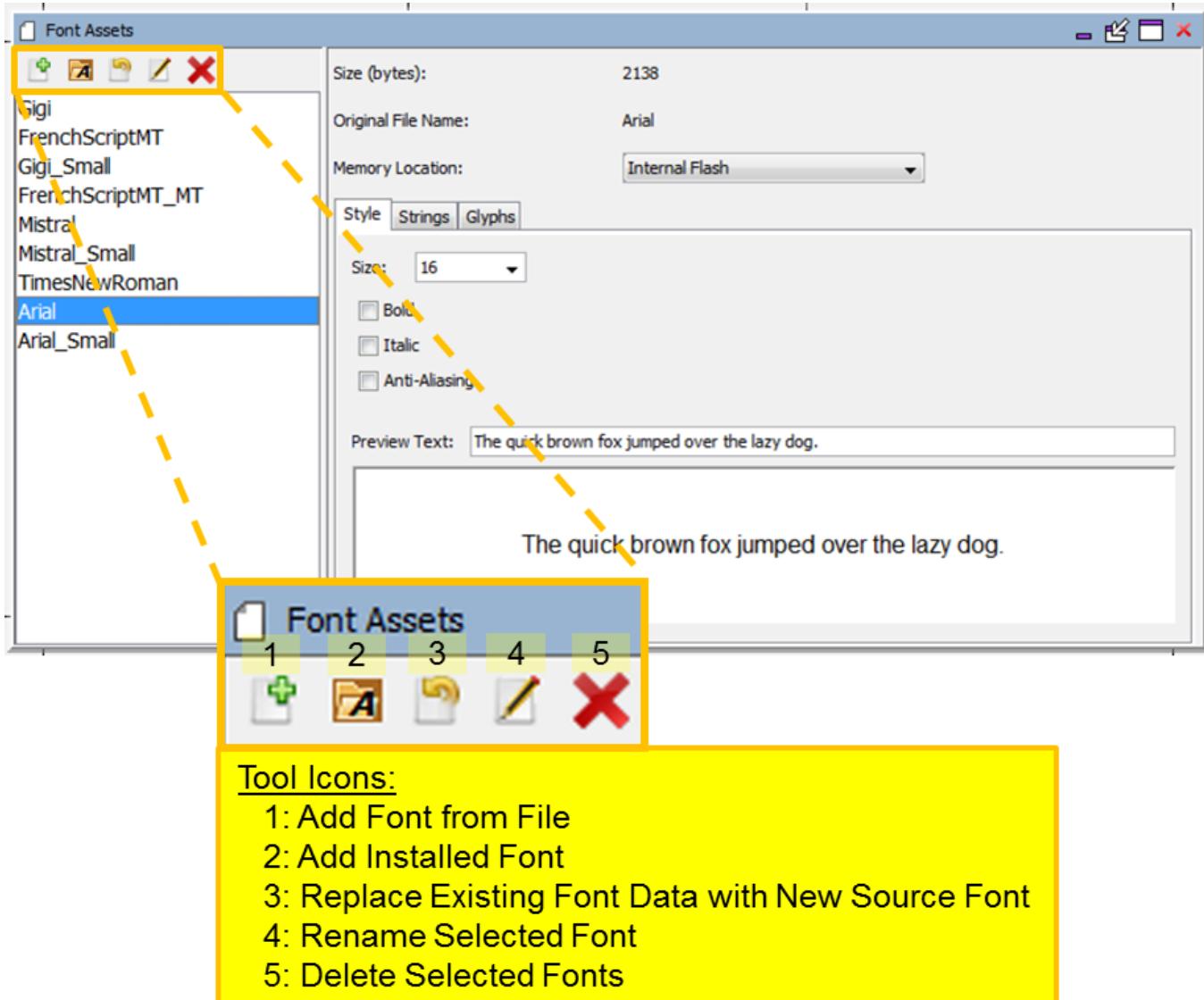
Description

The Font Assets window is launched from the Graphics Composer's Asset menu.

Note: There are three dimensions to text support: Languages, Fonts, and Strings. Language "ID" strings are identified when an application supports more than one language. (In the case of single language support, the language default is provided.) Fonts are imported and organized using the Font Assets window. Strings are defined by a string name, and this name is used by widgets to reference the string. For each string and each language supported the glyphs are defined to spell out the string's text and the font is chosen for that text.

- Languages are managed within the [String Table Configuration](#) window
- Fonts are managed within the Font Assets window (this topic)
- Strings are managed within the [String Assets](#) window

The following figure shows the Font Assets window from the Aria Coffee Maker demonstration.



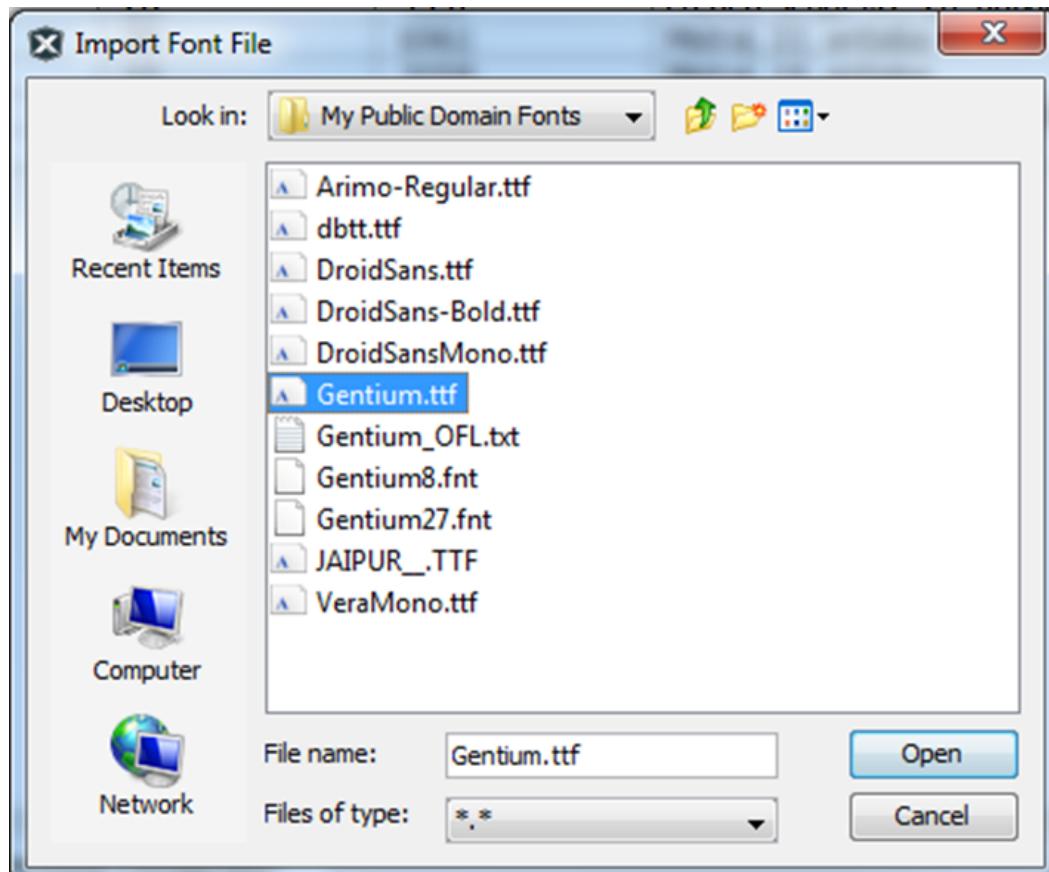
The Size (bytes): for a Font asset shows how much memory is needed to store all the glyphs used by the application from this font. For static strings MHGC determines which glyphs are used by the application's pre-defined strings and builds these glyphs into the application. For dynamic strings (i.e. strings created during run time) ranges of glyphs are selected by the designer and these ranges are also included in the application by MHGC. The memory needed to store all these glyphs is shown by Size

(bytes): .

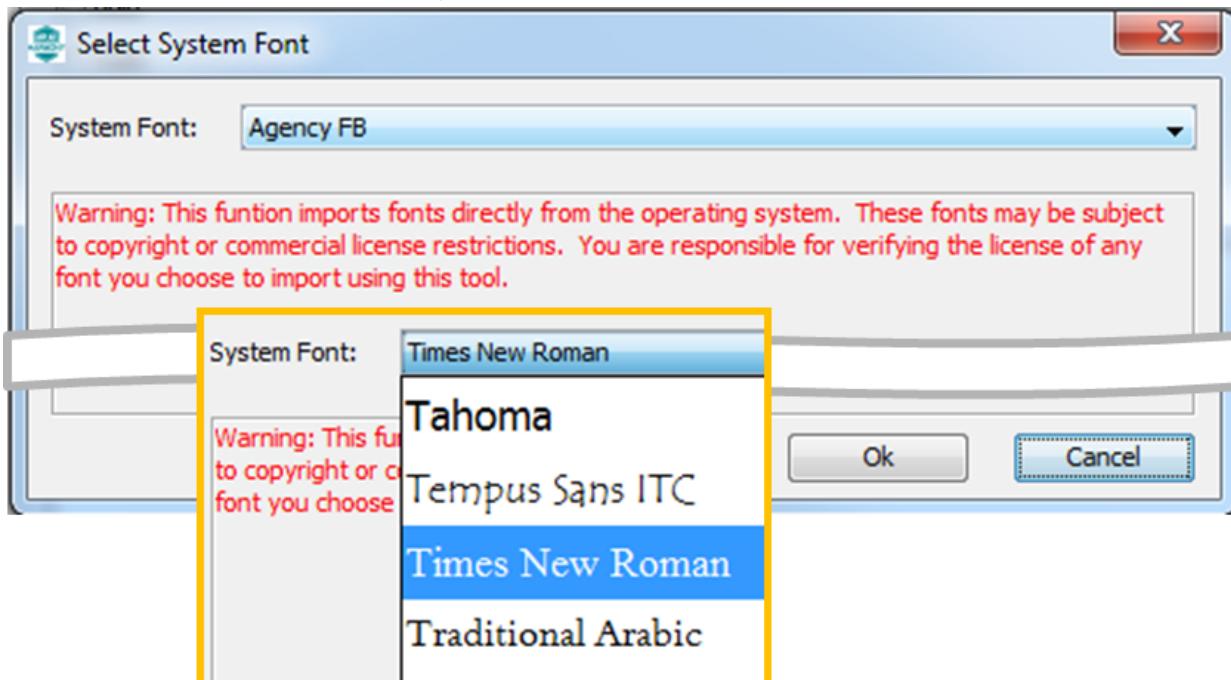
Window Toolbar

There are five icons on the toolbar below the Images tab:

1. **Add Font From File** – Adds a font asset from a file.



2. **Add Installed Font** – Add a font installed on your computer.

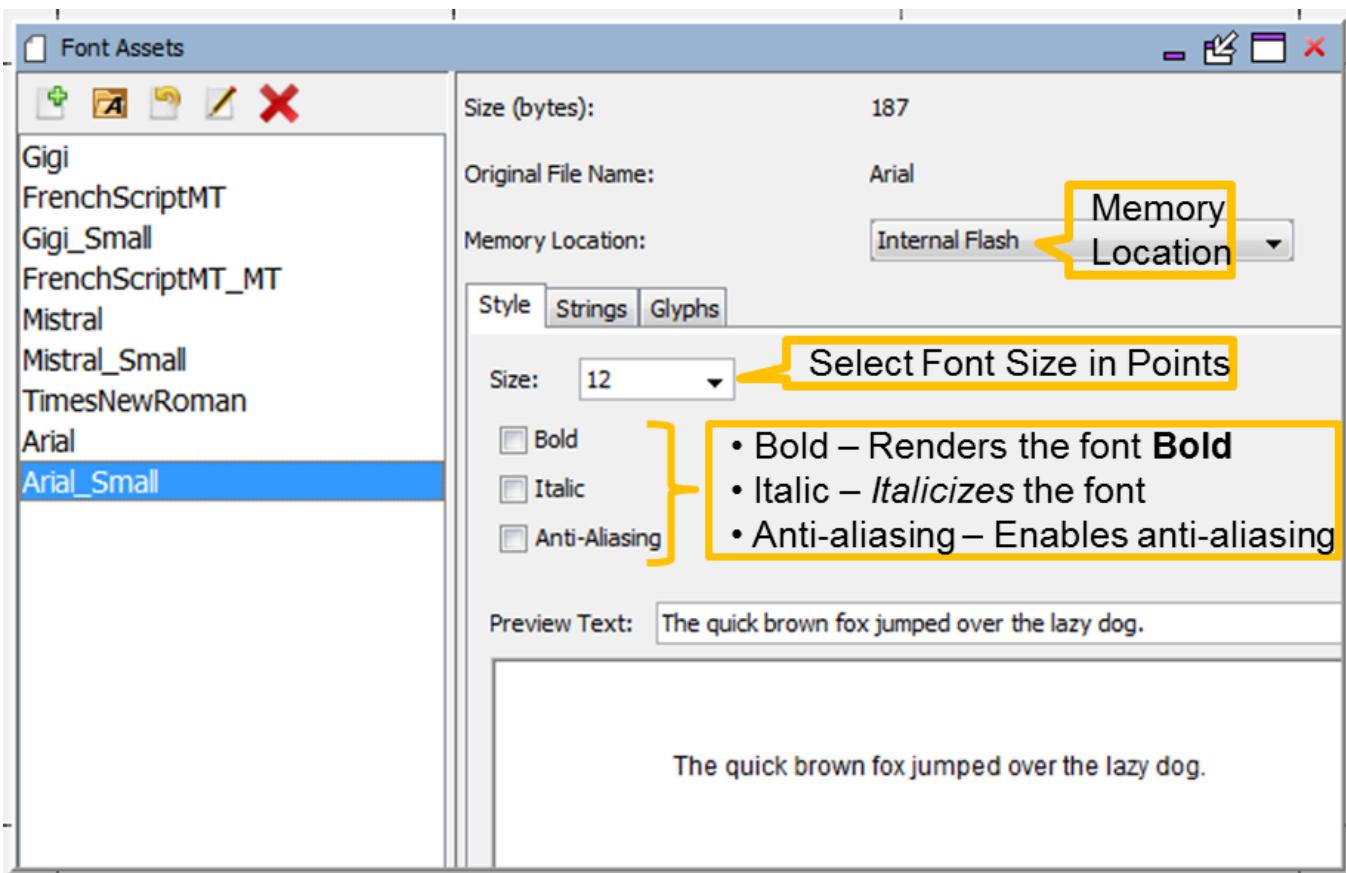


3. **Replace Existing Font Data with New Source Font** – Both Add Font From File and Add Installed Font create a new font asset. This icon allows the user to update an existing font asset, importing from a file or using a font installed on the user's computer.
4. **Rename Selected Font** – Renames an existing font asset. In the example above, the Arial font was installed twice, first as a 16 point font and second as a 12 point font. If added to the fonts assets in this order, the 12 point font will have the name Arial_1. This font asset was renamed to Arial_Small using this tool.
5. **Delete Selected Fonts** – Removes selected font assets from the application.

Sub-tabs

There are three sub-tabs to this window.

Style Sub-tab



The Size (bytes): shown represents the memory needed to store all the font's glyphs. The application only stores the glyphs that are used by static (build-time) strings and by predefined glyph ranges to support dynamic (run-time) strings.

The choices for Memory Location must be defined before the font can be assigned. Go to the Memory Configuration window to add a new location before using it in this sub-tab.

Each font asset consists of a font, size, and some combination of the { Bold, Italic, Anti-Aliasing } options, including selecting none of these options. If bold is needed for one set of strings and italic for another, then the user will need two font assets, one with Bold checked and a second with Italic checked. The same applies for font sizes. Each font size requires its own font asset. Thus if the user needs two sizes of Arial, with plain, bold, and italic for each size, there will have to be 6 separate assets (6 = 2 Sizes x 3).

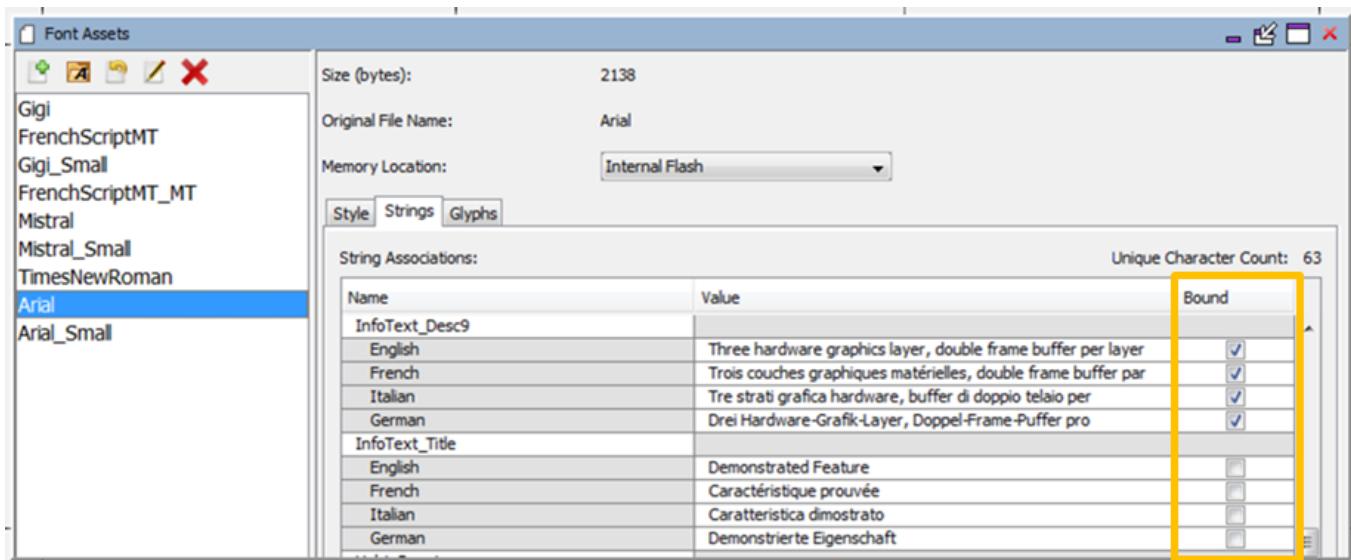
Glyphs are normally (Anti-Aliasing off) stored as a pixel bit array, with each pixel represented by only one bit. Turning on Anti-Aliasing replaces each pixel bit with an 8-bit gray scale, thereby increasing font storage by a factor of 8!

What if a font is chosen that does not support the character types of the text used for a particular language in the application? How can the user test and debug this? There are basically two ways:

- Use an external font viewer to examine if the needed glyphs exist
- Configure, build, and run the application and verify the strings are correctly rendered

If the glyphs are not available they will be rendered as rectangles (□).

Strings Sub-tab



The Bound check box accomplishes the same thing as assigning a font to a text string in the Strings Assets window (Window:Strings menu). Assigning a string to a font means that the font will generate glyphs for that string. This is just another way to accomplish the binding of the string text to font.

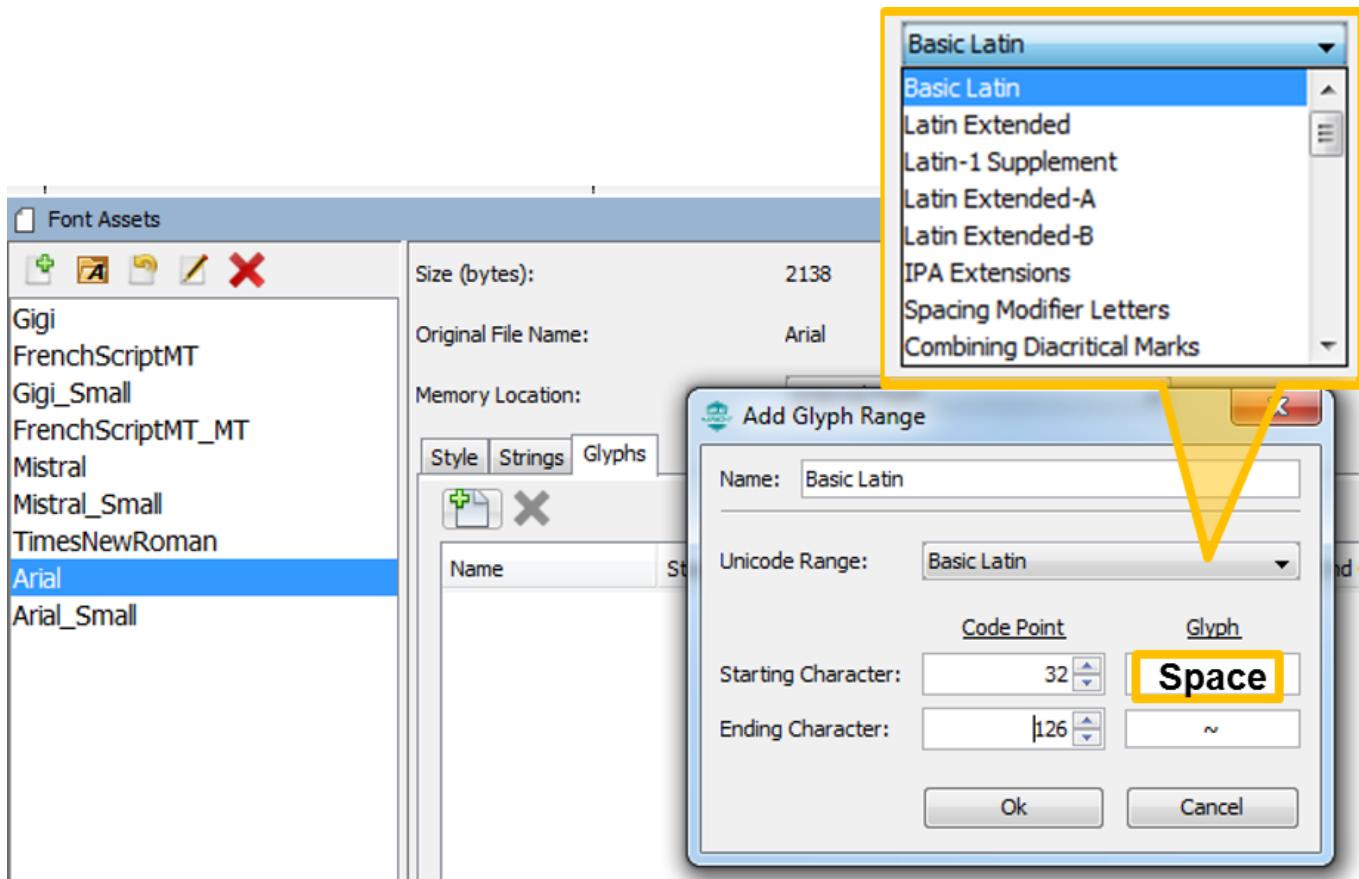
This sub-tab is also useful in a complicated graphics design to see how many strings use a particular font. Lightly-used or unused fonts can be eliminated to free up internal Flash memory.

Glyphs Sub-tab

 **Note:** The word “glyph” comes from the Greek for “carving”, as seen in the word hieroglyph – Greek for “sacred writing”. In modern usage a glyph is an elemental or atomic symbol representing a readable character for purposes of communicating via writing.

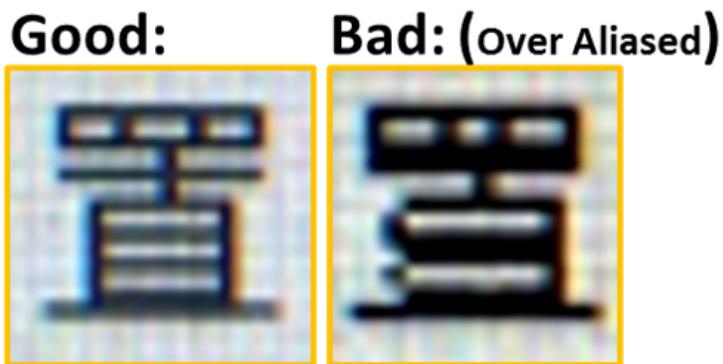
The Glyph sub-tab is only used when your application supports dynamic strings. For static (build-time) strings MHGC automatically determines which font glyphs are used based on the characters present in all the strings used by the application’s graphics widgets. Only these glyphs are included as part of the application’s font assets. With dynamic (i.e. run-time) strings this is not possible. This sub-tab allows you to specify which range of glyphs will be used by run-time strings. Once glyph ranges are defined, these glyphs are added to the font glyphs used by static strings.

The Create New Custom Import Range icon () allows the user to input a new glyph range for the font. Selecting this icon opens the Font Assets window.



NOTE:

Installing fonts already installed in your OS may produce over-aliased glyphs. It is better to use the installation source files rather than the installed files. This is especially true of fonts using non-Roman alphabets, such as Chinese, Japanese, or Korean. Here is an example:



String Table Configuration

Provides information on the String Assets features.

Description

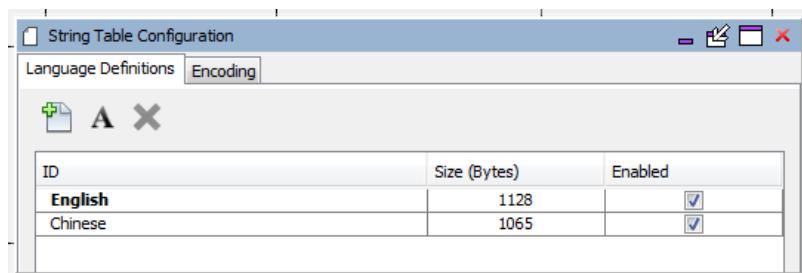
The String Table Configuration window is launched from the Graphics Composer's Asset menu.



Note: There are three dimensions to text support: Languages, Fonts, and Strings. Language "ID" strings are identified when an application supports more than one language. (In the case of single language support, the language default is provided.) Fonts are imported and organized using the Font Assets window. Strings are defined by a string name, and this name is used by widgets to reference the string. For each string and each language supported the glyphs are defined to spell out the string's text and the font is chosen for that text.

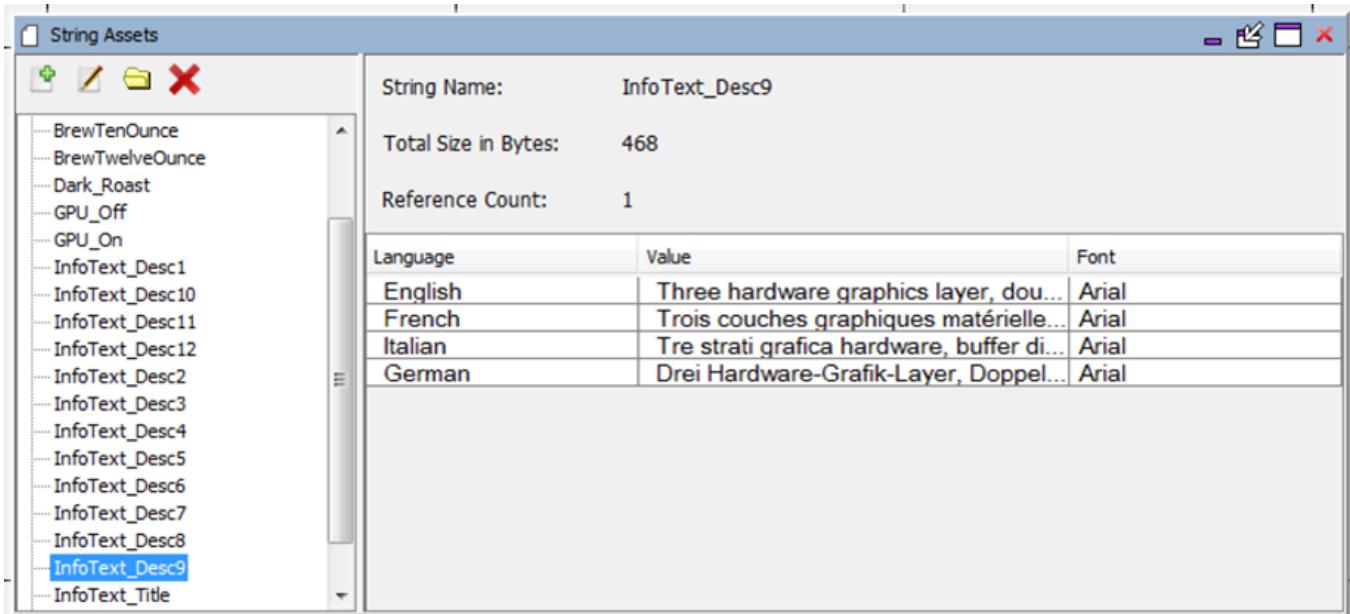
- Languages are managed within the String Table Configuration window (this topic)
- Fonts are managed within the [Font Assets](#) window
- Strings are managed within the [String Assets](#) window

Within this window, the Languages supported by the application are defined and the encoding for all application glyphs selected.



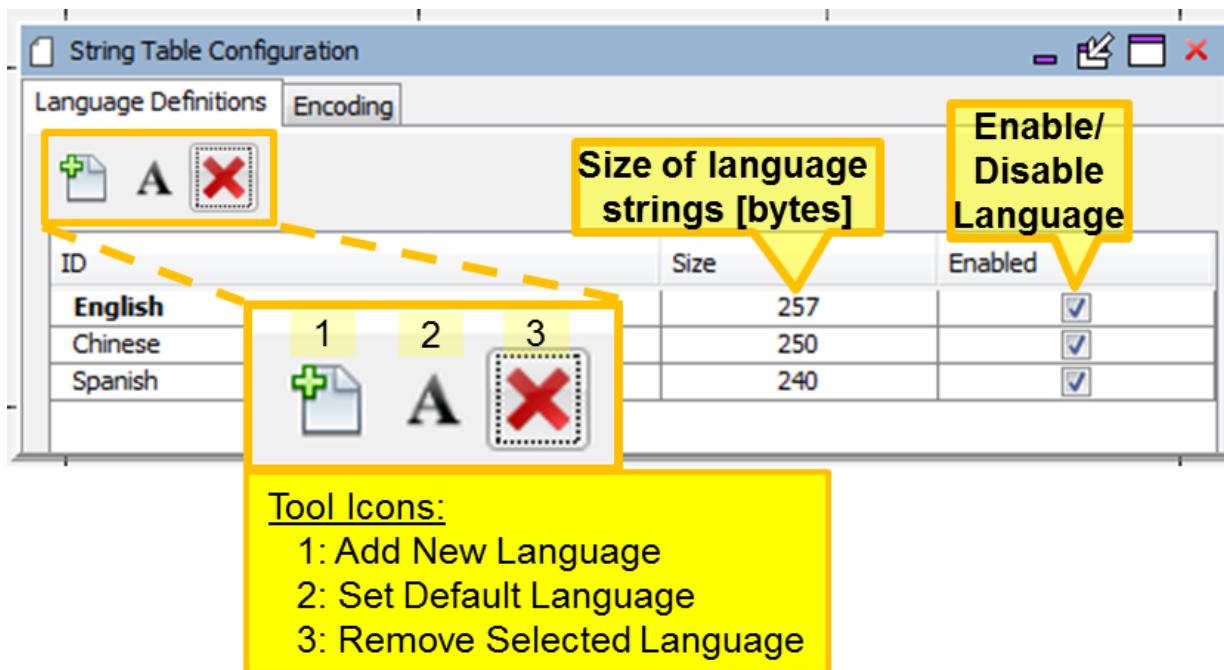
The "ID" string used for each language is merely for ease of use in building the texts to be used. "English", "American", or any other string can be used to identify that language, as long as it is understood by the application's creator when selecting the text to be used for that particular language. Then the application can switch to supporting one of its languages using "ID" strings defined.

Here is an example string asset definition, taken from the Aria Coffee Maker demonstration. This application supports English, French, Italian, and German. The text string "InfoText_Desc9" uses the Arial font, and text for each language is specified within the String Assets window.



Any number of languages can be defined as long as there is memory to store the strings needed.

The following figure shows the String Table Configuration for an application that uses English, Spanish, and Chinese.



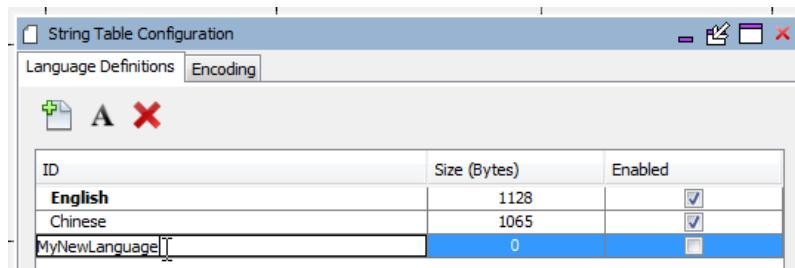
The size of all the strings for each language is shown in the Size column. String size represents the memory allocated for glyph indices for all the strings supporting that language. A language can be enabled/disabled via the check box in the Enabled column. Disabling a language removes it from the application build but keeps it in the project.

Window Toolbar

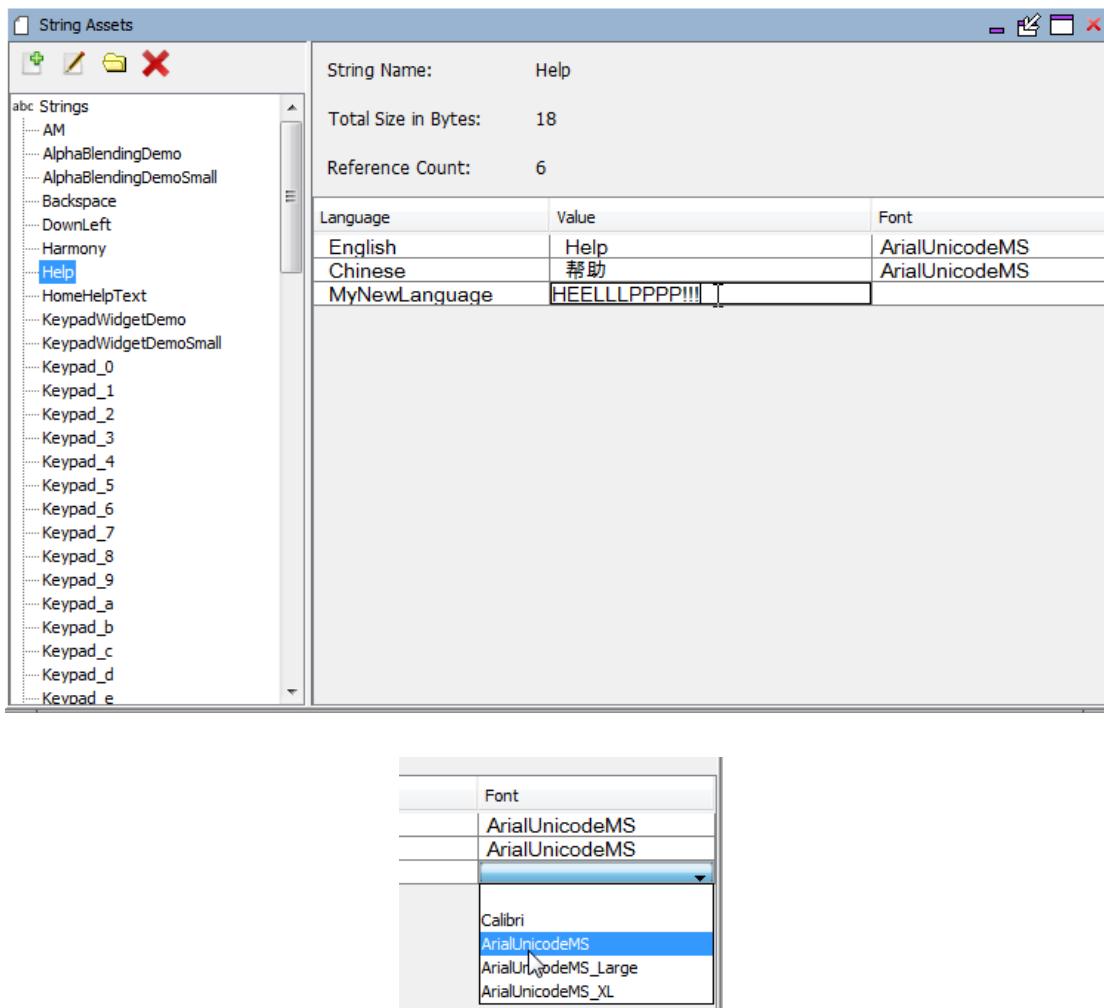
There are three icons on the toolbar:

1. **Add New Language** – Adds a new Language.
2. **Set Default Language** – Sets the application's default language. Note, this is different than the abc tool on the Graphics Composer Window toolbar. The abc icon sets the preview language for the Screen Designer panel only. This icon sets the language used by the application after boot-up.
3. **Remove Selected Language** – Removes language from the application.

Clicking **Add New Language** opens a new line, allowing the user to select and edit the new language's "ID" string.



Then, for every string defined in the application there will be a line to define the needed text, and to specify the font to be used.



If a value is not provided for the new language the string will be output as a null (empty string). If a Font selection is not provided then the string will be output as a series of blocks (?).

The [Aria User Interface Library](#) primitive, `LIB_EXPORT void laContext_SetStringLanguage(uint32_t id)`, allows the application to switch between languages using the Language ID #defines are specified in the application's `gfx_assets.h` file.

Sub-tabs

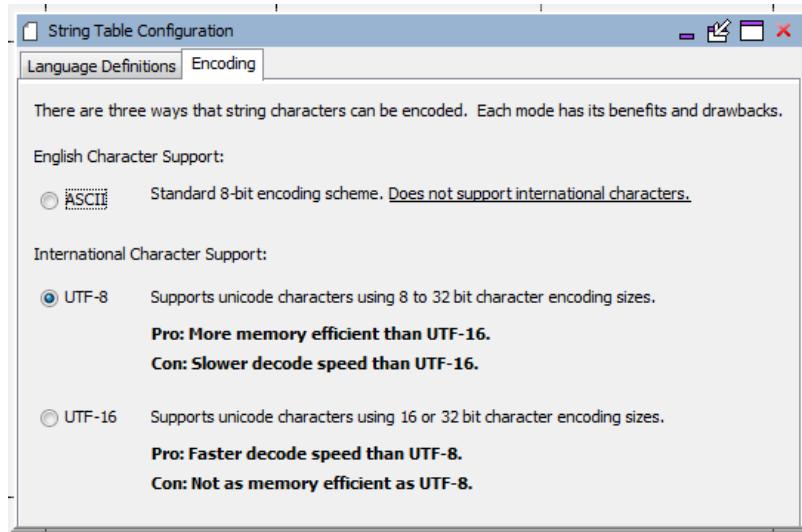
There are two sub-tabs to this window.

Language Definitions Sub-tab

This sub-tab shows the languages defined for the application. A Language can be enabled/disabled to include or exclude it from the application's generation/regeneration under MPLAB Harmony Configurator (MHC). New languages can be added by specifying a text string for the language. With a new language, go to the String Assets window to specify the text and fonts for all defined strings.

Encoding Sub-tab

Selecting the Character Encoding Format Selection Dialog icon gives you three choices for how the characters in all strings in the graphics application are encoded:



The default is ASCII. It is typically the most efficient in terms of memory and processing, but it does not support as many glyphs. Chinese text should be encoded in UTF-8 or UTF-16, but Western language text can be encoded in ASCII to save memory. The trade-off between ASCII, UTF-8, and UTF-16 depends on the application. Changing from UTF-8 to UTF-16 will double the size of all strings in the application. This is because the sizes of all glyph indices double in size. (String sizes are the sizes of glyph reference indices, not the size of the particular font glyphs used to write out the string.)

The memory utilization resulting from an encoding choice can be seen in the [Summary](#) sub-tab of the [Memory Configuration](#) window.

String Assets

Provides information on the String Assets features.

Description

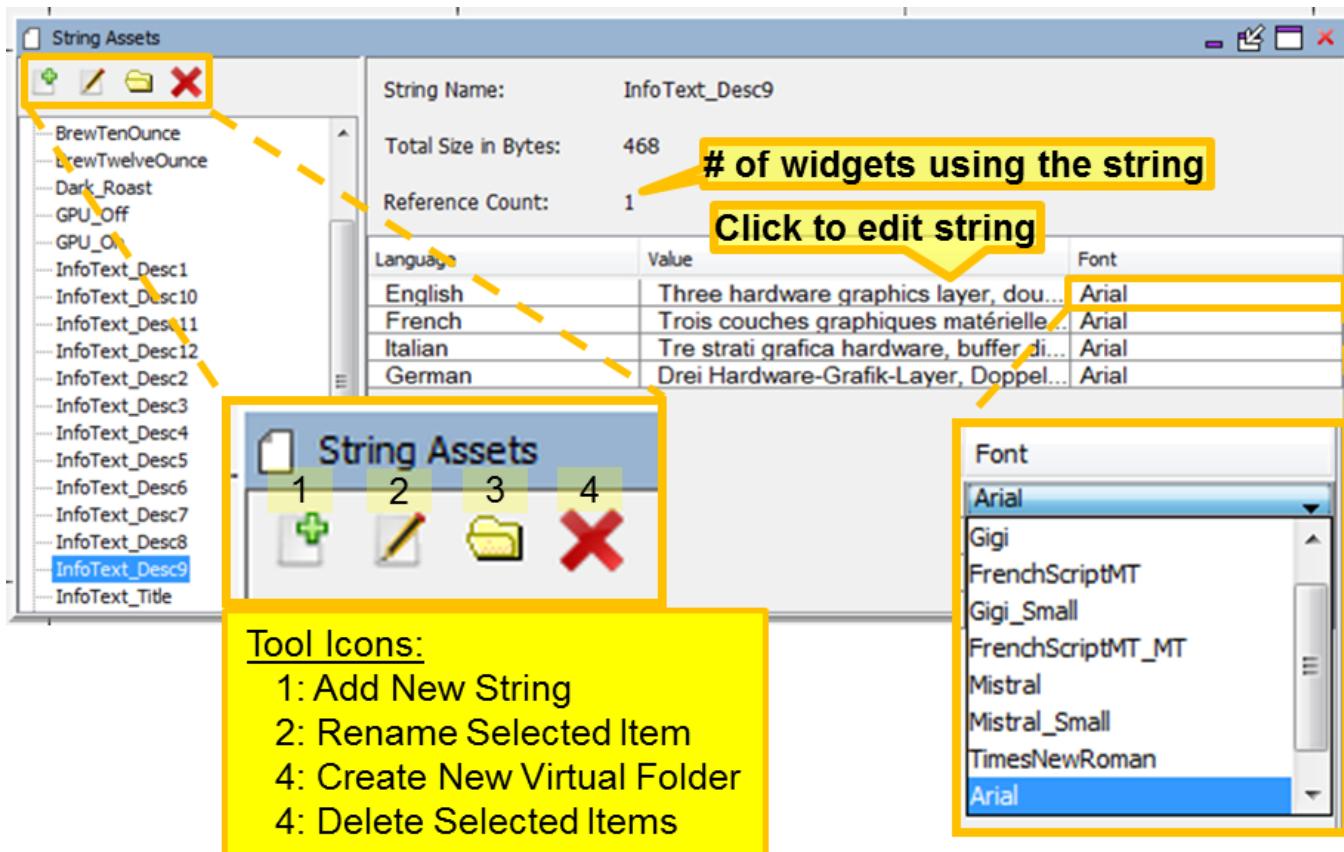
The String Assets window is launched from the Graphics Composer's Asset menu.

The String Assets window supports managing the strings in the application. Strings are referenced by graphic widgets using an application-wide unique name. This unique name is built into an enumeration that the application's C code uses. For each language supported text is defined and a font asset selected.

 **Note:** There are three dimensions to text support: Languages, Fonts, and Strings. Language "ID" strings are identified when an application supports more than one language. (In the case of single language support, the language default is provided.) Fonts are imported and organized using the Font Assets window. Strings are defined by a string name, and this name is used by widgets to reference the string. For each string and each language supported the glyphs are defined to spell out the string's text and the font is chosen for that text.

- Languages are managed within the [String Table Configuration](#) window
- Fonts are managed within the [Font Assets](#) window
- Strings are managed within the String Assets window (this topic)

The following figure shows an example taken from the Aria Coffee Maker demonstration. The string name, InfoText_Desc9, defines a string asset that is used by the application.



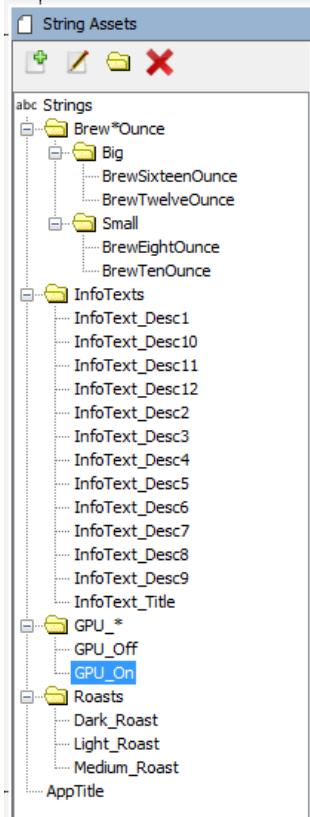
The Total Size in Byte: for a string asset represents the memory needed to store the glyph indices for all the text defined for that string asset. Adding more text will increase the number of glyph indices needed thus increasing the size of the string's memory. Adding another language will do the same, since the number of glyph indices also increases. Changing the font does not increase the size of the string's memory, but may increase the size of the font chosen if it is a "bigger" font and adds more glyphs to the new font. (By "bigger" we mean a font with more pixels, for example because it is bigger in size, or perhaps because it is anti-aliased and the original font was not.)

Note: The Reference Count shown reflects the number of build-time references to the string. Dynamic uses of a string, such as through macros or events, is not reflected in this number.

Window Toolbar

There are four icons on the toolbar:

1. **Add New String** – Adds a new string.
2. **Rename Selected Item** – Allows renaming the string.
3. **Describe Selected String** - Provides a *Description* field value for selected string.
4. **Create New Virtual Folder** – Creates a new virtual folder, allowing the user to organize strings in a hierarchy. Here's an example reorganization of the existing strings. Note the order of virtual folders or items in the list is strictly alphabetical. Virtual folders and string asset organization is merely for the convenience of the developer. Neither has an effect on how the application is built.

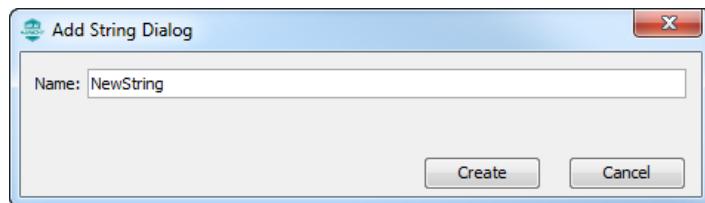


5. **Delete Selected Items** – Deletes selected strings from the application.
6. **Import String Table** - Imports an Excel CSV (Comma Separated Value) file to replace the current string table.
7. **Export String Table** - Exports the current string table as an Excel CSV (Comma Separated Value) text file.

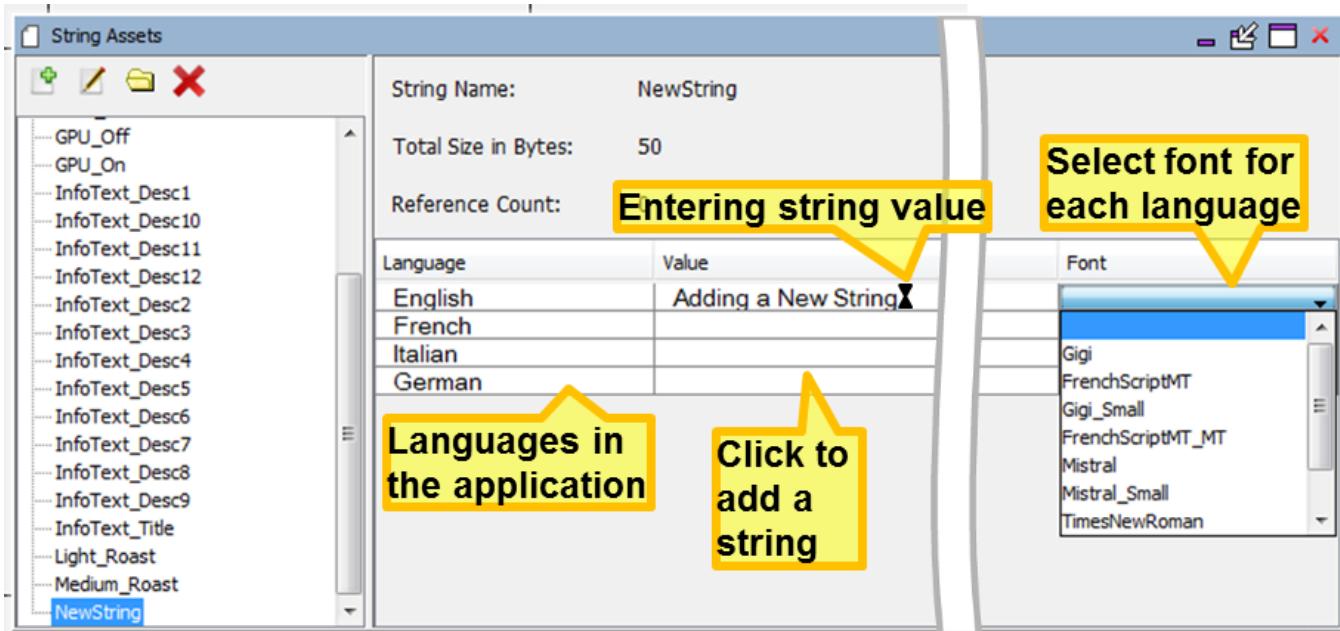
Creating New Strings

To create a new string, click Add New String ().

Selecting this icon opens the Add String dialog to name the string. The text chosen for the string name should be acceptable as a C variable.



After entering the new string's name and click Create, the following String Assets window appears.



In the String Assets window, there will be a line for each of the languages defined for the application. Provide the string text and font for each of the languages. An empty string will be used if the text is not provided. Not providing a font causes the string to be rendered as a string of boxes (□).

Importing and Exporting String Tables

Importing an Excel CSV (Comma Separated Values) file replaces the existing string assets table. Exporting creates an Excel CSV file that can be imported into another project or target configuration. Exported string tables can be manipulated in Excel, even combining multiple string tables into a single string table that can then be imported.

If the string asset table contains UTF-8 then the file cannot be directly loaded into Excel. Instead, within Excel create a new sheet. Import the string table using *Get Data*, selecting *From File*, *From Text*, or *CSV*. Then in the dialog window change the *File Origin* to *Unicode (UTF-8)*.



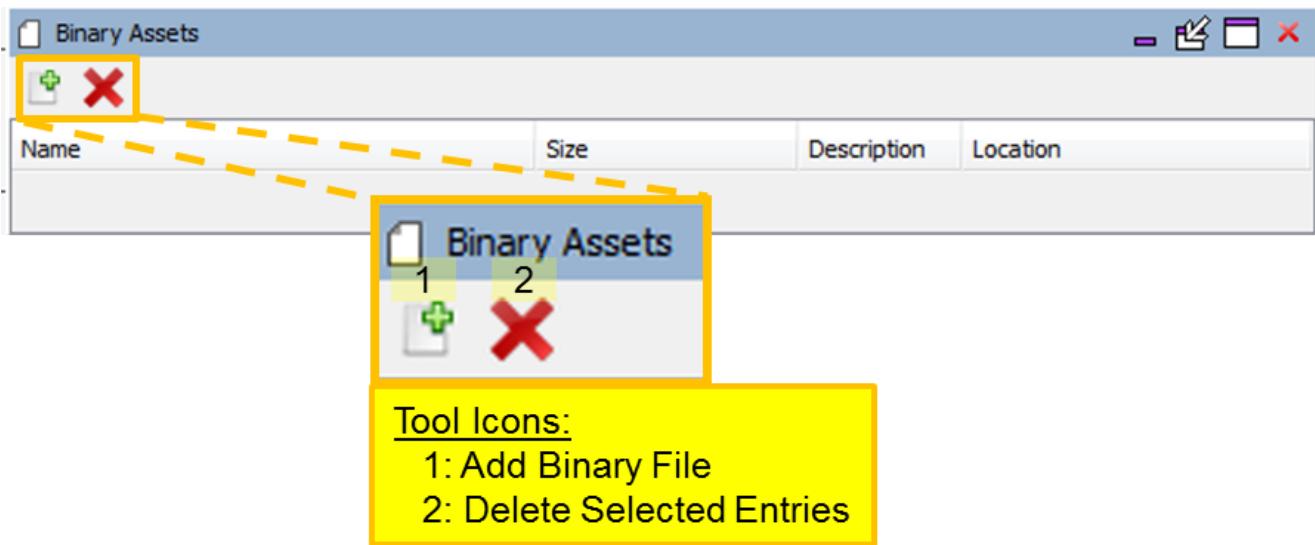
Note: Excel does not support importing UTF-16.

Binary Assets

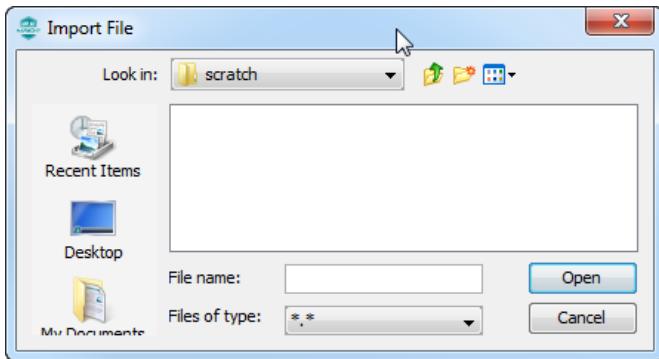
Provides information on the Binary Assets features.

Description

The Binary Assets window is launched from the Graphics Composer's Asset menu.



Selecting the Add Binary File icon () opens the Import File dialog.



This supports any formatted binary file. Developers can then add a custom-coded decoder to support the format implied by the imported file. (A future version of the GFX library will include a bin2code utility in support of this feature.)

MHGC Tools

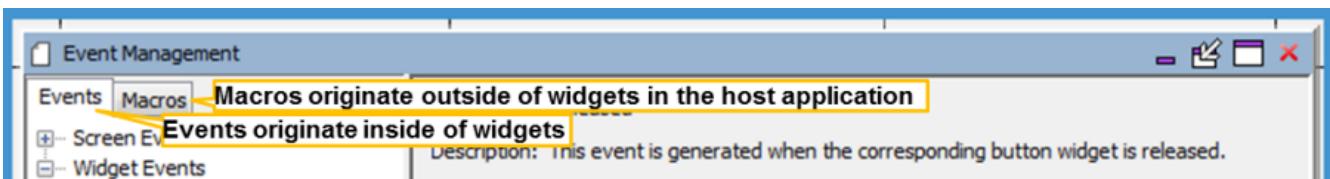
The Tools menu supports managing all graphics events, using a global palette, and estimating heap memory usage.

Event Manager

This section provide information on the Event Manager.

Description

The Graphics Composer Event Manager provides a GUI interface to manage all of the events associated with a graphics application. In a general sense, an event is an action or occurrence that is processed by software using an "event handler". Button pushes or keystrokes are widely recognized and handled events. Events related to a touch screen are commonly called "gestures". This GUI allows the assignment of actions to events associated with graphics widgets and to events outside of the graphics library. Under the Graphics Composer Event Manager tab there are two sub-tabs, one for "Events" and a second for "Macros".



The following table summarizes the difference between "events" and "macros" and provides examples of each instance of source to destination:

Differences Between Events and Macros

Source	Inside of Graphics (Destination)	Outside of Graphics (Destination)
Inside of Graphics	"Event" Example: Button changes button text	"Event" Example: Button changes board LED
Outside of Graphics	"Macro" Example: Mounting SD card changes screen	Not supported by Event Manager Tool

"Events" under the first tab are generated from within graphics widgets and can manipulate the properties of screen widgets or set semaphores that engage with the rest of the application. "Macros" are executed outside of graphics widgets by other parts of the application. "Macros" allow the application to change widget properties or behavior.

Both "Events" and "Macros" event handlers can be built using collections of "Template" actions or using "Custom" developer-provided code. Most widget properties have an associated Template action that can be used to manipulate that property in an event handler (either "Event" or "Macro"). For more information on properties and related actions, see the discussion on the Properties Window below.

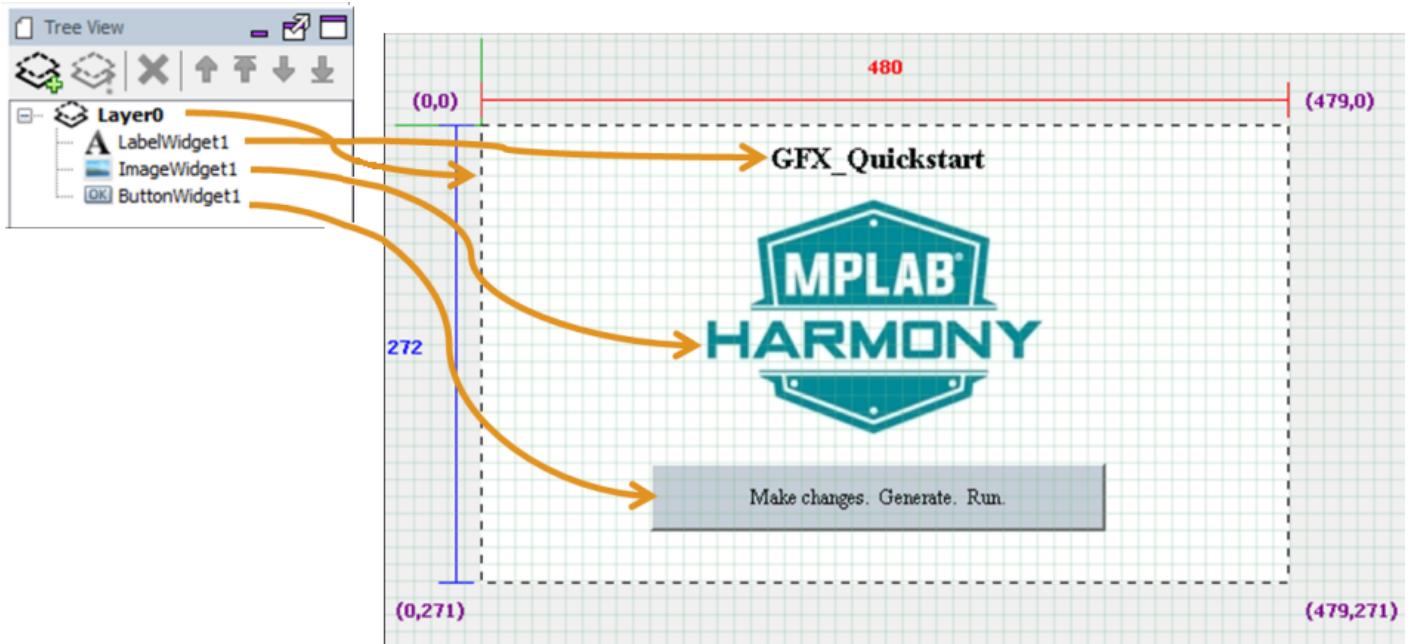
To explore these capabilities, let's look at the [Aria Quickstart](#) project after the completion of the [Adding an Event to the Aria Quickstart Demonstration](#) Quick Start Guide.

Graphics Composer Events

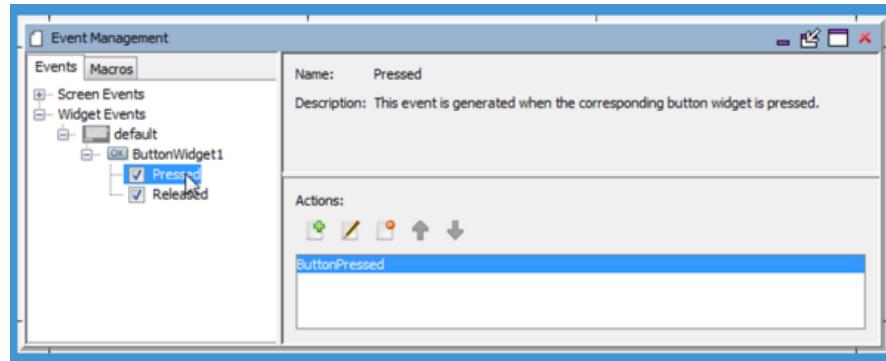
Events implement event handlers for events that originate inside of graphics primitives.

Description

The Graphics Composer Screen Designer shows that there is one layer and three widgets in this demonstration.



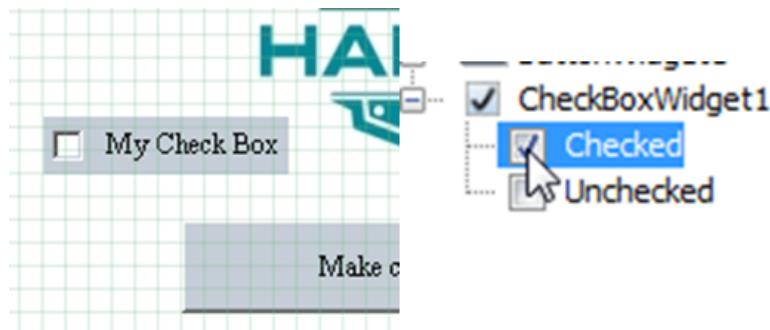
Of the three widgets shown above, only ButtonWidget1 can have events associated with it, one for button pressed and a second for button released. This can be seen in the Graphics Composer Event Manager window, which is available from the Tools menu:



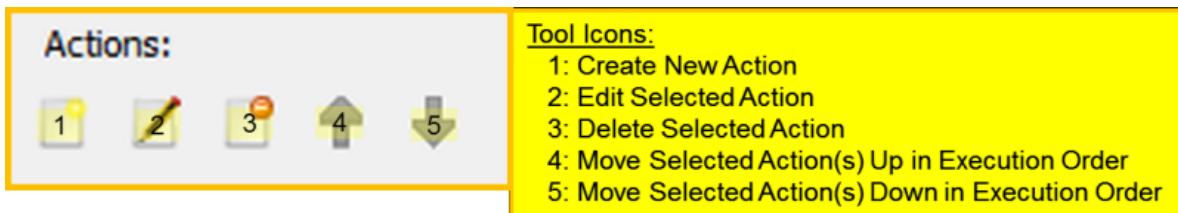
The events shown under “ButtonWidget1” are mirrored in the widget’s properties. Selecting or clearing an event in one window does the same in the other window, thus enabling (selecting) or disabling (clearing) the corresponding event.



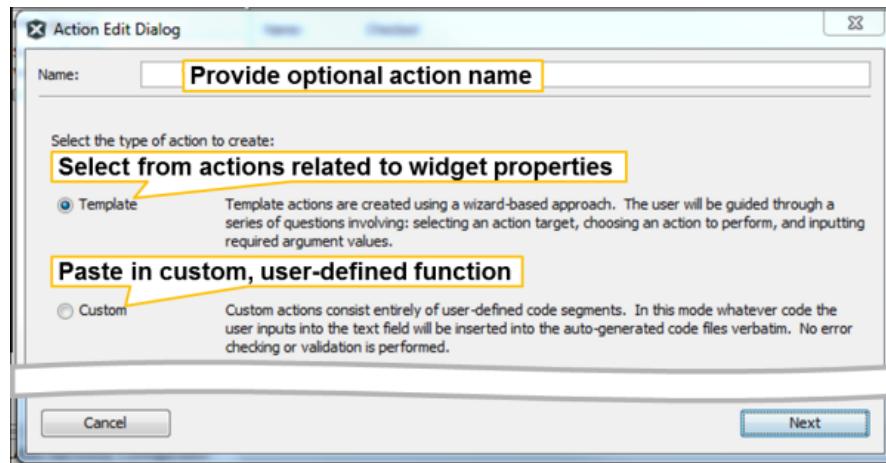
We can add a Check Box widget to the applications display and then use the Event Manager to assign actions to the widget’s events. A Check Box widget has two events, one for being “Checked” (i.e., selected) and another for being “Unchecked” (i.e., cleared). Enabling the “Checked” event then allows the selection of the action or actions for that event.



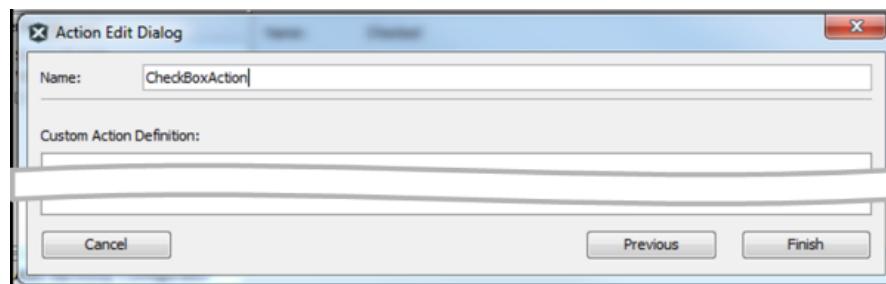
The Actions: sub-window has five tool icons for managing the actions associated with an event:



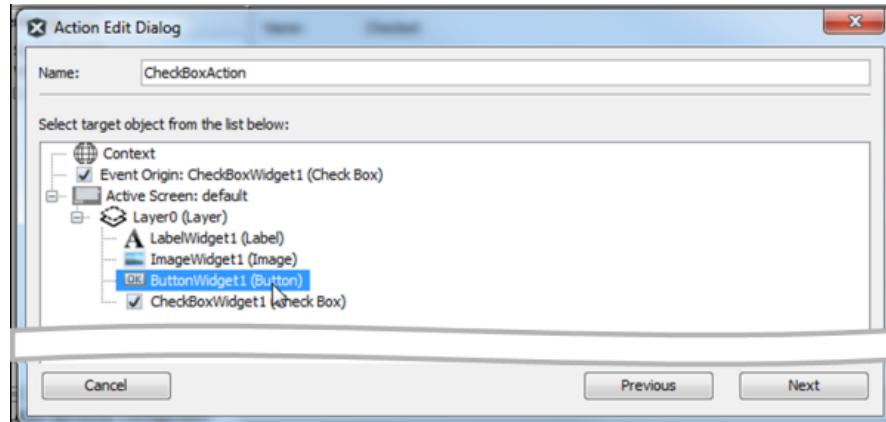
Clicking the Create New Action icon () opens the Action Edit dialog.



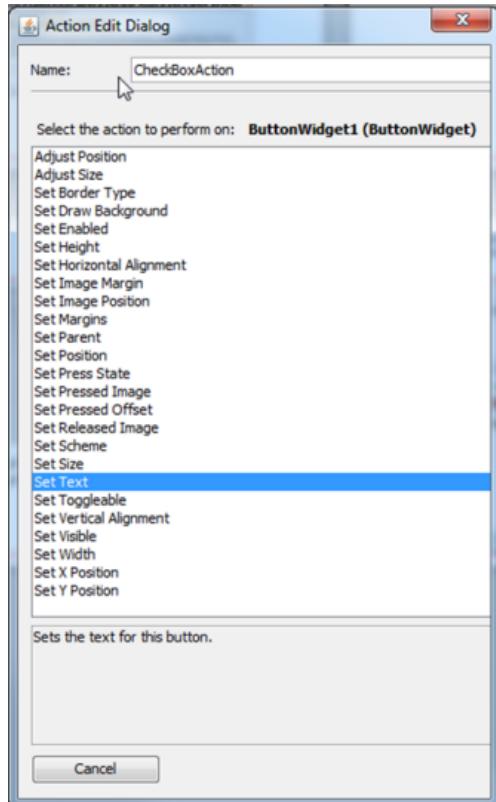
Select Custom and click **Next**, the user will see the following dialog. Unfortunately, there is no C code error checking with this window. It just copies the code into libaria.c and libaria.h. If there is a problem with the code, the user will not know about it until the user try to build the application. An alternative is just to type a comment such as /*My event goes here*/, generate the code, and then find out where this comment landed in the code. (Typically, inside libaria_events.c, or libaria_macros.c) The user can then write the action routine from within the MPLAB X IDE editor and compile just that file to debug the code written.



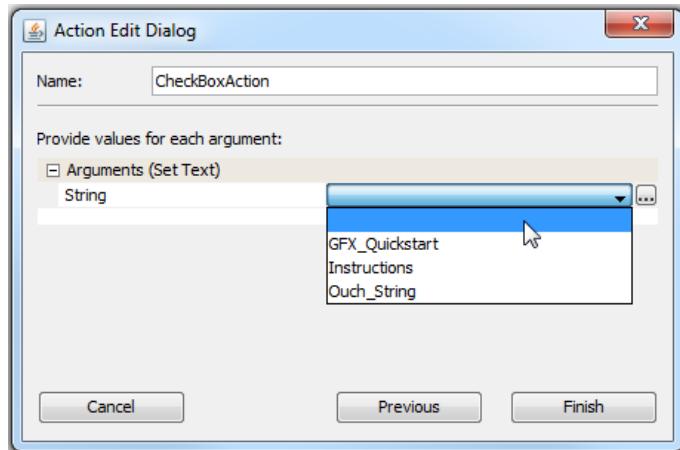
If Template is selected, the Action Edit dialog will update, as follows. Select ButtonWidget1.



As shown previously, the user next needs to select the widget that if the desire is to manipulate with this action. Note that the event originated with CheckBoxWidget1, but the event's action can manipulate any of the existing widgets. In this case, ButtonWidget1 has been selected. Clicking Next will then bring up a list of the actions available in manipulating a button widget.



Select the “Set Text” action, which will then change the button’s text property, followed by NEXT, which will open a dialog to select the text string for this action.



Next, select from the available (already defined) strings which text to use for the button’s text field. Press the Finish button to complete the definition of this action.

Screen vs. Widget Events

All screens can generate events but not all widgets can do so.

Description

Screen Events

As shown previously, the Graphics Composer Event Manager, Events sub-tab supports screen events when the screen is visible

(On Show) and hidden (On Hide). These events can define event handlers based on Template actions or Custom, user-defined code.

Widget Events

Not all widgets can generate an event. For example, a Label Widget has nothing to generate, it just sits there on the screen, labeling. Here is a list of the widgets that can generate an event:

- **Button** – Pressed and Released events
- **Check Box** – Checked and Unchecked events
- **Draw Surface** – Draw Notification event
- **Image Sequence** – Image Changed event
- **Key Pad** – Key Click event
- **List Wheel** – Select Item Changed event
- **List** – Selection Changed event
- **Progress Bar** – Value Changed event
- **Radio Button** – Selected and Deselected event
- **Scroll Bar** – Value Changed event
- **Slider Widget** – Value Changed event
- **Text Field** – Text Changed event
- **Touch Test** – Point Added event

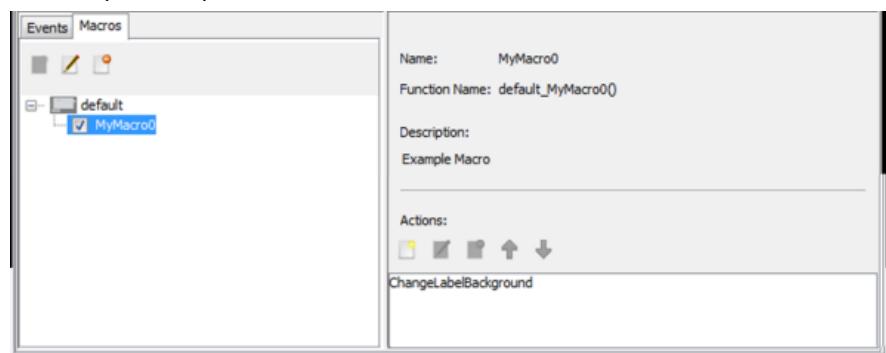
Graphics Composer Macros

Macros implement event handlers for events that originate outside of graphics primitives.

Description

Macros implement event handlers for events that originate outside of graphics primitives such as widgets and are designed to change or manipulate widgets inside of the graphics part of an application. (Events that originate outside of graphics and don't touch the graphics part of the application are outside of the scope of the Graphics Event Manager and are not discussed here.)

The following figure shows a simple example of a macro.



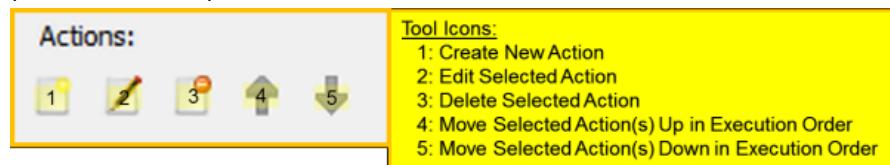
The toolbar for Macros has three icons.



Creating a new macro and selecting its actions is just like that of a widget event:

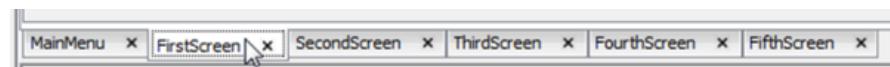
1. Create a new macro using the “Create New Macro” tool. The check box to the left of the new macro’s name enables/disables the macro. Clearing it removes the macro from the next code generation.
2. Select the new macro and edit it using the second icon (shown previously).
3. In the Actions: window, select Create New Action. An optional name can be provided in the Name: box. The user can then

choose to use a Template and select a predefined action or Custom to create a customized action.

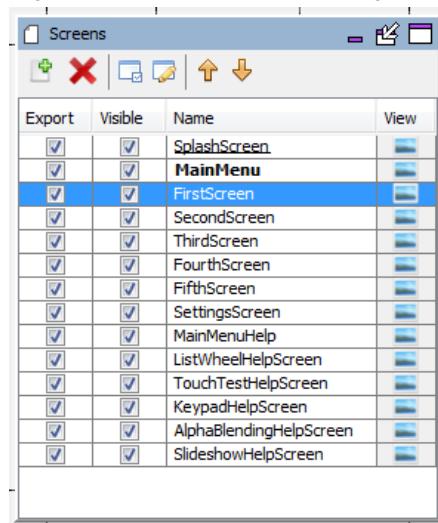


4. A “Custom” action is picked, proceed as discussed previous in Graphics Composer Events. When using templates the next step is to choose the target widget for the action. This choice is limited to those only the widgets in the currently “active” screen. If the application has multiple screens and the widget that is being targeted is not part of the currently active screen, the user would need to change the active screen.

- Changing the active screen can be done by selecting the corresponding screen tab at the bottom of the Graphics Composer Screen Designer



- Alternately, the screen can be switched using the Graphics Composer Manager: Screens tab



5. After selecting the target widget for this macro, click Next button to select an action related to this widget. (Just as with template-based widget events.) The macro can contain more than one action, targeting more than one widget.

Graphics Events Testbed

Additional examples of graphics events and macros are found in the Graphics Events Testbed.

Description

Additional examples of graphics events and macros are found in the Graphics Events Testbed. For the details go to [Advanced Topics > Graphics Events Testbed](#).

Global Palette

Provides information on the Global Palette features.

Description

The Global Palette window is launched from the Graphics Composer’s Asset pull-down menu.

Using a Global Palette enables frame buffer compression for the LCC graphics controller. It creates a 256 color look up table (LUT) and then changes the entire user interface design to adhere to that LUT. Frame buffers are stored as 8 bits/pixel (bpp) indices rather than 16-32 bpp colors. The display driver performs a LUT operation to change each LUT index into a color before writing to the display/controller memory. This enables the use of double buffering, without using external memory, on devices that could not support it before. It also supports single buffering on larger displays. Of course, running the LUT requires more processing on the host. Currently only the LCC graphics controller supports this feature. The Aria demonstration Aria Basic Motion is an example of how using a Global Palette greatly improves the efficiency and capabilities of a design.

Enable the Global Palette by clicking on the Enable Global Palette check box in the window or using the *File > Settings > Color Settings* menu. (The Global palette can always be disabled. MHGC will then restore the project back to its original configuration.)

If the global palette is enabled here, you will have to check the "Enable Global Palette Mode?" in the Configuration Option for the Aria Graphics Library component in MHC's Project Graph:



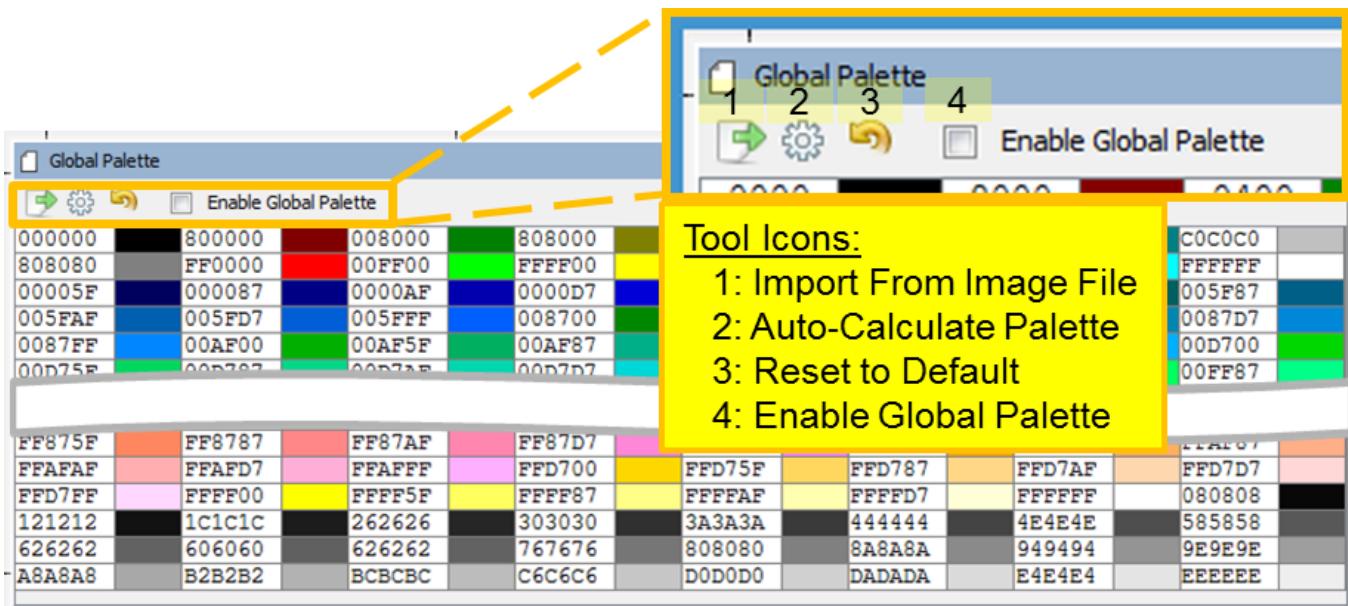
The results of enabling the Global Palette:

- 8bpp frame buffers. In the case of the most common demonstrations this means a 50% reduction in the size of the frame buffer.
- This also opens up the capability to support a single frame buffer for some larger displays.

What is lost by enabling the Global Palette:

- First and foremost - No Dynamic Colors. Dynamic colors are unlikely to match up with an entry in the global palette's look-up table.
- No alpha blending capability. The level of alpha blending can be changed during run-time. (See No Dynamic Colors.)
- No JPEGs or PNGs. Again, no dynamic colors. All images in MGHC will be changed to the color mode of the project, and generated as Raw.
- No font anti-aliasing. Again, no dynamic colors. While the 8-bits/pixel for each glyph is known, the color of the text depends on the color scheme used, and color schemes can change at run time.
- Additional overhead when performing LUT (index->color) operations in the display driver.

The following figure shows the default "Global Palette" when Project Color Mode is set to RGB_888.



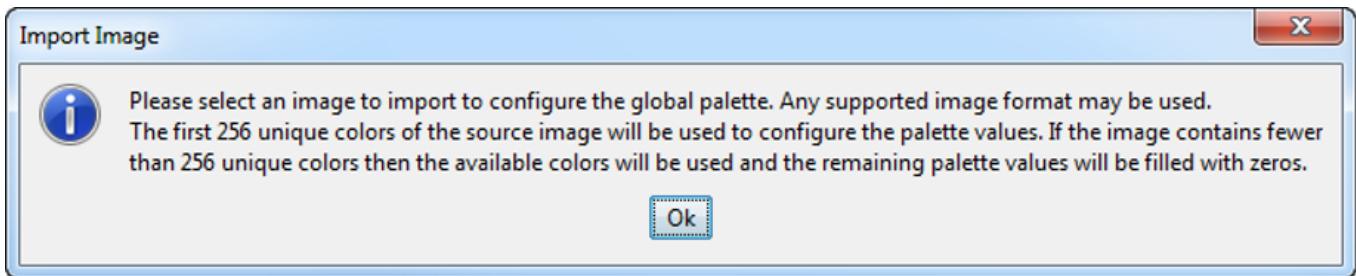
This default palette is good for designs that use a wide array of colors. MHGC also supports developing a custom palette by importing an image defining the palette or by analyzing the pixel colors already in use by the application's images. The palette's color mode is determined by the Project Color Mode, which is determined by the graphics controller.

Clicking on an entry in the palette will bring up the Color Picker dialog window, allowing you to edit the entry's color.

