MPLAB® Harmony Help - Graphics Libraries

MPLAB Harmony Integrated Software Framework v1.11

Graphics Libraries Help

This section provides information on the Graphics Libraries that are available in MPLAB Harmony.

MPLAB Harmony Graphics Library

This section provides information on the MPLAB Harmony Graphics Library.

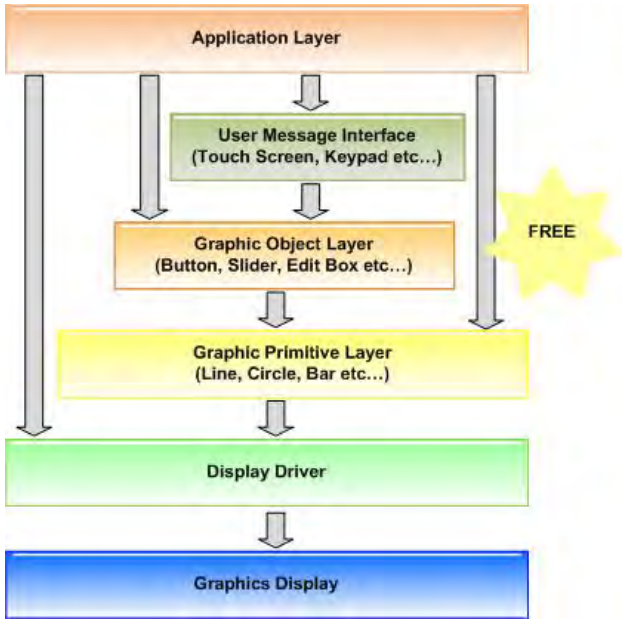
Introduction

This topic provides an overview of the Graphics Library.

Description

The Graphics Library is a free, modular library optimized for Microchip microcontrollers. The library structure is shown in the following figure.

Graphics Library Architecture



- Application Layer - This is the program that utilizes the Graphics Library
- User Message Interface - This is a layer that should be implemented by the user to provide messages for the library
- Graphics Object Layer - This layer renders the control objects such as button, list box, progress bar, meter, and so on
- Graphics Primitives Layer - This layer implements the primitive rendering functions
- Device Display Driver - This layer is the graphics display driver component that is optimized to the actual display module used
- Graphics Display Module - This is the actual display module

The library includes features such as alpha blending, gradient fills, and anti-aliased fonts. These features can be enabled or disabled through build configurations. Applications can take advantage of these features to enhance the user experience while delivering performance required by the application.

Graphics Library Porting Guide

This topic provides porting information for the Graphics Library.

Introduction

This section provides information for porting from the Microchip Library of Applications (MLA) version of the Graphics Library to the Graphics Library within MPLAB Harmony.

Upgrading from the MLA Graphics Library to the MPLAB Harmony Graphics Library

The MPLAB Harmony Graphics Library is a completely new distribution, which is included as part of the MPLAB Harmony installation. No files need to be maintained from an existing MLA Graphics Library installation.

MPLAB Harmony Graphics Library Key Features

This topic provides a description of the key features of the MPLAB Harmony Graphics Library.

Description

Note: The MPLAB Harmony Graphics Library *does not* support any Microchip 16-bit family devices.

The following are the key features of the MPLAB Harmony Graphics Library:

- Multiple instances
- Fully dynamic:
  - Stack initialization/deinitialization
  - Resource management
  - Configurable
- Improved modularity and stack layout
- Interrupt driven operation
- RTOS friendly, with easy RTOS integration

MPLAB Harmony Graphics Library Structure

This topic describes the structure of the MPLAB Harmony Graphics Library.

Description

The MPLAB Harmony Graphics Library consists of a graphic widget and primitive library, a display layer, a graphics driver, and demonstration applications, all of which support the following structure:

- Lowercase file names
- Underscores are used
- System services are used instead of direct access to peripherals
- Interrupts
- Drivers
- Timer Services
- File System
- Board Support Package (BSP)

By default, the Graphics Library and its corresponding display drivers, Board Support Package (BSP) and demonstration applications are placed into the following locations during installation of MPLAB Harmony:

- `/microchip/harmony/<version>/framework/gfx`
- `/microchip/harmony/<version>/framework/driver/gfx`
- `/microchip/harmony/<version>/apps/gfx`

MPLAB Harmony Graphics Library Design

This topic discusses the design of the MPLAB Harmony Graphics Library.

Description

The MPLAB Harmony Graphics Library is designed as a part of a system running other applications, middleware, etc. Therefore, it makes use of the system services that are available to other modules in the system or to the application, such as:

- File System
- System Interrupts
- System Driver Services
- System Timers
- System Device Drivers
- System Command Processor
- System Console
- System Debug Services
- Header files located in `./framework/gfx/src/system` that expose the system-wide available API

MPLAB Harmony Graphics Library API Changes

This topic discusses the reasons behind the API changes to the MPLAB Harmony Graphics Library.

Description

The API is backward compatible in functionality only. All function names and type definitions have changed to reflect the MPLAB Harmony philosophy.

The changes in the library API have been minimized so that the porting process is straightforward and integration with other dependents is made simple. New functionality has been added because of new features, such as dynamic configuration and non-block of the library. Other than these new features, the changes to the API were also made to support commonality between the MPLAB Harmony Graphics Library and the 16-bit Graphics Library.

The API changes are at all layers of the Graphics Library which include the GOL, Primitive, and Driver layers.

MPLAB Harmony Graphics Library Access Changes

This topic describes the stack access updates that were made to the MPLAB Harmony Graphics Library.

Description

The Graphics Library provides the `gfx.h` and `gfx.c` file as the central interface to the library.

The Graphics Library can provide mutually exclusive access to multiple clients. Each client calls the GFX_Initialize function using a client index and appropriate initialization parameters.

The GFX_Initialize function should be called before using the Graphics Object Layer and Primitive Layer.

The Graphics Library no longer initializes the display driver, but requires the application to initialize the driver before calling any Graphics Library functions. For example: GFX_DRV_lcc_Initialize

Note: Direct access to the Graphics Driver rendering functions *is not* supported.

Graphic Primitive Layer Changes

This topic discusses the changes made to the MLA Graphics Library `primitive.h` file to produce the equivalent MPLAB Harmony Graphics Library `gfx_primitive.h` file.

Description

Changes

The changes are as follows:

- *Prefix:*
  - `GFX_` is the generic prefix for functions
  - `gfx_` is the generic prefix for primitive level files
- Multi-instance support - all re-entrant functions require a client instance index to be passed in as a parameter

Graphic Object Layer (GOL) Changes

This topic discusses the changes made to the MLA Graphics Library `GOL.h` file to produce the equivalent MPLAB Harmony Graphics Library `gfx_gol.h` file.

Description

Changes

The changes are as follows.

- *Prefix:*
  - `GFX_GOL_` is the generic prefix for all Graphic Object Level functions
  - `gfx_gol_` is the generic prefix for all Graphic Object Level files
- Pre-emption Level support - a three-level self preemption configuration has been added for time-critical rendering
- Multi-instance support - all re-entrant functions require a client instance index to be passed in as a parameter

Graphic Display Driver Layer Changes

This topic discusses the changes made to the `DisplayDriver.h` file.

This topic discusses the changes made to the MLA Graphics Library `DisplayDriver.h` file to produce the equivalent MPLAB Harmony Graphics Library `drv_gfx_display.h` file.

Description

Changes

The changes are as follows:

- *Prefix:*
  - `DRV_GFX` is the generic prefix for all Graphics Driver level functions
  - `drv_gfx` is the generic prefix for all Graphics Driver level files
- *Common API Interface* – This interface establishes the basic set of driver functions and parameters required by the Graphics Primitive Layer to display graphics content on a Liquid Crystal Display. It also lists the initialization and deinitialization functions and parameters required by the application to establish a driver.

MPLAB Harmony Graphics Library Initialization Changes

This topic describes the initialization updates to the MPLAB Harmony Graphics Library.

Description

The Graphics Library include header files, `gfx_gol.h` and `gfx_primitive.h`, have been updated as follows.

To initialize the library the call is now: `GFX_Initialize`

The default Graphics Library configuration is provided with the library distribution and consists of selecting **Graphics Library** through the MHC Options tab.

New MPLAB Harmony Graphics Library API Functions

This topic describes some of the important new API functions in the MPLAB Harmony Graphics Library.

Description

There are many new functions available that have been introduced to support the MPLAB Harmony philosophy. However, there is no concern for the porting process regarding the new API calls as the previous library implementation did not support this kind of service.

Existing applications will not port without using the new APIs:

- Initialization
- Steady State

MPLAB Harmony Graphics Library Utilities

This topic describes the MPLAB Harmony Graphics Library utilities.

Description

The MPLAB Harmony Graphics Library includes the following utilities:

- Graphics Display Designer X (GDDX) - This utility can be used for designing and building graphical user interfaces (GUIs) from graphics objects
- Graphics Resource Converter (GRC) - This utility can be used for converting data into logically deployable entities that can be used in a graphics application

Porting Applications from the MLA Graphics Library to the MPLAB Harmony Graphics Library

Porting an application from the MLA Graphics Library into the Graphics Library within MPLAB Harmony consists of updating existing Graphics Library function names in your application to the new MPLAB Harmony function names.

MPLAB Harmony Graphics Library File Name Compliance

This topic provides header file name mapping tables that show the MLA Graphics Library file name and the compliant name within the MPLAB Harmony Graphics Library.

Description

The replacement names chosen to make existing Graphics Library functions compliant with MPLAB Harmony are shown in individual tables per header file name.

The files listed in the tables are located in the following two MPLAB Harmony installation folders:

`.\harmony\<version>\framework\gfx`

`.\harmony\<version>\framework\gfx\src`

The following table shows the original and new names used. Note that other than name changes, no other modifications were made to the middleware functions.

MLA Graphics Library File Name	MPLAB Harmony Graphics Library File Name
AnalogClock.h	gfx_gol_analog_clock.h
Button.h	gfx_gol_button.h
Chart.h	gfx_gol_chart.h
CheckBox.h	gfx_gol_check_box.h
DigitalMeter.h	gfx_gol_digital_meter.h
EditBox.h	gfx_gol_edit_box.h
GOL.h	gfx_gol.h
Graphics.h	gfx.h
Grid.h	gfx_gol_grid.h (not supported in this release)
GroupBox.h	gfx_gol_group_box.h
ListBox.h	gfx_gol_list_box.h
Meter.h	gfx_gol_meter.h
Palette.h	gfx_palette.h
Picture.h	gfx_gol_picture.h
Primitive.h	gfx_primitive.h
ProgressBar.h	gfx_gol_progress_bar.h
Radiobutton.h	gfx_gol_radio_button.h
RoundDial.h	gfx_gol_round_dial.h
Slider.h	gfx_gol_scroll_bar.h
StaticText.h	gfx_gol_static_text.h
TextEntry.h	gfx_gol_text_entry.h
Transitions.h	gfx_transitions.h
Window.h	gfx_gol_window.h

MPLAB Harmony Graphics Library Function Name Compliance

This topic provides header file function mapping tables that show the MLA Graphics Library function name and the compliant name within the MPLAB Harmony Graphics Library.

Description

The replacement function names chosen to make the existing MLA Graphics Library function compliant with MPLAB Harmony are shown in individual tables per header file name.

The files listed in the tables are located in the following MPLAB Harmony installation folder: `./harmony/<version>/framework/gfx`.

Header File: `gfx_display_driver.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
InitGraph()	GFX_PixelsPut()
PutPixel	GFX_BarFill()
GetPixel()	GFX_PixelArrayPut()
SetColor()	GFX_PixelArrayGet()
PutPixel	GFX_PixelPut()
SetColor ()	GFX_SetColor()

GetTransparentColor()	GFX_SetInstance()
TransparentColorEnable()	GFX_SetPage()
TransparentColorDisable()	GFX_Layer()
Bar()	GFX_PixelGet()
N/A	GFX_AlphaBlendWindow()

Header File: `gfx_primitive.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
InitGraph()	GFX_Initialize()
PutPixel	GFX_PixelPut()
GetPixel()	GFX_PixelGet()
SetColor()	GFX_ColorSet()
GetColor	GFX_ColorGet()
TransparentColorEnable()	GFX_TransparentColorSet()
GetTransparentColor()	GFX_TransparentColorGet()
TransparentColorEnable()	GFX_TransparentColorEnable()
TransparentColorDisable()	GFX_TransparentColorDisable()
Bar()	GFX_BarDraw()

Header File: `gfx_palette.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
PaletteInit()	GFX_PaletteInit()
EnablePalette()	GFX_PaletteEnable()
DisablePalette()	GFX_PaletteDisable()
IsPaletteEnabled()	GFX_IsPaletteEnabled()
GetPaletteChangeError()	GFX_PaletteChangeErrorGet()
ClearPaletteChangeError()	GFX_PaletteChangeErrorClear()
SetPaletteBpp()	GFX_PaletteBppSet()
RequestPaletteChange()	GFX_PaletteChangeRequest()
RequestEntirePaletteChange()	GFX_EntirePaletteChangeRequest()
SetPalette()	GFX_PaletteSet()
SetEntirePalette()	GFX_EntirePalette()
SetPaletteFlash()	GFX_PaletteFlashSet()

Header File: `gfx_gol.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
GetState()	GFX_GOL_ObjectStateGet()
ClearState()	GFX_GOL_ObjectStateClear()
SetState()	GFX_GOL_ObjectStateSet()
GOLAddObject()	GFX_GOL_ObjectAdd()
GOLFindObject()	GFX_GOL_ObjectFind()
GOLRedraw()	GFX_GOL_ObjectListRedraw()
GOLRedrawRec()	GFX_GOL_ObjectRectangleRedraw()
GOLDraw()	GFX_GOL_ObjectListDraw()
GOLDrawComplete()	GFX_GOL_ObjectDrawComplete()
GOLDrawCallback()	GFX_GOL_DRAW_CALLBACK_FUNC
GOLFree()	GFX_GOL_ObjectListFree()

GetObjType()	GFX_GOL_ObjectTypeGet()
GetObjID	GFX_GOL_ObjectNextGet()
GetObjNext()	GFX_GOL_ObjectNext()
GOLDeleteObject()	GFX_GOL_ObjectDelete()
GOLDeleteObjectByID()	GFX_GOL_ObjectByIDDelete()
GOLNewList()	GFX_GOL_ObjectListNew()
GOLGetList()	GFX_GOL_ObjectListGet()
GOLSetFocus()	GFX_GOL_ObjectFocusSet()
IsObjUpdated()	GFX_GOL_ObjectUpdated()
GOLInit()	GFX_Initialize()
GOLGetFocus()	GFX_GOL_ObjectFocusGet()
GOLCanBeFocused()	GFX_GOL_ObjectCanBeFocused()
GOLGetFocusNext()	GFX_GOL_ObjectFocusNextGet()
GOLGetFocusPrev()	GFX_GOL_ObjectFocusPrevGet()
GOLPanelDraw()	GFX_GOL_PanelParameterSet()
GOLPanelDrawTsk()	GFX_GOL_PanelDraw()
GOLTwoTonePanelDraw()	GFX_GOL_PanelTwoToneDraw()