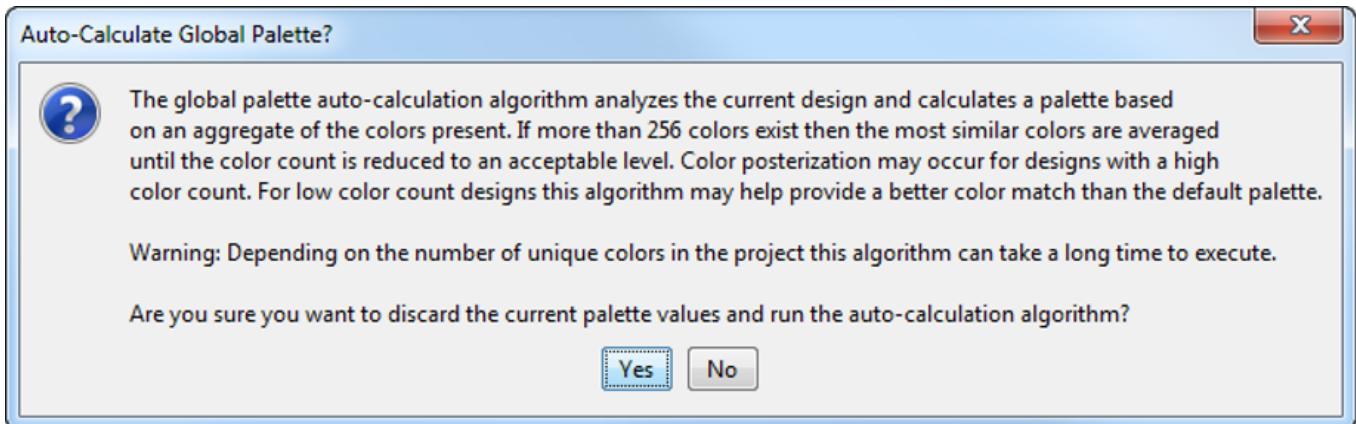
Window Toolbar

There are four icons on the toolbar:

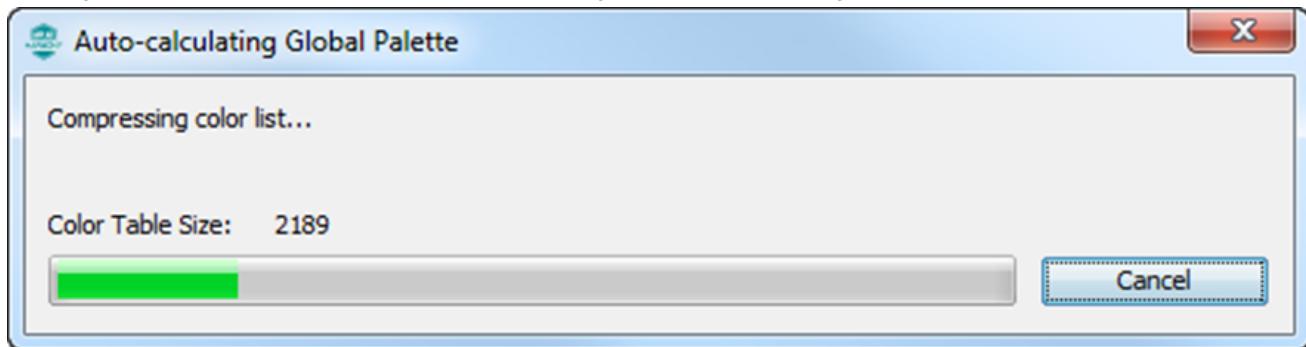
1. **Import From Image File** - Importing a global palette from an image file. Selecting this brings up the following warning. Images can be imported as a BMP,.GIF, JPEG, and PNG (but not TIFF).



2. **Auto-Calculate Palette** – Calculates a new palette using the current design. Selecting this brings up the following warning.

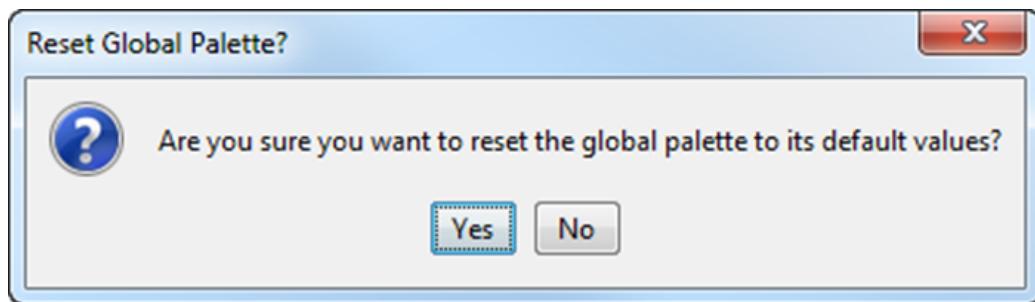


- Selecting **Yes** opens a status window that shows the progress made in selecting a palette of 256 colors

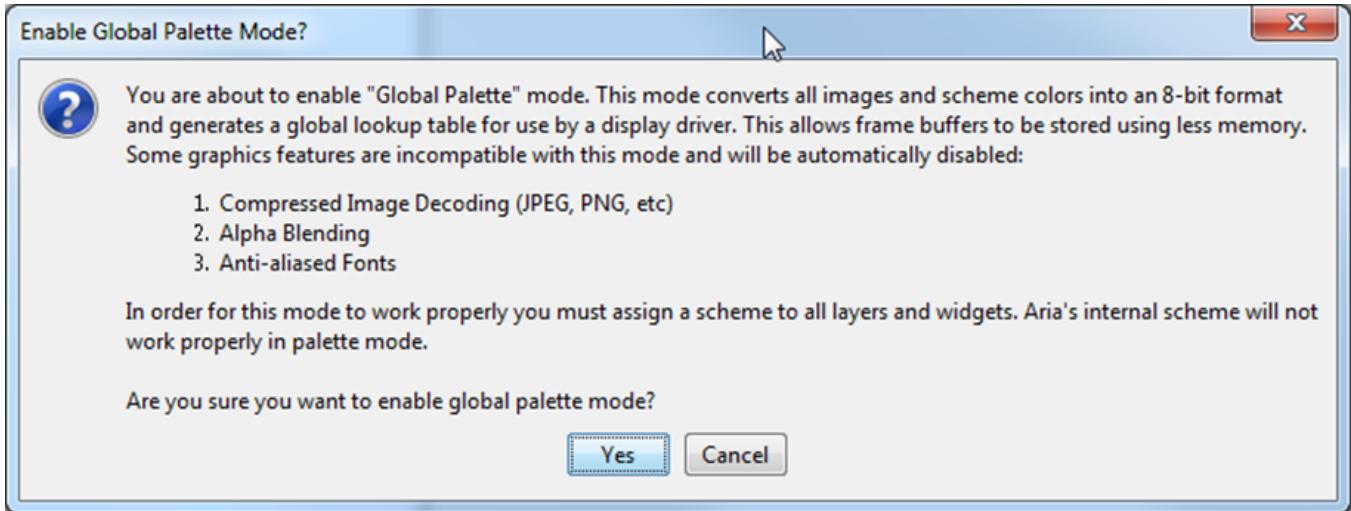


- This can be lengthy operation, but it will effectively generate a palette better tailored to the design. However, extreme (or rare) colors will be changed to nearby, more-plentiful colors, thereby eliminating some of the contrast in images. Whites will tend to darken and blacks lighten. This can be remedied by editing the calculated palette to whiten the whites, darken the blacks, and make other colors closer to the original. This of course may increase the posterization of the image, but that is a natural trade-off in using only 256 colors.

3. **Reset to Default** – This returns the Global Palette to its default values, which opens the Reset Global Palette dialog.



4. **Enable Global Palette** – This performs the same function as *File > Settings: Using a Global Palette*. Selecting this opens the Enable Global Palette Mode warning.



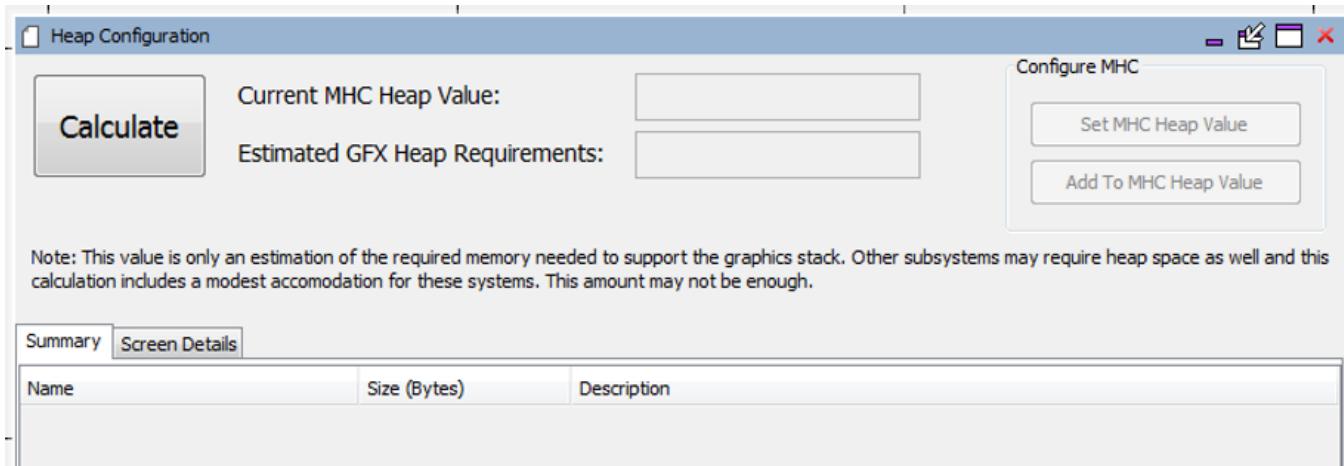
Heap Estimator

Provides information on heap space allocation.

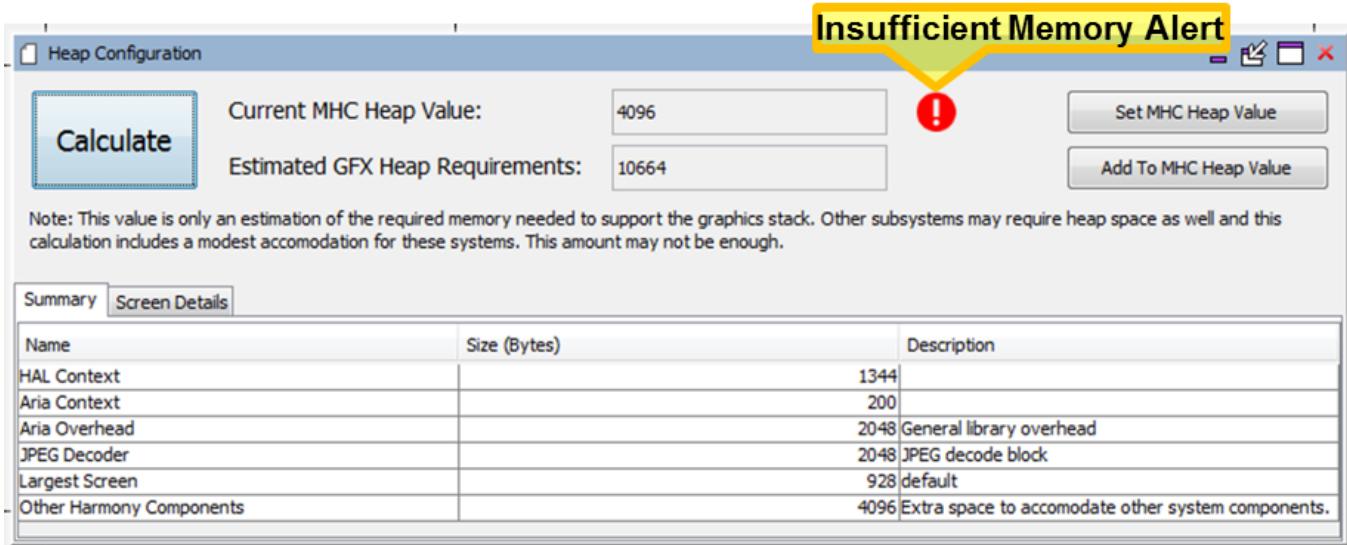
Description

Many parts of a graphics design are implemented using memory allocated from the application's heap space. Therefore, it is important to allocate sufficient memory for the heap. This tool can estimate heap usage by the allocation based on the widgets, layers, screens, and decoders currently in the design.

When launching the tool from the Tools menu, the Heap Configuration window appears.



Clicking **Calculate** estimates heap usage. The following figure shows what occurs within the [Aria Quickstart](#) demonstration if the heap space is only 4096 bytes:



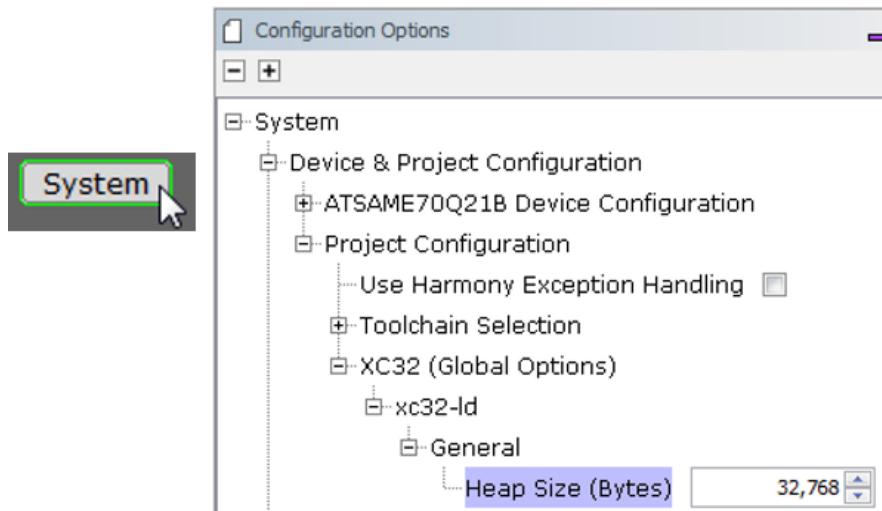
The **Summary** tab shows how the estimated heap requirements was derived by summing up all the sizes shown under the "Size (Bytes)" column. Note that the largest contribution comes from the screen requiring the largest heap allocation (in this case MainMenu).

If there is insufficient memory allocated to the heap, an exclamation point (!) appears in the window. Hovering the mouse pointer over this icon and the following message appears:

The current MHC heap value is not high enough to accommodate the estimated requirements of the graphics stack. Please adjust the value.

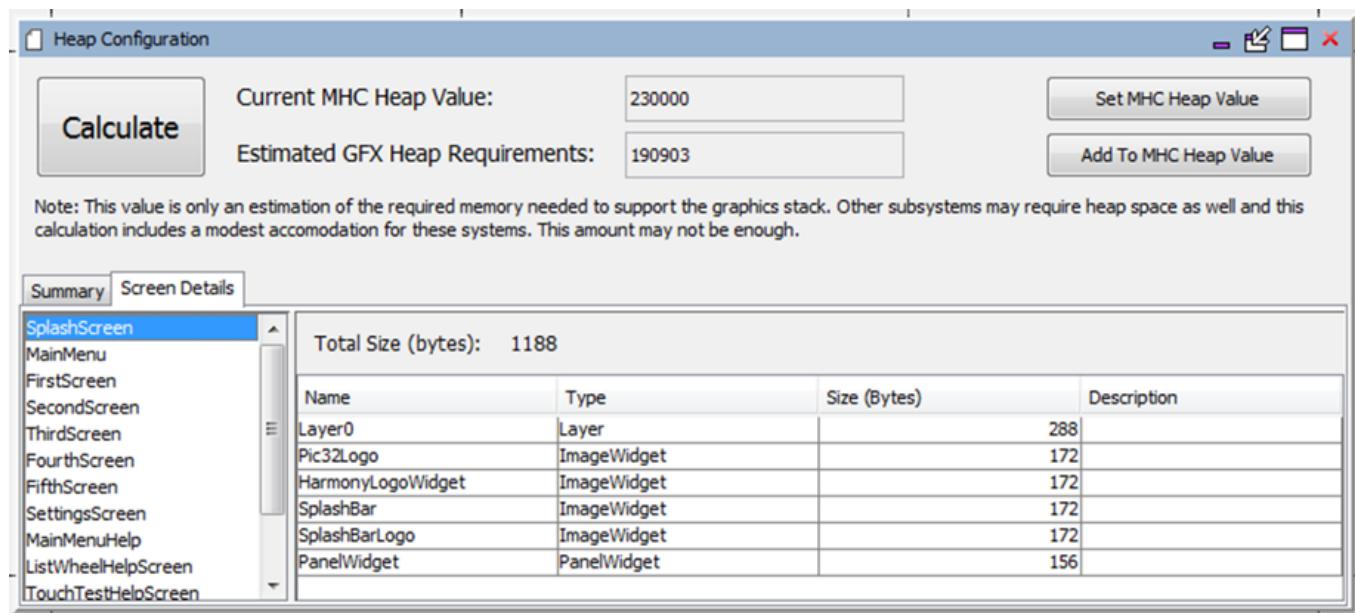
The user can click **Set MHC Heap Value** to reset the heap allocation to match the estimated requirements. Selecting **Add to MHC Heap Value** adds the estimated heap requirements to the current heap value. (In the case above, this would change the heap allocation to 4096+10664 bytes.)

Alternately, the user can set the heap allocation to a larger value by going to MHC's Project Graph panel, clicking on the System component, and then editing the Heap Size shown under the Configuration Options panel:



The Screen Details tab (from the Aria Showcase demonstration) shows screen-by-screen the heap space needed for each layer and widget on the screen selected.

 **Note:** After you have updated the Heap Size, either using the Heap Estimator tool or by directly editing the value as shown above, you must regenerate the project using the Generate Code button. This will update the actual heap size value used in building the application.



Clicking the “Name” column will alphabetize the list. Clicking the “Size (Bytes)” column sorts the assets by size, with the largest at the top and smallest at the bottom.

This sub-tab can help in managing the application’s utilization of heap space. For example, excess use of cached backgrounds for widgets can become ruinously expensive, expanding the application’s need for heap well beyond the capabilities of the device. As an example, consider a screen label from the Aria Showcase demonstration.



The Heap Estimator tool shows that if caching is enabled for the label’s background, this widget requires 23699 bytes of heap to store the widget. Note that the label is twice the size of the text it contains, so one way of reducing the cost of the widget is to make it smaller, thereby reducing the number of background pixels that must be stored. If the label is resized, the heap allocation is reduced to 11688 bytes, which is a drop of approximately 50%. Finally, if the background is changed from “Cache” to “Fill” the widget only needs 188 bytes.

The lesson learned is to use Cache as a background only for widgets where it is absolutely necessary and to make the “cached” widgets as small as possible.

Widget Colors

Provides information on widget coloring.

Description

Widget Colors

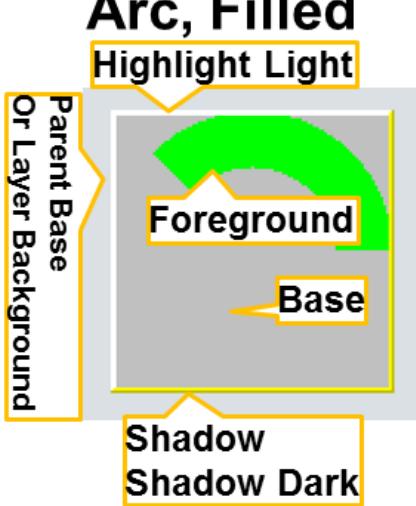
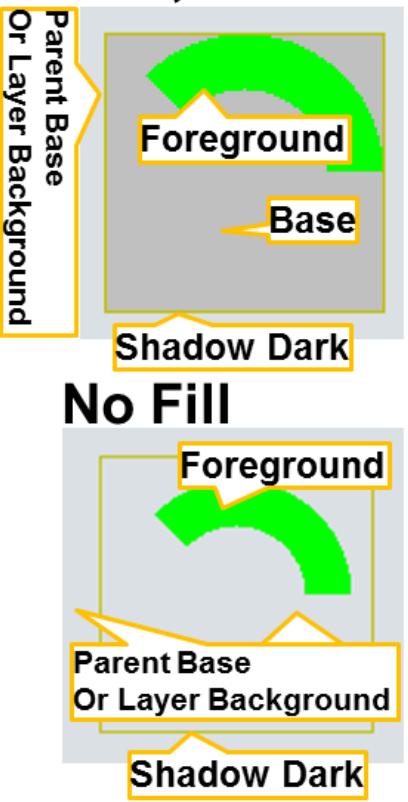
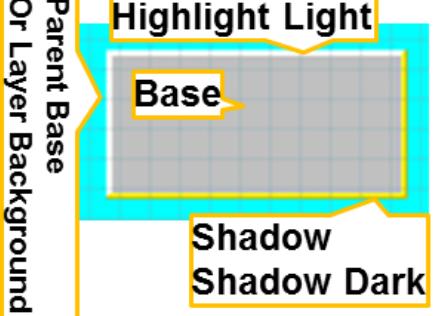
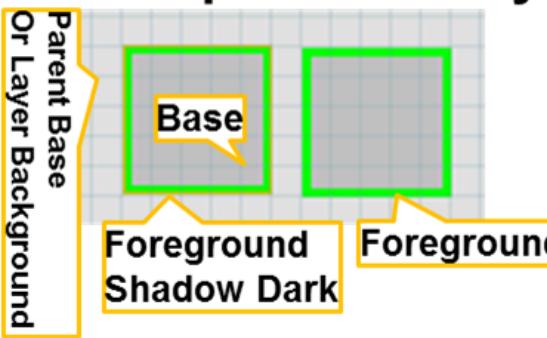
Widget coloring can be customized by creating additional color schemes and assigning these customized schemes to a subset of the widgets uses. For example, a ButtonColorScheme could be customized and used only for Button Widgets.

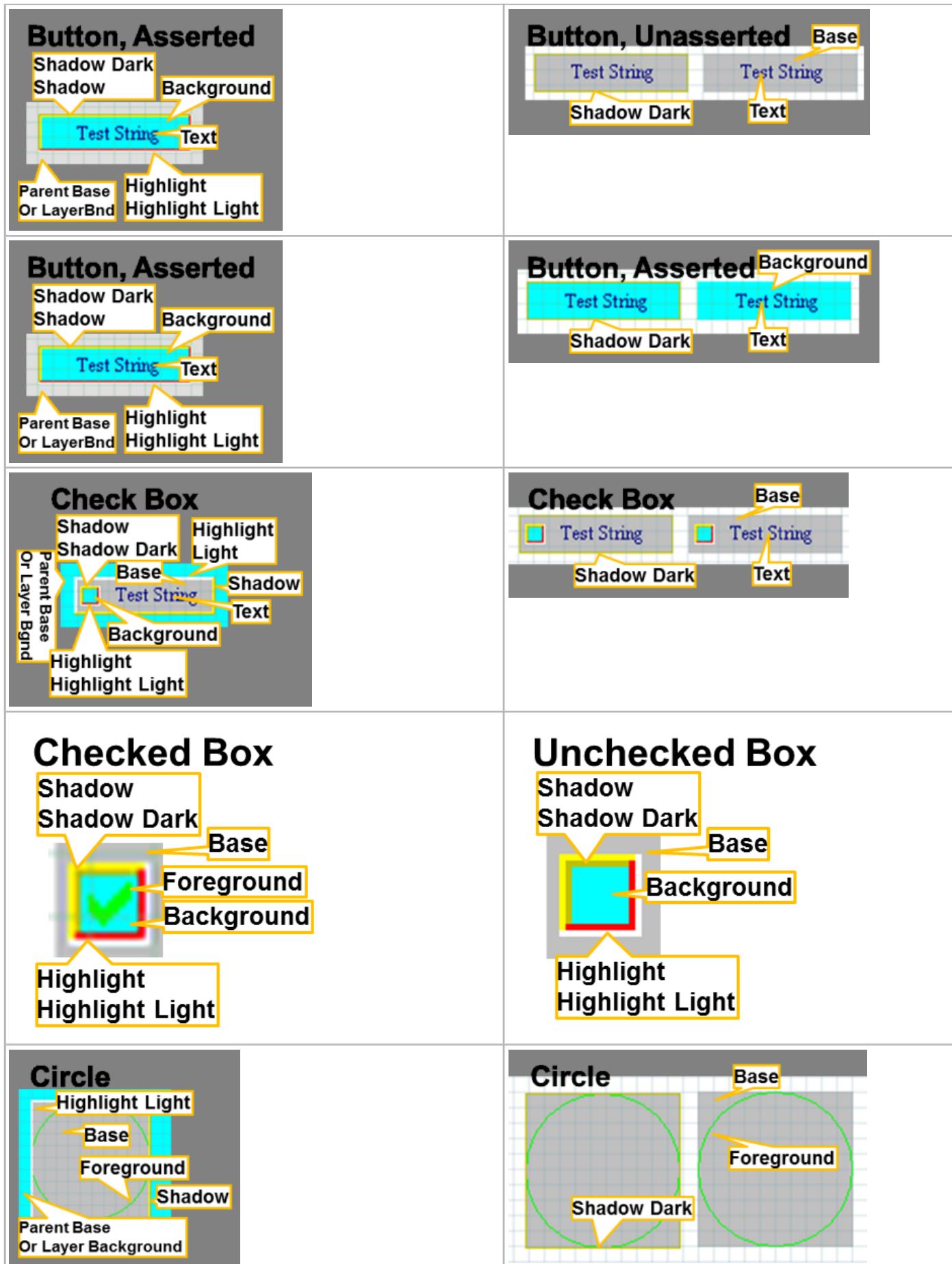
To help highlight the different colors available for each widget, a “CrazyScheme”, with extreme contrast among the 16 available colors, was used as the color scheme for each widget:

Scheme	
Name	CrazyScheme
Colors	
[+] Base	[192,192,192]
[+] Highlight	[255,0,0]
[+] Highlight Light	[255,255,255]
[+] Shadow	[255,255,0]
[+] Shadow Dark	[192,192,0]
[+] Foreground	[0,255,0]
[+] Foreground Inactive	[0,128,0]
[+] Foreground Disabled	[0,82,0]
[+] Background	[0,255,255]
[+] Background Inactive	[0,192,192]
[+] Background Disabled	[0,128,128]
[+] Text	[0,0,128]
[+] Text Highlight	[0,0,255]
[+] Text Highlight Text	[224,224,224]
[+] Text Inactive	[178,188,191]
[+] Text Disabled	[96,101,102]

Use this color scheme to help identify the relevant colors for the widgets listed below.

The left column shows the coloring assignments for a Bezel boarder. The right side shows Line/No Border color assignments.

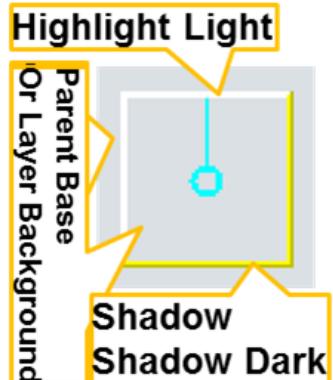
Widget With Bezel Border	Widget With Line Or No Borders
<p>Arc Widget:</p>  <p>Arc, Filled</p> <p>Highlight Light</p> <p>Foreground</p> <p>Base</p> <p>Shadow Shadow Dark</p> <p>Parent Base Or Layer Background</p> <p>No Fill</p> <p>Highlight Light</p> <p>Foreground</p> <p>Parent Base Or Layer Background</p> <p>Shadow Shadow Dark</p>	 <p>Arc, Filled</p> <p>Foreground</p> <p>Base</p> <p>Shadow Dark</p> <p>Parent Base Or Layer Background</p> <p>No Fill</p> <p>Foreground</p> <p>Parent Base Or Layer Background</p> <p>Shadow Dark</p>  <p>Arc, Filled</p> <p>Foreground</p> <p>Base</p> <p>Parent Base Or Layer Background</p> <p>No Fill</p> <p>Foreground</p> <p>Parent Base Or Layer Background</p>
<p>Bar Graph:</p>  <p>Bar Graph Boundary</p> <p>Highlight Light</p> <p>Base</p> <p>Shadow Shadow Dark</p> <p>Parent Base Or Layer Background</p>	 <p>Bar Graph Boundary</p> <p>Base</p> <p>Foreground</p> <p>Shadow Dark</p> <p>Foreground</p> <p>Parent Base Or Layer Background</p>



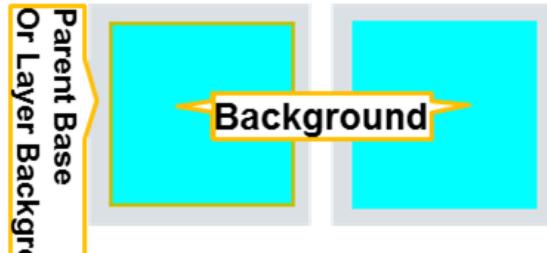
Circular Slider Border, Filled



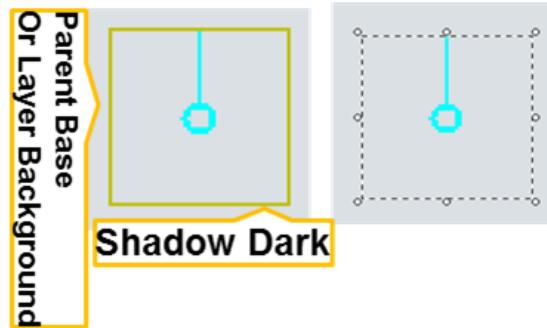
No Fill



Circular Slider Border, Filled



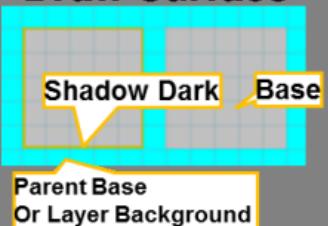
No Fill



Draw Surface



Draw Surface



Gradient



Gradient



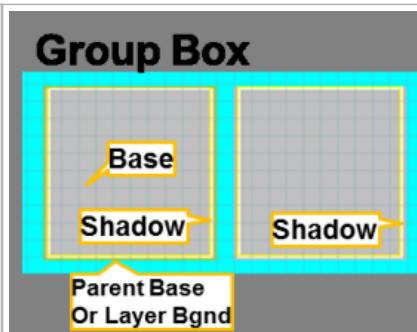
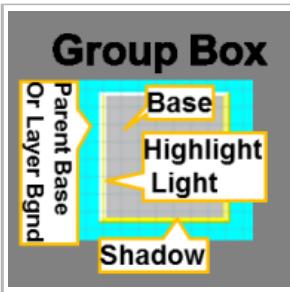
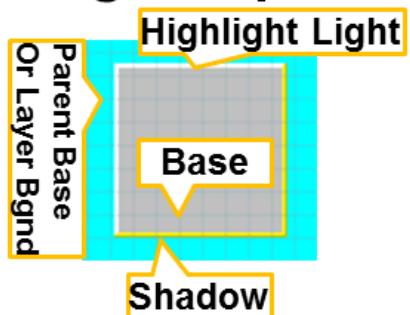


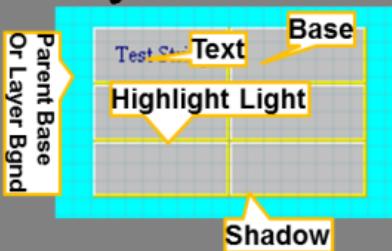
Image or Image Sequence



Image, Image Plus, or Image Sequence



Key Pad



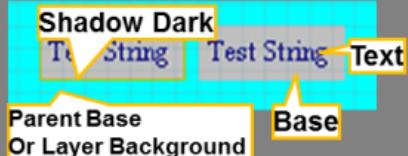
Key Pad



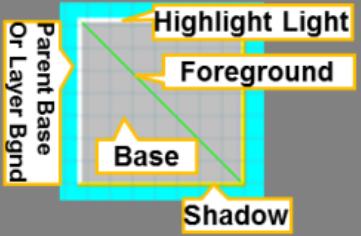
Label



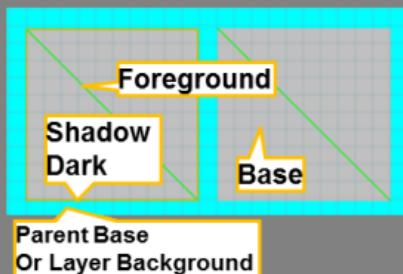
Label



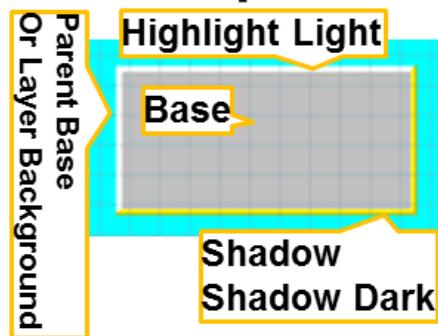
Line



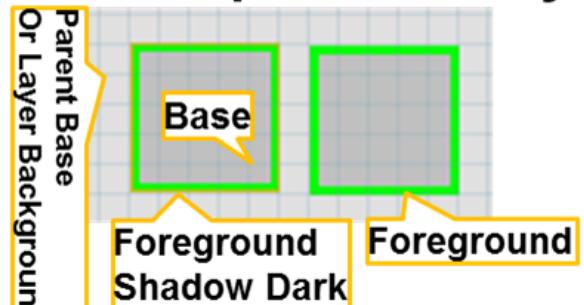
Line



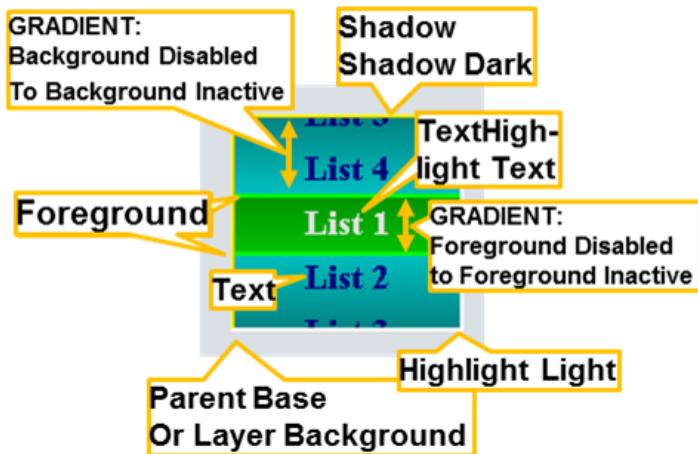
Line Graph Boundary



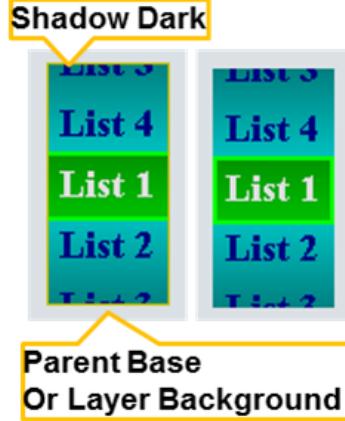
Line Graph Boundary



List Wheel

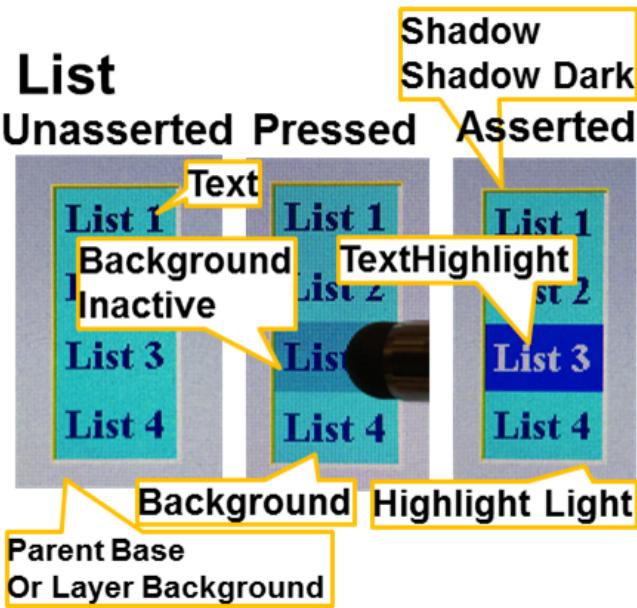


List Wheel



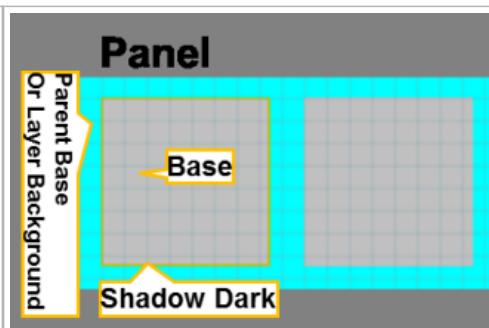
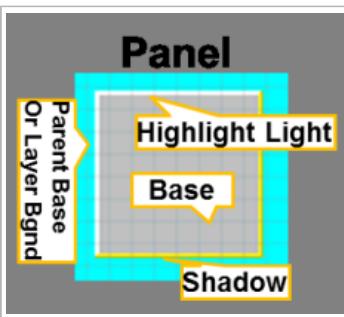
List

Unasserted Pressed Asserted

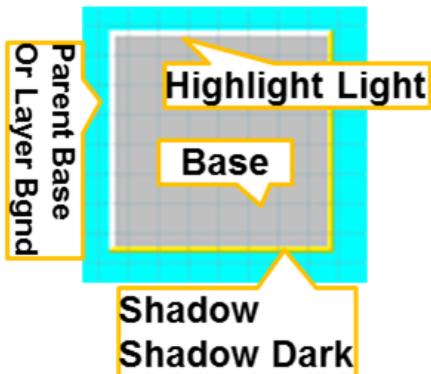


List





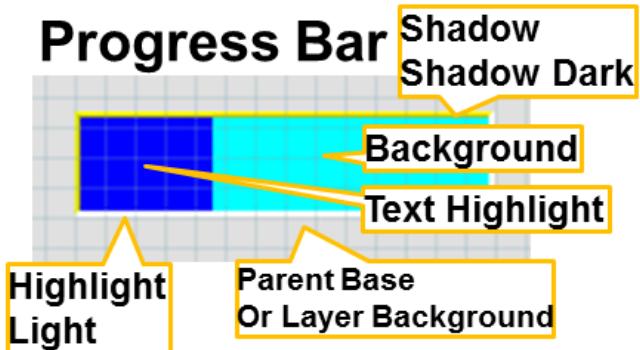
Pie Chart Boundary



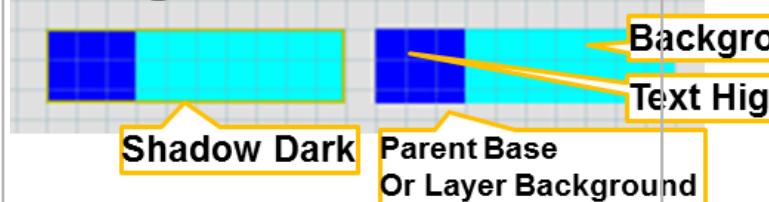
Pie Chart Boundary



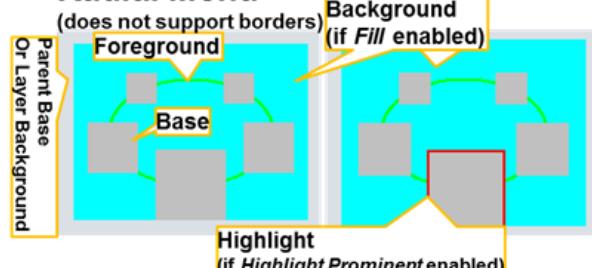
Progress Bar

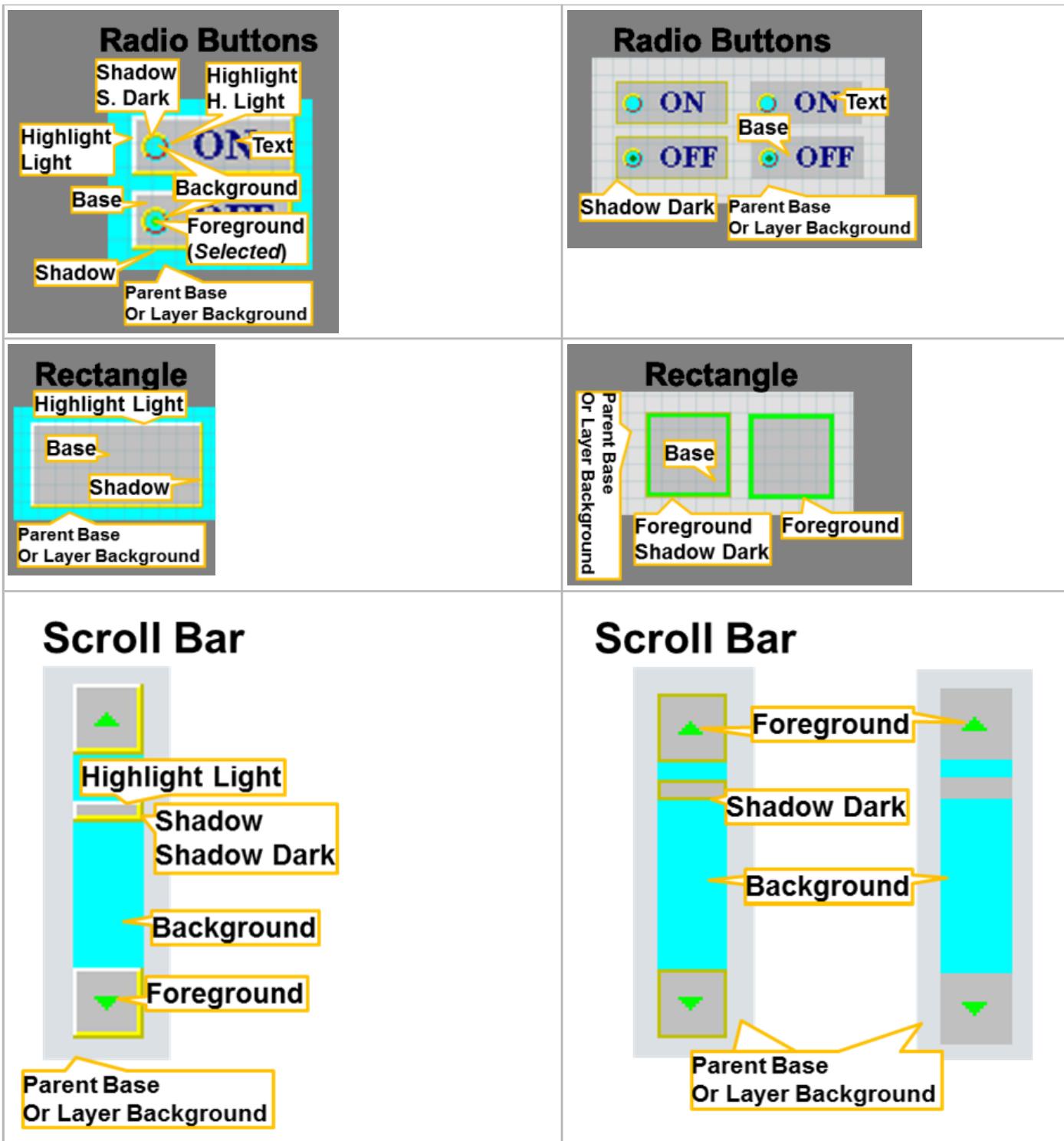


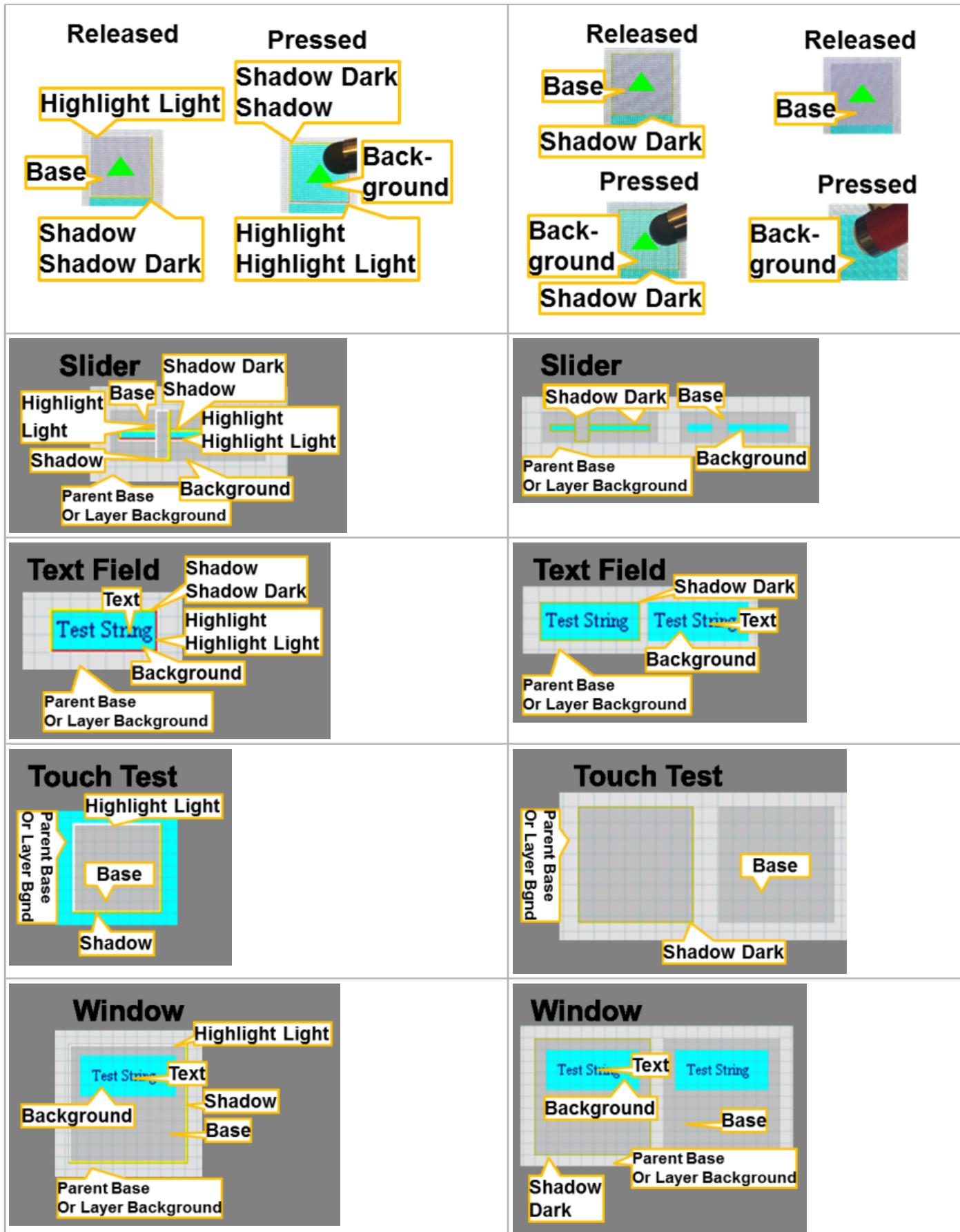
Progress Bar



Radial Menu







Code Generation

This topic describes using the graphics composer to generate code.

Description

MPLAB Harmony Graphics Composer data is generated the same way as the rest of the project within MHC through the Generate button.

- **libaria_harmony.h/c** – These files provide the interface that binds libaria to the overall MPLAB Harmony framework. They contain the implementations for the standard state management, variable storage, and initialization and tasks functions. If the touch functionality is enabled then the touch bindings are also generated in `libaria_harmony.c`.
- **libaria_init.h/c** - These files contain the main initialization functions for the library state and screens. The header file contains all predefined information for the library state including screen IDs, schemes, and widget pointers. The main initialization function initializes all schemes and screens, creates all screen objects, and sets the initial state of the library context. As each screen must be capable of being created at any time, each screen has a unique create function that can be called at any time by the library. The `libaria_init.c` file contains these create functions.
- **libaria_events.h/c** – The event files contain the definitions and implementations of all enabled MHGC events. Each event implementation will contain all generated actions for that event.
- **libaria_macros.h/c** – The macro files contain the definitions and implementations of all defined MHGC screen macros. A macro is similar to an event in that it can contain actions. However, it is meant to be called from an external source such as the main application.
- **libaria_config.h** – This file contains configuration values for the library. These are controlled through settings defined in MHC.
- **gfx_display_def.c** – This file contains generated definitions for enabled graphics displays.
- **gfx_driver_def.c** – This file contains generated definitions for enabled graphics drivers.
- **gfx_processor_def.c** – This file contains generated definitions for enabled graphics processors.
- **gfx_assets.h/c** – These files contain generated asset data.

Advanced Topics

This section provides advanced information topics for MHGC.

Speed and Performance of Different Image Decode Formats in MHGC

Provides information and recommendations for image decode formats.

Description

MHGC supports various image formats and the MHGC Image Assets Manager provides the ability to convert and store a source image into to the following formats

- Bitmap RAW
- Bitmap Raw Run-Length Encoded (RLE)
- JPEG
- PNG
- Predecoded RAW Bitmap in DDR (PIC32MZ DA)

The following table shows the relative rendering time and Flash memory requirements of the different image formats in the MPLAB Harmony Graphics Library. The rendering time includes decoding the image and drawing it to the screen. This information is helpful when optimizing a MPLAB Harmony graphics project for performance and Flash memory space. For example, as shown by the red highlighted text in the table, a 40x40 pixel 16-bit RAW image renders 2.38 times faster and uses 2.59 times more Flash space than a JPEG image.

Image Format	Resolution (Pixels)	Size In Flash (Bytes)	Relative Frame Update Rate		Relative Size In Flash	
			Versus RAW (16-bit)	Versus JPEG (24-bit)	Versus RAW (16-bit)	Versus JPEG (24-bit)
Raw 16-bit	40x40	3200	1	2.38	1	2.59
	100x100	20000	1	2.73	1	6.23
	200x200	80000	1	2.67	1	11.23
Raw 16-bit RLE	40x40	1796	0.71	1.68	0.56	1.45
	100x100	9288	0.57	1.55	0.46	2.89
	200x200	29916	0.56	1.5	0.37	4.2
JPG (24-bit)	40x40	1237	0.42	1	0.39	1
	100x100	3212	0.37	1	0.16	1
	200x200	7123	0.38	1	0.09	1
PNG (32-bit)	40x40	1999	0.34	0.8	0.62	1.62
	100x100	6782	0.25	0.68	0.34	2.11
Predecoded RAW in DDR (from JPG)	40x40	1237	0.81	1.93	0.39	1
	100x100	3212	2.68	7.32	0.16	1
	200x200	7123	10.06	26.83	0.09	1

Predecoded Images in DDR (RAW)

For PIC32MZ DA devices with DDR, the MHGC Image Asset Manager provides an option to predecode images from Flash and store them into DDR as RAW images. The GPU is used to render the decoded image from DDR to the frame buffer. This provides a faster render time than an equivalent RAW image in Flash memory, specifically for large images (up to 10 times faster for a 200x200 image). Conversely, predecoding small images 40x40 pixels or smaller in DDR may not render faster due to the additional overhead of setting up the GPU.

Recommendations:

- If there is adequate DDR memory available, consider predecoding images to DDR for best performance
- Using JPEG images and predecoding them into DDR can provide the best rendering performance and most Flash memory savings.



Note: The images are decoded from Flash to DDR memory by the Graphics Library during initialization and may introduce delay at boot-up, depending on the number and size of the images.

RAW Images

RAW images provide fast rendering time, as there is no decoding needed. However, depending on image content, it can be two times larger than a Run-Length Encoded (RLE) image and about 3 to 10 times larger than a JPEG.

Recommendation:

For small images that are to be rendered frequently, consider using a RAW image for better performance

JPEG Images

JPEG images provide the most Flash space savings, but are slower to render compared to RAW and RAW RLE.

Recommendations:

- If images are large and not used frequently, consider using the JPEG image format to save flash memory space
- If DDR memory is available, consider predecoding JPEG images in DDR for better rendering performance

Run-Length Encoded RAW Images

In terms of rendering speed and size, RAW RLE images are in between RAW and other compressed formats like JPEG or PNG. Depending on the image contents, RAW RLE can be approximately 1.5 times faster than JPEG, but could be significantly larger in size for large images. Again, depending on the image content, RAW RLE can be about half the size and performance of a RAW image.

Recommendation:

If optimizing your application for both speed and flash size consider using RAW RLE images

PNG Images

Among the image formats, PNG is slowest to render and requires more memory to decode.

Recommendations:

- Unless fine levels of alpha-blending are needed, it is better to use other image formats to achieve the best performance. Use

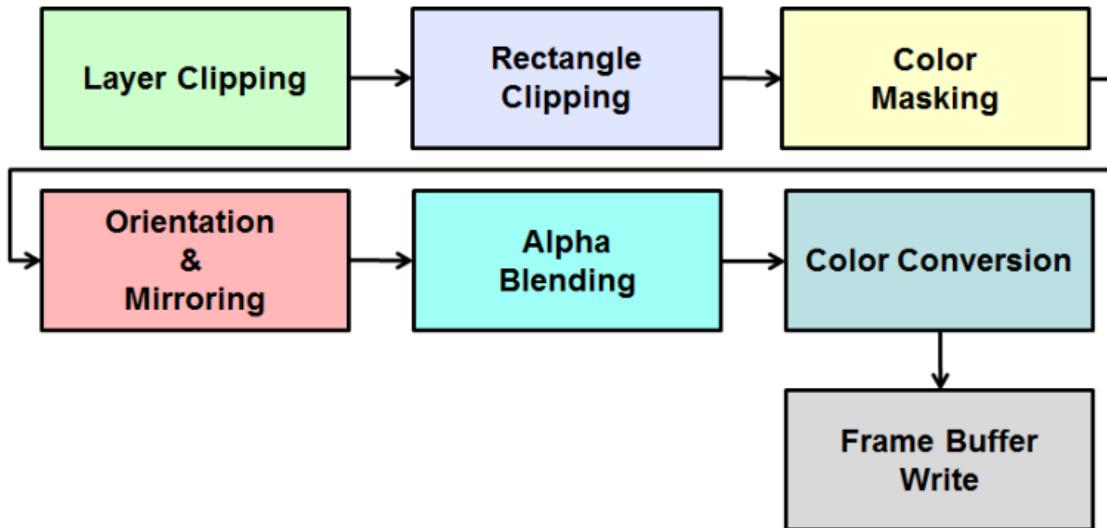
- the MHGC Asset Manager to convert the source PNG image and store it in a different image format.
- If you would like to use an image with a transparent background, it may be better to use a RAW RLE image with background color masking to achieve the same effect with better performance than a PNG. Color masking is supported in the MHGC Image Asset Manager.

Draw Pipeline Options

This section details how to use the Graphics Pipeline.

Description

The nominal rendering pipeline for an image is shown in the following figure.



The order of rendering for other widgets may differ. For example, for a colored rectangle the color mask is first checked. If the rectangle's fill matches the mask color defined then there is nothing to draw.

Graphics Pipeline

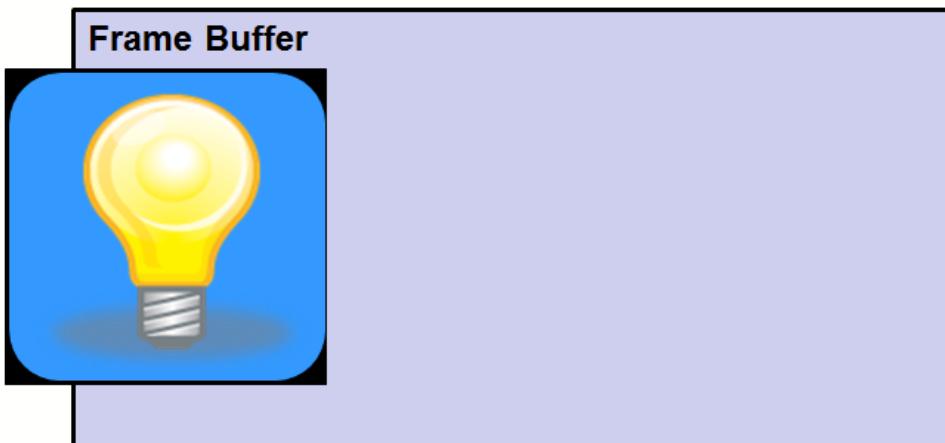
Provides information on the graphics pipeline.

Description

Layer Clipping

In order of the processing, Layer Clipping is first applied to the image. If the image extends beyond the edges of the layer that contains it then those pixels are not drawn. Failure to clip out-of-bound pixels can cause the application to crash. The following figures shows an example of layer clipping:

Before applying layer boundaries:



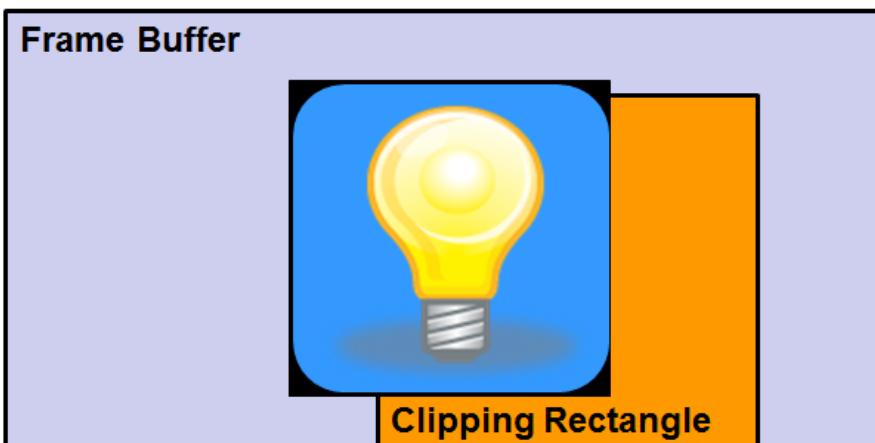
After applying layer boundaries:



Rectangle Clipping

Next, the image is clipped to the boundaries of any widgets that contain it as a parent, such as a rectangle.

Before applying the clipping rectangle.:



After applying the clipping rectangle:

Frame Buffer



Color Masking of Pixels

Pixels in the image are matched to a mask color. If the colors match the pixel is discarded (not drawn). In the following example, the black border of the image is removed by defining the mask color to be black.

Before applying color mask:

Frame Buffer



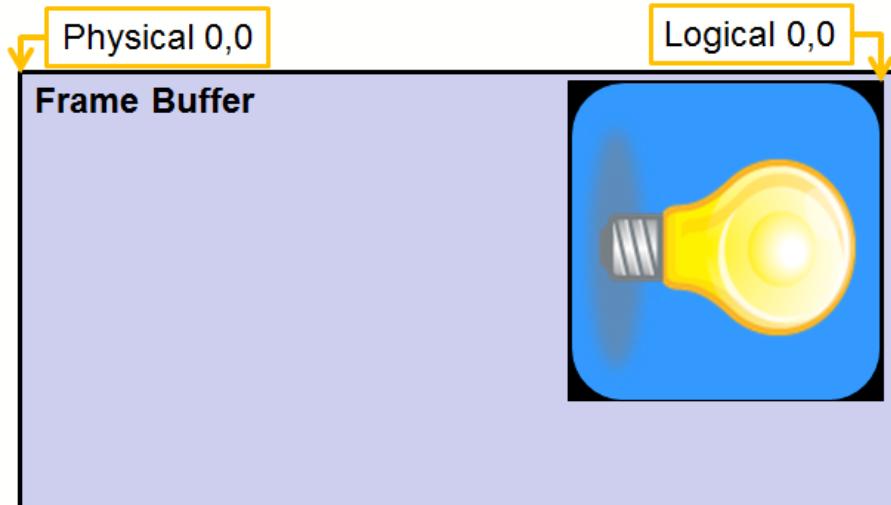
After applying color mask:

Frame Buffer

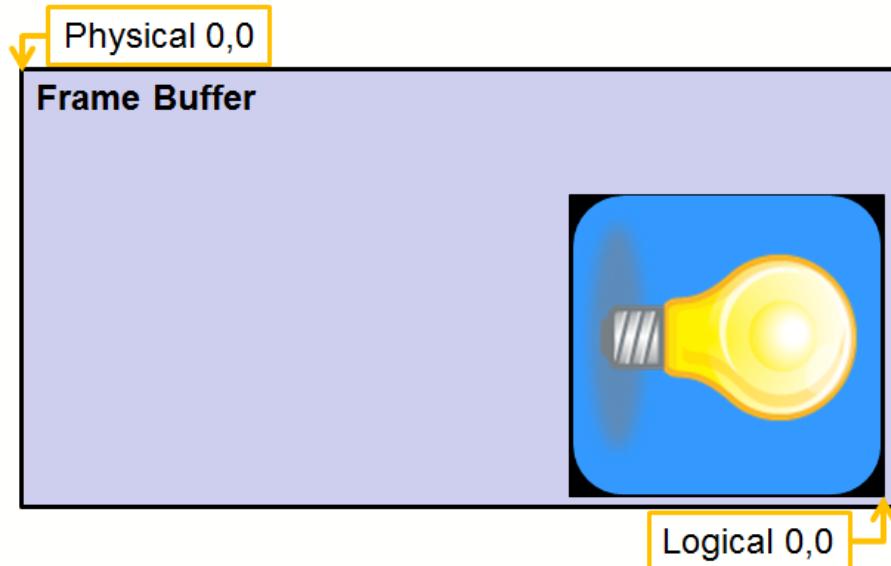


Orientation and Mirroring

The logical orientation of the graphics design may not match the physical layout of the display. Pixels may need to be reoriented from logical to physical space before being rendered.



Pixels may also need to be flipped (mirrored) before being rendered.



Alpha Blending

Each pixel drawn is a composite of the image color and the background color based on the alpha blend value defined by a global alpha value, the pixels alpha value, or both.

Before alpha blending:



After alpha blending:

Frame Buffer



Color Conversion

The image color format may not be the same as the destination frame buffer. Each pixel must be converted before it is written. In the following example, the image is stored using 24 bits per pixel; however, the frame buffer uses 16 bits per pixel.

Frame Buffer (RGB_565)



Image (RGB_888)

Frame Buffer Write

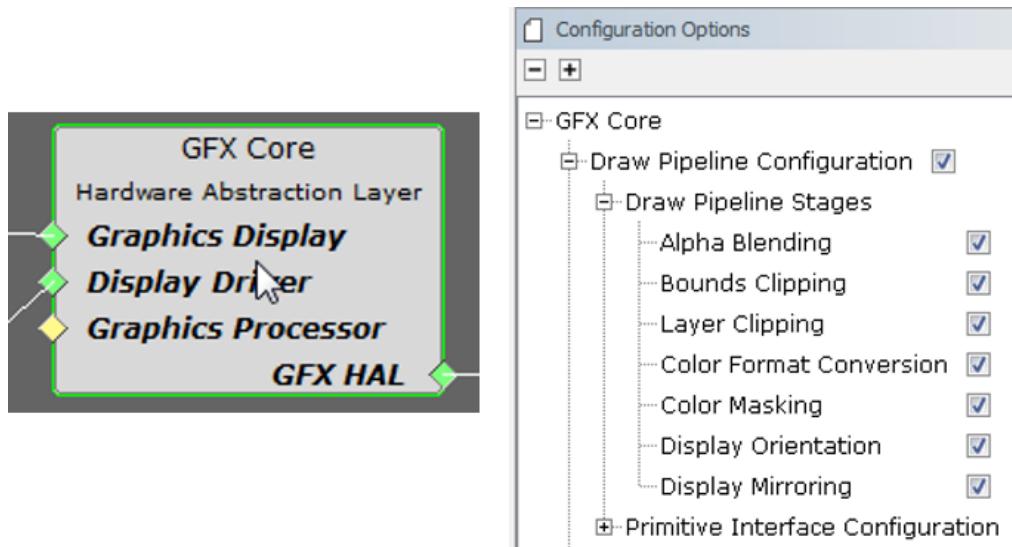
The final stage in rendering an image is to write each-color converted pixel to the frame buffer.

Graphics Pipeline Options

Provides information graphics pipeline options.

Description

Each stage in the graphics pipeline adds overhead to the rendering. Stages can be removed from processing by clicking on the GFX Core component and then disabling stages in the Configuration Options panel belonging to the GFX Core:



For example, the Alpha Blending stage can be disabled if your graphics application does not use alpha blending. If the color mode of the display matches the color mode of all images you can disable Color Conversion. Disabling unneeded stages can improve performance and reduce code size.

Also, a graphics controller driver may add additional stages, or opt to bypass stages completely depending on the capabilities of the graphics hardware supported by the driver.

Improved Touch Performance with Phantom Buttons

This topic provides information on the use of phantom buttons to improve touch performance.

aria_coffeemaker Demonstration Example

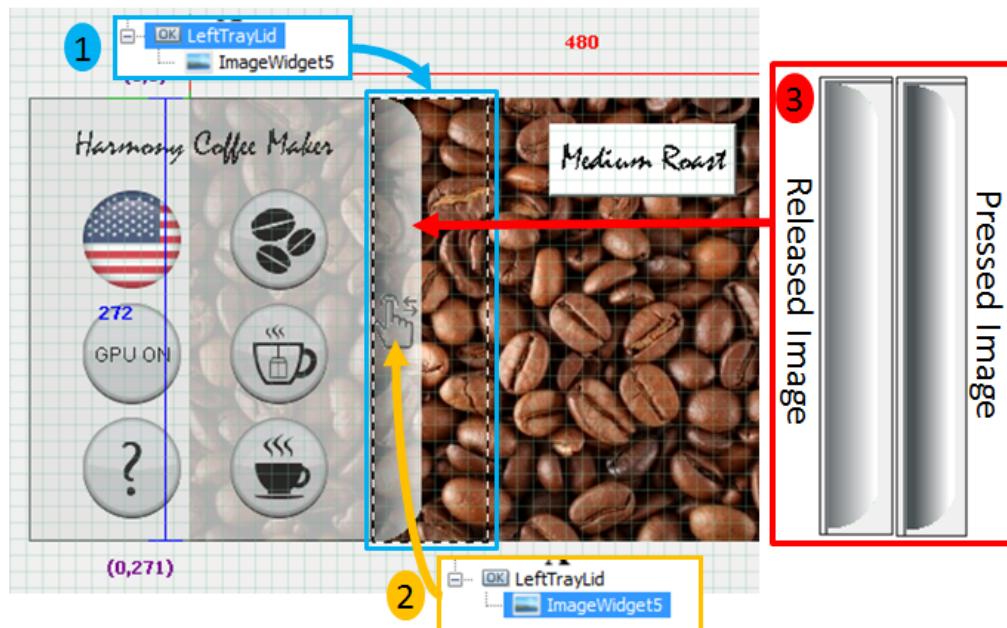
Provides image examples with buttons in the aria_coffeemaker demonstration.

Description

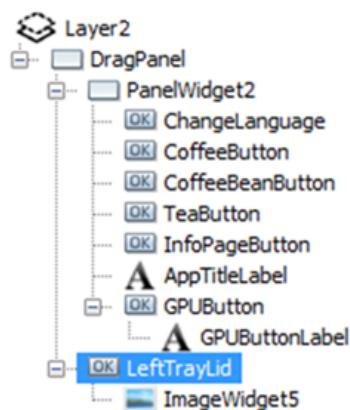
Small buttons are hard to activate on the screen. The use of phantom (invisible) buttons can improve touch performance without increasing the size of the visible footprint of the button on the display.

The aria_coffee_maker has a sliding tray on each side of the display. Sliding a tray in, or out, is accomplished by a phantom (invisible) button. Looking at the left tray, we see the three parts of this phantom button.

1. *LeftTrayLid*: An invisible button widget, whose outline is shown in blue. This area is the touch field.
2. *ImageWidget5*: An image widget containing a hand icon, providing a visual clue as to how to manipulate the tray.
3. *The Release Image and Pressed Image*: These are defined as part of the button widget properties. The Pressed Image has a darker coloring than the Released Image. This difference is what shows the user that the button has been pressed.

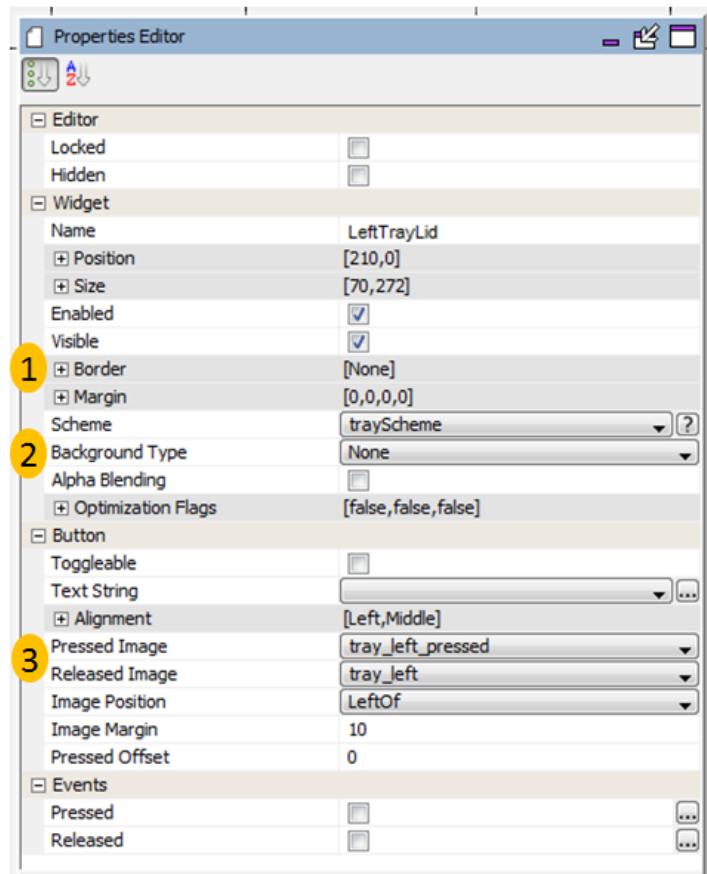


The drawing hierarchy for this part of the design is shows that ImageWidget5 is a daughter widget to the LeftTrayLid button widget.



Examining the properties of the LeftTrayLid button widget reveals more about how this works. The following figure demonstrates these three properties.

1. The *Border* is defined as *None*.
2. *Background Type* is defined as *None*.
3. The different images used will show when the button is *Pressed* or *Released*.



By setting the border and background to *None*, the button is invisible. Only by providing different images for *Released* versus *Pressed* does the user know when the button has been pressed.

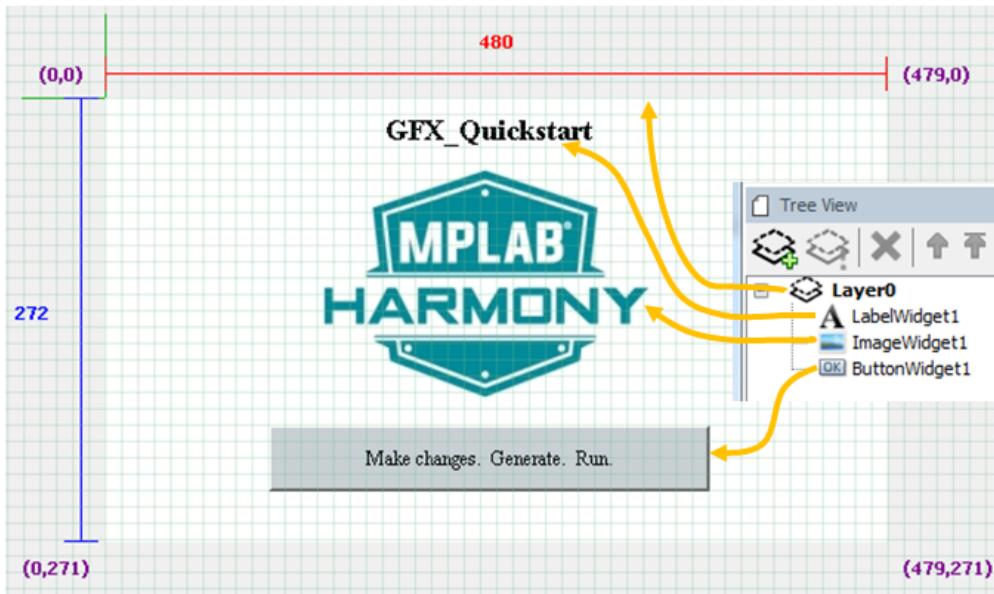
The actual touch region defined by the button is much larger than the images shown on the display. This extra area increases the touch response of the display.

Small Buttons Controlled by Phantom Buttons

Provides information on phantom button control of small buttons.

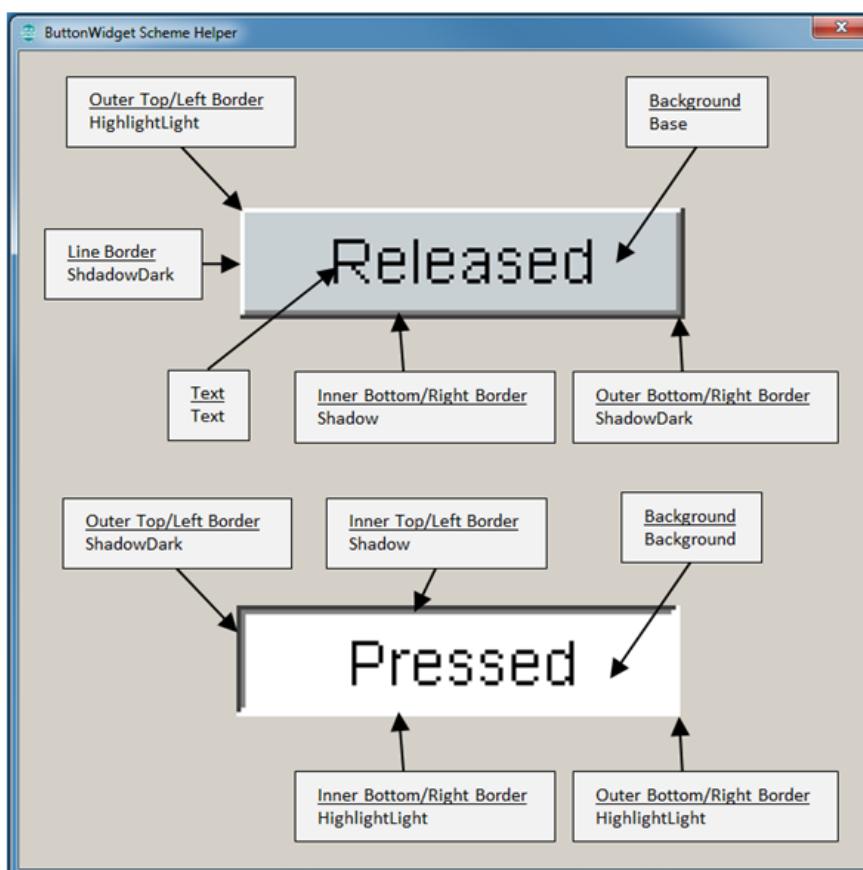
Description

When the border is not set to *None*, and the background is not set to *None*, the button widget provides a direct visible clue to the user when it is pressed. Which can be seen in the following figure with the button from *aria_quickstart*. In *aria_quickstart*, ButtonWidget1 has a bevel border, and a fill background.



Let's use `aria_quickstart` to demonstrate how to control `ButtonWidget1` using a phantom button to surround it, thereby increasing touch responsiveness.

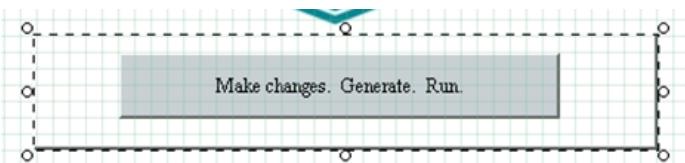
When using a bevel border and filled background, the button provides visible feedback when it is asserted.



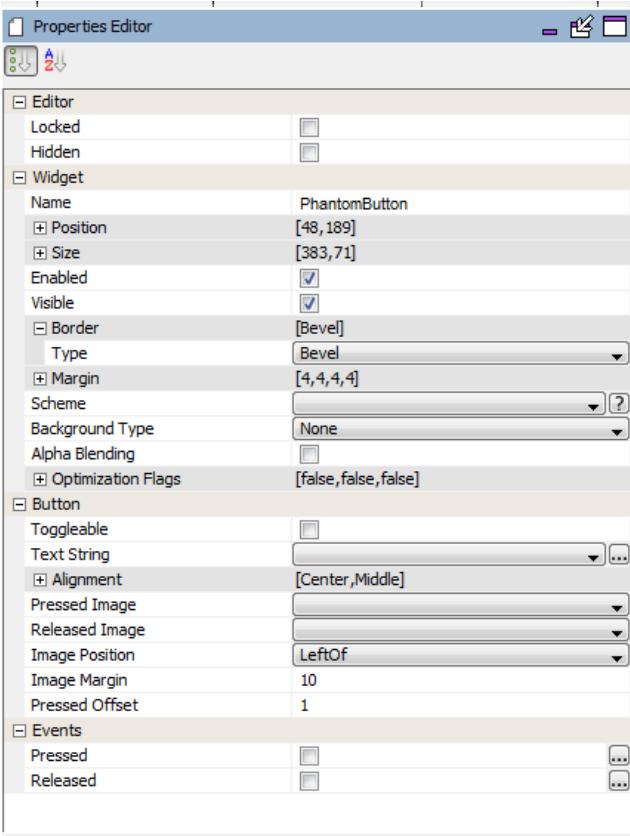
To use this feedback mechanism instead of images, there is a way to have a small button on the display, with a larger touch zone provided by another phantom button.

Steps:

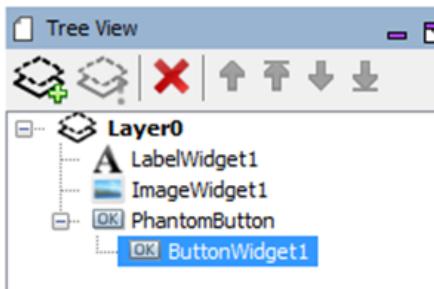
1. Click on `ButtonWidget1` in the *Screen Designer* panel. Go to the *Properties Editor* panel for the widget and uncheck the *Enabled* property to disable the button. Enable *Toggleable* so that this button will have a memory.
2. Drag a new button from the *Widget Tool Box* panel and center it around `ButtonWidget1`. In the *Properties Editor* panel for this new button, change the name of the widget to *PhantomButton*. Change the *Background Type* to *None*. Leave the *Border* set as *Bevel* for now. The following figure displays the new button in the *Screen Designer* panel:



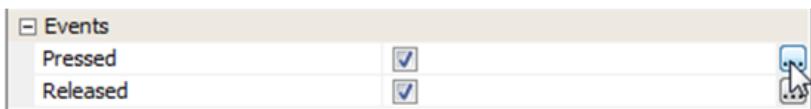
The *Properties Editor* panel should display the following information.



3. In the *Tree View* panel, drag *ButtonWidget1* to be a daughter widget of *PhantomWidget*. When *PhantomWidget* is moved, *ButtonWidget1* will move along with the parent.

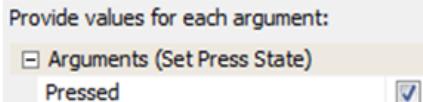


4. Click on *PhantomButton* again in the *Screen Designer panel* and move to the *Properties Editor*. Enable both the *Pressed* and *Released* events. Then click on the (...) icon to define the events. (See the following two steps.)



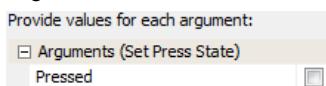
5. Defining the Pressed Event.

Click on the (...) icon. In the *Event Editor*, under *Pressed* dialog, click the *New* icon to define a new event. In the *Action Edit Dialog* that next appears, leave the selection on the template and hit the *Next* button. In the next window, select the target of the event. We want to change the state of *ButtonWidget1*, so select it and hit *Next*. The next dialog shows all the template actions that we can use to modify *ButtonWidget1*. Choose *Set Pressed State* and hit *Next*. Set the Argument to *Enable Pressed*. Name this event *Set Press state for ButtonWidget1* then hit *Finish*. Leave the *Event Editor* by hitting *Ok*.



6. Defining the Released Event.

Click on the (...) icon. In the *Event Editor*, under *Released dialog*, click the *New* icon to define a new event. In the *Action Edit Dialog* that next appears, leave the selection on the template and hit the *Next* button. In the next window, select the target of the event. We want to change the state of ButtonWidget1, so select it and hit *Next*. Choose *Set Pressed State* and hit *Next*. Leave the *Argument* disabled. Name this event *Unset Press state for ButtonWidget1* then hit *Finish*. Leave the *Event Editor* by hitting *Ok*.



7. Generate the application from the MPLAB Harmony Configurator main menu.

8. From the MPLAB main menu, build and run the project. To verify that ButtonWidget1 does change, click outside of the original boundaries.

9. As a final step, hide the PhantomButton by changing its border to *None*. Next, Generate the code again from MHC. Finally, build and run the project from MPLAB and see how much easier it is to assert ButtonWidget1 using a phantom button.

Custom Event Handling and Dynamic Widget Creation

This section illustrates custom event handling and dynamic widget creation in steps by creating the Graphics Events Testbed.

Description

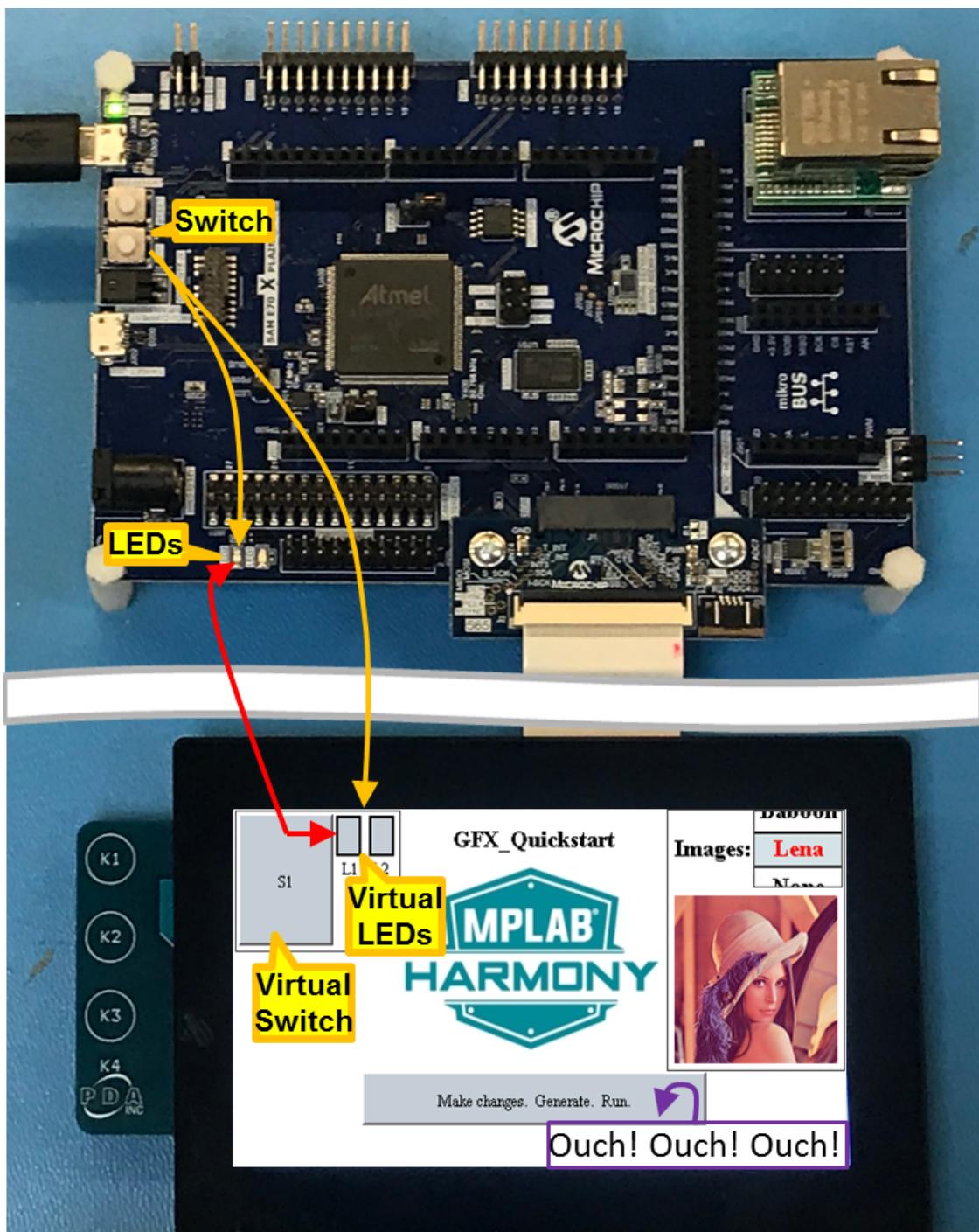
This example is based on the Quick Start Guide “Adding an Event to the Aria Quickstart Demonstration” found in the Graphics Library documentation.

This example has a target configuration for the SAM E7 Xplained Ultra Board with a 4.3” WQVGA Diaplay. For setting up these boards see the instructions for the Aria Quickstart demonstration under Graphics Demonstrations.

This project demonstrates the following events/macros:

Source	Inside of Graphics	Destination	
		Outside of Graphics	
Inside of Graphics	"Event" Button press changes text from "Make Changes. Generate. Run" to "Ouch! Ouch! Ouch!"	"Event" Virtual Switch S1 changes board LEDs via boolean semaphore. List wheel changes image selected via state variable.	
Outside of Graphics	"Macro" APP_Tasks changes color scheme for Virtual LEDs.	<i>Not supported by Event Manager Tool</i> <i>Board S1 changes board LEDs in APP_Tasks</i>	

Asserting the “Make Changes. Generate. Run” button on the display changes its text to “Ouch! Ouch! Ouch!”. Pressing the board’s Switch changes the LEDs on the board as well as changing the virtual LEDs on the display. Pressing the display’s virtual S1 switch does the same. The list wheel changes the image displayed among { None | Baboon | Lena }:



The application's events are defined in `libaria_events.c`:

```
#include "gfx/libaria/libaria_events.h"
// CUSTOM CODE - DO NOT DELETE
#include "app.h"
extern APP_DATA appData;
// END OF CUSTOM CODE

// Display_S1 - PressedEvent
void Display_S1_PressedEvent(laButtonWidget* btn)
{
    // CUSTOM CODE - DO NOT DELETE
    appData.bDisplay_S1State = true;
    // END OF CUSTOM CODE
}
```

```

// Display_S1 - ReleasedEvent
void Display_S1_ReleasedEvent(laButtonWidget* btn)
{
    // CUSTOM CODE - DO NOT DELETE
    appData.bDisplay_S1State = false;
    // END OF CUSTOM CODE
}

// SelectImage - SelectedItemChangedEvent
void SelectImage_SelectedItemChangedEvent(laListWheelWidget* whl, uint32_t idx)
{
    // CUSTOM CODE - DO NOT CHANGE
    // ChangeImage - Set Image - ImageSelected
    switch (idx)
    {
        case 0:
            appData.image_selected = APP_IMAGE_NOIMAGE;
            break;

        case 1:
            appData.image_selected = APP_IMAGE_BABOON;
            break;

        case 2:
            appData.image_selected = APP_IMAGE_LENA;
            break;
    }
    // END OF CUSTOM CODE
}

// ButtonWidget1 - PressedEvent
void ButtonWidget1_PressedEvent(laButtonWidget* btn)
{
    // ButtonDown - Set Text - ButtonWidget1
    laButtonWidget_SetText((laButtonWidget*)ButtonWidget1,
                          laString_CreateFromID(string_OuchOuchOuch));
}

// ButtonWidget1 - ReleasedEvent
void ButtonWidget1_ReleasedEvent(laButtonWidget* btn)
{
    // ButtonUp - Set Text - ButtonWidget1
    laButtonWidget_SetText((laButtonWidget*)ButtonWidget1,
                          laString_CreateFromID(string_Instructions));
}

```

The **SelectImage** list wheel selects which image is shown in the **ImageSelected** image by setting the variable **appData.image_selected**.

The widget **ButtonWidget1** changes its text using the **laButtonWidget_SetText** function. Details on how this is accomplished are discussed in the Quick Start Guide “Adding an Event to the Aria Quickstart Demonstration”.

The **Display_S1** widget just sets a Boolean semaphore **appData.bDisplay_S1State**.

The application's macros are defined in **libaria_macros.c**. They change the coloring scheme for the display's virtual LEDs:

```
#include "gfx/libaria/libaria_macros.h"
```

```

void LEDsTurnOn(void)
{
    if(laContext_GetActiveScreenIndex() != default_ID)
        return;

    // TurnOnDisplayD6 - Set Scheme - SAME70_LED1
    laWidget_SetScheme((laWidget*)SAME70_LED1 &LED1_ON);
    // TurnOnDisplayD7 - Set Scheme - SAME70_LED2
    laWidget_SetScheme((laWidget*) SAME70_LED2, &LED2_ON);
}

```

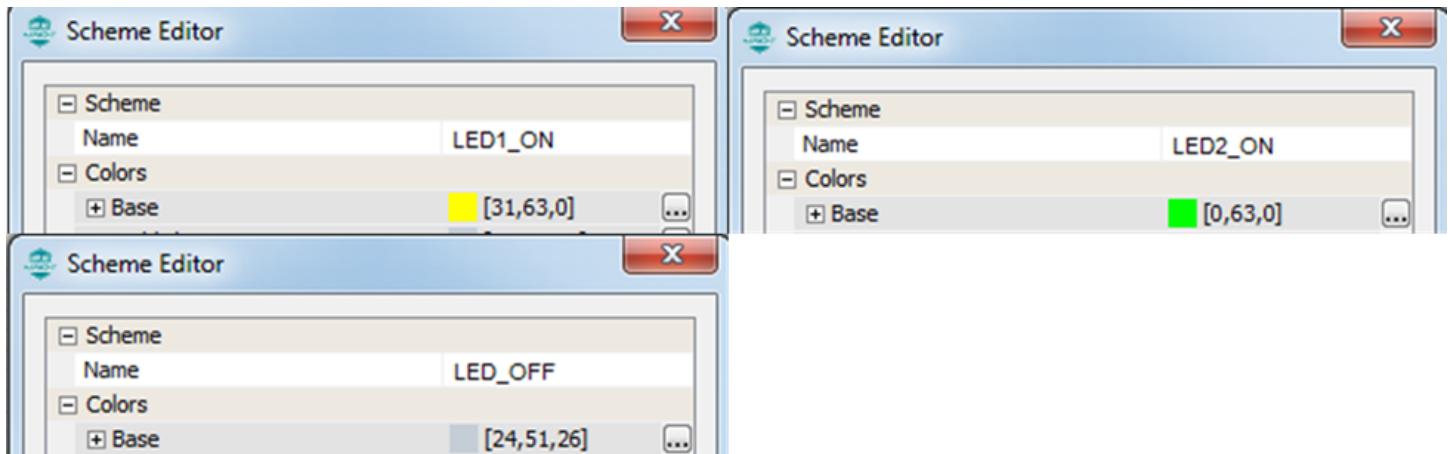
```

void LEDsTurnOff(void)
{
    if(laContext_GetActiveScreenIndex() != default_ID)
        return;

    // TurnOffDisplayD6 - Set Scheme - SAME70_LED1
    laWidget_SetScheme((laWidget*)MEB2_LED_D6, &LED_OFF);
    // TurnOffDisplayD7 - Set Scheme - SAME70_LED2
    laWidget_SetScheme((laWidget*) SAME70_LED2, &LED_OFF);
}

```

The difference among the color schemes is only in the base color:



The macros **LEDsTurnOn** and **LEDsTurnOff** are called from the application's main task loop, **APP_Tasks**. The work of controlling the LEDs is done in the **APP_STATE_SERVICE_TASKS** state:

```

void APP_Image_Tasks( void );

static bool bS1State = false;
static bool bLED_State = false;
static bool bLED_StateNow;

char *msgBuffer = "\r\nApplication created " __DATE__ " " __TIME__
                  " initialized!\r\n";

void APP_Tasks ( void )
{
    /* Check the application's current state. */
    switch ( appData.state )
    {
        /* Application's initial state. */
        case APP_STATE_INIT:
        {
            bool appInitialized = true;
            /* Show Hyperterminal is working using available output functions */
            SYS_CONSOLE_Write(SYS_CONSOLE_INDEX_0, STDOUT_FILENO,
                              msgBuffer, strlen(msgBuffer));
            if (appInitialized)
            {
                appData.state = APP_STATE_SERVICE_TASKS;
            }
            break;
        }

        case APP_STATE_SERVICE_TASKS:
        {
            APP_Image_Tasks(); // Handle images
            bS1State = !SW1_Get(); // Closed --> grounded
            bLED_StateNow = bS1State || appData.bDisplay_S1State; // S1 and virtual S1
            if ( bLED_State != bLED_StateNow )
                // LED state has changed
        }
    }
}

```

```

        if ( bLED_StateNow )
    {
        LED1_On(); // LED1 On
        LED2_On(); // LED2 On
        LEDsTurnOn(); // Turn display LEDs off; // Turn display LEDs on
    }
    else
    {
        LED1_Off(); // LED1 Off
        LED2_Off(); // LED2 Off
        LEDsTurnOff(); // Turn display LEDs off
    } //end if ( bMEB2_S1State || bDisplay_S1State )
    bLED_State = bLED_StateNow; // Remember new state
}
break;
}

default:
{ /* The default state should never be executed. */
    #define UNKNOWN_APP_STATE 0
    assert(UNKNOWN_APP_STATE);
    break;
}

} //end switch ( appData.state )

} //end APP_Tasks

```

The image widget is managed in **APP_Image_Tasks**. This state machine is responsible for the dynamic changing of the image displayed based on the list wheel **Select_Image**.

```

void APP_Image_Tasks( void )
{
    static APP_IMAGE_SELECTED image_selected_old = APP_IMAGE_NOIMAGE;

    if ( laContext_IsDrawing() )
    { // Don't do anything. Wait until all drawing is done.
        return;
    }

    switch ( appData.image_selected )
    {
        case APP_IMAGE_NOIMAGE:
            if ( appData.bImageDrawn )
            { // If image is visible, hide it
                laWidget_SetVisible((laWidget *)ImageSelected,LA_FALSE);
                appData.bImageDrawn = false;
            }
            image_selected_old = APP_IMAGE_NOIMAGE;
            break;

        case APP_IMAGE_BABOON:
            if ( !appData.bImageCreated )
            { // Create image for the first time
                APP_Image_Create();
                appData.bImageCreated = true;
                appData.bImageDrawn = true;
            }
            if ( !appData.bImageDrawn )
            { // Image hidden, make it visible
                laWidget_SetVisible((laWidget *)ImageSelected,LA_TRUE);
                appData.bImageDrawn = true;
            }
            if ( image_selected_old != APP_IMAGE_BABOON )
            { // Select Baboon as the image displayed
                laImageWidget_SetImage((laImageWidget *)ImageSelected,
                                      &Baboon_quartersized);
            image_selected_old = APP_IMAGE_BABOON;

```

```

        }

        break;

    case APP_IMAGE_LENA:
        if ( !appData.bImageCreated )
        { // Create image for the first time
            _APP_Image_Create();
            appData.bImageCreated = true;
            appData.bImageDrawn = true;
        }
        if ( !appData.bImageDrawn )
        { // Image hidden, make it visible
            laWidget_SetVisible((laWidget *)ImageSelected, LA_TRUE);
            appData.bImageDrawn = true;
        }
        if ( image_selected_old != APP_IMAGE_LENA )
        { // Select Lena as the image displayed
            laImageWidget_SetImage((laImageWidget*)ImageSelected,
                                  &Lena_quartersized);
            image_selected_old = APP_IMAGE_LENA;
        }
        break;

    default:
        #define UNKNOWN_IMAGE_TASK_STATE 0
        assert(UNKNOWN_IMAGE_TASK_STATE);
        break;

    } //end switch ( appData.image_state )

} //end APP_Image_Tasks( void )

```

At boot-up the widget **ImageSelected** has not been created. Only when the image is changed from *None* to *Baboon* or *Lena* is the image created for the first time. Then if the image is changed back to *None* the **APP_Image_Tasks** hides the image by making it invisible.

```

static void _APP_Image_Create( void )
{ // Create the image widget for the first time.
    while ( laContext_IsDrawing() )
    { // Don't do anything. Wait until all drawing is done.
    }

    ImageSelected = laImageWidget_New();
    laWidget_SetPosition((laWidget*)ImageSelected, 6, 66);
    laWidget_SetSize((laWidget*)ImageSelected, 128, 128);
    laWidget_SetBackgroundType((laWidget*)ImageSelected, LA_WIDGET_BACKGROUND_NONE);
    laWidget_SetBorderType((laWidget*)ImageSelected, LA_WIDGET_BORDER_NONE);
    laWidget_AddChild((laWidget*)Images, (laWidget*)ImageSelected);

} //end APP_Image_Create

```

If **laContext_IsDrawing** returns true, then neither **APP_Image_Tasks** or **_APP_Image_Create** runs. This is to prevent conflicts in the redraws by the graphics library.

Importing and Exporting Graphics Data

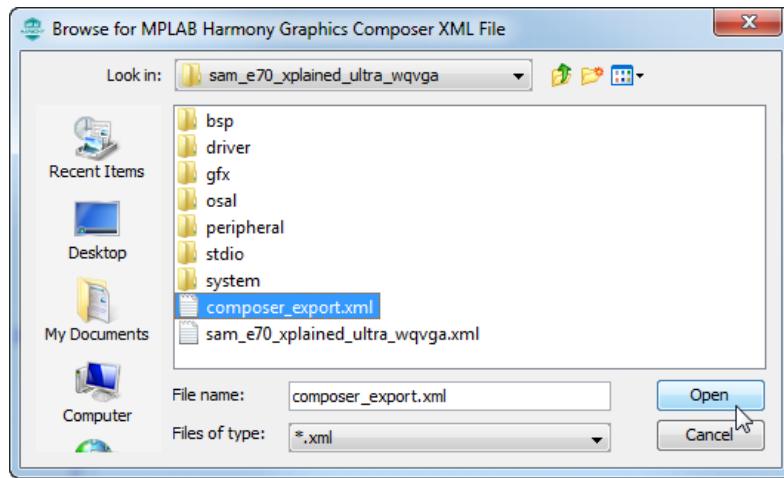
This topic provides information on importing and exporting graphics composer-related data.

Description

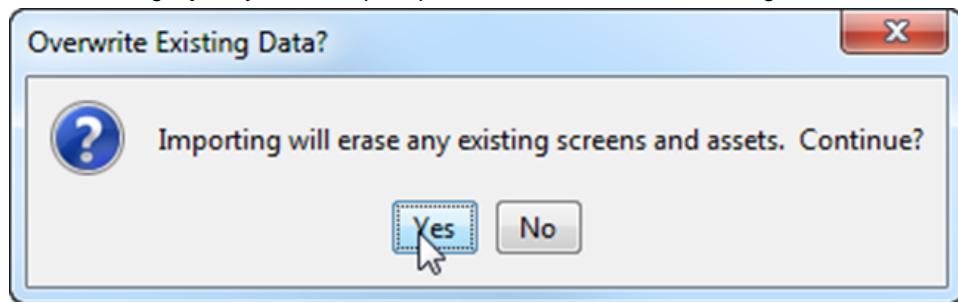
The MPLAB Harmony Graphics Composer (MHGC) provides the capability for users to import and export graphics designs. The user can export the state of an existing graphics composer configuration or import another graphics composer configuration from another project.

Importing Data

1. To import a graphics design into MHGC, select *File > Import*. The Browse for MPLAB Harmony Graphics Composer XML file dialog appears, which allows the selection of a previously exported Graphics Composer .xml file that contains the desired graphics design.

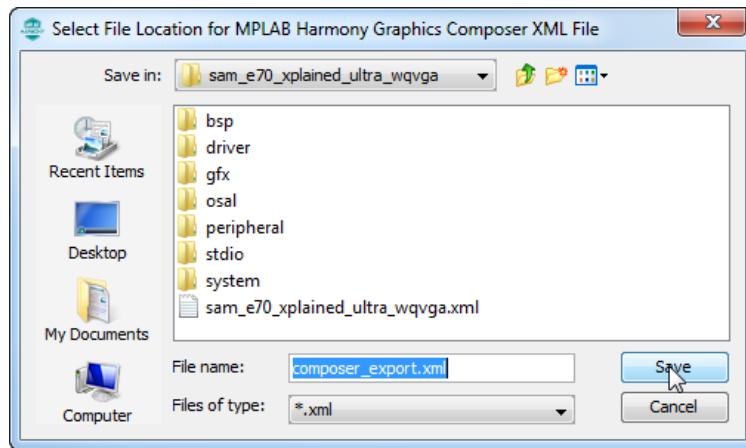


2. After selecting a file and clicking **Open**, you will be prompted whether to overwrite existing data.



Exporting Data

To export a graphics design from MHGC, select *File > Export*. The **Select File Location for MPLAB Harmony Graphics Composer XML** file dialog appears.



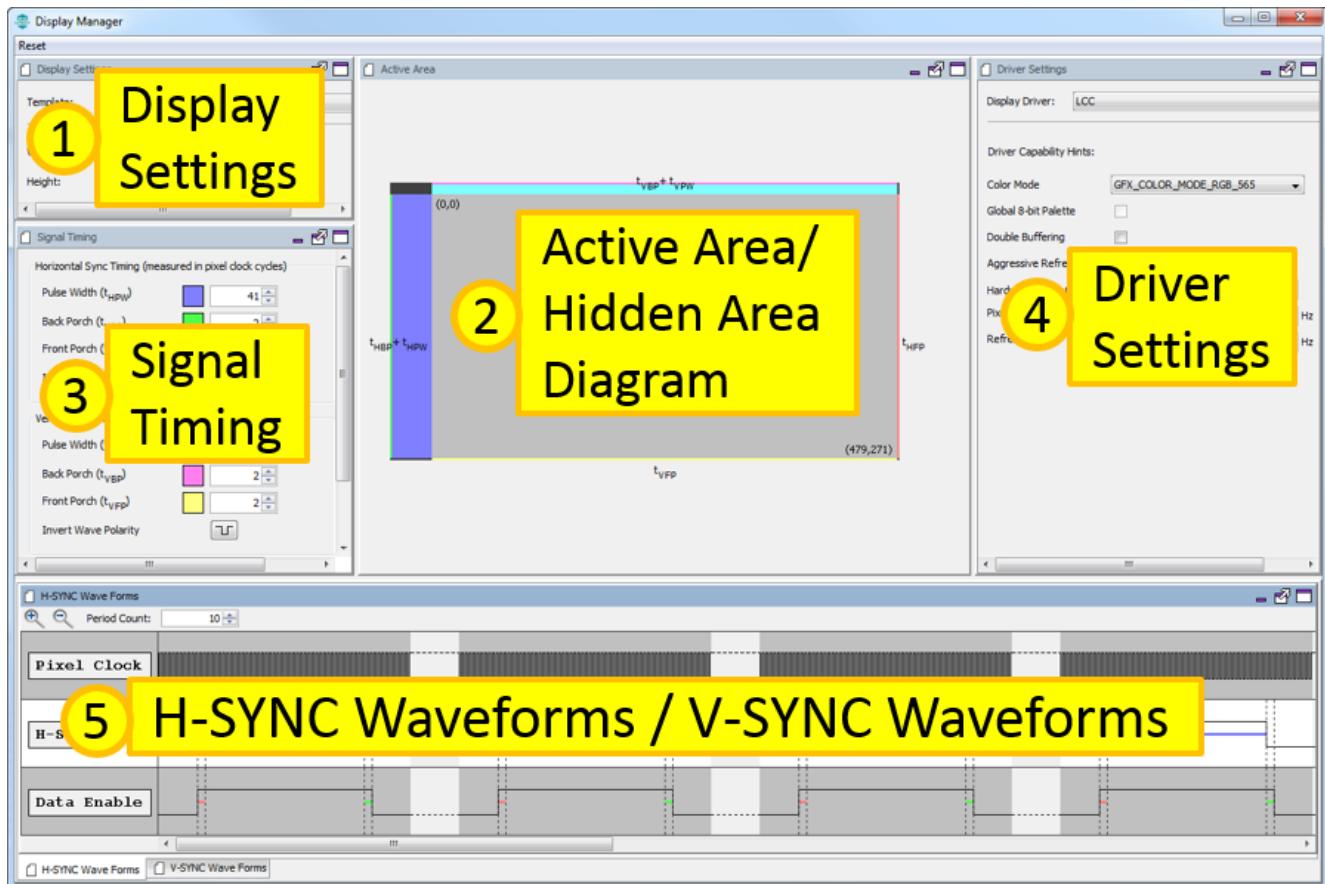
MPLAB Harmony Display Manager User's Guide

Summary

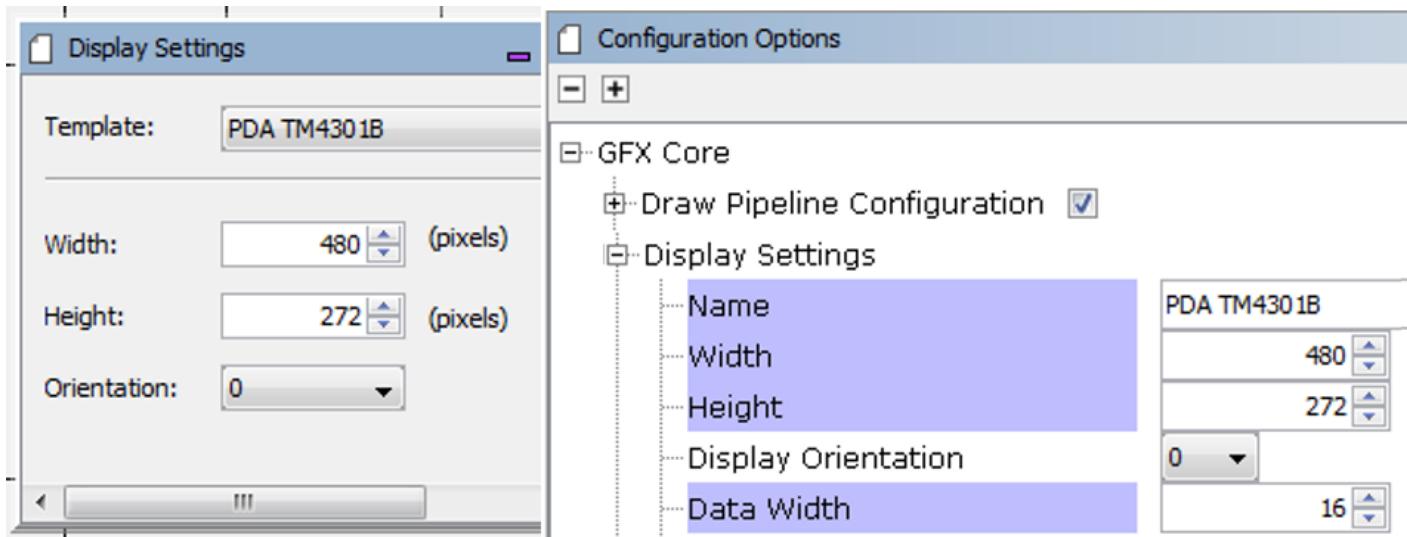
Provides a brief overview of the various features of the MPLAB Harmony Display Manager plug-in.

Description

The following figure illustrates the Display Manager screen. Descriptions for each numbered area follow the figure.



1. **Display Settings** – The Display Settings panel controls the size and orientation of the display. Under the Configuration Options of the GFX Core component is parallel set of display settings.



2. **Active Area / Hidden Area Diagram** – This mimics the display active area transposed over the hidden area as often shown in display data sheets. It also provides a simulation of how the display behaves on the hardware. Note that in Harmony 3

horizontal and vertical syncs are not shown.

3. **Signal Timing** – This panel supports the display timing of the display. These timing values are also shown on the Active Area / Hidden Area Diagram.



These settings are also available in GFX Core Configuration Options:

Configuration Options

- [] +

Display Settings

- Horizontal Attributes
 - Horizontal Pulse Width: 41
 - Horizontal Back Porch: 2
 - Horizontal Front Porch: 2
- Vertical Attributes
 - Vertical Pulse Width: 10
 - Vertical Back Porch: 2
 - Vertical Front Porch: 2
- Inverted Left Shift:

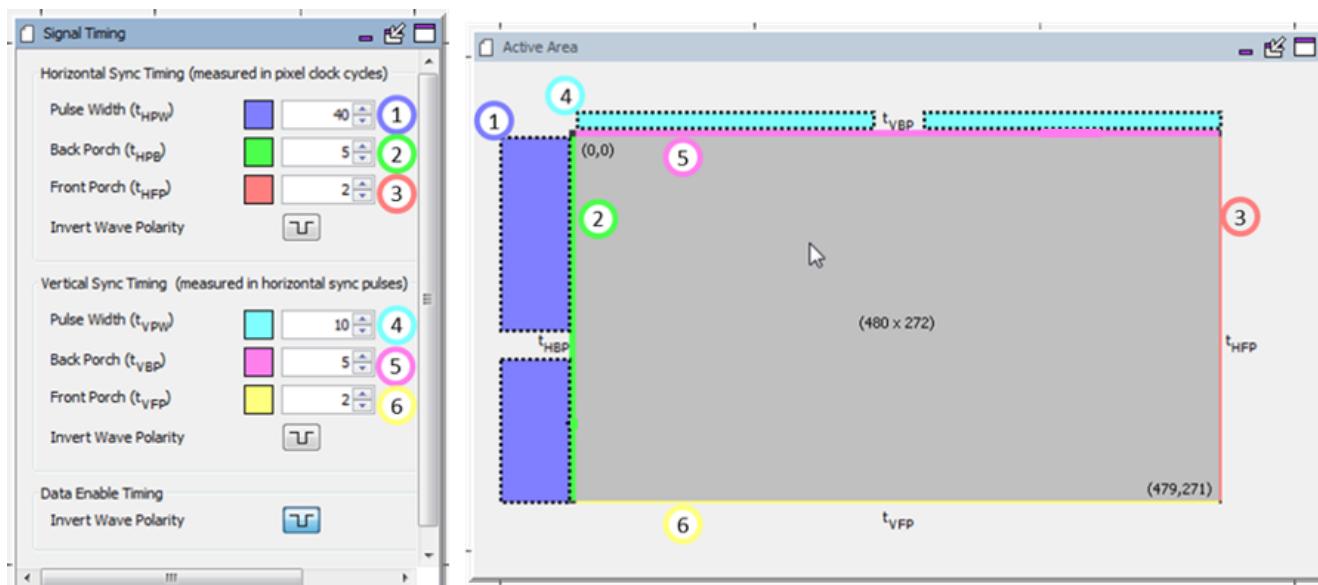
Configuration Options

- [] +

Display Settings

- + Polarity Settings
 - Use Data Enable?
 - Data Enable Polarity Negative?
 - Use Reset?
 - Reset Polarity Positive?
 - Use Chip Select?
 - Chip Select Polarity Positive?

Here is a map between the Display Settings and the Active Area diagram in Display Manager:



4. Driver Settings –This Display Manager panel controls how the graphics driver is setup. There is an equivalent set of controls under GFX Core's Configuration Options > Driver Configuration Hints.

The screenshot shows two main panels: **Driver Settings** and **Configuration Options**.

Driver Settings Panel:

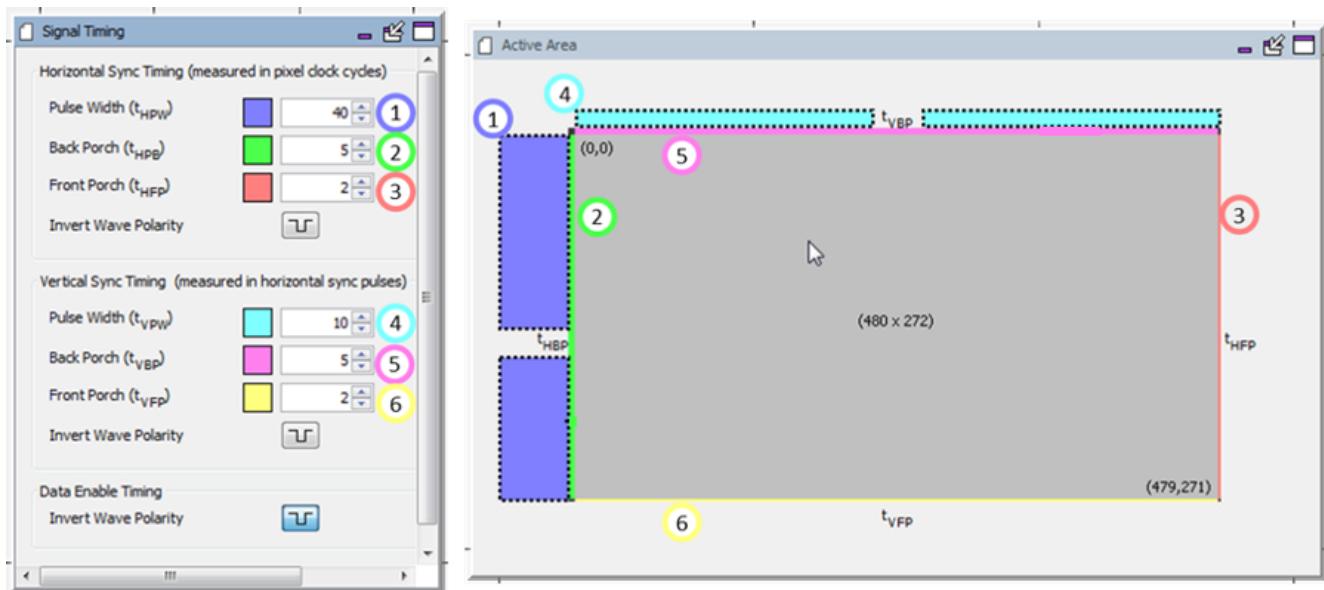
- Display Driver: LCC
- Driver Capability Hints:
 - Color Mode: GFX_COLOR_MODE_RGB_565
 - Global 8-bit Palette: (checkbox)
 - Double Buffering: (checkbox)
 - Aggressive Refresh: (checkbox)
 - Hardware Layer Count: 1
 - Pixel Clock: 6,250,000 Hz
 - Refresh Rate: 41 Hz

Configuration Options Panel:

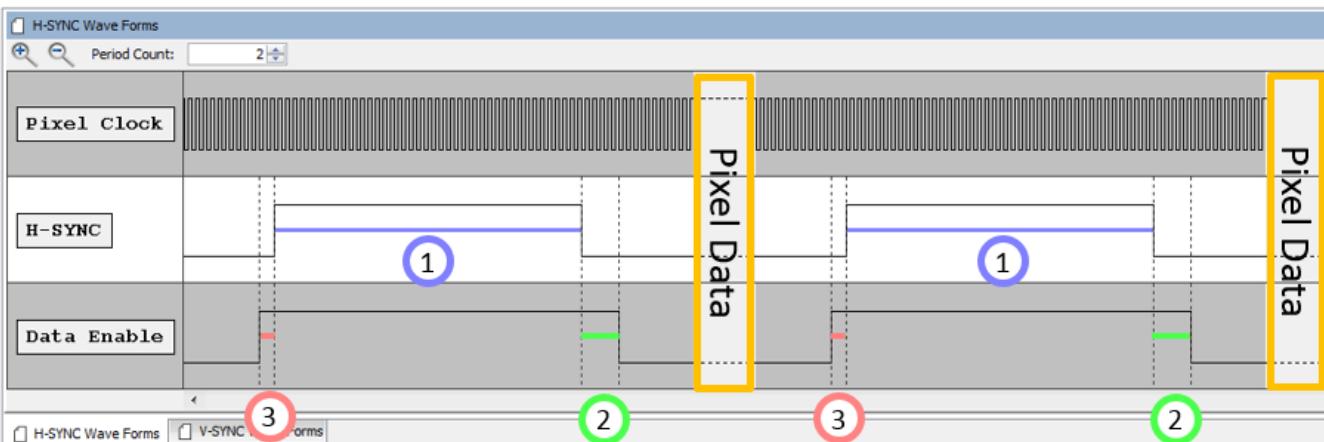
- GFX Core:**
 - Draw Pipeline Configuration: (checkbox, checked)
 - Display Settings:
 - Name: PDA TM4301B
 - Width: 480
 - Height: 272
 - Display Orientation: 0
 - Data Width: 16
 - Pixel Clock (Hz): 6,250,000
 - Refresh Rate (Hz): 41
 - Horizontal Attributes
- Graphics Processor Information:**
 - Driver Configuration Hints:
 - Display Color Mode: GFX_COLOR_MODE_RGB_565
 - Global Palette Mode: (checkbox)
 - Double Buffer Mode: (checkbox)
 - Aggressive Refresh: (checkbox)
 - Hardware Layer Count: 1

The pixel clock for the E70 LCC driver is fixed at about 45 MHz. This is because the master clock that feeds the DMA is shared with the other peripherals. Since the setup is fixed it not shown in the Driver Settings panel.

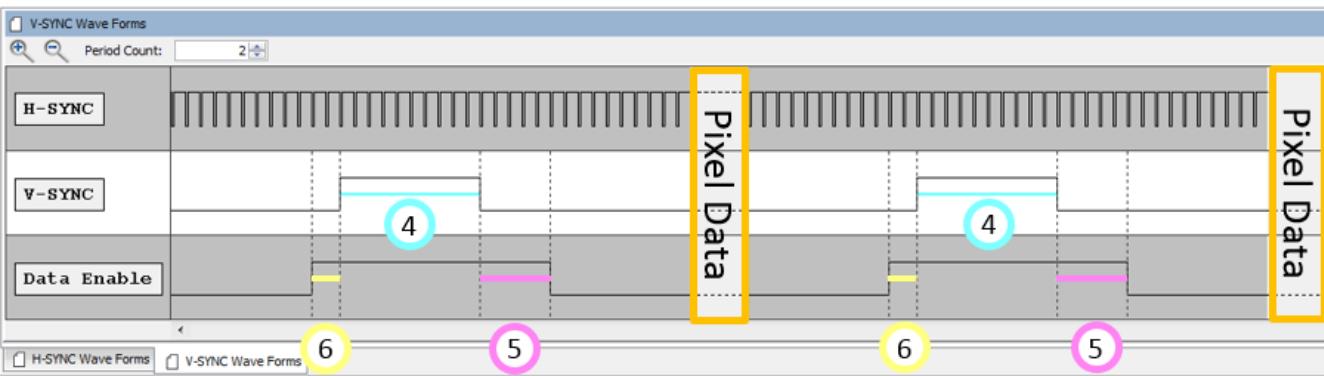
5. **H-SYNC Waveforms / V-SYNC Waveforms** – This panel provides timing diagrams for the horizontal and vertical sync signals. Here is a repeat of the figure shown in 3 above with the back porches set to 5 to distinguish between front and back porches:



The corresponding Horizontal Sync waveforms are:



The corresponding Vertical Sync waveforms are:



Graphics Library Help

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

Currently these libraries consist of:

- MPLAB Harmony Graphics Composer Suite
- ILI9488 Display Controller Driver Library

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 32-bit 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](#).

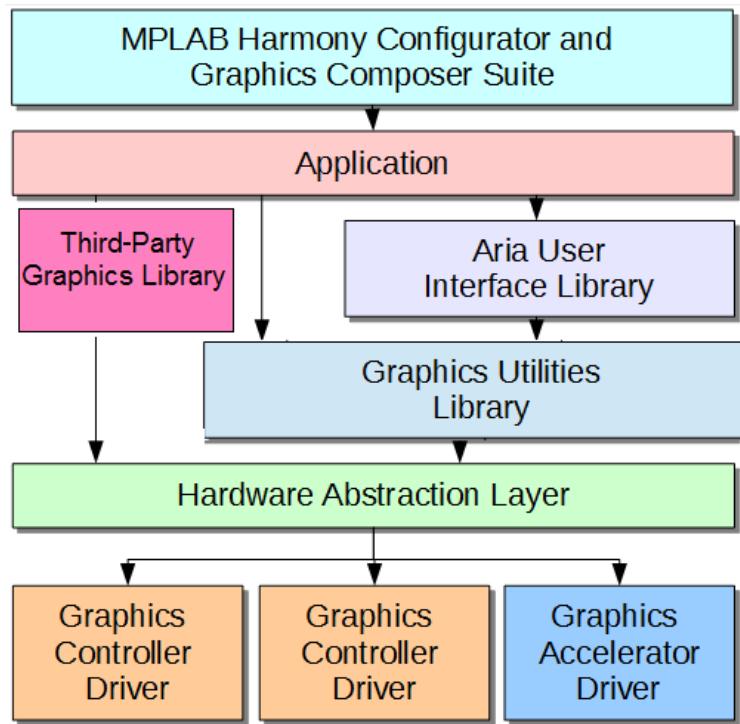
The MPLAB Harmony Graphics Stack consists of several layers that directly build on the capabilities of lower layers to provide a robust framework for displaying rich graphics on supported display devices. Higher layers can be removed as needed if their capabilities are not required.

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 the 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 in <install-directory>/gfx.

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

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 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 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.

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.

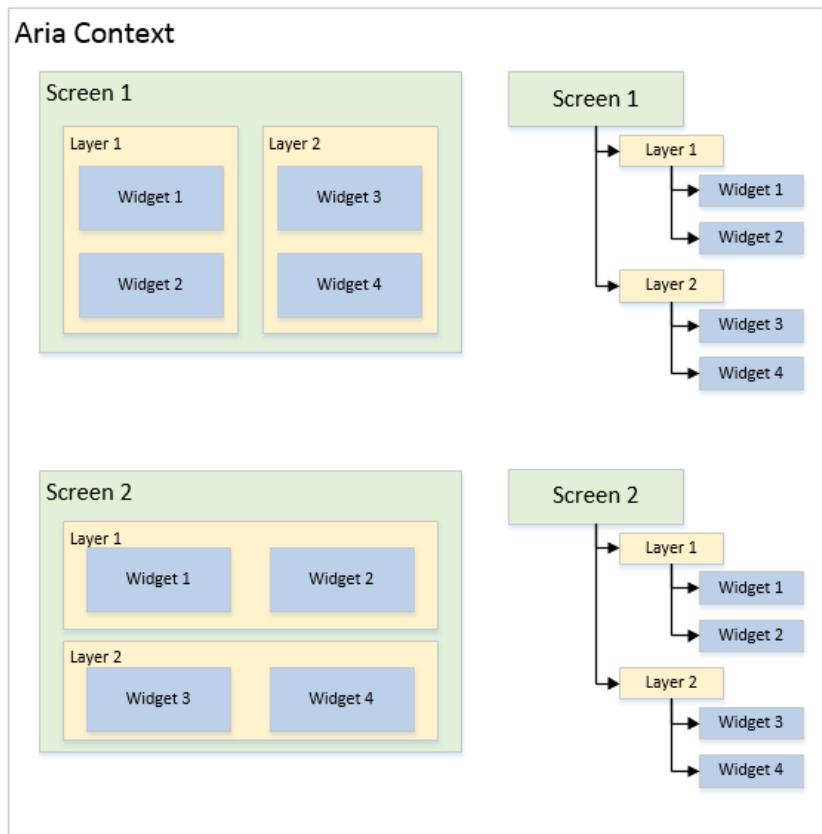
Heterogeneous Space

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 are 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 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: “`IaWidget_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 “`IaWidget_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
≡◊	laArcWidget_GetCenterAngle	Gets the center angle of the arc widget
≡◊	laArcWidget_GetRadius	Gets the radius of a arc widget
≡◊	laArcWidget_GetRoundEdge	Returns true if the arc has round edges
≡◊	laArcWidget_GetStartAngle	Returns the start angle of a arc widget
≡◊	laArcWidget_GetThickness	Gets the thickness of the arc
≡◊	laArcWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	laArcWidget_SetCenterAngle	Sets the center angle of the arc widget
≡◊	laArcWidget_SetRadius	Sets the radius of a arc widget
≡◊	laArcWidget_SetRoundEdge	Sets the arc edge to round
≡◊	laArcWidget_SetStartAngle	Sets the start angle of a arc widget
≡◊	laArcWidget_SetThickness	Sets the thickness of the arc widget
≡◊	laBarGraphWidget_AddCategory	Adds a category to the graph

<code>laBarGraphWidget_AddDataToSeries</code>	Adds a data (value) to the specified series at categoryID index
<code>laBarGraphWidget_AddSeries</code>	Adds a series to the graph
<code>laBarGraphWidget_DestroyAll</code>	Destroys data, series and categories and frees the memory allocated
<code>laBarGraphWidget_GetCategoryAxisLabelsVisible</code>	Returns GFX_TRUE if the category axis labels are visible
<code>laBarGraphWidget_GetCategoryAxisTicksPosition</code>	Returns the position of the ticks in the category axis
<code>laBarGraphWidget_GetCategoryAxisTicksVisible</code>	Returns GFX_TRUE if the category axis ticks are visible
<code>laBarGraphWidget_GetCategoryText</code>	Gets a copy of the string used to label the category
<code>laBarGraphWidget_GetFillGraphArea</code>	Returns GFX_TRUE if the category axis labels are visible
<code>laBarGraphWidget_GetGridlinesVisible</code>	Returns GFX_TRUE if the axis gridlines are visible
<code>laBarGraphWidget_GetMaxValue</code>	Returns the max value of the axis
<code>laBarGraphWidget_GetMinValue</code>	Returns the min value of the axis
<code>laBarGraphWidget_GetSeriesScheme</code>	Returns scheme of the specified series
<code>laBarGraphWidget_GetStacked</code>	Returns GFX_TRUE if the bars are stacked
<code>laBarGraphWidget_GetTickLength</code>	Returns the length of the ticks
<code>laBarGraphWidget_GetValueAxisLabelsVisible</code>	Returns GFX_TRUE if the value axis labels are visible
<code>laBarGraphWidget_GetValueAxisSubtickInterval</code>	Returns the interval between minor ticks in the value axis
<code>laBarGraphWidget_GetValueAxisSubticksPosition</code>	Returns the position of the subticks in the axis
<code>laBarGraphWidget_GetValueAxisSubticksVisible</code>	Returns GFX_TRUE if the value axis subticks are visible
<code>laBarGraphWidget_GetValueAxisTickInterval</code>	Returns the interval between major ticks in the value axis
<code>laBarGraphWidget_GetValueAxisTicksPosition</code>	Returns the position of the ticks in the axis
<code>laBarGraphWidget_GetValueAxisTicksVisible</code>	Returns GFX_TRUE if the value axis ticks are visible
<code>laBarGraphWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laBarGraphWidget_SetCategoryAxisLabelsVisible</code>	Shows/Hides the category axis labels
<code>laBarGraphWidget_SetCategoryAxisTicksPosition</code>	Sets the position of the ticks in the category axis
<code>laBarGraphWidget_SetCategoryAxisTicksVisible</code>	Shows/Hides the category axis ticks
<code>laBarGraphWidget_SetCategoryText</code>	Sets the string used to label the category
<code>laBarGraphWidget_SetDataInSeries</code>	Sets the value of the entry in the series index. The entry should have been previously
<code>laBarGraphWidget_SetFillGraphArea</code>	Sets the graph area filled or not
<code>laBarGraphWidget_SetGridlinesVisible</code>	Shows/Hides the gridlines
<code>laBarGraphWidget_SetMaxValue</code>	Sets the max value of the axis
<code>laBarGraphWidget_SetMinValue</code>	Sets the min value of the axis
<code>laBarGraphWidget_SetSeriesScheme</code>	Sets the color scheme of the series
<code>laBarGraphWidget_SetStacked</code>	Stacks the bar graph
<code>laBarGraphWidget_SetStringTable</code>	Sets the string table used for the generated axis labels
<code>laBarGraphWidget_SetTickLength</code>	Sets the length of the ticks
<code>laBarGraphWidget_SetTicksLabelsStringID</code>	Sets the ID of the superset string used for the value axis tick labels

<code>laBarGraphWidget_SetValueAxisLabelsVisible</code>	Shows/Hides the labels in the value axis
<code>laBarGraphWidget_SetValueAxisSubtickInterval</code>	Sets the minor tick interval in the value axis
<code>laBarGraphWidget_SetValueAxisSubticksPosition</code>	Sets the position of the subticks in the value axis
<code>laBarGraphWidget_SetValueAxisSubticksVisible</code>	Shows/Hides the subticks in the value axis
<code>laBarGraphWidget_SetValueAxisTickInterval</code>	Sets the tick interval in the value axis
<code>laBarGraphWidget_SetValueAxisTicksPosition</code>	Sets the position of the ticks in the value axis
<code>laBarGraphWidget_SetValueAxisTicksVisible</code>	Shows/Hides the ticks in the value axis
<code>laButtonWidget_GetHAlignment</code>	Gets the horizontal alignment setting for a button
<code>laButtonWidget_GetImageMargin</code>	Gets the distance between the icon and the text
<code>laButtonWidget_GetImagePosition</code>	Gets the position of the button icon
<code>laButtonWidget_GetPressed</code>	Gets the pressed state of a button
<code>laButtonWidget_GetPressedEventCallback</code>	Gets the callback associated with the button pressed event
<code>laButtonWidget_GetPressedImage</code>	Gets the pressed image asset pointer for a button
<code>laButtonWidget_GetPressedOffset</code>	Gets the offset of the button internals when pressed
<code>laButtonWidget_GetReleasedEventCallback</code>	Gets the callback for the button released event
<code>laButtonWidget_GetReleasedImage</code>	Gets the currently used released icon
<code>laButtonWidget_GetText</code>	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.
<code>laButtonWidget_GetTextLineSpace</code>	Returns the line spacing in pixels for the button text. If < 0, the ascent value of the string's font is used.
<code>laButtonWidget_GetToggleable</code>	Gets the value of this button's toggle flag
<code>laButtonWidget_GetVAlignment</code>	Gets the vertical alignment setting for a button
<code>laButtonWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laButtonWidget_SetHAlignment</code>	Sets the horizontal alignment value for a button
<code>laButtonWidget_SetImageMargin</code>	Sets the distance between the icon and text
<code>laButtonWidget_SetImagePosition</code>	Sets the position of the button icon
<code>laButtonWidget_SetPressed</code>	Sets the pressed state for a button.
<code>laButtonWidget_SetPressedEventCallback</code>	Sets the pressed event callback for the button
<code>laButtonWidget_SetPressedImage</code>	Sets the image to be used as a pressed icon
<code>laButtonWidget_SetPressedOffset</code>	Sets the offset of the button internals when pressed
<code>laButtonWidget_SetReleasedEventCallback</code>	Sets the callback for the button released event
<code>laButtonWidget_SetReleasedImage</code>	Sets the image to be used as the released icon
<code>laButtonWidget_SetText</code>	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.
<code>laButtonWidget_SetTextLineSpace</code>	Sets the line space in pixels of the text in the button widget. A value < 0 sets the spacing to the ascent value of the string's font.

	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_SetVAlignment	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
	laCircleWidget_GetFilled	Gets the filled state of a circle widget
	laCircleWidget_GetOrigin	Gets the origin coordiates of a circle widget
	laCircleWidget_GetRadius	Gets the radius of a circle widget
	laCircleWidget_GetThickness	Gets the thickness 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_SetFilled	Sets the filled state of a circle widget
	laCircleWidget_SetOrigin	Sets the origin coordiates of a circle widget
	laCircleWidget_SetRadius	Sets the radius of a circle widget
	laCircleWidget_SetThickness	Sets the thickness of a circle widget
	laCircularGaugeWidget_AddAngularArc	Adds an 'angular arc' in the gauge.
	laCircularGaugeWidget_AddMinorTickLabels	Adds minor tick labels in the gauge.
	laCircularGaugeWidget_AddMinorTicks	Adds minor ticks in the gauge.
	laCircularGaugeWidget_AddValueArc	Adds a 'value arc' in the gauge.
	laCircularGaugeWidget_DeleteArcs	Deletes all arcs in the gauge widget
	laCircularGaugeWidget_DeleteMinorTickLabels	Deletes all the minor tick labels in the gauge widget

<code>=♀ laCircularGaugeWidget_DeleteMinorTicks</code>	Deletes all the minor ticks in the gauge widget
<code>=♀ laCircularGaugeWidget_GetCenterAngle</code>	Returns the center angle of the circular gauge
<code>=♀ laCircularGaugeWidget_GetCenterCircleRadius</code>	Returns radius of the center circle
<code>=♀ laCircularGaugeWidget_GetCenterCircleThickness</code>	Returns thickness of the center circle
<code>=♀ laCircularGaugeWidget_GetCenterCircleVisible</code>	Returns GFX_TRUE if the center circle is visible
<code>=♀ laCircularGaugeWidget_GetDirection</code>	Returns the direction of the gauge.
<code>=♀ laCircularGaugeWidget_GetEndValue</code>	Returns the end value of the gauge
<code>=♀ laCircularGaugeWidget_GetHandRadius</code>	Returns the radius/length of the gauge hand in pixels
<code>=♀ laCircularGaugeWidget_GetHandVisible</code>	Returns GFX_TRUE if the gauge hand is visible
<code>=♀ laCircularGaugeWidget_GetRadius</code>	Gets the radius of a gauge widget
<code>=♀ laCircularGaugeWidget_GetStartAngle</code>	Returns the start angle of the circular gauge
<code>=♀ laCircularGaugeWidget_GetStartValue</code>	Returns the start value of the gauge
<code>=♀ laCircularGaugeWidget_GetTickLabelsVisible</code>	Returns GFX_TRUE if the tick labels are visible
<code>=♀ laCircularGaugeWidget_GetTickLength</code>	Returns the length of the ticks in the gauge in pixels
<code>=♀ laCircularGaugeWidget_GetTicksVisible</code>	Returns GFX_TRUE if the ticks in the gauge are visible
<code>=♀ laCircularGaugeWidget_GetTickCount</code>	Returns the tick increment value in the gauge
<code>=♀ laCircularGaugeWidget_GetValue</code>	Returns the value of the gauge hand
<code>=♀ laCircularGaugeWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>=♀ laCircularGaugeWidget_SetCenterAngle</code>	Sets the center angle of the gauge.
<code>=♀ laCircularGaugeWidget_SetCenterCircleRadius</code>	Sets the center radius of the center circle
<code>=♀ laCircularGaugeWidget_SetCenterCircleThickness</code>	Sets the thickness of the center circle
<code>=♀ laCircularGaugeWidget_SetCenterCircleVisible</code>	Sets the center circle visible/invisible
<code>=♀ laCircularGaugeWidget_SetEndValue</code>	Sets the end value of the gauge
<code>=♀ laCircularGaugeWidget_SetHandRadius</code>	Sets the radius/length of the hand
<code>=♀ laCircularGaugeWidget_SetHandVisible</code>	Sets the hand visible/invisible
<code>=♀ laCircularGaugeWidget_SetRadius</code>	Sets the radius of a gauge widget
<code>=♀ laCircularGaugeWidget_SetStartAngle</code>	Sets the start angle of the gauge.
<code>=♀ laCircularGaugeWidget_SetStartValue</code>	Sets the start value of the gauge
<code>=♀ laCircularGaugeWidget_SetStringTable</code>	Sets the string table to be used for the tick labels
<code>=♀ laCircularGaugeWidget_SetTickLabelsVisible</code>	Sets the tick labels visible/invisible
<code>=♀ laCircularGaugeWidget_SetTickLength</code>	Sets the length of the ticks
<code>=♀ laCircularGaugeWidget_SetTicksLabelsStringID</code>	Sets the ID of the string character superset to be used for the tick labels
<code>=♀ laCircularGaugeWidget_SetTicksVisible</code>	Sets the increments/distance between ticks
<code>=♀ laCircularGaugeWidget_SetTickCount</code>	Sets the increments/distance between ticks
<code>=♀ laCircularGaugeWidget_SetValue</code>	Sets the value of the gauge hand
<code>=♀ laCircularGaugeWidget_SetValueChangedEventCallback</code>	Sets the function to be called back when the gauge value changes.
<code>=♀ laCircularSliderWidget_GetArcRadius</code>	Returns the radius of an arc in the slider widget
<code>=♀ laCircularSliderWidget_GetArcScheme</code>	Returns the scheme of an arc in the slider widget
<code>=♀ laCircularSliderWidget_GetArcThickness</code>	Returns the thickness of an arc in the slider widget
<code>=♀ laCircularSliderWidget_GetArcVisible</code>	Returns true if the specified arc is visible
<code>=♀ laCircularSliderWidget_GetDirection</code>	Returns direction of the slider widget
<code>=♀ laCircularSliderWidget_GetEndValue</code>	Gets the end value of the slider widget
<code>=♀ laCircularSliderWidget_GetOrigin</code>	Gets the origin coordinates of a slider widget

<code>laCircularSliderWidget_GetRadius</code>	Gets the radius of a slider widget
<code>laCircularSliderWidget_GetRoundEdges</code>	Returns true if the slider has rounded edges
<code>laCircularSliderWidget_GetStartAngle</code>	Returns the start angle of a slider widget
<code>laCircularSliderWidget_GetStartValue</code>	Gets the start value of the slider widget
<code>laCircularSliderWidget_GetStickyButton</code>	Returns true if the slider button sticks to the start/end value
<code>laCircularSliderWidget_GetTouchOnButtonOnly</code>	Returns true if the slider slider only responds to touch inside the button area
<code>laCircularSliderWidget_GetValue</code>	Gets the value of the slider widget
<code>laCircularSliderWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laCircularSliderWidget_SetArcRadius</code>	Sets the start value of the slider widget
<code>laCircularSliderWidget_SetArcScheme</code>	Sets the scheme of the specified arc
<code>laCircularSliderWidget_SetArcThickness</code>	Sets the thickness of an arc in the slider widget
<code>laCircularSliderWidget_SetArcVisible</code>	Shows/Hides the specified arc visible
<code>laCircularSliderWidget_SetDirection</code>	Sets the direction of the slider widget
<code>laCircularSliderWidget_SetEndValue</code>	Sets the end value of the slider widget
<code>laCircularSliderWidget_SetOrigin</code>	Sets the origin coordinates of a slider widget
<code>laCircularSliderWidget_SetPressedEventCallback</code>	Sets the function that gets called when the slider button is pressed
<code>laCircularSliderWidget_SetRadius</code>	Sets the radius of a slider widget
<code>laCircularSliderWidget_SetReleasedEventCallback</code>	Sets the function that gets called when the slider button is released
<code>laCircularSliderWidget_SetRoundEdges</code>	If round = true, the slider active area edges are round
<code>laCircularSliderWidget_SetStartAngle</code>	Sets the start angle of a slider widget
<code>laCircularSliderWidget_SetStartValue</code>	Sets the start value of the slider widget
<code>laCircularSliderWidget_SetStickyButton</code>	If snap = true, the slider button sticks to the start/end value of the slider
<code>laCircularSliderWidget_SetTouchOnButtonOnly</code>	If buttonOnly = true, the slider will only respond to touches inside the button area
<code>laCircularSliderWidget_SetValue</code>	Sets the value of the slider widget
<code>laCircularSliderWidget_SetValueChangedEventCallback</code>	Sets the function that gets called when the slider value changes
<code>laContext_AddScreen</code>	Add screen to the list of screens in the current context
<code>laContext_Create</code>	Creates an instance of the Aria user interface library
<code>laContext_Destroy</code>	Destroys an Aria instance
<code>laContext_GetActive</code>	Returns the current active context.
<code>laContext_GetActiveScreen</code>	Returns the active screen of the current context
<code>laContext_GetActiveScreenIndex</code>	Return the index of the active screen
<code>laContext_GetColorMode</code>	Returns the color mode of the current context
<code>laContext_GetDefaultScheme</code>	Returns the pointer to the default scheme of the current context
<code>laContext_GetEditWidget</code>	Gets the widget that is currently receiving all widget edit events.
<code>laContext_GetFocusWidget</code>	Return a pointer to the widget in focus
<code>laContext_GetPreemptionLevel</code>	Returns the preemption level for the screen
<code>laContext_GetScreenRect</code>	Returns the display rectangle structure of the physical display

<code>laContext_GetStringLanguage</code>	Returns the language index of the current context
<code>laContext_GetStringTable</code>	Get a pointer to the <code>GFXU_StringTableAsset</code> structure that maintains the strings, associated fonts, etc
<code>laContext_HideActiveScreen</code>	Hide the active screen
<code>laContext_IsDrawing</code>	Indicates if any layers of the active screen are currently drawing a frame.
<code>laContext_IsLayerDrawing</code>	Indicates if the layer at the given index of the active screen is currently drawing.
<code>laContext_RedrawAll</code>	Forces the library to redraw the currently active screen in its entirety.
<code>laContext_RemoveScreen</code>	Remove the specified screen from the list of screens in the current context
<code>laContext_SetActive</code>	Make the specified context active
<code>laContext_SetActiveScreen</code>	Change the active screen to the one specified by the index argument
<code>laContext_SetActiveScreenChangedCallback</code>	Set the callback function pointer when the screen change event occurs
<code>laContext_SetEditWidget</code>	Sets the currently active edit widget.
<code>laContext_SetFocusWidget</code>	Set into focus the widget specified as the argument
<code>laContext_SetLanguageChangedCallback</code>	Set the callback function pointer when the language change event occurs
<code>laContext_SetPreemptionLevel</code>	Set the preemption level to the specified value
<code>laContext_SetStringLanguage</code>	Set the language index of the current context
<code>laContext_SetStringTable</code>	Set the StringTable pointer to the specified new StringTableAsset structure
<code>laContext_Update</code>	Runs the update loop for a library instance.
<code>laDraw_1x2BevelBorder</code>	Internal utility function to draw a 1x2 bevel border
<code>laDraw_2x1BevelBorder</code>	Internal utility function to draw a 2x1 bevel border
<code>laDraw_2x2BevelBorder</code>	Internal utility function to draw a 2x2 bevel border
<code>laDraw_LineBorder</code>	Internal utility function to draw a basic line border
<code>laDrawSurfaceWidget_GetDrawCallback</code>	Returns the pointer to the currently set draw callback.
<code>laDrawSurfaceWidget_New</code>	Allocates memory for a new DrawSurface widget.
<code>laDrawSurfaceWidget_SetDrawCallback</code>	Sets the draw callback pointer for the draw surface widget.
<code>laEvent_AddEvent</code>	Add the mentioned event callback to the list of events maintained by the current context
<code>laEvent_ClearList</code>	Clear the event list maintained by the current context.
<code>laEvent_GetCount</code>	Returns the number of events listed in the current context
<code>laEvent_ProcessEvents</code>	Processes the screen change as well as touch events
<code>laEvent_SetFilter</code>	Set callback pointer for current context filter event
<code>laGradientWidget_GetDirection</code>	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 management 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 alignment 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 management 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.
≡◊	lalImagePlusWidget_GetImage	Gets the image asset pointer for the widget.
≡◊	lalImagePlusWidget_GetInteractive	Returns true if the widget is configured to respond to input events
≡◊	lalImagePlusWidget_GetPreserveAspectEnabled	Returns the boolean value of the 'preserve aspect' property
≡◊	lalImagePlusWidget_GetResizeFilter	Returns the resize filter setting for this image widget
≡◊	lalImagePlusWidget_GetStretchEnabled	Returns the boolean value of the 'stretch to fit' property
≡◊	lalImagePlusWidget_GetTransformHeight	Returns the image scale height coefficient
≡◊	lalImagePlusWidget_GetTransformWidth	Returns the image scale width coefficient
≡◊	lalImagePlusWidget_GetTransformX	Returns the image transform x coefficient
≡◊	lalImagePlusWidget_GetTransformY	Returns the image transform y coefficient
≡◊	lalImagePlusWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	lalImagePlusWidget_ResetTransform	Resets the image transform values to zero
≡◊	lalImagePlusWidget_SetImage	Sets the image asset pointer for the widget.
≡◊	lalImagePlusWidget_SetInteractive	Sets the widget interactive flag
≡◊	lalImagePlusWidget_SetPreserveAspectEnabled	Sets the boolean value of the 'preserve aspect' property
≡◊	lalImagePlusWidget_SetResizeFilter	Sets the resize filter value of the widget
≡◊	lalImagePlusWidget_SetStretchEnabled	Sets the boolean value of the stretch property
≡◊	lalImagePlusWidget_SetTransformHeight	Sets the image scale height coefficient. This value is in pixels not percentage
≡◊	lalImagePlusWidget_SetTransformWidth	Sets the image scale width coefficient. This value is in pixels not percentage
≡◊	lalImagePlusWidget_SetTransformX	Sets the image transform x coefficient
≡◊	lalImagePlusWidget_SetTransformY	Sets the image transform y coefficient
≡◊	lalImageSequenceWidget_GetImage	Gets the image asset pointer for an entry.
≡◊	lalImageSequenceWidget_GetImageChangedEventCallback	Gets the image changed event callback pointer.
≡◊	lalImageSequenceWidget_GetImageCount	Gets the number of image entries for this widget.
≡◊	lalImageSequenceWidget_GetImageDelay	Gets the image delay for an entry.
≡◊	lalImageSequenceWidget_GetImageHAlignment	Gets the horizontal alignment for an image entry
≡◊	lalImageSequenceWidget_GetImageVAlignment	Sets the vertical alignment for an image entry
≡◊	lalImageSequenceWidget_GetRepeat	Indicates if the widget will repeat through the image entries.
≡◊	lalImageSequenceWidget_IsPlaying	Indicates if the widget is currently cycling through the image entries.

<code>lalImageSequenceWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>lalImageSequenceWidget_Play</code>	Starts the widget automatically cycling through the image entries.
<code>lalImageSequenceWidget_Rewind</code>	Resets the current image sequence display index to zero.
<code>lalImageSequenceWidget_SetImage</code>	Sets the image asset pointer for an entry.
<code>lalImageSequenceWidget_SetImageChangedEventCallback</code>	Sets the image changed event callback pointer. This callback is called whenever the active display index is changed.
<code>lalImageSequenceWidget_SetImageCount</code>	Sets the number of image entries for this widget. An image entry that is null will show nothing.
<code>lalImageSequenceWidget_SetImageDelay</code>	Sets the image delay for an entry.
<code>lalImageSequenceWidget_SetImageHAlignment</code>	Sets the horizontal alignment for an image entry.
<code>lalImageSequenceWidget_SetImageVAlignment</code>	Sets the vertical alignment value for an image entry
<code>lalImageSequenceWidget_SetRepeat</code>	Sets the repeat flag for the widget
<code>lalImageSequenceWidget_ShowImage</code>	Sets the active display index to the indicated value.
<code>lalImageSequenceWidget_ShowNextImage</code>	Sets the active display index to the next index value.
<code>lalImageSequenceWidget_ShowPreviousImage</code>	Sets the active display index to the previous index value.
<code>lalImageSequenceWidget_Stop</code>	Stops the widget from automatically cycling through the image entries.
<code>lalImageWidget_GetHAlignment</code>	Gets the image horizontal alignment value.
<code>lalImageWidget_GetImage</code>	Gets the image asset pointer for the widget.
<code>lalImageWidget_GetVAlignment</code>	Gets the image vertical alignment value.
<code>lalImageWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>lalImageWidget_SetHAlignment</code>	Sets the image horizontal alignment value.
<code>lalImageWidget_SetImage</code>	Sets the image asset pointer for the widget.
<code>lalImageWidget_SetVAlignment</code>	Sets the image vertical alignment value.
<code>lalInitialize</code>	Function to perform basic initialization of the Aria library state.
<code>lalInput_GetEnabled</code>	Returns the input enabled status of the current context
<code>lalInput.InjectTouchDown</code>	Register and track the touch down event and queue it for handling by associated widgets
<code>lalInput.InjectTouchMoved</code>	Register and track the touch moved event and queue it for handling by associated widgets
<code>lalInput.InjectTouchUp</code>	Register and track the touch up event and queue it for handling by associated widgets
<code>lalInput_SetEnabled</code>	Sets the input status of the current context with the specified input argument
<code>laKeyPadWidget.GetKeyAction</code>	Gets the key pad cell action for a cell at row/column
<code>laKeyPadWidget.GetKeyClickEventCallback</code>	Gets the current key click event callback pointer
<code>laKeyPadWidget.GetKeyDrawBackground</code>	Gets the background type for a key pad cell at row/column
<code>laKeyPadWidget.GetKeyEnabled</code>	Gets the enabled flag for a cell at a given row/column

<code>=♀ laKeyPadWidget_GetKeyImageMargin</code>	Gets the key pad cell image margin value
<code>=♀ laKeyPadWidget_GetKeyImagePosition</code>	Gets the image position for a key pad cell
<code>=♀ laKeyPadWidget_GetKeyPressedImage</code>	Gets the pressed icon image asset pointer for the display image for a key pad cell
<code>=♀ laKeyPadWidget_GetKeyReleasedImage</code>	Gets the released icon image asset pointer for the display image for a key pad cell
<code>=♀ laKeyPadWidget_GetKeyText</code>	Returns a copy of the display text for a given cell at row/column
<code>=♀ laKeyPadWidget_GetKeyValue</code>	Gets the edit text value for a given key pad cell.
<code>=♀ laKeyPadWidget_New</code>	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.
<code>=♀ laKeyPadWidget_SetKeyAction</code>	Sets the cell action type for a key pad cell at row/column
<code>=♀ laKeyPadWidget_SetKeyBackgroundType</code>	Sets the background type for a key pad cell at row/column
<code>=♀ laKeyPadWidget_SetKeyClickEventCallback</code>	Sets the current key click event callback pointer
<code>=♀ laKeyPadWidget_SetKeyEnabled</code>	Sets the enabled flag for a cell at the given row/column
<code>=♀ laKeyPadWidget_SetKeyImageMargin</code>	Sets the key pad cell image margin value for a given cell at row/column
<code>=♀ laKeyPadWidget_SetKeyImagePosition</code>	
<code>=♀ laKeyPadWidget_SetKeyPressedImage</code>	Sets the pressed icon image asset pointer for a key pad cell
<code>=♀ laKeyPadWidget_SetKeyReleasedImage</code>	Sets the released icon image asset pointer for a key pad cell
<code>=♀ laKeyPadWidget_SetKeyText</code>	Sets the display text for a given cell at row/column
<code>=♀ laKeyPadWidget_SetKeyValue</code>	Sets the edit value for a given key pad cell.
<code>=♀ laLabelWidget_GetHAlignment</code>	Gets the text horizontal alignment value.
<code>=♀ laLabelWidget_GetText</code>	Gets the text value for the label.
<code>=♀ laLabelWidget_GetTextLineSpace</code>	Returns the line spacing in pixels for the label text. If < 0, the ascent value of the string's font is used.
<code>=♀ laLabelWidget_GetVAlignment</code>	Gets the current vertical text alignment
<code>=♀ laLabelWidget_New</code>	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.
<code>=♀ laLabelWidget_SetHAlignment</code>	Sets the text horizontal alignment value
<code>=♀ laLabelWidget_SetText</code>	Sets the text value for the label.
<code>=♀ laLabelWidget_SetTextLineSpace</code>	Sets the line space in pixels of the text in the label widget. A value < 0 sets the spacing to the ascent value of the string's font.
<code>=♀ laLabelWidget_SetVAlignment</code>	Sets the vertical text alignment value
<code>=♀ laLayer_AddDamageRect</code>	Adds a damaged rectangle to the list. Damage rectangles are used in minimal redraw algorithms.
<code>=♀ laLayer_Delete</code>	Destructor for the layer object
<code>=♀ laLayer_GetAllowInputPassThrough</code>	Gets the layer's input passthrough setting
<code>=♀ laLayer_GetAlphaAmount</code>	Get's the amount of alpha blending for a given layer
<code>=♀ laLayer_GetAlphaEnable</code>	Gets the layer alpha enable flag
<code>=♀ laLayer_GetBufferCount</code>	Return the buffer count for the current layer

<code>laLayer_GetEnabled</code>	Returns the boolean value of the layer enabled property
<code>laLayer_GetInputRect</code>	Gets the layer's input rectangle.
<code>laLayer_GetInputRectLocked</code>	Gets the layer's input rect locked flag
<code>laLayer_GetMaskColor</code>	Returns the mask color value for the current layer
<code>laLayer_GetMaskEnable</code>	Gets the layer mask enable flag
<code>laLayer_GetVSync</code>	Gets the layer's vsync flag setting
<code>laLayer_IsDrawing</code>	Queries a layer to find out if it is currently drawing a frame.
<code>laLayer_New</code>	Constructor for a new layer
<code>laLayer_SetAllowInputPassthrough</code>	Sets the layer's input passthrough flag.
<code>laLayer_SetAlphaAmount</code>	Set's the amount of alpha blending for a given layer
<code>laLayer_SetAlphaEnable</code>	Sets the layer alpha enable flag to the specified value
<code>laLayer_SetBufferCount</code>	Set the buffer count for the current layer to the specified value
<code>laLayer_SetEnabled</code>	Sets the boolean value of the layer enabled property
<code>laLayer_SetInputRect</code>	Sets the layer's input rect dimensions.
<code>laLayer_SetInputRectLocked</code>	Sets the layer's input rect locked flag.
<code>laLayer_SetMaskColor</code>	Set the mask color value for the current layer to the specified value
<code>laLayer_SetMaskEnable</code>	Sets the layer mask enable flag to the specified value
<code>laLayer_SetVSync</code>	Sets the layer's vsync flag.
<code>laLineGraphWidget_AddCategory</code>	Adds a category to the graph
<code>laLineGraphWidget_AddDataToSeries</code>	Adds a data (value) to the specified series at categoryID index
<code>laLineGraphWidget_AddSeries</code>	Adds a series to the graph
<code>laLineGraphWidget_DestroyAll</code>	Destroys data, series and categories and frees the memory allocated
<code>laLineGraphWidget_GetCategoryAxisLabelsVisible</code>	Returns GFX_TRUE if the category axis labels are visible
<code>laLineGraphWidget_GetCategoryAxisTicksPosition</code>	Returns the position of the ticks in the category axis
<code>laLineGraphWidget_GetCategoryAxisTicksVisible</code>	Returns GFX_TRUE if the category axis ticks are visible
<code>laLineGraphWidget_GetCategoryText</code>	Gets a copy of the string used to label the category
<code>laLineGraphWidget_GetFillGraphArea</code>	Returns GFX_TRUE if the graph area is filled
<code>laLineGraphWidget_GetFillSeriesArea</code>	Returns GFX_TRUE if the series area are filled
<code>laLineGraphWidget_GetGridlinesVisible</code>	Returns GFX_TRUE if the axis gridlines are visible
<code>laLineGraphWidget_GetMaxValue</code>	Returns the max value of the axis
<code>laLineGraphWidget_GetMinValue</code>	Returns the min value of the axis
<code>laLineGraphWidget_GetSeriesFillPoints</code>	Returns GFX_TRUE if the series points are filled
<code>laLineGraphWidget_GetSeriesLinesVisible</code>	Returns GFX_TRUE if the series lines are visible
<code>laLineGraphWidget_GetSeriesPointSize</code>	Returns the size of the drawn point
<code>laLineGraphWidget_GetSeriesPointType</code>	Returns the type of point drawn for the series data points
<code>laLineGraphWidget_GetSeriesScheme</code>	Returns scheme of the specified series
<code>laLineGraphWidget_GetStacked</code>	Returns GFX_TRUE if the bars are stacked

<code>laLineGraphWidget_GetTickLength</code>	Returns the length of the ticks
<code>laLineGraphWidget_GetValueAxisLabelsVisible</code>	Returns GFX_TRUE if the value axis labels are visible
<code>laLineGraphWidget_GetValueAxisSubtickInterval</code>	Returns the interval between minor ticks in the value axis
<code>laLineGraphWidget_GetValueAxisSubticksPosition</code>	Returns the position of the subticks in the axis
<code>laLineGraphWidget_GetValueAxisSubticksVisible</code>	Returns GFX_TRUE if the value axis subticks are visible
<code>laLineGraphWidget_GetValueAxisTickInterval</code>	Returns the interval between major ticks in the value axis
<code>laLineGraphWidget_GetValueAxisTicksPosition</code>	Returns the position of the ticks in the axis
<code>laLineGraphWidget_GetValueAxisTicksVisible</code>	Returns GFX_TRUE if the value axis ticks are visible
<code>laLineGraphWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laLineGraphWidget_SetCategoryAxisLabelsVisible</code>	Shows/Hides the category axis labels
<code>laLineGraphWidget_SetCategoryAxisTicksPosition</code>	Sets the position of the ticks in the category axis
<code>laLineGraphWidget_SetCategoryAxisTicksVisible</code>	Shows/Hides the category axis ticks
<code>laLineGraphWidget_SetCategoryText</code>	Sets the string used to label the category
<code>laLineGraphWidget_SetDataInSeries</code>	Sets the value of the entry in the series index. The entry should have been previously
<code>laLineGraphWidget_SetFillGraphArea</code>	Sets the graph area filled or not
<code>laLineGraphWidget_SetFillSeriesArea</code>	Sets the series area filled or not
<code>laLineGraphWidget_SetGridlinesVisible</code>	Shows/Hides the gridlines
<code>laLineGraphWidget_SetMaxValue</code>	Sets the max value of the axis
<code>laLineGraphWidget_SetMinValue</code>	Sets the min value of the axis
<code>laLineGraphWidget_SetSeriesFillPoints</code>	Sets the series points filled
<code>laLineGraphWidget_SetSeriesLinesVisible</code>	Shows/hides the lines between series points
<code>laLineGraphWidget_SetSeriesPointSize</code>	Sets the size of the point drawn for the series data
<code>laLineGraphWidget_SetSeriesPointType</code>	Sets the type of point drawn for the series data points
<code>laLineGraphWidget_SetSeriesScheme</code>	Sets the color scheme of the series
<code>laLineGraphWidget_SetStacked</code>	Stacks the line graph
<code>laLineGraphWidget_SetStringTable</code>	Sets the string table used for the generated axis labels
<code>laLineGraphWidget_SetTickLength</code>	Sets the length of the ticks
<code>laLineGraphWidget_SetTicksLabelsStringID</code>	Sets the ID of the superset string used for the value axis tick labels
<code>laLineGraphWidget_SetValueAxisLabelsVisible</code>	Shows/Hides the labels in the value axis
<code>laLineGraphWidget_SetValueAxisSubtickInterval</code>	Sets the minor tick interval in the value axis
<code>laLineGraphWidget_SetValueAxisSubticksPosition</code>	Sets the position of the subticks in the value axis
<code>laLineGraphWidget_SetValueAxisSubticksVisible</code>	Shows/Hides the subticks in the value axis
<code>laLineGraphWidget_SetValueAxisTickInterval</code>	Sets the tick interval in the value axis
<code>laLineGraphWidget_SetValueAxisTicksPosition</code>	Sets the position of the ticks in the value axis
<code>laLineGraphWidget_SetValueAxisTicksVisible</code>	Shows/Hides the ticks in the value axis
<code>laLineWidget_GetEndPoint</code>	Gets the coordinates for the second point of the line.
<code>laLineWidget_GetStartPoint</code>	Gets the coordinates for the first point of the line.

<code>IaLineWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>IaLineWidget_SetEndPoint</code>	Sets the coordinate for the second point of the line
<code>IaLineWidget_SetStartPoint</code>	Sets the coordinate for the first point of the line
<code>IaList_Assign</code>	Assignes a new pointer to an index in the list
<code>IaList_Clear</code>	Removes all nodes from a given list
<code>IaList_Copy</code>	Creates a duplicate of an existing list
<code>IaList_Create</code>	Initializes a new linked list
<code>IaList_Destroy</code>	Removes all nodes from a given list and frees the data of each node
<code>IaList_Find</code>	Retrieves the index of a value from the list
<code>IaList_Get</code>	Retrieves a value from the list
<code>IaList_InsertAt</code>	Inserts an item into a list at a given index. All existing from index are shifted right one place.
<code>IaList_PopBack</code>	Removes the last value from the list
<code>IaList_PopFront</code>	Removes the first value from the list
<code>IaList_PushBack</code>	Pushes a new node onto the back of the list
<code>IaList_PushFront</code>	Pushes a new node onto the front of the list
<code>IaList_Remove</code>	Removes an item from the list
<code>IaList_RemoveAt</code>	Removes an item from the list at an index
<code>IaListWheelWidget_AppendItem</code>	Appends a new item entry to the list. The initial value of the item will be empty.
<code>IaListWheelWidget_GetAlignment</code>	Gets the horizontal alignment for the list widget
<code>IaListWheelWidget_GetFlickInitSpeed</code>	Returns the flick init speed for the wheel.
<code>IaListWheelWidget_GetIconMargin</code>	Gets the icon margin value for the list wheel widget
<code>IaListWheelWidget_GetIconPosition</code>	Sets the icon position for the list wheel widget.
<code>IaListWheelWidget_GetIndicatorArea</code>	Returns the spacing for the selected item indicator bars.
<code>IaListWheelWidget_GetItemCount</code>	Gets the number of items currently contained in the list
<code>IaListWheelWidget_GetItemIcon</code>	Gets the pointer to the image asset for the icon for the item at the given index.
<code>IaListWheelWidget_GetItemText</code>	Gets the text value for an item in the list.
<code>IaListWheelWidget_GetMaxMomentum</code>	Returns the maximum momentum value for the wheel.
<code>IaListWheelWidget_GetMomentumFalloffRate</code>	Returns the momentum falloff rate for the wheel.
<code>IaListWheelWidget_GetRotationUpdateRate</code>	Returns the wheel rotation update rate.
<code>IaListWheelWidget_GetSelectedItem</code>	Returns the index of the currently selected item.
<code>IaListWheelWidget_GetSelectedItemChangedEventCallback</code>	Gets the callback for the item selected changed event
<code>IaListWheelWidget_GetShaded</code>	Returns true if the list is using gradient shading to illustrate depth
<code>IaListWheelWidget_GetShowIndicators</code>	Returns true if the list is displaying its selected item indicators
<code>IaListWheelWidget_GetVisibleItemCount</code>	Returns the list's visible item count
<code>IaListWheelWidget_InsertItem</code>	Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right.