Header File: `gfx_gol_button.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
BtnCreate()	GFX_GOL_ButtonCreate()
BtnDraw()	GFX_GOL_ButtonDraw()
BtnGetText()	GFX_GOL_ButtonTextGet()
BtnSetText()	GFX_GOL_ButtonTextSet()
BtnGetBitmap()	GFX_GOL_ButtonPressStateImageGet()
BtnSetBitmap()	GFX_GOL_ButtonPressStateImageSet()
BtnMsgDefault()	GFX_GOL_ButtonActionSet()
BtnTranslateMsg()	GFX_GOL_ButtonActionGet()

Header File: `gfx_gol_check_box.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
CbCreate()	GFX_GOL_CheckBoxCreate()
CbDraw()	GFX_GOL_CheckBoxDraw()
CbGetText()	GFX_GOL_CheckBoxTextGet()
CbSetText()	GFX_GOL_CheckBoxTextSet()
CbMsgDefault()	GFX_GOL_CheckBoxActionSet()
CbTranslateMsg()	GFX_GOL_CheckBoxActionGet()

Header File: `gfx_gol_custom_control.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
No equivalent	GFX_GOL_CustomControlGetPos ()
No equivalent	GFX_GOL_CustomControlSetPos ()
No equivalent	GFX_GOL_CustomControlCreate
No equivalent	GFX_GOL_CustomControlActionGet ()
No equivalent	GFX_GOL_CustomControlActionGet ()
No equivalent	GFX_GOL_CustomControlActionSet ()
No equivalent	GFX_GOL_CustomControlDraw ()

Header File: `gfx_gol_digital_meter.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
DmCreate()	GFX_GOL_DigitalMeterCreate()
DmDraw()	GFX_GOL_DigitalMeterDraw()
DmGetValue()	GFX_GOL_DigitalMeterValueGet()
DmSetValue()	GFX_GOL_DigitalMeterValueSet()
DmDecVal()	GFX_GOL_DigitalMeterDecrement()
DmIncVal()	GFX_GOL_DigitalMeterIncrement()
DmTranslateMsg()	GFX_GOL_DigitalMeterActionGet()

Header File: `gfx_gol_edit_ box.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
EbCreate()	GFX_GOL_EditBoxCreate()
EbDraw()	GFX_GOL_EditBoxDraw()
EbGetText()	GFX_GOL_EditBoxTextGet()
EbSetText()	GFX_GOL_EditBoxTextSet()
EbAddChar()	GFX_GOL_EditBoxCharAdd(()
EbDeleteChar()	GFX_GOL_EditBoxCharDelete()
EbMsgDefault()	GFX_GOL_EditBoxActionSet()
EbTranslateMsg()	GFX_GOL_EditBoxActionGet()

Header File: `gfx_gol_group_box.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
GbCreate()	GFX_GOL_GroupBoxCreate()
GbDraw()	GFX_GOL_GroupBoxDraw()
GbGetText()	GFX_GOL_GroupBoxTextGet()
GbSetText()	GFX_GOL_GroupBoxTextGet()
GbTranslateMsg()	GFX_GOL_GroupBoxActionGet()
GbCreate()	GFX_GOL_GroupBoxCreate()
GbDraw()	GFX_GOL_GroupBoxDraw()

Header File: `gfx_gol_list_box.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
LbCreate()	GFX_GOL_ListBoxCreate()
LbDraw()	GFX_GOL_ListBoxDraw()
LbGetItemList()	GFX_GOL_ListBoxItemListGet()
LbAddItem()	GFX_GOL_ListBoxtemAdd()
LbAddItem()	GFX_GOL_ListBoxItemAdd()
LbDelItem()	GFX_GOL_ListBoxItemRemove()
LbChangeSel()	GFX_GOL_ListBoxChangeSelection()
LbSetSel()	GFX_GOL_ListBoxSelectionSet()
LbGetSel()	GFX_GOL_ListBoxSelectionGet()
LbGetFocusedItem()	GFX_GOL_ListBoxFocusedItemGet()
LbSetFocusedItem()	GFX_GOL_ListBoxFocusedItemSet()
LbGetCount(pLb)	GFX_GOL_ListBoxCountGet()
LbGetVisibleCount()	GFX_GOL_ListBoxVisibleCountGet()
LbSetBitmap()	GFX_GOL_ListBoxBitmapSet()

LbGetBitmap(pItem)	GFX_GOL_ListBoxBitmapSet()

Header File: gfx_gol_meter.h

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
MtrCreate()	GFX_GOL_MeterCreate()
MtrDraw()	GFX_GOL_MeterDraw()
MtrSetVa()	GFX_GOL_MeterValueSet()
MtrGetVal()	GFX_GOL_MeterValueGet()
MtrDecVal()	GFX_GOL_MeterDecrementl()
MtrIncVal	GFX_GOL_MeterIncrement()
MtrSetScaleColors	GFX_GOL_MeterScaleColorsSet()
MtrSetTitleFont(pMtr, pNewFont)	GFX_GOL_MeterTitleFontSet()
MtrSetValueFont(pMtr, pNewFont)	GFX_GOL_MeterValueFontSet()
MtrMsgDefault()	GFX_GOL_MeterDefaultActionSet()
MtrTranslateMsg()	GFX_GOL_MeterActionGet()

Header File: `gfx_gol_picture.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
PictCreat()	GFX_GOL_PictureControlCreate()
PictDraw()	GFX_GOL_PictureControlDraw()
PictSetBitmap()	GFX_GOL_PictureControlBitmapSet()
PictGetBitmap()	GFX_GOL_PictureControlBitmapGet()
PictGetScale()	GFX_GOL_PictureControlScaleGet()
PictSetScale()	GFX_GOL_PictureControlScaleGet()
PictTranslateMsg()	GFX_GOL_PictureControlActionGet()

Header File: `gfx_gol_progress_bar.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
PbCreate()	GFX_GOL_ProgressBarCreate()
PbDraw()	GFX_GOL_ProgressBarDraw()
PbSetRange()	GFX_GOL_ProgressBarRangeSet()
PbGetRange()	GFX_GOL_ProgressBarRangeGet()
PbSetPos()	GFX_GOL_ProgressBarPositionSet()
PbGetPos()	GFX_GOL_ProgressBarActionSet()
PbTranslateMsg()	GFX_GOL_ProgressBarActionGet()

Header File: `gfx_gol_radio_button.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
RbCreate()	GFX_GOL_RadioButtonCreate()
RbDraw()	GFX_GOL_RadioButtonDraw()
RbGetCheck()	GFX_GOL_RadioButtonCheckGet()
RbSetCheck()	GFX_GOL_RadioButtonCheckGet()
RbGetText(pRb)	GFX_GOL_RadioButtonTextGet()
RbSetText()	GFX_GOL_RadioButtonTextGet()
RbMsgDefault()	GFX_GOL_RadioButtonActionSet()

Header File: `gfx_gol_scrollbar.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
SldCreate()	GFX_GOL_ScrollBarCreate()
SldDraw()	GFX_GOL_ScrollBarDraw()
SldSetPage()	GFX_GOL_ScrollBarPageSet()
SldGetPage()	GFX_GOL_ScrollBarPageGet()
SldSetPos()	GFX_GOL_ScrollBarPositionSet()
SldGetPos(pSld)	GFX_GOL_ScrollBarPositionGet()
SldSetRange()	GFX_GOL_ScrollBarSetRange()
SldGetRange()	GFX_GOL_ScrollBarRangeGet()
SldIncPos()	GFX_GOL_ScrollBarPositionIncrement()
SldDecPos()	GFX_GOL_ScrollBarPositionDecrement()
SldMsgDefault()	GFX_GOL_ScrollBarActionSet()

Header File: `gfx_gol_static_text.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
StCreate()	GFX_GOL_StaticTextCreate()
StDraw()	GFX_GOL_StaticTextDraw()
StGetText()	GFX_GOL_StaticTextGet()
StSetText()	GFX_GOL_StaticTextSet()
StTranslateMsg()	GFX_GOL_StaticTextActionGet()

Header File: `gfx_gol_text_entry.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
TeCreate()	GFX_GOL_TextEntryCreate()
TeDraw()	GFX_GOL_TextEntryDraw()
TeGetBuffer()	GFX_GOL_TextEntryBufferGet()
TeSetBuffer()	GFX_GOL_TextEntryBufferSet()
TeClearBuffer()	GOL_GOL_TextEntryBufferClear()
TeGetKeyCommand()	GOL_GOL_TextEntryKeyCommandGet()
TeSetKeyCommand()	GOL_GOL_TextEntryKeyCommandSet()
TeCreateKeyMembers()	GOL_GOL_TextEntryKeyMembersCreate()
TeAddChar()	GOL_GOL_TextEntryCharAdd()
TeIsKeyPressed()	GOL_GOL_TextEntryKeyPressed()
TeSpaceChar()	GOL_GOL_TextEntryCharSpace()
TeDelKeyMembers()	GOL_GOL_TextEntryKeyMembersDelete()
TeSetKeyText()	GOL_GOL_TextEntrySetKeyText()
TeMsgDefault()	GOL_GOL_TextEntryActionSet()
TeTranslateMsg()	GOL_GOL_TextEntryActionGet()

Header File: `gfx/gfx_gol_window.h`

MLA Graphics Library Function Name	MPLAB Harmony Graphics Library Function Name
WndCreate()	GFX_GOL_WindowCreate()
WndDraw()	GFX_GOL_WindowDraw()
WndGetText()	GFX_GOL_WindowTextGet()
WndSetText()	GFX_GOL_WindowTextSet()
WndTranslateMsg()	GFX_GOL_WindowActionGet()

Using the Library

This topic describes the architecture of the Graphics Library and the components in the Primitive and Object Layers. Information on how to use the Primitive Layer and Object Layer is also provided.

Description

The Graphics Library is composed of the following layers:

- Application Layer - This is the program that utilizes the Graphics Library
- User Message Interface - This is a layer should be implemented by user to provide messages for the library
- Graphics Object Layer - This layer renders the control objects such as button, list box, progress bar, meter and so on
- Graphics Primitives Layer - This layer implements the primitive rendering functions
- Device Display Driver - This layer is the graphics display driver component that is optimized to the actual display module used
- Graphics Display Module - This is the actual display module

Library Overview

The library interface routines are divided into various sub-sections, which address one of the blocks or the overall operation of the Graphics Library.

Library Interface Section	Description
Graphics Primitive Layer Functions	Functions for managing the Primitive Layer of the Graphics Library
Graphics Object Layer Functions	Functions for managing the Object Layer of the Graphics Library
Graphics Functions	Graphics Library configuration functions.

Graphics Objects

This section describes the basics of the Graphics Object Layer.

Description

The Graphics Object Layer is composed of objects that can act as control widgets in an application. There are several predefined objects in the library. Users are not limited to these predefined objects. It is possible to create custom objects. Creating custom objects will require a deeper understanding of the library's architecture. Creating custom objects is described in the Microchip application note, AN1136 *"How to Use Widgets in Microchip Graphics Library*" (DS01136).

Each object will have its own create function. In the create function the following functions are initialized:

- DRAW_FUNC - This is the function that renders the object on the screen
- FREE_FUNC - This is the function that frees the memory used by the object when the object is removed from memory
- ACTIONGET_FUNC - This optional function returns the a translated action of the object. A translated action is actually a planned action that indicates the intended change in the state of the object. The translated action is basically the response of the object to the user interaction on the object. For example, when a user presses on a button object on a screen, a valid planned action of the object is to change its state to a pressed state. Translated actions are specific to each object. See the GFX_GOL_TRANSLATED_ACTION enumeration for the listing of all translated actions of all objects in the library. This list does not include user-defined actions. Translated actions are determined using the messaging scheme of the library. Messaging is discussed in a separate section of this documentation. When this function is not defined, the create function of the object must set the function pointer to NULL.
- ACTIONSET_FUNC - This optional function performs the translated action returned by the ACTIONGET_FUNC. This is the actual function that changes the state of the object. When this function is not defined, the create function of the object must set the function pointer to NULL.

Each of these functions are implemented in each object. The draw function of the object is specific to the object. The draw function should not be called directly by the application. Instead, applications should use the GFX_GOL_ObjectListDraw function so that the management of the object rendering will be done by the Graphics Object Layer.

Object Layer Rendering

This section describes how objects are rendered.

Description

Object List Rendering

Initially, an application will create the objects that it will use by calling the create function of the objects. As the objects are created, a linked list of objects is also created. This linked list of objects is the list that the library will use to manage the rendering of the objects. The application will only need to call the GFX_GOL_ObjectListDraw function to execute the rendering of all the objects. It is also through this list that the library is able to

parse all the objects to determine if an object is affected by the user action. This is discussed in the Object Layer Messaging section.

Each object, has a state variable that is composed of state bits. There are two main groups in the state bits:

- Property State bits - define the general shape of the object
- Draw State bits - determine if the object will be redrawn. The redrawing can be a full object redraw or partial object redraw. Depending on the object, definition, a partial redraw means that only portion of the object is redrawn. This makes the redraw of the object fast and efficient. For example, redrawing the full progress bar object will be longer than just updating the level of the progress bar.

The Property State bits are usually defined once at the object creation. Conversely, the Draw State bits, dynamically change and usually change because of user action on the object. These user actions are encoded as messages. The messages are created in user interface drivers such as touch screen or key pads on the system (see the Object Layer Messaging section for details).

The objects in the library implements the rendering sequence using rendering state machine. The rendering state is not to be confused with the Draw State Bits of the objects. Rendering states are defined in each object's rendering function. An object is basically drawn using combination of primitive calls. Bars, lines, and text can be drawn to represent an object. The purpose of these rendering states is to divide the primitive calls into steps. By doing this, each primitive call is now a step in the process. If one of the primitive call cannot proceed due to busy hardware resource, the object's rendering sequence can be paused. Since, the length of the delay in the rendering cannot be predicted, it is best to pause the rendering of the object and give back control of the CPU to application. Each of the object's rendering function coupled with GFX_GOL_ObjectListDraw, has this capability. This effectively pauses the rendering of the object.

For the rendering to continue, GFX_GOL_ObjectListDraw is called again by the application. When this is called, the specific object rendering function that was called is then resumed and in the object rendering function, the specific step (or primitive call) is executed again. This basically continues the rendering exactly where it left off.

Object Draw Callback Feature

To an allow application to control the rendering of the objects, a callback function is provided (see GFX_GOL_DRAW_CALLBACK_FUNC). The callback function is an application implemented function. The library provides the GFX_GOL_DrawCallbackSet(GFX_GOL_DRAW_CALLBACK_FUNC) API to set the callback function. GFX_GOL_DRAW_CALLBACK_FUNC parameter in this function is the pointer to the defined callback.

The callback allows the application to insert application defined rendering.

For example, a digital signal is sampled on a pin. The value of the signal determines if a scroll bar will increment or decrement its value. To implement the change in the value of a scroll bar, a callback function can be implemented to do that.

```
void App_Tasks()
{
   switch(appData.state)
    {
        case APP_STATE_GFX_READY:
            if ( GFX_Status (gfxObject) == SYS_STATUS_READY )
                appData.state = APP_STATE_GFX_INIT;

            break;

        case APP_STATE_GFX_INIT:

            GFX_GOL_MessageCallbackSet(GFX_INDEX_0, &GFX_GOL_MessageCallback);
            GFX_GOL_DrawCallbackSet(GFX_INDEX_0, &GFX_GOL_DrawCallback);

            appData.state = APP_STATE_RUNNING;
            break;
    }
}

//*****************************************************************
// The draw callback is called within GFX_GOL_ObjectListDraw()
//*****************************************************************
bool APP_ObjectDrawCallback(void)
{
// assume pScrollBar is pointer to the scroll bar
bool signalAssert;
// assume signal is sampled at a pin RC1
if (PORTCbits.RC1 == 1)
signalAssert = true;
else
signalAssert - false;
if (signalAssert == true)
{
// increment the value
GFX_GOL_ScrollBarPositionIncrement(pScrollBar);
}
else
```

```
{
// increment the value
GFX_GOL_ScrollBarPositionDecrement(pScrollBar);
}
// redraw only the thumb
GFX_GOL_ObjectStateSet(pScrollBar, GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE);
return (true);
}
```

Since the callback is called in GFX_GOL_ObjectListDraw and GFX_GOL_ObjectListDraw is called in the main loop, the callback can potentially be executed unnecessarily if the sampled signal frequency of change is slow. To counter that, the callback can be modified to be sampling the signal after some predefined time delay.

```
//*****************************************************************
// The draw callback is called within GFX_GOL_ObjectListDraw()
//*****************************************************************
bool APP_ObjectDrawCallback(void)
{
// assume pScrollBar is pointer to the scrollbar
// assume tick is a system counter, DEFINED_DELAY is
// a predefined delay tuned to the signal frequency
bool signalAssert;
static uint32_t prevtick = 0;
if (tick - prevtick > DEFINED_DELAY)
{
prevtick = tick;
// assume signal is sampled at a pin RC1
if (PORTCbits.RC1 == 1)
signalAssert = true;
else
signalAssert - false;
if (signalAssert == true)
{
// increment the value
GFX_GOL_ScrollBarPositionIncrement(pScrollBar);
}
else
{
// increment the value
GFX_GOL_ScrollBarPositionDecrement(pScrollBar);
}
// redraw only the thumb
GFX_GOL_ObjectStateSet(pScrollBar, GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE);
}
return (true);
```

Object Layer Messaging

This section describes the basics of object messages.

Description

User inputs can also change the state of objects. This is done through messaging. The library passes user action to the objects using the message structure (see GFX_GOL_MESSAGE). The purpose of the messaging is to translate user action (i.e., a user press on a touch screen) to a change in state of the objects in the screen. At the same time, this user action will cause an application function to execute.

For example, a press on a button on the screen will result in an LED being turned on. While pressing on the button, the button on the screen also changes states and is redrawn to a pressed state. When the user releases the button, the button redraws back to an unpressed state and the LED turns off.

The mechanism of messaging and object state change is explained in the next section.

Messaging Structure

The messages are encoded in the GFX_GOL_MESSAGE structure.

```
    typedef struct
    {
        uint8_t type;           // Specifies the type of input device.
        uint8_t uiEvent;        // An event that occurred in the input device.
        int16_t param1;         // Parameter 1, definition and usage is
                                // dependent on the type of input device.
        int16_t param2;         // Parameter 2, definition and usage is
                                // dependent on the type of input device.
```

```
    } GFX_GOL_MESSAGE;
```

The type defines the type of input device. See INPUT_DEVICE_TYPE for the currently supported type of devices. The uiEvent is also a set of predefined events. See INPUT_DEVICE_EVENT for a list of supported events. The two parameters, param1 and param2, change in usage depending on the type of input device and event.

The messages are encoded in the drivers for the input devices. The drivers provide functions for the application to call. These functions should return the message whenever there is a valid user action on the input device.

The following code example shows TouchGetMsg as the function provided by the touch screen driver to retrieve user inputs on the touchscreen.

```
    // GOL message structure
    GFX_GOL_MESSAGE msg;

    ...
    // set the message callback function pointer
    GFX_GOL_MessageCallbackSet(APP_ObjectMessageCallback);
    ...

    uint16_t nQSizes[] = SYS_MSG_BUFFER_SIZES;

    sSysMsgInit.nQSizes = nQSizes;
    hSysMsg = SYS_MSG_Initialize(iSysMsg,(SYS_OBJ_HANDLE)&sSysMsgInit);
    hMsgType = SYS_MSG_TypeCreate(iSysMsg, TYPE_TOUCHSCREEN , 0);
    hMailbox = SYS_MSB_MailboxOpen( iSysMsg, (SYS_MSG_RECEIVE_CALLBACK)&TouchMessageCallback );
    SYS_MSG_MailboxMsgAdd(hMailbox, hMsgType);

void TouchMessageCallback(SYS_MSG_OBJECT *pMsg)
{
    static GFX_GOL_MESSAGE gMsg;

    gMsg.type = pMsg->nMessageTypeID;
    gMsg.uiEvent = pMsg->param0;
    gMsg.param1 = pMsg->param1;
    gMsg.param2 = pMsg->param2;

    GFX_GOL_ObjectMessage(GFX_INDEX_0, &gMsg);
}
```

Graphics Object Message Function

After the input device driver encodes the message, the message is sent by the application to the Graphics Library through the GFX_GOL_ObjectMessage. This function parses the active list of objects and determines if the message affects one or more of the objects. The mechanism to check each object is simple: the GFX_GOL_ObjectMessage calls each of the object's ACTIONGET_FUNC. The ACTIONGET_FUNC of an object is the function that determines if the message affects the object. This function returns a translated action (see the GFX_GOL_TRANSLATED_ACTION enumeration).

If the object is not affected it replies with GFX_GOL_OBJECT_ACTION_INVALID. If the object is affected, it replies with the appropriate translated action.

Object Message Callback Feature

If the object is affected, GFX_GOL_ObjectMessage() then calls the message callback function. The message callback function is an application defined function. This is set initially using the GFX_GOL_MessageCallbackSet(GFX_GOL_MESSAGE_CALLBACK_FUNC) call where GFX_GOL_MESSAGE_CALLBACK_FUNC is a pointer to the defined callback function.

The message callback is an opportunity for the application to respond to user inputs. It is also an opportunity to block the library from performing default state change to objects. See the callback function documentation for details.

The message callback returns true if the application wants to perform the default action set of the objects. If the callback returns false, the application assumes all changes in the states of the object.

After the callback is executed, with a return value of true, the ACTIONSET_FUNC of the affected object is called by GFX_GOL_ObjectMessage. This function performs the state change of the object due to the user input. The object is then redrawn when GFX_GOL_ObjectListDraw is called by the application.

The following code example demonstrates usage of the message callback function. A scroll bar is incremented and decremented by two buttons.

```
//*****************************************************************
// The message callback is called within GFX_GOL_ObjectMessage()
//*****************************************************************

// assume that two buttons and a scroll bar are present in the system
bool APP_MsgCallback(
            uint16_t objMsg,
            GFX_GOL_OBJ_HEADER *pObj,
            GFX_GOL_MESSAGE *pMsg)
```

```
{
    uint16_t objectID;
    GFX_GOL_SCROLLBAR *pScrollBar;
    objectID = GFX_GOL_ObjectIDGet(pObj);
    if(objectID == ID_BTN1)
    {
        // check if button is pressed
        if(objMsg == GFX_GOL_BUTTON_ACTION_PRESSED)
        {
            // find slider pointer
            pScrollBar = (GFX_GOL_SCROLLBAR *)GFX_GOL_ObjectFind(ID_SLD1);
            // decrement the slider position
            GFX_GOL_ScrollBarPositionDecrement(pScrollBar);
            // redraw only the thumb
            GFX_GOL_ObjectStateSet(pScrollBar, GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE);
        }
    }
    if(objectID == ID_BTN2)
    {
        if(objMsg == GFX_GOL_BUTTON_ACTION_PRESSED)
        {
            // find slider pointer
            pScrollBar = (GFX_GOL_SCROLLBAR *)GFX_GOL_ObjectFind(ID_SLD1);
            // increment the slider position
            GFX_GOL_ScrollBarPositionIncrement(pScrollBar);
            // redraw only the thumb
            GFX_GOL_ObjectStateSet(pScrollBar, GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE);
        }
    }
    return (true);
}
```

Notice that the callback does not check if the scroll bar was affected by the message. Since the callback returns true, the scroll bar ACTIONSET_FUNC will be the one that will perform the state change of the scroll bar if the user directly interacted with the scroll bar.

Graphics Object Action Get and Action Set Functions

Each object has a predefined translated actions (see the GFX_GOL_TRANSLATED_ACTION enumeration). These actions are returned by the object's ACTIONGET_FUNC. The ACTIONGET_FUNC is called by the GFX_GOL_ObjectMessage function.

ACTIONGET_FUNC evaluates the message to determine if the user action affected the object. If the object is affected, it replies with one of the predefined translated action of the object.

For a button object the following translated actions are returned when the button is affected:

- GFX_GOL_BUTTON_ACTION_PRESSED - button is pressed
- GFX_GOL_BUTTON_ACTION_STILLPRESSED - button is continuously pressed
- GFX_GOL_BUTTON_ACTION_RELEASED - button is released
- GFX_GOL_BUTTON_ACTION_CANCELPRESS - button press canceled

When the button is not affected it returns:

- GFX_GOL_OBJECT_ACTION_INVALID - invalid message response

All objects returns GFX_GOL_OBJECT_ACTION_INVALID when they are not affected by the message.

the ACTIONSET_FUNC on the other hand performs the actual state change of the object. The draw state bits of the object are modified to indicate that the object will be redrawn in the next call to the GFX_GOL_ObjectListDraw.

How the Library Works

This section describes how the Graphics Library works and how the Primitive and Object layers are used.

Using the Primitive Layer

This section contains information on how to use the Primitive Layer of the Graphics Library.

Line Rendering

This section describes how to render lines.

Description

When rendering lines, two things need to be set:

- Set the color (GFX_ColorSet)
- Set the line style (GFX_LineStyleSet)

After the color and style is set, GFX_LineDraw can be called to render the line. GFX_LineDraw can be called multiple times to render lines that have the same color and line style. GFX_ColorSet and GFX_LineStyleSet will only be called when the color or the line style needs to be changed. For example, when one of the lines to be rendered has to change color, the GFX_ColorSet function must be called to change the color.

Note: Alpha blended lines as well as anti-aliased lines are not yet supported.

Polygon Rendering

This section describes how to render polygons.

Unfilled Polygon Rendering

This section describes how to render unfilled polygons.

Description

Unfilled polygons are rectangles and rounded rectangles. Circles is a special case of a rounded rectangle. Rendering unfilled polygons uses the line styles.

Similar to rendering lines, two things need to be set and one action performed:

- Set the color (GFX_ColorSet)
- Set the line style (GFX_LineStyleSet)
- Call the specific polygon function

Filled Polygon Rendering

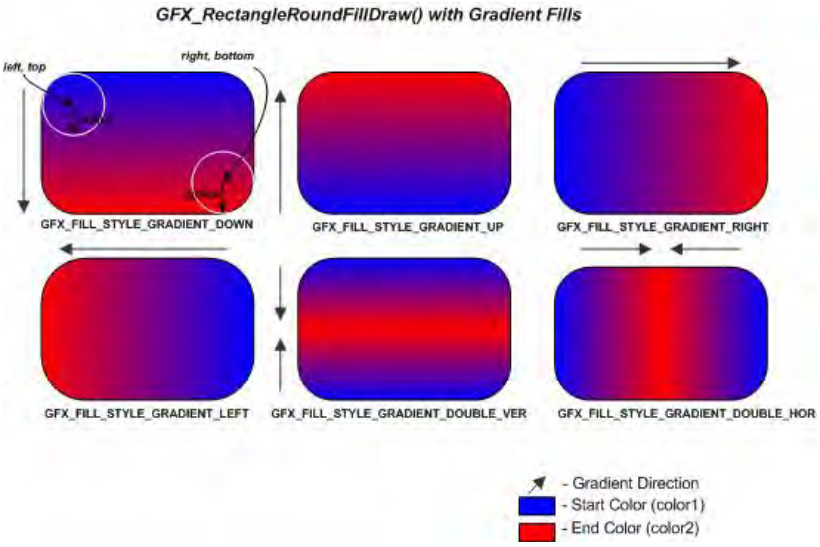
This section describes how to render filled polygons.

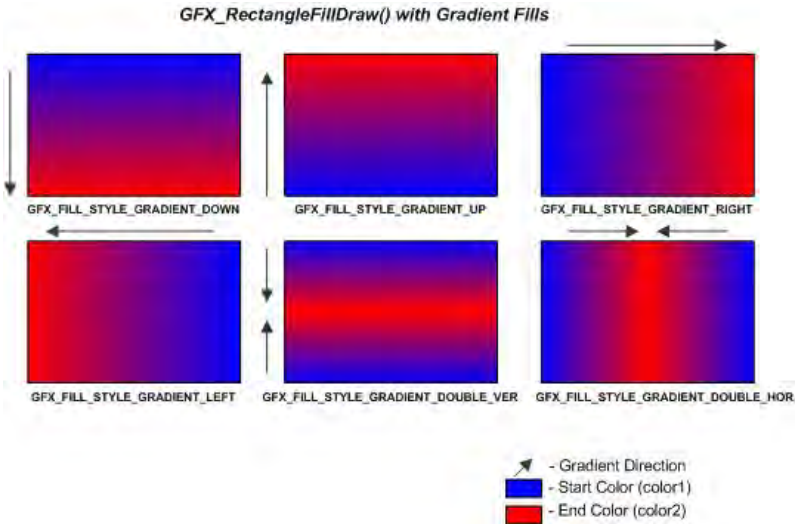
Description

Filled polygons are filled rectangles and filled rounded rectangles. Bar is a special case of a filled rectangle and filled circle is a special case of a filled rounded rectangle.

For filled polygons, the fill styles can be set:

Fill Style	Description
GFX_FILL_STYLE_NONE	The fill will be set to no color.
GFX_FILL_STYLE_ALPHA_COLOR	Sets that the fill style is alpha blended. The alpha blending effect will be dependent on the background type set (see GFX_BackgroundTypeSet). The level of alpha blending will depend on the last GFX_AlphaBlendingValueSet call.
GFX_FILL_STYLE_GRADIENT_DOWN	The effect is a gradient fill in the vertical down direction.
GFX_FILL_STYLE_GRADIENT_UP	The effect is a gradient fill in the vertical up direction.
GFX_FILL_STYLE_GRADIENT_RIGHT	The effect is a gradient fill in the horizontal right direction.
GFX_FILL_STYLE_GRADIENT_LEFT	The effect is a gradient fill in the horizontal left direction.
GFX_FILL_STYLE_GRADIENT_DOUBLE_VER	The effect is a gradient fill in the vertical direction but with two transitions.
GFX_FILL_STYLE_GRADIENT_DOUBLE_HOR	The effect is a gradient fill in the horizontal direction but with two transitions.