<code>laListWheelWidget_New</code>	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laListWheelWidget_RemoveAllItems</code>	Attempts to remove all items from the list.
<code>laListWheelWidget_RemoveItem</code>	Attempts to remove an item from the list.
<code>laListWheelWidget_SelectNextItem</code>	Attempts to move the selected item index to the next item in the list.
<code>laListWheelWidget_SelectPreviousItem</code>	Attempts to move the selected item index to the previous item in the list.
<code>laListWheelWidget_SetAlignment</code>	Sets the horizontal alignment mode for the list widget.
<code>laListWheelWidget_SetFlickInitSpeed</code>	Configures the flick init speed for the list wheel
<code>laListWheelWidget_SetIconMargin</code>	Sets the icon margin value for the list widget.
<code>laListWheelWidget_SetIconPosition</code>	Sets the icon position for the list wheel widget
<code>laListWheelWidget_SetIndicatorArea</code>	Configures the display area for the list selection indicator bars
<code>laListWheelWidget_SetItemIcon</code>	Sets the icon pointer for a given index.
<code>laListWheelWidget_SetItemText</code>	Sets the text value for an item in the list.
<code>laListWheelWidget_SetMaxMomentum</code>	Configures the maximum momentum value for the wheel
<code>laListWheelWidget_SetMomentumFalloffRate</code>	Configures the momentum falloff rate for the wheel
<code>laListWheelWidget_SetRotationUpdateRate</code>	Configures the rotation update rate for a wheel
<code>laListWheelWidget_SetSelectedItem</code>	Attempts to set the selected item index
<code>laListWheelWidget_SetSelectedItemChangedEventCallback</code>	
<code>laListWheelWidget_SetShaded</code>	Configures the list to use gradient or flat background shading
<code>laListWheelWidget_SetShowIndicators</code>	Configures the list to display the selected item indicator bars
<code>laListWheelWidget_SetVisibleItemCount</code>	Sets the number of visible items in the list. Must be greater than or equal to three and must be an odd number.
<code>laListWidget_AppendItem</code>	Appends a new item entry to the list. The initial value of the item will be empty.
<code>laListWidget_DeselectAll</code>	Attempts to set all item states as not selected.
<code>laListWidget_GetAlignment</code>	Gets the horizontal alignment for the list widget
<code>laListWidget_GetAllowEmptySelection</code>	Returns true if the list allows an empty selection set
<code>laListWidget_GetFirstSelectedItem</code>	Returns the lowest selected item index.
<code>laListWidget_GetIconMargin</code>	Gets the icon margin value for the list widget
<code>laListWidget_GetIconPosition</code>	Gets the icon position for the list
<code>laListWidget_GetItemCount</code>	Gets the number of items currently contained in the list
<code>laListWidget_GetItemEnable</code>	Returns the enable state of the item in the list widget
<code>laListWidget_GetItemIcon</code>	Gets the pointer to the image asset for the icon for the item at the given index.
<code>laListWidget_GetItemSelected</code>	Returns true if the item at the given index is currently selected.
<code>laListWidget_GetItemText</code>	Gets the text value for an item in the list.
<code>laListWidget_GetLastSelectedItem</code>	Returns the highest selected item index.
<code>laListWidget_GetSelectedItemChangedEventCallback</code>	Gets the callback for the item selected changed event
<code>laListWidget_GetSelectionCount</code>	Returns the number of selected items in the list.

<code>laListWidget_GetSelectionMode</code>	Gets the selection mode for the list
<code>laListWidget_InsertItem</code>	Attempts to insert a new item at the desired index. Existing items at idx or greater will be shuffled one index to the right.
<code>laListWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laListWidget_RemoveAllItems</code>	Attempts to remove all items from the list.
<code>laListWidget_RemoveItem</code>	Attempts to remove an item from the list.
<code>laListWidget_SelectAll</code>	Attempts to set all item states to selected.
<code>laListWidget_SetAlignment</code>	Sets the horizontal alignment mode for the list widget.
<code>laListWidget_SetAllowEmptySelection</code>	Configures the list to allow an empty selection set.
<code>laListWidget_SetIconMargin</code>	Sets the icon margin value for the list widget.
<code>laListWidget_SetIconPosition</code>	Sets the icon position for the list widget
<code>laListWidget_SetItemEnable</code>	Enables or disables an item in the list. A disable item becomes un-selectable.
<code>laListWidget_SetItemIcon</code>	Sets the icon pointer for a given index.
<code>laListWidget_SetItemSelected</code>	Attempts to set the item at idx as selected.
<code>laListWidget_SetItemText</code>	Sets the text value for an item in the list.
<code>laListWidget_SetItemVisible</code>	
<code>laListWidget_SetSelectedItemChangedEventCallback</code>	Sets the callback for the item selected changed event
<code>laListWidget_SetSelectionMode</code>	Set the list selection mode
<code>laListWidget_ToggleItemSelected</code>	Attempts to toggle the selected state of the item at idx.
<code>laPieChartWidget_AddEntry</code>	Adds an entry to the pie chart
<code>laPieChartWidget_DeleteEntries</code>	Deletes ALL the data in the pie chart
<code>laPieChartWidget_GetCenterAngle</code>	Sets the center angle of the chart widget
<code>laPieChartWidget_GetEntryOffset</code>	Returns the offset of the entry at the specified index
<code>laPieChartWidget_GetEntryRadius</code>	Returns the radius of the entry at the specified index
<code>laPieChartWidget_GetEntryScheme</code>	Returns the scheme of the entry at the specified index
<code>laPieChartWidget_GetEntryValue</code>	Returns the value of the entry at the specified index
<code>laPieChartWidget_GetLabelsOffset</code>	Gets the offsets of the labels from the center
<code>laPieChartWidget_GetLabelsVisible</code>	Returns GFX_TRUE if the labels are shown, GFX_FALSE if hidden
<code>laPieChartWidget_GetOrigin</code>	Gets the origin coordinates of a chart widget
<code>laPieChartWidget_GetStartAngle</code>	Returns the start angle of a chart widget
<code>laPieChartWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laPieChartWidget_SetCenterAngle</code>	Sets the center angle of the chart widget
<code>laPieChartWidget_SetEntryOffset</code>	Sets the offset of an entry at index
<code>laPieChartWidget_SetEntryRadius</code>	Sets the radius of an entry at index
<code>laPieChartWidget_SetEntryScheme</code>	Sets the scheme of an entry at index
<code>laPieChartWidget_SetEntryValue</code>	Sets the value of an entry at index
<code>laPieChartWidget_SetLabelsOffset</code>	Sets the offsets of the labels from the center
<code>laPieChartWidget_SetLabelsStringID</code>	Sets the string asset for the labels

<code>laPieChartWidget_SetLabelsVisible</code>	Shows/hides the data entry labels
<code>laPieChartWidget_SetOrigin</code>	Sets the origin coordinates of a chart widget
<code>laPieChartWidget_SetPressedEventCallback</code>	Sets the function called when the chart is pressed/touched
<code>laPieChartWidget_SetStartAngle</code>	Sets the start angle of a chart widget
<code>laPieChartWidget_SetStringTable</code>	Sets the string table for the labels
<code>laProgressBarWidget_GetDirection</code>	Gets the fill direction value for a progress bar widget
<code>laProgressBarWidget_GetValue</code>	Gets the percentage value for a progress bar.
<code>laProgressBarWidget_GetValueChangedEventCallback</code>	Gets the currently set value changed event callback.
<code>laProgressBarWidget_New</code>	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laProgressBarWidget_SetDirection</code>	Sets the fill direction for a progress bar widget
<code>laProgressBarWidget_SetValue</code>	Sets the percentage value for a progress bar. Valid values are 0 - 100.
<code>laProgressBarWidget_SetValueChangedCallback</code>	Sets the desired value changed event callback pointer
<code>laRadialMenuWidget_AddWidget</code>	Add a widget to the radial menu
<code>laRadialMenuWidget_ClearItems</code>	Clears all items in the radial menu widget
<code>laRadialMenuWidget_GetItemProminenceChangedEventCallback</code>	Gets the current radial menu item prominence change event callback
<code>laRadialMenuWidget_GetItemSelectedEventCallback</code>	Gets the current radial menu item selected event callback
<code>laRadialMenuWidget_GetProminentIndex</code>	Gets the index of the widget within the radial menu that is prominent
<code>laRadialMenuWidget_GetTheta</code>	Gets the theta value for the radial menu.
<code>laRadialMenuWidget_GetWidget</code>	Gets the pointer for the widget by index
<code>laRadialMenuWidget_IsProminent</code>	Returns true if this radial menu item is currently in the primary selectable position
<code>laRadialMenuWidget_New</code>	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laRadialMenuWidget_RemoveWidget</code>	Removes a widget to the radial menu
<code>laRadialMenuWidget_SetAlphaScaleMinMax</code>	Sets the minimum and maximum alpha scaling ratio
<code>laRadialMenuWidget_SetAlphaScaling</code>	Enables per item alpha scaling for the radial menu
<code>laRadialMenuWidget_SetDrawEllipse</code>	Enables drawing the elliptical track for the radial menu
<code>laRadialMenuWidget_SetEllipseType</code>	Sets the ellipse type for the radial menu track.
<code>laRadialMenuWidget_SetHighlightProminent</code>	Sets the item widget to highlight when it is at the prominent location
<code>laRadialMenuWidget_SetItemProminenceChangedEventCallback</code>	Sets the deselected callback pointer
<code>laRadialMenuWidget_SetItemSelectedEventCallback</code>	Sets the radial menu item selected event callback
<code>laRadialMenuWidget_SetNumberOfItemsShown</code>	Sets the number of items visible on the menu
<code>laRadialMenuWidget_SetProminent</code>	Sets this widget as prominent.
<code>laRadialMenuWidget_SetProminentIndex</code>	Sets a widget with index within the radial menu as prominent
<code>laRadialMenuWidget_SetSizeScaleMinMax</code>	Sets the minimum and maximum size scaling ratio in percent
<code>laRadialMenuWidget_SetSizeScaling</code>	Enables per item size scaling for the radial menu

<code>laRadialMenuWidget_SetTheta</code>	Sets the theta value for the radial menu.
<code>laRadialMenuWidget_SetTouchArea</code>	Sets the area that touch input is allowed within the radial menu widget
<code>laRadialMenuWidget_SetWidgetAt</code>	Insert a widget to the radial menu at the index specified
<code>laRadioButtonGroup_AddButton</code>	Add a button widget to the button list of the selected Radio button group.
<code>laRadioButtonGroup_Create</code>	This function creates a GFX_GOL_RADIOBUTTON group with the provided button list.
<code>laRadioButtonGroup_Destroy</code>	This function destroys the GFX_GOL_RADIOBUTTON group
<code>laRadioButtonGroup_RemoveButton</code>	Remove a button widget to the button list of the selected Radio button group.
<code>laRadioButtonGroup_SelectButton</code>	Select the button widget specified from the button list for the Radio button group.
<code>laRadioButtonWidget_GetCircleButtonSize</code>	Gets the diameter/size of the default circle button
<code>laRadioButtonWidget_GetDeselectedEventCallback</code>	Gets the current radio button deselected event callback
<code>laRadioButtonWidget_GetGroup</code>	Returns the pointer to the currently set radio button group.
<code>laRadioButtonWidget_GetHAlignment</code>	Gets the horizontal alignment setting for a button
<code>laRadioButtonWidget_GetImageMargin</code>	Gets the distance between the icon and the text
<code>laRadioButtonWidget_GetImagePosition</code>	Gets the current image position setting for the radio button
<code>laRadioButtonWidget_GetSelected</code>	Returns true if this radio button is currently selected
<code>laRadioButtonWidget_GetSelectedEventCallback</code>	Gets the current radio button selected event callback
<code>laRadioButtonWidget_GetSelectedImage</code>	Gets the selected image asset pointer for a button
<code>laRadioButtonWidget_GetText</code>	Gets the text value for the button.
<code>laRadioButtonWidget_GetUnselectedImage</code>	Gets the image asset pointer currently used as the unselected icon
<code>laRadioButtonWidget_GetVAlignment</code>	Sets the vertical alignment for a button
<code>laRadioButtonWidget_New</code>	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
<code>laRadioButtonWidget_SetCircleButtonSize</code>	Sets the size of the default circle button
<code>laRadioButtonWidget_SetDeselectedEventCallback</code>	Sets the deselected callback pointer
<code>laRadioButtonWidget_SetHAlignment</code>	Sets the horizontal alignment value for a button
<code>laRadioButtonWidget_SetImageMargin</code>	Sets the distance between the icon and text
<code>laRadioButtonWidget_SetImagePosition</code>	Sets the image relative position setting for the radio button
<code>laRadioButtonWidget_SetSelected</code>	Sets this button as selected.
<code>laRadioButtonWidget_SetSelectedEventCallback</code>	Sets the radio button selected event callback
<code>laRadioButtonWidget_SetSelectedImage</code>	Sets the image to be used as a selected icon
<code>laRadioButtonWidget_SetText</code>	Sets the text value for the button.
<code>laRadioButtonWidget_SetUnselectedImage</code>	Sets the asset pointer for the radio button's unselected image icon
<code>laRadioButtonWidget_SetVAlignment</code>	Sets the vertical alignment for a button
<code>laRectangleWidget_GetThickness</code>	Gets the rectangle border thickness setting

	laRectangleWidget_New	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	laRectangleWidget_SetThickness	Sets the rectangle border thickness setting
	laRectArray_Clear	Removes all values from a given array. Array capacity remains the same.
	laRectArray_Copy	Creates a duplicate of an existing array
	laRectArray_Create	Initializes a new rectangle array.
	laRectArray_Destroy	Removes all nodes from a given array and frees the memory owned by the array. Resets array capacity to zero.
	laRectArray_InsertAt	Inserts a rectangle into an array at a given index. All existing nodes from index are shifted right one place.
	laRectArray_MergeSimilar	Analyzes an array and merges any rectangles similar in size.
	laRectArray_PopBack	Removes the last rectangle from the array
	laRectArray_PopFront	Removes the first value from the array. Shuffles all other nodes forward one index.
	laRectArray_PushBack	Pushes a new rectangle onto the back of the array
	laRectArray_PushFront	Pushes a new rectangle onto the front of the array. Shuffles all other nodes backward one index.
	laRectArray_RemoveAt	Removes a rectangle from the array at an index
	laRectArray_RemoveDuplicates	Removes any duplicate rectangles from an array.
	laRectArray_RemoveOverlapping	Sorts the array by size and then removes any rectangles that are completely overlapped by another larger rectangle.
	laRectArray_Resize	Resizes the capacity of the array. If the array shrinks, any nodes beyond the new capacity will be discarded.
	laRectArray_SortBySize	Sorts a given array largest to smallest.
	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_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 This function has been deprecated in favor of laContext_SetActiveScreen
	laScreen_New	Create a new screen, initialize it to the values specified.

<code>laScreen_SetHideEventCallback</code>	Set the hide call back event function pointer for the specified screen
<code>laScreen_SetLayer</code>	Assigns the provided layer pointer to the screen at the given index This function has been deprecated in favor of <code>laContext_SetActiveScreen</code>
<code>laScreen_SetLayerSwapSync</code>	Sets the layer swap sync setting for the specified screen
<code>laScreen_SetMirrored</code>	Sets the mirror setting for the specified screen
<code>laScreen_SetOrientation</code>	Sets the orientation object to the specified screen
<code>laScreen_SetShowEventCallback</code>	Set the show call back event function pointer for the specified screen
<code>laScreen_Show</code>	Make the specified screen active and show it on the display
<code>laScrollBarWidget_GetExtentValue</code>	Gets the current scroll bar extent value
<code>laScrollBarWidget_GetMaximumValue</code>	Gets the maximum scroll value
<code>laScrollBarWidget_GetOrientation</code>	Gets the orientation value for the scroll bar
<code>laScrollBarWidget_GetScrollPercentage</code>	Gets the current scroll value as a percentage
<code>laScrollBarWidget_GetScrollValue</code>	Gets the current scroll value
<code>laScrollBarWidget_GetStepSize</code>	Gets the current discreet step size
<code>laScrollBarWidget.GetValueChangedEventCallback</code>	Gets the current value changed callback function pointer
<code>laScrollBarWidget_New</code>	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.
<code>laScrollBarWidget_SetExtentValue</code>	Sets the scroll bar extent value
<code>laScrollBarWidget_SetMaximumValue</code>	Sets the maximum scroll value
<code>laScrollBarWidget_SetOrientation</code>	Sets the orientation value of the scroll bar
<code>laScrollBarWidget_SetScrollPercentage</code>	Sets the current scroll value using a percentage. Percentage should be a value from 0 - 100
<code>laScrollBarWidget_SetScrollValue</code>	Sets the current scroll value
<code>laScrollBarWidget_SetStepSize</code>	Sets the current step size
<code>laScrollBarWidget_SetValueChangedEventCallback</code>	Sets the value changed event callback pointer
<code>laScrollBarWidget_StepBackward</code>	Moves the scroll value back by the current step size
<code>laScrollBarWidget_StepForward</code>	Moves the scroll value forward by the current step size
<code>laShutdown</code>	Function to shutdown the active Aria library state.
<code>laSliderWidget_GetGripSize</code>	Gets the current grip size of the slider
<code>laSliderWidget_GetMaximumValue</code>	Gets the maximum value for the slider
<code>laSliderWidget_GetMinimumValue</code>	Gets the minimum value for the slider
<code>laSliderWidget_GetOrientation</code>	Gets the orientation value for the slider
<code>laSliderWidget_GetSliderPercentage</code>	Gets the slider value as a percentage
<code>laSliderWidget_GetSliderValue</code>	Gets the current slider value
<code>laSliderWidget.GetValueChangedEventCallback</code>	Gets the current value changed event callback pointer
<code>laSliderWidget_New</code>	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.
<code>laSliderWidget_SetGripSize</code>	Sets the grip size of the slider

<code>laSliderWidget_SetMaximumValue</code>	Sets the maximum value for the slider
<code>laSliderWidget_SetMinimumValue</code>	Sets the minimum value for the slider
<code>laSliderWidget_SetOrientation</code>	
<code>laSliderWidget_SetSliderPercentage</code>	Sets the slider value using a percentage. Value must be from 0 - 100.
<code>laSliderWidget_SetSliderValue</code>	Sets the current slider value
<code>laSliderWidget_SetValueChangedEventCallback</code>	Sets the value changed event callback pointer
<code>laSliderWidget_Step</code>	Moves the slider by a given amount
<code>laString_Allocate</code>	Attempts to resize the local data buffer for a string.
<code>laString_Append</code>	Appends a string onto the end of another string
<code>laString_Capacity</code>	Returns the capacity of a string
<code>laString_CharAt</code>	Extracts the code point for the character in a string at a given index.
<code>laString_Clear</code>	Sets a string's length to zero and its string table reference to NULL. Does not free any associated data and preserves capacity.
<code>laString_Compare</code>	Compares two string objects
<code>laString_CompareBuffer</code>	Compares a string object and a <code>GFXU_CHAR*</code> buffer
<code>laString_Copy</code>	Copies the values from one string into another
<code>laString_CreateFromBuffer</code>	Creates a string object from a <code>GFXU_CHAR</code> buffer and a font asset pointer
<code>laString_CreateFromCharBuffer</code>	Creates a string object from a <code>const char*</code> buffer and a font asset pointer. This method provides compatibility with standard c-style strings. Input string will be converted from 8-bit width to 32-bit width.
<code>laString_CreateFromID</code>	Creates a string object that simply references a string in the string table.
<code>laString_Delete</code>	Deletes all memory associated with a string object
<code>laString_Destroy</code>	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.
<code>laString_Draw</code>	Wrapper around GFX Utility string draw function for Aria user interface library. Internal use only.
<code>laString_DrawClipped</code>	Wrapper around GFX Utility string draw function for Aria user interface library. Draws only a clipped area of a string. Internal use only.
<code>laString_DrawSubStringClipped</code>	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.
<code>laString_ExtractFromTable</code>	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.
<code>laString_GetAscent</code>	Returns the ascent of a string by referencing its associated font asset data.
<code>laString_GetCharIndexAtPoint</code>	Given an offset in pixels returns the corresponding character index.
<code>laString_GetCharOffset</code>	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.
≡◊	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_GetFocusChangedEventCallback	Gets the current focus changed event callback pointer
≡◊	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 management 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.

<code>laTextFieldWidget_SetCursorDelay</code>	Sets the cursor delay value
<code>laTextFieldWidget_SetCursorEnabled</code>	Sets the cursor enabled value flag
<code>laTextFieldWidget_SetCursorPosition</code>	Sets the position of the cursor
<code>laTextFieldWidget_SetFocusChangedEventCallback</code>	Sets the focus changed event callback pointer
<code>laTextFieldWidget_SetText</code>	Sets the text value for the box.
<code>laTextFieldWidget_SetTextChangedEventCallback</code>	Sets the text changed event callback pointer
<code>laTouchTest_AddPoint</code>	Adds a point to the touch test widget. The point will then be displayed.
<code>laTouchTest_ClearPoints</code>	Clears all of the existing touch points
<code>laTouchTestWidget_GetPointAddedEventCallback</code>	Gets the current point added event callback
<code>laTouchTestWidget_New</code>	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.
<code>laTouchTestWidget_SetPointAddedEventCallback</code>	Sets the point added event callback
<code>laUpdate</code>	Aria library update (tasks) function.
<code>laUpdateRTOS</code>	RTOS version of the Aria library update (tasks) function.
<code>laUtils_ArrangeRectangle</code>	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.
<code>laUtils_ArrangeRectangleRelative</code>	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.
<code>laUtils_ChildIntersectsParent</code>	Performs an intersection test between a parent widget and a child widget
<code>laUtils_ClipRectToParent</code>	Clips a rectangle to the parent of a widget
<code>laUtils_GetLayer</code>	Finds the root parent of a widget, which should be a layer
<code>laUtils_GetNextHighestWidget</code>	Gets the next highest Z order widget in the tree from 'wgt'
<code>laUtils_ListOcclusionCullTest</code>	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.
<code>laUtils_OcclusionCullTest</code>	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.
<code>laUtils_Pick</code>	Finds the top-most visible widget in the tree at the given coordinates.
<code>laUtils_PickFromLayer</code>	Finds the top-most visible widget in a layer at the given coordinates.
<code>laUtils_PickRect</code>	Finds all of the visible widgets in the given rectangular area.
<code>laUtils_PointScreenToLocalSpace</code>	Converts a point from layer space into the local space of a widget
<code>laUtils_PointToLayerSpace</code>	Converts a point from widget space into layer space

<code>laUtils_RectFromLayerSpace</code>	Converts a rectangle from layer space to widget local space
<code>laUtils_RectFromParentSpace</code>	Converts a rectangle from widget parent space to widget local space
<code>laUtils_RectToLayerSpace</code>	Converts a rectangle from widget local space to layer space
<code>laUtils_RectToParentSpace</code>	Converts a rectangle from widget local space to widget parent space. Widget must be a child of a layer for this to function.
<code>laUtils_RectToScreenSpace</code>	Converts a rectangle from widget local space to screen space
<code>laUtils_ScreenToMirroredSpace</code>	Takes a point in screen space and returns a transformed version in mirrored space.
<code>laUtils_ScreenToOrientedSpace</code>	Takes a point in screen space and returns a transformed version in oriented space.
<code>laUtils_WidgetIsOccluded</code>	Returns LA_TRUE if the specified widget is occluded in the specified rectangle area.
<code>laUtils_WidgetLayerRect</code>	Returns the bounding rectangle of a widget in layer space
<code>laUtils_WidgetLocalRect</code>	Returns the bounding rectangle of a widget in local space
<code>laWidget_AddChild</code>	Adds the child to the parent widget specified in the argument
<code>laWidget_Delete</code>	Delete the widget object specified
<code>laWidget_DeleteAllDescendants</code>	Deletes all of the descendants of the given widget.
<code>laWidget_GetAlphaAmount</code>	Return the widget's global alpha amount
<code>laWidget_GetAlphaEnable</code>	Return the alpha enable property of the widget
<code>laWidget_GetBackgroundType</code>	Return the property value 'background type' associated with the widget object
<code>laWidget_GetBorderType</code>	Return the border type associated with the widget object
<code>laWidget_GetChildAtIndex</code>	Fetches the child at the specified index from the children list of the specified parent widget
<code>laWidget_GetChildCount</code>	Returns the size of the children list of the specified parent widget
<code>laWidget_GetCornerRadius</code>	Returns the corner radius of the widget
<code>laWidget_GetCumulativeAlphaAmount</code>	Calculates the cumulative alpha amount for a hierarchy of widgets
<code>laWidget_GetCumulativeAlphaEnable</code>	Determines if this or any ancestor widget has alpha enabled
<code>laWidget_GetEnabled</code>	Returns the boolean value of the widget enabled property
<code>laWidget_GetHeight</code>	Returns the widget rectangles height
<code>laWidget_GetIndexOfChild</code>	Fetches the index of the child from the children list of the specified parent widget
<code>laWidget_GetMargin</code>	Returns the margin value associated with the widget in the <code>laMargin</code> pointer
<code>laWidget_GetOptimizationFlags</code>	Returns the optimization flags for the widget
<code>laWidget_GetScheme</code>	Returns the scheme associated with the specified widget
<code>laWidget_GetVisible</code>	Returns the boolean value of the widget visible property
<code>laWidgetGetWidth</code>	Returns the widget rectangles width
<code>laWidgetGetX</code>	Returns the widget rectangles upper left corner x-coordinate

<code>laWidget_GetY</code>	Returns the widget rectangle's upper left corner y-coordinate.
<code>laWidget_HasFocus</code>	Checks if the widget specified has focus in the current context.
<code>laWidget_Invalidate</code>	Invalidates the specified widget.
<code>laWidget_isOpaque</code>	Returns true if the widget is considered opaque.
<code>laWidget_New</code>	Create a new widget.
<code>laWidget_OverrideTouchDownEvent</code>	Replace the TouchDownEvent callback for the widget with the new function pointer specified.
<code>laWidget_OverrideTouchMovedEvent</code>	Replace the TouchMovedEvent callback for the widget with the new function pointer specified.
<code>laWidget_OverrideTouchUpEvent</code>	Replace the TouchUpEvent callback for the widget with the new function pointer specified.
<code>laWidget_RectToLayerSpace</code>	Transforms a widget rectangle from local space to its root layer space.
<code>laWidget_RectToParentSpace</code>	Returns the rectangle containing the parent of the widget specified.
<code>laWidget_RectToScreenSpace</code>	Transforms a widget rectangle from local space to screen space coordinates.
<code>laWidget_RemoveChild</code>	Removes the child from the parent widget specified in the argument.
<code>laWidget_Resize</code>	Changes the widget size by the new defined width and height increments.
<code>laWidget_SetAlphaAmount</code>	Set the widget's global alpha amount to the specified alpha amount.
<code>laWidget_SetAlphaEnable</code>	Set the alpha enable property of the widget with the boolean value specified.
<code>laWidget_SetBackgroundType</code>	Set the property value 'background type' associated with the widget object.
<code>laWidget_SetBorderType</code>	Set the border type associated with the widget object.
<code>laWidget_SetCornerRadius</code>	Sets the widget corner radius.
<code>laWidget_SetEnabled</code>	Sets the boolean value of the widget enabled property.
<code>laWidget_SetFocus</code>	Set the widget into focus for the current context.
<code>laWidget_SetHeight</code>	Sets the widget's height value.
<code>laWidget_SetMargins</code>	Set the margin value for left, right, top and bottom margins associated with the widget.
<code>laWidget_SetOptimizationFlags</code>	Sets the optimizations for a widget.
<code>laWidget_SetParent</code>	Sets the parent of the child widget to that specified in the argument list.
<code>laWidget_SetPosition</code>	Changes the widget position to the new defined x and y coordinates.
<code>laWidget_SetScheme</code>	Sets the scheme variable for the specified widget.
<code>laWidget_SetSize</code>	Changes the widget size to the new defined width and height dimensions.
<code>laWidget_SetVisible</code>	Sets the boolean value of the widget visible property.
<code>laWidget_SetWidth</code>	Sets the widget's width value.
<code>laWidget_SetX</code>	Sets the widget's x coordinate position.
<code>laWidget_SetY</code>	Sets the widget's y coordinate position.
<code>laWidget_Translate</code>	Changes the widget position by moving the widget by the defined x and y increments.
<code>laWindowWidget_GetIcon</code>	Gets the currently used window icon.
<code>laWindowWidget_GetIconMargin</code>	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 management 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.

b) Data Types and Constants

	Name	Description
	GFX_Point	A two dimensional Cartesian point.
	GFX_Rect	A rectangle definition.
	laArcWidget_t	Implementation of a arc widget.
	laBackgroundType	Specifies the different background types used for the widgets in the library
	laBackgroundType_t	Specifies the different background types used for the widgets in the library
	laBarGraphCategory_t	Contains the Category properties
	laBarGraphDataSeries_t	The data series object that contains the series properties and data
	laBarGraphTickPosition_t	The tick position relative to axis
	laBarGraphValueAxis_t	The value axis index value
	laBarGraphWidget_t	Implementation of a bar graph widget.
	laBool	libaria bool values
	laBool_t	libaria bool values
	laBorderType	Specifies the different border types used for the widgets in the library
	laBorderType_t	Specifies the different border types used for the widgets in the library
	laButtonState	Controls the button pressed state
	laButtonState_t	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	Button pressed event function callback type
	laButtonWidget_ReleasedEvent	Button released event function callback type
	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	Implementation of a checkbox widget.
	laCheckBoxWidget_CheckedEvent	Checkbox checked event function callback type
	laCheckBoxWidget_t	Implementation of a checkbox widget.
	laCheckBoxWidget_UncheckedEvent	Checkbox unchecked event function callback type
	laCircleWidget	Implementation of a circle widget.
	laCircleWidget_t	Implementation of a circle widget.
	laCircularGaugeArc_t	Internal structure for the arcs in the circular gauge widget
	laCircularGaugeLabel_t	Label object for the circular gauge
	laCircularGaugeTick_t	Tick object for the circular gauge
	laCircularGaugeWidget_t	Implementation of a circular gauge widget.
	laCircularGaugeWidgetArcType_t	Type of arc
	laCircularGaugeWidgetDir_t	Direction of the gauge
	laCircularGaugeWidgetLabelPosition_t	Direction of the gauge
	laCircularSliderArc_t	Internal structure for the arcs in the circular slider widget

	laCircularSliderButtonState_t	State of the slider button
	laCircularSliderWidget_t	Implementation of a slider widget.
	laCircularSliderWidgetArcType_t	The arcs that compose the circular slider
	laCircularSliderWidgetDir_t	Direction of the slider
	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.
	laContext_t	An instance of the Aria user interface library.
	laContextFrameState	Possible values for context frame state.
	laContextFrameState_t	Possible values for context frame state.
	laContextUpdateState	Possible values for context update state.
	laContextUpdateState_t	Possible values for context update state.
	laDrawSurfaceWidget	Implementation of a Drawsurface widget.
	laDrawSurfaceWidget_DrawCallback	Draw surface draw event function callback type
	laDrawSurfaceWidget_t	Implementation of a Drawsurface widget.
	laEditWidget	Specifies the edit widget structure to manage all properties and events associated with edit widgets
	laEditWidget_t	Specifies the edit widget structure to manage all properties and events associated with edit widgets
	laEvent	Basic UI event definition
	laEvent_FilterEvent	Function pointer to define an event filter. Event filters allow a receiver to discard undesirable events
	laEvent_t	Basic UI event definition
	laEventID	Defines internal event type IDs
	laEventID_t	Defines internal event type IDs
	laEventResult	Defines what happened when processing an event
	laEventResult_t	Defines what happened when processing an event
	laEventState	Structure to manage the event lists, state and call back pointers
	laEventState_t	Structure to manage the event lists, state and call back pointers
	laGestureID	Placeholder for eventual gesture support.
	laGestureID_t	Placeholder for eventual gesture support.
	laGradientWidget	Gradient widget struct definition.
	laGradientWidget_t	Gradient widget struct definition.
	laGradientWidgetDirection	Implementation of a gradient widget.
	laGradientWidgetDirection_t	Implementation of a gradient widget.
	laGroupBoxWidget	Group box struct definition.
	laGroupBoxWidget_t	Group box struct definition.
	laHAlignment	libaria horizontal alignment values
	lalImagePlusWidget_ResizeFilter_t	
	lalImagePlusWidget_t	Image plus widget struct definition
	lalImageSequenceEntry	Image sequence entry definition
	lalImageSequenceEntry_t	Image sequence entry definition
	lalImageSequenceImageChangedEvent_FnPtr	Image changed event function callback type
	lalImageSequenceWidget	Image sequence widget struct definition
	lalImageSequenceWidget_t	Image sequence widget struct definition
	lalImageWidget	Image widget struct definition
	lalImageWidget_DrawEventCallback	
	lalImageWidget_t	Image widget struct definition
	laInput_TouchDownEvent	Register and handle the touch press detect event
	laInput_TouchDownEvent_t	Register and handle the touch press detect event
	laInput_TouchMovedEvent	Register and handle the touch coordinates changed event

	<code>laInput_TouchMovedEvent_t</code>	Register and handle the touch coordinates changed event
	<code>laInput_TouchUpEvent</code>	Register and handle the touch release detect event
	<code>laInput_TouchUpEvent_t</code>	Register and handle the touch release detect event
	<code>laInputState</code>	Maintain a history of touch states; currently libaria keeps track of the last touch state only.
	<code>laInputState_t</code>	Maintain a history of touch states; currently libaria keeps track of the last touch state only.
	<code>laKey</code>	All values possible for key entry from the libaria keyboard widget
	<code>laKey_t</code>	All values possible for key entry from the libaria keyboard widget
	<code>laKeyPadActionTrigger</code>	Defines the trigger for keypad action and events
	<code>laKeyPadActionTrigger_t</code>	Defines the trigger for keypad action and events
	<code>laKeyPadCell</code>	Defines a key pad cell struct
	<code>laKeyPadCell_t</code>	Defines a key pad cell struct
	<code>laKeyPadCellAction</code>	Defines an assigned action to a key pad cell
	<code>laKeyPadCellAction_t</code>	Defines an assigned action to a key pad cell
	<code>laKeyPadWidget</code>	Defines a key pad widget struct
	<code>laKeyPadWidget_GetKeyPadActionTrigger</code>	Gets the current trigger for keypad edit action and events
	<code>laKeyPadWidget_KeyClickEvent</code>	Key click event function callback type
	<code>laKeyPadWidget_SetKeyPadActionTrigger</code>	Sets the current trigger for keypad edit action and events
	<code>laKeyPadWidget_t</code>	Defines a key pad widget struct
	<code>laLabelWidget</code>	Implementation of a label widget struct
	<code>laLabelWidget_t</code>	Implementation of a label widget struct
	<code>laLayer</code>	Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent.
	<code>laLayer_t</code>	Primary definition of a layer. Builds on base functions of a standard widget. Should never have a direct parent.
	<code>laLayerBuffer</code>	Structure to maintain the buffer type and track the buffer location for each layer
	<code>laLayerBuffer_t</code>	Structure to maintain the buffer type and track the buffer location for each layer
	<code>laLayerBufferType</code>	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.
	<code>laLayerBufferType_t</code>	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.
	<code>laLayerFrameState</code>	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.
	<code>laLayerFrameState_t</code>	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.
	<code>laLineGraphCategory_t</code>	Contains the Category properties
	<code>laLineGraphDataPointType_t</code>	The graph data point type
	<code>laLineGraphDataSeries_t</code>	The data series object that contains the series properties and data
	<code>laLineGraphTickPosition_t</code>	The tick position relative to axis
	<code>laLineGraphValueAxis_t</code>	The value axis index value
	<code>laLineGraphWidget_t</code>	Implementation of a line graph widget.
	<code>laLineWidget</code>	Defines the implementation of a line widget struct
	<code>laLineWidget_t</code>	Defines the implementation of a line widget struct
	<code>laList</code>	Linked list definition
	<code>laList_t</code>	Linked list definition
	<code>laListItem</code>	Defines a list item struct
	<code>laListItem_t</code>	Defines a list item struct

	laListNode	Linked list node definition
	laListNode_t	Linked list node definition
	laListWheelIndicatorFill	Indicates the fill type for the listwheel indicator area.
	laListWheelIndicatorFill_t	Indicates the fill type for the listwheel indicator area.
	laListWheelItem	Implementation of a list wheel widget item struct
	laListWheelItem_t	Implementation of a list wheel widget item struct
	laListWheelWidget	Implementation of a list wheel widget struct
	laListWheelWidget_GetAutoHideWheel	Returns the list wheel auto hide setting
	laListWheelWidget_GetIndicatorFill	Gets the indicator area fill type
	laListWheelWidget_GetZoomEffects	Gets the list wheel zoom effect
	laListWheelWidget_SelectedItemChangedEvent	Selected item changed event function callback type
	laListWheelWidget_SetAutoHideWheel	Sets the list wheel to auto hide when not active
	laListWheelWidget_SetIndicatorFill	Sets the indicator fill type
	laListWheelWidget_SetZoomEffects	Sets the list wheel zoom effect
	laListWheelWidget_t	Implementation of a list wheel widget struct
	laListWheelZoomEffects	Indicates the zoom effects for the list wheel items.
	laListWidget	Defines the implementation of a list widget
	laListWidget_SelectedItemChangedEvent	Selected item changed event function callback type
	laListWidget_SelectionMode	Defines the list selection modes
	laListWidget_SelectionMode_t	Defines the list selection modes
	laListWidget_t	Defines the implementation of a list widget
	laMargin	libaria margin values
	laMargin_t	libaria margin values
	laMouseButton	All values possible for mouse key entry from the libaria mouse input
	laMouseButton_t	All values possible for mouse key entry from the libaria mouse input
	laPieChartPie_t	
	laPieChartWidget_t	Implementation of a pie chart widget.
	laPreemptionLevel	libaria pre-emption level values
	laProgressBar_ValueChangedEventCallback	Value changed event function callback type
	laProgressBarDirection	Defines the valid values for the progress bar widget fill directions.
	laProgressBarDirection_t	Defines the valid values for the progress bar widget fill directions.
	laProgressBarWidget	Implementation of a progressbar widget struct
	laProgressBarWidget_t	Implementation of a progressbar widget struct
	laRadialMenuEllipseType_t	This is record laRadialMenuEllipseType_t.
	laRadialMenuItemNode_t	This is record laRadialMenuItemNode_t.
	laRadialMenuWidget_t	Implementation of a radial menu widget struct
	laRadialMenuWidgetScaleType_t	This is record laRadialMenuWidgetScaleType_t.
	laRadialMenuWidgetState_t	
	laRadioButtonGroup	Defines the structure used for the Radio Button group.
	laRadioButtonGroup_t	Defines the structure used for the Radio Button group.
	laRadioButtonWidget	Implementation of a radio button widget struct
	laRadioButtonWidget_DeselectedEvent	Radio button deselected function callback type
	laRadioButtonWidget_SelectedEvent	Radio button selected function callback type
	laRadioButtonWidget_t	Implementation of a radio button widget struct
	laRectangleWidget	Implementation of a rectangle widget struct
	laRectangleWidget_t	Implementation of a rectangle widget struct
	laRectArray	Rectangle array definition
	laRectArray_t	Rectangle array definition
	laRelativePosition	libaria relative position values

	<code>laResult</code>	libaria results (success and failure codes).
☝	<code>laResult_t</code>	libaria results (success and failure codes).
	<code>laScheme</code>	This structure specifies the style scheme components of an object.
☝	<code>laScheme_t</code>	This structure specifies the style scheme components of an object.
	<code>laScreen</code>	The structure to maintain the screen related variables and event handling
	<code>laScreen_CreateCallback_FnPtr</code>	Callback pointer for a new screen create event notification. This is called when the library attempts to create a screen.
	<code>laScreen_ShowHideCallback_FnPtr</code>	Callback pointer for the active screen show or hide event change notification.
☝	<code>laScreen_t</code>	The structure to maintain the screen related variables and event handling
	<code>laScreenOrientation</code>	Possible values for screen orientation.
☝	<code>laScreenOrientation_t</code>	Possible values for screen orientation.
	<code>laScrollBarOrientation</code>	Defines the scroll bar direction values
☝	<code>laScrollBarOrientation_t</code>	Defines the scroll bar direction values
	<code>laScrollBarState</code>	Defines the various scroll bar state values
☝	<code>laScrollBarState_t</code>	Defines the various scroll bar state values
	<code>laScrollBarWidget</code>	Implementation of a scroll bar widget.
☝	<code>laScrollBarWidget_t</code>	Implementation of a scroll bar widget.
	<code>laScrollBarWidget_ValueChangedEvent</code>	Value changed event function callback type
	<code>laSliderOrientation</code>	Slider orientations
☝	<code>laSliderOrientation_t</code>	Slider orientations
	<code>laSliderState</code>	Describes various slider states
☝	<code>laSliderState_t</code>	Describes various slider states
	<code>laSliderWidget</code>	Implementation of a slider widget struct
☝	<code>laSliderWidget_t</code>	Implementation of a slider widget struct
	<code>laSliderWidget_ValueChangedEvent</code>	Value changed event function callback type
	<code>laString</code>	String definition
☝	<code>laString_t</code>	String definition
	<code>laTextFieldWidget</code>	Implementation of a text field widget.
☝	<code>laTextFieldWidget_t</code>	Implementation of a text field widget.
	<code>laTextFieldWidget_TextChangedCallback</code>	Text changed event function callback type
	<code>laTouchState</code>	Manage the touch input state and track the touch coordinate
☝	<code>laTouchState_t</code>	Manage the touch input state and track the touch coordinate
	<code>laTouchTestState</code>	Touch test states
☝	<code>laTouchTestState_t</code>	Touch test states
	<code>laTouchTestWidget</code>	Implementation of a touch test widget struct
	<code>laTouchTestWidget_PointAddedEventCallback</code>	Point added event function callback type
☝	<code>laTouchTestWidget_t</code>	Implementation of a touch test widget struct
	<code>laVAlignment</code>	libaria vertical alignment values
	<code>laWidget</code>	Specifies Graphics widget structure to manage all properties and events associated with the widget
☝	<code>laWidget_t</code>	Specifies Graphics widget structure to manage all properties and events associated with the widget
	<code>laWidgetDirtyState</code>	Specifies the different dirty states the widget can be assigned
☝	<code>laWidgetDirtyState_t</code>	Specifies the different dirty states the widget can be assigned
	<code>laWidgetDrawState</code>	Specifies the different draw states the widget can be assigned
☝	<code>laWidgetDrawState_t</code>	Specifies the different draw states the widget can be assigned
	<code>laWidgetEvent</code>	Basic widget event definition
☝	<code>laWidgetEvent_t</code>	Basic widget event definition
	<code>laWidgetOptimizationFlags</code>	Specifies the different draw optimization flags for a widget

	laWidgetOptimizationFlags_t	Specifies the different draw optimization flags for a widget
	laWidgetType	Specifies the different widget types used in the library
	laWidgetType_t	Specifies the different widget types used in the library
	laWidgetUpdateState	Specifies the different update states the widget can be assigned
	laWidgetUpdateState_t	Specifies the different update states the widget can be assigned
	laWindowWidget	Implementation of a window widget struct
	laWindowWidget_t	Implementation of a window widget struct
	NUM_BUTTONS	This is macro NUM_BUTTONS.
	NUM_KEYS	This is macro NUM_KEYS.

Description

This section Aria User Interface Library Interface.

a) Functions

[laArcWidget_GetCenterAngle Function](#)

Gets the center angle of the arc widget

File

[libaria_widget_arc.h](#)

C

```
LIB_EXPORT int32_t laArcWidget_GetCenterAngle(laArcWidget* arc);
```

Returns

int32_t

Parameters

Parameters	Description
laArcWidget* arc	the widget

Function

int32_t laArcWidget_GetCenterAngle(laArcWidget* arc)

[laArcWidget_GetRadius Function](#)

Gets the radius of a arc widget

File

[libaria_widget_arc.h](#)

C

```
LIB_EXPORT uint32_t laArcWidget_GetRadius(laArcWidget* arc);
```

Returns

uint32_t

Parameters

Parameters	Description
laArcWidget* arc	the widget

Function

```
uint32_t laArcWidget_GetRadius(laArcWidget* arc)
```

laArcWidget_GetRoundEdge Function

Returns true if the arc has round edges

File

[libaria_widget_arc.h](#)

C

```
LIB_EXPORT laBool laArcWidget_GetRoundEdge(laArcWidget* arc);
```

Returns

[laBool](#)

Parameters

Parameters	Description
laArcWidget* arc	the widget

Function

```
laBool laArcWidget_GetRoundEdge(laArcWidget* arc)
```

laArcWidget_GetStartAngle Function

Returns the start angle of a arc widget

File

[libaria_widget_arc.h](#)

C

```
LIB_EXPORT int32_t laArcWidget_GetStartAngle(laArcWidget* arc);
```

Returns

[uint32_t](#)

Parameters

Parameters	Description
laArcWidget* arc	the widget

Function

```
int32_t laArcWidget_GetStartAngle(laArcWidget* arc)
```

laArcWidget_GetThickness Function

Gets the thickness of the arc

File

[libaria_widget_arc.h](#)

C

```
LIB_EXPORT uint32_t laArcWidget_GetThickness(laArcWidget* arc);
```

Returns

uint32_t

Parameters

Parameters	Description
laArcWidget* arc	the widget

Function

uint32_t laArcWidget_GetThickness(laArcWidget* arc)

laArcWidget_New Function

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

File[libaria_widget_arc.h](#)**C**LIB_EXPORT laArcWidget* **laArcWidget_New()**;**Returns**

laArcWidget*

Function

laArcWidget* laArcWidget_New()

laArcWidget_SetCenterAngle Function

Sets the center angle of the arc widget

File[libaria_widget_arc.h](#)**C**LIB_EXPORT laResult **laArcWidget_SetCenterAngle(laArcWidget* arc, int32_t angle);****Returns**

laResult - the operation result

Parameters

Parameters	Description
laArcWidget* arc	the widget
int32_t angle	the desired center angle value

Function[laResult laArcWidget_SetCenterAngle\(laArcWidget* arc, int32_t angle\)](#)***laArcWidget_SetRadius Function***

Sets the radius of a arc widget

File[libaria_widget_arc.h](#)**C**

```
LIB_EXPORT laResult laArcWidget_SetRadius(laArcWidget* arc, uint32_t rad);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laArcWidget* arc	the widget
uint32_t red	the desired radius value

Function

[laResult laArcWidget_SetRadius\(laArcWidget* arc, uint32_t rad\)](#)

laArcWidget_SetRoundEdge Function

Sets the arc edge to round

File[libaria_widget_arc.h](#)**C**

```
LIB_EXPORT laResult laArcWidget_SetRoundEdge(laArcWidget* arc, laBool round);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laArcWidget* arc	the widget
laBool round	sets the arc edge round if LA_TRUE

Function

[laResult laArcWidget_GetRoundEdge\(laArcWidget* arc, laBool round\)](#)

laArcWidget_SetStartAngle Function

Sets the start angle of a arc widget

File[libaria_widget_arc.h](#)**C**

```
LIB_EXPORT laResult laArcWidget_SetStartAngle(laArcWidget* arc, int32_t angle);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laArcWidget* arc	the widget
int32_t angle	the desired start angle value

Function

[laResult laArcWidget_SetStartAngle\(laArcWidget* arc, int32_t angle\)](#)

laArcWidget_SetThickness Function

Sets the thickness of the arc widget

File

[libaria_widget_arc.h](#)

C

```
LIB_EXPORT laResult laArcWidget_SetThickness(laArcWidget* arc, uint32_t thickness);
```

Returns

[laResult - the operation result](#)

Parameters

Parameters	Description
laArcWidget* arc	the widget
uint32_t thickness	the desired thickness value

Function

[laResult laArcWidget_SetThickness\(laArcWidget* arc, uint32_t thickness\)](#)

laBarGraphWidget_AddCategory Function

Adds a category to the graph

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_AddCategory(laBarGraphWidget* graph, uint32_t * id);
```

Returns

[laResult - the result of the operation](#)

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t * id	destination of the ID of the new category

Function

[laResult laBarGraphWidget_AddCategory\(laBarGraphWidget* graph, uint32_t * id\)](#)

laBarGraphWidget_AddDataToSeries Function

Adds a data (value) to the specified series at categoryID index

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_AddDataToSeries(laBarGraphWidget* graph, uint32_t
seriesID, int32_t value, uint32_t * index);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t seriesID	the series ID
int32_t value	the value
uint32_t * index	the destination to return the index of the added data

Function

[laResult](#) `laBarGraphWidget_AddDataToSeries(laBarGraphWidget* graph, uint32_t seriesID, uint32_t categoryID, int32_t value)`

laBarGraphWidget_AddSeries Function

Adds a series to the graph

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_AddSeries(laBarGraphWidget* graph, uint32_t * seriesID);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t * seriesID	destination of the returned series ID

Function

[laResult](#) `laBarGraphWidget_AddSeries(laBarGraphWidget* graph, uint32_t * seriesID)`

laBarGraphWidget_DestroyAll Function

Destroys data, series and categories and frees the memory allocated

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_DestroyAll(laBarGraphWidget* graph);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[laResult](#) laBarGraphWidget_DestroyAll([laBarGraphWidget](#)* graph)

laBarGraphWidget_GetCategoryAxisLabelsVisible Function

Returns GFX_TRUE if the category axis labels are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetCategoryAxisLabelsVisible(laBarGraphWidget* graph);
```

Returns

[laBool](#) - GFX_TRUE if the category axis labels are visible

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[laBool](#) laBarGraphWidget_GetCategoryAxisLabelsVisible([laBarGraphWidget](#)* graph)

laBarGraphWidget_GetCategoryAxisTicksPosition Function

Returns the position of the ticks in the category axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBarGraphTickPosition
laBarGraphWidget_GetCategoryAxisTicksPosition(laBarGraphWidget* graph);
```

Returns

[laBarGraphTickPosition](#) - position of the ticks in the category axis

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[laBarGraphTickPosition](#) laBarGraphWidget_GetCategoryAxisTicksPosition([laBarGraphWidget](#)* graph)

laBarGraphWidget_GetCategoryAxisTicksVisible Function

Returns GFX_TRUE if the category axis ticks are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetCategoryAxisTicksVisible(laBarGraphWidget* graph);
```

Returns

[laBool](#) - GFX_TRUE if the category axis ticks are visible

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[laBool](#) laBarGraphWidget_GetCategoryAxisTicksVisible([laBarGraphWidget](#)* graph)

laBarGraphWidget_GetCategoryText Function

Gets a copy of the string used to label the category

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_GetCategoryText(laBarGraphWidget* graph, uint32_t
categoryID, laString * str);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t categoryID	category ID
laString * str	destination of the copied string

Function

[laResult](#) laBarGraphWidget_GetCategoryText([laBarGraphWidget](#)* graph, [uint32_t](#) categoryID, [laString](#) * str)

laBarGraphWidget_GetFillGraphArea Function

Returns GFX_TRUE if the category axis labels are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetFillGraphArea(laBarGraphWidget* graph);
```

Returns

`laBool` - GFX_TRUE if the category axis labels are visible

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget

Function

`laBool laBarGraphWidget_GetFillGraphArea(laBarGraphWidget* graph)`

laBarGraphWidget_GetGridlinesVisible Function

Returns GFX_TRUE if the axis gridlines are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetGridlinesVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis);
```

Returns

`laBool` - GFX_TRUE if the axis gridlines are visible

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget
<code>laBarGraphValueAxis axis</code>	the value axis index

Function

`laBool laBarGraphWidget_GetGridlinesVisible(laBarGraphWidget* graph, laBarGraphValueAxis axis)`

laBarGraphWidget_GetMaxValue Function

Returns the max value of the axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT uint32_t laBarGraphWidget_GetMaxValue(laBarGraphWidget* graph, laBarGraphValueAxis
axis);
```

Returns

`uint32_t` - max value

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget

Function

`uint32_t laBarGraphWidget_GetMaxValue(laBarGraphWidget* graph, laBarGraphValueAxis axis)`

laBarGraphWidget_GetMinValue Function

Returns the min value of the axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT uint32_t laBarGraphWidget_GetMinValue(laBarGraphWidget* graph, laBarGraphValueAxis axis);
```

Returns

uint32_t - min value

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

```
uint32_t laBarGraphWidget_GetMinValue(laBarGraphWidget* graph, laBarGraphValueAxis axis)
```

laBarGraphWidget_GetSeriesScheme Function

Returns scheme of the specified series

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laScheme * laBarGraphWidget_GetSeriesScheme(laBarGraphWidget* graph, uint32_t seriesID);
```

Returns

[laScheme](#) * - scheme of the specified series

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

```
laScheme * laBarGraphWidget_GetSeriesScheme(laBarGraphWidget* graph, uint32_t seriesID)
```

laBarGraphWidget_GetStacked Function

Returns GFX_TRUE if the bars are stacked

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetStacked(laBarGraphWidget* graph);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[laBool laBarGraphWidget_GetStacked\(laBarGraphWidget* graph\)](#)

laBarGraphWidget_GetTickLength Function

Returns the length of the ticks

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT uint32_t laBarGraphWidget_GetTickLength(laBarGraphWidget* graph);
```

Returns

uint32_t - tick length

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[uint32_t laBarGraphWidget_GetTickLength\(laBarGraphWidget* graph\)](#)

laBarGraphWidget_GetValueAxisLabelsVisible Function

Returns GFX_TRUE if the value axis labels are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetValueAxisLabelsVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis);
```

Returns

[laBool](#) - GFX_TRUE if the value axis labels are visible

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

Function

[laBool laBarGraphWidget_GetValueAxisLabelsVisible\(laBarGraphWidget* graph, laBarGraphValueAxis axis\)](#)

laBarGraphWidget_GetValueAxisSubtickInterval Function

Returns the interval between minor ticks in the value axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT uint32_t laBarGraphWidget_GetValueAxisSubtickInterval(laBarGraphWidget* graph,
laBarGraphValueAxis axis);
```

Returns

uint32_t - ticks in pixels

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index

Function

uint32_t laBarGraphWidget_GetValueAxisSubtickInterval(laBarGraphWidget* graph, laBarGraphValueAxis axis)

laBarGraphWidget_GetValueAxisSubticksPosition Function

Returns the position of the subticks in the axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBarGraphTickPosition
laBarGraphWidget_GetValueAxisSubticksPosition(laBarGraphWidget* graph, laBarGraphValueAxis
axis);
```

Returns

laBarGraphTickPosition - the subtick position

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the index of the value axis

Function

laBarGraphTickPosition laBarGraphWidget_GetValueAxisSubticksPosition(laBarGraphWidget* graph, laBarGraphValueAxis axis)

laBarGraphWidget_GetValueAxisSubticksVisible Function

Returns GFX_TRUE if the value axis subticks are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetValueAxisSubticksVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis);
```

Returns

`laBool` - GFX_TRUE if the value axis subticks are visible

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget

Function

```
laBool laBarGraphWidget_GetValueAxisSubticksVisible(laBarGraphWidget* graph, laBarGraphValueAxis axis)
```

laBarGraphWidget_GetValueAxisTickInterval Function

Returns the interval between major ticks in the value axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT uint32_t laBarGraphWidget_GetValueAxisTickInterval(laBarGraphWidget* graph,
laBarGraphValueAxis axis);
```

Returns

`uint32_t` - ticks in pixels

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget
<code>laBarGraphValueAxis axis</code>	the value axis index

Function

```
uint32_t laBarGraphWidget_GetValueAxisTickInterval(laBarGraphWidget* graph, laBarGraphValueAxis axis)
```

laBarGraphWidget_GetValueAxisTicksPosition Function

Returns the position of the ticks in the axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBarGraphTickPosition laBarGraphWidget_GetValueAxisTicksPosition(laBarGraphWidget*
graph, laBarGraphValueAxis axis);
```

Returns

`laBarGraphTickPosition` - the tick position

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget
<code>laBarGraphValueAxis axis</code>	the index of the value axis

Function

`laBarGraphTickPosition laBarGraphWidget_GetValueAxisTicksPosition(laBarGraphWidget* graph, laBarGraphValueAxis axis)`

laBarGraphWidget_GetValueAxisTicksVisible Function

Returns GFX_TRUE if the value axis ticks are visible

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBool laBarGraphWidget_GetValueAxisTicksVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis);
```

Returns

`laBool` - GFX_TRUE if the value axis ticks are visible

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget

Function

`laBool laBarGraphWidget_GetValueAxisTicksVisible(laBarGraphWidget* graph, laBarGraphValueAxis axis)`

laBarGraphWidget_New Function

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

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laBarGraphWidget* laBarGraphWidget_New();
```

Returns

`laBarGraphWidget*`

Function

`laBarGraphWidget* laBarGraphWidget_New()`

laBarGraphWidget_SetCategoryAxisLabelsVisible Function

Shows/Hides the category axis labels

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetCategoryAxisLabelsVisible(laBarGraphWidget* graph,
laBool visible);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBool visible	if GFX_TRUE, the axis labels are shown

Function

LIB_EXPORT [laResult](#) laBarGraphWidget_SetCategoryAxisLabelsVisible(laBarGraphWidget* graph, [laBool](#) visible)

laBarGraphWidget_SetCategoryAxisTicksPosition Function

Sets the position of the ticks in the category axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetCategoryAxisTicksPosition(laBarGraphWidget* graph,  
laBarGraphTickPosition position);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphTickPosition position	position of the ticks

Function

[laResult](#) laBarGraphWidget_SetCategoryAxisTicksPosition(laBarGraphWidget* graph, laBarGraphTickPosition position)

laBarGraphWidget_SetCategoryAxisTicksVisible Function

Shows/Hides the category axis ticks

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetCategoryAxisTicksVisible(laBarGraphWidget* graph,  
laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBool visible	if GFX_TRUE, the axis ticks are shown

Function

[laResult](#) laBarGraphWidget_SetCategoryAxisTicksVisible(laBarGraphWidget* graph, [laBool](#) visible)

laBarGraphWidget_SetCategoryText Function

Sets the string used to label the category

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetCategoryText(laBarGraphWidget* graph, int32_t
categoryID, laString str);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
int32_t categoryID	category ID, if -1 the last category is assigned
laString str	the string to use

Function

```
laResult laBarGraphWidget_SetCategoryText(laBarGraphWidget* graph, uint32_t categoryID, laString str)
```

laBarGraphWidget_SetDataInSeries Function

Sets the value of the entry in the series index. The entry should have been previously

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetDataInSeries(laBarGraphWidget* graph, uint32_t
seriesID, uint32_t index, int32_t value);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t seriesID	the series ID
uint32_t index	the index of the data
int32_t value	the value

Function

```
laResult laBarGraphWidget_SetDataInSeries(laBarGraphWidget* graph,
uint32_t seriesID,
uint32_t index,
int32_t value);
```

laBarGraphWidget_SetFillGraphArea Function

Sets the graph area filled or not

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetFillGraphArea(laBarGraphWidget* graph, laBool fill);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBool fill	if GFX_TRUE, fills the graph area

Function

[laResult laBarGraphWidget_SetFillGraphArea\(laBarGraphWidget* graph, laBool fill\)](#)

laBarGraphWidget_SetGridlinesVisible Function

Shows/Hides the gridlines

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetGridlinesVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis, laBool visible);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	category ID
laBool visible	shows the gridlines if GFX_TRUE

Function

[laResult laBarGraphWidget_SetGridlinesVisible\(laBarGraphWidget* graph, laBarGraphValueAxis axis, laBool visible\)](#)

laBarGraphWidget_SetMaxValue Function

Sets the max value of the axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetMaxValue(laBarGraphWidget* graph, laBarGraphValueAxis
```

```
axis, int32_t value);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
int32_t value	max value

Function

[laResult](#) laBarGraphWidget_SetMaxValue(laBarGraphWidget* graph, laBarGraphValueAxis axis, int32_t value)

laBarGraphWidget_SetMinValue Function

Sets the min value of the axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetMinValue(laBarGraphWidget* graph, laBarGraphValueAxis
axis, int32_t value);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
int32_t value	min value

Function

[laResult](#) laBarGraphWidget_SetMinValue(laBarGraphWidget* graph, laBarGraphValueAxis axis, int32_t value)

laBarGraphWidget_SetSeriesScheme Function

Sets the color scheme of the series

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetSeriesScheme(laBarGraphWidget* graph, int32_t seriesID,
laScheme * scheme);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget

int32_t seriesID	the series ID, if negative the last series is referenced
laScheme * scheme	the color scheme

Function

`laResult laBarGraphWidget_SetSeriesScheme(laBarGraphWidget* graph, uint32_t seriesID, laScheme * scheme)`

laBarGraphWidget_SetStacked Function

Stacks the bar graph

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetStacked(laBarGraphWidget* graph, laBool stacked);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBool stacked	if GFX_TRUE, the bars are stacked

Function

`laResult laBarGraphWidget_SetStacked(laBarGraphWidget* graph, laBool stacked)`

laBarGraphWidget_SetStringTable Function

Sets the string table used for the generated axis labels

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetStringTable(laBarGraphWidget* graph,
GFXU_StringTableAsset * stringTable);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
GFXU_StringTableAsset * stringTable	the string table

Function

`laResult laBarGraphWidget_SetStringTable(laBarGraphWidget* graph, GFXU_StringTableAsset * stringTable)`

laBarGraphWidget_SetTickLength Function

Sets the length of the ticks

File[libaria_widget_bar_graph.h](#)**C**

```
LIB_EXPORT laResult laBarGraphWidget_SetTickLength(laBarGraphWidget* graph, uint32_t length);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t length	length in pixels

Function

[laResult](#) laBarGraphWidget_SetTickLength(laBarGraphWidget* graph, uint32_t length)

laBarGraphWidget_SetTicksLabelsStringID Function

Sets the ID of the superset string used for the value axis tick labels

File[libaria_widget_bar_graph.h](#)**C**

```
LIB_EXPORT laResult laBarGraphWidget_SetTicksLabelsStringID(laBarGraphWidget* graph, uint32_t
stringID);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
uint32_t stringID	the string ID

Function

[laResult](#) laBarGraphWidget_SetTicksLabelsStringID(laBarGraphWidget* graph, uint32_t stringID)

laBarGraphWidget_SetValueAxisLabelsVisible Function

Shows/Hides the labels in the value axis

File[libaria_widget_bar_graph.h](#)**C**

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisLabelsVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis, laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
laBool visible	shows the labels if GFX_TRUE

Function

`laResult laBarGraphWidget_SetValueAxisLabelsVisible(laBarGraphWidget* graph, laBarGraphValueAxis axis, laBool visible)`

laBarGraphWidget_SetValueAxisSubtickInterval Function

Sets the minor tick interval in the value axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisSubtickInterval(laBarGraphWidget* graph,
    laBarGraphValueAxis axis, uint32_t interval);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
uint32_t interval	tick interval in pixels

Function

`laResult laBarGraphWidget_SetValueAxisSubtickInterval(laBarGraphWidget* graph, laBarGraphValueAxis axis, uint32_t interval)`

laBarGraphWidget_SetValueAxisSubticksPosition Function

Sets the position of the subticks in the value axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisSubticksPosition(laBarGraphWidget* graph,
    laBarGraphValueAxis axis, laBarGraphTickPosition position);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
laBarGraphTickPosition position	position of the subticks

Function

`laResult laBarGraphWidget_SetValueAxisSubticksPosition(laBarGraphWidget* graph, laBarGraphValueAxis axis, laBarGraphTickPosition position)`

laBarGraphWidget_SetValueAxisSubticksVisible Function

Shows/Hides the subticks in the value axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisSubticksVisible(laBarGraphWidget* graph,
    laBarGraphValueAxis axis, laBool visible);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget
<code>laBarGraphValueAxis axis</code>	the value axis index
<code>laBool visible</code>	shows the subticks if GFX_TRUE

Function

`laResult laBarGraphWidget_SetValueAxisSubticksVisible(laBarGraphWidget* graph, laBarGraphValueAxis axis, laBool visible)`

laBarGraphWidget_SetValueAxisTickInterval Function

Sets the tick interval in the value axis

File

[libaria_widget_bar_graph.h](#)

C

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisTickInterval(laBarGraphWidget* graph,
    laBarGraphValueAxis axis, uint32_t interval);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
<code>laBarGraphWidget* graph</code>	the widget
<code>laBarGraphValueAxis axis</code>	the value axis index
<code>uint32_t interval</code>	tick interval in pixels

Function

`laResult laBarGraphWidget_SetValueAxisTickInterval(laBarGraphWidget* graph, laBarGraphValueAxis axis, uint32_t interval)`

laBarGraphWidget_SetValueAxisTicksPosition Function

Sets the position of the ticks in the value axis

File[libaria_widget_bar_graph.h](#)**C**

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisTicksPosition(laBarGraphWidget* graph,
laBarGraphValueAxis axis, laBarGraphTickPosition position);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
laBarGraphTickPosition position	the tick position

Function

```
laResult laBarGraphWidget_SetValueAxisTicksPosition(laBarGraphWidget* graph, laBarGraphValueAxis axis,
laBarGraphTickPosition position)
```

laBarGraphWidget_SetValueAxisTicksVisible Function

Shows/Hides the ticks in the value axis

File[libaria_widget_bar_graph.h](#)**C**

```
LIB_EXPORT laResult laBarGraphWidget_SetValueAxisTicksVisible(laBarGraphWidget* graph,
laBarGraphValueAxis axis, laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laBarGraphWidget* graph	the widget
laBarGraphValueAxis axis	the value axis index
laBool visible	shows the ticks if GFX_TRUE

Function

```
laResult laBarGraphWidget_SetValueAxisTicksVisible(laBarGraphWidget* graph, laBarGraphValueAxis axis, laBool visible)
```

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

[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
<code>laButtonWidget* btn</code>	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
<code>laButtonWidget* btn</code>	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_GetTextLineSpace Function

Returns the line spacing in pixels for the button text. If < 0, the ascent value of the string's font is used.

File

[libaria_widget_button.h](#)

C

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

Returns

[int32_t](#) - the line spacing in pixels

Parameters

Parameters	Description
laButtonWidget* btn	the button to reference

Function

[int32_t](#) laButtonWidget_GetTextLineSpace([laButtonWidget*](#) btn)

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 management 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_SetTextLineSpace Function

Sets the line space in pixels of the text in the button widget. A value < 0 sets the spacing to the ascent value of the string's font.

File

[libaria_widget_button.h](#)

C

```
LIB_EXPORT laResult laButtonWidget_SetTextLineSpace(laButtonWidget* btn, int32_t pixels);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laButtonWidget* btn	the button to modify
int32_t pixels	the line space, in pixels

Function

[laResult laButtonWidget_SetTextLineSpace\(laButtonWidget* btn, int32_t pixels\)](#)

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 management 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_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)
```

laCheckBoxWidget_SetImagePosition Function

Sets the image position of the check box

File

[libaria_widget_checkbox.h](#)

<code>laCheckBoxWidget_UncheckedEvent cb</code>	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
<code>laCheckBoxWidget* cbox</code>	the widget
<code>GFXU_ImageAsset* img</code>	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
<code>laCheckBoxWidget* cbox</code>	the widget
<code>laVAlignment align</code>	the valign value

Function

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

laCircleWidget_GetFilled Function

Gets the filled state of a circle widget

File

[libaria_widget_circle.h](#)

C

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

Returns

uint32_t

Parameters

Parameters	Description
laCircleWidget* cir	the widget

Function

[laBool laCircleWidget_GetFilled\(laCircleWidget* cir\)](#)

laCircleWidget_GetOrigin Function

Gets the origin coordinates 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_GetThickness Function

Gets the thickness of a circle widget

File

[libaria_widget_circle.h](#)

C

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

Returns

uint32_t

Parameters

Parameters	Description
laCircleWidget* cir	the widget

Function

uint32_t laCircleWidget_GetThickness([laCircleWidget*](#) cir)

laCircleWidget_New Function

Allocates memory for and initializes a new widget of this type. The application is responsible for the management 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_SetFilled Function

Sets the filled state of a circle widget

File

[libaria_widget_circle.h](#)

C

```
LIB_EXPORT laResult laCircleWidget_SetFilled(laCircleWidget* cir, laBool filled);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laCircleWidget* cir	the widget
laBool thickness	filled or not

Function

```
laResult laCircleWidget_SetFilled(laCircleWidget* cir, laBool filled)
```

laCircleWidget_SetOrigin Function

Sets the origin coordinates 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)`

laCircleWidget_SetThickness Function

Sets the thickness of a circle widget

File

[libaria_widget_circle.h](#)

C

```
LIB_EXPORT laResult laCircleWidget_SetThickness(laCircleWidget* cir, uint32_t thickness);
```

Returns

`laResult` - the operation result

Parameters

Parameters	Description
<code>laCircleWidget* cir</code>	the widget
<code>uint32_t thickness</code>	the desired thickness value

Function

`laResult laCircleWidget_SetThickness(laCircleWidget* cir, uint32_t thickness)`

laCircularGaugeWidget_AddAngularArc Function

Adds an 'angular arc' in the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_AddAngularArc(laCircularGaugeWidget* gauge, int32_t startAngle, int32_t endAngle, uint32_t radius, uint32_t thickness, laScheme* scheme);
```

Returns

`laResult`

Description

Angular arcs are drawn inside the gauge from the same origin/center but are not bound by the start/end angle/value of the gauge.

Parameters

Parameters	Description
<code>laCircularGaugeWidget* gauge</code>	the widget
<code>int32_t startAngle</code>	the start angle of the arc (relative to gauge starting angle)
<code>int32_t endAngle</code>	the end angle of the arc (relative to gauge starting angle)
<code>uint32_t radius</code>	the radius of the arc
<code>uint32_t thickness</code>	the fill thickness of the arc
<code>laScheme* scheme</code>	scheme used for drawing the arc

Function

`laResult laCircularGaugeWidget_AddAngularArc(laCircularGaugeWidget* gauge, int32_t startAngle,`

```
int32_t endAngle,
uint32_t radius,
uint32_t thickness,
laScheme* scheme)
```

laCircularGaugeWidget_AddMinorTickLabels Function

Adds minor tick labels in the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_AddMinorTickLabels(laCircularGaugeWidget* gauge,
int32_t startValue, int32_t endValue, uint32_t radius, laCircularGaugeWidgetLabelPosition
position, uint32_t interval, laScheme* scheme);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t startValue	the start value of the reference tick points (must be within gauge range)
int32_t endValue	the end value of the reference tick points (must be within gauge range)
uint32_t radius	the radius of the reference tick points
uint32_t position	the position of the label relative to the tick points
uint32_t interval	the interval between ticks
laScheme* scheme	scheme used for drawing the tick (uses foreground)

Function

```
IlaResult laCircularGaugeWidget_AddMinorTickLabels(laCircularGaugeWidget* gauge,
int32_t startValue,
int32_t endValue,
uint32_t radius,
laCircularGaugeWidgetLabelPosition position,
uint32_t interval,
laScheme* scheme)
```

laCircularGaugeWidget_AddMinorTicks Function

Adds minor ticks in the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_AddMinorTicks(laCircularGaugeWidget* gauge, int32_t
startValue, int32_t endValue, uint32_t radius, uint32_t length, uint32_t interval, laScheme*
scheme);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t startValue	the start value of the ticks (must be within gauge range)
int32_t endValue	the end value of the ticks (must be within gauge range)
uint32_t radius	the radius of the ticks
uint32_t length	the length of the ticks (drawn inward towards center)
uint32_t interval	the interval between ticks
laScheme* scheme	scheme used for drawing the tick (uses foreground)

Function

```
laCircularGaugeWidget_AddMinorTicks(laCircularGaugeWidget* gauge,
int32_t startValue,
int32_t endValue,
uint32_t radius,
uint32_t length,
uint32_t interval,
laScheme* scheme)
```

laCircularGaugeWidget_AddValueArc Function

Adds a 'value arc' in the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_AddValueArc(laCircularGaugeWidget* gauge, int32_t
startValue, int32_t endValue, uint32_t radius, uint32_t thickness, laScheme* scheme);
```

Returns

[laResult](#)

Description

Value arcs are drawn inside the gauge from the same origin/center but are bound by the start and end value of the gauge. A value arc that exceeds the start or end value of the gauge will not be drawn.

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t startAngle	the start angle of the arc (relative to gauge starting angle)
int32_t endAngle	the end angle of the arc (relative to gauge starting angle)
uint32_t radius	the radius of the arc
uint32_t thickness	the fill thickness of the arc
laScheme* scheme	scheme used for drawing the arc

Function

```
laResult laCircularGaugeWidget_AddValueArc(laCircularGaugeWidget* gauge,
int32_t startValue,
int32_t endValue,
uint32_t radius,
uint32_t thickness,
```

[laScheme*](#) scheme)

laCircularGaugeWidget_DeleteArcs Function

Deletes all arcs in the gauge widget

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_DeleteArcs(laCircularGaugeWidget* gauge);
```

Returns

[laResult](#)

Description

Deletes all arcs in the gauge widget

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

[laResult laCircularGaugeWidget_DeleteArcs\(laCircularGaugeWidget* gauge\)](#)

laCircularGaugeWidget_DeleteMinorTickLabels Function

Deletes all the minor tick labels in the gauge widget

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_DeleteMinorTickLabels(laCircularGaugeWidget* gauge);
```

Returns

[laResult](#)

Description

Deletes all the tick labels in the gauge widget

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

[laResult laCircularGaugeWidget_DeleteMinorTickLabels\(laCircularGaugeWidget* gauge\)](#)

laCircularGaugeWidget_DeleteMinorTicks Function

Deletes all the minor ticks in the gauge widget

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_DeleteMinorTicks(laCircularGaugeWidget* gauge);
```

Returns

[laResult](#)

Description

Deletes all the minor ticks in the gauge widget

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

[laResult](#) laCircularGaugeWidget_DeleteMinorTicks([laCircularGaugeWidget*](#) gauge)

laCircularGaugeWidget_GetCenterAngle Function

Returns the center angle of the circular gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT int32_t laCircularGaugeWidget_GetCenterAngle(laCircularGaugeWidget* gauge);
```

Returns

int32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

int32_t laCircularGaugeWidget_GetCenterAngle([laCircularGaugeWidget*](#) gauge)

laCircularGaugeWidget_GetCenterCircleRadius Function

Returns radius of the center circle

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT uint32_t laCircularGaugeWidget_GetCenterCircleRadius(laCircularGaugeWidget* gauge);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
uint32_t laCircularGaugeWidget_GetCenterCircleRadius(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_GetCenterCircleThickness Function

Returns thickness of the center circle

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT uint32_t laCircularGaugeWidget_GetCenterCircleThickness(laCircularGaugeWidget* gauge);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
uint32_t laCircularGaugeWidget_GetCenterCircleThickness(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_GetCenterCircleVisible Function

Returns GFX_TRUE if the center circle is visible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laBool laCircularGaugeWidget_GetCenterCircleVisible(laCircularGaugeWidget* gauge);
```

Returns

laBool

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
laBool laCircularGaugeWidget_GetCenterCircleVisible(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_GetDirection Function

Returns the direction of the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laCircularGaugeWidgetDir laCircularGaugeWidget_GetDirection(laCircularGaugeWidget*
```

```
gauge);
```

Returns

`laCircularGaugeWidgetDir`

Description

The direction is automatically set to CW if the center angle is negative, and CCW if the center angle is positive.

Parameters

Parameters	Description
<code>laCircularGaugeWidget* gauge</code>	the widget

Function

`laCircularGaugeWidgetDir laCircularGaugeWidget_GetDirection(laCircularGaugeWidget* gauge)`

laCircularGaugeWidget_GetEndValue Function

Returns the end value of the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT int32_t laCircularGaugeWidget_GetEndValue(laCircularGaugeWidget* gauge);
```

Returns

`int32_t`

Parameters

Parameters	Description
<code>laCircularGaugeWidget* gauge</code>	the widget

Function

`int32_t laCircularGaugeWidget_GetEndValue(laCircularGaugeWidget* gauge)`

laCircularGaugeWidget_GetHandRadius Function

Returns the radius/length of the gauge hand in pixels

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT uint32_t laCircularGaugeWidget_GetHandRadius(laCircularGaugeWidget* gauge);
```

Returns

`uint32_t`

Parameters

Parameters	Description
<code>laCircularGaugeWidget* gauge</code>	the widget

Function

`uint32_t laCircularGaugeWidget_GetHandRadius(laCircularGaugeWidget* gauge)`

laCircularGaugeWidget_GetHandVisible Function

Returns GFX_TRUE if the gauge hand is visible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laBool laCircularGaugeWidget_GetHandVisible(laCircularGaugeWidget* gauge);
```

Returns

laBool

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

[laBool laCircularGaugeWidget_GetHandVisible\(laCircularGaugeWidget* gauge\)](#)

laCircularGaugeWidget_GetRadius Function

Gets the radius of a gauge widget

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT uint32_t laCircularGaugeWidget_GetRadius(laCircularGaugeWidget* gauge);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

[uint32_t laCircularGaugeWidget_GetRadius\(laCircularGaugeWidget* gauge\)](#)

laCircularGaugeWidget_GetStartAngle Function

Returns the start angle of the circular gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT int32_t laCircularGaugeWidget_GetStartAngle(laCircularGaugeWidget* gauge);
```

Returns

int32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
int32_t laCircularGaugeWidget_GetStartAngle(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_GetStartValue Function

Returns the start value of the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT int32_t laCircularGaugeWidget_GetStartValue(laCircularGaugeWidget* gauge);
```

Returns

int32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
int32_t laCircularGaugeWidget_GetStartValue(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_GetTickLabelsVisible Function

Returns GFX_TRUE if the tick labels are visible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laBool laCircularGaugeWidget_GetTickLabelsVisible(laCircularGaugeWidget* gauge);
```

Returns

laBool

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
laBool laCircularGaugeWidget_GetTickLabelsVisible(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_GetTickLength Function

Returns the length of the ticks in the gauge in pixels

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT uint32_t laCircularGaugeWidget_GetTickLength(laCircularGaugeWidget* gauge);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

uint32_t laCircularGaugeWidget_GetTickLength(laCircularGaugeWidget* gauge)

laCircularGaugeWidget_GetTicksVisible Function

Returns GFX_TRUE if the ticks in the gauge are visible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laBool laCircularGaugeWidget_GetTicksVisible(laCircularGaugeWidget* gauge);
```

Returns

laBool

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

laBool laCircularGaugeWidget_GetTicksVisible(laCircularGaugeWidget* gauge)

laCircularGaugeWidget_GetTickCount Function

Returns the tick increment value in the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT int32_t laCircularGaugeWidget_GetTickCount(laCircularGaugeWidget* gauge);
```

Returns

int32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

int32_t laCircularGaugeWidget_GetTickCount(laCircularGaugeWidget* gauge)

laCircularGaugeWidget_GetValue Function

Returns the value of the gauge hand

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT int32_t laCircularGaugeWidget_GetValue(laCircularGaugeWidget* gauge);
```

Returns

int32_t

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget

Function

```
int32_t laCircularGaugeWidget_GetValue(laCircularGaugeWidget* gauge)
```

laCircularGaugeWidget_New Function

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

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laCircularGaugeWidget* laCircularGaugeWidget_New();
```

Returns

laCircularGaugeWidget*

Function

```
laCircularGaugeWidget* laCircularGaugeWidget_New()
```

laCircularGaugeWidget_SetCenterAngle Function

Sets the center angle of the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetCenterAngle(laCircularGaugeWidget* gauge, int32_t angle);
```

Returns

laResult

Description

A positive center angle draws the gauge, ticks, and hand in CCW, while a negative center angle draws in CW.

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t angle	center angle of the gauge

Function

[laResult laCircularGaugeWidget_SetCenterAngle\(laCircularGaugeWidget* gauge, int32_t angle\)](#)

laCircularGaugeWidget_SetCenterCircleRadius Function

Sets the center radius of the center circle

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetCenterCircleRadius(laCircularGaugeWidget* gauge,
uint32_t rad);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
uint32_t rad	radius of the center circle

Function

[laResult laCircularGaugeWidget_SetCenterCircleRadius\(laCircularGaugeWidget* gauge,
uint32_t rad\)](#)

laCircularGaugeWidget_SetCenterCircleThickness Function

Sets the thickness of the center circle

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetCenterCircleThickness(laCircularGaugeWidget* gauge,
uint32_t thickness);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
uint32_t thickness	thickness of the center circle

Function

[laResult laCircularGaugeWidget_SetCenterCircleThickness\(laCircularGaugeWidget* gauge,
uint32_t thickness\)](#)

laCircularGaugeWidget_SetCenterCircleVisible Function

Sets the center circle visible/invisible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetCenterCircleVisible(laCircularGaugeWidget* gauge,  
laBool visible);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
laBool visible	sets visible if GFX_TRUE

Function

```
laResult laCircularGaugeWidget_SetCenterCircleVisible(laCircularGaugeWidget* gauge,  
laBool visible)
```

laCircularGaugeWidget_SetEndValue Function

Sets the end value of the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetEndValue(laCircularGaugeWidget* gauge, int32_t  
value);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t value	end value of the gauge

Function

```
laResult laCircularGaugeWidget_SetEndValue(laCircularGaugeWidget* gauge,  
int32_t value)
```

laCircularGaugeWidget_SetHandRadius Function

Sets the radius/length of the hand

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetHandRadius(laCircularGaugeWidget* gauge, uint32_t rad);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
uint32_t length	length of the hand in pixels

Function

```
laResult laCircularGaugeWidget_SetHandRadius(laCircularGaugeWidget* gauge,  
uint32_t length)
```

laCircularGaugeWidget_SetHandVisible Function

Sets the hand visible/invisible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetHandVisible(laCircularGaugeWidget* gauge, laBool visible);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
laBool visible	hand is visible if GFX_TRUE

Function

```
laResult laCircularGaugeWidget_SetHandVisible(laCircularGaugeWidget* gauge,  
laBool visible)
```

laCircularGaugeWidget_SetRadius Function

Sets the radius of a gauge widget

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetRadius(laCircularGaugeWidget* gauge, uint32_t rad);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
uint32_t red	the desired radius value

Function

[laResult laCircularGaugeWidget_SetRadius\(laCircularGaugeWidget* gauge, uint32_t rad\)](#)

laCircularGaugeWidget_SetStartAngle Function

Sets the start angle of the gauge.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetStartAngle(laCircularGaugeWidget* gauge, int32_t
angle);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t angle	start angle of the gauge

Function

[laResult laCircularGaugeWidget_SetStartAngle\(laCircularGaugeWidget* gauge, int32_t angle\)](#)

laCircularGaugeWidget_SetStartValue Function

Sets the start value of the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetStartValue(laCircularGaugeWidget* gauge, int32_t
value);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t value	start value of the gauge

Function

[laResult laCircularGaugeWidget_SetStartValue\(laCircularGaugeWidget* gauge,
int32_t value\)](#)

laCircularGaugeWidget_SetStringTable Function

Sets the string table to be used for the tick labels

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetStringTable(laCircularGaugeWidget* gauge,  
GFXU_StringTableAsset * stringTable);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
GFXU_StringTableAsset * stringTable	pointer to the string table

Function

```
laResult laCircularGaugeWidget_SetStringTable(laCircularGaugeWidget* gauge,  
GFXU_StringTableAsset * stringTable)
```

laCircularGaugeWidget_SetTickLabelsVisible Function

Sets the tick labels visible/invisible

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetTickLabelsVisible(laCircularGaugeWidget* gauge,  
laBool visible);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
laBool visible	tick labels are visible if GFX_TRUE

Function

```
laResult laCircularGaugeWidget_SetTickLabelsVisible(laCircularGaugeWidget* gauge,  
laBool visible)
```

laCircularGaugeWidget_SetTickLength Function

Sets the length of the ticks

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetTickLength(laCircularGaugeWidget* gauge, uint32_t length);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
uint32_t length	length of the ticks in pixels

Function

```
laResult laCircularGaugeWidget_SetTickLength(laCircularGaugeWidget* gauge,  
uint32_t length)
```

laCircularGaugeWidget_SetTicksLabelsStringID Function

Sets the ID of the string character superset to be used for the tick labels

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetTicksLabelsStringID(laCircularGaugeWidget* gauge,  
uint32_t stringID);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
uint32_t stringID	string ID

Function

```
laResult laCircularGaugeWidget_SetTicksLabelsStringID(laCircularGaugeWidget* gauge,  
uint32_t stringID)
```

laCircularGaugeWidget_SetTicksVisible Function

Sets the increments/distance between ticks

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetTicksVisible(laCircularGaugeWidget* gauge, laBool  
visible);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t value	the distance between ticks

Function

```
laCircularGaugeWidget_SetTicksVisible(laCircularGaugeWidget* gauge,
                                      laBool visible)
```

laCircularGaugeWidget_SetTickCount Function

Sets the increments/distance between ticks

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetTickCount(laCircularGaugeWidget* gauge, int32_t
                                                       value);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t value	the distance between ticks

Function

```
laResult laCircularGaugeWidget_SetTickCount(laCircularGaugeWidget* gauge,
                                            int32_t value)
```

laCircularGaugeWidget_SetValue Function

Sets the value of the gauge hand

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetValue(laCircularGaugeWidget* gauge, int32_t value);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
int32_t value	value of the gauge hand

Function

```
laResult laCircularGaugeWidget_SetValue(laCircularGaugeWidget* gauge,
                                         int32_t value)
```

laCircularGaugeWidget_SetValueChangedEventCallback Function

Sets the function to be called back when the gauge value changes.