To render alpha blended filled polygons:

1. Set the color (GFX_ColorSet).
2. Set the background type (GFX_BackgroundTypeSet).
3. Set the alpha blending value (GFX_AlphaBlendingValueSet).
4. Set the fill style to alpha blending (GFX_FillStyleSet).
5. Call the specific polygon fill function:
   - GFX_RectangleFillDraw
   - GFX_RectangleRoundFillDraw
   - GFX_BarDraw
   - GFX_CircleFillDraw

To render gradient filled polygons:

1. Set the color (GFX_ColorSet).
2. Set the two gradient start and end colors (GFX_GradientColorSet).
3. Set the fill style to one of the gradient fill types (GFX_FillStyleSet).
4. Call the specific polygon fill function:
   - GFX_RectangleFillDraw
   - GFX_RectangleRoundFillDraw
   - GFX_BarDraw
   - GFX_CircleFillDraw

The following functions renders the same shapes:

- GFX_RectangleRoundDraw(x, y, x, y, radius) is equivalent to GFX_CircleDraw(x, y, radius).
- GFX_RectangleRoundFillDraw(x, y, x, y, radius) is equivalent to GFX_CircleFillDraw(x, y, radius).
- GFX_RectangleFillDraw(x1, y1, x2, y2) is equivalent to GFX_BarDraw(x1, y1, x2, y2).

Notes:

1. Alpha blended unfilled polygons are not yet supported.
2. Anti-aliased unfilled polygons are not yet supported.
3. Alpha blended, gradient fills are not yet supported.
4. Background is not needed when using gradient fills.

Text Rendering and Font Features

This section describes how to render text and strings and other features that are available for font resources.

Text Rendering

This section describes how to render text and strings.

Description

Text rendering in the Graphics Library requires a font table. The font table is considered one of the resources that is used in the library (see GFX_RESOURCE_FONT of the GFX_RESOURCE_HDR structure).

Once the font resource is available, text rendering can be performed by one of the following functions:

- GFX_TextCharDraw - Use to render a single character
- GFX_TextStringDraw - Use to render a string of characters
- GFX_TextStringBoxDraw - Use to render formatted string of characters

Character Rendering

To render a character:

1. Set the color (GFX_ColorSet).
2. Set the font resource to use (GFX_FontSet).
3. Set the location where the character will be rendered (GFX_TextCursorPositionSet).
4. Render the character (GFX_TextCharDraw).

String Rendering

To render a string:

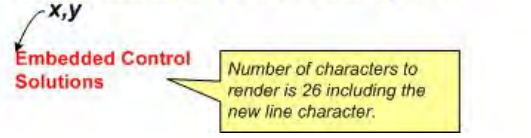
1. Set the color (GFX_ColorSet).
2. Set the font resource to use (GFX_FontSet).
3. Render the string (GFX_TextStringDraw).

GFX_TextStringDraw parameters sets the location where the string will be rendered. The length parameter allows versatility in rendering. When this parameter is set to '0', the rendering will terminate when the string terminator is encountered. When the length is non-zero, the rendering can be performed with the length as the terminating condition. This allows applications to render a portion of a longer string.

```
GFX_XCHAR Text[] =
"Microchip Technology Inc.
The Embedded Control
Solutions Company";
GFX_XCHAR *pString = Text;

GFX_TextStringDraw(x, y, pString, 0);
```


```
pString += 30;
GFX_TextStringDraw(x, y, pString, 26);
```



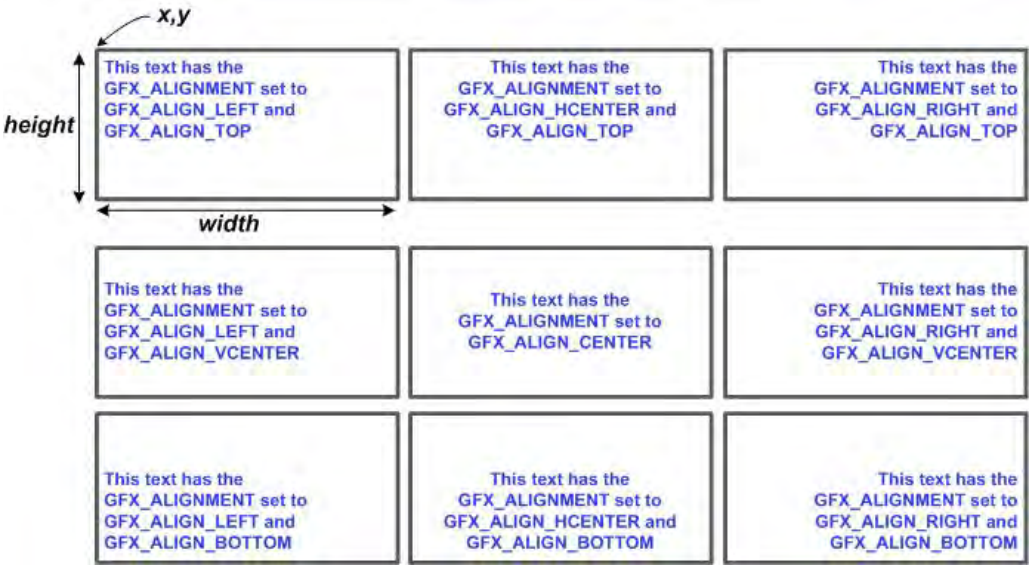
Multiple lines of strings are also supported. The string can contain the new line characters and rendering results in multiple lines of text. The use GFX_TextStringDraw will always render strings left aligned. To render text with alignment use GFX_TextStringBoxDraw.

Formatted String Rendering

To render string with formatting:

1. Set the color (GFX_ColorSet).
2. Set the font resource to use (GFX_FontSet).
3. Render the string (GFX_TextStringBoxDraw).

Parameters of GFX_TextStringBoxDraw determines the rectangular area where the string will be rendered. The alignment (see GFX_ALIGNMENT) specifies how the text will be aligned in the defined text area. The termination of the rendering is the same as the GFX_TextStringDraw where the length parameter can be used to determine how many characters will be rendered.



Added to these rendering functions, support functions are also provided for versatility:

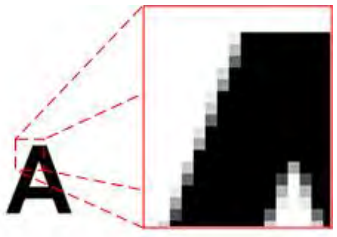
- GFX_TextStringWidthGet - Use to get the length of a string in pixels
- GFX_TextStringHeightGet - Use to get the height of the given font. All characters in a given font resource will have a constant height.
- GFX_FontSet - Use to set the font resource to use
- GFX_FontGet - Use to check the currently used font resource

Anti-aliased Fonts

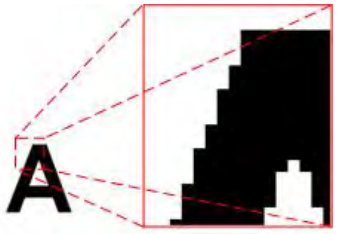
This section describes the anti-aliasing feature of the library.

Description

The Graphics Library supports rendering of anti-aliased font resource. Anti-aliasing is a technique used to make the edges of text appear smooth. This is useful especially with characters like 'A', 'O', etc., which has slant or curved lines. Since the pixels of the display are arranged in rectangular fashion, slant edges cannot be represented smoothly. To make them appear smooth, a pixel adjacent to the pixels is painted with an average of the foreground and background colors as depicted in the following figure.



When anti-aliasing is turned off, the pixels abruptly changes from background color to foreground color shown in the following figure. To implement anti-aliasing, adjacent pixels transitions from background to foreground color using 25% or 75% mid-color values from background to foreground colors. This feature in fonts will require roughly twice the size of memory storage required for font glyphs with no anti-aliasing.



Since the average of foreground and background colors needs to be calculated at runtime, the rendering of anti-aliased fonts take more time than rendering normal fonts.

Anti-aliasing transparency can be set to one of the following using GFX_FontAntiAliasSet (see GFX_FONT_ANTIALIAS_TYPE):

- GFX_FONT_ANTIALIAS_OPAQUE - Mid colors are calculated once while rendering each character which is ideal for rendering text over a constant background. This is the default setting of the library.
- GFX_FONT_ANTIALIAS_TRANSLUCENT - The mid values are calculated for every pixel and this feature is useful while rendering text over an image or on a non-constant color background.

As a result, rendering anti-aliased text takes longer with translucent type than compared to opaque type.

To use anti-aliasing, the font resource must contain glyphs that are anti-aliased enabled. The Graphics Resource Converter (GRC) tool that comes with the installation of the Graphics Library can generate such font resources.

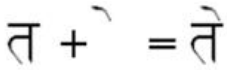
Note: An application may use anti-aliased and non-anti-aliased fonts at the same time.

Extended Glyphs

This section describes the extended glyphs feature of the library.

Description

Extended glyphs are needed to render characters of certain languages which use more than one byte to represent a single character. For example: Asian languages like Thai, Hindi, etc. In these character sets, more than one glyph overlaps each other to form a single character of that language, as shown in the following figure.



Extended glyph fonts can be generated using the Graphics Resource Converter (GRC) utility that is included with the installation of the Graphics Library.

Note: The fonts used with extended glyphs are normal ANSI fonts and not Unicode fonts.

Using the Graphics Object Layer

This section contains information on how to use the Graphics Object Layer of the Graphics Library.

Object Rendering and Style Schemes

This section describes how style schemes are used in rendering the objects of the Graphics Library.

Description

Style Scheme

All of the objects in the library utilize the style scheme to define how the object will be rendered on the screen. The style scheme defines the colors, images, fonts, and text that will be used. Choosing the appropriate colors and settings allows the object to be rendered in many ways.

Style schemes are applied to the objects by assigning the style scheme pointer to the object at creation. When the objects are rendered, the current settings of the style scheme will then be used to render the object. Objects cannot be created without a valid style scheme. This means that it cannot be null or undefined when the object is created. Style scheme can be modified anytime. It can be replaced using the GFX_GOL_ObjectStyleSchemeSet.

Style scheme is divided into these groups (See GFX_GOL_OBJ_SCHEME for details):

Text - The parameter that defines the text used in the object

- pFont - defines the font used in the style of the object

Colors - The parameters that defines the shape of the object

- EmbossDkColor - defines the dark emboss color
- EmbossLtColor - defines the light emboss color
- TextColor0 - defines the first text color option
- TextColor1 - defines the second text color option
- TextColorDisabled - defines the third text color option
- Color0 - defines the first face color option
- Color1 - defines the second face color option
- ColorDisabled - defines the third face color option
- EmbossDkColor - defines the dark emboss color

Background - The parameters that describes the background of the object. The background information defines how the object will be drawn with the background taken into account.

- CommonBkColor - Background color used to hide when the background type (GFX_BACKGROUND_TYPE) is set to GFX_BACKGROUND_COLOR or GFX_BACKGROUND_NONE
- CommonBkLeft - the horizontal starting position of the background
- CommonBkTop - the vertical starting position of the background
- CommonBkType - specifies the type of background to use
- pCommonBkImage - pointer to the background image used when the background type is set to GFX_BACKGROUND_IMAGE

Fill - The parameters that defines how the fill will be performed.

- fillStyle - specifies the fill style to be used (see GFX_FILL_STYLE)
- AlphaValue - alpha value used for alpha blending. This will be used then fillStyle is set to GFX_FILL_STYLE_ALPHA_COLOR.
- gradientStartColor - start color of the gradient fill. This will be used then fillStyle is set to any of the gradient fill styles.
- gradientEndColor - end color of the gradient fill. This will be used then fillStyle is set to any of the gradient fill styles.

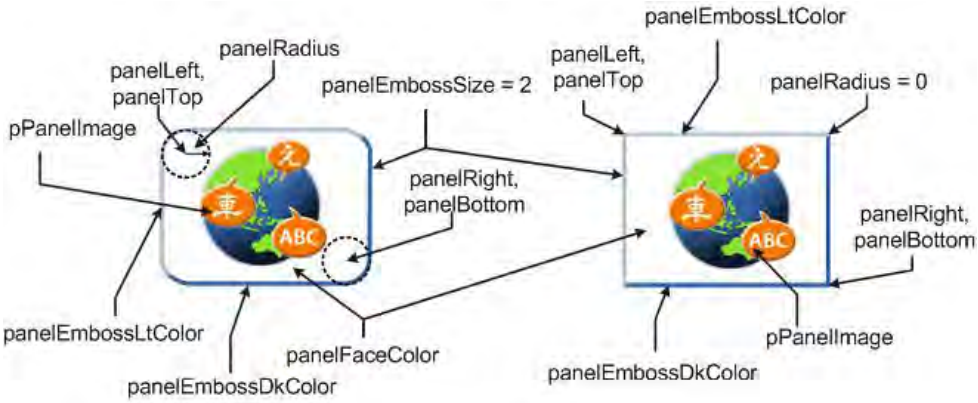
Style - The parameter that defines the 3-D effect of the object.

- EmbossSize - defines the emboss size of the panel for 3-D effect. This parameter should be set to '0' when not used.

Object Panel

Most of the objects in the Graphics Library utilizes a generic panel to define the object's shape. When using the provided objects in the library, the details of how the panel is utilized can be transparent to the users. In cases, where users defines their own objects the following detailed discussion of the panel is important.

A panel is a rectangular shape that is utilized by the objects to define its dimension. The panel is then drawn with the GOL_PANEL_PARAM to define the shape's colors. The colors are taken from the currently assigned style scheme to the object.



Panel Style

The assignment of the panel colors from the style scheme is shown in the following table.

Style Scheme to Panel Assignment

Style Scheme Parameter	Panel Parameter
EmbossDkColor	panelEmbossDkColor
EmbossLtColor	panelEmbossLtColor
Color0, Color1, ColorDisabled	panelFaceColor
fillStyle	panelFillStyle
EmbossSize	panelEmbossSize

TextColor0, TextColor1, TextColorDisabled, pFont, CommonBkColor, CommonBkLeft, CommonBkTop, CommonBkType, pCommonBkImage	Not set in the panel.
AlphaValue,	panelApha - use is dependent on the panelFillStyle setting
gradientStartColor,	panelGradientStartColor - use is dependent on the panelFillStyle setting
gradientEndColor	panelGradientEndColor - use is dependent on the panelFillStyle setting
N/A	panelImage - set by the object that supports images

The text parameters are not handled in the panel. These are handled by each object that supports text.

Refer to Panel Object for the summary of APIs that support panel rendering.

Background Feature

The Primitive Layer provides a global background information variable. This allows users to set a background and easily refresh parts of the screen that the background occupies (see GFX_BACKGROUND for details). The style scheme of the objects also allows association of the object to a background. This feature allows the objects to be easily refreshed on the screen.