File

[libaria_widget_circular_gauge.h](#)

C

```
LIB_EXPORT laResult laCircularGaugeWidget_SetValueChangedEventCallback(laCircularGaugeWidget* gauge, laCircularGaugeWidget_ValueChangedEvent cb);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularGaugeWidget* gauge	the widget
laCircularGaugeWidget_ValueChangedEvent cb	callback function

Function

```
laResult laCircularGaugeWidget_SetValueChangedEventCallback(laCircularGaugeWidget* gauge, laCircularGaugeWidget_ValueChangedEvent cb)
```

laCircularSliderWidget_GetArcRadius Function

Returns the radius of an arc in the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetArcRadius(laCircularSliderWidget* slider, laCircularSliderWidgetArcType type);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidgetArcType type	the type of arc

Function

```
uint32_t laCircularSliderWidget_GetArcRadius(laCircularSliderWidget* slider, laCircularSliderWidgetArcType type)
```

laCircularSliderWidget_GetArcScheme Function

Returns the scheme of an arc in the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laScheme * laCircularSliderWidget_GetArcScheme(laCircularSliderWidget* slider,
laCircularSliderWidgetArcType type);
```

Returns

[laScheme *](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidgetArcType type	the type of arc

Function

[laScheme * laCircularSliderWidget_GetArcScheme\(laCircularSliderWidget* slider, laCircularSliderWidgetArcType type\)](#)

laCircularSliderWidget_GetArcThickness Function

Returns the thickness of an arc in the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetArcThickness(laCircularSliderWidget* slider,
laCircularSliderWidgetArcType type);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidgetArcType type	the type of arc

Function

[uint32_t laCircularSliderWidget_GetArcThickness\(laCircularSliderWidget* slider, laCircularSliderWidgetArcType type\)](#)

laCircularSliderWidget_GetArcVisible Function

Returns true if the specified arc is visible

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laBool laCircularSliderWidget_GetArcVisible(laCircularSliderWidget* slider,
laCircularSliderWidgetArcType type);
```

Returns

laBool

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget

laCircularSliderWidgetArcType type	the type of arc
------------------------------------	-----------------

Function

```
laBool laCircularSliderWidget_GetArcVisible(laCircularSliderWidget* slider, laCircularSliderWidgetArcType type)
```

laCircularSliderWidget_GetDirection Function

Returns direction of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laCircularSliderWidgetDir  
laCircularSliderWidget_GetDirection(laCircularSliderWidget* slider);
```

Returns

laCircularSliderWidgetDir

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget

Function

```
laCircularSliderWidgetDir laCircularSliderWidget_GetDirection(laCircularSliderWidget* slider)
```

laCircularSliderWidget_GetEndValue Function

Gets the end value of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetEndValue(laCircularSliderWidget* slider);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget

Function

```
uint32_t laCircularSliderWidget_GetEndValue(laCircularSliderWidget* slider)
```

laCircularSliderWidget_GetOrigin Function

Gets the origin coordinates of a slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_GetOrigin(laCircularSliderWidget* slider, int32_t*  
x, int32_t* y);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laCircularSliderWidget* slider	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](#) `laCircularSliderWidget_GetOrigin(laCircularSliderWidget* slider, int32_t* x, int32_t* y)`

laCircularSliderWidget_GetRadius Function

Gets the radius of a slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetRadius(laCircularSliderWidget* slider);
```

Returns

uint32_t

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget

Function

`uint32_t laCircularSliderWidget_GetRadius(laCircularSliderWidget* slider)`

laCircularSliderWidget_GetRoundEdges Function

Returns true if the slider has rounded edges

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laBool laCircularSliderWidget_GetRoundEdges(laCircularSliderWidget* slider);
```

Returns

[laBool](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget

Function

`laBool laCircularSliderWidget_GetRoundEdges(laCircularSliderWidget* slider)`

laCircularSliderWidget_GetStartAngle Function

Returns the start angle of a slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetStartAngle(laCircularSliderWidget* slider);
```

Returns

`uint32_t`

Parameters

Parameters	Description
<code>laCircularSliderWidget* slider</code>	the widget

Function

`int32_t laCircularSliderWidget_GetStartAngle(laCircularSliderWidget* slider)`

laCircularSliderWidget_GetStartValue Function

Gets the start value of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetStartValue(laCircularSliderWidget* slider);
```

Returns

`uint32_t`

Parameters

Parameters	Description
<code>laCircularSliderWidget* slider</code>	the widget

Function

`uint32_t laCircularSliderWidget_GetStartValue(laCircularSliderWidget* slider)`

laCircularSliderWidget_GetStickyButton Function

Returns true if the slider button sticks to the start/end value

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laBool laCircularSliderWidget_GetStickyButton(laCircularSliderWidget* slider);
```

Returns

`laBool`

Parameters

Parameters	Description
<code>laCircularSliderWidget* slider</code>	the widget

Function

`laBool laCircularSliderWidget_GetStickyButton(laCircularSliderWidget* slider)`

laCircularSliderWidget_GetTouchOnButtonOnly Function

Returns true if the slider slider only responds to touch inside the button area

File

`libaria_widget_circular_slider.h`

C

```
LIB_EXPORT laBool laCircularSliderWidget_GetTouchOnButtonOnly(laCircularSliderWidget* slider);
```

Returns

`laBool`

Parameters

Parameters	Description
<code>laCircularSliderWidget* slider</code>	the widget

Function

`laBool laCircularSliderWidget_GetTouchOnButtonOnly(laCircularSliderWidget* slider)`

laCircularSliderWidget_GetValue Function

Gets the value of the slider widget

File

`libaria_widget_circular_slider.h`

C

```
LIB_EXPORT uint32_t laCircularSliderWidget_GetValue(laCircularSliderWidget* slider);
```

Returns

`uint32_t`

Parameters

Parameters	Description
<code>laCircularSliderWidget* slider</code>	the widget

Function

`uint32_t laCircularSliderWidget_GetValue(laCircularSliderWidget* slider)`

laCircularSliderWidget_New Function

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

until the widget is added to a widget tree.

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laCircularSliderWidget* laCircularSliderWidget_New();
```

Returns

laCircularSliderWidget*

Function

[laCircularSliderWidget* laCircularSliderWidget_New\(\)](#)

laCircularSliderWidget_SetArcRadius Function

Sets the start value of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetArcRadius(laCircularSliderWidget* slider,
laCircularSliderWidgetArcType type, uint32_t radius);
```

Returns

laResult

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget uint32_t value

Function

[laResult laCircularSliderWidget_SetStartValue\(laCircularSliderWidget* slider, uint32_t value\)](#)

laCircularSliderWidget_SetArcScheme Function

Sets the scheme of the specified arc

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetArcScheme(laCircularSliderWidget* slider,
laCircularSliderWidgetArcType type, laScheme * scheme);
```

Returns

laBool

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidgetArcType type	the type of arc
laScheme * scheme	scheme

Function

```
laResult laCircularSliderWidget_SetArcScheme(laCircularSliderWidget* slider,  

laCircularSliderWidgetArcType type,  

laScheme * scheme)
```

laCircularSliderWidget_SetArcThickness Function

Sets the thickness of an arc in the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetArcThickness(laCircularSliderWidget* slider,  

laCircularSliderWidgetArcType type, uint32_t thickness);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget * slider	the widget
laCircularSliderWidgetArcType type	the type of arc
uint32_t thickness	the thickness of the arc

Function

```
laResult laCircularSliderWidget_SetArcThickness(laCircularSliderWidget* slider,  

laCircularSliderWidgetArcType type,  

uint32_t thickness)
```

laCircularSliderWidget_SetArcVisible Function

Shows/Hides the specified arc visible

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetArcVisible(laCircularSliderWidget* slider,  

laCircularSliderWidgetArcType type, laBool visible);
```

Returns

[laBool](#)

Parameters

Parameters	Description
laCircularSliderWidget * slider	the widget
laCircularSliderWidgetArcType type	the type of arc
laBool visible	show/hide

Function

```
laResult laCircularSliderWidget_SetArcVisible(laCircularSliderWidget* slider,  

laCircularSliderWidgetArcType type,
```

[laBool](#) visible)

laCircularSliderWidget_SetDirection Function

Sets the direction of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetDirection(laCircularSliderWidget* slider,
laCircularSliderWidgetDir dir);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidgetDir dir	direction

Function

[laResult](#) laCircularSliderWidget_SetDirection([laCircularSliderWidget*](#) slider, [laCircularSliderWidgetDir](#) dir)

laCircularSliderWidget_SetEndValue Function

Sets the end value of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetEndValue(laCircularSliderWidget* slider, uint32_t
value);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget uint32_t value

Function

[laResult](#) laCircularSliderWidget_SetEndValue([laCircularSliderWidget*](#) slider, [uint32_t](#) value)

laCircularSliderWidget_SetOrigin Function

Sets the origin coordinates of a slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetOrigin(laCircularSliderWidget* slider, int32_t x,
```

```
int32_t y);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
int32_t x	the desired x origin coordinate
int32_t y	the desired y origin coordinate

Function

[laResult](#) laCircularSliderWidget_SetOrigin(laCircularSliderWidget* slider, int32_t x, int32_t y)

laCircularSliderWidget_SetPressedEventCallback Function

Sets the function that gets called when the slider button is pressed

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetPressedEventCallback(laCircularSliderWidget* slider, laCircularSliderWidget_PressedEvent cb);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidget_PressedEvent cb	the callback function

Function

laCircularSliderWidget_SetPressedEventCallback(laCircularSliderWidget* slider, laCircularSliderWidget_ValueChangedEvent cb)

laCircularSliderWidget_SetRadius Function

Sets the radius of a slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetRadius(laCircularSliderWidget* slider, uint32_t rad);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget

uint32_t red	the desired radius value
--------------	--------------------------

Function

```
laResult laCircularSliderWidget_SetRadius(laCircularSliderWidget* slider, uint32_t rad)
```

laCircularSliderWidget_SetReleasedEventCallback Function

Sets the function that gets called when the slider button is released

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetReleasedEventCallback(laCircularSliderWidget* slider, laCircularSliderWidget_ReleasedEvent cb);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidget_ReleasedEvent cb	the callback function

Function

```
laCircularSliderWidget_SetReleasedEventCallback(laCircularSliderWidget* slider, laCircularSliderWidget_ReleasedEvent cb)
```

laCircularSliderWidget_SetRoundEdges Function

If round = true, the slider active area edges are round

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetRoundEdges(laCircularSliderWidget* slider, laBool round);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget laBool round

Function

```
laResult laCircularSliderWidget_SetRoundEdges(laCircularSliderWidget* slider, laBool round)
```

laCircularSliderWidget_SetStartAngle Function

Sets the start angle of a slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetStartAngle(laCircularSliderWidget* slider,
uint32_t angle);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
int32_t angle	the desired start angle value

Function

[laResult](#) [laCircularSliderWidget_SetStartAngle](#)(laCircularSliderWidget* slider, int32_t angle)

laCircularSliderWidget_SetStartValue Function

Sets the start value of the slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetStartValue(laCircularSliderWidget* slider,
uint32_t value);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget uint32_t value

Function

[laResult](#) [laCircularSliderWidget_SetStartValue](#)(laCircularSliderWidget* slider, uint32_t value)

laCircularSliderWidget_SetStickyButton Function

If snap = true, the slider button sticks to the start/end value of the slider

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetStickyButton(laCircularSliderWidget* slider,
laBool sticky);
```

Returns

[laResult](#)

Description

When sticky is enabled, the slider button will not immediately wrap past the start or end value if the button is dragged past the start angle. If the button is dragged past the threshold of 90 degrees beyond the start angle, only then will the slider value wrap around.

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laBool sticky	set button as sticky

Function

`laResult laCircularSliderWidget_SetStickyButton(laCircularSliderWidget* slider, laBool c)`

laCircularSliderWidget_SetTouchOnButtonOnly Function

If buttonOnly = true, the slider will only respond to touches inside the button area

File

`libaria_widget_circular_slider.h`

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetTouchOnButtonOnly(laCircularSliderWidget* slider,
laBool buttonOnly);
```

Returns

`laResult`

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laBool buttonOnly	set touch to button only

Function

`laResult laCircularSliderWidget_SetTouchOnButtonOnly(laCircularSliderWidget* slider, laBool buttonOnly)`

laCircularSliderWidget_SetValue Function

Sets the value of the slider widget

File

`libaria_widget_circular_slider.h`

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetValue(laCircularSliderWidget* slider, uint32_t
value);
```

Returns

`laResult`

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget uint32_t value

Function

`laResult laCircularSliderWidget_SetValue(laCircularSliderWidget* slider, uint32_t value)`

laCircularSliderWidget_SetValueChangedEventCallback Function

Sets the function that gets called when the slider value changes

File

[libaria_widget_circular_slider.h](#)

C

```
LIB_EXPORT laResult laCircularSliderWidget_SetValueChangedEventCallback(laCircularSliderWidget*
slider, laCircularSliderWidget_ValueChangedEvent cb);
```

Returns

[laResult](#)

Parameters

Parameters	Description
laCircularSliderWidget* slider	the widget
laCircularSliderWidget_ValueChangedEvent cb	the callback function

Function

```
laCircularSliderWidget_SetValueChangedEventCallback(laCircularSliderWidget* slider,
laCircularSliderWidget_ValueChangedEvent cb)
```

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
<code>GFX_ColorMode</code>	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
<code>laContext*</code>	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 receiving 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_GetPreemptionLevel Function

Returns the preemption level for the screen

File

[libaria_context.h](#)

C

```
LIB_EXPORT laPreemptionLevel laContext_GetPreemptionLevel();
```

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_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\(uint32_t idx\)](#)

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
<code>laWidget*</code>	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_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\)](#)

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)
```

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 management 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 management 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 management 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 management 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 align	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)

laImagePlusWidget_GetImage Function

Gets the image asset pointer for the widget.

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT GFXU_ImageAsset* laImagePlusWidget_GetImage(laImagePlusWidget* img);
```

Returns

[GFXU_ImageAsset](#)* - the image asset pointer

Parameters

Parameters	Description
laImagePlusWidget* img	the widget

Function

[GFXU_ImageAsset](#)* laImagePlusWidget_GetImage(laImagePlusWidget* img)

laImagePlusWidget_GetInteractive Function

Returns true if the widget is configured to respond to input events

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laBool laImagePlusWidget_GetInteractive(laImagePlusWidget* img);
```

Returns

laBool - the value of the interactive flag

Description

This widget can be configured to respond to input events. It can react to single and double touch events. These events will directly modify the transform state and cause the image to translate and resize as desired.

Parameters

Parameters	Description
lalImagePlusWidget* img	the widget

Function

[laBool lalImagePlusWidget_GetInteractive\(lalImagePlusWidget* img\)](#)

lalImagePlusWidget_GetPreserveAspectEnabled Function

Returns the boolean value of the 'preserve aspect' property

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laBool laImagePlusWidget_GetPreserveAspectEnabled(laImagePlusWidget* img);
```

Returns

[laBool](#) - the value of the aspect flag

Description

Returns the boolean value of the 'preserve aspect' property. If the stretch flag is enabled then this flag will cause the resized image to fill as much of the container widget as possible while still preserving its original aspect ratio.

Parameters

Parameters	Description
lalImagePlusWidget* img	the widget

Function

[laBool lalImagePlusWidget_GetPreserveAspectEnabled\(lalImagePlusWidget* img\)](#)

lalImagePlusWidget_GetResizeFilter Function

Returns the resize filter setting for this image widget

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laImagePlusWidget_ResizeFilter laImagePlusWidget_GetResizeFilter(laImagePlusWidget* img);
```

Returns

[lalImagePlusWidget_ResizeFilter](#) - the filter setting

Description

Returns the resize filter setting for this image widget. This flag affects the speed and quality of the image resize operation.

Parameters

Parameters	Description
lalImagePlusWidget* img	the widget

Function

[lalImagePlusWidget_ResizeFilter lalImagePlusWidget_GetResizeFilter\(lalImagePlusWidget* img\)](#)

laImagePlusWidget_GetStretchEnabled Function

Returns the boolean value of the 'stretch to fit' property

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laBool laImagePlusWidget_GetStretchEnabled(laImagePlusWidget* img);
```

Returns

laBool - the value of the resize flag

Description

Returns the boolean value of the 'stretch to fit' property. This flag will cause the image to be stretched to fill the space of the container widget. Widget margins are still considered.

Parameters

Parameters	Description
laWidget* wgt	the widget

Function

`laBool laImagePlusWidget_GetStretchEnabled(laImagePlusWidget* img)`

laImagePlusWidget_GetTransformHeight Function

Returns the image scale height coefficient

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT int32_t laImagePlusWidget_GetTransformHeight(laImagePlusWidget* img);
```

Returns

int32_t - the scale width coordinate value in pixels

Description

Returns the image scale height coefficient

Parameters

Parameters	Description
laImagePlusWidget* img	the widget

Function

`int32_t laImagePlusWidget_GetTransformHeight(laImagePlusWidget* img)`

laImagePlusWidget_GetTransformWidth Function

Returns the image scale width coefficient

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT int32_t laImagePlusWidget_GetTransformWidth(laImagePlusWidget* img);
```

Returns

int32_t - the scale width coordinate value in pixels

Description

Returns the image scale width coefficient

Parameters

Parameters	Description
laImagePlusWidget* img	the widget

Function

```
int32_t laImagePlusWidget_GetTransformWidth(laImagePlusWidget* img)
```

laImagePlusWidget_GetTransformX Function

Returns the image transform x coefficient

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT int32_t laImagePlusWidget_GetTransformX(laImagePlusWidget* img);
```

Returns

int32_t - the x translation coefficient

Description

Returns the image transform x coefficient

Parameters

Parameters	Description
laImagePlusWidget* img	the widget

Function

```
int32_t laImagePlusWidget_GetTransformX(laImagePlusWidget* img)
```

laImagePlusWidget_GetTransformY Function

Returns the image transform y coefficient

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT int32_t laImagePlusWidget_GetTransformY(laImagePlusWidget* img);
```

Returns

int32_t - the y translation coefficient

Description

Returns the image transform y coefficient

Parameters

Parameters	Description
lalimagePlusWidget* img	the widget

Function

`int32_t lalimagePlusWidget_GetTransformY(lalimagePlusWidget* img)`

lalimagePlusWidget_New Function

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

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laImagePlusWidget* laImagePlusWidget_New();
```

Returns

`lalimagePlusWidget* - the widget`

Function

`lalimagePlusWidget* lalimagePlusWidget_New()`

lalimagePlusWidget_ResetTransform Function

Resets the image transform values to zero

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_ResetTransform(laImagePlusWidget* img);
```

Returns

`laResult - result of the operation`

Description

Resets the image transform values to zero

Parameters

Parameters	Description
lalimagePlusWidget* img	the widget

Function

`laResult lalimagePlusWidget_ResetTransform(lalimagePlusWidget* img)`

lalimagePlusWidget_SetImage Function

Sets the image asset pointer for the widget.

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetImage(laImagePlusWidget* img, GFXU_ImageAsset* imgAst);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
GFXU_ImageAsset* imgAst	the image asset pointer

Function

```
laResult laImagePlusWidget_SetImage(laImagePlusWidget* img,  
                                  GFXU_ImageAsset* imgAst)
```

laImagePlusWidget_SetInteractive Function

Sets the widget interactive flag

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetInteractive(laImagePlusWidget* img, laBool  
interactive);
```

Returns

[laResult](#) - the operation result

Description

Sets the widget interactive flag

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
laBool	the desired flag value

Function

```
laResult laImagePlusWidget_SetInteractive(laImagePlusWidget* img, laBool interactive)
```

laImagePlusWidget_SetPreserveAspectEnabled Function

Sets the boolean value of the 'preserve aspect' property

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetPreserveAspectEnabled(laImagePlusWidget* img, laBool  
enable);
```

Returns

[laResult](#) - the operation result

Description

Sets the boolean value of the 'preserve aspect' property

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
laBool enable	the desired flag value

Function

[laResult laImagePlusWidget_SetStretchEnabled\(laImagePlusWidget* img, laBool enable\)](#)

laImagePlusWidget_SetResizeFilter Function

Sets the resize filter value of the widget

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetResizeFilter(laImagePlusWidget* img,
laImagePlusWidget_ResizeFilter filter);
```

Returns

[laResult - the operation result](#)

Description

Sets the resize filter value of the widget

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
laImagePlusWidget_ResizeFilter filter	the desired filter

Function

[laResult laImagePlusWidget_SetResizeFilter\(laImagePlusWidget* img, laImagePlusWidget_ResizeFilter filter\)](#)

laImagePlusWidget_SetStretchEnabled Function

Sets the boolean value of the stretch property

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetStretchEnabled(laImagePlusWidget* img, laBool enable);
```

Returns

[laResult - the operation result](#)

Description

Sets the boolean value of the stretch property

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
laBool	the desired flag value

Function

[laResult laImagePlusWidget_SetStretchEnabled\(laImagePlusWidget* img, laBool enable\)](#)

laImagePlusWidget_SetTransformHeight Function

Sets the image scale height coefficient. This value is in pixels not percentage

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetTransformHeight(laImagePlusWidget* img, int32_t
height);
```

Returns

[laResult - result of the operation](#)

Description

Sets the image scale height coefficient. This value is in pixels not percentage

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
int32_t height	the desired height value, must be > 0

Function

[laResult laImagePlusWidget_SetTransformHeight\(laImagePlusWidget* img,
int32_t height\)](#)

laImagePlusWidget_SetTransformWidth Function

Sets the image scale width coefficient. This value is in pixels not percentage

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetTransformWidth(laImagePlusWidget* img, int32_t width);
```

Returns

[laResult - result of the operation](#)

Description

Sets the image scale width coefficient. This value is in pixels not percentage

Parameters

Parameters	Description
laImagePlusWidget* img	the widget

int32_t width	the desired width value, must be > 0
---------------	--------------------------------------

Function

```
laResult laImagePlusWidget_SetTransformWidth(laImagePlusWidget* img,
int32_t width)
```

laImagePlusWidget_SetTransformX Function

Sets the image transform x coefficient

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetTransformX(laImagePlusWidget* img, int32_t x);
```

Returns

laResult - result of the operation

Description

Sets the image transform x coefficient

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
int32_t x	the desired x value

Function

```
laResult laImagePlusWidget_SetTransformX(laImagePlusWidget* img, int32_t x)
```

laImagePlusWidget_SetTransformY Function

Sets the image transform y coefficient

File

[libaria_widget_imageplus.h](#)

C

```
LIB_EXPORT laResult laImagePlusWidget_SetTransformY(laImagePlusWidget* img, int32_t y);
```

Returns

laResult - result of the operation

Description

Sets the image transform y coefficient

Parameters

Parameters	Description
laImagePlusWidget* img	the widget
int32_t y	the desired y value

Function

```
laResult laImagePlusWidget_SetTransformY(laImagePlusWidget* img, int32_t y)
```

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 management 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 management 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
<code>laImageSequenceWidget* img</code>	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
<code>laImageSequenceWidget* img</code>	the widget
<code>uint32_t idx</code>	the index
<code>GFXU_ImageAsset* imgAst</code>	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
<code>lalimageSequenceWidget* img</code>	the widget
<code>lalimageSequenceImageChangedEvent_FnPtr cb</code>	a valid callback pointer or NULL

Function

```
laResult lalimageSequenceWidget_SetImageChangedEventCallback(lalimageSequenceWidget* img,
                                                               lalimageSequenceImageChangedEvent_FnPtr cb)
```

lalimageSequenceWidget_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
<code>lalimageSequenceWidget* img</code>	the widget
<code>uint32_t count</code>	the desired number of entries

Function

```
laResult lalimageSequenceWidget_SetImageCount(lalimageSequenceWidget* img,
                                              uint32_t count)
```

lalimageSequenceWidget_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
<code>laImageSequenceWidget* img</code>	the widget
<code>uint32_t idx</code>	the index
<code>uint32_t delay</code>	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
<code>laImageSequenceWidget* img</code>	the widget
<code>uint32_t idx</code>	the index
<code>laHAlignment align</code>	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
lalImageSequenceWidget* img	the widget
uint32_t idx	the index
laVAlignment align	the vertical alignment setting

Function

```
laResult lalImageSequenceWidget_SetImageVAlignment(lalImageSequenceWidget* img,
    uint32_t idx,
    laVAlignment align)
```

lalImageSequenceWidget_SetRepeat Function

Sets the repeat flag for the widget

File

[libaria_widget_imagesequence.h](#)

C

```
LIB_EXPORT laResult lalImageSequenceWidget_SetRepeat(lalImageSequenceWidget* img, laBool repeat);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
lalImageSequenceWidget* img	the widget
laBool repeat	the desired repeat setting

Function

```
laResult lalImageSequenceWidget_SetRepeat(lalImageSequenceWidget* img,
    laBool repeat)
```

lalImageSequenceWidget_ShowImage Function

Sets the active display index to the indicated value.

File

[libaria_widget_imagesequence.h](#)

C

```
LIB_EXPORT laResult lalImageSequenceWidget_ShowImage(lalImageSequenceWidget* img, uint32_t idx);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
lalImageSequenceWidget* img	the widget
uint32_t idx	the desired index

Function

```
laResult lalImageSequenceWidget_ShowImage(lalImageSequenceWidget* 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
<code>lalimageSequenceWidget* img</code>	the widget

Function

`laResult lalimageSequenceWidget_Stop(lalimageSequenceWidget* img)`

lalimageWidget_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
<code>lalimageWidget* img</code>	the widget

Function

`laHAlignment lalimageWidget_GetHAlignment(lalimageWidget* img)`

lalimageWidget_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
<code>lalimageWidget* img</code>	the widget

Function

`GFXU_ImageAsset* lalimageWidget_GetImage(lalimageWidget* img)`

lalimageWidget_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 management 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
laImageWidget* 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 align	the vertical alignment setting

Function

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

laInitialize Function

Function to perform basic initialization of the Aria library state.

File

[libaria.h](#)

C

```
LIB_EXPORT laResult laInitialize();
```

Description

Type: [laResult](#) laInitialize()

Function to perform basic initialization of the Aria library state. Must be called before any other library operations are performed.

Remarks

Returns: [laResult](#) - the result of the initialization operation

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
<code>laKeyPadWidget* pad</code>	the widget
<code>uint32_t row</code>	the indicated row
<code>uint32_t col</code>	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
<code>laKeyPadWidget* pad</code>	the widget
<code>uint32_t row</code>	the indicated row
<code>uint32_t col</code>	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 management 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_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)
```

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 interactability 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_GetTextLineSpace Function

Returns the line spacing in pixels for the label text. If < 0, the ascent value of the string's font is used.

File

[libaria_widget_label.h](#)

C

```
LIB_EXPORT int32_t laLabelWidget_GetTextLineSpace(laLabelWidget* lbl);
```

Returns

int32_t - the line spacing in pixels

Parameters

Parameters	Description
laLabelWidget* lbl	the label to reference

Function

int32_t laLabelWidget_GetTextLineSpace([laLabelWidget](#)* lbl)

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 management 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_SetTextLineSpace Function

Sets the line space in pixels of the text in the label widget. A value < 0 sets the spacing to the ascent value of the string's font.

File

[libaria_widget_label.h](#)

C

```
LIB_EXPORT laResult laLabelWidget_SetTextLineSpace(laLabelWidget* lbl, int32_t pixels);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laLabelWidget* lbl	the label to modify
int32_t pixels	the line space, in pixels

Function

[laResult laLabelWidget_SetTextLineSpace\(\[laLabelWidget\]\(#\)* lbl, int32_t pixels\)](#)

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_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)
```

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_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_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_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_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
<code>const laLayer* layer</code>	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
<code>const laLayer* layer</code>	the layer

Function

`laBool laLayer_GetInputRectLocked(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_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_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_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
<code>const laLayer* layer</code>	the layer

Function

`laResult laLayer_SetAllowInputPassthrough(laLayer* layer, laBool enable)`

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
<code>laLayer* layer</code>	the layer
<code>uint32_t amount</code>	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_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)`

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)

[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)
```

laLineGraphWidget_AddCategory Function

Adds a category to the graph

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_AddCategory(laLineGraphWidget* graph, uint32_t * id);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t * id	destination of the ID of the new category

Function

```
laResult laLineGraphWidget_AddCategory(laLineGraphWidget* graph, uint32_t * id)
```

laLineGraphWidget_AddDataToSeries Function

Adds a data (value) to the specified series at categoryID index

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_AddDataToSeries(laLineGraphWidget* graph, uint32_t
seriesID, int32_t value, uint32_t * index);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t seriesID	the series ID
int32_t value	the value
uint32_t * index	the destination to return the index of the added data

Function

```
laResult laLineGraphWidget_AddDataToSeries(laLineGraphWidget* graph, uint32_t seriesID, uint32_t categoryID, int32_t value)
```

laLineGraphWidget_AddSeries Function

Adds a series to the graph

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_AddSeries(laLineGraphWidget* graph, uint32_t * seriesID);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t * seriesID	destination of the returned series ID

Function

```
laResult laLineGraphWidget_AddSeries(laLineGraphWidget* graph, uint32_t * seriesID)
```

laLineGraphWidget_DestroyAll Function

Destroys data, series and categories and frees the memory allocated

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_DestroyAll(laLineGraphWidget* graph);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[laResult laLineGraphWidget_DestroyAll\(laLineGraphWidget* graph\)](#)

laLineGraphWidget_GetCategoryAxisLabelsVisible Function

Returns GFX_TRUE if the category axis labels are visible

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetCategoryAxisLabelsVisible(laLineGraphWidget* graph);
```

Returns

laBool - GFX_TRUE if the category axis labels are visible

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[laBool laLineGraphWidget_GetCategoryAxisLabelsVisible\(laLineGraphWidget* graph\)](#)

laLineGraphWidget_GetCategoryAxisTicksPosition Function

Returns the position of the ticks in the category axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laLineGraphTickPosition
laLineGraphWidget_GetCategoryAxisTicksPosition(laLineGraphWidget* graph);
```

Returns

laLineGraphTickPosition - position of the ticks in the category axis

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[laLineGraphTickPosition laLineGraphWidget_GetCategoryAxisTicksPosition\(laLineGraphWidget* graph\)](#)

laLineGraphWidget_GetCategoryAxisTicksVisible Function

Returns GFX_TRUE if the category axis ticks are visible

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laBool laLineGraphWidget_GetCategoryAxisTicksVisible(laLineGraphWidget* graph);
```

Returns

laBool - GFX_TRUE if the category axis ticks are visible

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[laBool laLineGraphWidget_GetCategoryAxisTicksVisible\(laLineGraphWidget* graph\)](#)

laLineGraphWidget_GetCategoryText Function

Gets a copy of the string used to label the category

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_GetCategoryText(laLineGraphWidget* graph, uint32_t
categoryID, laString * str);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t categoryID	category ID
laString * str	destination of the copied string

Function

[laResult laLineGraphWidget_GetCategoryText\(laLineGraphWidget* graph, uint32_t categoryID, laString * str\)](#)

laLineGraphWidget_GetFillGraphArea Function

Returns GFX_TRUE if the graph area is filled

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laBool laLineGraphWidget_GetFillGraphArea(laLineGraphWidget* graph);
```

Returns

laBool - GFX_TRUE if the graph area is filled

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

`laBool laLineGraphWidget_GetFillGraphArea(laLineGraphWidget* graph)`

laLineGraphWidget_GetFillSeriesArea Function

Returns GFX_TRUE if the series area are filled

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetFillSeriesArea(laLineGraphWidget* graph);
```

Returns

`laBool` - GFX_TRUE if the series area is filled

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

`laBool laLineGraphWidget_GetFillSeriesArea(laLineGraphWidget* graph)`

laLineGraphWidget_GetGridlinesVisible Function

Returns GFX_TRUE if the axis gridlines are visible

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetGridlinesVisible(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

`laBool` - GFX_TRUE if the axis gridlines are visible

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index

Function

`laBool laLineGraphWidget_GetGridlinesVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis)`

laLineGraphWidget_GetMaxValue Function

Returns the max value of the axis

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT uint32_t laLineGraphWidget_Get.MaxValue(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

uint32_t - max value

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

uint32_t laLineGraphWidget_Get.MaxValue(laLineGraphWidget* graph, laLineGraphValueAxis axis)

laLineGraphWidget_Get.MinValue Function

Returns the min value of the axis

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT uint32_t laLineGraphWidget_Get.MinValue(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

uint32_t - min value

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

uint32_t laLineGraphWidget_Get.MinValue(laLineGraphWidget* graph, laLineGraphValueAxis axis)

laLineGraphWidget_Get.SeriesFillPoints Function

Returns GFX_TRUE if the series points are filled

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laBool laLineGraphWidget_Get.SeriesFillPoints(laLineGraphWidget* graph, uint32_t
seriesID);
```

Returns

laBool - GFX_TRUE if the series points are filled

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

`laBool laLineGraphWidget_GetSeriesFillPoints(laLineGraphWidget* graph, uint32_t seriesID)`

laLineGraphWidget_GetSeriesLinesVisible Function

Returns GFX_TRUE if the series lines are visible

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetSeriesLinesVisible(laLineGraphWidget* graph, uint32_t
seriesID);
```

Returns

`laBool` - GFX_TRUE if the series lines are visible

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

`laBool laLineGraphWidget_GetSeriesLinesVisible(laLineGraphWidget* graph, uint32_t seriesID)`

laLineGraphWidget_GetSeriesPointSize Function

Returns the size of the drawn point

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT uint32_t laLineGraphWidget_GetSeriesPointSize(laLineGraphWidget* graph, uint32_t
seriesID);
```

Returns

`uint32_t` - the point size

Description

For circular points, this value is the radius For square points, the length of each side is twice the value

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t seriesID	the series ID

Function

`uint32_t laLineGraphWidget_GetSeriesPointSize(laLineGraphWidget* graph, uint32_t seriesID)`

laLineGraphWidget_GetSeriesPointType Function

Returns the type of point drawn for the series data points

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laLineGraphDataPointType laLineGraphWidget_GetSeriesPointType(laLineGraphWidget* graph, uint32_t seriesID);
```

Returns

laLineGraphDataPointType - the point type

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t seriesID	the series ID,

Function

laLineGraphDataPointType laLineGraphWidget_GetSeriesPointType(laLineGraphWidget* graph, uint32_t seriesID)

laLineGraphWidget_GetSeriesScheme Function

Returns scheme of the specified series

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laScheme * laLineGraphWidget_GetSeriesScheme(laLineGraphWidget* graph, uint32_t seriesID);
```

Returns

laScheme * - scheme of the specified series

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

laScheme * laLineGraphWidget_GetSeriesScheme(laLineGraphWidget* graph, uint32_t seriesID)

laLineGraphWidget_GetStacked Function

Returns GFX_TRUE if the bars are stacked

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetStacked(laLineGraphWidget* graph);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[laBool laLineGraphWidget_GetStacked\(laLineGraphWidget* graph\)](#)

laLineGraphWidget_GetTickLength Function

Returns the length of the ticks

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT uint32_t laLineGraphWidget_GetTickLength(laLineGraphWidget* graph);
```

Returns

uint32_t - tick length

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[uint32_t laLineGraphWidget_GetTickLength\(laLineGraphWidget* graph\)](#)

laLineGraphWidget_GetValueAxisLabelsVisible Function

Returns GFX_TRUE if the value axis labels are visible

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetValueAxisLabelsVisible(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

[laBool](#) - GFX_TRUE if the value axis labels are visible

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

[laBool laLineGraphWidget_GetValueAxisLabelsVisible\(laLineGraphWidget* graph, laLineGraphValueAxis axis\)](#)

laLineGraphWidget_GetValueAxisSubtickInterval Function

Returns the interval between minor ticks in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT uint32_t laLineGraphWidget_GetValueAxisSubtickInterval(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

uint32_t - ticks in pixels

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index

Function

uint32_t laLineGraphWidget_GetValueAxisSubticksPosition(laLineGraphWidget* graph, laLineGraphValueAxis axis)

laLineGraphWidget_GetValueAxisSubticksPosition Function

Returns the position of the subticks in the axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laLineGraphTickPosition
laLineGraphWidget_GetValueAxisSubticksPosition(laLineGraphWidget* graph, laLineGraphValueAxis
axis);
```

Returns

laLineGraphTickPosition - the subtick position

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the index of the value axis

Function

laLineGraphTickPosition laLineGraphWidget_GetValueAxisSubticksPosition(laLineGraphWidget* graph, laLineGraphValueAxis axis)

laLineGraphWidget_GetValueAxisSubticksVisible Function

Returns GFX_TRUE if the value axis subticks are visible

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetValueAxisSubticksVisible(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

`laBool` - GFX_TRUE if the value axis subticks are visible

Parameters

Parameters	Description
<code>laLineGraphWidget* graph</code>	the widget

Function

```
laBool laLineGraphWidget_GetValueAxisSubticksVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis)
```

laLineGraphWidget_GetValueAxisTickInterval Function

Returns the interval between major ticks in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT uint32_t laLineGraphWidget_GetValueAxisTickInterval(laLineGraphWidget* graph,
laLineGraphValueAxis axis);
```

Returns

`uint32_t` - ticks in pixels

Parameters

Parameters	Description
<code>laLineGraphWidget* graph</code>	the widget
<code>laLineGraphValueAxis axis</code>	the value axis index

Function

```
uint32_t laLineGraphWidget_GetValueAxisTickInterval(laLineGraphWidget* graph, laLineGraphValueAxis axis)
```

laLineGraphWidget_GetValueAxisTicksPosition Function

Returns the position of the ticks in the axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laLineGraphTickPosition
laLineGraphWidget_GetValueAxisTicksPosition(laLineGraphWidget* graph, laLineGraphValueAxis
axis);
```

Returns

`laLineGraphTickPosition` - the tick position

Parameters

Parameters	Description
<code>laLineGraphWidget* graph</code>	the widget

laLineGraphValueAxis axis	the index of the value axis
---------------------------	-----------------------------

Function

```
laLineGraphTickPosition laLineGraphWidget_GetValueAxisTicksPosition(laLineGraphWidget* graph, laLineGraphValueAxis axis)
```

laLineGraphWidget_GetValueAxisTicksVisible Function

Returns GFX_TRUE if the value axis ticks are visible

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laBool laLineGraphWidget_GetValueAxisTicksVisible(laLineGraphWidget* graph,  
laLineGraphValueAxis axis);
```

Returns

laBool - GFX_TRUE if the value axis ticks are visible

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget

Function

```
laBool laLineGraphWidget_GetValueAxisTicksVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis)
```

laLineGraphWidget_New Function

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

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laLineGraphWidget* laLineGraphWidget_New();
```

Returns

laLineGraphWidget*

Function

```
laLineGraphWidget* laLineGraphWidget_New()
```

laLineGraphWidget_SetCategoryAxisLabelsVisible Function

Shows/Hides the category axis labels

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetCategoryAxisLabelsVisible(laLineGraphWidget* graph,  
laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laBool visible	if GFX_TRUE, the axis labels are shown

Function

LIB_EXPORT [laResult](#) laLineGraphWidget_SetCategoryAxisLabelsVisible(laLineGraphWidget* graph, [laBool](#) visible)

laLineGraphWidget_SetCategoryAxisTicksPosition Function

Sets the position of the ticks in the category axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetCategoryAxisTicksPosition(laLineGraphWidget* graph,
laLineGraphTickPosition position);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphTickPosition position	position of the ticks

Function

[laResult](#) laLineGraphWidget_SetCategoryAxisTicksPosition(laLineGraphWidget* graph, laLineGraphTickPosition position)

laLineGraphWidget_SetCategoryAxisTicksVisible Function

Shows/Hides the category axis ticks

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetCategoryAxisTicksVisible(laLineGraphWidget* graph,
laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laBool visible	if GFX_TRUE, the axis ticks are shown

Function

[laResult](#) laLineGraphWidget_SetCategoryAxisTicksVisible(laLineGraphWidget* graph, [laBool](#) visible)

laLineGraphWidget_SetCategoryText Function

Sets the string used to label the category

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetCategoryText(laLineGraphWidget* graph, int32_t
categoryID, laString str);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
int32_t categoryID	category ID, if -1 the last category is assigned
laString str	the string to use

Function

```
laResult laLineGraphWidget_SetCategoryText(laLineGraphWidget* graph, uint32_t categoryID, laString str)
```

laLineGraphWidget_SetDataInSeries Function

Sets the value of the entry in the series index. The entry should have been previously

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetDataInSeries(laLineGraphWidget* graph, uint32_t
seriesID, uint32_t index, int32_t value);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t seriesID	the series ID
uint32_t index	the index of the data
int32_t value	the value

Function

```
laResult laLineGraphWidget_SetDataInSeries(laLineGraphWidget* graph,
uint32_t seriesID,
uint32_t index,
int32_t value);
```

laLineGraphWidget_SetFillGraphArea Function

Sets the graph area filled or not

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laResult laLineGraphWidget_SetFillGraphArea(laLineGraphWidget* graph, laBool fill);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laBool fill	if GFX_TRUE, fills the graph area

Function

[laResult laLineGraphWidget_SetFillGraphArea\(laLineGraphWidget* graph, laBool fill\)](#)

laLineGraphWidget_SetFillSeriesArea Function

Sets the series area filled or not

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laResult laLineGraphWidget_SetFillSeriesArea(laLineGraphWidget* graph, laBool fill);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laBool fill	if GFX_TRUE, fills the series area

Function

[laResult laLineGraphWidget_SetFillGraphArea\(laLineGraphWidget* graph, laBool fill\)](#)

laLineGraphWidget_SetGridlinesVisible Function

Shows/Hides the gridlines

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laResult laLineGraphWidget_SetGridlinesVisible(laLineGraphWidget* graph,
laLineGraphValueAxis axis, laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	category ID
laBool visible	shows the gridlines if GFX_TRUE

Function

[laResult](#) laLineGraphWidget_SetGridlinesVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis, [laBool](#) visible)

laLineGraphWidget_SetMaxValue Function

Sets the max value of the axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetMaxValue(laLineGraphWidget* graph,
laLineGraphValueAxis axis, int32_t value);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
int32_t value	max value

Function

[laResult](#) laLineGraphWidget_SetMaxValue(laLineGraphWidget* graph, laLineGraphValueAxis axis, int32_t value)

laLineGraphWidget_SetMinValue Function

Sets the min value of the axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetMinValue(laLineGraphWidget* graph,
laLineGraphValueAxis axis, int32_t value);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index

int32_t value	min value
---------------	-----------

Function

```
laResult laLineGraphWidget_SetMinValue(laLineGraphWidget* graph, laLineGraphValueAxis axis, int32_t value)
```

laLineGraphWidget_SetSeriesFillPoints Function

Sets the series points filled

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetSeriesFillPoints(laLineGraphWidget* graph, int32_t
seriesID, laBool fill);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
int32_t seriesID	the series ID, if negative the last series is referenced
laBool fill	fills the points if GFX_TRUE

Function

```
laResult laLineGraphWidget_SetSeriesFillPoints(laLineGraphWidget* graph, int32_t seriesID, laBool fill)
```

laLineGraphWidget_SetSeriesLinesVisible Function

Shows/hides the lines between series points

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetSeriesLinesVisible(laLineGraphWidget* graph, int32_t
seriesID, laBool visible);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
int32_t seriesID	the series ID, if negative the last series is referenced
laBool fill	Shows the lines between series data points if GFX_TRUE

Function

```
laResult laLineGraphWidget_SetSeriesLinesVisible(laLineGraphWidget* graph, int32_t seriesID, laBool visible)
```

laLineGraphWidget_SetSeriesPointSize Function

Sets the size of the point drawn for the series data

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetSeriesPointSize(laLineGraphWidget* graph, int32_t
seriesID, uint32_t size);
```

Returns

[laResult](#) - the result of the operation

Description

For circular points, this value sets the radius For square points, the length of each side is twice the value

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
int32_t seriesID	the series ID, if negative the last series is referenced
uint32_t size	size in pixels

Function

[laResult](#) [laLineGraphWidget_SetSeriesPointSize](#)(laLineGraphWidget* graph, int32_t seriesID, uint32_t size)

laLineGraphWidget_SetSeriesPointType Function

Sets the type of point drawn for the series data points

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetSeriesPointType(laLineGraphWidget* graph, int32_t
seriesID, laLineGraphDataPointType type);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
int32_t seriesID	the series ID, if negative the last series is referenced
laLineGraphDataPointType type	point type

Function

[laResult](#) [laLineGraphWidget_SetSeriesPointType](#)(laLineGraphWidget* graph, int32_t seriesID, laLineGraphDataPointType type)

laLineGraphWidget_SetSeriesScheme Function

Sets the color scheme of the series

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_SetSeriesScheme(laLineGraphWidget* graph, int32_t
seriesID, laScheme * scheme);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
int32_t seriesID	the series ID, if negative the last series is referenced
laScheme * scheme	the color scheme

Function

```
laResult laLineGraphWidget_SetSeriesScheme(laLineGraphWidget* graph, uint32_t seriesID, laScheme * scheme)
```

laLineGraphWidget_SetStacked Function

Stacks the line graph

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_SetStacked(laLineGraphWidget* graph, laBool stacked);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laBool stacked	if GFX_TRUE, the bars are stacked

Function

```
laResult laLineGraphWidget_SetStacked(laLineGraphWidget* graph, laBool stacked)
```

laLineGraphWidget_SetStringTable Function

Sets the string table used for the generated axis labels

File[libaria_widget_line_graph.h](#)**C**

```
LIB_EXPORT laResult laLineGraphWidget_SetStringTable(laLineGraphWidget* graph,
GFXU_StringTableAsset * stringTable);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
GFXU_StringTableAsset * stringTable	the string table

Function

`laResult laLineGraphWidget_SetStringTable(laLineGraphWidget* graph, GFXU_StringTableAsset * stringTable)`

laLineGraphWidget_SetTickLength Function

Sets the length of the ticks

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetTickLength(laLineGraphWidget* graph, uint32_t length);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t length	length in pixels

Function

`laResult laLineGraphWidget_SetTickLength(laLineGraphWidget* graph, uint32_t length)`

laLineGraphWidget_SetTicksLabelsStringID Function

Sets the ID of the superset string used for the value axis tick labels

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetTicksLabelsStringID(laLineGraphWidget* graph, uint32_t stringID);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
uint32_t stringID	the string ID

Function

`laResult laLineGraphWidget_SetTicksLabelsStringID(laLineGraphWidget* graph, uint32_t stringID)`

laLineGraphWidget_SetValueAxisLabelsVisible Function

Shows/Hides the labels in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laResult laLineGraphWidget_SetValueAxisLabelsVisible(laLineGraphWidget* graph,
laLineGraphValueAxis axis, laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
laBool visible	shows the labels if GFX_TRUE

Function

```
laResult laLineGraphWidget_SetValueAxisLabelsVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis, laBool visible)
```

laLineGraphWidget_SetValueAxisSubtickInterval Function

Sets the minor tick interval in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laResult laLineGraphWidget_SetValueAxisSubtickInterval(laLineGraphWidget* graph,
laLineGraphValueAxis axis, uint32_t interval);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
uint32_t interval	tick interval in pixels

Function

```
laResult laLineGraphWidget_SetValueAxisSubtickInterval(laLineGraphWidget* graph, laLineGraphValueAxis axis, uint32_t
interval)
```

laLineGraphWidget_SetValueAxisSubticksPosition Function

Sets the position of the subticks in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetValueAxisSubticksPosition(laLineGraphWidget* graph,
laLineGraphValueAxis axis, laLineGraphTickPosition position);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
laLineGraphTickPosition position	position of the subticks

Function

```
laResult laLineGraphWidget_SetValueAxisSubticksPosition(laLineGraphWidget* graph, laLineGraphValueAxis axis,
laLineGraphTickPosition position)
```

laLineGraphWidget_SetValueAxisSubticksVisible Function

Shows/Hides the subticks in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetValueAxisSubticksVisible(laLineGraphWidget* graph,
laLineGraphValueAxis axis, laBool visible);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
laBool visible	shows the subticks if GFX_TRUE

Function

```
laResult laLineGraphWidget_SetValueAxisSubticksVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis, laBool visible)
```

laLineGraphWidget_SetValueAxisTickInterval Function

Sets the tick interval in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetValueAxisTickInterval(laLineGraphWidget* graph,
laLineGraphValueAxis axis, uint32_t interval);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
uint32_t interval	tick interval in pixels

Function

`laResult laLineGraphWidget_SetValueAxisTickInterval(laLineGraphWidget* graph, laLineGraphValueAxis axis, uint32_t interval)`

laLineGraphWidget_SetValueAxisTicksPosition Function

Sets the position of the ticks in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetValueAxisTicksPosition(laLineGraphWidget* graph,
    laLineGraphValueAxis axis, laLineGraphTickPosition position);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
laLineGraphTickPosition position	the tick position

Function

`laResult laLineGraphWidget_SetValueAxisTicksPosition(laLineGraphWidget* graph, laLineGraphValueAxis axis,
 laLineGraphTickPosition position)`

laLineGraphWidget_SetValueAxisTicksVisible Function

Shows/Hides the ticks in the value axis

File

[libaria_widget_line_graph.h](#)

C

```
LIB_EXPORT laResult laLineGraphWidget_SetValueAxisTicksVisible(laLineGraphWidget* graph,
    laLineGraphValueAxis axis, laBool visible);
```

Returns

`laResult` - the result of the operation

Parameters

Parameters	Description
laLineGraphWidget* graph	the widget
laLineGraphValueAxis axis	the value axis index
laBool visible	shows the ticks if GFX_TRUE

Function

`laResult laLineGraphWidget_SetValueAxisTicksVisible(laLineGraphWidget* graph, laLineGraphValueAxis axis, laBool visible)`

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
<code>laLineWidget* line</code>	the widget
<code>int32_t* x</code>	pointer to an int to store the x coordinate
<code>int32_t* y</code>	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
<code>laLineWidget* line</code>	the widget
<code>int32_t* x</code>	pointer to an int to store the x coordinate
<code>int32_t* y</code>	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 coordinate 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 coordinate 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)`

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
<code>laList* list</code>	pointer to the list to modify
<code>size_t idx</code>	the index to modify
<code>void* val</code>	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
<code>laList* list</code>	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
<code>laList*</code> <code>list</code>	pointer to the list to modify
<code>void*</code> <code>val</code>	the value to insert
<code>uint32_t</code> <code>idx</code>	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
<code>laList*</code> <code>list</code>	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
<code>laList*</code> <code>list</code>	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)`

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

[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_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
<code>laListWheelWidget*</code> whl	the widget

Function

`laRelativePosition laListWheelWidget_GetIconPosition(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
<code>laListWheelWidget*</code> whl	the widget

Function

`uint32_t laListWheelWidget_GetIndicatorArea(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
<code>laListWheelWidget* whl</code>	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
<code>laListWheelWidget* whl</code>	the widget
<code>uint32_t idx</code>	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
<code>laListWheelWidget* whl</code>	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_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_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 occurred

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_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
<code>laListWidget* lst</code>	the widget

Function

`uint32_t laListWheelWidget_GetVisibleItemCount(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
<code>laListWheelWidget* whl</code>	the widget
<code>uint32_t idx</code>	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 management 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_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_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_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_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* img	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_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 momentum 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 if it 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_SetSelectedItem Function

Attempts to set the selected item 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](#)

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)
```

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 if 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
<code>laListWidget* lst</code>	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
<code>laListWidget* lst</code>	the widget

Function

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

laListWidget_GetItemEnable Function

Returns the enable state of the item in the list widget

File

[libaria_widget_list.h](#)

C

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

Returns

`laBool` - the enable state of the item

Parameters

Parameters	Description
<code>laListWidget* lst</code>	the widget
<code>uint32_t index</code>	the index of the item in the list

Function

```
laBool laListWidget_GetItemEnable(laListWidget* lst,
uint32_t idx)
```

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 management 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
<code>laListWidget* lst</code>	the widget
<code>laHAlignment align</code>	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
<code>laListWidget* lst</code>	the widget
<code>laBool allow</code>	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
<code>laListWidget* lst</code>	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_SetItemEnable Function

Enables or disables an item in the list. A disable item becomes un-selectable.

File

[libaria_widget_list.h](#)

C

```
LIB_EXPORT laResult laListWidget_SetItemEnable(laListWidget* lst, uint32_t idx, laBool enable);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laListWidget* lst	the widget
uint32_t index	the index of the item in the list
laBool enable	enable/disable the item

Function

```
laResult laListWidget_SetItemEnable(laListWidget* lst,
                                    uint32_t idx,
                                    laBool newEnableState)
```

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 whenever an item's 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)

laPieChartWidget_AddEntry Function

Adds an entry to the pie chart

File

[libaria_widget_pie_chart.h](#)

C

LIB_EXPORT laResult [laPieChartWidget_AddEntry](#)(laPieChartWidget* chart, int32_t * index);

Returns

[laResult](#) - the operation result

Description

The entry is always added to the end of the set. The index of the new entry is returned thru the index parameter.

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
int32_t * index	returns the index of the entry

Function

[laResult](#) [laPieChartWidget_AddEntry](#)(laPieChartWidget* chart, int32_t * index)

laPieChartWidget_DeleteEntries Function

Deletes ALL the data in the pie chart

File

[libaria_widget_pie_chart.h](#)

C

LIB_EXPORT laResult [laPieChartWidget_DeleteEntries](#)(laPieChartWidget* chart);

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget

Function

`laResult laPieChartWidget_DeleteEntries(laPieChartWidget* chart)`

laPieChartWidget_GetCenterAngle Function

Sets the center angle of the chart widget

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT int32_t laPieChartWidget_GetCenterAngle(laPieChartWidget* chart);
```

Returns

`int32_t`

Parameters

Parameters	Description
<code>laPieChartWidget* chart</code>	the widget

Function

`int32_t laPieChartWidget_GetCenterAngle(laPieChartWidget* chart)`

laPieChartWidget_GetEntryOffset Function

Returns the offset of the entry at the specified index

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT uint32_t laPieChartWidget_GetEntryOffset(laPieChartWidget* chart, uint32_t index);
```

Returns

`uint32_t`

Parameters

Parameters	Description
<code>laPieChartWidget* chart</code>	the widget
<code>uint32_t index</code>	the entry index

Function

`uint32_t laPieChartWidget_GetEntryOffset(laPieChartWidget* chart, uint32_t index)`

laPieChartWidget_GetEntryRadius Function

Returns the radius of the entry at the specified index

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT uint32_t laPieChartWidget_GetEntryRadius(laPieChartWidget* chart, uint32_t index);
```

Returns

uint32_t

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t index	the entry index

Function

uint32_t laPieChartWidget_GetEntryRadius(laPieChartWidget* chart, uint32_t index)

laPieChartWidget_GetEntryScheme Function

Returns the scheme of the entry at the specified index

File[libaria_widget_pie_chart.h](#)**C**

LIB_EXPORT laScheme * laPieChartWidget_GetEntryScheme(laPieChartWidget* chart, uint32_t index);

Returns

laScheme *

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t index	the entry index

Function

laScheme * laPieChartWidget_GetEntryScheme(laPieChartWidget* chart, uint32_t index)

laPieChartWidget_GetEntryValue Function

Returns the value of the entry at the specified index

File[libaria_widget_pie_chart.h](#)**C**

LIB_EXPORT uint32_t laPieChartWidget_GetEntryValue(laPieChartWidget* chart, uint32_t index);

Returns

uint32_t

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t index	the entry index

Function

uint32_t laPieChartWidget_GetEntryValue(laPieChartWidget* chart, uint32_t index)

laPieChartWidget_GetLabelsOffset Function

Gets the offsets of the labels from the center

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT uint32_t laPieChartWidget_GetLabelsOffset(laPieChartWidget* chart);
```

Returns

uint32_t - the offset

Parameters

Parameters	Description
laPieChartWidget* chart	the widget

Function

```
uint32_t laPieChartWidget_GetLabelsOffset(laPieChartWidget* chart)
```

laPieChartWidget_GetLabelsVisible Function

Returns GFX_TRUE if the labels are shown, GFX_FALSE if hidden

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laBool laPieChartWidget_GetLabelsVisible(laPieChartWidget* chart);
```

Returns

laBool

Parameters

Parameters	Description
laPieChartWidget* chart	the widget

Function

```
laBool laPieChartWidget_GetLabelsVisible(laPieChartWidget* chart)
```

laPieChartWidget_GetOrigin Function

Gets the origin coordinates of a chart widget

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_GetOrigin(laPieChartWidget* chart, int32_t* x, int32_t* y);
```

Returns

laResult - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	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 laPieChartWidget_GetOrigin\(laPieChartWidget* chart, int32_t* x, int32_t* y\)](#)

laPieChartWidget_GetStartAngle Function

Returns the start angle of a chart widget

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT int32_t laPieChartWidget_GetStartAngle(laPieChartWidget* chart);
```

Returns

uint32_t

Parameters

Parameters	Description
laPieChartWidget* chart	the widget

Function

[int32_t laPieChartWidget_GetStartAngle\(laPieChartWidget* chart\)](#)

laPieChartWidget_New Function

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

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laPieChartWidget* laPieChartWidget_New();
```

Returns

laPieChartWidget*

Function

[laPieChartWidget* laPieChartWidget_New\(\)](#)

laPieChartWidget_SetCenterAngle Function

Sets the center angle of the chart widget

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetCenterAngle(laPieChartWidget* chart, int32_t angle);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
int32_t angle	the desired center angle value

Function

[laResult](#) [laPieChartWidget_SetCenterAngle](#)(laPieChartWidget* chart, int32_t angle)

laPieChartWidget_SetEntryOffset Function

Sets the offset of an entry at index

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetEntryOffset(laPieChartWidget* chart, uint32_t index,
uint32_t offset);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t offset	entry offset
uint32_t rad	entry radius

Function

[laResult](#) [laPieChartWidget_SetEntryOffset](#)(laPieChartWidget* chart, uint32_t index, uint32_t offset)

laPieChartWidget_SetEntryRadius Function

Sets the radius of an entry at index

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetEntryRadius(laPieChartWidget* chart, uint32_t index,
uint32_t rad);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget

uint32_t index	entry index
uint32_t rad	entry radius

Function

[laResult laPieChartWidget_SetEntryRadius\(laPieChartWidget* chart, uint32_t index, uint32_t rad\)](#)

laPieChartWidget_SetEntryScheme Function

Sets the scheme of an entry at index

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetEntryScheme(laPieChartWidget* chart, uint32_t index,
laScheme * scheme);
```

Returns

[laResult - the operation result](#)

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t scheme	entry scheme
uint32_t rad	entry radius

Function

[laResult laPieChartWidget_SetEntryScheme\(laPieChartWidget* chart, uint32_t index, uint32_t scheme\)](#)

laPieChartWidget_SetEntryValue Function

Sets the value of an entry at index

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetEntryValue(laPieChartWidget* chart, uint32_t index,
uint32_t value);
```

Returns

[laResult - the operation result](#)

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t index	entry index
uint32_t value	entry value

Function

[laResult laPieChartWidget_SetEntryValue\(laPieChartWidget* chart, uint32_t index, uint32_t value\)](#)

laPieChartWidget_SetLabelsOffset Function

Sets the offsets of the labels from the center

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetLabelsOffset(laPieChartWidget* chart, uint32_t offset);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget uint32_t offset

Function

[laResult](#) `laPieChartWidget_SetLabelsOffset(laPieChartWidget* chart, uint32_t offset)`

laPieChartWidget_SetLabelsStringID Function

Sets the string asset for the labels

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetLabelsStringID(laPieChartWidget* chart, uint32_t stringID);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
uint32_t stringID	the ID of the string asset to use for labels

Function

`laPieChartWidget_SetLabelsStringID(laPieChartWidget* chart, uint32_t stringID)`

laPieChartWidget_SetLabelsVisible Function

Shows/hides the data entry labels

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetLabelsVisible(laPieChartWidget* chart, laBool visible);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget laBool visible

Function

[laResult](#) laPieChartWidget_SetLabelsVisible(laPieChartWidget* chart, [laBool](#) visible)

laPieChartWidget_SetOrigin Function

Sets the origin coordinates of a chart widget

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetOrigin(laPieChartWidget* chart, int32_t x, int32_t y);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
int32_t x	the desired x origin coordinate
int32_t y	the desired y origin coordinate

Function

[laResult](#) laPieChartWidget_SetOrigin(laPieChartWidget* chart, int32_t x, int32_t y)

laPieChartWidget_SetPressedEventCallback Function

Sets the function called when the chart is pressed/touched

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetPressedEventCallback(laPieChartWidget* chart,
laPieChartWidget_PressedEvent cb);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
laPieChartWidget_PressedEvent cb	callback function

Function

aResult laPieChartWidget_SetPressedEventCallback(laPieChartWidget* chart, laPieChartWidget_PressedEvent cb)

laPieChartWidget_SetStartAngle Function

Sets the start angle of a chart widget

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetStartAngle(laPieChartWidget* chart, int32_t angle);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
int32_t angle	the desired start angle value

Function

[laResult laPieChartWidget_SetStartAngle\(laPieChartWidget* chart, int32_t angle\)](#)

laPieChartWidget_SetStringTable Function

Sets the string table for the labels

File

[libaria_widget_pie_chart.h](#)

C

```
LIB_EXPORT laResult laPieChartWidget_SetStringTable(laPieChartWidget * chart,
GFXU_StringTableAsset * stringTable);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laPieChartWidget* chart	the widget
GFXU_StringTableAsset * stringTable	the string table to use

Function

[laResult laPieChartWidget_SetStringTable\(laPieChartWidget* chart, GFXU_StringTableAsset * stringTable\)](#)

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_ValueChangedEventCallback Function](#)

Gets the currently set value changed event callback.

File

[libaria_widget_progressbar.h](#)

C

```
LIB_EXPORT laProgressBar_ValueChangedEventCallback  
laProgressBarWidget_ValueChangedEventCallback(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 management 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)`

laRadialMenuWidget_AddWidget Function

Add a widget to the radial menu

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_AddWidget(laRadialMenuWidget* mn, laWidget* widget);
```

Returns

`laResult` - the operation result

Description

Adds a widget for the radial menu to manage, increments the total item count

Parameters

Parameters	Description
laRadialMenuWidget* mn	the radial menu widget
laWidget* widget	the item widget

Function

`laResult laRadialMenuWidget_AddWidget(laRadialMenuWidget* mn, laWidget* widget)`

laRadialMenuWidget_ClearItems Function

Clears all items in the radial menu widget

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_ClearItems(laRadialMenuWidget* mn);
```

Returns

`laResult` - the operation result

Description

Removes all items

Parameters

Parameters	Description
<code>laRadialMenuWidget* mn</code>	the widget

Function

`laResult laRadialMenuWidget_ClearItems(laRadialMenuWidget* mn)`

laRadialMenuWidget_GetItemProminenceChangedEventCallback Function

Gets the current radial menu item prominence change event callback

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laRadialMenuWidget_ItemProminenceChangedEvent
laRadialMenuWidget_GetItemProminenceChangedEventCallback(laRadialMenuWidget* mn);
```

Returns

`laRadialMenuWidget_ItemProminenceChangedEvent` - a valid callback pointer or NULL

Parameters

Parameters	Description
<code>laRadialMenuWidget* mn</code>	the widget

Function

`laRadialMenuWidget_ItemProminenceChangedEvent`
`laRadialMenuWidget_GetItemProminenceChangedEventCallback(laRadialMenuWidget* mn)`

laRadialMenuWidget_GetItemSelectedEventCallback Function

Gets the current radial menu item selected event callback

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laRadialMenuItemSelectedEvent
laRadialMenuItemSelectedEventCallback(laRadialMenuItemSelectedEvent* mn);
```

Returns

laRadialMenuItemSelectedEvent - a valid callback pointer or NULL

Parameters

Parameters	Description
laRadialMenuItemSelectedEvent* mn	the widget

Function

laRadialMenuItemSelectedEvent laRadialMenuItemSelectedEventCallback(laRadialMenuItemSelectedEvent* mn)

laRadialMenuItemSelectedEventCallback Function

Gets the index of the widget within the radial menu that is prominent

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT int32_t laRadialMenuItemSelectedEventCallback(laRadialMenuItemSelectedEvent* mn);
```

Returns

int32_t - the index of the widget, returns -1 if there is a failure

Description

Returns the index of the widget that is in the primary selectable position.

Parameters

Parameters	Description
laRadialMenuItemSelectedEvent* mn	the radial menu widget

Function

int32_t laRadialMenuItemSelectedEventCallback(laRadialMenuItemSelectedEvent* mn)

laRadialMenuItemSelectedEventCallback Function

Gets the theta value for the radial menu.

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT int32_t laRadialMenuItemSelectedEventCallback(laRadialMenuItemSelectedEvent* mn);
```

Returns

laResult - the operation result

Description

This function returns the current value of theta

Parameters

Parameters	Description
laRadialMenuItem* mn	the widget

Function

```
int32_t laRadialMenuItem_GetTheta(laRadialMenuItem* mn)
```

laRadialMenuItem_GetWidget Function

Gets the pointer for the widget by index

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laWidget* laRadialMenuItem_GetWidget(laRadialMenuItem* mn, int32_t index);
```

Returns

laWidget* - the pointer to the widget at the index value on the list Returns NULL if the index is out-of-range of the current list

Description

Returns the pointer to the widget

Parameters

Parameters	Description
laRadialMenuItem* mn	the radial menu widget
int32_t index	index value

Function

```
laWidget* laRadialMenuItem_GetWidget(laRadialMenuItem* mn, int32_t index)
```

laRadialMenuItem_IsProminent Function

Returns true if this radial menu item is currently in the primary selectable position

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laBool laRadialMenuItem_IsProminent(laRadialMenuItem* mn, laWidget* widget);
```

Returns

laBool - true if this widget is currently prominent

Parameters

Parameters	Description
laRadialMenuItem* mn	the radial menu widget
laWidget* mn	the item widget to inspect

Function

`laBool laRadialMenuWidget_IsProminent(laRadialMenuWidget* mn, laWidget* widget)`

laRadialMenuWidget_New Function

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

File

`libaria_widget_radial_menu.h`

C

```
LIB_EXPORT laRadialMenuWidget* laRadialMenuWidget_New();
```

Returns

`laRadialMenuWidget*`

Function

`laRadialMenuWidget* laRadialMenuWidget_New()`

laRadialMenuWidget_RemoveWidget Function

Removes a widget to the radial menu

File

`libaria_widget_radial_menu.h`

C

```
LIB_EXPORT laResult laRadialMenuWidget_RemoveWidget(laRadialMenuWidget* mn, laWidget* widget);
```

Returns

`laResult` - the operation result, returns `LA_FAIL` if a match is not found

Description

Removes a widget for the radial menu to manage, decrements the total item count

Parameters

Parameters	Description
<code>laRadialMenuWidget* mn</code>	the radial menu widget
<code>laWidget* widget</code>	the item widget

Function

`laResult laRadialMenuWidget_RemoveWidget(laRadialMenuWidget* mn, laWidget* widget)`

laRadialMenuWidget_SetAlphaScaleMinMax Function

Sets the minimum and maximum alpha scaling ratio

File

`libaria_widget_radial_menu.h`

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetAlphaScaleMinMax(laRadialMenuWidget* mn, int32_t min,
```

```
int32_t max);
```

Returns

[laResult](#) - the operation result

Description

Sets the minimum and maximum alpha scaling ratio for the items. These values are only used if [laRadialMenuWidget_SetAlphaScaling](#) is called and alpha scaling is enabled

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
int32_t min	the minimum alpha scale, it should be between 0 - 255
int32_t max	the maximum alpha scale, it should be between 0 - 255

Function

```
laResult laRadialMenuWidget_SetAlphaScaleMinMax(laRadialMenuWidget* mn,
int32_t min, int32_t max)
```

laRadialMenuWidget_SetAlphaScaling Function

Enables per item alpha scaling for the radial menu

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetAlphaScaling(laRadialMenuWidget* mn,
laRadialMenuWidgetScaleType setting);
```

Returns

[laResult](#) - the operation result

Description

Enable/Disable the ability for the radial menu to scale the items alpha as they travel into the background

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
laRadialMenuWidgetScaleType setting	setting flag

Function

```
laResult laRadialMenuWidget_SetAlphaScaling(laRadialMenuWidget* mn,
laRadialMenuWidgetScaleType setting)
```

laRadialMenuWidget_SetDrawEllipse Function

Enables drawing the elliptical track for the radial menu

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetDrawEllipse(laRadialMenuWidget* mn, laBool enable);
```

Returns

[laResult](#) - the operation result

Description

Enable/Disable the drawing of the elliptical track of travel of the items in the

- radial menu

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
laRadialMenuWidgetScaleType setting	setting flag

Function

[laResult laRadialMenuWidget_SetDrawEllipse\(laRadialMenuWidget* mn, laBool enable\)](#)

laRadialMenuWidget_SetEllipseType Function

Sets the ellipse type for the radial menu track.

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetEllipseType(laRadialMenuWidget* mn,  
laRadialMenuEllipseType type);
```

Returns

[laResult](#) - the operation result

Description

This function sets the ellipse type for the radial menu. There are three types: DEFAULT - the elliptical track is best-fit based on the width and height of the radial menu widget and all the item widgets with scaling factored in. The theta angle value is used for the ellipse but only in a limited way. ORBITAL - the elliptical track is flatter and factors in the theta angle value more. Some of the items may be clipped out of the widget area depending on the size of radial widget and the theta angle ROLLODEX - the elliptical track is vertical in-nature and visually simular to a rolodex

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
laRadialMenuEllipseType type	type of ellipse

Function

[laResult laRadialMenuWidget_SetEllipseType\(laRadialMenuWidget* mn,
laRadialMenuEllipseType type\)](#)

laRadialMenuWidget_SetHighlightProminent Function

Sets the item widget to highlight when it is at the prominent location

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetHighlightProminent(laRadialMenuWidget* mn, laBool
```

```
enable);
```

Returns

laResult - the operation result

Description

This tells the radial menu widget to enable/disable the ability for the prominent item widget to be highlighted when it comes to rest at the prominent location.

Parameters

Parameters	Description
laRadialMenuWidget* mn	the radial menu widget
laBool enable	the item widget to highlight when prominent

Function

laResult laRadialMenuWidget_SetHighlightProminent(laRadialMenuWidget* mn, laBool enable)

laRadialMenuWidget_SetItemProminenceChangedEventCallback Function

Sets the deselected callback pointer

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult
laRadialMenuWidget_SetItemProminenceChangedEventCallback(laRadialMenuWidget* mn,
laRadialMenuWidget_ItemProminenceChangedEvent cb);
```

Returns

laResult - the operation result

Description

This callback is called when this radial menu is deselected

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
laRadialMenuWidget_DeselectedEvent	a valid callback pointer or NULL

Function

laResult laRadialMenuWidget_SetItemProminenceChangedEventCallback(laRadialMenuWidget* mn, laRadialMenuWidget_ItemProminenceChangedEvent cb)

laRadialMenuWidget_SetItemSelectedEventCallback Function

Sets the radial menu item selected event callback

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetItemSelectedEventCallback(laRadialMenuWidget* mn,
laRadialMenuWidget_ItemSelectedEvent cb);
```

Returns

[laResult](#) - the operation result

Description

This callback is called when an item widget in the radial menu becomes selected

Parameters

Parameters	Description
laRadialMenuItemWidget* mn	the widget
laRadialMenuItemWidget_ItemSelectedEvent cb	a valid callback pointer or NULL

Function

[laResult](#) laRadialMenuItemWidget_SetItemSelectedEventCallback(laRadialMenuItemWidget* mn, laRadialMenuItemWidget_ItemSelectedEvent cb)

laRadialMenuItemWidget_SetNumberOfItemsShown Function

Sets the number of items visible on the menu

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuItemWidget_SetNumberOfItemsShown(laRadialMenuItemWidget* mn, int32_t num);
```

Returns

[laResult](#) - the operation result

Description

Sets the number of items visible on the menu, this number can be less than or equal to the total number of items

Parameters

Parameters	Description
laRadialMenuItemWidget* mn	the radial menu widget
int32_t num	the number of widgets shown

Function

[laResult](#) laRadialMenuItemWidget_SetNumberOfItemsShown(laRadialMenuItemWidget* mn, int32_t num)

laRadialMenuItemWidget_SetProminent Function

Sets this widget as prominent.

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuItemWidget_SetProminent(laRadialMenuItemWidget* mn, laWidget* widget);
```

Returns

[laResult](#) - the operation result

Description

If this widget belongs to a radial menu then this function will cycle the radial menu to show the widget in the primary selectable position.

Parameters

Parameters	Description
laRadialMenuWidget* mn	the radial menu widget
laWidget* widget	the item widget to show prominence

Function

[laResult laRadialMenuWidget_SetProminent\(laRadialMenuWidget* mn, laWidget* widget\)](#)

laRadialMenuWidget_SetProminentIndex Function

Sets a widget with index within the radial menu as prominent

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetProminentIndex(laRadialMenuWidget* mn, uint32_t index);
```

Returns

[laResult - the operation result](#)

Description

If the index supplied is within the range of widgets within te radial menu then this function will cycle the radial menu to show the widget in the primary selectable position.

Parameters

Parameters	Description
laRadialMenuWidget* mn	the radial menu widget
int32_t index	the item widget to show prominence

Function

[laResult laRadialMenuWidget_SetProminentIndex\(laRadialMenuWidget* mn, int32_t index\)](#)

laRadialMenuWidget_SetSizeScaleMinMax Function

Sets the minimum and maximum size scaling ratio in percent

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetSizeScaleMinMax(laRadialMenuWidget* mn, int32_t min, int32_t max);
```

Returns

[laResult - the operation result](#)

Description

Sets the minimum and maximum size scaling ratio for the items. These values are only used if

[laRadialMenuWidget_SetSizeScaling](#) is called and size scaling is enabled

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
int32_t min	the minimum size scale percentage, it should be between 1 - 200
int32_t max	the maximum size scale percentage, it should be between 1 - 200

Function

```
laResult laRadialMenuWidget_SetSizeScaleMinMax(laRadialMenuWidget* mn,
int32_t min, int32_t max)
```

laRadialMenuWidget_SetSizeScaling Function

Enables per item size scaling for the radial menu

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetSizeScaling(laRadialMenuWidget* mn,
laRadialMenuWidgetScaleType setting);
```

Returns

[laResult](#) - the operation result

Description

Enable/Disable the ability for the radial menu to scale the items sizes as they travel into the background

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
laRadialMenuWidgetScaleType setting	setting flag

Function

```
laResult laRadialMenuWidget_SetSizeScaling(laRadialMenuWidget* mn,
laRadialMenuWidgetScaleType setting)
```

laRadialMenuWidget_SetTheta Function

Sets the theta value for the radial menu.

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetTheta(laRadialMenuWidget* mn, int32_t theta);
```

Returns

[laResult](#) - the operation result

Description

This function sets the theta (angle of rotation of the ellipse)

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
int32_t theta	angle in degrees of rotation relative to the y-axis of the ellipse

Function

```
laResult laRadialMenuWidget_SetTheta(laRadialMenuWidget* mn,
int32_t theta)
```

laRadialMenuWidget_SetTouchArea Function

Sets the area that touch input is allowed within the radial menu widget

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetTouchArea(laRadialMenuWidget* mn, int32_t x, int32_t
y, int32_t width, int32_t height);
```

Returns

[laResult](#) - the operation result

Description

Sets the area that touch input is permitted. This area has to be at or smaller than the rectangular area of the entire radial menu widget. The default is the bottom half of the widget.

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
GFX_Rect rect	rectangular area, x-y represents offsets in local space

Function

```
laResult laRadialMenuWidget_SetTouchArea(laRadialMenuWidget* mn,
int32_t x, int32_t y, int32_t width, int32_t height)
```

laRadialMenuWidget_SetWidgetAt Function

Insert a widget to the radial menu at the index specified

File

[libaria_widget_radial_menu.h](#)

C

```
LIB_EXPORT laResult laRadialMenuWidget_SetWidgetAt(laRadialMenuWidget* mn, laWidget* widget,
int32_t index);
```

Returns

[laResult](#) - the operation result, returns LA_SUCCESS if the set was successful returns fail if the index value is out of the range of the widget list

Parameters

Parameters	Description
laRadialMenuWidget* mn	the widget
laWidget* widget	the item widget
int32_t index	index value

Function

```
laResult laRadialMenuWidget_SetWidgetAt(laRadialMenuWidget* mn,
                                         laWidget* widget, int32_t index)
```

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 the 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_GetCircleButtonSize Function

Gets the diameter/size of the default circle button

File

[libaria_widget_radiobutton.h](#)

C

```
LIB_EXPORT uint32_t laRadioButtonWidget_GetCircleButtonSize(laRadioButtonWidget* btn);
```

Returns

uint32_t size - the size of the default circle button

Parameters

Parameters	Description
laRadioButtonWidget * btn	the widget

Function

```
LIB_EXPORT uint32_t laRadioButtonWidget_GetCircleButtonSize( 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
<code>laRadioButtonWidget* btn</code>	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
<code>laRadioButtonWidget* btn</code>	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
<code>laRadioButtonWidget* btn</code>	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
<code>laRadioButtonWidget* btn</code>	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
<code>laRadioButtonWidget* btn</code>	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_SetVAlignment Function

Sets the vertical alignment for a button

File

[libaria_widget_radiobutton.h](#)

C

```
LIB_EXPORT laVAlignment laRadioButtonWidget_SetVAlignment(laRadioButtonWidget* btn);
```

Returns

[laVAlignment](#) align - the desired vertical alignment setting

Parameters

Parameters	Description
laRadioButtonWidget* btn	the widget

Function

```
laVAlignment laRadioButtonWidget_SetVAlignment(laRadioButtonWidget* btn)
```

laRadioButtonWidget_New Function

Allocates memory for a new widget of this type. The application is responsible for the management 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_SetCircleButtonSize Function

Sets the size of the default circle button

File

[libaria_widget_radiobutton.h](#)

C

```
LIB_EXPORT laResult laRadioButtonWidget_SetCircleButtonSize(laRadioButtonWidget* btn, uint32_t size);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laRadioButtonWidget* btn	the widget
uint32_t size	the diameter of the circle, in pixels

Function

[laResult laRadioButtonWidget_SetCircleButtonSize\(\[laRadioButtonWidget*\]\(#\) btn,
uint32_t size\)](#)

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_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)
```

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 cb	a valid callback pointer or NULL

Function

```
laResult laRadioButtonWidget_SetSelectedEventCallback(laRadioButtonWidget* btn,
GFXU_ImageAsset* img);
```

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

<code>laVAlignment align</code>	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
<code>laRectangleWidget* rect</code>	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 management 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()
```

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)
```

laRectArray_Clear Function

Removes all values from a given array. Array capacity remains the same.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_Clear(laRectArray* arr);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	the array to modify

Function

```
laResult laRectArray_Clear(laRectArray* arr)
```

laRectArray_Copy Function

Creates a duplicate of an existing array

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_Copy(laRectArray* src, laRectArray* dest);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* src	the source array
laRectArray* dest	the result array

Function

```
laResult laRectArray_Copy(laRectArray* l, laRectArray* r)
```

laRectArray_Create Function

Initializes a new rectangle array.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_Create(laRectArray* arr);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to initilaize

Function

[laResult](#) [laRectArray_Create](#)([laRectArray](#)* arr)

laRectArray_Destroy Function

Removes all nodes from a given array and frees the memory owned by the array. Resets array capacity to zero.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_Destroy(laRectArray* arr);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	the array to modify

Function

[laResult](#) [laRectArray_Destroy](#)([laRectArray](#)* arr)

laRectArray_InsertAt Function

Inserts a rectangle into an array at a given index. All existing nodes from index are shifted right one place.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_InsertAt(laRectArray* arr, uint32_t idx, const GFX_Rect* rect);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to modify
uint32_t idx	the position to insert the value
const GFX_Rect* rect	the rectangle value

Function

```
laResult laRectArray_InsertAt(laRectArray* arr,
    uint32_t idx,
    const GFX_Rect* rect);
```

laRectArray_MergeSimilar Function

Analyzes an array and merges any rectangles similar in size.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_MergeSimilar(laRectArray* arr);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	the array to analyze

Function

```
laResult laRectArray_MergeSimilar(laRectArray* arr)
```

laRectArray_PopBack Function

Removes the last rectangle from the array

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_PopBack(laRectArray* arr);
```

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to modify

Function

```
void laRectArray_PopBack( laRectArray* arr)
```

laRectArray_PopFront Function

Removes the first value from the array. Shuffles all other nodes forward one index.

File[libaria_rectarray.h](#)**C**

```
LIB_EXPORT laResult laRectArray_PopFront(laRectArray* arr);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to modify

Function

```
void laRectArray_PopFront( laRectArray* arr)
```

laRectArray_PushBack Function

Pushes a new rectangle onto the back of the array

File[libaria_rectarray.h](#)**C**

```
LIB_EXPORT laResult laRectArray_PushBack(laRectArray* arr, const GFX_Rect* rect);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to modify
const GFX_Rect* rect	the rectangle value

Function

```
laResult laRectArray_PushBack(laRectArray* arr, const GFX_Rect* rect)
```

laRectArray_PushFront Function

Pushes a new rectangle onto the front of the array. Shuffles all other nodes backward one index.

File[libaria_rectarray.h](#)**C**

```
LIB_EXPORT laResult laRectArray_PushFront(laRectArray* arr, const GFX_Rect* rect);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to modify

const GFX_Rect* rect	the rectangle value
----------------------	---------------------

Function

```
laResult laRectArray_PushFront(laRectArray* arr, const GFX_Rect* rect)
```

laRectArray_RemoveAt Function

Removes a rectangle from the array at an index

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_RemoveAt(laRectArray* arr, uint32_t idx);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to modify
uint32_t idx	the index of the value to remove

Function

```
laResult laRectArray_RemoveAt(laRectArray* arr, uint32_t idx)
```

laRectArray_RemoveDuplicates Function

Removes any duplicate rectangles from an array.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_RemoveDuplicates(laRectArray* arr);
```

Returns

laResult - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	the array to analyze

Function

```
laResult laRectArray_RemoveDuplicates(laRectArray* arr)
```

laRectArray_RemoveOverlapping Function

Sorts the array by size and then removes any rectangles that are completely overlapped by another larger rectangle.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_RemoveOverlapping(laRectArray* arr);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	the array to modify

Function

```
laResult laRectArray_RemoveOverlapping(laRectArray* arr)
```

laRectArray_Resize Function

Resizes the capacity of the array. If the array shrinks, any nodes beyond the new capacity will be discarded.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_Resize(laRectArray* arr, uint32_t sz);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	pointer to the array to resize
uint32_t sz	the desired capacity

Function

```
laResult laRectArray_Resize(laRectArray* arr)
```

laRectArray_SortBySize Function

Sorts a given array largest to smallest.

File

[libaria_rectarray.h](#)

C

```
LIB_EXPORT laResult laRectArray_SortBySize(laRectArray* arr);
```

Returns

[laResult](#) - the result of the operation

Parameters

Parameters	Description
laRectArray* arr	the array to analyze

Function

```
laResult laRectArray_SortBySize(laRectArray* arr)
```

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_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_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_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_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)`

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)`

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
<code>laScreen* scr</code>	the screen to modify
<code>laScreenOrientation</code>	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
<code>laScreen* scr</code>	the screen to modify
<code>laScreen_ShowHideCallback_FnPtr</code>	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 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_GetMaximumValue Function

Gets the maximum scroll value

File

`libaria_widget_scrollbar.h`

C

```
LIB_EXPORT uint32_t laScrollBarWidget_GetMaximumValue(laScrollBarWidget* bar);
```

Returns

`uint32_t` - the maximum scroll value

Parameters

Parameters	Description
laScrollBarWidget* bar	the widget

Function

`uint32_t laScrollBarWidget_GetMaximumValue(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 management 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)
```

laShutdown Function

Function to shutdown the active Aria library state.

File

[libaria.h](#)

C

```
LIB_EXPORT void laShutdown();
```

Description

Type: void laShutdown()

Function to shutdown the active Aria library state.

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_GetMaximumValue Function

Gets the maximum value for the slider

File

[libaria_widget_slider.h](#)

C

```
LIB_EXPORT uint32_t lasliderWidget_GetMaximumValue(laSliderWidget* sld);
```

Returns

uint32_t - the maximum value for the slider

Parameters

Parameters	Description
laSliderWidget* sld	the widget

Function

```
uint32_t laSliderWidget_GetMaximumValue( laSliderWidget* sld)
```

laSliderWidget_GetMinimumValue 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 management 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
int32_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)
```

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 width 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_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_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_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_GetAscent Function

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

File

[libaria_string.h](#)

C

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

Returns

uint32_t - the ascent of the string

Parameters

Parameters	Description
laString* str	the string to reference

Function

```
uint32_t laString_GetAscent( laString* str)
```

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_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, int32_t lineSpace);
```

Returns

void

Parameters

Parameters	Description
laString* str	the string to reference
GFX_Rect* rect	the calculated string rectangle result
int32_t lineSpace	the space between lines. if -1, will use font ascent

Function

```
void laString_GetMultiLineRect( laString* str, GFX_Rect* rect)
```

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_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)

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
<code>laString* src</code>	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_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)

[*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)
```

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_GetCursorPosition(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_GetFocusChangedEventCallback Function

Gets the current focus changed event callback pointer

File

[libaria_widget_textfield.h](#)

C

```
LIB_EXPORT laTextFieldWidget_FocusChangedCallback
laTextFieldWidget_GetFocusChangedEventCallback(laTextFieldWidget* txt);
```

Returns

`laTextFieldWidget_FocusChangedCallback` - a valid pointer or NULL

Parameters

Parameters	Description
laTextFieldWidget* txt	the widget

Function

`laTextFieldWidget_FocusChangedCallback laTextFieldWidget_GetFocusChangedEventCallback(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 management 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_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\(\[laTextFieldWidget* txt\]\(#\),
\[laBool clear\\)\]\(#\)](#)

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_SetFocusChangedEventCallback Function

Sets the focus changed event callback pointer

File

[libaria_widget_textfield.h](#)

C

```
LIB_EXPORT laResult laTextFieldWidget_SetFocusChangedEventCallback(laTextFieldWidget* txt,
                                                               laTextFieldWidget_FocusChangedCallback cb);
```

Returns

[laResult](#) - the operation result

Parameters

Parameters	Description
laTextFieldWidget* txt	the widget
laTextFieldWidget_FocusChangedCallback	a valid pointer or NULL

Function

```
laResult laTextFieldWidget_SetFocusChangedEventCallback(laTextFieldWidget* txt,
                                                       laTextFieldWidget_FocusChangedCallback cb)
```

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
<code>laTouchTestWidget* tch</code>	the widget
<code>GFX_Point* pnt</code>	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
<code>laTouchTestWidget* tch</code>	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
<code>laTouchTestWidget* tch</code>	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 management 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\(laTouchTestWidget* txt, laTouchTestWidget_PointAddedEventCallback cb\)](#)

laUpdate Function

Aria library update (tasks) function.

File

[libaria.h](#)

C

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

Description

Type: void laUpdate()

This function updates the active Aria library context state. It performs event processing as well as servicing of the widget paint loop. It should be called often.

Parameters

Parameters	Description
uint32_t dt	a delta time, typically expressed in milliseconds, since the last time laUpdate was called. If this value is zero then time-dependent features will not advance.

laUpdate_RTOS Function

RTOS version of the Aria library update (tasks) function.

File

[libaria_rtos.h](#)

C

```
LIB_EXPORT void laUpdate_RTOS(laBool fullBlock, uint32_t dt);
```

Description

Type: void laUpdate_RTOS()

This function updates the active Aria library context state. It performs event processing as well as servicing of the widget paint loop.

This function may block to wait for events to process. Setting fullBlock = LA_TRUE will fully block without timeout, otherwise it won't block.

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_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_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
<code>int32_t x</code>	the x coordinate of the pick point
<code>int32_t y</code>	the y coordinate of the pick point

Function

```
laWidget* laUtils_Pick(int32_t x, int32_t y)
```

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
<code>int32_t x</code>	the x coordinate of the pick point
<code>int32_t y</code>	the y coordinate of the pick point

Function

```
laWidget* laUtils_PickFromLayer(const laLayer* layer, 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
<code>laLayer* layer</code>	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_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_PointToLocalSpace( laWidget\* widget, GFX\_Point\* pnt)
```

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_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)
```

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_WidgetIsOccluded Function

Returns LA_TRUE if the specified widget is occluded in the specified rectangle area.

File

[libaria_utils.h](#)

C

```
laBool laUtils_WidgetIsOccluded(laWidget* wgt, const GFX_Rect* rect);
```

Returns

void

Parameters

Parameters	Description
laWidget* wgt	the widget to test
GFX_Rect * rect	the occlusion area

Function

[laBool](#) [laUtils_WidgetIsOccluded](#)([laWidget](#)* wgt, const [GFX_Rect](#)* rect)

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)

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
<code>laWidget* widget</code>	the subject widget

Function

```
GFX_Rect laUtils_WidgetLocalRect(laWidget* widget)
```

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
<code>laWidget* parent</code>	the parent widget
<code>laWidget* child</code>	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_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)
```

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_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_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_GetChildIndex 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_GetChildIndex(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_GetChildIndex\(laWidget* parent, uint32_t idx\)](#)

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_GetCornerRadius Function

Returns the corner radius of the widget

File

[libaria_widget.h](#)

C

```
LIB_EXPORT uint32_t laWidget_GetCornerRadius(laWidget* wgt);
```

Returns

uint32_t - the corner radius

Description

Returns the corner radius of the widget

Parameters

Parameters	Description
laWidget* wgt	the widget

Function

```
uint32_t laWidget_GetCornerRadius( laWidget* wgt)
```

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 implicitly be blended at 50%. If a child further explicitly 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
<code>lawidget* wgt</code>	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 interaction if it is false. Widgets must individually support this flag.

Parameters

Parameters	Description
<code>laWidget* wgt</code>	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

<code>laMargin* mg</code>	a pointer to an <code>laMargin</code> object to store the margin values
---------------------------	---

Function

`laResult laWidget_GetMargin (laWidget* wgt, laMargin* mg)`

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
<code>laWidget* wgt</code>	the widget

Function

`laBool laWidget_GetOptimizationFlags(laWidget* wgt)`

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
<code>laWidget* wgt</code>	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 if 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 capabilities.

Parameters

Parameters	Description
<code>laWidget* wgt</code>	the widget
<code>laWidget_TouchDownEvent_FnPtr ptr</code>	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 capabilities.

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 capabilities.

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
<code>laWidget*</code> parent	the parent widget
<code>laWidget*</code> 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
<code>laWidget*</code> wgt	the widget
<code>int32_t</code> width	the amount to change the width by
<code>int32_t</code> height	the amount to 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_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_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_SetCornerRadius Function

Sets the widget corner radius.

File[libaria_widget.h](#)**C**

```
LIB_EXPORT laResult laWidget_SetCornerRadius(laWidget* wgt, uint32_t radius);
```

Returns

[laResult](#) - the operation result

Description

Sets the widget corner radius. A widget with non-zero corner radius will have round edges. This only supports widgets with non-beveled borders.

Parameters

Parameters	Description
laWidget* wgt	the widget
uint32_t radius	the radius

Function

```
laResult laWidget_SetCornerRadius(laWidget* wgt,
                                 uint32_t radius)
```

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_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)
```

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
<code>laWidget* wgt</code>	the widget
<code>int32_t y</code>	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
<code>laWidget* wgt</code>	the widget
<code>int32_t x</code>	the amount to move in x
<code>int32_t y</code>	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
laWidget* 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 management 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* img	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\(laWindowWidget* win, laString str\)](#)

b) Data Types and Constants

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](#)

laArcWidget_t Structure

Implementation of a arc widget.

File

[libaria_widget_arc.h](#)

C

```
struct laArcWidget_t {
    laWidget widget;
    uint32_t radius;
    uint32_t startAngle;
    int32_t centerAngle;
    uint32_t endAngle;
    uint32_t thickness;
    laBool roundEdge;
};
```

Members

Members	Description
laWidget widget;	base widget header
uint32_t radius;	the radius of the arc
uint32_t startAngle;	the start angle of the arc
int32_t centerAngle;	the center angle of the arc
uint32_t endAngle;	the end angle of the arc
uint32_t thickness;	the thickness of the arc
laBool roundEdge;	draws round edge if true.

Description

Structure: laArcWidget_t

A arc widget draws a arc of the specified origin and radius inside the widget bounds. All coordinates are expressed in local widget space.

The color of the arc is determined by the widget scheme's 'foreground' color.

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.

laBarGraphCategory_t Structure

Contains the Category properties

File

[libaria_widget_bar_graph.h](#)

C

```
struct laBarGraphCategory_t {
    laString text;
};
```

Description

Structure: laBarGraphCategory_t

Remarks

None.

laBarGraphDataSeries_t Structure

The data series object that contains the series properties and data

File

[libaria_widget_bar_graph.h](#)

C

```
struct laBarGraphDataSeries_t {
    laScheme * scheme;
    laArray data;
    laBarGraphValueAxis axis;
};
```

Description

Structure: laBarGraphDataSeries_t

Remarks

None.

laBarGraphTickPosition_t Enumeration

The tick position relative to axis

File

[libaria_widget_bar_graph.h](#)

C

```
enum laBarGraphTickPosition_t {
    BAR_GRAPH_TICK_IN,
    BAR_GRAPH_TICK_OUT,
    BAR_GRAPH_TICK_CENTER
};
```

Description

Enumeration: laBarGraphTickPosition_t

laBarGraphValueAxis_t Enumeration

The value axis index value

File

[libaria_widget_bar_graph.h](#)

C

```
enum laBarGraphValueAxis_t {
    BAR_GRAPH_AXIS_0 = 0
};
```

Description

Enumeration: laBarGraphValueAxis_t

laBarGraphWidget_t Structure

Implementation of a bar graph widget.

File

[libaria_widget_bar_graph.h](#)

C

```
struct laBarGraphWidget_t {
    laWidget widget;
    uint32_t tickLength;
    laBool fillGraphArea;
    int32_t maxValue;
    int32_t minValue;
    uint32_t tickInterval;
    uint32_t subtickInterval;
    laBool valueAxisLabelsVisible;
    laBool valueAxisTicksVisible;
    laBool valueAxisSubticksVisible;
    laBool valueGridlinesVisible;
    laBool stacked;
    laArray dataSeries;
    GFXU_StringTableAsset * stringTable;
    uint32_t ticksLabelsStringID;
    laBarGraphTickPosition valueAxisTicksPosition;
    laBarGraphTickPosition valueAxisSubticksPosition;
    laBool categAxisLabelsVisible;
    laBool categAxisTicksVisible;
    laBarGraphTickPosition categAxisTicksPosition;
    laArray categories;
    GFXU_ExternalAssetReader* reader;
};
```

Members

Members	Description
laWidget widget;	base widget header
int32_t maxValue;	Value axis properties
GFXU_StringTableAsset * stringTable;	string table
uint32_t ticksLabelsStringID;	ID of Superset string containing numbers
laBool categAxisLabelsVisible;	Category axis properties
GFXU_ExternalAssetReader* reader;	asset reader

Description

Structure: laBarGraphWidget_t

A bar graph widget draws a bar graph. All coordinates are expressed in local widget space.

The color of the graph is determined by the widget scheme's 'foreground' color.

Remarks

None.

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.

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 and released

state.

File

[libaria_widget_keypad.h](#)

C

```
typedef struct laButtonWidget_t laButtonWidget;
```

Description

Structure: [laButtonWidget_t](#)

Remarks

None.

laButtonWidget_PressedEvent Type

Button pressed event function callback type

File

[libaria_widget_button.h](#)

C

```
typedef void (* laButtonWidget_PressedEvent)(laButtonWidget*);
```

Description

Function Pointer: [laButtonWidget_PressedEvent](#)

laButtonWidget_ReleasedEvent Type

Button released event function callback type

File

[libaria_widget_button.h](#)

C

```
typedef void (* laButtonWidget_ReleasedEvent)(laButtonWidget*);
```

Description

Function Pointer: [laButtonWidget_ReleasedEvent](#)

laButtonWidget_t Structure

Implementation of a button widget. A button is an interactive element that simulates a typical button with a pressed and released state.

File

[libaria_widget_button.h](#)

C

```
struct laButtonWidget_t {
    laWidget widget;
    laButtonState state;
    uint8_t toggleable;
    laString text;
    int32_t textLineSpace;
    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
int32_t textLineSpace;	the space between lines of text (pixels)
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.

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;	position 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_UncheckedEvent 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

Checkbox checked event function callback type

File

[libaria_widget_checkbox.h](#)

C

```
typedef void (* laCheckBoxWidget_CheckedEvent)(laCheckBoxWidget*);
```

Description

Function Pointer: laCheckBoxWidget_CheckedEvent

laCheckBoxWidget_UncheckedEvent Type

Checkbox unchecked event function callback type

File

[libaria_widget_checkbox.h](#)

C

```
typedef void (* laCheckBoxWidget_UncheckedEvent)(laCheckBoxWidget*);
```

Description

Function Pointer: 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;
    int32_t thickness;
    laBool filled;
} laCircleWidget;
```

Members

Members	Description
laWidget widget;	base widget header
int32_t x;	the origin x coordinate
int32_t y;	the origin y coordinate
int32_t radius;	the radius of the circle
int32_t thickness;	the thickness of the circle outline
laBool filled;	fills the circle area

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.

laCircularGaugeArc_t Structure

Internal structure for the arcs in the circular gauge widget

File

[libaria_widget_circular_gauge.h](#)

C

```
struct laCircularGaugeArc_t {
    laCircularGaugeWidgetArcType type;
    int32_t startAngle;
    int32_t endAngle;
    int32_t startValue;
    int32_t endValue;
    uint32_t radius;
    uint32_t thickness;
    laScheme* scheme;
};
```

Description

Structure: laCircularGaugeArc_t

Describes the arc instances in the circular gauge widget

Remarks

None.

laCircularGaugeLabel_t Structure

Label object for the circular gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
struct laCircularGaugeLabel_t {
    int32_t startValue;
    int32_t endValue;
    uint32_t interval;
    uint32_t radius;
    laCircularGaugeWidgetLabelPosition position;
    laScheme* scheme;
};
```

Description

Structure: typedef struct laCircularGaugeLabel_t

Contains properties of the labels in the gauge

Remarks

None.

laCircularGaugeTick_t Structure

Tick object for the circular gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
struct laCircularGaugeTick_t {
    int32_t startValue;
    int32_t endValue;
    uint32_t interval;
    uint32_t radius;
    uint32_t length;
    laScheme* scheme;
};
```

Description

Structure: laCircularGaugeTick_t

Contains properties of the ticks in the gauge

Remarks

None.

laCircularGaugeWidget_t Structure

Implementation of a circular gauge widget.

File

[libaria_widget_circular_gauge.h](#)

C

```
struct laCircularGaugeWidget_t {
    laWidget widget;
    int32_t value;
    int32_t startValue;
    int32_t endValue;
    uint32_t radius;
    uint32_t startAngle;
    int32_t centerAngle;
    laCircularGaugeWidgetDir dir;
    GFX_Bool ticksVisible;
    uint32_t tickLength;
    int32_t tickValue;
    laBool tickLabelsVisible;
    GFXU_StringTableAsset * stringTable;
    uint32_t ticksLabelsStringID;
    laBool handVisible;
    uint32_t handRadius;
    laBool centerCircleVisible;
    uint32_t centerCircleRadius;
    uint32_t centerCircleThickness;
    laArray arcsArray;
    laArray ticksArray;
    laArray labelsArray;
    laCircularGaugeWidget_ValueChangedEvent cb;
};
```

Members

Members	Description
laWidget widget;	base widget header
int32_t value;	Widget properties
uint32_t radius;	the radius of the circular gauge
uint32_t startAngle;	the start angle of the outer arc
int32_t centerAngle;	the center angle of the outer arc
laCircularGaugeWidgetDir dir;	the turn direction of the gauge
GFX_Bool ticksVisible;	are ticks visible
uint32_t tickLength;	length of ticks (towards center)
int32_t tickValue;	tick value (delta)
laBool tickLabelsVisible;	are tick labels visible
GFXU_StringTableAsset * stringTable;	string table
uint32_t ticksLabelsStringID;	ID of Superset string containing numbers
laBool handVisible;	hand properties
laArray arcsArray;	ArcsArray list
laArray ticksArray;	ArcsArray list
laArray labelsArray;	ArcsArray list
laCircularGaugeWidget_ValueChangedEvent cb;	value changed event callback

Description

Structure: laCircularGaugeWidget_t

A circular gauge widget draws a circular gauge of the specified properties inside the widget bounds. All coordinates are expressed in local widget space.

Remarks

None.

laCircularGaugeWidgetArcType_t Enumeration

Type of arc

File

[libaria_widget_circular_gauge.h](#)

C

```
enum laCircularGaugeWidgetArcType_t {
    ANGLE_ARC,
    VALUE_ARC
};
```

Description

Structure: laCircularGaugeWidgetArcType

Type of arc

Remarks

None.

laCircularGaugeWidgetDir_t Enumeration

Direction of the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
enum laCircularGaugeWidgetDir_t {
    CIRCULAR_GAUGE_DIR_CLOCKWISE,
    CIRCULAR_GAUGE_DIR_COUNTER_CLOCKWISE
};
```

Description

Structure: laCircularGaugeWidgetDir_t

Direction of the gauge

Remarks

None.

laCircularGaugeWidgetLabelPosition_t Enumeration

Direction of the gauge

File

[libaria_widget_circular_gauge.h](#)

C

```
enum laCircularGaugeWidgetLabelPosition_t {
    CIRCULAR_GAUGE_LABEL_OUTSIDE,
    CIRCULAR_GAUGE_LABEL_INSIDE
};
```

Description

Structure: [laCircularGaugeWidgetDir_t](#)

Direction of the gauge

Remarks

None.

laCircularSliderArc_t Structure

Internal structure for the arcs in the circular slider widget

File

[libaria_widget_circular_slider.h](#)

C

```
struct laCircularSliderArc_t {
    laBool visible;
    int32_t startAngle;
    int32_t centerAngle;
    uint32_t radius;
    uint32_t thickness;
    laScheme* scheme;
};
```

Description

Structure: [laCircularSliderArc_t](#)

Describes the arc instances in the circular gauge widget

Remarks

None.

laCircularSliderButtonState_t Enumeration

State of the slider button

File

[libaria_widget_circular_slider.h](#)

C

```
enum laCircularSliderButtonState_t {
    LA_CIRCULAR_SLIDER_STATE_UP,
    LA_CIRCULAR_SLIDER_STATE_DOWN
};
```

Description

Structure: [laCircularSliderButtonState_t](#)

State of the slider button

Remarks

None.

laCircularSliderWidget_t Structure

Implementation of a slider widget.

File

[libaria_widget_circular_slider.h](#)

C

```
struct laCircularSliderWidget_t {
    laWidget widget;
    uint32_t radius;
    uint32_t startAngle;
    uint32_t value;
    uint32_t startValue;
    uint32_t endValue;
    float degPerUnit;
    laBool roundEdges;
    laBool sticky;
    laBool buttonTouch;
    laCircularSliderWidgetDir direction;
    laCircularSliderArc activeArc;
    laCircularSliderArc inActiveArc;
    laCircularSliderArc outsideBorderArc;
    laCircularSliderArc insideBorderArc;
    laCircularSliderArc circleButtonArc;
    laCircularSliderButtonState btnState;
    laCircularSliderWidget_PressedEvent pressedCallback;
    laCircularSliderWidget_ValueChangedEvent valueChangedCallback;
    laCircularSliderWidget_ReleasedEvent releasedCallback;
};
```

Members

Members	Description
laWidget widget;	base widget header
uint32_t radius;	the radius of the slider
uint32_t startAngle;	the start angle of the slider
uint32_t value;	the value of the slider
uint32_t startValue;	the start value of the slider
uint32_t endValue;	the end value of the slider
float degPerUnit;	degrees per unit in the slider
laBool roundEdges;	round edges
laBool sticky;	sends to start value before wrapping around
laBool buttonTouch;	only button is active to touch
laCircularSliderWidgetDir direction;	the direction of the slider
laCircularSliderArc activeArc;	the arc for the slider value
laCircularSliderArc inActiveArc;	the arc for the slider remainder
laCircularSliderArc outsideBorderArc;	the arc for the outside border
laCircularSliderArc insideBorderArc;	the arc for the inside border
laCircularSliderArc circleButtonArc;	the arc for the circle button
laCircularSliderButtonState btnState;	the state of the circular slider button

Description

Structure: laCircularSliderWidget_t

A slider widget draws a slider of the specified origin and radius inside the widget bounds. All coordinates are expressed in local widget space.

The color of the slider is determined by the widget scheme's 'foreground' color.

Remarks

None.

laCircularSliderWidgetArcType_t Enumeration

The arcs that compose the circular slider

File

[libaria_widget_circular_slider.h](#)

C

```
enum laCircularSliderWidgetArcType_t {
    OUTSIDE_CIRCLE_BORDER,
    INSIDE_CIRCLE_BORDER,
    ACTIVE_AREA,
    INACTIVE_AREA,
    CIRCLE_BUTTON
};
```

Description

Enumeration: `laCircularSliderWidgetArcType_t`

The arcs that compose the circular slider

Remarks

None.

laCircularSliderWidgetDir_t Enumeration

Direction of the slider

File

[libaria_widget_circular_slider.h](#)

C

```
enum laCircularSliderWidgetDir_t {
    CIRCULAR_SLIDER_DIR_COUNTER_CLOCKWISE,
    CIRCULAR_SLIDER_DIR_CLOCKWISE
};
```

Description

Structure: `laCircularSliderWidgetDir_t`

Direction of the slider

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.

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.

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.

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.

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

Draw surface draw event function callback type

File

[libaria_widget_drawsurface.h](#)

C

```
typedef laBool (* laDrawSurfaceWidget_DrawCallback)(laDrawSurfaceWidget* sfc, GFX_Rect* bounds);
```

Description

Function Pointer: laDrawSurfaceWidget_DrawCallback

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.

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

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

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.

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.

laImagePlusWidget_ResizeFilter_t Enumeration

File

[libaria_widget_imageplus.h](#)

C

```
enum laImagePlusWidget_ResizeFilter_t {
    LA_IMAGEFILTER_NEARESTNEIGHBOR = 0x0,
    LA_IMAGEFILTER_BILINEAR
};
```

Section

Data Types and Constants

laImagePlusWidget_t Structure

Image plus widget struct definition

File[libaria_widget_imageplus.h](#)**C**

```
struct laImagePlusWidget_t {
    laWidget widget;
    GFXU_ImageAsset* image;
    int32_t transformX;
    int32_t transformY;
    int32_t transformWidth;
    int32_t transformHeight;
    laBool resizeToFit;
    laBool preserveAspect;
    laImagePlusWidget_ResizeFilter resizeFilter;
    laBool inputEnabled;
    GFX_Point touch0;
    laBool touch0_down;
    GFX_Point touch1;
    laBool touch1_down;
    GFX_PixelBuffer buffer;
};
```

Members

Members	Description
laWidget widget ;	widget base class
GFXU_ImageAsset* image ;	pointer to image asset

DescriptionEnumeration: **laImagePlusWidget_t**

An image plus widget displays an image asset and can translate and resize that image.

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

Image changed event function callback type

File

[libaria_widget_imagesequence.h](#)

C

```
typedef void (* laImageSequenceImageChangedEvent_FnPtr)(laImageSequenceWidget*);
```

Description

Function Pointer: 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 activelIdx;	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
<code>laWidget widget;</code>	widget base class
<code>laHAlignment halign;</code>	image horizontal alignment
<code>laVAlignment valign;</code>	image vertical alignment
<code>GFXU_ImageAsset* image;</code>	pointer to image asset
<code>GFXU_ExternalAssetReader* reader;</code>	asset reader

Description

Enumeration: `laImageWidget_t`

An image widget displays an image asset.

Remarks

None.

laImageWidget_DrawEventCallback Type

File

[libaria_widget_image.h](#)

C

```
typedef void (* laImageWidget_DrawEventCallback)(laImageWidget*);
```

Section

Data Types and Constants

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_ESC,
    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.

laKeyPadActionTrigger Enumeration

Defines the trigger for keypad action and events

File

[libaria_widget_keypad.h](#)

C

```

typedef enum laKeyPadActionTrigger_t {
    LA_KEYPAD_TRIGGER_KEYRELEASED,
    LA_KEYPAD_TRIGGER_KEYPRESSED
} laKeyPadActionTrigger;

```

Description

Structure: laKeyPadActionTrigger_t

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_GetKeyPadActionTrigger Function

Gets the current trigger for keypad edit action and events

File

[libaria_widget_keypad.h](#)

C

```
LIB_EXPORT laKeyPadActionTrigger laKeyPadWidget_GetKeyPadActionTrigger(laKeyPadWidget* pad);
```

Returns

[laKeyPadActionTrigger](#) - the trigger

Parameters

Parameters	Description
laKeyPadWidget* pad	the widget

Function

[laKeyPadActionTrigger laKeyPadWidget_SetKeyPadActionTrigger\(laKeyPadWidget* pad\)](#)

laKeyPadWidget_KeyClickEvent Type

Key click event function callback type

File

[libaria_widget_keypad.h](#)

C

```
typedef void (* laKeyPadWidget_KeyClickEvent)(laKeyPadWidget*, laButtonWidget*, uint32_t,
uint32_t);
```

Description

Function Pointer: [laKeyPadWidget_KeyClickEvent](#)

laKeyPadWidget_SetKeyPadActionTrigger Function

Sets the current trigger for keypad edit action and events

File

[libaria_widget_keypad.h](#)

C

```
LIB_EXPORT laResult laKeyPadWidget_SetKeyPadActionTrigger(laKeyPadWidget* pad,
laKeyPadActionTrigger trigger);
```

Returns

[laResult](#) - the result of the operation

Description

Keypad actions and events callback will be called based on the trigger

Parameters

Parameters	Description
laKeyPadWidget* pad	the widget
laKeyPadActionTrigger trigger	the trigger

Function

```
laResult laKeyPadWidget_SetKeyPadActionTrigger(laKeyPadWidget* pad,
laKeyPadActionTrigger trigger)
```

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;
    int32_t textLineSpace;
} 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
int32_t textLineSpace;	new line space per pixel

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.

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.

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.

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.

laLineGraphCategory_t Structure

Contains the Category properties

File

[libaria_widget_line_graph.h](#)

C

```
struct laLineGraphCategory_t {
    laString text;
    int32_t stackValue;
};
```

Description

Structure: laLineGraphCategory_t

Remarks

None.

laLineGraphDataPointType_t Enumeration

The graph data point type

File

[libaria_widget_line_graph.h](#)

C

```
enum laLineGraphDataPointType_t {
    LINE_GRAPH_DATA_POINT_NONE,
    LINE_GRAPH_DATA_POINT_CIRCLE,
    LINE_GRAPH_DATA_POINT_SQUARE
};
```

Description

Enumeration: laLineGraphDataPointType_t

laLineGraphDataSeries_t Structure

The data series object that contains the series properties and data

File

[libaria_widget_line_graph.h](#)

C

```
struct laLineGraphDataSeries_t {
    laScheme * scheme;
    laArray data;
    laLineGraphValueAxis axis;
    laLineGraphDataPointType pointType;
    uint32_t pointSize;
    laBool fillPoints;
    laBool drawLines;
};
```

Description

Structure: laLineGraphDataSeries_t

Remarks

None.

laLineGraphTickPosition_t Enumeration

The tick position relative to axis

File

[libaria_widget_line_graph.h](#)

C

```
enum laLineGraphTickPosition_t {
    LINE_GRAPH_TICK_IN,
    LINE_GRAPH_TICK_OUT,
    LINE_GRAPH_TICK_CENTER
};
```

Description

Enumeration: laLineGraphTickPosition_t

laLineGraphValueAxis_t Enumeration

The value axis index value

File

[libaria_widget_line_graph.h](#)

C

```
enum laLineGraphValueAxis_t {
    LINE_GRAPH_AXIS_0 = 0
};
```

Description

Enumeration: laLineGraphValueAxis_t

laLineGraphWidget_t Structure

Implementation of a line graph widget.

File

[libaria_widget_line_graph.h](#)

C

```
struct laLineGraphWidget_t {
    laWidget widget;
    uint32_t tickLength;
    laBool fillGraphArea;
    laBool fillValueArea;
    int32_t maxValue;
    int32_t minValue;
    uint32_t tickInterval;
    uint32_t subtickInterval;
    laBool valueAxisLabelsVisible;
    laBool valueAxisTicksVisible;
    laBool valueAxisSubticksVisible;
    laBool valueGridlinesVisible;
    laBool stacked;
    laArray dataSeries;
    GFXU_StringTableAsset * stringTable;
    uint32_t ticksLabelsStringID;
    laLineGraphTickPosition valueAxisTicksPosition;
    laLineGraphTickPosition valueAxisSubticksPosition;
    laBool categAxisLabelsVisible;
    laBool categAxisTicksVisible;
    laLineGraphTickPosition categAxisTicksPosition;
    laArray categories;
    GFXU_ExternalAssetReader* reader;
};
```

Members

Members	Description
laWidget widget;	base widget header
int32_t maxValue;	Value axis properties
GFXU_StringTableAsset * stringTable;	string table
uint32_t ticksLabelsStringID;	ID of Superset string containing numbers
laBool categAxisLabelsVisible;	Category axis properties
GFXU_ExternalAssetReader* reader;	asset reader

Description

Structure: laLineGraphWidget_t

A line graph widget draws a line graph. All coordinates are expressed in local widget space.

The color of the graph is determined by the widget scheme's 'foreground' color.

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.

laList Type

Linked list definition

File[libaria_utils.h](#)**C**

```
typedef struct laList_t laList;
```

DescriptionStructure: [laList_t](#)**Remarks**

None.

laList_t Structure

Linked list definition

File[libaria_list.h](#)**C**

```
struct laList_t {
    laListNode* head;
    laListNode* tail;
    size_t size;
};
```

DescriptionStructure: [laList_t](#)**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;
    laBool enabled;
} 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
laBool enabled;	enable or disable the item

Description

Structure: laListItem_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.

laListWheelIndicatorFill Enumeration

Indicates the fill type for the listwheel indicator area.

File

[libaria_widget_listwheel.h](#)

C

```
typedef enum laListWheelIndicatorFill_t {
    LA_LISTWHEEL_INDICATOR_FILL_NONE,
    LA_LISTWHEEL_INDICATOR_FILL_SOLID,
    LA_LISTWHEEL_INDICATOR_FILL_GRADIENT
} laListWheelIndicatorFill;
```

Description

Enumeration: laListWheelIndicatorFill

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;	determines if the background of the widget uses gradient shading to show depth
int32_t cycleDistance;	determines 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_GetAutoHideWheel Function

Returns the list wheel auto hide setting

File

[libaria_widget_listwheel.h](#)

C

```
LIB_EXPORT laBool laListWheelWidget_GetAutoHideWheel(laListWheelWidget* whl);
```

Returns

`laBool` - true if the list wheel is set to auto hide

Parameters

Parameters	Description
laListWheelWidget* whl	the widget

Function

[laListWheelIndicatorFill](#) laListWheelWidget_GetAutoHideWheel([laListWheelWidget*](#) whl)

laListWheelWidget_GetIndicatorFill Function

Gets the indicator area fill type

File

[libaria_widget_listwheel.h](#)

C

```
LIB_EXPORT laListWheelIndicatorFill laListWheelWidget_GetIndicatorFill(laListWheelWidget* whl);
```

Returns

[laListWheelIndicatorFill](#) - the indicator area fill type

Parameters

Parameters	Description
laListWheelWidget* whl	the widget

Function

[laListWheelIndicatorFill](#) laListWheelWidget_GetIndicatorFill([laListWheelWidget*](#) whl)

laListWheelWidget_GetZoomEffects Function

Gets the list wheel zoom effect

File

[libaria_widget_listwheel.h](#)

C

```
LIB_EXPORT laListWheelZoomEffects laListWheelWidget_GetZoomEffects(laListWheelWidget* whl);
```

Returns

[laListWheelZoomEffects](#) - the list wheel zoom effect type

Parameters

Parameters	Description
laListWheelWidget* whl	the widget

Function

[laListWheelZoomEffects](#) laListWheelWidget_GetZoomEffects([laListWheelWidget*](#) whl)

laListWheelWidget_SelectedItemChangedEvent Type

Selected item changed event function callback type

File

[libaria_widget_listwheel.h](#)

C

```
typedef void (* laListWheelWidget_SelectedItemChangedEvent)(laListWheelWidget*, uint32_t idx);
```

Description

Function Pointer: laListWheelWidget_SelectedItemChangedEvent

laListWheelWidget_SetAutoHideWheel Function

Sets the list wheel to auto hide when not active

File

[libaria_widget_listwheel.h](#)

C

```
LIB_EXPORT laResult laListWheelWidget_SetAutoHideWheel(laListWheelWidget* whl, laBool autoHide);
```

Returns

laResult - the operation result

Description

Sets the list wheel to auto hide when not active

Parameters

Parameters	Description
laListWheelWidget* whl	the widget
laBool autoHide	sets the list wheel to auto hide

Function

laResult laListWheelWidget_SetAutoHideWheel([laListWheelWidget*](#) whl, [laBool](#) autoHide)

laListWheelWidget_SetIndicatorFill Function

Sets the indicator fill type

File

[libaria_widget_listwheel.h](#)

C

```
LIB_EXPORT laResult laListWheelWidget_SetIndicatorFill(laListWheelWidget* whl,  
laListWheelIndicatorFill fill);
```

Returns

laResult - the operation result

Description

Sets the indicator fill type

Parameters

Parameters	Description
laListWheelWidget* whl	the widget
laListWheelIndicatorFill fill	fill type

Function

laResult laListWheelWidget_SetIndicatorFill([laListWheelWidget*](#) whl, [laListWheelIndicatorFill](#) fill)

laListWheelWidget_SetZoomEffects Function

Sets the list wheel zoom effect

File

[libaria_widget_listwheel.h](#)

C

```
LIB_EXPORT laResult laListWheelWidget_SetZoomEffects(laListWheelWidget* whl,
laListWheelZoomEffects zoomEffects);
```

Returns

[laResult](#) - the operation result

Description

Sets the list wheel zoom effect

Parameters

Parameters	Description
laListWheelWidget* whl	the widget
laListWheelZoomEffects zoomEffects	the zoom effect

Function

[laResult laListWheelWidget_SetZoomEffects\(laListWheelWidget* whl, laListWheelZoomEffects zoomEffects\)](#)

laListWheelZoomEffects Enumeration

Indicates the zoom effects for the list wheel items.

File

[libaria_widget_listwheel.h](#)

C

```
typedef enum laListWheelZoomEffects_t {
    LA_LISTWHEEL_ZOOM_EFFECT_NONE,
    LA_LISTWHEEL_ZOOM_EFFECT_VSCALE,
    LA_LISTWHEEL_ZOOM_EFFECT_FULL_SCALE,
    LA_LISTWHEEL_ZOOM_EFFECT_FIXED_SCALE
} laListWheelZoomEffects;
```

Description

Enumeration: laListWheelEffects

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_SelectedItemChangedEvent 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_SelectedItemChangedEvent 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_SelectedItemChangedEvent Type

Selected item changed event function callback type

File

[libaria_widget_list.h](#)

C

```

typedef void (* laListWidget_SelectedItemChangedEvent)(laListWidget*, uint32_t idx, laBool
selected);

```

Description

Function Pointer: 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 series is allowed at any one time

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.

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.

laPieChartPie_t Structure

File

[libaria_widget_pie_chart.h](#)

C

```
struct laPieChartPie_t {
    uint32_t value;
    uint32_t radius;
    uint32_t offset;
    laScheme* scheme;
};
```

Section

Data Types and Constants

laPieChartWidget_t Structure

Implementation of a pie chart widget.

File

[libaria_widget_pie_chart.h](#)

C

```
struct laPieChartWidget_t {
    laWidget widget;
    uint32_t startAngle;
    int32_t centerAngle;
    laArray pieArray;
    laBool labelsVisible;
    uint32_t labelsOffset;
    GFXU_StringTableAsset * stringTable;
    uint32_t labelsStringID;
    laPieChartWidget_PressedEvent pressedCallback;
};
```

Members

Members	Description
laWidget widget;	base widget header
uint32_t startAngle;	the start angle of the chart
int32_t centerAngle;	the center angle of the chart
laArray pieArray;	list of pie/data
laBool labelsVisible;	are labels visible
uint32_t labelsOffset;	offset of labels from center of pie
GFXU_StringTableAsset * stringTable;	string table
uint32_t labelsStringID;	ID of Superset string containing numbers

Description

Structure: laPieChartWidget_t

A chart widget draws a chart of the specified origin and radius inside the widget bounds. All coordinates are expressed in local widget space.

The color of the chart is determined by the widget scheme's 'foreground' color.

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.

laProgressBar_ValueChangedEventCallback Type

Value changed event function callback type

File

[libaria_widget_progressbar.h](#)

C

```
typedef void (* laProgressBar_ValueChangedEventCallback)(laProgressBar*, uint32_t);
```

Description

Function Pointer: 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.

laRadialMenuEllipseType_t Enumeration**File**

[libaria_widget_radial_menu.h](#)

C

```
enum laRadialMenuEllipseType_t {
```

```

LA_RADIAL_MENU_ELLIPSE_TYPE_DEFAULT,
LA_RADIAL_MENU_ELLIPSE_TYPE_ORBITAL,
LA_RADIAL_MENU_ELLIPSE_TYPE_ROLLODEX
} ;

```

Description

This is record `laRadialMenuItemType_t`.

laRadialMenuItemNode_t Structure

File

[libaria_widget_radial_menu.h](#)

C

```

struct laRadialMenuItemNode_t {
    laWidget* widget;
    int32_t t;
    laWidget_TouchDownEvent_FnPtr origTouchDown;
    laWidget_TouchUpEvent_FnPtr origTouchUp;
    laWidget_TouchMovedEvent_FnPtr origTouchMoved;
    GFX_Rect origSize;
    uint32_t origAlphaAmount;
} ;

```

Members

Members	Description
<code>laWidget* widget;</code>	point to the widget of the item
<code>int32_t t;</code>	the angle in degrees between 0 - 360, representing the relative position of the item on the track
<code>laWidget_TouchDownEvent_FnPtr origTouchDown;</code>	the widget item's original touch down event, allows the radial menu to work as a hub to route to the appropriate widget
<code>laWidget_TouchUpEvent_FnPtr origTouchUp;</code>	the widget item's original touch up event allows the radial menu to work as a hub to route to the appropriate widget
<code>laWidget_TouchMovedEvent_FnPtr origTouchMoved;</code>	the widget item's original touch move event
<code>GFX_Rect origSize;</code>	the original size of the widget, it is a reference point for scaling
<code>uint32_t origAlphaAmount;</code>	the original alpha value of the widget, it is a reference point for scaling

Description

This is record `laRadialMenuItemNode_t`.

laRadialMenuItemWidget_t Structure

Implementation of a radial menu widget struct

File

[libaria_widget_radial_menu.h](#)

C

```

struct laRadialMenuItemWidget_t {
    laWidget widget;
    laRadialMenuItemWidgetState state;
    int32_t prominentIndex;
    int32_t lastProminentIndex;
    int32_t userRequestedAngleDiff;
    int32_t targetAngleDiff;
    int32_t userRequestedDirection;
}

```

```

laBool drawEllipse;
laBool highlightProminent;
laHAlignment halign;
laVAlignment valign;
laImageWidget* highlighter;
int32_t a;
int32_t b;
int32_t theta;
laBool touchPressed;
laBool ellipseChanged;
laRadialMenuWidgetScaleType scaleItemSize;
int32_t maxSizePercent;
int32_t minSizePercent;
GFX_Rect touchArea;
laRadialMenuWidgetScaleType scaleItemAlpha;
int32_t maxAlphaAmount;
int32_t minAlphaAmount;
int32_t itemsShown;
laList widgetList;
laList shownList;
laList hiddenList;
laMenuItemNode* widestWidgetItem;
laMenuItemNode* tallestWidgetItem;
laRadialMenuWidget_ItemSelectedEvent itemSelectedEvent;
laRadialMenuWidget_ItemProminenceChangedEvent itemProminenceChangedEvent;
laRadialMenuEllipseType ellipseType;
GFXU_ExternalAssetReader* reader;
};

}

```

Members

Members	Description
laWidget widget;	widget base class
int32_t userRequestedAngleDiff;	the angle for the radial menu to rotate as requested by user
int32_t targetAngleDiff;	the angle for the radial menu to rotate for prominent item to be in front
int32_t userRequestedDirection;	tracks the direction that the user requested for rotation
laBool drawEllipse;	indicates if the radial menu is selected
laBool highlightProminent;	highlight the prominent widget
laHAlignment halign;	horizontal alignment
laVAlignment valign;	vertical alignment
laImageWidget* highlighter;	this widget manages the selector art asset
int32_t a;	the half-length of the 0-180 axis of the ellipse
int32_t b;	the half-length of the 90-270 axis of the ellipse
int32_t theta;	the angle of rotation of the entire ellipse
laBool touchPressed;	keep track of users touch input
laBool ellipseChanged;	keeps track if the elliptical track has changed
laRadialMenuWidgetScaleType scaleItemSize;	the enable item size scaling within the widget
int32_t maxSizePercent;	the maximum size scale between 1 - 200
int32_t minSizePercent;	the minimum size scale between 1 - 200
GFX_Rect touchArea;	the area specified within the widget that touch events are detected
laRadialMenuWidgetScaleType scaleItemAlpha;	the enable item alpha scaling within the widget
int32_t maxAlphaAmount;	the maximum alpha between 0 - 255
int32_t minAlphaAmount;	the minimum alpha between 0 - 255
int32_t itemsShown;	keeps count of how many items to visible, this number should be less than or equal to total number of widget items
laList widgetList;	this is the list of widgets
laList shownList;	this is the partial list of widgets shown

<code>laList hiddenList;</code>	this is the partial list of widgets hidden
<code>laRadialMenuItemNode* widestWidgetItem;</code>	keeps track of which widget is the widest for major axis calculation
<code>laRadialMenuItemNode* tallestWidgetItem;</code>	keeps track of which widget is the tallest for minor axis calculation
<code>laRadialMenuWidget_ItemSelectedEvent itemSelectedEvent;</code>	an item is selected event callback
<code>laRadialMenuWidget_ItemProminenceChangedEvent itemProminenceChangedEvent;</code>	whenever a new item is in prominence event callback
<code>GFXU_ExternalAssetReader* reader;</code>	asset reader

Description

Enumeration: `laRadialMenuWidget_t`

A radial menu is a master widget that manages the movement, in an elliptical track, of a list of widgets. It also manages the draw order and scaling of each widget item.

It is essentially a group of widgets which provides a mutually exclusive selection capability so that only one item may be selected at any one time.

Remarks

None.

laRadialMenuWidgetScaleType_t Enumeration

File

[libaria_widget_radial_menu.h](#)

C

```
enum laRadialMenuWidgetScaleType_t {
    LA_RADIAL_MENU_SCALE_OFF,
    LA_RADIAL_MENU_SCALE_GRADUAL,
    LA_RADIAL_MENU_SCALE_PROMINENT
};
```

Description

This is record `laRadialMenuWidgetScaleType_t`.

laRadialMenuWidgetState_t Enumeration

File

[libaria_widget_radial_menu.h](#)

C

```
enum laRadialMenuWidgetState_t {
    LA_RADIAL_MENU_INIT,
    LA_RADIAL_MENU_INPUT_READY,
    LA_RADIAL_MENU_HANDLE_USER_MOVE_REQUEST,
    LA_RADIAL_MENU_RESET_TO_INPUT_POS
};
```

Section

Data Types and Constants

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.

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.

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;	button 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

Radio button deselected function callback type

File

[libaria_widget_radiobutton.h](#)

C

```
typedef void (* laRadioButtonWidget_DeselectedEvent)(laRadioButtonWidget*);
```

Description

Function Pointer: laRadioButtonWidget_DeselectedEvent

laRadioButtonWidget_SelectedEvent Type

Radio button selected function callback type

File

[libaria_widget_radiobutton.h](#)

C

```
typedef void (* laRadioButtonWidget_SelectedEvent)(laRadioButtonWidget*);
```

Description

Function Pointer: 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.

laRectArray Type

Rectangle array definition

File

[libaria_widget.h](#)

C

```
typedef struct laRectArray_t laRectArray;
```

Description

Structure: laRectArray_t

Remarks

None.

laRectArray_t Structure

Rectangle array definition

File

[libaria_rectarray.h](#)

C

```
struct laRectArray_t {
    GFX_Rect* rects;
    uint32_t size;
    uint32_t capacity;
};
```

Description

Structure: laRectArray_t

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.

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.

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 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.

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

Value changed event function callback type

File

[libaria_widget_scrollbar.h](#)

C

```
typedef void (* laScrollBarWidget_ValueChangedEvent)(laScrollBarWidget*);
```

Description

Function Pointer: 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

Value changed event function callback type

File

[libaria_widget_slider.h](#)

C

```
typedef void (* laSliderWidget_ValueChangedEvent)(laSliderWidget*);
```

Description

Function Pointer: laSliderWidget_ValueChangedEvent

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.

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;
    laTextFieldWidget_FocusChangedCallback focusChangedEvent;
    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 visibility flag
laBool clearOnFirstEdit;	needs clear on first edit
laTextFieldWidget_TextChangedCallback textChangedEvent;	text changed event
laTextFieldWidget_FocusChangedCallback focusChangedEvent;	focus 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

Text changed event function callback type

File

[libaria_widget_textfield.h](#)

C

```
typedef void (* laTextFieldWidget_TextChangedCallback)(laTextFieldWidget*);
```

Description

Function Pointer: 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

Point added event function callback type

File

[libaria_widget_touchtest.h](#)

C

```
typedef void (* laTouchTestWidget_PointAddedEventCallback)(laTouchTestWidget*, GFX_Point*);
```

Description

Function Pointer: laTouchTestWidget_PointAddedEventCallback

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.

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;	language 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.

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

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.

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.

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.

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.

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.

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_rectarray.h	An array implementation for storing rectangles for the Aria user interface library
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_arc.h	
libaria_widget_bar_graph.h	
libaria_widget_button.h	Defines a button widget
libaria_widget_checkbox.h	
libaria_widget_circle.h	
libaria_widget_circular_gauge.h	
libaria_widget_circular_slider.h	
libaria_widget_drawsurface.h	
libaria_widget_gradient.h	
libaria_widget_groupbox.h	
libaria_widget_image.h	
libaria_widget_imageplus.h	
libaria_widget_imagesequence.h	
libaria_widget_keypad.h	
libaria_widget_label.h	
libaria_widget_line.h	
libaria_widget_line_graph.h	
libaria_widget_list.h	
libaria_widget_listwheel.h	
libaria_widget_pie_chart.h	
libaria_widget_progressbar.h	
libaria_widget_radial_menu.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

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.

	IaContextFrameState	Possible values for context frame state.
	IaContextUpdateState	Possible values for context update state.

Functions

	Name	Description
≡◊	IaContext_AddScreen	Add screen to the list of screens in the current context
≡◊	IaContext_Create	Creates an instance of the Aria user interface library
≡◊	IaContext_Destroy	Destroys an Aria instance
≡◊	IaContext_GetActive	Returns the current active context.
≡◊	IaContext_GetActiveScreen	Returns the active screen of the current context
≡◊	IaContext_GetActiveScreenIndex	Return the index of the active screen
≡◊	IaContext_GetColorMode	Returns the color mode of the current context
≡◊	IaContext_GetDefaultScheme	Returns the pointer to the default scheme of the current context
≡◊	IaContext_GetEditWidget	Gets the widget that is currently receiving all widget edit events.
≡◊	IaContext_GetFocusWidget	Return a pointer to the widget in focus
≡◊	IaContext_GetPreemptionLevel	Returns the preemption level for the screen
≡◊	IaContext_GetScreenRect	Returns the display rectangle structure of the physical display
≡◊	IaContext_GetStringLanguage	Returns the language index of the current context
≡◊	IaContextGetStringTable	Get a pointer to the GFUX_StringTableAsset structure that maintains the strings, associated fonts, etc
≡◊	IaContext_HideActiveScreen	Hide the active screen
≡◊	IaContext_IsDrawing	Indicates if any layers of the active screen are currently drawing a frame.
≡◊	IaContext_IsLayerDrawing	Indicates if the layer at the given index of the active screen is currently drawing.
≡◊	IaContext_RedrawAll	Forces the library to redraw the currently active screen in its entirety.
≡◊	IaContext_RemoveScreen	Remove the specified screen from the list of screens in the current context
≡◊	IaContext_SetActive	Make the specified context active
≡◊	IaContext_SetActiveScreen	Change the active screen to the one specified by the index argument
≡◊	IaContext_SetActiveScreenChangedCallback	Set the callback function pointer when the screen change event occurs
≡◊	IaContext_SetEditWidget	Sets the currently active edit widget.
≡◊	IaContext_SetFocusWidget	Set into focus the widget specified as the argument
≡◊	IaContext_SetLanguageChangedCallback	Set the callback function pointer when the language change event occurs
≡◊	IaContext_SetPreemptionLevel	Set the preemption level to the specified value
≡◊	IaContext_SetStringLanguage	Set the language index of the current context
≡◊	IaContext_SetStringTable	Set the StringTable pointer to the specified new StringTableAsset structure
≡◊	IaContext_Update	Runs the update loop for a library instance.

Structures

	Name	Description
◆◊	IaContext_t	An instance of the Aria user interface library.

Types

	Name	Description
	IaContext_ActiveScreenChangedCallback_FnPtr	Callback pointer for the active screen change notification.
	IaContext_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**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

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
	<code>laGestureID_t</code>	Placeholder for eventual gesture support.
	<code>laKey_t</code>	All values possible for key entry from the libaria keyboard widget
	<code>laMouseButton_t</code>	All values possible for mouse key entry from the libaria mouse input
	<code>laGestureID</code>	Placeholder for eventual gesture support.
	<code>laKey</code>	All values possible for key entry from the libaria keyboard widget
	<code>laMouseButton</code>	All values possible for mouse key entry from the libaria mouse input

Functions

	Name	Description
	<code>laInput_GetEnabled</code>	Returns the input enabled status of the current context
	<code>laInput.InjectTouchDown</code>	Register and track the touch down event and queue it for handling by associated widgets
	<code>laInput.InjectTouchMoved</code>	Register and track the touch moved event and queue it for handling by associated widgets
	<code>laInput.InjectTouchUp</code>	Register and track the touch up event and queue it for handling by associated widgets
	<code>laInput_SetEnabled</code>	Sets the input status of the current context with the specified input argument

Macros

	Name	Description
	<code>NUM_BUTTONS</code>	This is macro NUM_BUTTONS.
	<code>NUM_KEYS</code>	This is macro NUM_KEYS.

Structures

	Name	Description
	<code>laInput_TouchDownEvent_t</code>	Register and handle the touch press detect event
	<code>laInput_TouchMovedEvent_t</code>	Register and handle the touch coordinates changed event
	<code>laInput_TouchUpEvent_t</code>	Register and handle the touch release detect event
	<code>laInputState_t</code>	Maintain a history of touch states; currently libaria keeps track of the last touch state only.
	<code>laTouchState_t</code>	Manage the touch input state and track the touch coordinate
	<code>laInput_TouchDownEvent</code>	Register and handle the touch press detect event
	<code>laInput_TouchMovedEvent</code>	Register and handle the touch coordinates changed event
	<code>laInput_TouchUpEvent</code>	Register and handle the touch release detect event
	<code>laInputState</code>	Maintain a history of touch states; currently libaria keeps track of the last touch state only.
	<code>laTouchState</code>	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_rectarray.h

An array implementation for storing rectangles for the Aria user interface library

Functions

	Name	Description
≡	laRectArray_Clear	Removes all values from a given array. Array capacity remains the same.
≡	laRectArray_Copy	Creates a duplicate of an existing array
≡	laRectArray_Create	Initializes a new rectangle array.
≡	laRectArray_Destroy	Removes all nodes from a given array and frees the memory owned by the array. Resets array capacity to zero.
≡	laRectArray_InsertAt	Inserts a rectangle into an array at a given index. All existing nodes from index are shifted right one place.
≡	laRectArray_MergeSimilar	Analyzes an array and merges any rectangles similar in size.
≡	laRectArray_PopBack	Removes the last rectangle from the array
≡	laRectArray_PopFront	Removes the first value from the array. Shuffles all other nodes forward one index.
≡	laRectArray_PushBack	Pushes a new rectangle onto the back of the array
≡	laRectArray_PushFront	Pushes a new rectangle onto the front of the array. Shuffles all other nodes backward one index.
≡	laRectArray_RemoveAt	Removes a rectangle from the array at an index
≡	laRectArray_RemoveDuplicates	Removes any duplicate rectangles from an array.
≡	laRectArray_RemoveOverlapping	Sorts the array by size and then removes any rectangles that are completely overlapped by another larger rectangle.

	laRectArray_Resize	Resizes the capacity of the array. If the array shrinks, any nodes beyond the new capacity will be discarded.
	laRectArray_SortBySize	Sorts a given array largest to smallest.

Structures

	Name	Description
	laRectArray_t	Rectangle array definition

Description

Module for Microchip Graphics Library - Aria User Interface Library

This is an array 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_rectarray.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 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 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.

<code>laString_Clear</code>	Sets a string's length to zero and its string table reference to NULL. Does not free any associated data and preserves capacity.
<code>laString_Compare</code>	Compares two string objects
<code>laString_CompareBuffer</code>	Compares a string object and a <code>GFXU_CHAR*</code> buffer
<code>laString_Copy</code>	Copies the values from one string into another
<code>laString_CreateFromBuffer</code>	Creates a string object from a <code>GFXU_CHAR</code> buffer and a font asset pointer
<code>laString_CreateFromCharBuffer</code>	Creates a string object from a const <code>char*</code> buffer and a font asset pointer. This method provides compatibility with standard c-style strings. Input string will be converted from 8-bit width to 32-bit width.
<code>laString_CreateFromID</code>	Creates a string object that simply references a string in the string table.
<code>laString_Delete</code>	Deletes all memory associated with a string object
<code>laString_Destroy</code>	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.
<code>laString_Draw</code>	Wrapper around GFX Utility string draw function for Aria user interface library. Internal use only.
<code>laString_DrawClipped</code>	Wrapper around GFX Utility string draw function for Aria user interface library. Draws only a clipped area of a string. Internal use only.
<code>laString_DrawSubStringClipped</code>	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.
<code>laString_ExtractFromTable</code>	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.
<code>laString_GetAscent</code>	Returns the ascent of a string by referencing its associated font asset data.
<code>laString_GetCharIndexAtPoint</code>	Given an offset in pixels returns the corresponding character index.
<code>laString_GetCharOffset</code>	Returns the offset of a given character index in pixels.
<code>laString_GetCharWidth</code>	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.
<code>laString_GetHeight</code>	Returns the height of a string by referencing its associated font asset data.
<code>laString_GetLineRect</code>	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.
<code>laString_GetMultiLineRect</code>	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 is the sum of the heights of the bounding rectangles for each line and the width is the widest among the bounding rectangles.
<code>laString_GetRect</code>	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.
<code>laString_Initialize</code>	Initializes a string struct to default
<code>laString_Insert</code>	Inserts a string into another string at a given index
<code>laString_IsEmpty</code>	Returns a boolean indicating if the provided string contains data or has a link to the string table.
<code>laString_Length</code>	Calculates the length of a string in characters
<code>laString_New</code>	Allocates a memory for a new string
<code>laString_ReduceLength</code>	Reduces the length of a string. This simply slides the null terminator to the left and does not affect the string's capacity value.
<code>laString_Remove</code>	Removes a number of characters from a string at a given index
<code>laString_Set</code>	Attempts to set the local data buffer of a string to an input buffer
<code>laString_SetCapacity</code>	Attempts to adjust the capacity of a string
<code>laString_ToCharBuffer</code>	Extracts the data buffer from a string and copies it into the provided buffer argument.

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 character 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 in 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	Returns LA_TRUE if the specified widget is occluded in the specified rectangle area.
	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_GetCornerRadius	Returns the corner radius of the 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
≡◊	laWidgetGetX	Returns the widget rectangles upper left corner x-coordinate
≡◊	laWidgetGetY	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_SetCornerRadius	Sets the widget corner radius.
≡◊	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.

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	Rectangle array definition

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_arc.h

Functions

	Name	Description
	laArcWidget_GetCenterAngle	Gets the center angle of the arc widget
	laArcWidget_GetRadius	Gets the radius of a arc widget
	laArcWidget_GetRoundEdge	Returns true if the arc has round edges
	laArcWidget_GetStartAngle	Returns the start angle of a arc widget
	laArcWidget_GetThickness	Gets the thickness of the arc
	laArcWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	laArcWidget_SetCenterAngle	Sets the center angle of the arc widget
	laArcWidget_SetRadius	Sets the radius of a arc widget
	laArcWidget_SetRoundEdge	Sets the arc edge to round
	laArcWidget_SetStartAngle	Sets the start angle of a arc widget
	laArcWidget_SetThickness	Sets the thickness of the arc widget

Structures

	Name	Description
	laArcWidget_t	Implementation of a arc widget.

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements arc drawing widget functions.

File Name

libaria_widget_arc.h

Company

Microchip Technology Inc.

libaria_widget_bar_graph.h

Enumerations

	Name	Description
	laBarGraphTickPosition_t	The tick position relative to axis
	laBarGraphValueAxis_t	The value axis index value

Functions

	Name	Description
	laBarGraphWidget_AddCategory	Adds a category to the graph
	laBarGraphWidget_AddDataToSeries	Adds a data (value) to the specified series at categoryID index
	laBarGraphWidget_AddSeries	Adds a series to the graph
	laBarGraphWidget_DestroyAll	Destroys data, series and categories and frees the memory allocated
	laBarGraphWidget_GetCategoryAxisLabelsVisible	Returns GFX_TRUE if the category axis labels are visible
	laBarGraphWidget_GetCategoryAxisTicksPosition	Returns the position of the ticks in the category axis
	laBarGraphWidget_GetCategoryAxisTicksVisible	Returns GFX_TRUE if the category axis ticks are visible
	laBarGraphWidget_GetCategoryText	Gets a copy of the string used to label the category
	laBarGraphWidget_GetFillGraphArea	Returns GFX_TRUE if the category axis labels are visible
	laBarGraphWidget_GetGridlinesVisible	Returns GFX_TRUE if the axis gridlines are visible
	laBarGraphWidget_GetMaxValue	Returns the max value of the axis
	laBarGraphWidget_GetMinValue	Returns the min value of the axis
	laBarGraphWidget_GetSeriesScheme	Returns scheme of the specified series
	laBarGraphWidget_GetStacked	Returns GFX_TRUE if the bars are stacked
	laBarGraphWidget_GetTickLength	Returns the length of the ticks
	laBarGraphWidget_GetValueAxisLabelsVisible	Returns GFX_TRUE if the value axis labels are visible
	laBarGraphWidget_GetValueAxisSubtickInterval	Returns the interval between minor ticks in the value axis
	laBarGraphWidget_GetValueAxisSubticksPosition	Returns the position of the subticks in the axis
	laBarGraphWidget_GetValueAxisSubticksVisible	Returns GFX_TRUE if the value axis subticks are visible
	laBarGraphWidget_GetValueAxisTickInterval	Returns the interval between major ticks in the value axis
	laBarGraphWidget_GetValueAxisTicksPosition	Returns the position of the ticks in the axis
	laBarGraphWidget_GetValueAxisTicksVisible	Returns GFX_TRUE if the value axis ticks are visible
	laBarGraphWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	laBarGraphWidget_SetCategoryAxisLabelsVisible	Shows/Hides the category axis labels
	laBarGraphWidget_SetCategoryAxisTicksPosition	Sets the position of the ticks in the category axis

	laBarGraphWidget_SetCategoryAxisTicksVisible	Shows/Hides the category axis ticks
	laBarGraphWidget_SetCategoryText	Sets the string used to label the category
	laBarGraphWidget_SetDataInSeries	Sets the value of the entry in the series index. The entry should have been previously
	laBarGraphWidget_SetFillGraphArea	Sets the graph area filled or not
	laBarGraphWidget_SetGridlinesVisible	Shows/Hides the gridlines
	laBarGraphWidget_SetMaxValue	Sets the max value of the axis
	laBarGraphWidget_SetMinValue	Sets the min value of the axis
	laBarGraphWidget_SetSeriesScheme	Sets the color scheme of the series
	laBarGraphWidget_SetStacked	Stacks the bar graph
	laBarGraphWidget_SetStringTable	Sets the string table used for the generated axis labels
	laBarGraphWidget_SetTickLength	Sets the length of the ticks
	laBarGraphWidget_SetTicksLabelsStringID	Sets the ID of the superset string used for the value axis tick labels
	laBarGraphWidget_SetValueAxisLabelsVisible	Shows/Hides the labels in the value axis
	laBarGraphWidget_SetValueAxisSubtickInterval	Sets the minor tick interval in the value axis
	laBarGraphWidget_SetValueAxisSubticksPosition	Sets the position of the subticks in the value axis
	laBarGraphWidget_SetValueAxisSubticksVisible	Shows/Hides the subticks in the value axis
	laBarGraphWidget_SetValueAxisTickInterval	Sets the tick interval in the value axis
	laBarGraphWidget_SetValueAxisTicksPosition	Sets the position of the ticks in the value axis
	laBarGraphWidget_SetValueAxisTicksVisible	Shows/Hides the ticks in the value axis

Structures

	Name	Description
	laBarGraphCategory_t	Contains the Category properties
	laBarGraphDataSeries_t	The data series object that contains the series properties and data
	laBarGraphWidget_t	Implementation of a bar graph widget.

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements bar graph drawing widget functions.

File Name

libaria_widget_bar_graph.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_GetTextLineSpace	Returns the line spacing in pixels for the button text. If < 0, the ascent value of the string's font is used.
	laButtonWidget_GetToggleable	Gets the value of this button's toggle flag
	laButtonWidget_SetVAlignment	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 management 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_SetTextLineSpace	Sets the line space in pixels of the text in the button widget. A value < 0 sets the spacing to the ascent value of the string's font.
	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 and released state.

Types

	Name	Description
	laButtonWidget_PressedEvent	Button pressed event function callback type
	laButtonWidget_ReleasedEvent	Button released event function callback type

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 management 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	Checkbox checked event function callback type
	laCheckBoxWidget_UncheckedEvent	Checkbox unchecked event function callback type

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_GetFilled	Gets the filled state of a circle widget
≡◊	laCircleWidget_GetOrigin	Gets the origin coordinates of a circle widget
≡◊	laCircleWidget_GetRadius	Gets the radius of a circle widget
≡◊	laCircleWidget_GetThickness	Gets the thickness of a circle widget
≡◊	laCircleWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	laCircleWidget_SetFilled	Sets the filled state of a circle widget
≡◊	laCircleWidget_SetOrigin	Sets the origin coordinates of a circle widget
≡◊	laCircleWidget_SetRadius	Sets the radius of a circle widget
≡◊	laCircleWidget_SetThickness	Sets the thickness 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_circular_gauge.h**Enumerations**

	Name	Description
⌚	laCircularGaugeWidgetArcType_t	Type of arc
⌚	laCircularGaugeWidgetDir_t	Direction of the gauge
⌚	laCircularGaugeWidgetLabelPosition_t	Direction of the gauge

Functions

	Name	Description
≡◊	laCircularGaugeWidget_AddAngularArc	Adds an 'angular arc' in the gauge.
≡◊	laCircularGaugeWidget_AddMinorTickLabels	Adds minor tick labels in the gauge.
≡◊	laCircularGaugeWidget_AddMinorTicks	Adds minor ticks in the gauge.
≡◊	laCircularGaugeWidget_AddValueArc	Adds a 'value arc' in the gauge.
≡◊	laCircularGaugeWidget_DeleteArcs	Deletes all arcs in the gauge widget
≡◊	laCircularGaugeWidget_DeleteMinorTickLabels	Deletes all the minor tick labels in the gauge widget
≡◊	laCircularGaugeWidget_DeleteMinorTicks	Deletes all the minor ticks in the gauge widget
≡◊	laCircularGaugeWidget_GetCenterAngle	Returns the center angle of the circular gauge

	laCircularGaugeWidget_GetCenterCircleRadius	Returns radius of the center circle
	laCircularGaugeWidget_GetCenterCircleThickness	Returns thickness of the center circle
	laCircularGaugeWidget_GetCenterCircleVisible	Returns GFX_TRUE if the center circle is visible
	laCircularGaugeWidget_GetDirection	Returns the direction of the gauge.
	laCircularGaugeWidget_GetEndValue	Returns the end value of the gauge
	laCircularGaugeWidget_GetHandRadius	Returns the radius/length of the gauge hand in pixels
	laCircularGaugeWidget_GetHandVisible	Returns GFX_TRUE if the gauge hand is visible
	laCircularGaugeWidget_GetRadius	Gets the radius of a gauge widget
	laCircularGaugeWidget_GetStartAngle	Returns the start angle of the circular gauge
	laCircularGaugeWidget_GetStartValue	Returns the start value of the gauge
	laCircularGaugeWidget_GetTickLabelsVisible	Returns GFX_TRUE if the tick labels are visible
	laCircularGaugeWidget_GetTickLength	Returns the length of the ticks in the gauge in pixels
	laCircularGaugeWidget_GetTicksVisible	Returns GFX_TRUE if the ticks in the gauge are visible
	laCircularGaugeWidget_GetTickCount	Returns the tick increment value in the gauge
	laCircularGaugeWidget_GetValue	Returns the value of the gauge hand
	laCircularGaugeWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	laCircularGaugeWidget_SetCenterAngle	Sets the center angle of the gauge.
	laCircularGaugeWidget_SetCenterCircleRadius	Sets the center radius of the center circle
	laCircularGaugeWidget_SetCenterCircleThickness	Sets the thickness of the center circle
	laCircularGaugeWidget_SetCenterCircleVisible	Sets the center circle visible/invisible
	laCircularGaugeWidget_SetEndValue	Sets the end value of the gauge
	laCircularGaugeWidget_SetHandRadius	Sets the radius/length of the hand
	laCircularGaugeWidget_SetHandVisible	Sets the hand visible/invisible
	laCircularGaugeWidget_SetRadius	Sets the radius of a gauge widget
	laCircularGaugeWidget_SetStartAngle	Sets the start angle of the gauge.
	laCircularGaugeWidget_SetStartValue	Sets the start value of the gauge
	laCircularGaugeWidget_SetStringTable	Sets the string table to be used for the tick labels
	laCircularGaugeWidget_SetTickLabelsVisible	Sets the tick labels visible/invisible
	laCircularGaugeWidget_SetTickLength	Sets the length of the ticks
	laCircularGaugeWidget_SetTicksLabelsStringID	Sets the ID of the string character superset to be used for the tick labels
	laCircularGaugeWidget_SetTicksVisible	Sets the increments/distance between ticks
	laCircularGaugeWidget_SetTickCount	Sets the increments/distance between ticks
	laCircularGaugeWidget_SetValue	Sets the value of the gauge hand
	laCircularGaugeWidget_SetValueChangedEventCallback	Sets the function to be called back when the gauge value changes.

Structures

	Name	Description
	laCircularGaugeArc_t	Internal structure for the arcs in the circular gauge widget
	laCircularGaugeLabel_t	Label object for the circular gauge
	laCircularGaugeTick_t	Tick object for the circular gauge
	laCircularGaugeWidget_t	Implementation of a circular gauge widget.

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements circular gauge drawing widget functions.

File Name

libaria_widget_circular_gauge.h

Company

Microchip Technology Inc.

libaria_widget_circular_slider.h

Enumerations

	Name	Description
⌚	laCircularSliderButtonState_t	State of the slider button
⌚	laCircularSliderWidgetArcType_t	The arcs that compose the circular slider
⌚	laCircularSliderWidgetDir_t	Direction of the slider

Functions

	Name	Description
⌚	laCircularSliderWidget_GetArcRadius	Returns the radius of an arc in the slider widget
⌚	laCircularSliderWidget_GetArcScheme	Returns the scheme of an arc in the slider widget
⌚	laCircularSliderWidget_GetArcThickness	Returns the thickness of an arc in the slider widget
⌚	laCircularSliderWidget_GetArcVisible	Returns true if the specified arc is visible
⌚	laCircularSliderWidget_GetDirection	Returns direction of the slider widget
⌚	laCircularSliderWidget_GetEndValue	Gets the end value of the slider widget
⌚	laCircularSliderWidget_GetOrigin	Gets the origin coordinates of a slider widget
⌚	laCircularSliderWidget_GetRadius	Gets the radius of a slider widget
⌚	laCircularSliderWidget_GetRoundEdges	Returns true if the slider has rounded edges
⌚	laCircularSliderWidget_GetStartAngle	Returns the start angle of a slider widget
⌚	laCircularSliderWidget_GetStartValue	Gets the start value of the slider widget
⌚	laCircularSliderWidget_GetStickyButton	Returns true if the slider button sticks to the start/end value
⌚	laCircularSliderWidget_GetTouchOnButtonOnly	Returns true if the slider slider only responds to touch inside the button area
⌚	laCircularSliderWidget_GetValue	Gets the value of the slider widget
⌚	laCircularSliderWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
⌚	laCircularSliderWidget_SetArcRadius	Sets the start value of the slider widget
⌚	laCircularSliderWidget_SetArcScheme	Sets the scheme of the specified arc
⌚	laCircularSliderWidget_SetArcThickness	Sets the thickness of an arc in the slider widget
⌚	laCircularSliderWidget_SetArcVisible	Shows/Hides the specified arc visible
⌚	laCircularSliderWidget_SetDirection	Sets the direction of the slider widget
⌚	laCircularSliderWidget_SetEndValue	Sets the end value of the slider widget
⌚	laCircularSliderWidget_SetOrigin	Sets the origin coordinates of a slider widget
⌚	laCircularSliderWidget_SetPressedEventCallback	Sets the function that gets called when the slider button is pressed
⌚	laCircularSliderWidget_SetRadius	Sets the radius of a slider widget
⌚	laCircularSliderWidget_SetReleasedEventCallback	Sets the function that gets called when the slider button is released
⌚	laCircularSliderWidget_SetRoundEdges	If round = true, the slider active area edges are round
⌚	laCircularSliderWidget_SetStartAngle	Sets the start angle of a slider widget
⌚	laCircularSliderWidget_SetStartValue	Sets the start value of the slider widget
⌚	laCircularSliderWidget_SetStickyButton	If snap = true, the slider button sticks to the start/end value of the slider
⌚	laCircularSliderWidget_SetTouchOnButtonOnly	If buttonOnly = true, the slider will only respond to touches inside the button area
⌚	laCircularSliderWidget_SetValue	Sets the value of the slider widget

	laCircularSliderWidget_SetValueChangedEventCallback	Sets the function that gets called when the slider value changes
---	---	--

Structures

	Name	Description
	laCircularSliderArc_t	Internal structure for the arcs in the circular slider widget
	laCircularSliderWidget_t	Implementation of a slider widget.

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements slider drawing widget functions.

File Name

`libaria_widget_circular_slider.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	Draw surface draw event function callback type

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 management 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 alignment 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 management 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
≡◊	lalimageWidget_GetHAlignment	Gets the image horizontal alignment value.
≡◊	lalimageWidget_GetImage	Gets the image asset pointer for the widget.
≡◊	lalimageWidget_GetVAlignment	Gets the image vertical alignment value.
≡◊	lalimageWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	lalimageWidget_SetHAlignment	Sets the image horizontal alignment value.
≡◊	lalimageWidget_SetImage	Sets the image asset pointer for the widget.
≡◊	lalimageWidget_SetVAlignment	Sets the image vertical alignment value.

Structures

	Name	Description
◆	lalimageWidget_t	Image widget struct definition
	lalimageWidget	Image widget struct definition

Types

	Name	Description
	lalimageWidget_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_imageplus.h

Enumerations

	Name	Description
≡◊	lalimagePlusWidget_ResizeFilter_t	

Functions

	Name	Description
≡◊	lalimagePlusWidget_GetImage	Gets the image asset pointer for the widget.
≡◊	lalimagePlusWidget_GetInteractive	Returns true if the widget is configured to respond to input events
≡◊	lalimagePlusWidget_GetPreserveAspectEnabled	Returns the boolean value of the 'preserve aspect' property
≡◊	lalimagePlusWidget_GetResizeFilter	Returns the resize filter setting for this image widget
≡◊	lalimagePlusWidget_GetStretchEnabled	Returns the boolean value of the 'stretch to fit' property
≡◊	lalimagePlusWidget_GetTransformHeight	Returns the image scale height coefficient
≡◊	lalimagePlusWidget_GetTransformWidth	Returns the image scale width coefficient
≡◊	lalimagePlusWidget_GetTransformX	Returns the image transform x coefficient

	lalimagePlusWidget_GetTransformY	Returns the image transform y coefficient
	lalimagePlusWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	lalimagePlusWidget_ResetTransform	Resets the image transform values to zero
	lalimagePlusWidget_SetImage	Sets the image asset pointer for the widget.
	lalimagePlusWidget_SetInteractive	Sets the widget interactive flag
	lalimagePlusWidget_SetPreserveAspectEnabled	Sets the boolean value of the 'preserve aspect' property
	lalimagePlusWidget_SetResizeFilter	Sets the resize filter value of the widget
	lalimagePlusWidget_SetStretchEnabled	Sets the boolean value of the stretch property
	lalimagePlusWidget_SetTransformHeight	Sets the image scale height coefficient. This value is in pixels not percentage
	lalimagePlusWidget_SetTransformWidth	Sets the image scale width coefficient. This value is in pixels not percentage
	lalimagePlusWidget_SetTransformX	Sets the image transform x coefficient
	lalimagePlusWidget_SetTransformY	Sets the image transform y coefficient

Structures

	Name	Description
	lalimagePlusWidget_t	Image plus widget struct definition

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
	lalimageSequenceWidget_GetImage	Gets the image asset pointer for an entry.
	lalimageSequenceWidget_GetImageChangedEventCallback	Gets the image changed event callback pointer.
	lalimageSequenceWidget_GetImageCount	Gets the number of image entries for this widget.
	lalimageSequenceWidget_GetImageDelay	Gets the image delay for an entry.
	lalimageSequenceWidget_GetImageHAlignment	Gets the horizontal alignment for an image entry
	lalimageSequenceWidget_GetImageVAlignment	Sets the vertical alignment for an image entry
	lalimageSequenceWidget_GetRepeat	Indicates if the widget will repeat through the image entries.
	lalimageSequenceWidget_IsPlaying	Indicates if the widget is currently cycling through the image entries.
	lalimageSequenceWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	lalimageSequenceWidget_Play	Starts the widget automatically cycling through the image entries.
	lalimageSequenceWidget_Rewind	Resets the current image sequence display index to zero.

	lalimageSequenceWidget_SetImage	Sets the image asset pointer for an entry.
	lalimageSequenceWidget_SetImageChangedEventCallback	Sets the image changed event callback pointer. This callback is called whenever the active display index is changed.
	lalimageSequenceWidget_SetImageCount	Sets the number of image entries for this widget. An image entry that is null will show nothing.
	lalimageSequenceWidget_SetImageDelay	Sets the image delay for an entry.
	lalimageSequenceWidget_SetImageHAlignment	Sets the horizontal alignment for an image entry.
	lalimageSequenceWidget_SetImageVAlignment	Sets the vertical alignment value for an image entry
	lalimageSequenceWidget_SetRepeat	Sets the repeat flag for the widget
	lalimageSequenceWidget>ShowImage	Sets the active display index to the indicated value.
	lalimageSequenceWidget>ShowNextImage	Sets the active display index to the next index value.
	lalimageSequenceWidget>ShowPreviousImage	Sets the active display index to the previous index value.
	lalimageSequenceWidget_Stop	Stops the widget from automatically cycling through the image entries.