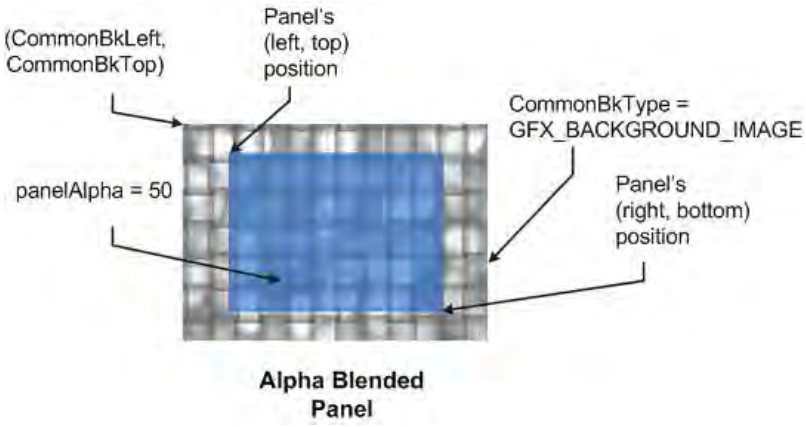
The requirement of the background feature is that the background should encompass the object. In other words, all the pixels of the object should be within the background dimension. Once this requirement is fulfilled, the parameters of the object that defines the location of the object (left, top, right bottom) is used to refresh the background pixels. Because of this Alpha blending, hiding or basic refresh of the object is optimized.

To add background information on the object the following style scheme parameters should be set:

- CommonBkLeft - The left most pixel location of the background
- CommonBkTop - The top most pixel location of the background
- CommonBkType - Should be set to one of the GFX_BACKGROUND_TYPE
- pCommonBkImage - If the CommonBkType is set to GFX_BACKGROUND_IMAGE, this should be set to the location of the image resource

Alpha Blending

When alpha blending is enabled, the object can be alpha blended with the background. Objects can also be removed from the screen easily by redrawing only the areas that the object occupies.



Alpha Blended Panel

Alpha Blending Objects

To implement an alpha blended object the following style scheme parameters should be set:

- fillStyle - Set to GFX_FILL_STYLE_ALPHA_COLOR
- AlphaValue - Set the alpha blending value. Set to 25, 50 or 75 when using primitive layer implementation of alpha blending

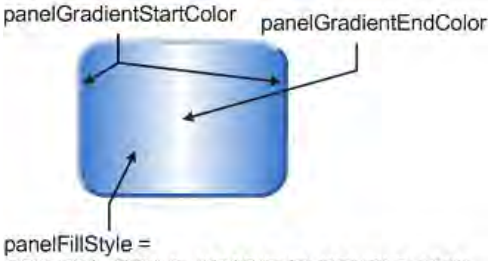
Gradient Fills

Objects can also be rendered with gradient fill on the panels. However, enabling gradient fill that are alpha blended is not supported yet. When AlphaValue is not 0 or 100 and fillStyle set to any of the gradient fill types, the alpha value will be ignored and assumed to be 100.

To implement an gradient filled object the following style scheme parameters should be set:

- fillStyle - Set to any of the gradient fill style (see GFX_FILL_STYLE for details):
  - GFX_FILL_STYLE_GRADIENT_DOWN
  - GFX_FILL_STYLE_GRADIENT_UP
  - GFX_FILL_STYLE_GRADIENT_RIGHT
  - GFX_FILL_STYLE_GRADIENT_LEFT
  - GFX_FILL_STYLE_GRADIENT_DOUBLE_VER
  - GFX_FILL_STYLE_GRADIENT_DOUBLE_HOR

- gradientStartColor - Set the gradient fill start color.
- gradientEndColor - Set the gradient fill end color.



Gradient Filled Panel

Notes:

1. Use of alpha blending on overlapping objects is not supported.
2. Alpha blended gradient fills is not supported.
3. Background feature can only be used if the background encompass the dimension of the object.
4. Certain objects do not support alpha blending and/or gradient fills. Refer to specific object documentation for details.

Double-Buffering Features

This section describes how to enable and use double buffering in the Graphics Primitive Layer.

Description

Manipulating pixels on the screen requires direct writes to the Frame Buffer. While these changes are being executed, the screen is also being refreshed. This means that the changes are displayed immediately as the frame buffer is being updated.

This is not a suitable scheme when the changes are slow, for example, decoding an image or having a large number of widgets on a screen. The display may appear slow in that case. One solution to this problem is to use a double-buffering scheme supported by the Microchip Graphics Library.

The double-buffering scheme does not render directly to the Frame Buffer, instead, rendering is performed to a separate buffer, called the Draw Buffer. After all the changes are made, the Draw Buffer and Frame Buffer are swapped. After the swap, the Draw Buffer becomes the new Frame Buffer and the Frame Buffer becomes the new Draw Buffer. This swapping appears as an instant change in the contents of the screen.

The time it takes to render to the Draw Buffer does not change; however, since rendering occurs in the buffer that is not shown on the screen, the user will not see the pixels being changed. Instead, what a user sees is the instant switch of the screen contents when the two buffers are swapped. The net effect is that the user experience is better since the user does not see the slow rendering.

The next rendering will now occur on the new Draw Buffer while the user views the screen with pixels from the Frame Buffer. The cycle of rendering to the Draw Buffer and displaying the Frame Buffer continues whenever new pixels are being drawn.

The use of the double-buffering feature will double the requirement of the memory for the buffers. Since this feature is dependent on the availability of the memory resources for buffers, the feature can only be used when the Display Driver Layer of the library supports double-buffering features.

The feature is enabled by default in configurations where double-buffering is supported. To disable the feature, define GFX_CONFIG_DOUBLE_BUFFERING_DISABLE in the `graphics_config.h` header file.

The following steps show how the double-buffering feature is enabled and used when rendering primitive shapes. However, for synchronization, there are two ways to perform it, which is differentiated in step 3. There are two ways to use the feature in rendering primitive shapes. Method A - use this method when the driver layer that implements the management of the swapping of buffers, uses some synchronization timing to perform the swap (usually timed to the vertical synchronization period of the display):

1. Enable the feature (GFX_DoubleBufferEnable)
2. Render primitive shapes. When rendering the shapes there are two ways to perform rendering and synchronization.
   - Render and register each new area
   - Render an area
   - Register the rectangular area that was modified (GFX_DoubleBufferAreaMark)
   - Render all primitives and rectangular areas and register the whole screen. This method will take more time since more pixels are being synchronized.
   - Render shapes and areas
   - Register the whole screen to be updated (GFX_DoubleBufferSyncAllStatusSet)
3. Synchronize the buffers so they will contain the same pixel information. There are also two ways to perform this.
   - Synchronize using a defined synchronization point. In this method, the driver is set up to perform the point of synchronization. For TFT displays, the vertical synchronization period is the best place to perform this since it will remove any possibility of showing portion of the screen with different data. This will happen when the refresh is in the middle of the screen and the buffers are swapped.
   - GFX_DoubleBufferSynchronizeRequest
   - Wait until the synchronization has finished using GFX_DoubleBufferSynchronizeStatusGet (i.e.,

```
while(GFX_DoubleBufferSynchronizeStatusGet() == GFX_FEATURE_ENABLED);
```

- Synchronize the buffers immediately
- GFX_DoubleBufferSynchronize

4. Repeat steps 2-3 for more rendering sequences.

If double buffering is disabled during run time, simply call GFX_DoubleBufferDisable. This function will perform one last synchronization to update the screen with recent rendering since the last synchronization.

Multi-instance Support

This section describes how to use two instances of the library.

Description

The application layer is considered a client of the Graphics Library. When GFX_Initialize is called it creates data structures for a single client referenced by the SYSTEM_MODULE_INDEX parameter (GFX client index). Each function of the library distinguishes each client through the use of the client index.

Multi-instance support is helpful when more than one driver and display is required. Multi-instance support is also helpful when two portions of a single display are to be rendered to separately.

Multi-instance support increases the code size of the application. If only one instance is to be used, configure GFX_INSTANCES_NUMBER to '0'. If multiple instances are to be used configure the macro accordingly.

```
#define GFX_INSTANCES_NUMBER 1
GFX_Initialize(GFX_INDEX_0, (SYS_MODULE_INIT*)&gfxInit1);
GFX_Initialize(GFX_INDEX_1, (SYS_MODULE_INIT*)&gfxInit2);
GFX_Tasks(GFX_INDEX_0);
GFX_Tasks(GFX_INDEX_1);
GFX_ColorSet(GFX_INDEX_0, CORAL);
GFX_ColorSet(GFX_INDEX_1, CYAN);
GFX_RectangleFillDraw(GFX_INDEX_0, 0,0,MaxXGet(),MaxYGet()/2);
GFX_RectangleFillDraw(GFX_INDEX_1,0,MaxYGet()/2,MaxXGet(), MaxYGet());
```

Surface Object

This section describes the Surface object.

Description

The Surface object provides a canvas onto which primitives and other objects can be drawn. The canvas can be the extent of the display or a smaller rectangular region. It can have a transparent, alpha-blended, or color background. It supports all draw and render states like all other GOL objects. An application can create multiple surfaces.

Unique to the Surface object is its callback function. The callback function allows the application to render application specific content onto the surface - which includes primitives. Because the Surface object is an Object, its drawing is rendered in sequence with all other objects.

Note: Refer to the Release Notes for any known limitations or restrictions. A PDF copy of the release notes is provided in the `<install-dir>/doc` folder of your installation.

Library Dependencies

This section lists any other modules (by name) that are required to build the Graphics module.

Description

To build the Graphics Library you must include the modules listed in the following table.

Library Name	Location (<install-dir>/framework/)
Interrupt Peripheral Library	peripheral/int
Clock System Service Library	system/clk
DMA System Service Library	system/dma
Common System Service Library	system/common
Timer System Service Library	system/tmr
Interrupt System Service Library	system/int
Ports System Service Library	system/ports
SPI Driver Library	driver/spi

Timer Driver Library	driver/tmr

Configuring the Library

Macros

	Name	Description
	GFX_CONFIG_ALPHABLEND_DISABLE	Macro that disables the Alpha Blending feature.
	GFX_CONFIG_COLOR_DEPTH	Macro that sets the color depth of the application.
	GFX_CONFIG_DOUBLE_BUFFERING_DISABLE	Macro disables the support for double buffering in rendering of pixels to the frame buffer.
	GFX_CONFIG_FOCUS_DISABLE	Macro that disables the focus feature in objects.
	GFX_CONFIG_FONT_ANTIALIASED_DISABLE	Macro that disables the use of anti-aliased fonts.
	GFX_CONFIG_FONT_CHAR_SIZE	Macro that sets the size of the characters used in the fonts.
	GFX_CONFIG_FONT_EXTERNAL_DISABLE	Macro that disables sourcing of font resources from external sources.
	GFX_CONFIG_FONT_FLASH_DISABLE	Macro that disables sourcing of font resources from internal flash memory.
	GFX_CONFIG_FONT_RAM_DISABLE	Macro that disables sourcing of font resources from RAM sources.
	GFX_CONFIG_GRADIENT_DISABLE	Macro that disables the gradient fill feature.
	GFX_CONFIG_IMAGE_EXTERNAL_DISABLE	Macro that disables sourcing of image resources from external sources.
	GFX_CONFIG_IMAGE_PADDING_DISABLE	Macro disables the padding of bits on images converted by the Graphics Resource Converter (GRC).
	GFX_CONFIG_IMAGE_RAM_DISABLE	Macro that disables sourcing of image resources from RAM sources.
	GFX_CONFIG_IPU_DECODE_DISABLE	Macro that disables RLE compression of images.
	GFX_CONFIG_PALETTE_DISABLE	Macro that disables the palette feature.
	GFX_CONFIG_PALETTE_EXTERNAL_DISABLE	Macro that disables the palette feature that are sourced in external resources.
	GFX_CONFIG_RLE_DECODE_DISABLE	Macro that disables RLE compression of images.
	GFX_CONFIG_TRANSPARENT_COLOR_DISABLE	Macro that disables the use of anti-aliased fonts.
	GFX_CONFIG_USE_KEYBOARD_DISABLE	This macro disables the keyboard support in objects.
	GFX_CONFIG_USE_TOUCHSCREEN_DISABLE	This macro disables the resistive touchscreen support in objects.
	GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE	Macro sets the size of the external font raster buffer.
	GFX_free	Macro that defines the free function for versatility when using Operating Systems.
	GFX_malloc	Macro that defines the malloc function for versatility when using Operating Systems.
	GFX_CONFIG_IMAGE_FLASH_DISABLE	Macro that disables sourcing of image resources from internal flash memory.
	GFX_INSTANCES_NUMBER	Macro to specify the number of possible client modules.

Description

The library can be configured to be used with a specific combination of features. This section enumerates the available configuration options as well as examples of combination of configurations.

GFX_CONFIG_ALPHABLEND_DISABLE Macro

Macro that disables the Alpha Blending feature.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_ALPHABLEND_DISABLE
```

Description

Macro: GFX_CONFIG_ALPHABLEND_DISABLE

Alpha blending feature is available to fill functions.

To disable the alpha blending feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_COLOR_DEPTH Macro

Macro that sets the color depth of the application.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_COLOR_DEPTH
```

Description

Macro: GFX_CONFIG_COLOR_DEPTH

This macro sets the color depth used in the application. The library functions are also set to use the set color depth. The valid values for the color depth are the following: 1, 4, 8, 16 and 24 bpp. Usage of each is dependent on the support available on the display driver used. See the specific display driver documentation to verify support of the chosen color depth.

If this macro is not set, a build error will be generated.

Remarks

None.

GFX_CONFIG_DOUBLE_BUFFERING_DISABLE Macro

Macro disables the support for double buffering in rendering of pixels to the frame buffer.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_DOUBLE_BUFFERING_DISABLE
```

Description

Macro: GFX_CONFIG_DOUBLE_BUFFERING_DISABLE

In cases where display drivers has the resources for more than one display buffer, double buffering can be implemented in the driver. This allows application to hide the rendering effects by rendering on a hidden buffer and displaying another one. Once the rendering is done, the buffer are swapped. This gives an instantaneous change in the buffers which makes the change in the screen contents fast. The display driver must support the feature for this mode to work.

In drivers where this feature is not supported, this macro has no effect. In drivers that supports this feature, adding this macro will disable the feature.

Remarks

None.

GFX_CONFIG_FOCUS_DISABLE Macro

Macro that disables the focus feature in objects.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_FOCUS_DISABLE
```

Description

Macro: GFX_CONFIG_FOCUS_DISABLE

This compile option allows keyboard input focus.

- GFX_GOL_ObjectFocusSet()
- GFX_GOL_ObjectFocusGet()

- GFX_GOL_ObjectFocusNextGet()
- GFX_GOL_ObjectFocusPrevGet()

functions will be available. Focus is also changed by touch screen.

To disable the focus feature in objects, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_FONT_ANTIALIASED_DISABLE Macro

Macro that disables the use of anti-aliased fonts.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_FONT_ANTIALIASED_DISABLE
```

Description

Macro: GFX_CONFIG_FONT_ANTIALIASED_DISABLE

Anti-aliased fonts are supported in the library. The fonts must be generated using the Graphics Resource Converter (GRC). The GRC is a utility used in the Graphics Library to generate application resources.

To disable the anti-aliased fonts feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_FONT_CHAR_SIZE Macro

Macro that sets the size of the characters used in the fonts.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_FONT_CHAR_SIZE
```

Description

Macro: GFX_CONFIG_FONT_CHAR_SIZE

This configuration sets the text character size used in the library.