Structures

	Name	Description
	lalimageSequenceEntry_t	Image sequence entry definition
	lalimageSequenceWidget_t	Image sequence widget struct definition
	lalimageSequenceEntry	Image sequence entry definition
	lalimageSequenceWidget	Image sequence widget struct definition

Types

	Name	Description
	lalimageSequenceImageChangedEvent_FnPtr	Image changed event function callback type

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
	laKeyPadActionTrigger_t	Defines the trigger for keypad action and events
	laKeyPadCellAction_t	Defines an assigned action to a key pad cell
	laKeyPadActionTrigger	Defines the trigger for keypad action and events
	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_GetKeyPadActionTrigger	Gets the current trigger for keypad edit action and events
	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 management 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_SetKeyPadActionTrigger	Sets the current trigger for keypad edit action and events
	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 and released state.
	laKeyPadWidget_KeyClickEvent	Key click event function callback type

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_GetTextLineSpace	Returns the line spacing in pixels for the label text. If < 0, the ascent value of the string's font is used.
≡◊	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 management 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_SetTextLineSpace	Sets the line space in pixels of the text in the label widget. A value < 0 sets the spacing to the ascent value of the string's font.
≡◊	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 management of this memory until the widget is added to a widget tree.
≡◊	laLineWidget_SetEndPoint	Sets the coordinate for the second point of the line
≡◊	laLineWidget_SetStartPoint	Sets the coordinate 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_line_graph.h

Enumerations

	Name	Description
	<code>laLineGraphDataPointType_t</code>	The graph data point type
	<code>laLineGraphTickPosition_t</code>	The tick position relative to axis
	<code>laLineGraphValueAxis_t</code>	The value axis index value

Functions

	Name	Description
	<code>laLineGraphWidget_AddCategory</code>	Adds a category to the graph
	<code>laLineGraphWidget_AddDataToSeries</code>	Adds a data (value) to the specified series at categoryID index
	<code>laLineGraphWidget_AddSeries</code>	Adds a series to the graph
	<code>laLineGraphWidget_DestroyAll</code>	Destroys data, series and categories and frees the memory allocated
	<code>laLineGraphWidget_GetCategoryAxisLabelsVisible</code>	Returns GFX_TRUE if the category axis labels are visible
	<code>laLineGraphWidget_GetCategoryAxisTicksPosition</code>	Returns the position of the ticks in the category axis
	<code>laLineGraphWidget_GetCategoryAxisTicksVisible</code>	Returns GFX_TRUE if the category axis ticks are visible
	<code>laLineGraphWidget_GetCategoryText</code>	Gets a copy of the string used to label the category
	<code>laLineGraphWidget_GetFillGraphArea</code>	Returns GFX_TRUE if the graph area is filled
	<code>laLineGraphWidget_GetFillSeriesArea</code>	Returns GFX_TRUE if the series area are filled
	<code>laLineGraphWidget_GetGridlinesVisible</code>	Returns GFX_TRUE if the axis gridlines are visible
	<code>laLineGraphWidget_GetMaxValue</code>	Returns the max value of the axis
	<code>laLineGraphWidget_GetMinValue</code>	Returns the min value of the axis
	<code>laLineGraphWidget_GetSeriesFillPoints</code>	Returns GFX_TRUE if the series points are filled
	<code>laLineGraphWidget_GetSeriesLinesVisible</code>	Returns GFX_TRUE if the series lines are visible
	<code>laLineGraphWidget_GetSeriesPointSize</code>	Returns the size of the drawn point
	<code>laLineGraphWidget_GetSeriesPointType</code>	Returns the type of point drawn for the series data points
	<code>laLineGraphWidget_GetSeriesScheme</code>	Returns scheme of the specified series
	<code>laLineGraphWidget_GetStacked</code>	Returns GFX_TRUE if the bars are stacked
	<code>laLineGraphWidget_GetTickLength</code>	Returns the length of the ticks
	<code>laLineGraphWidget_GetValueAxisLabelsVisible</code>	Returns GFX_TRUE if the value axis labels are visible
	<code>laLineGraphWidget_GetValueAxisSubtickInterval</code>	Returns the interval between minor ticks in the value axis
	<code>laLineGraphWidget_GetValueAxisSubticksPosition</code>	Returns the position of the subticks in the axis
	<code>laLineGraphWidget_GetValueAxisSubticksVisible</code>	Returns GFX_TRUE if the value axis subticks are visible
	<code>laLineGraphWidget_GetValueAxisTickInterval</code>	Returns the interval between major ticks in the value axis
	<code>laLineGraphWidget_GetValueAxisTicksPosition</code>	Returns the position of the ticks in the axis
	<code>laLineGraphWidget_GetValueAxisTicksVisible</code>	Returns GFX_TRUE if the value axis ticks are visible

	laLineGraphWidget_New	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	laLineGraphWidget_SetCategoryAxisLabelsVisible	Shows/Hides the category axis labels
	laLineGraphWidget_SetCategoryAxisTicksPosition	Sets the position of the ticks in the category axis
	laLineGraphWidget_SetCategoryAxisTicksVisible	Shows/Hides the category axis ticks
	laLineGraphWidget_SetCategoryText	Sets the string used to label the category
	laLineGraphWidget_SetDataInSeries	Sets the value of the entry in the series index. The entry should have been previously
	laLineGraphWidget_SetFillGraphArea	Sets the graph area filled or not
	laLineGraphWidget_SetFillSeriesArea	Sets the series area filled or not
	laLineGraphWidget_SetGridlinesVisible	Shows/Hides the gridlines
	laLineGraphWidget_SetMaxValue	Sets the max value of the axis
	laLineGraphWidget_SetMinValue	Sets the min value of the axis
	laLineGraphWidget_SetSeriesFillPoints	Sets the series points filled
	laLineGraphWidget_SetSeriesLinesVisible	Shows/hides the lines between series points
	laLineGraphWidget_SetSeriesPointSize	Sets the size of the point drawn for the series data
	laLineGraphWidget_SetSeriesPointType	Sets the type of point drawn for the series data points
	laLineGraphWidget_SetSeriesScheme	Sets the color scheme of the series
	laLineGraphWidget_SetStacked	Stacks the line graph
	laLineGraphWidget_SetStringTable	Sets the string table used for the generated axis labels
	laLineGraphWidget_SetTickLength	Sets the length of the ticks
	laLineGraphWidget_SetTicksLabelsStringID	Sets the ID of the superset string used for the value axis tick labels
	laLineGraphWidget_SetValueAxisLabelsVisible	Shows/Hides the labels in the value axis
	laLineGraphWidget_SetValueAxisSubtickInterval	Sets the minor tick interval in the value axis
	laLineGraphWidget_SetValueAxisSubticksPosition	Sets the position of the subticks in the value axis
	laLineGraphWidget_SetValueAxisSubticksVisible	Shows/Hides the subticks in the value axis
	laLineGraphWidget_SetValueAxisTickInterval	Sets the tick interval in the value axis
	laLineGraphWidget_SetValueAxisTicksPosition	Sets the position of the ticks in the value axis
	laLineGraphWidget_SetValueAxisTicksVisible	Shows/Hides the ticks in the value axis

Structures

	Name	Description
	laLineGraphCategory_t	Contains the Category properties
	laLineGraphDataSeries_t	The data series object that contains the series properties and data
	laLineGraphWidget_t	Implementation of a line graph widget.

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements line graph drawing widget functions.

File Name

libaria_widget_line_graph.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 if 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_GetItemEnable	Returns the enable state of the item in the list widget
	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 management 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_SetItemEnable	Enables or disables an item in the list. A disable item becomes un-selectable.
	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_SelectedItemChangedEvent	Selected item changed event function callback type

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

Enumerations

	Name	Description
	laListWheelIndicatorFill_t	Indicates the fill type for the listwheel indicator area.
	laListWheelZoomEffects_t	Indicates the zoom effects for the list wheel items.
	laListWheelIndicatorFill	Indicates the fill type for the listwheel indicator area.
	laListWheelZoomEffects	Indicates the zoom effects for the list wheel items.

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_GetAutoHideWheel	Returns the list wheel auto hide setting
	laListWheelWidget_GetFlickInitSpeed	Returns the flick init speed for the wheel.
	laListWheelWidget_GetIconMargin	Gets the icon margin value for the list wheel widget
	laListWheelWidget_SetIconPosition	Sets the icon position for the list wheel widget.
	laListWheelWidget_GetIndicatorArea	Returns the spacing for the selected item indicator bars.
	laListWheelWidget_GetIndicatorFill	Gets the indicator area fill type
	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_GetZoomEffects	Gets the list wheel zoom effect
	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 management 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_SetAutoHideWheel	Sets the list wheel to auto hide when not active
	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_SetIndicatorFill	Sets the indicator fill type
	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 selected item 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.
	laListWheelWidget_SetZoomEffects	Sets the list wheel zoom effect

Structures

	Name	Description
	laListWidgetItem_t	Implementation of a list wheel widget item struct
	laListWheelWidget_t	Implementation of a list wheel widget struct
	laListWidgetItem	Implementation of a list wheel widget item struct
	laListWheelWidget	Implementation of a list wheel widget struct

Types

	Name	Description
	laListWheelWidget_SelectedItemChangedEvent	Selected item changed event function callback type

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_pie_chart.h`

Functions

	Name	Description
≡◊	<code>laPieChartWidget_AddEntry</code>	Adds an entry to the pie chart
≡◊	<code>laPieChartWidget_DeleteEntries</code>	Deletes ALL the data in the pie chart
≡◊	<code>laPieChartWidget_GetCenterAngle</code>	Sets the center angle of the chart widget
≡◊	<code>laPieChartWidget_GetEntryOffset</code>	Returns the offset of the entry at the specified index
≡◊	<code>laPieChartWidget_GetEntryRadius</code>	Returns the radius of the entry at the specified index
≡◊	<code>laPieChartWidget_GetEntryScheme</code>	Returns the scheme of the entry at the specified index
≡◊	<code>laPieChartWidget_GetEntryValue</code>	Returns the value of the entry at the specified index
≡◊	<code>laPieChartWidget_GetLabelsOffset</code>	Gets the offsets of the labels from the center
≡◊	<code>laPieChartWidget_GetLabelsVisible</code>	Returns GFX_TRUE if the labels are shown, GFX_FALSE if hidden
≡◊	<code>laPieChartWidget_GetOrigin</code>	Gets the origin coordinates of a chart widget
≡◊	<code>laPieChartWidget_GetStartAngle</code>	Returns the start angle of a chart widget
≡◊	<code>laPieChartWidget_New</code>	Allocates memory for and initializes a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	<code>laPieChartWidget_SetCenterAngle</code>	Sets the center angle of the chart widget
≡◊	<code>laPieChartWidget_SetEntryOffset</code>	Sets the offset of an entry at index
≡◊	<code>laPieChartWidget_SetEntryRadius</code>	Sets the radius of an entry at index
≡◊	<code>laPieChartWidget_SetEntryScheme</code>	Sets the scheme of an entry at index
≡◊	<code>laPieChartWidget_SetEntryValue</code>	Sets the value of an entry at index
≡◊	<code>laPieChartWidget_SetLabelsOffset</code>	Sets the offsets of the labels from the center
≡◊	<code>laPieChartWidget_SetLabelsStringID</code>	Sets the string asset for the labels
≡◊	<code>laPieChartWidget_SetLabelsVisible</code>	Shows/hides the data entry labels
≡◊	<code>laPieChartWidget_SetOrigin</code>	Sets the origin coordinates of a chart widget
≡◊	<code>laPieChartWidget_SetPressedEventCallback</code>	Sets the function called when the chart is pressed/touched
≡◊	<code>laPieChartWidget_SetStartAngle</code>	Sets the start angle of a chart widget
≡◊	<code>laPieChartWidget_SetStringTable</code>	Sets the string table for the labels

Structures

	Name	Description
◆◊	<code>laPieChartPie_t</code>	
◆◊	<code>laPieChartWidget_t</code>	Implementation of a pie chart widget.

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements pie chart drawing widget functions.

File Name

libaria_widget_pie_chart.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_ValueChangedEventCallback	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 management 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_ValueChangedEventCallback	Value changed event function callback type

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_radial_menu.h**Enumerations**

	Name	Description
	laRadialMenuEllipseType_t	This is record laRadialMenuEllipseType_t.

	laRadialMenuItemNode_t	This is record laRadialMenuItemNode_t.
	laRadialMenuWidget_t	

Functions

	Name	Description
	laRadialMenuWidget_AddWidget	Add a widget to the radial menu
	laRadialMenuWidget_ClearItems	Clears all items in the radial menu widget
	laRadialMenuWidget_GetItemProminenceChangedEventCallback	Gets the current radial menu item prominence change event callback
	laRadialMenuWidget_GetItemSelectedEventCallback	Gets the current radial menu item selected event callback
	laRadialMenuWidget_GetProminentIndex	Gets the index of the widget within the radial menu that is prominent
	laRadialMenuWidget_GetTheta	Gets the theta value for the radial menu.
	laRadialMenuWidget_GetWidget	Gets the pointer for the widget by index
	laRadialMenuWidget_IsProminent	Returns true if this radial menu item is currently in the primary selectable position
	laRadialMenuWidget_New	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
	laRadialMenuWidget_RemoveWidget	Removes a widget to the radial menu
	laRadialMenuWidget_SetAlphaScaleMinMax	Sets the minimum and maximum alpha scaling ratio
	laRadialMenuWidget_SetAlphaScaling	Enables per item alpha scaling for the radial menu
	laRadialMenuWidget_SetDrawEllipse	Enables drawing the elliptical track for the radial menu
	laRadialMenuWidget_SetEllipseType	Sets the ellipse type for the radial menu track.
	laRadialMenuWidget_SetHighlightProminent	Sets the item widget to highlight when it is at the prominent location
	laRadialMenuWidget_SetItemProminenceChangedEventCallback	Sets the deselected callback pointer
	laRadialMenuWidget_SetItemSelectedEventCallback	Sets the radial menu item selected event callback
	laRadialMenuWidget_SetNumberOfItemsShown	Sets the number of items visible on the menu
	laRadialMenuWidget_SetProminent	Sets this widget as prominent.
	laRadialMenuWidget_SetProminentIndex	Sets a widget with index within the radial menu as prominent
	laRadialMenuWidget_SetSizeScaleMinMax	Sets the minimum and maximum size scaling ratio in percent
	laRadialMenuWidget_SetSizeScaling	Enables per item size scaling for the radial menu
	laRadialMenuWidget_SetTheta	Sets the theta value for the radial menu.
	laRadialMenuWidget_SetTouchArea	Sets the area that touch input is allowed within the radial menu widget
	laRadialMenuWidget_SetWidgetAt	Insert a widget to the radial menu at the index specified

Structures

	Name	Description
	laRadialMenuItemNode_t	This is record laRadialMenuItemNode_t.
	laRadialMenuWidget_t	Implementation of a radial menu widget struct

Description

Module for Microchip Graphics Library - Aria User Interface Library

This module implements radial menu widget functions.

File Name

libaria_widget_radial_menu.h

Company

Microchip Technology Inc.

libaria_widget_radiobutton.h**Functions**

	Name	Description
≡◊	laRadioButtonWidget_GetCircleButtonSize	Gets the diameter/size of the default circle button
≡◊	laRadioButtonWidget_GetDeselectedEventCallback	Gets the current radio button deselected event callback
≡◊	laRadioButtonWidget_GetGroup	Returns the pointer to the currently set radio button group.
≡◊	laRadioButtonWidget_SetHAlignment	Gets the horizontal alignment setting for a button
≡◊	laRadioButtonWidget_SetImageMargin	Gets the distance between the icon and the text
≡◊	laRadioButtonWidget_SetImagePosition	Gets the current image position setting for the radio button
≡◊	laRadioButtonWidget_SetSelected	Returns true if this radio button is currently selected
≡◊	laRadioButtonWidget_SetSelectedEventCallback	Gets the current radio button selected event callback
≡◊	laRadioButtonWidget_SetSelectedImage	Gets the selected image asset pointer for a button
≡◊	laRadioButtonWidget_SetText	Gets the text value for the button.
≡◊	laRadioButtonWidget_SetUnselectedImage	Gets the image asset pointer currently used as the unselected icon
≡◊	laRadioButtonWidget_SetVAlignment	Sets the vertical alignment for a button
≡◊	laRadioButtonWidget_New	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	laRadioButtonWidget_SetCircleButtonSize	Sets the size of the default circle button
≡◊	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	Radio button deselected function callback type
	laRadioButtonWidget_SelectedEvent	Radio button selected function callback type

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
≡◊	<code>laRectangleWidget_GetThickness</code>	Gets the rectangle border thickness setting
≡◊	<code>laRectangleWidget_New</code>	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	<code>laRectangleWidget_SetThickness</code>	Sets the rectangle border thickness setting

Structures

	Name	Description
◆◊	<code>laRectangleWidget_t</code>	Implementation of a rectangle widget struct
	<code>laRectangleWidget</code>	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
☝◊	<code>laScrollBarOrientation_t</code>	Defines the scroll bar direction values
☝◊	<code>laScrollBarState_t</code>	Defines the various scroll bar state values
	<code>laScrollBarOrientation</code>	Defines the scroll bar direction values
	<code>laScrollBarState</code>	Defines the various scroll bar state values

Functions

	Name	Description
≡◊	<code>laScrollBarWidget_GetExtentValue</code>	Gets the current scroll bar extent value
≡◊	<code>laScrollBarWidget_GetMaximumValue</code>	Gets the maximum scroll value
≡◊	<code>laScrollBarWidget_GetOrientation</code>	Gets the orientation value for the scroll bar
≡◊	<code>laScrollBarWidget_GetScrollPercentage</code>	Gets the current scroll value as a percentage
≡◊	<code>laScrollBarWidget_GetScrollValue</code>	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	Value changed event function callback type

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_GetMaximumValue	Gets the maximum value for the slider
	laSliderWidget_GetMinimumValue	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 management 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	Value changed event function callback type

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_GetFocusChangedEventCallback	Gets the current focus changed event callback pointer
	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 management 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_SetFocusChangedEventCallback	Sets the focus changed event callback pointer
	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	Text changed event function callback type

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 management of this memory until the widget is added to a widget tree.
	laTouchTestWidget_SetPointAddedEventCallback	Sets the point added event callback

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	Point added event function callback type

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
≡◊	laWidget_GetIcon	Gets the currently used window icon
≡◊	laWidget_GetIconMargin	Gets the current image icon margin
≡◊	laWidget_GetTitle	Gets the title text for this window.
≡◊	laWidget_New	Allocates memory for a new widget of this type. The application is responsible for the management of this memory until the widget is added to a widget tree.
≡◊	laWidget_SetIcon	Sets the image to be used as a window icon
≡◊	laWidget_SetIconMargin	Sets the image icon margin
≡◊	laWidget_SetTitle	Sets the title text for the window.

Structures

	Name	Description
◆	laWidget_t	Implementation of a window widget struct
	laWidget	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

mismanagment. 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.

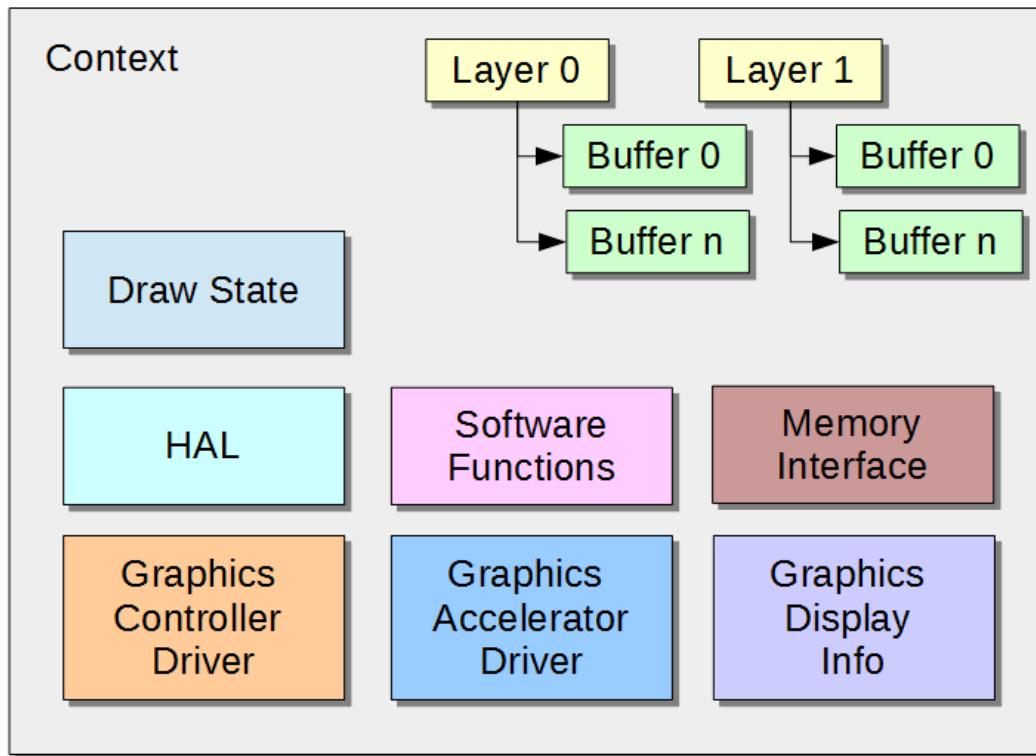
HAL Context

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 Freemaker 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



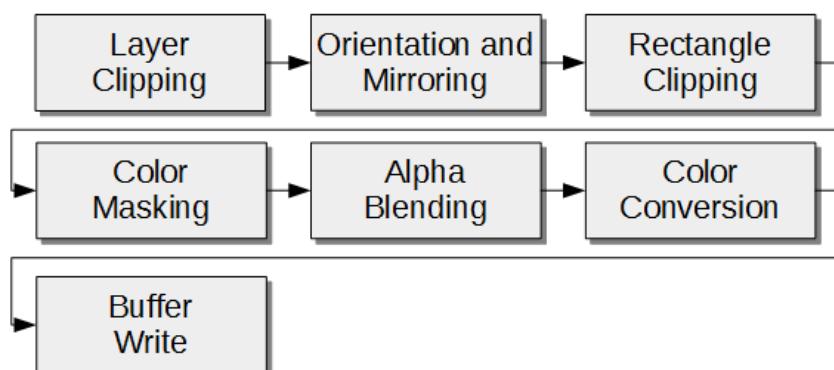
Notes:

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:**
- 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.
 - 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 the 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.



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_ActiveContext	Gets the current set active HAL context.
≡◊	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_ColorModelInfoGet	
≡◊	GFX_ColorValue	Used for getting a color value by name.
≡◊	GFX_ContextActiveSet	Sets the active context
≡◊	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.
≡◊	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.
	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 rectangular section of pixels from a pixel buffer.
	GFX_PixelBufferAreaGet_Unsafe	Extracts a rectangular 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_RectClipAdj	Returns the rectangle clipped using r_rect, and also adjusts the third rectangle
	GFX_RectCombine	Combines the area of two rectangles into a single rectangle.
	GFX_RectCompare	Returns 1 if the two rectangles have the same position and dimensions
	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	Returns a GFX_Rect structure based on 2 points

	GFX_RectIntersects	Determines if two rectangles are intersecting
	GFX_RectsAreSimilar	Returns GFX_TRUE if the two rectangles are adjacent and vertically or horizontally aligned
	GFX_RectSplit	Splits two overlapping rectangles into several (up to 4) non-overlapping rectangles
	GFX_RectToPoints	Returns the points for the upper left and lower right vertices of a rectangle

b) Data Types and Constants

	Name	Description
	GFX_ANTIALIAS_MODE_COUNT	This is macro GFX_ANTIALIAS_MODE_COUNT.
	GFX_AntialiasMode	Enables anti-aliased drawing hint
	GFX_AntialiasMode_t	Enables anti-aliased drawing hint
	GFX_ASSERT	This is macro GFX_ASSERT.
	GFX_BitsPerPixel	List of available bits-per-pixel sizes.
	GFX_BitsPerPixel_t	List of available bits-per-pixel sizes.
	GFX_BlendMode	Blend mode masks
	GFX_BlendMode_t	Blend mode masks
	GFX_BufferSelection	Buffer selector used when querying layers for certain buffer states.
	GFX_BufferSelection_t	Buffer selector used when querying layers for certain buffer states.
	GFX_BufferState	Frame buffer memory states
	GFX_BufferState_t	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_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_ColorInfo	This is variable GFX_ColorInfo.
	GFX_ColorMask	Maskable list of color values.
	GFX_ColorMask_t	Maskable list of color values.
	GFX_ColorMode	List of available color modes.
	GFX_ColorMode_t	List of available color modes.
	GFX_ColorModelInfo	Struct that provides information about a color mode.
	GFX_ColorModelInfo_t	Struct that provides information about a color mode.
	GFX_ColorName	Color name reference table
	GFX_ColorName_t	Color name reference table
	GFX_Context	An instance of the hardware abstraction layer.
	GFX_Context_t	An instance of the hardware abstraction layer.
	GFX_DEPRECATED	This is macro GFX_DEPRECATED.
	GFX_DisplayInfo	Describes a graphical display device.
	GFX_DisplayInfo_t	Describes a graphical display device.
	GFX_DRAW_MODE_COUNT	This is macro GFX_DRAW_MODE_COUNT.
	GFX_DrawMode	Gradient draw modes.
	GFX_DrawMode_t	Gradient draw modes.
	GFX_DrawPipeline	
	GFX_DrawState	A list of drawing hints for shape drawing algorithms
	GFX_DrawState_t	A list of drawing hints for shape drawing algorithms
	GFX_DriverInfo	A driver description structure.
	GFX_DriverInfo_t	A driver description structure.

	GFX_Flag	Hardware abstraction state interface flags. See gfx.h for a comprehensive description.
	GFX_Flag_t	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_FrameBuffer_t	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_Layer	Layers describe basic display drawing areas and are meant to map directly to graphics controller hardware layers.
	GFX_Layer_t	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_MemoryIntf_t	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_NUM_FLAGS	This is macro GFX_NUM_FLAGS.
	GFX_Orientation	Orthogonal orientation settings.
	GFX_Orientation_t	Orthogonal orientation settings.
	GFX_PIPELINE_MODE_COUNT	This is macro GFX_PIPELINE_MODE_COUNT.
	GFX_PipelineMode	Hardware draw path settings.
	GFX_PipelineMode_t	Hardware draw path 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 length in bytes.
	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 length in bytes.
	GFX_Point_t	A two dimensional Cartesian point.
	GFX_Realloc_FnPtr	Simple wrapper around the standard realloc function pointer. Used for redirecting memory allocation to other memory management systems.
	GFX_Rect_t	A rectangle definition.
	GFX_ResizeMode	The algorithm used for resizing a frame
	GFX_ResizeMode_t	The algorithm used for resizing a frame
	GFX_Size	A two dimensional indication of size. Values are signed but should never be negative.
	GFX_Size_t	A two dimensional indication of size. Values are signed but should never be negative.

Description

This section describes the Aria User Interface Library Hardware Abstraction Layer interface.

a) Functions

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_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_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_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_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_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 width 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 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_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 width 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 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_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 width 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
	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* displayRect	the rectangle of the display
GFX_Layer* 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_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_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_LayerRectToOrientedSpace(const GFX_Rect* layerRect,
                                       const GFX_Rect* rect,
                                       GFX_Orientation ori,
                                       GFX_Bool mirrored)
```

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_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* displayRect	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_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_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 rectangular 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 rectangular 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* const 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 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_RectClipAdj Function

Returns the rectangle clipped using l_rect, and also adjusts the third rectangle

File

[gfx_rect.h](#)

C

```
LIB_EXPORT GFX_Rect GFX_RectClipAdj(const GFX_Rect* l_rect, const GFX_Rect* r_rect, GFX_Rect* adj);
```

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* adj	the adjust rectangle

Function

```
GFX_Rect GFX_RectClipAdj(const GFX_Rect* l_rect,
const           GFX_Rect* r_rect,
           GFX_Rect* adj)
```

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_RectCompare Function

Returns 1 if the two rectangles have the same position and dimensions

File

[gfx_rect.h](#)

C

```
LIB_EXPORT int32_t GFX_RectCompare(const GFX_Rect* l, const GFX_Rect* r);
```

Returns

int32_t

Function

```
int32_t GFX_RectCompare(const GFX_Rect* l,  
                      const GFX_Rect* r)
```

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 completely 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_ContainsRect(const [GFX_Rect](#)* l_rect, const [GFX_Rect](#)* r_rect)

[GFX_RectFromPoints Function](#)

Returns a [GFX_Rect](#) structure based on 2 points

File

[gfx_rect.h](#)

C

```
LIB_EXPORT GFX_Rect GFX\_RectFromPoints(const GFX_Point* p1, const GFX_Point* p2);
```

Returns

[GFX_Rect](#)

Parameters

Parameters	Description
const GFX_Point* p1	the first point
const GFX_Point* p2	the second point

Function

[GFX_Rect](#) [GFX_RectFromPoints](#)(const [GFX_Point](#)* p1,
const [GFX_Point](#)* p2)

[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_RectsAreSimilar Function

Returns GFX_TRUE if the two rectangles are adjacent or vertically or horizontally aligned

File

[gfx_rect.h](#)

C

```
LIB_EXPORT GFX_Boolean GFX_RectsAreSimilar(const GFX_Rect* l, const GFX_Rect* r);
```

Returns

GFX_Boolean

Parameters

Parameters	Description
const GFX_Rect* l	the first rectangle
const GFX_Rect* r	the second rectangle

Function

```
GFX_Boolean GFX_RectsAreSimilar(const GFX_Rect* l,  
const GFX_Rect* r)
```

GFX_RectSplit Function

Splits two overlapping rectangles into several (up to 4) non-overlapping rectangles

File

[gfx_rect.h](#)

C

```
LIB_EXPORT uint32_t GFX_RectSplit(const GFX_Rect* sub, const GFX_Rect* obj, GFX_Rect res[4]);
```

Returns

uint32_t - the number of non-overlapping rectangles returned

Parameters

Parameters	Description
const GFX_Rect* sub	the first rectangle
const GFX_Rect* obj	the second rectangle
GFX_Rect res[4]	the output rectangles

Function

```
uint32_t GFX_RectSplit(const GFX_Rect* sub,  
const GFX_Rect* obj,  
GFX_Rect res[4])
```

GFX_RectToPoints Function

Returns the points for the upper left and lower right vertices of a rectangle

File

[gfx_rect.h](#)

C

```
LIB_EXPORT void GFX_RectToPoints(const GFX_Rect* rect, GFX_Point* p1, GFX_Point* p2);
```

Returns

[GFX_Rect](#)

Parameters

Parameters	Description
const GFX_Rect* rect	the rectangle
GFX_Point* p1	the point of upper left vertex
GFX_Point* p2	the point of the lower right vertex

Function

```
void GFX_RectToPoints(const GFX_Rect* rect,
                      GFX_Point* p1,
                      GFX_Point* p2)
```

b) Data Types and Constants***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_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_ASSERT Macro

File

[gfx_common.h](#)

C

```
#define GFX_ASSERT(x) { }
```

Description

This is macro GFX_ASSERT.

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_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 malloc 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_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_ColorInfo Variable

File

[gfx_color.h](#)

C

```
LIB_EXPORT GFX_ColorModeInfo GFX_ColorInfo[GFX_COLOR_MODE_COUNT];
```

Description

This is variable GFX_ColorInfo.

GFX_ColorMask Enumeration

Maskable list of color values.

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_RGB_888 | GFX_COLOR_MASK_RGBA_8888 | GFX_COLOR_MASK_ARGB_8888 | GFX_COLOR_MASK_YUV
} GFX_ColorMask;
```

Description

Enumeration: GFX_ColorMask_t

GFX_ColorMode Enumeration

List of available color modes.

File

[gfx_color.h](#)

C

```
typedef 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
} 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
 shift - the shifts 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

Structure: 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_DEPRECATED Macro

File

[gfx_common.h](#)

C

```
#define GFX_DEPRECATED __attribute__ ((deprecated))
```

Description

This is macro GFX_DEPRECATED.

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_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_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_DrawPipeline Type

File

[gfx_draw.h](#)

C

```
typedef struct GFX_DrawPipeline_t GFX_DrawPipeline;
```

Section

Data Types and Constants

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_DriverInfo Structure

A driver description structure.

File

[gfx_driver_interface.h](#)

C

```
typedef struct GFX_DriverInfo_t {
    char name[16];
    GFX_ColorMask color_formats;
    uint32_t layer_count;
} 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_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 function 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_NUM_FLAGS Macro

File

[gfx_common.h](#)

C

```
#define GFX_NUM_FLAGS GFXF_LAST_FLAG
```

Description

This is macro GFX_NUM_FLAGS.

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_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_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_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.

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_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_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

GFX_ResizeMode Enumeration

The algorithm used for resizing a frame

File

[gfx_draw.h](#)

C

```
typedef enum GFX_ResizeMode_t {
    GFX_RESIZE_NEARESTNEIGHBOR = 0x0,
    GFX_RESIZE_BILINEAR
} GFX_ResizeMode;
```

Description

Enumeration: GFX_ResizeMode_t

Remarks

None.

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

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_arc.h	This is file gfx_draw_arc.h.
gfx_draw.blit.h	This is file gfx_draw.blit.h.
gfx_draw_circle.h	This is file gfx_draw_circle.h.
gfx_draw_ellipse.h	This is file gfx_draw_ellipse.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_draw_stretchblit.h	This is file gfx_draw_stretchblit.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_pixel_buffer.h	Defines a general purpose pixel buffer construct.
gfx_processor_interface.h	Defines MPLAB Harmony Graphics Hardware Abstraction Layer graphics processor interface struct
gfx_rect.h	Defines general purposes rectangle functions.

Description

gfx_common.h

Common definitions for MPLAB Harmony Graphics Hardware Abstraction Layer

Enumerations

	Name	Description
	GFX_BBlendMode_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_BBlendMode	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_NUM_FLAGS	This is macro GFX_NUM_FLAGS.
	GFX_PIPELINE_MODE_COUNT	This is macro GFX_PIPELINE_MODE_COUNT.

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_Calloc_FnPtr	Simple wrapper around the standard calloc function pointer. Used for redirecting memory allocation to other memory management systems.
	GFX_Free_FnPtr	Simple wrapper around the standard free function pointer. Used for redirecting memory free to other memory management systems.
	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_Realloc_FnPtr	Simple wrapper around the standard realloc function pointer. Used for redirecting memory allocation to other memory management systems.

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 values.
	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 values.
	GFX_ColorMode	List of available color modes.
	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_ColorModelInfoGet	
	GFX_ColorValue	Used for getting a color value by name.

Macros

	Name	Description
	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.

Structures

	Name	Description
	GFX_ColorModelInfo_t	Struct that provides information about a color mode.
	GFX_ColorModelInfo	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	The algorithm used for resizing a frame
	GFX_AntialiasMode	Enables anti-aliased drawing hint
	GFX_DrawMode	Gradient draw modes.
	GFX_ResizeMode	The algorithm used for resizing a frame

Functions

	Name	Description
	GFX_DrawArc	Draws a filled arc from using the specified dimensions and the current draw state.
	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_DrawEllipse	Draws a filled ellipse 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_DrawPixelByDrawState	Sets the pixel at X and Y using a specified 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.

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

Types

	Name	Description
	GFX_DrawPipeline	

Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Shape drawing functions.

File Name

gfx_draw.h

Company

Microchip Technology Inc.

gfx_draw_arc.h

This is file gfx_draw_arc.h.

gfx_draw.blit.h

This is file gfx_draw.blit.h.

gfx_draw_circle.h

This is file gfx_draw_circle.h.

gfx_draw_ellipse.h

This is file gfx_draw_ellipse.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_draw_stretchblit.h

This is file gfx_draw_stretchblit.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.
	GFX_DriverInfo	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

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_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 rectangular section of pixels from a pixel buffer.
≡◊	GFX_PixelBufferAreaGet_Unsafe	Extracts a rectangular 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 length 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 length 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_processor_interface.h

Defines MPLAB Harmony Graphics Hardware Abstraction Layer graphics processor interface struct

Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Graphics processor information, internal use.

File Name

gfx_processor_interface.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	Returns the rectangle clipped using r_rect, and also adjusts the third rectangle
	GFX_RectCombine	Combines the area of two rectangles into a single rectangle.
	GFX_RectCompare	Returns 1 if the two rectangles have the same position and dimensions
	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	Returns a GFX_Rect structure based on 2 points
	GFX_RectIntersects	Determines if two rectangles are intersecting
	GFX_RectsAreSimilar	Returns GFX_TRUE if the two rectangles are adjacent and vertically or horizontally aligned
	GFX_RectSplit	Splits two overlapping rectangles into several (up to 4) non-overlapping rectangles
	GFX_RectToPoints	Returns the points for the upper left and lower right vertices of a rectangle

Description

Module for Microchip Graphics Library - Hardware Abstraction Layer
Rectangle management functions.

File Name

gfx_rect.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

This 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_il9488.h`

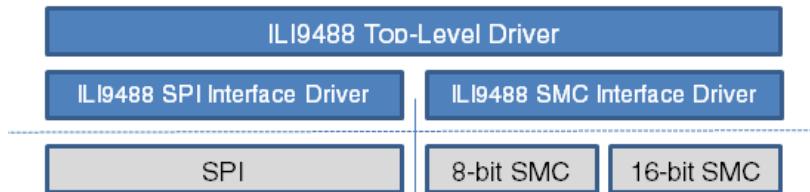
The top-level interface to the ILI9488 Display Controller Driver Library is defined in the `drv_gfx_il9488.h` header file. Any C language source (.c), particularly in the graphics framework, file that uses the ILI9488 Display Controller Driver Library should include `drv_gfx_il9488.h`.

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_Ifnt_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_Ifnt_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, therefore 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.

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_Boolean 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
	GFX_AbsoluteValue	Calculates the absolute value of a signed integer.

GFX_Atan	Performs the inverse tangent operation
GFX_Clampf	Clamps a float between a min and max
GFX_Clampi	Clamps an integer between a min and max
GFX_DivideRounding	Calculates integer division of num/denom with rounding based on LSB/2.
GFX_DrawArc	Draws a filled arc from using the specified dimensions and the current draw state.
GFX_DrawEllipse	Draws a filled ellipse from using the specified dimensions and the current draw state.
GFX_DrawPixelByDrawState	Sets the pixel at X and Y using a specified draw state.
GFX_EllipsePoint	Performs a cosine lookup to determine the points in an arc at specified radius and angle (polar > cartesian conversion)
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_Normalize360	normalize an angle between 0 - 360
GFX_Percent	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.
GFX_PercentOf	Calculates the percentage of a number. Returns a whole number with no decimal component.
GFX_PercentOfDec	Calculates the percentage of a number. Returns a whole number and the tenths digit decimal component.
GFX_PercentWholeRounded	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.
GFX_PolarToXY	Performs a cosine lookup to determine the points in an arc at specified radius and angle (polar to Cartesian conversion)
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.
GFX_SineCosineGet	Performs a cosine lookup to determine the result of sine/cosine
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_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_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_GetStringAscent	Gets the ascent of a string contained in the string table.
	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.
	GFXU_PaletteGetColor	Gets a color from a palette asset given an index value.
	GFXU_PreprocessImage	Preprocesses an image to a specified memory address.

b) Data Types and Constants

	Name	Description
	GFX_TRIG_FUNCTION_TYPE	This is type GFX_TRIG_FUNCTION_TYPE.
	GFXU_ASSET_LOCATION_INTERNAL	This is macro GFXU_ASSET_LOCATION_INTERNAL.
	GFXU_AssetHeader	Defines a common header for all assets supported by the generic decoder interface.
	GFXU_AssetHeader_t	Defines a common header for all assets supported by the generic decoder interface.
	GFXU_AssetType	Enumerates known asset types.
	GFXU_AssetType_t	Enumerates known asset types.
	GFXU_BinaryAsset	A binary asset type. Generic data that can be of any type
	GFXU_BinaryAsset_t	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_ExternalAssetReader_t	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. The argument is the state machine to process.
	GFXU_ExternalAssetReaderStatus	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
	GFXU_ExternalAssetReaderStatus_t	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

	GFXU_FontAsset	Describes a font asset. A font asset is a series of raster images that represent 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 - ... more
	GFXU_FontAsset_t	Describes a font asset. A font asset is a series of raster images that represent 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 - ... more
	GFXU_FontAssetBPP	Indicates the bits per pixel mode of a font
	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 count - number of ranges in the index table ranges - the glyph range array
	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 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 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_FontGlyphRange_t	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
	GFXU_ImageAsset	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... more
	GFXU_ImageAsset_t	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... more
	GFXU_ImageCompressionType	Indicates the image compression type, only applies to RAW types
	GFXU_ImageCompressionType_t	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

	GFXU_ImageFormat_t	Indicates the image encoding format
	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. 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. The argument is the asset that needs to be read. 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. The argument is the reader that requested the memory.
	GFXU_MemoryIntf	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.
	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 . 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. header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image
	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. header - standard asset header colorCount - the number of colors contained in the palette colorMode - the color mode of the image
	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_StringEncodingMode	Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16
	GFXU_StringEncodingMode_t	Indicates the string encoding mode type. Any characters above 255 must use UTF8 or UTF16
	GFXU_StringTableAsset	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... more

	GFXU_StringTableAsset_t	<p>Describes a string table asset. There is typically only ever one of these defined at any one time.</p> <p>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</p>
---	-------------------------	---

Description

This section describes the interface for the Graphics Utilities Library.

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_Atan Function

Performs the inverse tangent operation

File

[gfx_math.h](#)

C

```
LIB_EXPORT double GFX_Atan(double val);
```

Returns

double - the angle in radians

Parameters

Parameters	Description
val	floating point value

Function

```
double GFX_Atan(int32_t x, int32_t y);
```

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_DivideRounding Function

Calculates integer division of num/denom with rounding based on LSB/2.

File[gfx_math.h](#)**C**

```
LIB_EXPORT int32_t GFX_DivideRounding(int32_t num, int32_t denom);
```

Returns

int32_t - result, equivalent to int32_t(((float)num)/((float)denom) + 0.5) without using floating point

Parameters

Parameters	Description
num	numerator
denom	denominator

Function

```
int32_t GFX_DivideRounding(int32_t num, int32_t denom);
```

GFX_DrawArc Function

Draws a filled arc from using the specified dimensions and the current draw state.

File[gfx_draw.h](#)**C**

```
LIB_EXPORT GFX_Result GFX_DrawArc(int32_t x, int32_t y, int32_t radius, int32_t startAngle, int32_t centerAngle);
```

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
startAngle	the starting angle, in degrees, of the arc
centerAngle	the center angle, in degrees, of the arc. A positive value draws CW while a negative value draws CCW.

Function

```
GFX_Result GFX_DrawArc(int32_t x, int32_t y, int32_t radius, int32_t startAngle, int32_t centerAngle);
```

GFX_DrawEllipse Function

Draws a filled ellipse from using the specified dimensions and the current draw state.

File[gfx_draw.h](#)

C

```
LIB_EXPORT GFX_Result GFX_DrawEllipse(int32_t x, int32_t y, int32_t a, int32_t b, int32_t theta,
                                      int32_t startAngle, int32_t centerAngle, int32_t width, int32_t height);
```

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
a	the length of the short side of the ellipse
b	the length of the long side of the ellipse
theta	the tilt angle, in degrees, of the ellipse
startAngle	the starting angle, in degrees, of the arc
centerAngle	the center angle, in degrees, of the arc. A positive value draws CW while a negative value draws CCW.
width	width, in pixels, of the rectangular draw area for the ellipse
height	height, in pixels, of the rectangular draw area for the ellipse

Function

```
GFX_Result GFX_DrawEllipse(int32_t x,
                           int32_t y,
                           int32_t a,
                           int32_t b,
                           int32_t theta,
                           int32_t startAngle,
                           int32_t centerAngle,
                           int32_t width,
                           int32_t height);
```

GFX_DrawPixelByDrawState Function

Sets the pixel at X and Y using a specified draw state.

File

[gfx_draw.h](#)

C

```
LIB_EXPORT GFX_Result GFX_DrawPixelByDrawState(int32_t x, int32_t y, GFX_DrawState state);
```

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
state	the draw state

Function

```
GFX_Result GFX_DrawPixelByDrawState(int32_t x,
```

```
int32_t y,
GFX_DrawState state);
```

GFX_EllipsePoint Function

Performs a cosine lookup to determine the points in an arc at specified radius and angle (polar > cartesian conversion)

File

[gfx_math.h](#)

C

```
LIB_EXPORT GFX_Result GFX_EllipsePoint(int32_t t, int32_t a, int32_t b, int32_t theta,
GFX_Point* p);
```

Returns

p - the output point in cartesian plane

Parameters

Parameters	Description
t	angle of the point on the ellipse (in degrees)
a	the half-length of 0-180 axis of the ellipse
b	the half-length of 90-270 axis of the ellipse
theta	angle of the ellipse (in degrees)

Function

```
GFX_Result GFX_EllipsePoint(int32_t t, int32_t a, int32_t b, int32_t theta, GFX_Point* p);
```

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_Min(float l, float r);
```

GFX_Min Function

Returns the smaller of two integers.

File

[gfx_math.h](#)

C

```
LIB_EXPORT int32_t GFX_Min(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_Min(int32_t l, int32_t r);
```

GFX_Normalize360 Function

normalize an angle between 0 - 360

File

[gfx_math.h](#)

C

```
LIB_EXPORT int16_t GFX_Normalize360(int16_t t);
```

Returns

int16_t - normalize an angle in degrees.

Example

t = -5, return value is 355 t = 450, return value is 90

Parameters

Parameters	Description
int16_t t	angle (in degrees)

Function

```
int16_t GFX_Normalize360(int16_t t);
```

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_PercentOfDec Function

Calculates the percentage of a number. Returns a whole number and the tenths digit decimal component.

File

[gfx_math.h](#)

C

```
LIB_EXPORT void GFX_PercentOfDec(uint32_t num, uint32_t percent, uint32_t* whl, uint32_t* dec);
```

Returns

none

Parameters

Parameters	Description
num	the number to consider
percent	the percentage to apply
whl	pointer to whole number value
dec	pointer to tenths digit as an integer

Function

```
void GFX_PercentOfDec(uint32_t num, uint32_t percent, uint32_t* whl, uint32_t* dec);
```

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_PolarToXY Function

Performs a cosine lookup to determine the points in an arc at specified radius and angle (polar to Cartesian conversion)

File

[gfx_math.h](#)

C

```
LIB_EXPORT GFX_Result GFX_PolarToXY(int32_t r, int32_t a, GFX_Point* p);
```

Returns

p - the output point in Cartesian plane

Parameters

Parameters	Description
r	radius
a	angle (in degrees)

Function

```
GFX_Result GFX_PolarToXY(int32_t r, int32_t a,     GFX_Point* p);
```

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);
```

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);
```

GFX_SineCosineGet Function

Performs a cosine lookup to determine the result of sine/cosine

File[gfx_math.h](#)**C**

```
LIB_EXPORT int16_t GFX_SineCosineGet(int16_t v, GFX_TRIG_FUNCTION_TYPE type);
```

Returns

int16_t - result of sine cosine fixed point value (times 256), calling function needs to divide by 256 to get good result
ex. "a * cos(v)" would be "a * GFX_SineCosineGet(v, GFX_TRIG_COSINE_TYPE) / 256";

Parameters

Parameters	Description
v	angle (in degrees)
type	function type sine or cosine

Function

```
int16_t GFX_SineCosineGet(int16_t v, GFX_TRIG_FUNCTION_TYPE type)
```

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, terminated 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, terminated 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_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_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_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_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_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_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_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_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 character

Function

```
GFXU_CHAR GFXU_GetCharAt(GFXU\_StringTableAsset* tbl,  
uint32_t id,  
uint32_t lang,  
uint32_t idx)
```

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, terminated with a zero value
GFXU_FontAsset* font	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_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_GetStringAscent Function

Gets the ascent of a string contained in the string table.

File

[gfxu_string.h](#)

C

```
LIB_EXPORT uint32_t GFXU_GetStringAscent(GFXU_StringTableAsset* tbl, uint32_t id, uint32_t lang);
```

Returns

uint32_t - the ascent 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_GetStringAscent( GFXU_StringTableAsset* tbl,
                               uint32_t id,
                               uint32_t lang)
```

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_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)
```

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_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 streaming 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);
```

b) Data Types and Constants**[GFX_TRIG_FUNCTION_TYPE Enumeration](#)****File**

[gfx_math.h](#)

C

```
typedef enum {
    GFX_TRIG_SINE_TYPE,
    GFX_TRIG_COSINE_TYPE
} GFX_TRIG_FUNCTION_TYPE;
```

Description

This is type GFX_TRIG_FUNCTION_TYPE.

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_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* mem_intf;
    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
 mem_intf - 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 represent 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_FontAsset_t Structure

Describes a font asset. A font asset is a series of raster images that represent 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_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 if 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 this 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_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

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_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

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;
```

Enumerations

	Name	Description
	GFXU_ImageFlags_t	A list of flags describing an image asset

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_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_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_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

Files

Files

Name	Description
gfx_math.h	Contains some general purpose math functions
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

gfx_math.h

Contains some general purpose math functions

Enumerations

	Name	Description
	GFX_TRIG_FUNCTION_TYPE	This is type GFX_TRIG_FUNCTION_TYPE.

Functions

	Name	Description
≡◊	GFX_AbsoluteValue	Calculates the absolute value of a signed integer.
≡◊	GFX_Atan	Performs the inverse tangent operation
≡◊	GFX_Clampf	Clamps a float between a min and max
≡◊	GFX_Clampi	Clamps an integer between a min and max
≡◊	GFX_DivideRounding	Calculates integer division of num/denom with rounding based on LSB/2.
≡◊	GFX_EllipsePoint	Performs a cosine lookup to determine the points in an arc at specified radius and angle (polar > cartesian conversion)
≡◊	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_Normalize360	normalize an angle between 0 - 360
≡◊	GFX_Percent	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.
≡◊	GFX_PercentOf	Calculates the percentage of a number. Returns a whole number with no decimal component.
≡◊	GFX_PercentOfDec	Calculates the percentage of a number. Returns a whole number and the tenths digit decimal component.
≡◊	GFX_PercentWholeRounded	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.
≡◊	GFX_PolarToXY	Performs a cosine lookup to determine the points in an arc at specified radius and angle (polar to Cartesian conversion)
≡◊	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.
≡◊	GFX_SineCosineGet	Performs a cosine lookup to determine the result of sine/cosine

Description

Module for Microchip Graphics Library - Hardware Abstraction Layer

Math support functions.

File Name

`gfx_layer.h`

Company

Microchip Technology Inc.

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 represent 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 -... 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 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 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 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 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. 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. 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. 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. 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. 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. 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. The argument is the asset that needs to be read. 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. 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. 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 if 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. 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

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. 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. 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.
	GFXUGetStringAscent	Gets the ascent of a string contained in the string table.
	GFXUGetStringHeight	Gets the height of a string contained in the string table.
	GFXUGetStringLength	Gets the length of a string contained in a string table.
	GFXUGetStringLineRect	Gets the bounding rectangle for a line in a string asset.
	GFXUGetStringRect	Gets the bounding rectangle for a string contained in the string table.
	GFXUGetStringSizeInBytes	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. 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. 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 represent 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 -... 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.

Description

Module for Microchip Graphics Library - Graphics Utilities Library

Internal use only

File Name

gfx_string_utils.h

Company

Microchip Technology Inc.

Support

This section provides support information for MPLAB Harmony.

Using the Help

This topic contains general information that is useful to know to maximize using the MPLAB Harmony help.

Description

Help Formats

MPLAB Harmony Help is provided in three formats:

- Stand-alone HyperText Markup Language (HTML)
- Microsoft Compiled HTML Help (CHM)
- Adobe® Portable Document Format (PDF)



TIP! When using the MPLAB Harmony Help PDF, be sure to open the "bookmarks" if they are not already visible to assist in document navigation. See Using the Help for additional information.

Help File Locations

Each of these help files are included in the installation of MPLAB Harmony in the following locations:

- HTML - <install-dir>/doc/html/index.html
- CHM - <install-dir>/doc/help_harmony.chm
- PDF - <install-dir>/doc/help_harmony.pdf

Refer to [Help Features](#) for more information on using each output format.

Where to Begin With the Help

The help documentation provides a comprehensive source of information on how to use and understand MPLAB Harmony. However, it is not required to read the entire document before starting to work with MPLAB Harmony.

Prior to using MPLAB Harmony, it is recommended to review the Release Notes for any known issues. A PDF copy of the release notes is provided in the <install-dir>/doc folder of your installation.

New Users

For new users to MPLAB Harmony, it is best to follow the Guided Tour provided in *Volume I: Getting Started With MPLAB Harmony*.

Experienced Users

For experienced users already somewhat familiar with the MPLAB Harmony installation and online resources and, looking to jump right into a specific topic, follow the links provided in the table in *Volume 1: Getting Started With MPLAB Harmony Libraries and Applications > Guided Tour*.

Trademarks

Provides information on trademarks used in this documentation.

Description

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo,

Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, All Rights Reserved.

Typographic Conventions

This topic describes the typographic conventions used in the MPLAB Harmony Help.

Description

The MPLAB Harmony Help uses the following typographic conventions:

Convention	Represents	Example
 TIP!	Provides helpful information to assist the user.	 TIP! Throughout this documentation, occurrences of <install-dir> refer to the default MPLAB Harmony installation path, which is: C:/microchip/harmony/<version>.
 Note:	Provides useful information to the user.	 Note: Refer to the individual Release Notes for the libraries and demonstrations for details on changes from the previous release of MPLAB Harmony.
 Important!	Provides important information to the user.	 Important! This tutorial is based on the PIC32MZ Embedded Connectivity (EC) Starter Kit . If you are using a different hardware platform, you may need to change some of the settings used in the tutorial (such as timer selection and clock rates) to appropriate values for your platform.
 Warning	Warns the user of a potentially harmful issue.	 Warning The cause of the interrupt must be removed before clearing the interrupt source or the interrupt may reoccur immediately after the source is cleared potentially causing an infinite loop. An infinite loop may also occur if the source is not cleared before the interrupt-handler returns.
 MHC Followed by Green italicized text	Indicates a process step that is automated by the MPLAB Harmony Configurator (MHC).	 MHC <i>Throughout the documentation, when you see this icon, it indicates that the MHC automates the associated task(s).</i>
Italic Characters	Referenced documentation and emphasized text.	<ul style="list-style-type: none"> <i>MPLAB X IDE User's Guide</i> ...is the <i>only</i> option.
Initial Capitalization	<ul style="list-style-type: none"> A window A dialog A menu selection 	<ul style="list-style-type: none"> the Output window the SaveAs dialog the Enable Programmer menu
Quotation Marks	A field name in a window or dialog.	"Save project before build"
Italic text with right angle bracket	A menu path.	<i>File > Save</i>
Bold Characters	<ul style="list-style-type: none"> Topic headings A dialog button or user action, such as clicking an icon or selecting an option 	<ul style="list-style-type: none"> Prerequisites Click OK

Courier New text enclosed in angle brackets	A key on the keyboard.	Press <Ctrl><V>.
Courier New text	<ul style="list-style-type: none"> • Sample source code • File names • File paths 	<ul style="list-style-type: none"> • #define START • system_config.h • <install-dir>/apps/examples
Square Brackets	Optional arguments.	command [options] file [options]
Curly Braces and Pipe Character	Choice of mutually exclusive arguments; an OR selection.	errorlevel {0 1}

Recommended Reading

The following Microchip documents are available and recommended as supplemental reference resources.

Description

Device Data Sheets

Refer to the appropriate device data sheet for device-specific information and specifications.

Reference information found in these data sheets includes:

- Device memory maps
- Device pin out and packaging details
- Device electrical specifications
- List of peripherals included on the devices

To access this documentation, please visit, <http://www.microchip.com/pic32/> and click **Documentation**. Then, expand **Data Sheets** to see the list of available documents.

MPLAB® XC32 C/C++ Compiler User's Guide (DS50001686)

This document details the use of Microchip's MPLAB XC32 Compiler for 32-bit microcontrollers to develop 32-bit applications. Please visit the Microchip website to access the latest version of this document.

MPLAB® X IDE User's Guide (DS50002027)

Consult this document for more information pertaining to the installation and implementation of the MPLAB X IDE software. Please visit the Microchip website to access the latest version of this document.

Documentation Feedback

This topic includes information on how to provide feedback on this documentation.

Description

Your valuable feedback can be provided to Microchip in several ways. Regardless of the method you use to provide feedback, please include the following information whenever possible:

- The Help platform you are viewing:
 - Adobe® Portable Document Format (PDF)
 - Windows® Compiled Help (CHM)
 - HyperText Markup Language (HTML)
- The title of the topic and the section in which it resides
- A clear description of the issue or improvement

How To Send Your Feedback

It is preferred that you use one of the following two methods to provide your feedback:

- Through the Documentation Feedback link, which is available in the header and footer of each topic when viewing compiled

Help (CHM) or HTML Help

- By email at: docerrors@microchip.com

If either of the two previous methods are inconvenient, you may also provide your feedback by:

- Contacting your local Field Applications Engineer
- Contacting Customer Support at: <http://support.microchip.com>

Help Features

Describes the features available in the Help files provided in MPLAB Harmony.

CHM Help Features

Provides detailed information on the features available in CHM Help files.

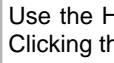
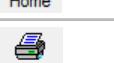
Description

The MPLAB Harmony CHM files are located in the ./doc subfolder of the package it documents. For example, documentation on the MPLAB Harmony 3 Configurator is found at ./mhc/doc/help_mhc.chm and documentation on the MPLAB Harmony Graphics Library is found at ./gfx/doc/help_harmony_gfx.chm.

Help Icons

Several icons are provided in the interface of the Help, which aid in accessing the Help content.

Table 1: Help Icon Features

Help Icon	Description
 Hide  Show	Use the Hide icon to turn off the left Help pane. Once the Hide icon is selected, it is replaced with the Show icon. Clicking the Show icon restores the left Help pane.
 Locate	Use the Locate icon to visually locate the Help topic you are viewing in the Contents. Clicking the Locate icon causes the current topic to be highlighted in blue in the Contents pane.
 Back	Use the Back icon to move back through the previously viewed topics in the order in which they were viewed.
 Forward	Use the Forward icon to move forward through the previously viewed topics in the order in which they were viewed.
 Home	Use the Home icon to return to the first topic in the Help.
 Print	Use the Print icon to print the current topic or the selected heading and all subtopics.
 Options	Use the Options icon to: <ul style="list-style-type: none"> • Hide tabs • Locate a topic • Go Back, Forward, and Home • Stop • Refresh • Set Internet Explorer options • Print topics • Turn Search Highlight Off and On

Topic Window

The Topic Window displays the current topic. In addition to the Help content, special links are provided in the upper portion of the window, as shown in Figure 2. Table 2 lists and describes the different links by their category

Figure 2: Help Links

Table 2: Help Links

Link Category	Description
Topic Path	The full path of the current topic is provided at the top and bottom of each topic, beginning with the top-level section name.
Support and Feedback Links:	<ul style="list-style-type: none"> • Documentation Feedback • Microchip Support
Main Help Links:	<ul style="list-style-type: none"> • Contents • Index • Home
Navigation Links:	<ul style="list-style-type: none"> • Previous • Up • Next

**Notes:**

1. To use the *Documentation Feedback* link, you must have an email system, such as Outlook configured. Clicking the link automatically opens a new email window and populates the recipient and subject lines.
2. The *Home* and *Index* links do not appear initially. Once you begin traversing the topics, they dynamically appear.

Tabs

The CHM Help provides four Tabbed windows: *Contents*, *Index*, *Search*, and *Favorites*.

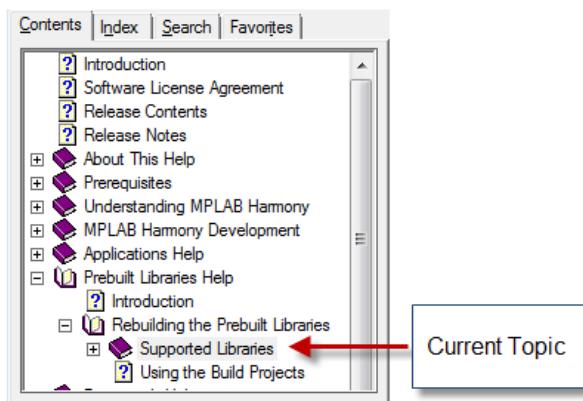
Contents

The Contents tab displays the top-level topics/sections. Figure 3 shows the initial view when the CHM Help is first opened.

Figure 3: Initial Contents Tab View

As topics are explored, the information in the Contents tab dynamically updates. For example, by clicking **Prebuilt Libraries Help** and using the [Next](#) link in the current topic to traverse through this section, the collapsed section automatically expands and the current topic is highlighted in light gray, as shown in Figure 4.

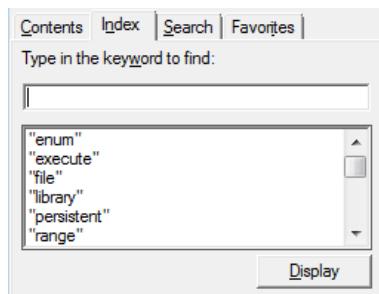
Figure 4: Current Topic Highlighting



Index

Clicking the Index tab results in an alphabetic list of all Help index entries. Figure 5 shows the default Index interface.

Figure 5: Default Index Interface

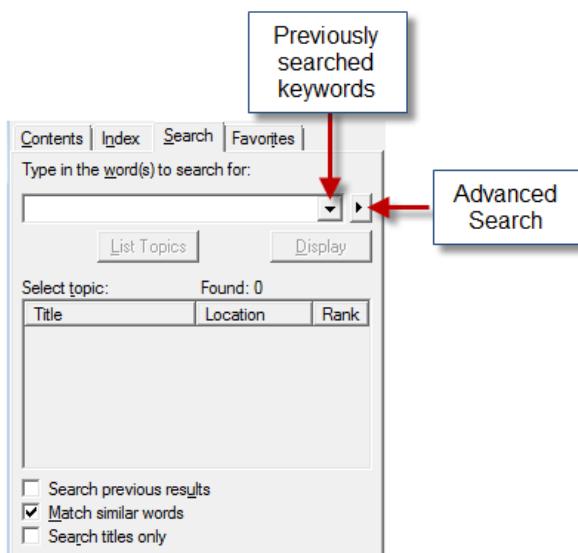


- To locate a specific entry, enter the keyword in the *Type in the keyword to find:* box. As you type, the index list dynamically updates.
- To display the desired item in the list, select the item and click **Display**, or double-click the desired item. The related content appears in the Help window.

Search

Clicking the Search tab provides an efficient way to find specific information. Figure 6 shows the default Search interface.

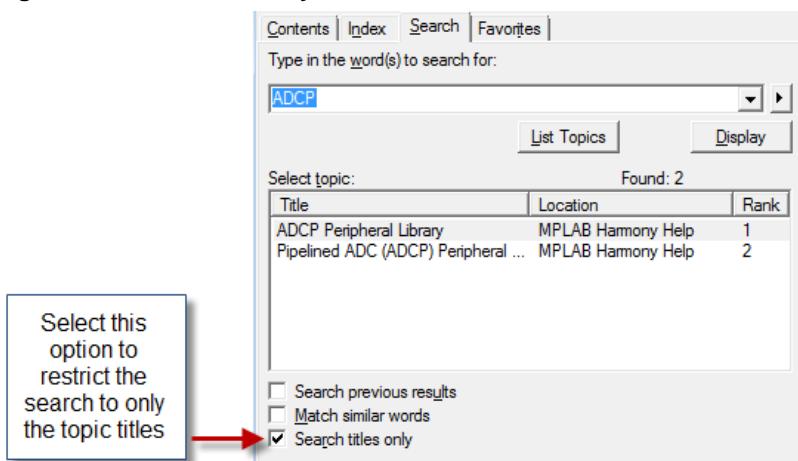
Figure 6: Default Search Interface



- Enter the specific word or words in the *Type in the word(s) to search for:* box
- Clicking the drop-down arrow provides the list of previously searched words
- The right arrow provides Advanced Search options: AND, OR, NEAR, and NOT
- Located at the bottom left of the Search window, three options are provided to narrow-down your search. By default, *Match similar words* is selected. To reduce the number of returned words, clear this box and select *Search titles only*, which restricts

the search to only the topic titles in the Help, as shown in Figure 7.

Figure 7: Search Titles Only



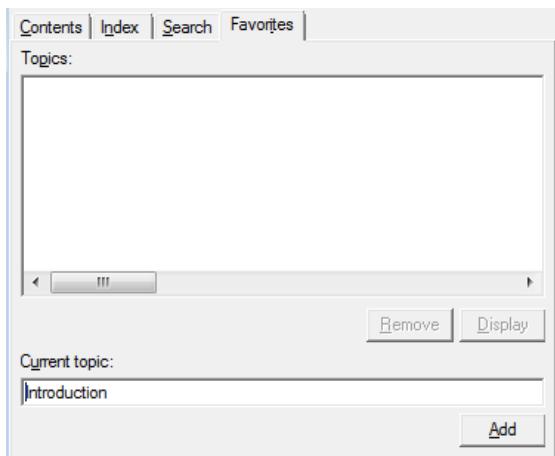
- The *Title* column provides the list of related topics
- The *Location* column lists in which Help system the topic was found (see **Note**)
- The *Rank* column determines the search result that most closely matches the specified word

 **Note:** The *Location* column is automatically included in the CHM Help when the Advanced Search features are implemented and cannot be excluded. Its purpose is to provide the name of the Help system in which the topic is located for Help output that is generated from multiple sources. Since the MPLAB Harmony Help is contained within a single Help system, this information is the same for all searches. Do not confuse this column to mean the actual topic location.

Favorites

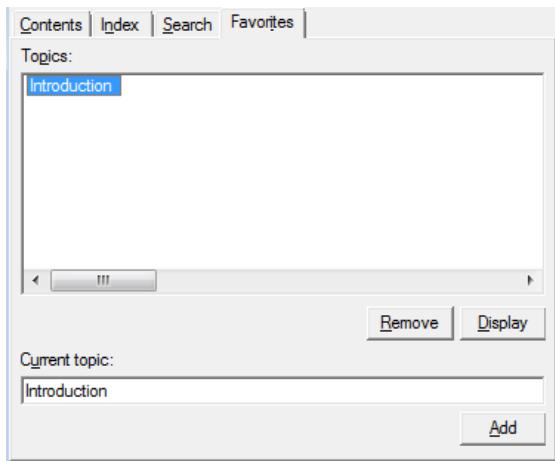
Use the Favorites tab to create a custom list of topics that you may want to repeatedly access. Figure 8 shows the default Favorites interface.

Figure 8: Default Favorites Interface



- The title of the current topic is shown in the *Current topic:* box.
- Click **Add** to add the topic to the *Topics:* list, as shown in Figure 9.
- Click **Display** to view the selected topic.
- Click **Remove** to remove the selected topic from the list of favorites.

Figure 9: Adding a Favorite Topic



HTML Help Features

Provides detailed information on the features available in the stand-alone HTML Help.

Description

The HTML Help output for MPLAB Harmony has two purposes. First, it can be used as "stand-alone" Help. Second, the HTML files are used by the MPLAB Harmony Configurator (MHC) when using MHC in MPLAB X IDE.

Stand-alone HTML Help

The MPLAB Harmony index.html file that is the root for all HTML help is located in the ./doc subfolder of the package it documents. For example, documentation on the MPLAB Harmony 3 Configurator is found at ./mhc/doc/index.html and documentation on the MPLAB Harmony Graphics Library is found at ./gfx/doc/index.html.

To use the HTML Help in a "stand-alone" manner, open the file `index.html` in your browser of choice. Click **Allow blocked content** if a message appears regarding ActiveX controls.

The following links are provided:

- *Table of Contents* - Located at the top left, clicking this link opens the Table of Contents in the right frame
- *Topic Path* - At the top and bottom of each topic, the full path to the current topic is listed
- *Microchip Logo* - Clicking this image opens a new browser tab and displays the Microchip website (www.microchip.com)
- *Contents* - The Contents topic is a static file, which displays and lists the major sections available in the Help in the left frame. Due to a restriction with the Help browser used by the MHC, a dynamic Contents topic cannot be used.
- *Home* - This link returns to the Introduction topic (see **Note 1**)
- *Previous* and *Next* navigation links - Use these links to traverse through the Help topics
- *Documentation Feedback* - Use this link to provide feedback in the form of an email (see **Note 2**)
- *Microchip Support* - Use this link to open the Support page of the Microchip website



- Notes:**
1. The *Home* link does not appear initially. Once you begin traversing the topics, it dynamically appears.
 2. To use the *Documentation Feedback* link, you must have an email system such as Outlook configured. Clicking the link automatically opens a new email message and populates the recipient and subject lines.

PDF Help Features

Provides detailed information on the features available in the PDF version of the Help.

Description

The MPLAB Harmony Help provided in Portable Document Format (PDF) provides many useful features. By default, PDF bookmarks should be visible when opening the file. If PDF bookmarks are not visible, click the PDF Bookmark icon, which is located near the top of the left navigation pane or by selecting *View > Show/Hide > Navigation Panes > Bookmarks*.

The MPLAB Harmony PDF files are located in the ./doc subfolder of the package it documents. For example, documentation on

the MPLAB Harmony 3 Configurator is found at ./mhc/doc/help_mhc.pdf and documentation on the MPLAB Harmony Graphics Library is found at ./gfx/doc/help_harmony_gfx.pdf.

To make full use of the PDF features, it is recommended that Adobe products be used to view the documentation (see **Note**).

Help on how to use the PDF features is available through your copy of Acrobat (or Acrobat Reader) by clicking **Help** in the main menu.



Note: The MPLAB Harmony Help PDF files can be viewed using a PDF viewer or reader that is compatible with Adobe PDF Version 7.0 or later.

Microchip Website

This topic provides general information on the Microchip website.

Description

The Microchip website can be accessed online at: <http://www.microchip.com>

Accessible by most Internet browsers, the following information is available:

Product Support

- Data sheets
- Silicon errata
- Application notes and sample programs
- Design resources
- User's guides
- Hardware support documents
- Latest software releases and archived software

General Technical Support

- Frequently Asked Questions (FAQs)
- Technical support requests
- Online discussion groups
- Microchip consultant program member listings

Business of Microchip

- Product selector and ordering guides
- Latest Microchip press releases
- Listings of seminars and events
- Listings of Microchip sales offices, distributors, and factory representatives

Microchip Forums

This topic provides information on the Microchip Web Forums.

Description

The Microchip Web Forums can be accessed online at: <http://www.microchip.com/forums>

Microchip provides additional online support via our web forums.

The Development Tools Forum is where the MPLAB Harmony forum discussion group is located. This forum handles questions and discussions concerning the MPLAB Harmony Integrated Software Framework and all associated libraries and components.

Additional Development Tool discussion groups include, but are not limited to:

- MPLAB X IDE - This forum handles questions and discussions concerning the released versions of the MPLAB X Integrated Development Environment (IDE)
- MPLAB XC32 - This forum handles questions and discussions concerning Microchip's 32-bit compilers, assemblers, linkers, and related tools for PIC32 microcontrollers
- Tips and Tricks - This forum provides shortcuts and quick workarounds for Microchip's development tools
- FAQs - This forum includes Frequently Asked Questions

Additional forums are also available.

Customer Support

This topic provides information for obtaining support from Microchip.

Description

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office (see [Contact Microchip Technology](#) to locate your local sales office)
- Field Application Engineer (FAE)
- Technical Support (<http://support.microchip.com>)

Contact Microchip Technology

Worldwide sales and service contact information for Microchip Technology Inc.

Description

Please visit the following Microchip Web page for contact information: <http://www.microchip.com/about-us/contact-us>

Glossary

This topic contains a glossary of general MPLAB Harmony terms.

Description

Glossary of MPLAB Harmony Terms

Term	Definition
Application	One or more application modules define the overall behavior of a MPLAB Harmony system. Applications are either demonstrations or examples provided with the installation or are implemented by you, using MPLAB Harmony libraries to accomplish a desired task.
Client	A client module is any module that uses the services (calls the interface functions) of another module.
Configuration	A MPLAB Harmony configuration consists of static definitions (C language <code>#define</code> statements), executable source code, and other definitions in a set of files that are necessary to create a working MPLAB Harmony system. (See the System Configurations section for additional information.)
Configuration Options	Configuration options are the specific set of <code>#define</code> statements that are required by any specific MPLAB Harmony library to specify certain parameters (such as buffer sizes, minimum and maximum values, etc.) build that library. Configuration options are defined in the <code>system_config.h</code> system-wide configuration header.
Driver	A "driver" (or device driver) is a MPLAB Harmony software module designed to control and provide access to a specific peripheral, either built into or external to the microcontroller.
Driver Index	Dynamic MPLAB Harmony drivers (and other dynamic modules) can manage the more than one instance of the peripheral (and other resources) that they control. The "driver index" is a static index number (0, 1, 2,...) that identifies which instance of the driver is to be used.  Note: The driver index is not necessarily identical to the peripheral index. The association between these two is made when the driver is initialized.
Driver Instance	An instance of a driver (or other module) consists of a complete set of the memory (and other resources) controlled by the driver's code. Selection of which set of resources to control is made using a driver index.  Note: Even though there may be multiple instances of the resources managed by a dynamic driver, there is only ever one instance of the actual object code. However, static drivers always maintain a 1:1 relationship between resource and code instances.
Framework	The MPLAB Harmony framework consists of a set of libraries (and the rules and conventions used to create those libraries) that can be used to create MPLAB Harmony systems.

Handle	A handle is a value that allows one software module to "hold" onto a specific instance of some object owned by another software module (analogous to the way a valet holds the handle of a suitcase), creating a link between the two software modules. A handle is an "opaque" value, meaning that the "client" module (the module that receives and holds the handle) must not attempt to interpret the contents or meaning of the handle value. The value of the handle is only meaningful to the "server" module (the module that provides the handle). Internal to the server module, the handle may represent a memory address or it may represent a zero-based index or any other value, as required by the "server" module to identify the "object" to which the client is linked by the handle.
Initialization Overrides	Initialization overrides are configuration options that can be defined to statically override (at build time) parameters that are normally passed into the "Initialize" function of a driver or other MPLAB Harmony module. This mechanism allows you to statically initialize a module, instead of dynamically initializing the module.
Interface	The interface to a module is the set of functions, data types, and other definitions that must be used to interact with that module.
Middleware	The term "middleware" is used to describe any software that fits between the application and the device drivers within a MPLAB Harmony system. This term is used to describe libraries that use drivers to access a peripheral, and then implement communication protocols (such as TCP/IP, USB protocols, and graphics image processing), as well as other more complex processing, which is required to use certain peripherals, but is not actually part of controlling the peripheral itself.
Module	A MPLAB Harmony software module is a closely related group of functions controlling a related set of resources (memory and registers) that can be initialized and maintained by the system. Most MPLAB Harmony modules provide an interface for client interaction. However, "headless" modules with no interface are possible.
Peripheral Index	A peripheral index is a static label (usually an C language "enum" value) that is used to identify a specific instance of a peripheral.  Note: Unlike a driver index, which always starts at '0', a peripheral index may be internally represented as any number, letter, or even a base address and the user should not rely on the value itself, but only the label.
Peripheral Instance	An instance of a peripheral is a complete set of the registers (and internal physical resources) necessary to provide the core functionality of a given type of peripheral (either built into or external to the microcontroller).  Note: A specific peripheral instance is identified using a peripheral index.
System	A MPLAB Harmony system is a complete set of libraries, applications, and configuration items loaded and executing on a specific hardware platform (microcontroller, board, and external peripherals) or the source items necessary to build such a system.  Note: Since a system can multiple configurations, one MPLAB Harmony project may support multiple systems through multiple supported configurations. See the demonstration applications included in the installation for examples.
System Service	A system service is a MPLAB Harmony module that provides access to and control of common system resources (memory and registers) with which other modules (drivers, middleware, libraries and application) may interact.  Note: System services, much like drivers, manage sharing of resources so as to avoid conflicts between modules that would otherwise occur if each module attempted to manage the share resource itself. But, unlike drivers, system services do not normally need to be "opened" to use them.

Index

(

- (1) Board Support Package 78
- (2) Display Component 79
- (3) Display Driver Component 81
- (4) GFX Core Component 82
- (5) Aria Graphics Library Component 85
- (6) Graphics Application Template 86
- (7) Input System Service Component 87
- (8) MaxTouch Controller Component 88

A

- Abstraction Model 747
- ACTIVE_AREA enumeration member 581
- Adding an Event to the Aria Quickstart Demonstration 28
- Advanced Topics 178
- ANGLE_ARC enumeration member 578
- Aria HAL Driver Examples 747
- Aria User Interface Library 203
- Aria User Interface Library Interface 209
- aria_benchmark 33
- aria_coffeemaker Demonstration Example 185
- aria_quickstart 38
- aria_showcase 47
- aria_showcase_reloaded 56
- aria_weather_forecast 66

B

- BAR_GRAPH_AXIS_0 enumeration member 569
- BAR_GRAPH_TICK_CENTER enumeration member 569
- BAR_GRAPH_TICK_IN enumeration member 569
- BAR_GRAPH_TICK_OUT enumeration member 569
- Binary Assets 156
- Building the Application 35, 42, 49, 59, 68
- BUTTON_LAST enumeration member 614
- BUTTON_LEFT enumeration member 614
- BUTTON_MIDDLE enumeration member 614
- BUTTON_NONE enumeration member 614
- BUTTON_RIGHT enumeration member 614
- BUTTON_WHEEL_DOWN enumeration member 614
- BUTTON_WHEEL_UP enumeration member 614

C

- CHM Help Features 802
- CIRCLE_BUTTON enumeration member 581
- CIRCULAR_GAUGE_DIR_CLOCKWISE enumeration member 578
- CIRCULAR_GAUGE_DIR_COUNTER_CLOCKWISE enumeration member 578
- CIRCULAR_GAUGE_LABEL_INSIDE enumeration member 578
- CIRCULAR_GAUGE_LABEL_OUTSIDE enumeration member 578
- CIRCULAR_SLIDER_DIR_CLOCKWISE enumeration member 581
- CIRCULAR_SLIDER_DIR_COUNTER_CLOCKWISE enumeration member 581
- Code Generation 178
- Components to Graph Map 77
- Configuring the Hardware 35, 42, 50, 60, 68
- Contact Microchip Technology 808
- Creating New Graphics Applications 17

Custom Event Handling and Dynamic Widget Creation 190

Customer Support 808

D

- DDR Organizer 135
- Demonstrations 33
 - Graphics Library 33
- Documentation Feedback 801
- Draw Pipeline Options 180

E

- Enabling Demo Mode in a MPLAB Harmony Graphics Application 31
- Event Manager 157
- Example Graphics Projects 12
- Exploring Aria Quickstart 75

F

- Files 640, 738, 790
- Font Assets 145

G

- GFX_AbsoluteValue function 756
- GFX_ActiveContext function 692
- GFX_ANTIALIAS_MODE_COUNT macro 720
- GFX_ANTIALIAS_OFF enumeration member 720
- GFX_ANTIALIAS_ON enumeration member 720
- GFX_AntialiasMode enumeration 720
- GFX_AntialiasMode_t enumeration 720
- GFX_ASSERT macro 721
- GFX_Atan function 756
- GFX_BitsPerPixel enumeration 721
- GFX_BitsPerPixel_t enumeration 721
- GFX_BLEND_ALL enumeration member 721
- GFX_BLEND_CHANNEL enumeration member 721
- GFX_BLEND_GLOBAL enumeration member 721
- GFX_BLEND_NONE enumeration member 721
- GFX_BlendMode enumeration 721
- GFX_BlendMode_t enumeration 721
- GFX_BPP1 enumeration member 721
- GFX_BPP16 enumeration member 721
- GFX_BPP24 enumeration member 721
- GFX_BPP32 enumeration member 721
- GFX_BPP4 enumeration member 721
- GFX_BPP8 enumeration member 721
- GFX_BS_ADDRESS enumeration member 722
- GFX_BS_MALLOC enumeration member 722
- GFX_BS_MANAGED enumeration member 722
- GFX_BS_NONE enumeration member 722
- GFX_BUFFER_READ enumeration member 721
- GFX_BUFFER_WRITE enumeration member 721
- GFX_BufferSelection enumeration 721
- GFX_BufferSelection_t enumeration 721
- GFX_BufferState enumeration 722
- GFX_BufferState_t enumeration 722
- GFX_Calloc_FnPtr type 722
- GFX_Clampf function 757
- GFX_Clampi function 757
- gfx_color.h 740
- GFX_COLOR_BLACK enumeration member 725

GFX_COLOR_BLUE enumeration member 725
GFX_COLOR_CYAN enumeration member 725
GFX_COLOR_DARKGRAY enumeration member 725
GFX_COLOR_GRAY enumeration member 725
GFX_COLOR_GREEN enumeration member 725
GFX_COLOR_LAST enumeration member 725
GFX_COLOR_LIGHTGRAY enumeration member 725
GFX_COLOR_LIME enumeration member 725
GFX_COLOR_MAGENTA enumeration member 725
GFX_COLOR_MAROON enumeration member 725
GFX_COLOR_MASK_ALL enumeration member 724
GFX_COLOR_MASK_ARGB_8888 enumeration member 724
GFX_COLOR_MASK_GS_8 enumeration member 724
GFX_COLOR_MASK_RGB_332 enumeration member 724
GFX_COLOR_MASK_RGB_565 enumeration member 724
GFX_COLOR_MASK_RGB_888 enumeration member 724
GFX_COLOR_MASK_RGBA_5551 enumeration member 724
GFX_COLOR_MASK_RGBA_8888 enumeration member 724
GFX_COLOR_MASK_YUV enumeration member 724
GFX_COLOR_MODE_ARGB_8888 enumeration member 724
GFX_COLOR_MODE_COUNT macro 723
GFX_COLOR_MODE_GS_8 enumeration member 724
GFX_COLOR_MODE_INDEX_1 enumeration member 724
GFX_COLOR_MODE_INDEX_4 enumeration member 724
GFX_COLOR_MODE_INDEX_8 enumeration member 724
GFX_COLOR_MODE_IS_ALPHA macro 723
GFX_COLOR_MODE_IS_INDEX macro 723
GFX_COLOR_MODE_IS_PIXEL macro 723
GFX_COLOR_MODE_LAST enumeration member 724
GFX_COLOR_MODE_LAST_COLOR macro 724
GFX_COLOR_MODE_RGB_332 enumeration member 724
GFX_COLOR_MODE_RGB_565 enumeration member 724
GFX_COLOR_MODE_RGB_888 enumeration member 724
GFX_COLOR_MODE_RGBA_5551 enumeration member 724
GFX_COLOR_MODE_RGBA_8888 enumeration member 724
GFX_COLOR_MODE_YUV enumeration member 724
GFX_COLOR_NAVY enumeration member 725
GFX_COLOR OLIVE enumeration member 725
GFX_COLOR_PURPLE enumeration member 725
GFX_COLOR_RED enumeration member 725
GFX_COLOR_SILVER enumeration member 725
GFX_COLOR_TEAL enumeration member 725
GFX_COLOR_WHITE enumeration member 725
GFX_COLOR_YELLOW enumeration member 725
GFX_ColorBlerp function 693
GFX_ColorBlend_RGBA_8888 function 693
GFX_ColorChannelAlpha function 694
GFX_ColorChannelBlue function 694
GFX_ColorChannelGreen function 695
GFX_ColorChannelRed function 695
GFX_ColorConvert function 695
GFX_ColorInfo variable 724
GFX_ColorLerp function 696
GFX_ColorMask enumeration 724
GFX_ColorMask_t enumeration 724
GFX_ColorMode enumeration 724
GFX_ColorMode_t enumeration 724
GFX_ColorModelInfo structure 725
GFX_ColorModelInfo_t structure 725
GFX_ColorModelInfoGet function 696
GFX_ColorName enumeration 725
GFX_ColorName_t enumeration 725
GFX_ColorValue function 697
gfx_common.h 738
GFX_Context structure 726
gfx_context.h 741
GFX_Context_t structure 726
GFX_ContextActiveSet function 697
gfx_default_impl.h 741
GFX_DEPRECATED macro 727
gfx_display.h 741
GFX_DisplayInfo structure 727
GFX_DisplayInfo_t structure 727
GFX_DivideRounding function 757
gfx_draw.h 742
gfx_draw_arc.h 743
gfx_draw.blit.h 743
gfx_draw_circle.h 743
gfx_draw_ellipse.h 743
GFX_DRAW_FILL enumeration member 728
GFX_DRAW_GRADIENT_LEFT_RIGHT enumeration member 728
GFX_DRAW_GRADIENT_TOP_BOTTOM enumeration member 728
GFX_DRAW_LINE enumeration member 728
gfx_draw_line.h 743
GFX_DRAW_MODE_COUNT macro 728
gfx_draw_pixel.h 743
gfx_draw_rect.h 743
gfx_draw_stretchblit.h 743
GFX_DrawArc function 758
GFX_DrawBlit function 697
GFX_DrawCircle function 698
GFX_DrawDirectBlit function 698
GFX_DrawEllipse function 758
GFX_DrawLine function 699
GFX_DrawMode enumeration 728
GFX_DrawMode_t enumeration 728
GFX_DrawPipeline type 729
GFX_DrawPixel function 700
GFX_DrawPixelByDrawState function 759
GFX_DrawRect function 700
GFX_DrawState structure 729
GFX_DrawState_t structure 729
GFX_DrawStretchBit function 701
gfx_driver_interface.h 743
GFX_DriverInfo structure 730
GFX_DriverInfo_t structure 730
GFX_EllipsePoint function 760
GFX_Flag enumeration 730
GFX_Flag_t enumeration 730
GFX_FrameBuffer structure 731
GFX_FrameBuffer_t structure 731
GFX_Free_FnPtr type 731
gfx_hal.h 744
gfx_interface.h 744
GFX_Layer structure 732
gfx_layer.h 744

GFX_Layer_t structure 732
GFX_LayerFromOrientedSpace function 702
GFX_LayerPointFromOrientedSpace function 702
GFX_LayerPointToOrientedSpace function 703
GFX_LayerReadBuffer function 703
GFX_LayerRectFromOrientedSpace function 703
GFX_LayerRectToOrientedSpace function 704
GFX_LayerRotate function 705
GFX_LayerSwap function 705
GFX_LayerToOrientedSpace function 705
GFX_LayerWriteBuffer function 706
GFX_Lerp function 760
GFX_Malloc_FnPtr type 733
gfx_math.h 790
GFX_Maxf function 760
GFX_Maxi function 761
GFX_Memcpy_FnPtr type 733
GFX_MemoryIntf structure 734
GFX_MemoryIntf_t structure 734
GFX_Memset_FnPtr type 734
GFX_Minf function 761
GFX_Mini function 762
GFX_Normalize360 function 762
GFX_NUM_FLAGS macro 734
GFX_Orientation enumeration 735
GFX_ORIENTATION_0 enumeration member 735
GFX_ORIENTATION_180 enumeration member 735
GFX_ORIENTATION_270 enumeration member 735
GFX_ORIENTATION_90 enumeration member 735
GFX_Orientation_t enumeration 735
GFX_Percent function 763
GFX_PercentOf function 763
GFX_PercentOfDec function 763
GFX_PercentWholeRounded function 764
GFX_PIPELINE_GCU enumeration member 735
GFX_PIPELINE_GCUGPU enumeration member 735
GFX_PIPELINE_GPU enumeration member 735
GFX_PIPELINE_MODE_COUNT macro 735
GFX_PIPELINE_SOFTWARE enumeration member 735
GFX_PipelineMode enumeration 735
GFX_PipelineMode_t enumeration 735
gfx_pixel_buffer.h 745
GFX_PixelBuffer structure 736
GFX_PixelBuffer_t structure 736
GFX_PixelBufferAreaFill function 706
GFX_PixelBufferAreaFill_Unsafe function 707
GFX_PixelBufferAreaGet function 707
GFX_PixelBufferAreaGet_Unsafe function 708
GFX_PixelBufferAreaSet function 708
GFX_PixelBufferAreaSet_Unsafe function 709
GFX_PixelBufferClipRect function 709
GFX_PixelBufferConvert function 710
GFX_PixelBufferCopy function 710
GFX_PixelBufferCreate function 711
GFX_PixelBufferDestroy function 711
GFX_PixelBufferGet function 712
GFX_PixelBufferGet_Unsafe function 712
GFX_PixelBufferGetIndex function 713
GFX_PixelBufferOffsetGet function 713
GFX_PixelBufferOffsetGet_Unsafe function 714
GFX_PixelBufferSet function 714
GFX_PixelBufferSet_Unsafe function 715
GFX_Point type 567
GFX_Point_t structure 736
GFX_PolarToXY function 764
gfx_processor_interface.h 746
GFX_Realloc_FnPtr type 736
GFX_Rect type 567
gfx_rect.h 746
GFX_Rect_t structure 737
GFX_RectClip function 715
GFX_RectClipAdj function 716
GFX_RectCombine function 716
GFX_RectCompare function 717
GFX_RectContainsPoint function 717
GFX_RectContainsRect function 717
GFX_RectFromPoints function 718
GFX_RectIntersects function 718
GFX_RectsAreSimilar function 719
GFX_RectSplit function 719
GFX_RectToPoints function 719
GFX_RESIZE_BILINEAR enumeration member 737
GFX_RESIZE_NEARESTNEIGHBOR enumeration member 737
GFX_ResizeMode enumeration 737
GFX_ResizeMode_t enumeration 737
GFX_ScaleInteger function 765
GFX_ScaleIntegerSigned function 765
GFX_SineCosineGet function 765
GFX_Size structure 737
GFX_Size_t structure 737
GFX_TRIG_FUNCTION_TYPE enumeration 780
GXF_BRIGHTNESS enumeration member 730
GXF_BRIGHTNESS_RANGE enumeration member 730
GXF_COLOR_MODE enumeration member 730
GXF_DISPLAY_COUNT enumeration member 730
GXF_DISPLAY_INFO enumeration member 730
GXF_DRAW_ALPHA_ENABLE enumeration member 730
GXF_DRAW_ALPHA_VALUE enumeration member 730
GXF_DRAW_BLEND_MODE enumeration member 730
GXF_DRAW_CLIP_ENABLE enumeration member 730
GXF_DRAW_CLIP_RECT enumeration member 730
GXF_DRAW_COLOR enumeration member 730
GXF_DRAW_GRADIENT_COLOR enumeration member 730
GXF_DRAW_MASK_ENABLE enumeration member 730
GXF_DRAW_MASK_VALUE enumeration member 730
GXF_DRAW_MODE enumeration member 730
GXF_DRAW_PALETTE enumeration member 730
GXF_DRAW_PIPELINE_MODE enumeration member 730
GXF_DRAW_RESIZE_MODE enumeration member 730
GXF_DRAW_TARGET enumeration member 730
GXF_DRAW_THICKNESS enumeration member 730
GXF_DRIVER_COUNT enumeration member 730
GXF_DRIVER_INFO enumeration member 730
GXF_GLOBAL_PALETTE enumeration member 730
GXF_HSYNC_CALLBACK enumeration member 730
GXF_LAST_FLAG enumeration member 730

GFXF_LAYER_ACTIVE enumeration member 730
 GFXF_LAYER_ALPHA_AMOUNT enumeration member 730
 GFXF_LAYER_ALPHA_ENABLE enumeration member 730
 GFXF_LAYER_BUFFER_ADDRESS enumeration member 730
 GFXF_LAYER_BUFFER_ALLOCATE enumeration member 730
 GFXF_LAYER_BUFFER_COHERENT enumeration member 730
 GFXF_LAYER_BUFFER_COUNT enumeration member 730
 GFXF_LAYER_BUFFER_FREE enumeration member 730
 GFXF_LAYER_COUNT enumeration member 730
 GFXF_LAYER_ENABLED enumeration member 730
 GFXF_LAYER_INVALID enumeration member 730
 GFXF_LAYER_MASK_COLOR enumeration member 730
 GFXF_LAYER_MASK_ENABLE enumeration member 730
 GFXF_LAYER_POSITION enumeration member 730
 GFXF_LAYER_SIZE enumeration member 730
 GFXF_LAYER_SWAP enumeration member 730
 GFXF_LAYER_SWAP_SYNC enumeration member 730
 GFXF_LAYER_VISIBLE enumeration member 730
 GFXF_LAYER_VSYNC enumeration member 730
 GFXF_MIRRORED enumeration member 730
 GFXF_NONE enumeration member 730
 GFXF_ORIENTATION enumeration member 730
 GFXF_VSYNC_CALLBACK enumeration member 730
 GFXU_ASSET_LOCATION_INTERNAL macro 781
 GFXU_ASSET_TYPE_BINARY enumeration member 781
 GFXU_ASSET_TYPE_FONT enumeration member 781
 GFXU_ASSET_TYPE_IMAGE enumeration member 781
 GFXU_ASSET_TYPE_PALETTE enumeration member 781
 GFXU_ASSET_TYPE_STRINGTABLE enumeration member 781
 GFXU_AssetHeader structure 781
 GFXU_AssetHeader_t structure 781
 GFXU_AssetType enumeration 781
 GFXU_AssetType_t enumeration 781
 gfxu_binary.h 791
 GFXU_BinaryAsset structure 782
 GFXU_BinaryAsset_t structure 782
 GFXU_CalculateCharStringWidth function 766
 GFXU_CalculatePartialCharStringWidth function 766
 GFXU_CalculatePartialStringWidth function 767
 GFXU_CalculateStringWidth function 767
 GFXU_CHAR type 782
 GFXU_CompareString function 768
 GFXU_DrawCharString function 768
 GFXU_DrawCharStringClipped function 769
 GFXU_DrawCharSubStringClipped function 770
 GFXU_DrawImage function 771
 GFXU_DrawString function 771
 GFXU_DrawStringClipped function 772
 GFXU_DrawSubStringClipped function 773
 GFXU_ExternalAssetReader structure 782
 GFXU_ExternalAssetReader_t structure 782
 GFXU_ExternalAssetReaderRun_FnPtr type 783
 GFXU_ExternalAssetReaderStatus enumeration 783
 GFXU_ExternalAssetReaderStatus_t enumeration 783
 GFXU_ExtractString function 774
 gfxu_font.h 792
 GFXU_FONT_BPP_1 enumeration member 784
 GFXU_FONT_BPP_8 enumeration member 784
 GFXU_FontAsset type 783
 GFXU_FontAsset_t structure 784
 GFXU_FontAssetBPP enumeration 784
 GFXU_FontGlyphIndexTable structure 784
 GFXU_FontGlyphIndexTable_t structure 784
 GFXU_FontGlyphRange structure 785
 GFXU_FontGlyphRange_t structure 785
 GFXU_GetCharAt function 774
 GFXU_GetCharStringLineRect function 775
 GFXU_GetCharWidth function 776
 GFXU_GetStringAscent function 776
 GFXU_GetStringHeight function 776
 GFXU_GetStringLength function 777
 GFXU_GetStringLineRect function 777
 GFXU_GetStringRect function 778
 GFXU_GetStringSizeInBytes function 778
 gfxu_global.h 793
 gfxu_image.h 794
 GFXU_IMAGE_COMPRESSION_NONE enumeration member 786
 GFXU_IMAGE_COMPRESSION_RLE enumeration member 786
 GFXU_IMAGE_DIRECT_BLIT enumeration member 786
 GFXU_IMAGE_FORMAT_JPEG enumeration member 786
 GFXU_IMAGE_FORMAT_PNG enumeration member 786
 GFXU_IMAGE_FORMAT_RAW enumeration member 786
 GFXU_IMAGE_SUPPORTS_CLIPPING enumeration member 786
 GFXU_IMAGE_USE_MASK enumeration member 786
 gfxu_image_utils.h 795
 GFXU_ImageAsset structure 785
 GFXU_ImageAsset_t structure 785
 GFXU_ImageCompressionType enumeration 786
 GFXU_ImageCompressionType_t enumeration 786
 GFXU_ImageFlags enumeration 786
 GFXU_ImageFlags_t enumeration 786
 GFXU_ImageFormat enumeration 786
 GFXU_ImageFormat_t enumeration 786
 GFXU_MediaCloseRequest_FnPtr type 787
 GFXU_MediaOpenRequest_FnPtr type 787
 GFXU_MediaReadRequest_FnPtr type 787
 GFXU_MediaReadRequestCallback_FnPtr type 787
 GFXU_MemoryIntf structure 788
 GFXU_MemoryIntf_t structure 788
 gfxu_palette.h 795
 GFXU_PaletteAsset structure 788
 GFXU_PaletteAsset_t structure 788
 GFXU_PaletteGetColor function 779
 GFXU_PreprocessImage function 779
 GFXU_READER_STATUS_ABORTED enumeration member 783
 GFXU_READER_STATUS_DRAWING enumeration member 783
 GFXU_READER_STATUS_FINISHED enumeration member 783
 GFXU_READER_STATUS_INVALID enumeration member 783
 GFXU_READER_STATUS_READY enumeration member 783
 GFXU_READER_STATUS_WAITING enumeration member 783
 gfxu_string.h 796
 GFXU_STRING_ARRAY_SIZE macro 789
 GFXU_STRING_ENCODING_ASCII enumeration member 789
 GFXU_STRING_ENCODING_UTF16 enumeration member 789
 GFXU_STRING_ENCODING_UTF8 enumeration member 789
 GFXU_STRING_ENTRY_SIZE macro 789

GFXU_STRING_MAX_CHAR_WIDTH macro 789
 gfxu_string_utils.h 797
 GFXU_StringEncodingMode enumeration 789
 GFXU_StringEncodingMode_t enumeration 789
 GFXU_StringTableAsset structure 790
 GFXU_StringTableAsset_t structure 790
 Global Palette 163
 Glossary 808
 Graphics Basics 2
 Graphics Composer Asset Management 130
 Graphics Composer Events 158
 Graphics Composer Macros 162
 Graphics Composer Suite Goals 203
 Graphics Composer Window User Interface 90
 Graphics Demonstrations 33
 Graphics Events Testbed 163
 Graphics Library Help 201
 Graphics Overview 2
 Graphics Pipeline 180
 Graphics Pipeline Options 184
 Graphics Stack Architecture 201
 Graphics Utilities Interface 751
 Graphics Utilities Library 749

H

Hardware Abstraction Layer (HAL) 683
 Heap Estimator 166
 Help Features 802
 How Does MPLAB Harmony Graphics Work? 9
 How the Library Works 748
 HTML Help Features 806

I

ILI9488 Display Controller Driver Library 747
 Image Assets 138
 Importing and Exporting Graphics Data 195
 Improved Touch Performance with Phantom Buttons 185
 INACTIVE_AREA enumeration member 581
 INSIDE_CIRCLE_BORDER enumeration member 581
 Introduction 33, 75, 89, 201, 203, 683, 747
 It's Fast 7
 It's Free 9
 It's Good 5

K

KEY_0 enumeration member 594
 KEY_1 enumeration member 594
 KEY_2 enumeration member 594
 KEY_3 enumeration member 594
 KEY_4 enumeration member 594
 KEY_5 enumeration member 594
 KEY_6 enumeration member 594
 KEY_7 enumeration member 594
 KEY_8 enumeration member 594
 KEY_9 enumeration member 594
 KEY_A enumeration member 594
 KEY_B enumeration member 594
 KEY_BACKQUOTE enumeration member 594
 KEY_BACKSLASH enumeration member 594

KEY_BACKSPACE enumeration member 594
 KEY_BRACKET_LEFT enumeration member 594
 KEY_BRACKET_RIGHT enumeration member 594
 KEY_C enumeration member 594
 KEY_CAPSLOCK enumeration member 594
 KEY_COMMA enumeration member 594
 KEY_D enumeration member 594
 KEY_DOWN enumeration member 594
 KEY_E enumeration member 594
 KEY_END enumeration member 594
 KEY_ENTER enumeration member 594
 KEY_EQUALS enumeration member 594
 KEY_ESCAPE enumeration member 594
 KEY_F enumeration member 594
 KEY_F1 enumeration member 594
 KEY_F10 enumeration member 594
 KEY_F11 enumeration member 594
 KEY_F12 enumeration member 594
 KEY_F2 enumeration member 594
 KEY_F3 enumeration member 594
 KEY_F4 enumeration member 594
 KEY_F5 enumeration member 594
 KEY_F6 enumeration member 594
 KEY_F7 enumeration member 594
 KEY_F8 enumeration member 594
 KEY_F9 enumeration member 594
 KEY_G enumeration member 594
 KEY_H enumeration member 594
 KEY_HOME enumeration member 594
 KEY_I enumeration member 594
 KEY_INSERT enumeration member 594
 KEY_J enumeration member 594
 KEY_K enumeration member 594
 KEY_KP_0 enumeration member 594
 KEY_KP_1 enumeration member 594
 KEY_KP_2 enumeration member 594
 KEY_KP_3 enumeration member 594
 KEY_KP_4 enumeration member 594
 KEY_KP_5 enumeration member 594
 KEY_KP_6 enumeration member 594
 KEY_KP_7 enumeration member 594
 KEY_KP_8 enumeration member 594
 KEY_KP_9 enumeration member 594
 KEY_KP_DIVIDE enumeration member 594
 KEY_KP_ENTER enumeration member 594
 KEY_KP_MINUS enumeration member 594
 KEY_KP_MULTIPLY enumeration member 594
 KEY_KP_PERIOD enumeration member 594
 KEY_KP_PLUS enumeration member 594
 KEY_L enumeration member 594
 KEY_LALT enumeration member 594
 KEY_LAST enumeration member 594
 KEY_LCTRL enumeration member 594
 KEY_LEFT enumeration member 594
 KEY_LMETA enumeration member 594
 KEY_LSHIFT enumeration member 594
 KEY_M enumeration member 594
 KEY_MINUS enumeration member 594

KEY_N enumeration member 594
 KEY_NULL enumeration member 594
 KEY_NUMLOCK enumeration member 594
 KEY_O enumeration member 594
 KEY_P enumeration member 594
 KEY_PAGEDOWN enumeration member 594
 KEY_PAGEUP enumeration member 594
 KEY_PAUSE enumeration member 594
 KEY_PERIOD enumeration member 594
 KEY_PRINTSCREEN enumeration member 594
 KEY_Q enumeration member 594
 KEY_QUOTE enumeration member 594
 KEY_R enumeration member 594
 KEY_RALT enumeration member 594
 KEY_RCTRL enumeration member 594
 KEY_RIGHT enumeration member 594
 KEY_RMETA enumeration member 594
 KEY_RSHIFT enumeration member 594
 KEY_S enumeration member 594
 KEY_SCROLLLOCK enumeration member 594
 KEY_SEMICOLON enumeration member 594
 KEY_SLASH enumeration member 594
 KEY_SPACE enumeration member 594
 KEY_T enumeration member 594
 KEY_TAB enumeration member 594
 KEY_U enumeration member 594
 KEY_UP enumeration member 594
 KEY_V enumeration member 594
 KEY_W enumeration member 594
 KEY_X enumeration member 594
 KEY_Y enumeration member 594
 KEY_Z enumeration member 594

L

LA_BUFFER_TYPE_ADDRESS enumeration member 602
 LA_BUFFER_TYPE_AUTO enumeration member 602
 LA_BUTTON_STATE_DOWN enumeration member 571
 LA_BUTTON_STATE_TOGGLED enumeration member 571
 LA_BUTTON_STATE_UP enumeration member 571
 LA_CIRCULAR_SLIDER_STATE_DOWN enumeration member 579
 LA_CIRCULAR_SLIDER_STATE_UP enumeration member 579
 LA_CONTEXT_FRAME_DRAWING enumeration member 583
 LA_CONTEXT_FRAME_POSTLAYER enumeration member 583
 LA_CONTEXT_FRAME_PREFRAME enumeration member 583
 LA_CONTEXT_FRAME_PRELAYER enumeration member 583
 LA_CONTEXT_FRAME_READY enumeration member 583
 LA_CONTEXT_UPDATE_DONE enumeration member 584
 LA_CONTEXT_UPDATE_PENDING enumeration member 584
 LA_EVENT_DEFERRED enumeration member 586
 LA_EVENT_HANDLED enumeration member 586
 LA_EVENT_NONE enumeration member 586
 LA_EVENT_RESET_QUEUE enumeration member 586
 LA_EVENT_SCREEN_CHANGE enumeration member 586
 LA_EVENT_TOUCH_DOWN enumeration member 586
 LA_EVENT_TOUCH_MOVED enumeration member 586
 LA_EVENT_TOUCH_UP enumeration member 586
 LA_FAILURE enumeration member 624
 LA_FALSE enumeration member 570

LA_GESTURE_NONE enumeration member 587
 LA_GRADIENT_DIRECTION_DOWN enumeration member 588
 LA_GRADIENT_DIRECTION_LEFT enumeration member 588
 LA_GRADIENT_DIRECTION_RIGHT enumeration member 588
 LA_GRADIENT_DIRECTION_UP enumeration member 588
 LA_IMAGEFILTER_BILINEAR enumeration member 589
 LA_IMAGEFILTER_NEARESTNEIGHBOR enumeration member 589
 LA_KEYPAD_CELL_ACTION_ACCEPT enumeration member 597
 LA_KEYPAD_CELL_ACTION_APPEND enumeration member 597
 LA_KEYPAD_CELL_ACTION_BACKSPACE enumeration member 597
 LA_KEYPAD_CELL_ACTION_CLEAR enumeration member 597
 LA_KEYPAD_CELL_ACTION_NONE enumeration member 597
 LA_KEYPAD_CELL_ACTION_SET enumeration member 597
 LA_KEYPAD_TRIGGER_KEYPRESSED enumeration member 596
 LA_KEYPAD_TRIGGER_KEYRELEASED enumeration member 596
 LA_LAYER_FRAME_COMPLETE enumeration member 602
 LA_LAYER_FRAME_IN_PROGRESS enumeration member 602
 LA_LAYER_FRAME_PREFRAME enumeration member 602
 LA_LAYER_FRAME_READY enumeration member 602
 LA_LIST_WIDGET_SELECTION_MODE_CONTIGUOUS enumeration member 613
 LA_LIST_WIDGET_SELECTION_MODE_MULTIPLE enumeration member 613
 LA_LIST_WIDGET_SELECTION_MODE_SINGLE enumeration member 613
 LA_LISTWHEEL_INDICATOR_FILL_GRADIENT enumeration member 607
 LA_LISTWHEEL_INDICATOR_FILL_NONE enumeration member 607
 LA_LISTWHEEL_INDICATOR_FILL_SOLID enumeration member 607
 LA_LISTWHEEL_ZOOM_EFFECT_FIXED_SCALE enumeration member 612
 LA_LISTWHEEL_ZOOM_EFFECT_FULL_SCALE enumeration member 612
 LA_LISTWHEEL_ZOOM_EFFECT_NONE enumeration member 612
 LA_LISTWHEEL_ZOOM_EFFECT_VSCALE enumeration member 612
 LA_PREEMPTION_LEVEL_0 enumeration member 616
 LA_PREEMPTION_LEVEL_1 enumeration member 616
 LA_PREEMPTION_LEVEL_2 enumeration member 616
 LA_PROGRESSBAR_DIRECTION_DOWN enumeration member 616
 LA_PROGRESSBAR_DIRECTION_LEFT enumeration member 616
 LA_PROGRESSBAR_DIRECTION_RIGHT enumeration member 616
 LA_PROGRESSBAR_DIRECTION_UP enumeration member 616
 LA_RADIAL_MENU_ELLIPSE_TYPE_DEFAULT enumeration member 617
 LA_RADIAL_MENU_ELLIPSE_TYPE_ORBITAL enumeration member 617
 LA_RADIAL_MENU_ELLIPSE_TYPE_ROLLODEX enumeration member 617
 LA_RADIAL_MENU_HANDLE_USER_MOVE_REQUEST enumeration member 620
 LA_RADIAL_MENU_INIT enumeration member 620
 LA_RADIAL_MENU_INPUT_READY enumeration member 620
 LA_RADIAL_MENU_RESET_TO_INPUT_POS enumeration member 620
 LA_RADIAL_MENU_SCALE_GRADUAL enumeration member 620
 LA_RADIAL_MENU_SCALE_OFF enumeration member 620
 LA_RADIAL_MENU_SCALE_PROMINENT enumeration member 620
 LA_RELATIVE_POSITION_ABOVE enumeration member 624
 LA_RELATIVE_POSITION_BEHIND enumeration member 624
 LA_RELATIVE_POSITION_BELOW enumeration member 624

LA_RELATIVE_POSITION_LEFTOF enumeration member 624
LA_RELATIVE_POSITION_RIGHTOF enumeration member 624
LA_SCREEN_ORIENTATION_0 enumeration member 627
LA_SCREEN_ORIENTATION_180 enumeration member 627
LA_SCREEN_ORIENTATION_270 enumeration member 627
LA_SCREEN_ORIENTATION_90 enumeration member 627
LA_SCROLLBAR_ORIENT_HORIZONTAL enumeration member 627
LA_SCROLLBAR_ORIENT_VERTICAL enumeration member 627
LA_SCROLLBAR_STATE_BOTTOM_INSIDE enumeration member 627
LA_SCROLLBAR_STATE_BOTTOM_PRESSED enumeration member 627
LA_SCROLLBAR_STATE_HANDLE_DOWN enumeration member 627
LA_SCROLLBAR_STATE_NONE enumeration member 627
LA_SCROLLBAR_STATE_TOP_INSIDE enumeration member 627
LA_SCROLLBAR_STATE_TOP_PRESSED enumeration member 627
LA_SLIDER_ORIENT_HORIZONTAL enumeration member 629
LA_SLIDER_ORIENT_VERTICAL enumeration member 629
LA_SLIDER_STATE_AREA_DOWN enumeration member 629
LA_SLIDER_STATE_HANDLE_DOWN enumeration member 629
LA_SLIDER_STATE_NONE enumeration member 629
LA_SUCCESS enumeration member 624
LA_TOUCHTEST_STATE_DOWN enumeration member 632
LA_TOUCHTEST_STATE_UP enumeration member 632
LA_TRUE enumeration member 570
LA_WIDGET_ARC enumeration member 637
LA_WIDGET_BACKGROUND_CACHE enumeration member 568
LA_WIDGET_BACKGROUND_FILL enumeration member 568
LA_WIDGET_BACKGROUND_LAST enumeration member 568
LA_WIDGET_BACKGROUND_NONE enumeration member 568
LA_WIDGET_BAR_GRAPH enumeration member 637
LA_WIDGET_BORDER_BEVEL enumeration member 571
LA_WIDGET_BORDER_LAST enumeration member 571
LA_WIDGET_BORDER_LINE enumeration member 571
LA_WIDGET_BORDER_NONE enumeration member 571
LA_WIDGET_BUTTON enumeration member 637
LA_WIDGET_CHECKBOX enumeration member 637
LA_WIDGET_CIRCLE enumeration member 637
LA_WIDGET_CIRCULAR_GAUGE enumeration member 637
LA_WIDGET_CIRCULAR_SLIDER enumeration member 637
LA_WIDGET_DIRTY_STATE_CHILD enumeration member 636
LA_WIDGET_DIRTY_STATE_CLEAN enumeration member 636
LA_WIDGET_DIRTY_STATE_DIRTY enumeration member 636
LA_WIDGET_DRAW_STATE_DONE enumeration member 636
LA_WIDGET_DRAW_STATE_READY enumeration member 636
LA_WIDGET_DRAWSURFACE enumeration member 637
LA_WIDGET_GRADIENT enumeration member 637
LA_WIDGET_GROUPBOX enumeration member 637
LA_WIDGET_IMAGE enumeration member 637
LA_WIDGET_IMAGEPLUS enumeration member 637
LA_WIDGET_IMAGESEQUENCE enumeration member 637
LA_WIDGET_KEYPAD enumeration member 637
LA_WIDGET_LABEL enumeration member 637
LA_WIDGET_LAYER enumeration member 637
LA_WIDGET_LINE enumeration member 637
LA_WIDGET_LINE_GRAPH enumeration member 637
LA_WIDGET_LIST enumeration member 637
LA_WIDGET_LISTWHEEL enumeration member 637
LA_WIDGET_OPT_DRAW_ONCE enumeration member 637
LA_WIDGET_OPT_LOCAL_REDRAW enumeration member 637
LA_WIDGET_OPT_OPAQUE enumeration member 637
LA_WIDGET_PIE_CHART enumeration member 637
LA_WIDGET_PROGRESSBAR enumeration member 637
LA_WIDGET_RADIAL_MENU enumeration member 637
LA_WIDGET_RADIOBUTTON enumeration member 637
LA_WIDGET_RECTANGLE enumeration member 637
LA_WIDGET_SCROLLBAR enumeration member 637
LA_WIDGET_SLIDER enumeration member 637
LA_WIDGET_TEXTFIELD enumeration member 637
LA_WIDGET_TOUCHTEST enumeration member 637
LA_WIDGET_UPDATE_STATE_DONE enumeration member 638
LA_WIDGET_UPDATE_STATE_PENDING enumeration member 638
LA_WIDGET_WIDGET enumeration member 637
LA_WIDGET_WINDOW enumeration member 637
laArcWidget_GetCenterAngle function 238
laArcWidget_GetRadius function 238
laArcWidget_GetRoundEdge function 239
laArcWidget_GetStartAngle function 239
laArcWidget_GetThickness function 239
laArcWidget_New function 240
laArcWidget_SetCenterAngle function 240
laArcWidget_SetRadius function 240
laArcWidget_SetRoundEdge function 241
laArcWidget_SetStartAngle function 241
laArcWidget_SetThickness function 242
laArcWidget_t structure 567
laBackgroundType enumeration 568
laBackgroundType_t enumeration 568
laBarGraphCategory_t structure 568
laBarGraphDataSeries_t structure 569
laBarGraphTickPosition_t enumeration 569
laBarGraphValueAxis_t enumeration 569
laBarGraphWidget_AddCategory function 242
laBarGraphWidget_AddDataToSeries function 243
laBarGraphWidget_AddSeries function 243
laBarGraphWidget_DestroyAll function 243
laBarGraphWidget_GetCategoryAxisLabelsVisible function 244
laBarGraphWidget_GetCategoryAxisTicksPosition function 244
laBarGraphWidget_GetCategoryAxisTicksVisible function 245
laBarGraphWidget_GetCategoryText function 245
laBarGraphWidget_GetFillGraphArea function 245
laBarGraphWidget_GetGridlinesVisible function 246
laBarGraphWidget_GetMaxValue function 246
laBarGraphWidget_GetMinValue function 247
laBarGraphWidget_GetSeriesScheme function 247
laBarGraphWidget_GetStacked function 247
laBarGraphWidget_GetTickLength function 248
laBarGraphWidget_GetValueAxisLabelsVisible function 248
laBarGraphWidget_GetValueAxisSubtickInterval function 249
laBarGraphWidget_GetValueAxisSubticksPosition function 249
laBarGraphWidget_GetValueAxisSubticksVisible function 249
laBarGraphWidget_GetValueAxisTickInterval function 250
laBarGraphWidget_GetValueAxisTicksPosition function 250
laBarGraphWidget_GetValueAxisTicksVisible function 251
laBarGraphWidget_New function 251
laBarGraphWidget_SetCategoryAxisLabelsVisible function 251
laBarGraphWidget_SetCategoryAxisTicksPosition function 252

laBarGraphWidget_SetCategoryAxisTicksVisible function 252
 laBarGraphWidget_SetCategoryText function 253
 laBarGraphWidget_SetDataInSeries function 253
 laBarGraphWidget_SetFillGraphArea function 254
 laBarGraphWidget_SetGridlinesVisible function 254
 laBarGraphWidget_SetMaxValue function 254
 laBarGraphWidget_SetMinValue function 255
 laBarGraphWidget_SetSeriesScheme function 255
 laBarGraphWidget_SetStacked function 256
 laBarGraphWidget_SetStringTable function 256
 laBarGraphWidget_SetTickLength function 256
 laBarGraphWidget_SetTicksLabelsStringID function 257
 laBarGraphWidget_SetValueAxisLabelsVisible function 257
 laBarGraphWidget_SetValueAxisSubtickInterval function 258
 laBarGraphWidget_SetValueAxisSubticksPosition function 258
 laBarGraphWidget_SetValueAxisSubticksVisible function 259
 laBarGraphWidget_SetValueAxisTickInterval function 259
 laBarGraphWidget_SetValueAxisTicksPosition function 259
 laBarGraphWidget_SetValueAxisTicksVisible function 260
 laBarGraphWidget_t structure 570
 laBool enumeration 570
 laBool_t enumeration 570
 laBorderType enumeration 571
 laBorderType_t enumeration 571
 laButtonState enumeration 571
 laButtonState_t enumeration 571
 laButtonWidget type 571
 laButtonWidget_GetHorizontalAlignment function 260
 laButtonWidget_GetImageMargin function 261
 laButtonWidget_GetImagePosition function 261
 laButtonWidget_GetPressed function 261
 laButtonWidget_GetPressedEventCallback function 262
 laButtonWidget_GetPressedImage function 262
 laButtonWidget_GetPressedOffset function 263
 laButtonWidget_GetReleasedEventCallback function 263
 laButtonWidget_GetReleasedImage function 263
 laButtonWidget_GetText function 264
 laButtonWidget_GetTextLineSpace function 264
 laButtonWidget_GetToggleable function 265
 laButtonWidget_SetVAlignment function 265
 laButtonWidget_New function 265
 laButtonWidget_PressedEvent type 572
 laButtonWidget_ReleasedEvent type 572
 laButtonWidget_SetHorizontalAlignment function 266
 laButtonWidget_SetImageMargin function 266
 laButtonWidget_SetImagePosition function 267
 laButtonWidget_SetPressed function 267
 laButtonWidget_SetPressedEventCallback function 267
 laButtonWidget_SetPressedImage function 268
 laButtonWidget_SetPressedOffset function 268
 laButtonWidget_SetReleasedEventCallback function 269
 laButtonWidget_SetReleasedImage function 269
 laButtonWidget_SetText function 270
 laButtonWidget_SetTextLineSpace function 270
 laButtonWidget_SetToggleable function 270
 laButtonWidget_SetVAlignment function 271
 laButtonWidget_t structure 572
 laCheckBoxWidget structure 573
 laCheckBoxWidget_CheckedEvent type 574
 laCheckBoxWidget_GetChecked function 271
 laCheckBoxWidget_GetCheckedEventCallback function 272
 laCheckBoxWidget_GetCheckedImage function 272
 laCheckBoxWidget_SetHAlignment function 272
 laCheckBoxWidget_SetImageMargin function 273
 laCheckBoxWidget_SetImagePosition function 273
 laCheckBoxWidget_SetText function 273
 laCheckBoxWidget_SetUncheckedEventCallback function 274
 laCheckBoxWidget_SetUncheckedImage function 274
 laCheckBoxWidget_SetVAlignment function 275
 laCheckBoxWidget_New function 275
 laCheckBoxWidget_SetChecked function 275
 laCheckBoxWidget_SetCheckedEventCallback function 276
 laCheckBoxWidget_SetCheckedImage function 276
 laCheckBoxWidget_SetHAlignment function 277
 laCheckBoxWidget_SetImageMargin function 277
 laCheckBoxWidget_SetImagePosition function 277
 laCheckBoxWidget_SetText function 278
 laCheckBoxWidget_SetUncheckedEventCallback function 278
 laCheckBoxWidget_SetUncheckedImage function 279
 laCheckBoxWidget_SetVAlignment function 279
 laCheckBoxWidget_t structure 573
 laCheckBoxWidget_UncheckedEvent type 574
 laCircleWidget structure 575
 laCircleWidget_GetFilled function 280
 laCircleWidget_GetOrigin function 280
 laCircleWidget_GetRadius function 280
 laCircleWidget_GetThickness function 281
 laCircleWidget_New function 281
 laCircleWidget_SetFilled function 281
 laCircleWidget_SetOrigin function 282
 laCircleWidget_SetRadius function 282
 laCircleWidget_SetThickness function 283
 laCircleWidget_t structure 575
 laCircularGaugeArc_t structure 575
 laCircularGaugeLabel_t structure 576
 laCircularGaugeTick_t structure 576
 laCircularGaugeWidget_AddAngularArc function 283
 laCircularGaugeWidget_AddMinorTickLabels function 284
 laCircularGaugeWidget_AddMinorTicks function 284
 laCircularGaugeWidget_AddValueArc function 285
 laCircularGaugeWidget_DeleteArcs function 286
 laCircularGaugeWidget_DeleteMinorTickLabels function 286
 laCircularGaugeWidget_DeleteMinorTicks function 286
 laCircularGaugeWidget_SetCenterAngle function 287
 laCircularGaugeWidget_SetCenterCircleRadius function 287
 laCircularGaugeWidget_SetCenterCircleThickness function 288
 laCircularGaugeWidget_SetCenterCircleVisible function 288
 laCircularGaugeWidget_SetDirection function 288
 laCircularGaugeWidget_SetEndValue function 289
 laCircularGaugeWidget_SetHandRadius function 289
 laCircularGaugeWidget_SetHandVisible function 290
 laCircularGaugeWidget_SetRadius function 290
 laCircularGaugeWidget_SetStartAngle function 290
 laCircularGaugeWidget_SetStartValue function 291
 laCircularGaugeWidget_SetTickLabelsVisible function 291
 laCircularGaugeWidget_SetTickLength function 291

laCircularGaugeWidget_GetTicksVisible function 292
laCircularGaugeWidget_GetTickValue function 292
laCircularGaugeWidget_GetValue function 293
laCircularGaugeWidget_New function 293
laCircularGaugeWidget_SetCenterAngle function 293
laCircularGaugeWidget_SetCenterCircleRadius function 294
laCircularGaugeWidget_SetCenterCircleThickness function 294
laCircularGaugeWidget_SetCenterCircleVisible function 295
laCircularGaugeWidget_SetEndValue function 295
laCircularGaugeWidget_SetHandRadius function 295
laCircularGaugeWidget_SetHandVisible function 296
laCircularGaugeWidget_SetRadius function 296
laCircularGaugeWidget_SetStartAngle function 297
laCircularGaugeWidget_SetStartValue function 297
laCircularGaugeWidget_SetStringTable function 298
laCircularGaugeWidget_SetTickLabelsVisible function 298
laCircularGaugeWidget_SetTickLength function 298
laCircularGaugeWidget_SetTicksLabelsStringID function 299
laCircularGaugeWidget_SetTicksVisible function 299
laCircularGaugeWidget_SetTickValue function 300
laCircularGaugeWidget_SetValue function 300
laCircularGaugeWidget_SetValueChangedEventCallback function 301
laCircularGaugeWidget_t structure 577
laCircularGaugeWidgetArcType_t enumeration 578
laCircularGaugeWidgetDir_t enumeration 578
laCircularGaugeWidgetLabelPosition_t enumeration 578
laCircularSliderArc_t structure 579
laCircularSliderButtonState_t enumeration 579
laCircularSliderWidget_GetArcRadius function 301
laCircularSliderWidget_GetArcScheme function 301
laCircularSliderWidget_GetArcThickness function 302
laCircularSliderWidget_GetArcVisible function 302
laCircularSliderWidget_GetDirection function 303
laCircularSliderWidget_GetEndValue function 303
laCircularSliderWidget_GetOrigin function 303
laCircularSliderWidget_GetRadius function 304
laCircularSliderWidget_GetRoundEdges function 304
laCircularSliderWidget_GetStartAngle function 305
laCircularSliderWidget_GetStartValue function 305
laCircularSliderWidget_GetStickyButton function 305
laCircularSliderWidget_GetTouchOnButtonOnly function 306
laCircularSliderWidget_GetValue function 306
laCircularSliderWidget_New function 306
laCircularSliderWidget_SetArcRadius function 307
laCircularSliderWidget_SetArcScheme function 307
laCircularSliderWidget_SetArcThickness function 308
laCircularSliderWidget_SetArcVisible function 308
laCircularSliderWidget_SetDirection function 309
laCircularSliderWidget_SetEndValue function 309
laCircularSliderWidget_SetOrigin function 309
laCircularSliderWidget_SetPressedEventCallback function 310
laCircularSliderWidget_SetRadius function 310
laCircularSliderWidget_SetReleasedEventCallback function 311
laCircularSliderWidget_SetRoundEdges function 311
laCircularSliderWidget_SetStartAngle function 311
laCircularSliderWidget_SetStartValue function 312
laCircularSliderWidget_SetStickyButton function 312
laCircularSliderWidget_SetTouchOnButtonOnly function 313
laCircularSliderWidget_SetValue function 313
laCircularSliderWidget_SetValueChangedEventCallback function 314
laCircularSliderWidget_t structure 580
laCircularSliderWidgetArcType_t enumeration 581
laCircularSliderWidgetDir_t enumeration 581
laContext type 581
laContext_ActiveScreenChangedCallback_FnPtr type 582
laContext_AddScreen function 314
laContext_Create function 314
laContext_Destroy function 315
laContext_GetActive function 315
laContext_GetActiveScreen function 316
laContext_GetActiveScreenIndex function 316
laContext_GetColorMode function 316
laContext_GetDefaultScheme function 317
laContext_GetEditWidget function 317
laContext_GetFocusWidget function 317
laContext_GetPreemptionLevel function 318
laContext_GetScreenRect function 318
laContext_GetStringLanguage function 318
laContext_GetStringTable function 319
laContext_HideActiveScreen function 319
laContext_IsDrawing function 319
laContext_IsLayerDrawing function 320
laContext_LanguageChangedCallback_FnPtr type 582
laContext_RedrawAll function 320
laContext_RemoveScreen function 321
laContext_SetActive function 321
laContext_SetActiveScreen function 321
laContext_SetActiveScreenChangedCallback function 322
laContext_SetEditWidget function 322
laContext_SetFocusWidget function 322
laContext_SetLanguageChangedCallback function 323
laContext_SetPreemptionLevel function 323
laContext_SetStringLanguage function 323
laContext_SetStringTable function 324
laContext_t structure 582
laContext_Update function 324
laContextFrameState enumeration 583
laContextFrameState_t enumeration 583
laContextUpdateState enumeration 584
laContextUpdateState_t enumeration 584
laDraw_1x2BevelBorder function 325
laDraw_2x1BevelBorder function 325
laDraw_2x2BevelBorder function 326
laDraw_LineBorder function 326
laDrawSurfaceWidget structure 584
laDrawSurfaceWidget_DrawCallback type 585
laDrawSurfaceWidget_GetDrawCallback function 326
laDrawSurfaceWidget_New function 327
laDrawSurfaceWidget_SetDrawCallback function 327
laDrawSurfaceWidget_t structure 584
laEditWidget structure 585
laEditWidget_t structure 585
laEvent structure 585
laEvent_AddEvent function 328
laEvent_ClearList function 328
laEvent_FilterEvent type 586

laEvent_GetCount function 328
laEvent_ProcessEvents function 329
laEvent_SetFilter function 329
laEvent_t structure 585
laEventID enumeration 586
laEventID_t enumeration 586
laEventResult enumeration 586
laEventResult_t enumeration 586
laEventState structure 587
laEventState_t structure 587
laGestureID enumeration 587
laGestureID_t enumeration 587
laGradientWidget structure 587
laGradientWidget_GetDirection function 330
laGradientWidget_New function 330
laGradientWidget_SetDirection function 330
laGradientWidget_t structure 587
laGradientWidgetDirection enumeration 588
laGradientWidgetDirection_t enumeration 588
laGroupBoxWidget structure 588
laGroupBoxWidget_GetAlignment function 331
laGroupBoxWidget_GetText function 331
laGroupBoxWidget_New function 332
laGroupBoxWidget_SetAlignment function 332
laGroupBoxWidget_SetText function 332
laGroupBoxWidget_t structure 588
laHAlignment enumeration 589
laImagePlusWidget_GetImage function 333
laImagePlusWidget_GetInteractive function 333
laImagePlusWidget_GetPreserveAspectEnabled function 334
laImagePlusWidget_GetResizeFilter function 334
laImagePlusWidget_GetStretchEnabled function 335
laImagePlusWidget_GetTransformHeight function 335
laImagePlusWidget_GetTransformWidth function 335
laImagePlusWidget_GetTransformX function 336
laImagePlusWidget_GetTransformY function 336
laImagePlusWidget_New function 337
laImagePlusWidget_ResetTransform function 337
laImagePlusWidget_ResizeFilter_t enumeration 589
laImagePlusWidget_SetImage function 337
laImagePlusWidget_SetInteractive function 338
laImagePlusWidget_SetPreserveAspectEnabled function 338
laImagePlusWidget_SetResizeFilter function 339
laImagePlusWidget_SetStretchEnabled function 339
laImagePlusWidget_SetTransformHeight function 340
laImagePlusWidget_SetTransformWidth function 340
laImagePlusWidget_SetTransformX function 341
laImagePlusWidget_SetTransformY function 341
laImagePlusWidget_t structure 589
laImageSequenceEntry structure 590
laImageSequenceEntry_t structure 590
laImageSequenceImageChangedEvent_FnPtr type 591
laImageSequenceWidget structure 591
laImageSequenceWidget_GetImage function 342
laImageSequenceWidget_GetImageChangedEventCallback function 342
laImageSequenceWidget_GetImageCount function 342
laImageSequenceWidget_GetImageDelay function 343
laImageSequenceWidget_GetImageHAlignment function 343
laImageSequenceWidget_GetImageVAlignment function 344
laImageSequenceWidget_GetRepeat function 344
laImageSequenceWidget_IsPlaying function 344
laImageSequenceWidget_New function 345
laImageSequenceWidget_Play function 345
laImageSequenceWidget_Rewind function 346
laImageSequenceWidget_SetImage function 346
laImageSequenceWidget_SetImageChangedEventCallback function 346
laImageSequenceWidget_SetImageCount function 347
laImageSequenceWidget_SetImageDelay function 347
laImageSequenceWidget_SetImageHAlignment function 348
laImageSequenceWidget_SetImageVAlignment function 348
laImageSequenceWidget_SetRepeat function 349
laImageSequenceWidget_ShowImage function 349
laImageSequenceWidget_ShowNextImage function 350
laImageSequenceWidget_ShowPreviousImage function 350
laImageSequenceWidget_Stop function 350
laImageSequenceWidget_t structure 591
laImageWidget structure 592
laImageWidget_DrawEventCallback type 592
laImageWidget_GetHAlignment function 351
laImageWidget_GetImage function 351
laImageWidget_GetVAlignment function 351
laImageWidget_New function 352
laImageWidget_SetHAlignment function 352
laImageWidget_SetImage function 353
laImageWidget_SetVAlignment function 353
laImageWidget_t structure 592
laInitialize function 353
laInput_GetEnabled function 354
laInput.InjectTouchDown function 354
laInput.InjectTouchMoved function 354
laInput.InjectTouchUp function 355
laInput_SetEnabled function 355
laInput_TouchDownEvent structure 593
laInput_TouchDownEvent_t structure 593
laInput_TouchMovedEvent structure 593
laInput_TouchMovedEvent_t structure 593
laInput_TouchUpEvent structure 593
laInput_TouchUpEvent_t structure 593
laInputState structure 594
laInputState_t structure 594
laKey enumeration 594
laKey_t enumeration 594
laKeyPadActionTrigger enumeration 596
laKeyPadActionTrigger_t enumeration 596
laKeyPadCell structure 596
laKeyPadCell_t structure 596
laKeyPadCellAction enumeration 597
laKeyPadCellAction_t enumeration 597
laKeyPadWidget structure 597
laKeyPadWidget.GetKeyAction function 355
laKeyPadWidget.GetKeyClickEventCallback function 356
laKeyPadWidget.GetKeyDrawBackground function 356
laKeyPadWidget.GetKeyEnabled function 357
laKeyPadWidget.GetKeyImageMargin function 357
laKeyPadWidget.GetKeyImagePosition function 358
laKeyPadWidget.GetKeyPadActionTrigger function 358

laKeyPadWidget_GetKeyPressedImage function 358
laKeyPadWidget_GetKeyReleasedImage function 359
laKeyPadWidget_GetKeyText function 359
laKeyPadWidget_GetKeyValue function 360
laKeyPadWidget_KeyClickEvent type 598
laKeyPadWidget_New function 360
laKeyPadWidget_SetKeyAction function 361
laKeyPadWidget_SetKeyBackgroundType function 361
laKeyPadWidget_SetKeyClickEventCallback function 362
laKeyPadWidget_SetKeyEnabled function 362
laKeyPadWidget_SetKeyImageMargin function 363
laKeyPadWidget_SetKeyImagePosition function 363
laKeyPadWidget_SetKeyPadActionTrigger function 599
laKeyPadWidget_SetKeyPressedImage function 364
laKeyPadWidget_SetKeyReleasedImage function 364
laKeyPadWidget_SetKeyText function 365
laKeyPadWidget_SetKeyValue function 366
laKeyPadWidget_t structure 597
laLabelWidget structure 599
laLabelWidget_GetHAlignment function 366
laLabelWidget_GetText function 367
laLabelWidget_GetTextLineSpace function 367
laLabelWidget_GetVAlignment function 367
laLabelWidget_New function 368
laLabelWidget_SetHAlignment function 368
laLabelWidget_SetText function 369
laLabelWidget_SetTextLineSpace function 369
laLabelWidget_SetVAlignment function 369
laLabelWidget_t structure 599
laLayer type 600
laLayer_AddDamageRect function 370
laLayer_Delete function 370
laLayer_GetAllowInputPassThrough function 371
laLayer_GetAlphaAmount function 371
laLayer_GetAlphaEnable function 371
laLayer_GetBufferCount function 372
laLayer_GetEnabled function 372
laLayer_GetInputRect function 373
laLayer_GetInputRectLocked function 373
laLayer_GetMaskColor function 374
laLayer_SetMaskEnable function 374
laLayer_SetVSync function 374
laLayer_IsDrawing function 375
laLayer_New function 375
laLayer_SetAllowInputPassthrough function 376
laLayer_SetAlphaAmount function 376
laLayer_SetAlphaEnable function 376
laLayer_SetBufferCount function 377
laLayer_SetEnabled function 377
laLayer_SetInputRect function 378
laLayer_SetInputRectLocked function 378
laLayer_SetMaskColor function 379
laLayer_SetMaskEnable function 379
laLayer_SetVSync function 380
laLayer_t structure 600
laLayerBuffer structure 601
laLayerBuffer_t structure 601
laLayerButtonType enumeration 602
laLayerFrameState enumeration 602
laLayerFrameState_t enumeration 602
laLineGraphCategory_t structure 602
laLineGraphDataPointType_t enumeration 603
laLineGraphDataSeries_t structure 603
laLineGraphTickPosition_t enumeration 603
laLineGraphValueAxis_t enumeration 604
laLineGraphWidget_AddCategory function 380
laLineGraphWidget_AddDataToSeries function 380
laLineGraphWidget_AddSeries function 381
laLineGraphWidget_DestroyAll function 381
laLineGraphWidget_GetCategoryAxisLabelsVisible function 382
laLineGraphWidget_GetCategoryAxisTicksPosition function 382
laLineGraphWidget_GetCategoryAxisTicksVisible function 382
laLineGraphWidget_GetCategoryText function 383
laLineGraphWidget_GetFillGraphArea function 383
laLineGraphWidget_GetFillSeriesArea function 384
laLineGraphWidget_GetGridlinesVisible function 384
laLineGraphWidget_GetMaxValue function 384
laLineGraphWidget_GetMinValue function 385
laLineGraphWidget_GetSeriesFillPoints function 385
laLineGraphWidget_GetSeriesLinesVisible function 386
laLineGraphWidget_GetSeriesPointSize function 386
laLineGraphWidget_GetSeriesPointType function 387
laLineGraphWidget_GetSeriesScheme function 387
laLineGraphWidget_GetStacked function 387
laLineGraphWidget_GetTickLength function 388
laLineGraphWidget_GetValueAxisLabelsVisible function 388
laLineGraphWidget_GetValueAxisSubtickInterval function 389
laLineGraphWidget_GetValueAxisSubticksPosition function 389
laLineGraphWidget_GetValueAxisSubticksVisible function 389
laLineGraphWidget_GetValueAxisTickInterval function 390
laLineGraphWidget_GetValueAxisTicksPosition function 390
laLineGraphWidget_GetValueAxisTicksVisible function 391
laLineGraphWidget_New function 391
laLineGraphWidget_SetCategoryAxisLabelsVisible function 391
laLineGraphWidget_SetCategoryAxisTicksPosition function 392
laLineGraphWidget_SetCategoryAxisTicksVisible function 392
laLineGraphWidget_SetCategoryText function 393
laLineGraphWidget_SetDataInSeries function 393
laLineGraphWidget_SetFillGraphArea function 394
laLineGraphWidget_SetFillSeriesArea function 394
laLineGraphWidget_SetGridlinesVisible function 394
laLineGraphWidget_SetMaxValue function 395
laLineGraphWidget_SetMinValue function 395
laLineGraphWidget_SetSeriesFillPoints function 396
laLineGraphWidget_SetSeriesLinesVisible function 396
laLineGraphWidget_SetSeriesPointSize function 397
laLineGraphWidget_SetSeriesPointType function 397
laLineGraphWidget_SetSeriesScheme function 397
laLineGraphWidget_SetStacked function 398
laLineGraphWidget_SetStringTable function 398
laLineGraphWidget_SetTickLength function 399
laLineGraphWidget_SetTicksLabelsStringID function 399
laLineGraphWidget_SetValueAxisLabelsVisible function 400
laLineGraphWidget_SetValueAxisSubtickInterval function 400
laLineGraphWidget_SetValueAxisSubticksPosition function 400

IaLineGraphWidget_SetValueAxisSubticksVisible function 401
IaLineGraphWidget_SetValueAxisTickInterval function 401
IaLineGraphWidget_SetValueAxisTicksPosition function 402
IaLineGraphWidget_SetValueAxisTicksVisible function 402
IaLineGraphWidget_t structure 604
IaLineWidget structure 605
IaLineWidget_GetEndPoint function 403
IaLineWidget_GetStartPoint function 403
IaLineWidget_New function 403
IaLineWidget_SetEndPoint function 404
IaLineWidget_SetStartPoint function 404
IaLineWidget_t structure 605
IaList type 605
IaList_Assign function 405
IaList_Clear function 405
IaList_Copy function 405
IaList_Create function 406
IaList_Destroy function 406
IaList_Find function 407
IaList_Get function 407
IaList_InsertAt function 407
IaList_PopBack function 408
IaList_PopFront function 408
IaList_PushBack function 408
IaList_PushFront function 409
IaList_Remove function 409
IaList_RemoveAt function 410
IaList_t structure 606
IaListItem structure 606
IaListItem_t structure 606
IaListNode structure 607
IaListNode_t structure 607
IaListWheelIndicatorFill enumeration 607
IaListWheelIndicatorFill_t enumeration 607
IaListWheelItem structure 607
IaListWheelItem_t structure 607
IaListWheelWidget structure 608
IaListWheelWidget_AppendItem function 410
IaListWheelWidget_GetAlignment function 410
IaListWheelWidget_GetAutoHideWheel function 609
IaListWheelWidget_GetFlickInitSpeed function 411
IaListWheelWidget_SetIconMargin function 411
IaListWheelWidget_SetIconPosition function 412
IaListWheelWidget_SetIndicatorArea function 412
IaListWheelWidget_SetIndicatorFill function 610
IaListWheelWidget_SetItemCount function 412
IaListWheelWidget_SetItemIcon function 413
IaListWheelWidget_SetItemText function 413
IaListWheelWidget_SetMaxMomentum function 414
IaListWheelWidget_SetMomentumFalloffRate function 414
IaListWheelWidget_SetRotationUpdateRate function 414
IaListWheelWidget_SetSelectedItem function 415
IaListWheelWidget_SetSelectedItemChangedEventCallback function 415
IaListWheelWidget_SetShaded function 416
IaListWheelWidget_SetShowIndicators function 416
IaListWheelWidget_SetVisibleItemCount function 416
IaListWheelWidget_SetZoomEffects function 610
IaListWheelWidget_InsertItem function 417
IaListWheelWidget_New function 417
IaListWheelWidget_RemoveAllItems function 417
IaListWheelWidget_RemoveItem function 418
IaListWheelWidget_SetSelectedEventArgs type 610
IaListWheelWidget_SelectNextItem function 418
IaListWheelWidget_SelectPreviousItem function 419
IaListWheelWidget_SetAlignment function 419
IaListWheelWidget_SetAutoHideWheel function 611
IaListWheelWidget_SetFlickInitSpeed function 420
IaListWheelWidget_SetIconMargin function 420
IaListWheelWidget_SetIconPosition function 421
IaListWheelWidget_SetIndicatorArea function 421
IaListWheelWidget_SetIndicatorFill function 611
IaListWheelWidget_SetItemIcon function 421
IaListWheelWidget_SetItemText function 422
IaListWheelWidget_SetMaxMomentum function 422
IaListWheelWidget_SetMomentumFalloffRate function 423
IaListWheelWidget_SetRotationUpdateRate function 423
IaListWheelWidget_SetSelectedItem function 424
IaListWheelWidget_SetSelectedItemChangedEventArgs function 424
IaListWheelWidget_SetShaded function 425
IaListWheelWidget_SetShowIndicators function 425
IaListWheelWidget_SetVisibleItemCount function 426
IaListWheelWidget_SetZoomEffects function 612
IaListWheelWidget_t structure 608
IaListWheelZoomEffects enumeration 612
IaListWheelZoomEffects_t enumeration 612
IaListWidget structure 612
IaListWidget_AppendItem function 426
IaListWidget_DeselectAll function 427
IaListWidget_GetAlignment function 427
IaListWidget_GetAllowEmptySelection function 427
IaListWidget_GetFirstSelectedItem function 428
IaListWidget_SetIconMargin function 428
IaListWidget_SetIconPosition function 428
IaListWidget_SetItemCount function 429
IaListWidget_SetItemEnable function 429
IaListWidget_SetItemIcon function 430
IaListWidget_SetItemSelected function 430
IaListWidget_SetItemText function 430
IaListWidget_SetLastSelectedItem function 431
IaListWidget_SetSelectedItemChangedEventArgs function 431
IaListWidget_SetSelectionCount function 432
IaListWidget_SetSelectionMode function 432
IaListWidget_InsertItem function 432
IaListWidget_New function 433
IaListWidget_RemoveAllItems function 433
IaListWidget_RemoveItem function 434
IaListWidget_SelectAll function 434
IaListWidget_SetSelectedItemEventArgs type 613
IaListWidget_SetSelectionMode enumeration 613
IaListWidget_SetSelectionMode_t enumeration 613
IaListWidget_SetAlignment function 434
IaListWidget_SetAllowEmptySelection function 435
IaListWidget_SetIconMargin function 435
IaListWidget_SetIconPosition function 436
IaListWidget_SetItemEnable function 436
IaListWidget_SetItemIcon function 437

laListWidget_SetItemSelected function 437
laListWidget_SetItemText function 438
laListWidget_SetItemVisible function 438
laListWidget_SetSelectedItemChangedEventCallback function 438
laListWidget_SetSelectionMode function 439
laListWidget_t structure 612
laListWidget_ToggleItemSelected function 439
laMargin structure 614
laMargin_t structure 614
laMouseButton enumeration 614
laMouseButton_t enumeration 614
laPieChartPie_t structure 615
laPieChartWidget_AddEntry function 440
laPieChartWidget_DeleteEntries function 440
laPieChartWidget_GetCenterAngle function 441
laPieChartWidget_GetEntryOffset function 441
laPieChartWidget_GetEntryRadius function 441
laPieChartWidget_GetEntryScheme function 442
laPieChartWidget_GetEntryValue function 442
laPieChartWidget_GetLabelsOffset function 443
laPieChartWidget_GetLabelsVisible function 443
laPieChartWidget_GetOrigin function 443
laPieChartWidget_GetStartAngle function 444
laPieChartWidget_New function 444
laPieChartWidget_SetCenterAngle function 444
laPieChartWidget_SetEntryOffset function 445
laPieChartWidget_SetEntryRadius function 445
laPieChartWidget_SetEntryScheme function 446
laPieChartWidget_SetEntryValue function 446
laPieChartWidget_SetLabelsOffset function 447
laPieChartWidget_SetLabelsStringID function 447
laPieChartWidget_SetLabelsVisible function 447
laPieChartWidget_SetOrigin function 448
laPieChartWidget_SetPressedEventCallback function 448
laPieChartWidget_SetStartAngle function 449
laPieChartWidget_SetStringTable function 449
laPieChartWidget_t structure 615
laPreemptionLevel enumeration 616
laProgressBar_ValueChangedEventCallback type 616
laProgressBarDirection enumeration 616
laProgressBarDirection_t enumeration 616
laProgressBarWidget structure 617
laProgressBarWidget_GetDirection function 449
laProgressBarWidget_GetValue function 450
laProgressBarWidget_SetValueChangedEventCallback function 450
laProgressBarWidget_New function 451
laProgressBarWidget_SetDirection function 451
laProgressBarWidget_SetValue function 451
laProgressBarWidget_SetValueChangedCallback function 452
laProgressBarWidget_t structure 617
laRadialMenuEllipseType_t enumeration 617
laRadialMenuItemNode_t structure 618
laRadialMenuWidget_AddWidget function 452
laRadialMenuWidget_ClearItems function 453
laRadialMenuWidget_SetItemProminenceChangedEventCallback function 453
laRadialMenuWidget_SetItemSelectedEventCallback function 453
laRadialMenuWidget_SetProminentIndex function 454
laRadialMenuWidget_GetTheta function 454
laRadialMenuWidget_GetWidget function 455
laRadialMenuWidget_IsProminent function 455
laRadialMenuWidget_New function 456
laRadialMenuWidget_RemoveWidget function 456
laRadialMenuWidget_SetAlphaScaleMinMax function 456
laRadialMenuWidget_SetAlphaScaling function 457
laRadialMenuWidget_SetDrawEllipse function 457
laRadialMenuWidget_SetEllipseType function 458
laRadialMenuWidget_SetHighlightProminent function 458
laRadialMenuWidget_SetItemProminenceChangedEventCallback function 459
laRadialMenuWidget_SetItemSelectedEventCallback function 459
laRadialMenuWidget_SetNumberOfItemsShown function 460
laRadialMenuWidget_SetProminent function 460
laRadialMenuWidget_SetProminentIndex function 461
laRadialMenuWidget_SetSizeScaleMinMax function 461
laRadialMenuWidget_SetSizeScaling function 462
laRadialMenuWidget_SetTheta function 462
laRadialMenuWidget_SetTouchArea function 463
laRadialMenuWidget_SetWidgetAt function 463
laRadialMenuWidget_t structure 618
laRadialMenuWidgetScaleType_t enumeration 620
laRadialMenuWidgetState_t enumeration 620
laRadioButtonGroup type 620
laRadioButtonGroup_AddButton function 464
laRadioButtonGroup_Create function 464
laRadioButtonGroup_Destroy function 465
laRadioButtonGroup_RemoveButton function 465
laRadioButtonGroup_SelectButton function 465
laRadioButtonGroup_t structure 621
laRadioButtonWidget structure 621
laRadioButtonWidget_DeselectedEvent type 622
laRadioButtonWidget_GetCircleButtonSize function 466
laRadioButtonWidget_GetDeselectedEventCallback function 466
laRadioButtonWidget_GetGroup function 467
laRadioButtonWidget_GetHAlignment function 467
laRadioButtonWidget_GetImageMargin function 467
laRadioButtonWidget_GetImagePosition function 468
laRadioButtonWidget_GetSelected function 468
laRadioButtonWidget_GetSelectedEventCallback function 468
laRadioButtonWidget_GetSelectedImage function 469
laRadioButtonWidget_GetText function 469
laRadioButtonWidget_GetUnselectedImage function 470
laRadioButtonWidget_GetVAlignment function 470
laRadioButtonWidget_New function 470
laRadioButtonWidget_SelectedEvent type 622
laRadioButtonWidget_SetCircleButtonSize function 471
laRadioButtonWidget_SetDeselectedEventCallback function 471
laRadioButtonWidget_SetHAlignment function 472
laRadioButtonWidget_SetImageMargin function 472
laRadioButtonWidget_SetImagePosition function 473
laRadioButtonWidget_SetSelected function 473
laRadioButtonWidget_SetSelectedEventCallback function 473
laRadioButtonWidget_SetSelectedImage function 474
laRadioButtonWidget_SetText function 474
laRadioButtonWidget_SetUnselectedImage function 475
laRadioButtonWidget_SetVAlignment function 475

laRadioButtonWidget_t structure 621
laRectangleWidget structure 623
laRectangleWidget_GetThickness function 476
laRectangleWidget_New function 476
laRectangleWidget_SetThickness function 476
laRectangleWidget_t structure 623
laRectArray type 623
laRectArray_Clear function 477
laRectArray_Copy function 477
laRectArray_Create function 478
laRectArray_Destroy function 478
laRectArray_InsertAt function 478
laRectArray_MergeSimilar function 479
laRectArray_PopBack function 479
laRectArray_PopFront function 479
laRectArray_PushBack function 480
laRectArray_PushFront function 480
laRectArray_RemoveAt function 481
laRectArray_RemoveDuplicates function 481
laRectArray_RemoveOverlapping function 481
laRectArray_Resize function 482
laRectArray_SortBySize function 482
laRectArray_t structure 623
laRelativePosition enumeration 624
laResult enumeration 624
laResult_t enumeration 624
laScheme structure 625
laScheme_Initialize function 483
laScheme_t structure 625
laScreen structure 625
laScreen_CreateCallback_FnPtr type 626
laScreen_Delete function 483
laScreen_GetHideEventCallback function 483
laScreen_GetLayerIndex function 484
laScreen_GetLayerSwapSync function 484
laScreen_GetMirrored function 485
laScreen_GetOrientation function 485
laScreen_GetShowEventCallback function 486
laScreen_Hide function 486
laScreen_New function 487
laScreen_SetHideEventCallback function 487
laScreen_SetLayer function 488
laScreen_SetLayerSwapSync function 488
laScreen_SetMirrored function 489
laScreen_SetOrientation function 489
laScreen_SetShowEventCallback function 490
laScreen_Show function 490
laScreen_ShowHideCallback_FnPtr type 626
laScreen_t structure 625
laScreenOrientation enumeration 627
laScreenOrientation_t enumeration 627
laScrollBarOrientation enumeration 627
laScrollBarOrientation_t enumeration 627
laScrollBarState enumeration 627
laScrollBarState_t enumeration 627
laScrollBarWidget structure 628
laScrollBarWidget_GetExtentValue function 491
laScrollBarWidget_GetMaximumValue function 491
laScrollBarWidget_GetOrientation function 491
laScrollBarWidget_GetScrollPercentage function 492
laScrollBarWidget_GetScrollValue function 492
laScrollBarWidget_GetStepSize function 493
laScrollBarWidget_GetValueChangedEventCallback function 493
laScrollBarWidget_New function 493
laScrollBarWidget_SetExtentValue function 494
laScrollBarWidget_SetMaximumValue function 494
laScrollBarWidget_SetOrientation function 495
laScrollBarWidget_SetScrollPercentage function 495
laScrollBarWidget_SetScrollValue function 495
laScrollBarWidget_SetStepSize function 496
laScrollBarWidget_SetValueChangedEventCallback function 496
laScrollBarWidget_StepBackward function 497
laScrollBarWidget_StepForward function 497
laScrollBarWidget_t structure 628
laScrollBarWidget_ValueChangedEvent type 628
laShutdown function 497
laSliderOrientation enumeration 629
laSliderOrientation_t enumeration 629
laSliderState enumeration 629
laSliderState_t enumeration 629
laSliderWidget structure 629
laSliderWidget_GetGripSize function 498
laSliderWidget_GetMaximumValue function 498
laSliderWidget_GetMinimumValue function 498
laSliderWidget_GetOrientation function 499
laSliderWidget_GetSliderPercentage function 499
laSliderWidget_GetSliderValue function 500
laSliderWidget_GetValueChangedEventCallback function 500
laSliderWidget_New function 500
laSliderWidget_SetGripSize function 501
laSliderWidget_SetMaximumValue function 501
laSliderWidget_SetMinimumValue function 502
laSliderWidget_SetOrientation function 502
laSliderWidget_SetSliderPercentage function 502
laSliderWidget_SetSliderValue function 503
laSliderWidget_SetValueChangedEventCallback function 503
laSliderWidget_Step function 504
laSliderWidget_t structure 629
laSliderWidget_ValueChangedEvent type 630
laString structure 630
laString_Allocate function 504
laString_Append function 505
laString_Capacity function 505
laString_CharAt function 505
laString_Clear function 506
laString_Compare function 506
laString_CompareBuffer function 506
laString_Copy function 507
laString_CreateFromBuffer function 507
laString_CreateFromCharBuffer function 508
laString_CreateFromID function 508
laString_Delete function 509
laString_Destroy function 509
laString_Draw function 510
laString_DrawClipped function 510
laString_DrawSubStringClipped function 511

laString_ExtractFromTable function 512
laString_GetAscent function 512
laString_GetCharIndexAtPoint function 512
laString_GetCharOffset function 513
laString_GetCharWidth function 513
laString_GetHeight function 514
laString_GetLineRect function 514
laString_GetMultiLineRect function 514
laString_GetRect function 515
laString_Initialize function 515
laString_Insert function 516
laString_IsEmpty function 516
laString_Length function 517
laString_New function 517
laString_ReduceLength function 517
laString_Remove function 518
laString_Set function 518
laString_SetCapacity function 519
laString_t structure 630
laString_ToCharBuffer function 519
laTextFieldWidget structure 631
laTextFieldWidget_GetAlignment function 520
laTextFieldWidget_GetCursorDelay function 520
laTextFieldWidget_GetCursorEnabled function 520
laTextFieldWidget_GetCursorPosition function 521
laTextFieldWidget_GetFocusChangedEventCallback function 521
laTextFieldWidget_GetText function 522
laTextFieldWidget_GetTextChangedEventCallback function 522
laTextFieldWidget_New function 522
laTextFieldWidget_SetAlignment function 523
laTextFieldWidget_SetClearOnFirstEdit function 523
laTextFieldWidget_SetCursorDelay function 524
laTextFieldWidget_SetCursorEnabled function 524
laTextFieldWidget_SetCursorPosition function 525
laTextFieldWidget_SetFocusChangedEventCallback function 525
laTextFieldWidget_SetText function 525
laTextFieldWidget_SetTextChangedEventCallback function 526
laTextFieldWidget_t structure 631
laTextFieldWidget_TextChangedCallback type 632
laTouchState structure 632
laTouchState_t structure 632
laTouchTest_AddPoint function 526
laTouchTest_ClearPoints function 527
laTouchTestState enumeration 632
laTouchTestState_t enumeration 632
laTouchTestWidget structure 633
laTouchTestWidget_GetPointAddedEventCallback function 527
laTouchTestWidget_New function 528
laTouchTestWidget_PointAddedEventCallback type 633
laTouchTestWidget_SetPointAddedEventCallback function 528
laTouchTestWidget_t structure 633
laUpdate function 528
laUpdateRTOS function 529
laUtils_ArrangeRectangle function 529
laUtils_ArrangeRectangleRelative function 530
laUtils_ChildIntersectsParent function 531
laUtils_ClipRectToParent function 531
laUtils_GetLayer function 532
laUtils_GetNextHighestWidget function 532
laUtils_ListOcclusionCullTest function 533
laUtils_OcclusionCullTest function 533
laUtils_Pick function 533
laUtils_PickFromLayer function 534
laUtils_PickRect function 534
laUtils_PointScreenToLocalSpace function 535
laUtils_PointToLayerSpace function 535
laUtils_RectFromLayerSpace function 536
laUtils_RectToLayerSpace function 536
laUtils_RectToParentSpace function 537
laUtils_RectToScreenSpace function 537
laUtils_ScreenToMirroredSpace function 538
laUtils_ScreenToOrientedSpace function 538
laUtils_WidgetIsOccluded function 539
laUtils_WidgetLayerRect function 539
laUtils_WidgetLocalRect function 539
laVAlignment enumeration 634
laWidget structure 634
laWidget_AddChild function 540
laWidget_Delete function 540
laWidget_DeleteAllDescendants function 541
laWidget_GetAlphaAmount function 541
laWidget_GetAlphaEnable function 541
laWidget_GetBackgroundType function 542
laWidget_GetBorderType function 542
laWidget_GetChildAtIndex function 543
laWidget_GetChildCount function 543
laWidget_GetCornerRadius function 544
laWidget_GetCumulativeAlphaAmount function 544
laWidget_GetCumulativeAlphaEnable function 544
laWidget_GetEnabled function 545
laWidget_GetHeight function 545
laWidget_GetIndexOfChild function 546
laWidget_GetMargin function 546
laWidget_GetOptimizationFlags function 547
laWidget_GetScheme function 547
laWidget_GetVisible function 547
laWidget_GetWidth function 548
laWidgetGetX function 548
laWidgetGetY function 549
laWidget_HasFocus function 549
laWidget_Invalidate function 550
laWidget_isOpaque function 550
laWidget_New function 551
laWidget.OverrideTouchDownEvent function 551
laWidget.OverrideTouchMovedEvent function 551
laWidget.OverrideTouchUpEvent function 552
laWidget.RectToLayerSpace function 552
laWidget.RectToParentSpace function 553
laWidget.RectToScreenSpace function 553
laWidget.RemoveChild function 554
laWidget.Resize function 554
laWidget_SetAlphaAmount function 555
laWidget_SetAlphaEnable function 555
laWidget_SetBackgroundType function 555
laWidget_SetBorderType function 556

laWidget_SetCornerRadius function 556
 laWidget_SetEnabled function 557
 laWidget_SetFocus function 557
 laWidget_SetHeight function 558
 laWidget_SetMargins function 558
 laWidget_SetOptimizationFlags function 559
 laWidget_SetParent function 559
 laWidget_SetPosition function 560
 laWidget_SetScheme function 560
 laWidget_SetSize function 561
 laWidget_SetVisible function 561
 laWidget_SetWidth function 562
 laWidget_SetX function 562
 laWidget_SetY function 563
 laWidget_t structure 634
 laWidget_Translate function 563
 laWidgetDirtyState enumeration 636
 laWidgetDirtyState_t enumeration 636
 laWidgetDrawState enumeration 636
 laWidgetDrawState_t enumeration 636
 laWidgetEvent structure 636
 laWidgetEvent_t structure 636
 laWidgetOptimizationFlags enumeration 637
 laWidgetOptimizationFlags_t enumeration 637
 laWidgetType enumeration 637
 laWidgetType_t enumeration 637
 laWidgetUpdateState enumeration 638
 laWidgetUpdateState_t enumeration 638
 laWindowWidget structure 638
 laWindowWidget_GetIcon function 564
 laWindowWidget_GetIconMargin function 564
 laWindowWidget_GetTitle function 564
 laWindowWidget_New function 565
 laWindowWidget_SetIcon function 565
 laWindowWidget_SetIconMargin function 566
 laWindowWidget_SetTitle function 566
 laWindowWidget_t structure 638
 libaria_common.h 641
 libaria_context.h 641
 libaria_draw.h 643
 libaria_editwidget.h 643
 libaria_event.h 643
 libaria_global.h 644
 libaria_input.h 645
 libaria_layer.h 646
 libaria_list.h 647
 libaria_math.h 647
 libaria_radiobutton_group.h 648
 libaria_rectarray.h 648
 libaria_scheme.h 649
 libaria_screen.h 649
 libaria_string.h 650
 libaria_utils.h 652
 libaria_widget.h 653
 libaria_widget_arc.h 655
 libaria_widget_bar_graph.h 656
 libaria_widget_button.h 657
 libaria_widget_checkbox.h 659
 libaria_widget_circle.h 660
 libaria_widget_circular_gauge.h 660
 libaria_widget_circular_slider.h 662
 libaria_widget_drawsurface.h 663
 libaria_widget_gradient.h 663
 libaria_widget_groupbox.h 664
 libaria_widget_image.h 665
 libaria_widget_imageplus.h 665
 libaria_widget_imagesequence.h 666
 libaria_widget_keypad.h 667
 libaria_widget_label.h 669
 libaria_widget_line.h 669
 libaria_widget_line_graph.h 670
 libaria_widget_list.h 672
 libaria_widget_listwheel.h 673
 libaria_widget_pie_chart.h 675
 libaria_widget_progressbar.h 676
 libaria_widget_radial_menu.h 676
 libaria_widget_radiobutton.h 678
 libaria_widget_rectangle.h 679
 libaria_widget_scrollbar.h 679
 libaria_widget_slider.h 680
 libaria_widget_textfield.h 681
 libaria_widget_touchtest.h 682
 libaria_widget_window.h 683
 Library Interface 689
 Library Overview 748
 LINE_GRAPH_AXIS_0 enumeration member 604
 LINE_GRAPH_DATA_POINT_CIRCLE enumeration member 603
 LINE_GRAPH_DATA_POINT_NONE enumeration member 603
 LINE_GRAPH_DATA_POINT_SQUARE enumeration member 603
 LINE_GRAPH_TICK_CENTER enumeration member 603
 LINE_GRAPH_TICK_IN enumeration member 603
 LINE_GRAPH_TICK_OUT enumeration member 603

M

Memory Configuration 130
 Menus 96
 MHC GFX Components 75
 MHGC Tools 157
 Microchip Forums 807
 Microchip Website 807
 MPLAB Harmony Display Manager User's Guide 197
 MPLAB Harmony Graphics Composer (MHGC) Suite 201
 MPLAB Harmony Graphics Composer User's Guide 89

N

New Project Wizard 101
 Next Steps 15
 NUM_BUTTONS macro 639
 NUM_KEYS macro 639

O

Object Properties 115
 Options 111
 OUTSIDE_CIRCLE_BORDER enumeration member 581

P

PDF Help Features 806

Properties Editor Panel 114

Q

Quick Start Guides 17

R

Recommended Reading 801

Running the Demonstration 37, 47, 51, 61, 70

S

Schemes Panel 109

Screen Designer Window 92

Screen vs. Widget Events 161

Screens Panel 107

Small Buttons Controlled by Phantom Buttons 187

Speed and Performance of Different Image Decode Formats in MHGC
178

String Assets 153

String Table Configuration 149

Summary 197

Support 799

T

Trademarks 799

Tree View Panel 104

Typographic Conventions 800

U

Using the Help 799

Using the Library 747

V

VALUE_ARC enumeration member 578

W

Why Graphics? 4

Why use MPLAB Harmony Graphics? 5

Widget Colors 168

Widget Tool Box Panel 111