To enable support for unicode fonts, GFX_CONFIG_FONT_CHAR_SIZE must be defined as 16 (size is 16 bits). For standard ascii fonts, this can be defined as 8. This changes the GFX_XCHAR definition. See GFX_XCHAR for details.

Set in configuration	GFX_XCHAR	Description
#define GFX_CONFIG_FONT_CHAR_SIZE 16	#define GFX_XCHAR uint16_t	Use multi-byte characters (0-2^16 range).
#define GFX_CONFIG_FONT_CHAR_SIZE 8	#define GFX_XCHAR uint8_t	Use byte characters (0-255 range).

If this macro is not set, the default size used is 8-bits.

```
// example to set characters to use 16 bits
#define GFX_CONFIG_FONT_CHAR_SIZE 16

// example to set characters to use 8 bits
#define GFX_CONFIG_FONT_CHAR_SIZE 8
```

Remarks

None.

GFX_CONFIG_FONT_EXTERNAL_DISABLE Macro

Macro that disables sourcing of font resources from external sources.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_FONT_EXTERNAL_DISABLE
```

Description

Macro: GFX_CONFIG_FONT_EXTERNAL_DISABLE

Font data can be placed in multiple locations. They can be placed in FLASH memory, RAM and external memory. To reduce code size, any one of these locations, when not used, can be disabled by defining the macros at build time.

This macro disables the use of font resources from external memory.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_FONT_FLASH_DISABLE Macro

Macro that disables sourcing of font resources from internal flash memory.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_FONT_FLASH_DISABLE
```

Description

Macro: GFX_CONFIG_FONT_FLASH_DISABLE

Font data can be placed in multiple locations. They can be placed in FLASH memory, RAM and external memory. To reduce code size, any one of these locations, when not used, can be disabled by defining the macros at build time.

This macro disables the use of font resources from internal flash memory.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_FONT_RAM_DISABLE Macro

Macro that disables sourcing of font resources from RAM sources.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_FONT_RAM_DISABLE
```

Description

Macro: GFX_CONFIG_FONT_RAM_DISABLE

Font data can be placed in multiple locations. They can be placed in FLASH memory, RAM and external memory. To reduce code size, any one of these locations, when not used, can be disabled by defining the macros at build time.

This macro disables the use of font resources from RAM.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_GRADIENT_DISABLE Macro

Macro that disables the gradient fill feature.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_GRADIENT_DISABLE
```

Description

Macro: GFX_CONFIG_GRADIENT_DISABLE

Gradient fill is available to fill functions.

To disable the gradient feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_IMAGE_EXTERNAL_DISABLE Macro

Macro that disables sourcing of image resources from external sources.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_IMAGE_EXTERNAL_DISABLE
```

Description

Macro: GFX_CONFIG_IMAGE_EXTERNAL_DISABLE

Images can be placed in multiple locations. They can be placed in FLASH memory, RAM and external memory. To reduce code size, any one of these locations, when not used, can be disabled by defining the macros at build time.

This macro disables the use of image resources from external memory.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_IMAGE_PADDING_DISABLE Macro

Macro disables the padding of bits on images converted by the Graphics Resource Converter (GRC).

File

gfx_config_template.h

C

```
#define GFX_CONFIG_IMAGE_PADDING_DISABLE
```

Description

Macro: GFX_CONFIG_IMAGE_PADDING_DISABLE

When converting images for use in the Graphics Library, the Graphics Resource Converter has an option to set the images to be padded or not padded. When images are padded, each horizontal line will start on a byte boundary.

Unpadded images allows the least resource space for an image. Unpadded images also allows support for display controllers with windowing and auto-increment features.

The images are created with padding by default. When the images are set to be not padded, application must define the macro in the configuration to correctly process the images.

Padded and unpadded images cannot be combined in one application.

Remarks

None.

GFX_CONFIG_IMAGE_RAM_DISABLE Macro

Macro that disables sourcing of image resources from RAM sources.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_IMAGE_RAM_DISABLE
```

Description

Macro: GFX_CONFIG_IMAGE_RAM_DISABLE

Images can be placed in multiple locations. They can be placed in FLASH memory, RAM and external memory. To reduce code size, any one of these locations, when not used, can be disabled by defining the macros at build time.

This macro disables the use of image resources from RAM.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_IPU_DECODE_DISABLE Macro

Macro that disables RLE compression of images.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_IPU_DECODE_DISABLE
```

Description

Macro: GFX_CONFIG_IPU_DECODE_DISABLE

Images can also be compressed using the DEFLATE algorithm. Using the drivers that supports DEFLATE (IPU of PIC24FJ256DA210 Family of devices), GFX_ImageDraw() will be able to render these images. This feature is enabled by default.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_PALETTE_DISABLE Macro

Macro that disables the palette feature.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_PALETTE_DISABLE
```

Description

Macro: GFX_CONFIG_PALETTE_DISABLE

The use of a palette is an option to run the application with 256 16, or 2 colors only. When using this feature, the color depth set for the application should be 8, 4 or 1 bpp (see GFX_CONFIG_COLOR_DEPTH for color depth settings).

To disable the palette feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_PALETTE_EXTERNAL_DISABLE Macro

Macro that disables the palette feature that are sourced in external resources.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_PALETTE_EXTERNAL_DISABLE
```

Description

Macro: GFX_CONFIG_PALETTE_EXTERNAL_DISABLE

Similar to fonts and images, Palettes are considered resources. The use of palette also allow the application to locate the palette resources in external sources. This macro disables the code that implements externally sourced palettes.

To disable the palette to be sourced from external resources, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_RLE_DECODE_DISABLE Macro

Macro that disables RLE compression of images.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_RLE_DECODE_DISABLE
```

Description

Macro: GFX_CONFIG_RLE_DECODE_DISABLE

Images that have 8bpp or 4 bpp color depth has the option to be RLE compressed. GFX_ImageDraw() will be able to render these images. This feature is enabled by default.

```
// example to use images that are NOT RLE encoded and sourced from
// internal flash only
#define GFX_CONFIG_RLE_DECODE_DISABLE
#define GFX_CONFIG_IMAGE_EXTERNAL_DISABLE
#define GFX_CONFIG_IMAGE_RAM_DISABLE
```

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_TRANSPARENT_COLOR_DISABLE Macro

Macro that disables the use of anti-aliased fonts.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_TRANSPARENT_COLOR_DISABLE
```

Description

Macro: GFX_CONFIG_TRANSPARENT_COLOR_DISABLE

Transparent color is a feature in GFX_ImageDraw() where pixels that equals the color set in the transparent color variable will not be rendered. This is useful in rendering rounded icons or images to the screen with complex background.

To disable the transparent color in images, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_USE_KEYBOARD_DISABLE Macro

This macro disables the keyboard support in objects.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_USE_KEYBOARD_DISABLE
```

Description

Macro: GFX_CONFIG_USE_KEYBOARD_DISABLE

Depending on the input devices used, messages that objects will process are encoded in the input device drivers. The keyboard is one of the input devices that is supported in selected objects.

This support is enabled by default. To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_CONFIG_USE_TOUCHSCREEN_DISABLE Macro

This macro disables the resistive touchscreen support in objects.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_USE_TOUCHSCREEN_DISABLE
```

Description

Macro: GFX_CONFIG_USE_TOUCHSCREEN_DISABLE

Depending on the input devices used, messages that objects will process are encoded in the input device drivers. The resistive touchscreen is one of the input devices that is supported in selected objects.

This support is enabled by default. To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE Macro

Macro sets the size of the external font raster buffer.

File

gfx_config_template.h

C

```
#define GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE
```

Description

Macro: GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE

This macro sets the size of the external font raster buffer. This buffer is used to store the character glyph when the glyph is being read from external source. The value that is needed will be calculated by the Graphics Resource Converter (GRC). The value will be shown in the external resource reference c file that the GRC generates.

```
// example to set the buffer size.
// value is taken from the output of the GRC
#define GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE 100
```

A warning will be issued when building the application when:

- when the configuration do not define this macro
- when the defined value is less than the required size

This macro will have no effect when fonts that are sourced externally is not used.

Remarks

None.

GFX_free Macro

Macro that defines the free function for versatility when using Operating Systems.

File

gfx_config_template.h

C

```
#define GFX_free(pObj) free(pObj)
```

Description

Macro: GFX_free()

This macro definition allows the application to replace the free function to an equivalent function implemented in the operating system used.

When using the library with object layer and without any replacement functions for the malloc function, this macro must be defined as:

```
#define GFX_free(pObj)      free(pObj)
```

Remarks

None.

GFX_malloc Macro

Macro that defines the malloc function for versatility when using Operating Systems.

File

gfx_config_template.h

C

```
#define GFX_malloc(size) malloc(size)
```

Description

Macro: GFX_malloc()

This macro definition allows the application to replace the malloc function to an equivalent function implemented in the operating system used.

When using the library with object layer and without any replacement functions for the malloc function, this macro must be defined as:

```
#define GFX_malloc(size)    malloc(size)
```

Remarks

None.

GFX_CONFIG_IMAGE_FLASH_DISABLE Macro

Macro that disables sourcing of image resources from internal flash memory.

File

gfx_config_template.h

C

```
#define GFX_CONFIG_IMAGE_FLASH_DISABLE
```

Description

Macro: GFX_CONFIG_IMAGE_FLASH_DISABLE

Images can be placed in multiple locations. They can be placed in FLASH memory, RAM and external memory. To reduce code size, any one of these locations, when not used, can be disabled by defining the macros at build time.

This macro disables the use of image resources from internal flash memory.

To disable this feature, add this macro in the configuration.

Remarks

None.

GFX_INSTANCES_NUMBER Macro

Macro to specify the number of possible client modules.

File

gfx_config_template.h

C

```
#define GFX_INSTANCES_NUMBER
```

Description

Macro: GFX_INSTANCES_NUMBER

This macro sets the maximum number of possible instances that can be created.

To set the maximum number of clients, add this macro in the configuration and the appropriate number.

Remarks

The number will affect memory usage.

None.

Configuration Examples

This section shows example of configuration combination when using the library.

Example 1

Configuration example 1.

Description

- Non-blocking is enabled on accelerated functions
- Alpha blending
- Gradient fills
- Palette is not used
- Focus on objects that supports it is enabled
- Double buffering is disabled
- Characters size is 16 bits to support UNICODE characters
- Fonts are sourced from two places:
  - Internal Flash
  - External memory
  - Fonts from RAM are disabled
- Fonts that are anti-aliased are enabled
- Fonts sourced from external memory are allocated 51 bytes of buffer
- Images are sourced from two places:
  - Internal Flash
  - External memory
  - Fonts from RAM is disabled
- Images that are RLE encoded are enabled
- Images that are DEFLATE (for IPU module) encoded are enabled
- Images can be rendered with transparent color feature
- Touch screen support is enabled
- Keyboard support is enabled
- GFX_malloc and GFX_free functions are used to manage memory for object layer

```
// Note: Configuration to disable the features are commented out
// to show the specific macros that disables them.

//#define GFX_CONFIG_NONBLOCKING_DISABLE
//#define GFX_CONFIG_ALPHABLEND_DISABLE
//#define GFX_CONFIG_GRADIENT_DISABLE
//#define GFX_CONFIG_FOCUS_DISABLE
#define GFX_CONFIG_PALETTE_DISABLE
#define GFX_CONFIG_PALETTE_EXTERNAL_DISABLE
#define GFX_CONFIG_DOUBLE_BUFFERING_DISABLE

#define GFX_CONFIG_FONT_CHAR_SIZE 16
#define GFX_CONFIG_COLOR_DEPTH 16

//#define GFX_CONFIG_FONT_FLASH_DISABLE
//#define GFX_CONFIG_FONT_EXTERNAL_DISABLE
```

```
#define GFX_CONFIG_FONT_RAM_DISABLE
//#define GFX_CONFIG_FONT_ANTIALIASED_DISABLE

//#define GFX_CONFIG_IMAGE_FLASH_DISABLE
//#define GFX_CONFIG_IMAGE_EXTERNAL_DISABLE
#define GFX_CONFIG_IMAGE_RAM_DISABLE

//#define GFX_CONFIG_RLE_DECODE_DISABLE
//#define GFX_CONFIG_IPU_DECODE_DISABLE

//#define GFX_CONFIG_TRANSPARENT_COLOR_DISABLE
#define GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE 51

//#define GFX_CONFIG_USE_TOUCHSCREEN_DISABLE
//#define GFX_CONFIG_USE_KEYBOARD_DISABLE

#define GFX_malloc(size) malloc(size)
#define GFX_free(pObj) free(pObj)
```

Example 2

Configuration example 2.

Description

- All primitive functions are blocking
- The following are disabled:
  - Alpha blending
  - Gradient fills
  - Palette
  - Focus on objects
  - Double buffering
  - Anti-aliased fonts
  - RLE and DEFLATE encoded images
  - Transparent color
  - Touchscreen
  - Keyboard
- Characters size is 8 bits
- Images are sourced only from internal Flash
- Fonts are sourced only from internal Flash and since fonts are sourced from internal Flash, the external font raster buffer is not needed
- Object layer is not used so GFX_malloc and GFX_free are never referenced

```
// Note: Configuration to disable the features are commented out
//       to show the specific macros that disables them.

#define GFX_CONFIG_NONBLOCKING_DISABLE
#define GFX_CONFIG_ALPHABLEND_DISABLE
#define GFX_CONFIG_GRADIENT_DISABLE
#define GFX_CONFIG_FOCUS_DISABLE
#define GFX_CONFIG_PALETTE_DISABLE
#define GFX_CONFIG_PALETTE_EXTERNAL_DISABLE
#define GFX_CONFIG_DOUBLE_BUFFERING_DISABLE

#define GFX_CONFIG_FONT_CHAR_SIZE 8
#define GFX_CONFIG_COLOR_DEPTH 16

//#define GFX_CONFIG_FONT_FLASH_DISABLE
#define GFX_CONFIG_FONT_EXTERNAL_DISABLE
#define GFX_CONFIG_FONT_RAM_DISABLE
#define GFX_CONFIG_FONT_ANTIALIASED_DISABLE

//#define GFX_CONFIG_IMAGE_FLASH_DISABLE
#define GFX_CONFIG_IMAGE_EXTERNAL_DISABLE
#define GFX_CONFIG_IMAGE_RAM_DISABLE

#define GFX_CONFIG_RLE_DECODE_DISABLE
#define GFX_CONFIG_IPU_DECODE_DISABLE
```

```
#define GFX_CONFIG_TRANSPARENT_COLOR_DISABLE
//#define GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE 51

#define GFX_CONFIG_USE_TOUCHSCREEN_DISABLE
#define GFX_CONFIG_USE_KEYBOARD_DISABLE

//#define GFX_malloc(size)    malloc(size)
//#define GFX_free(pObj)      free(pObj)
```

Building the Library

This section lists the files that are available in the Graphics Library.

Description

The following three tables list and describe the header (`.h`) and source (`.c`) files that implement this library. The parent folder for these files is `<install-dir>/framework/gfx`.

Interface File(s)

This table lists and describes the header files that must be included (i.e., using `#include`) by any code that uses this library.

Source File Name	Description
gfx.h	Graphics Library initialization header file.

Required File(s)

All of the required files listed in the following table are automatically added into the MPLAB X IDE project by the MHC when the library is selected for use.

The following table lists and describes the source and header files that must *always* be included in the MPLAB X IDE project to build the LCC Library.

Source File Name	Description
/src/drv_gfx_lcc_int.c	LCC Driver Library small buffer internal memory framebuffer driver.
/src/drv_gfx_lcc_int_pe.c	LCC Driver Library large buffer internal memory framebuffer driver.
/src/drv_gfx_lcc_ext.c	LCC Driver Library external memory framebuffer driver for wqvga.
/src/drv_gfx_lcc_ext_wvga.c	LCC Driver Library external memory framebuffer driver for wvga.

the following table lists and describes the source and header files that must *always* be included in the MPLAB X IDE project to build the Graphics Library.

Source File Name	Description
gfx.h	Graphics Library initialization header file.
/src/gfx.c	Graphics Library initialization implementation.
gfx_primitive.h	Primitive Layer run-time header file.
/src/gfx_primitive.c	Primitive Layer run-time implementation.
gfx_gol.h	Object Layer run-time header.
/src/gfx_gol.c	Object Layer run-time implementation.

Optional File(s)

This table lists and describes the source and header files that may optionally be included for the Graphics Library if required for the desired implementation.

Source File Name	Description
gfx_colors.h	Default color definitions
gfx_colors_w3.h	W3 color definitions
gfx_colors_x11.h	X11 color definitions
gfx_gol_button.h	Button widget header
/src/gfx_gol_button.c	Button widget implementation

gfx_gol_check_box.h	Check Box widget header
/src/gfx_gol_check_box.c	Check Box widget implementation
gfx_gol_custom_control.h	Custom control widget header
/src/gfx_gol_custom_control.c	Custom control widget implementation
gfx_gol_digital_meter.h	Digital meter widget header
/src/gfx_gol_digital_meter.c	Digital meter widget header
gfx_gol_edit_box.h	Edit Box widget header
/src/gfx_gol_edit_box.c	Edit Box widget implementation
gfx_gol_group_box.h	Group Box widget header
/src/gfx_gol_group_box.c	Group Box widget implementation
gfx_gol_list_box.h	List Box widget implementation
/src/gfx_gol_list_box.c	List Box widget implementation
gfx_gol_meter.h	Analog meter widget header
/src/gfx_gol_meter_c	Analog meter widget implementation
gfx_gol_picture.h	Picture widget header
/src/gfx_gol_picture.c	Picture widget implementation
gfx_gol_progress_bar.h	Progress Bar widget header
/src/gfx_gol_progress_bar.c	Progress Bar widget implementation
gfx_gol_radio_button.h	Radio Button widget header
/src/gfx_gol_radio_button.c	Radio Button widget implementation
gfx_gol_scan_codes.h	Keyboard codes header
gfx_gol_scheme.h	Object style scheme header
/src/gfx_gol_scheme_default.c	Object style scheme implementation
gfx_gol_scroll_bar.h	Scroll bar widget header
/src/gfx_gol_scroll_bar.c	Scroll bar widget implementation
gfx_gol_static_text.h	Static Text widget header
/src/gfx_gol_static_text.c	Static Text widget implementation
/src/gfx_gol_surface.h	Surface object widget header
/src/gfx_gol_surface.c	Surface object widget implementation
/src/gfx_gol_text_entry.c	Text Entry widget header
/src/gfx_gol_text_entry.c	Text Entry widget implementation
gfx_gol_window.h	Window widget header
/src/gfx_gol_window.c	Window widget implementation
gfx_image_decoder.h	Image decoder header
/src/gfx_image_decoder.c	Image decoder implementation
gfx_types_font.h	Font types
gfx_types_image.h	Image types
gfx_types_macros.h	GFX system wide types and macros
gfx_types_palette.h	Palette types
gfx_types_resource.h	Resource types for images fonts palette and binary types

Module Dependencies

The File System Service Library depends on the following modules:

- Graphics Driver Library

Library Interface

This section describes the Application Programming Interface (API) functions of the Graphics Library.

Refer to each section for a detailed description.

Graphics Primitive Layer

This section describes the API and usage of the Graphics Library Primitive Layer.

Graphics Primitive Layer API

Application Programming Interface functions for the Primitive Layer of the Graphics Library.

Initialization Functions

Functions

	Name	Description
	GFX_PRIMITIVE_Initialize	Initialize the GFX PRIMITIVE HAL Library.
	GFX_PRIMITIVE_Deinitialize	Deinitializes the specified instance of the GFX Module.
	GFX_PRIMITIVE_Close	Closes an instance of the primitive layer specified by handle.
	GFX_PRIMITIVE_Open	Opens a client instance of the primitive layer specified by index.
	GFX_ScreenClear	Clears the screen to the currently set color (GFX_ColorSet()).

Description

Graphics Library Primitive Layer initialization API.

GFX_PRIMITIVE_Initialize Function

Initialize the GFX PRIMITIVE HAL Library.

File

gfx_primitive.h

C

```
SYS_MODULE_OBJ GFX_PRIMITIVE_Initialize(const SYS_MODULE_INDEX moduleIndex, const SYS_MODULE_INIT * const moduleInit);
```

Returns

If successful, returns a valid handle to a primitive object. Otherwise, it returns SYS_MODULE_OBJ_INVALID.

Description

This function initializes the GFX Library Primitive HAL.

Remarks

This routine must be called before other GFX PRIMITIVE HAL functions.

This function initialize the Graphics Library primitive layer. The following default settings are set when this function is called.

1. font - Set to NULL. GFX_FontSet() must be called prior to any text rendering.
2. line type - Set to GFX_LINE_TYPE_THIN_SOLID (see GFX_LINE_STYLE).
3. fill type - Set to GFX_FILL_TYPE_COLOR (see GFX_FILL_STYLE).
4. text anti-alias type - Set to GFX_FONT_ANTIALIAS_OPAQUE (see GFX_FONT_ANTIALIAS_TYPE). This only affects fonts with anti-aliasing enabled.
5. Set transparent color feature in image draw functions

to be disabled.

6. Set alpha blending value to 100 (or no alpha blending) if

alpha-blending feature is enabled.

7. Set background information to no background.

This function does not clear the screen and does not assign any color to the currently set color. Application should set the color and clear the screen.

Preconditions

None.

Example

```
GFX_PRIMITIVE_INIT gfxInit;
SYS_MODULE_OBJ primObj;

// GFX Module initialization
primObj = GFX_PRIMITIVE_Initialize(GFX_PRIMITIVE_INDEX_0, &gfxInit);

if (SYS_MODULE_OBJ_INVALID == primObj)
{
    // Handle error
}
```

Parameters

Parameters	Description
moduleIndex	client instance request.
moduleInit	initialization data for the instance.

Function

void GFX_PRIMITIVE_Initialize (const SYS_MODULE_INDEX moduleIndex,

const SYS_MODULE_INIT * const moduleInit);

GFX_PRIMITIVE_Deinitialize Function

Deinitializes the specified instance of the GFX Module.

File

gfx_primitive.h

C

```
void GFX_PRIMITIVE_Deinitialize(SYS_MODULE_OBJ moduleObject);
```

Returns

None.

Description

Deinitializes the specified instance of the GFX Module. All internal data structures will be reset.

Remarks

Once the Initialize operation has been called, the Deinitialize operation must be called before the Initialize operation can be called again. This routine will not block waiting for hardware.

Preconditions

Function GFX_GOL_Initialize should have been called before calling this function.

Example

```
SYS_MODULE_OBJ      object;     //  Returned from GFX_PRIMITIVE_Initialize
SYS_STATUS          status;


GFX_PRIMITIVE_Deinitialize(object);

status = GFX_Status(object);
if (SYS_MODULE_DEINITIALIZED != status)
{
    // Check again later if you need to know
    // when the driver is deinitialized.
}
```

Parameters

Parameters	Description
object	GFX layer object handle, returned from the GFX_PRIMITIVE_Initialize routine

Function

void GFX_PRIMITIVE_Deinitialize(SYS_MODULE_OBJ object)

GFX_PRIMITIVE_Close Function

Closes an instance of the primitive layer specified by handle.

File

gfx_primitive.h

C

```
void GFX_PRIMITIVE_Close(GFX_PRIMITIVE_HANDLE handle);
```

Description

This function closes the client instance.

Parameters

Parameters	Description
handle	primitive handle

Function

void GFX_PRIMITIVE_Close(GFX_PRIMITIVE_HANDLE handle)

GFX_PRIMITIVE_Open Function

Opens a client instance of the primitive layer specified by index.

File

gfx_primitive.h

C

```
GFX_PRIMITIVE_HANDLE GFX_PRIMITIVE_Open(const SYS_MODULE_INDEX index);
```

Returns

GFX_PRIMITIVE HANDLE - handle to the primitive layer.

Description

This function opens a client instance. THe user must have called GFX_PRIMITIVE_Initialize prior to calling this functions. It opens the graphics driver.

Parameters

Parameters	Description
index	instance of the driver

Function

GFX_PRIMITIVE_HANDLE GFX_PRIMITIVE_Open(const SYS_MODULE_INDEX index)

GFX_ScreenClear Function

Clears the screen to the currently set color (GFX_ColorSet()).

File

gfx_primitive.h

C

```
GFX_STATUS GFX_ScreenClear(SYS_MODULE_INDEX gfxIndex);
```

Returns

The status of the screen clearing. GFX_STATUS_SUCCESS - screen was cleared. GFX_STATUS_FAILURE - screen is not yet cleared

Description

This function clears the screen with the current color and sets the line cursor position to (0, 0).

If color is not set, before this function is called, the output is undefined.

If the function returns GFX_STATUS_FAILURE, clearing is not yet finished. Application must call the function again to continue the clearing.

Preconditions

Color must be set by GFX_ColorSet().

None.

Example

```
GFX_ColorSet(GFX_INDEX_0, BLACK);
GFX_ScreenClear(GFX_INDEX_0);
```

Function

GFX_STATUS GFX_ScreenClear(SYS_MODULE_INDEX gfxIndex)

Display Functions

Functions

	Name	Description
	GFX_MaxXGet	This function returns the maximum horizontal pixel coordinate.
	GFX_MaxYGet	This function returns the maximum vertical pixel coordinate.

Description

GFX_MaxXGet Function

This function returns the maximum horizontal pixel coordinate.

File

gfx_primitive.h

C

```
inline uint16_t GFX_MaxXGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Maximum horizontal pixel coordinate.

Description

This function returns the maximum horizontal pixel coordinate of the chosen display. The value returned is equal to the horizontal dimension minus 1. The horizontal dimension is also dependent on the rotation of the screen. Rotation can be set in the hardware configuration.

Preconditions

None.

Example

None.

****

#if (DISP_ORIENTATION == 90) #define GFX_MaxXGet() (DISP_VER_RESOLUTION - 1) #elif (DISP_ORIENTATION == 270) #define GFX_MaxXGet() (DISP_VER_RESOLUTION - 1) #elif (DISP_ORIENTATION == 10) #define GFX_MaxXGet() (DISP_HOR_RESOLUTION - 1) #elif (DISP_ORIENTATION == 180) #define GFX_MaxXGet() (DISP_HOR_RESOLUTION - 1) #endif

Function

uint16_t GFX_MaxXGet(void)

GFX_MaxYGet Function

This function returns the maximum vertical pixel coordinate.

File

gfx_primitive.h

C

```
inline uint16_t GFX_MaxYGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Maximum vertical pixel coordinate.

Description

This function returns the maximum vertical pixel coordinate of the chosen display. The value returned is equal to the vertical dimension minus 1. The vertical dimension is also dependent on the rotation of the screen. Rotation can be set in the hardware configuration.

Preconditions

None.

Example

None.

****

#if (DISP_ORIENTATION == 90) #define GFX_MaxYGet() (DISP_HOR_RESOLUTION - 1) #elif (DISP_ORIENTATION == 270) #define GFX_MaxYGet() (DISP_HOR_RESOLUTION - 1) #elif (DISP_ORIENTATION == 10) #define GFX_MaxYGet() (DISP_VER_RESOLUTION - 1) #elif (DISP_ORIENTATION == 180) #define GFX_MaxYGet() (DISP_VER_RESOLUTION - 1) #endif

Function

uint16_t GFX_MaxYGet(void)

Line Rendering Functions

Functions

	Name	Description
	GFX_LineDraw	This function renders a line from x1,y1 to x2,y2 using the currently set line style (see GFX_LineStyleSet()).
	GFX_LinePositionRelativeSet	This function sets the line cursor to a new position relative to the current position.
	GFX_LinePositionSet	This function sets the line cursor to a new position.
	GFX_LinePositionXGet	This function returns the current x position of the line cursor.
	GFX_LinePositionYGet	This function returns the current y position of the line cursor.
	GFX_LineToDraw	This function renders a line from current line cursor position (x,y) to (x2,y2) using the currently set line style (see GFX_LineStyleSet()).
	GFX_LineToRelativeDraw	This function renders a line from current line cursor position (x,y) to (x+dX,y+dY) using the currently set line style (see GFX_LineStyleSet()).

Description

Graphics Library Primitive Layer line rendering API.

GFX_LineDraw Function

This function renders a line from x1,y1 to x2,y2 using the currently set line style (see GFX_LineStyleSet()).

File

gfx_primitive.h

C

```
GFX_STATUS GFX_LineDraw(SYS_MODULE_INDEX gfxIndex, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2);
```

Returns

Status of the line rendering. GFX_STATUS_SUCCESS - line rendering done. GFX_STATUS_FAILURE - line rendering is not done.

Description

This function renders a line from x1,y1 to x2,y2 using the currently set line style set by GFX_LineStyleSet(). The color used is the color set by the last call to GFX_ColorSet().

If x1,y1 and/or x2,y2 is not on the frame buffer, then the behavior is undefined. If color is not set, before this function is called, the output is undefined.

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_DOTTED);
GFX_LineDraw(GFX_INDEX_0, 10, 10,100,  10);
```

Parameters

Parameters	Description
x1	x coordinate of the line start point.
y1	y coordinate of the line start point.
x2	x coordinate of the line end point.
y2	y coordinate of the line end point.

Function

GFX_STATUS GFX_LineDraw(

SYS_MODULE_INDEX gfxIndex,

uint16_t x1,

uint16_t y1,

uint16_t x2,

uint16_t y2)

GFX_LinePositionRelativeSet Function

This function sets the line cursor to a new position relative to the current position.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_LinePositionRelativeSet(SYS_MODULE_INDEX gfxIndex, int16_t x, int16_t y);
```

Returns

Status of the relative line position set. GFX_STATUS_SUCCESS - relative line position set done. GFX_STATUS_FAILURE - relative line position set not done.

Description

This function sets the line cursor to a new (x,y) position relative to the current cursor position. The new position is calculated by (x+dX, y + dY).

Line cursor is used as a starting point of the line rendered by the GFX_LineToDraw() and GFX_LineToRelativeDraw() functions. Note that the parameters dX and dY are signed integers. This allows the new line cursor position to be placed to any direction from the current line cursor position.

If (x+dX) and/or (y+dY) results in a position that is not on the frame buffer, then the behavior of GFX_LineToDraw() and GFX_LineToRelativeDraw() functions are undefined. If color is not set, before this function is called, the output is undefined.

Preconditions

None.

Example

See GFX_LineToDraw().

Parameters

Parameters	Description
dX	the offset for the x position that will define the new x-coordinate position of the line cursor.
dY	the offset for the y position that will define the new y-coordinate position of the line cursor.

Function

GFX_STATUS GFX_LinePositionRelativeSet(
SYS_MODULE_INDEX gfxIndex,
int16_t dX,
int16_t dY)

GFX_LinePositionSet Function

This function sets the line cursor to a new position.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_LinePositionSet(SYS_MODULE_INDEX gfxIndex, uint16_t x, uint16_t y);
```

Returns

Status of the line position set. GFX_STATUS_SUCCESS - line position set done. GFX_STATUS_FAILURE - line position set not done.

Description

This function sets the line cursor to a new (x,y) position. Line cursor is used as a starting point of the line rendered by the GFX_LineToDraw() and GFX_LineToRelativeDraw() functions.

If x and/or y does not lie on the frame buffer, then the behavior of GFX_LineToDraw() and GFX_LineToRelativeDraw() functions are undefined.

Preconditions

None.

Example

See GFX_LinePositionXGet().

Parameters

Parameters	Description
x	new x coordinate position of the line cursor.
y	new y coordinate position of the line cursor.

Function

GFX_STATUS GFX_LinePositionSet(
SYS_MODULE_INDEX gfxIndex,
uint16_t x,
uint16_t y)

GFX_LinePositionXGet Function

This function returns the current x position of the line cursor.

File

gfx_primitive.h

C

```
int16_t GFX_LinePositionXGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current line cursor x position.

Description

This function returns the current x position of the line cursor. Line cursor is used as a starting point of the line rendered by the GFX_LineToDraw() and GFX_LineToRelativeDraw() functions.

Preconditions

None.

Example

```
// implementation of the GFX_LineToRelativeDraw()
GFX_STATUS GFX_LineToRelativeDraw(
                                GFX_INDEX_0,
                                int16_t dX,
                                int16_t dY)
{
    return (GFX_LineDraw(
                            GFX_INDEX_0,
                            GFX_LinePositionXGet(gfxIndex),
                            GFX_LinePositionYGet(gfxIndex),
                            GFX_LinePositionXGet(gfxIndex) + dX,
                            GFX_LinePositionYGet(gfxIndex) + dY));
}
```

Function

int16_t GFX_LinePositionXGet(SYS_MODULE_INDEX gfxIndex)

GFX_LinePositionYGet Function

This function returns the current y position of the line cursor.

File

gfx_primitive.h

C

```
int16_t GFX_LinePositionYGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current line cursor y position.

Description

This function returns the current y position of the line cursor. Line cursor is used as a starting point of the line rendered by the GFX_LineToDraw() and GFX_LineToRelativeDraw() functions.

Preconditions

None.

Example

See GFX_LinePositionXGet().

Function

int16_t GFX_LinePositionYGet(SYS_MODULE_INDEX gfxIndex)

GFX_LineToDraw Function

This function renders a line from current line cursor position (x,y) to (x2,y2) using the currently set line style (see GFX_LineStyleSet()).

File

gfx_primitive.h

C

```
GFX_STATUS GFX_LineToDraw(SYS_MODULE_INDEX gfxIndex, int16_t x2, int16_t y2);
```

Returns

Status of the line rendering. GFX_STATUS_SUCCESS - line rendering done. GFX_STATUS_FAILURE - line rendering is not done.

Description

This function renders a line from current line cursor position (x,y) to (x2,y2) using the currently set line style set by GFX_LineStyleSet(). The color used is the color set by the last call to GFX_ColorSet().

If x2 and/or y2 does not lie on the frame buffer, then the behavior is undefined. If color is not set, before this function is called, the output is undefined.

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THICK_DOTTED);

GFX_LinePositionSet(GFX_INDEX_0, 100, 100);
GFX_LineToDraw(GFX_INDEX_0, 100,  10);
GFX_LinePositionRelativeSet(GFX_INDEX_0, -90,  0);
GFX_LineToDraw(GFX_INDEX_0, 100,  10);
GFX_LinePositionRelativeSet(GFX_INDEX_0,   0, 90);
GFX_LineToDraw(GFX_INDEX_0,  10, 100);
GFX_LinePositionRelativeSet(GFX_INDEX_0,  90,-90);
GFX_LineToDraw(GFX_INDEX_0,  10, 100);
GFX_LinePositionRelativeSet(GFX_INDEX_0,   0,-90);
GFX_LineToDraw(GFX_INDEX_0, 100, 100);
GFX_LinePositionRelativeSet(GFX_INDEX_0,  -90, 0);
GFX_LineToDraw(GFX_INDEX_0, 10,  10);
```

Parameters

Parameters	Description
x2	x coordinate of the line end point.
y2	y coordinate of the line end point.

Function

GFX_STATUS GFX_LineToDraw(
SYS_MODULE_INDEX gfxIndex,
int16_t x2,
int16_t y2)

GFX_LineToRelativeDraw Function

This function renders a line from current line cursor position (x,y) to (x+dX,y+dY) using the currently set line style (see GFX_LineStyleSet()).

File

gfx_primitive.h

C

```
GFX_STATUS GFX_LineToRelativeDraw(SYS_MODULE_INDEX gfxIndex, int16_t dX, int16_t dY);
```

Returns

Status of the line rendering. GFX_STATUS_SUCCESS - line rendering done. GFX_STATUS_FAILURE - line rendering is not done.

Description

This function renders a line from current line cursor position (x,y) to (x+dX,y+dY) using the currently set line style set by GFX_LineStyleSet(). The color used is the color set by the last call to GFX_ColorSet(). Note that the parameters dX and dY are signed integers. This allows the line to be drawn from the line cursor to any direction.

If (x+dX) and/or (y+dY) results in a position that is not on the frame buffer, then the behavior is undefined. If color is not set, before this function is called, the output is undefined.

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
```

```
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_DOTTED);

GFX_LinePositionSet(GFX_INDEX_0, 100, 100);
GFX_LineToRelativeDraw(GFX_INDEX_0,   0, -90);
GFX_LineToRelativeDraw(GFX_INDEX_0, -90,   0);
GFX_LineToRelativeDraw(GFX_INDEX_0,  90,  90);
GFX_LineToRelativeDraw(GFX_INDEX_0, -90,   0);
GFX_LineToRelativeDraw(GFX_INDEX_0,  90, -90);
GFX_LinePositionSet(GFX_INDEX_0, 10, 10);
GFX_LineToRelativeDraw(GFX_INDEX_0,  0,  90);
```

Parameters

Parameters	Description
dX	the offset for the x starting position that will define the x-coordinate of the end of the line.
dY	the offset for the y starting position that will define the y-coordinate of the end of the line.

Function

GFX_STATUS GFX_LineToRelativeDraw(

SYS_MODULE_INDEX gfxIndex,

int16_t dX,

int16_t dY)

Polygon Rendering Functions

Functions

	Name	Description
	GFX_CircleDraw	This function renders a circular shape using the currently set line style and color.
	GFX_PolygonDraw	This function renders a polygon using the currently set line style and color.
	GFX_RectangleDraw	This function renders a rectangular shape using the currently set line style and color.
	GFX_RectangleRoundDraw	This function renders a rounded corner rectangular shape using the currently set line style and color.

Description

Graphics Library Primitive Layer polygon rendering API.

GFX_CircleDraw Function

This function renders a circular shape using the currently set line style and color.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_CircleDraw(SYS_MODULE_INDEX gfxIndex, uint16_t x, uint16_t y, uint16_t radius);
```

Returns

Status of the circle rendering. GFX_STATUS_SUCCESS - circle rendering done. GFX_STATUS_FAILURE - circle rendering is not done.

Description

This function renders a circular shape using the center (x,y) and radius. The shape is rendered using the currently set line style by GFX_LineStyleSet(). The color used is the color set by the last call to GFX_ColorSet().

When x,y falls outside the buffer, the behavior is undefined. When color is not set before this function is called, the behavior is undefined. When any of the following x+radius, x-radius, y+radius and y-radius falls outside the buffer, the behavior is undefined.

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
// draw a circle using bright red solid line
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
```

```
GFX_CircleDraw(GFX_INDEX_0, 50, 50, 40);
```

Parameters

Parameters	Description
x	defines the x-coordinate position of the center of the circle.
y	defines the y-coordinate position of the center of the circle.
radius	defines the radius of the circle.

Function

GFX_STATUS GFX_CircleDraw(

SYS_MODULE_INDEX gfxIndex,

uint16_t x,

uint16_t y,

uint16_t radius)

GFX_PolygonDraw Function

This function renders a polygon using the currently set line style and color.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_PolygonDraw(SYS_MODULE_INDEX gfxIndex, uint16_t numPoints, uint16_t * polyPoints);
```

Returns

Status of the polygon rendering. GFX_STATUS_SUCCESS - polygon rendering done. GFX_STATUS_FAILURE - polygon rendering is not done.

Description

This function renders a polygon using the currently set line style (see GFX_LineStyleSet()) and color (see GFX_ColorSet()). The shape of the polygon is determined by the polygon points (an ordered array of x,y pairs) where the pair count is equal to the parameter sides.

If any of the x,y pairs do not lie on the frame buffer, then the behavior is undefined. If color is not set, before this function is called, the output is undefined.

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
uint16_t OpenShapeXYPoints[6] = {10, 10, 20, 10, 20, 20};
uint16_t ClosedShapeXYPoints[8] = {10, 10, 20, 10, 20, 20, 10, 10};

GFX_ColorSet(GFX_INDEX_0, WHITE);                        // set color
SetLineType(GFX_INDEX_0, GFX_LINE_STYLE_THIN_DOTTED);    // set line style
GFX_PolygonDraw(GFX_INDEX_0, 3, OpenShapeXYPoints); // draw an open shape
GFX_PolygonDraw(GFX_INDEX_0, 4, ClosedShapeXYPoints);    // draw a  closed shape
```

Parameters

Parameters	Description
sides	the number of sides of the polygon.
pPoints	Pointer to the array of polygon points. The array defines the x,y points of the polygon. The sequence should be x0, y0, x1, y1, x2, y2, ... xn, yn where n is the # of polygon sides.

Function

GFX_STATUS GFX_PolygonDraw(

SYS_MODULE_INDEX gfxIndex,

uint16_t sides,

uint16_t *pPoints)

GFX_RectangleDraw Function

This function renders a rectangular shape using the currently set line style and color.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_RectangleDraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right,
uint16_t bottom);
```

Returns

Status of the rectangle rendering. GFX_STATUS_SUCCESS - rectangle rendering done. GFX_STATUS_FAILURE - rectangle rendering is not done.

Description

This function renders a rectangular shape using the given left, top, right and bottom parameters to define the shape dimension. The shape is rendered using the currently set line style by GFX_LineStyleSet(). The color used is the color set by the last call to GFX_ColorSet().

The rendering of this shape becomes undefined when any one of the following is true:

- Any of the following pixel locations left,top, right,bottom falls outside the frame buffer.
- Color is not set, before this function is called.
- right < left
- bottom < top

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
// draw a bright red rectangle
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
GFX_RectangleDraw(GFX_INDEX_0, 30, 30, 88, 88, 15);

// draw a bright blue round rectangle
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_DASHED);
GFX_ColorSet(GFX_INDEX_0, BRIGHTBLUE);
GFX_RectangleRoundDraw(GFX_INDEX_0, 130, 30, 188, 88, 15);
```

Parameters

Parameters	Description
left	defines the left most pixel of the shape.
top	defines the top most pixel of the shape.
right	defines the right most pixel of the shape.
bottom	defines the bottom most pixel of the shape.

Function

GFX_STATUS GFX_RectangleDraw(

SYS_MODULE_INDEX gfxIndex,

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom)

GFX_RectangleRoundDraw Function

This function renders a rounded corner rectangular shape using the currently set line style and color.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_RectangleRoundDraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right,
uint16_t bottom, uint16_t radius);
```

Returns

Status of the rectangle rendering. GFX_STATUS_SUCCESS - rectangle rendering done. GFX_STATUS_FAILURE - rectangle rendering is not done.

Description

This function renders a rectangular shape with rounded corner using the given left, top, right, bottom and radius parameters to define the shape dimension. radius defines the rounded corner shape. The shape is rendered using the currently set line style by GFX_LineStyleSet(). The color used is the color set by the last call to GFX_ColorSet().

Left most pixel location is defined by left - radius. Top most pixel location is defined by top - radius. Right most pixel location is defined by right + radius. Bottom most pixel location is defined by bottom + radius. When radius = 0, there are no rounded corners. In this case (left,top) will define the left, top corner and (right,bottom) will define the right, bottom corner of the shape.

When left = right and top = bottom, with radius > 0, a circular object is drawn. When left < right and top < bottom and radius = 0, a rectangular object is drawn.

The rendering of this shape becomes undefined when any one of the following is true:

- Any of the following pixel locations left-rad , top-rad, right+rad, bottom+rad falls outside the frame buffer.
- Color is not set, before this function is called.
- right < left
- bottom < top

Preconditions

Color must be set by GFX_ColorSet(). Line style must be set by GFX_LineStyleSet().

Example

```
// draw a bright red rectangle
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
GFX_RectangleDraw(GFX_INDEX_0, 30, 30, 88, 88, 15);

// draw a bright blue round rectangle
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_DASHED);
GFX_ColorSet(GFX_INDEX_0, BRIGHTBLUE);
GFX_RectangleRoundDraw(GFX_INDEX_0, 130, 30, 188, 88, 15);
```

Parameters

Parameters	Description
left	Along with rad (left - rad), defines the left most pixel of the shape.
top	Along with rad (top - rad), defines the top most pixel of the shape.
right	Along with rad (right + rad), defines the right most pixel of the shape.
bottom	Along with rad (bottom + rad), defines the bottom most pixel of the shape.
radius	defines the rounded corners. When this is set to zero, there will be no rounded corners.

Function

GFX_STATUS GFX_RectangleRoundDraw(

SYS_MODULE_INDEX gfxIndex,

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom,

uint16_t radius)

Polygon Fill Rendering Functions

Functions

	Name	Description
	GFX_BarDraw	This function renders a bar shape using the currently set fill style and color.
	GFX_CircleFillDraw	This function renders a filled circle shape using the currently set fill style and colors.
	GFX_RectangleFillDraw	This function renders a filled rectangular shape using the currently set fill style and colors.
	GFX_RectangleRoundFillDraw	This function renders a filled rectangular shape with round corners using the currently set fill style and colors.

Description

Graphics Library Primitive Layer polygon fill rendering API.

GFX_BarDraw Function

This function renders a bar shape using the currently set fill style and color.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_BarDraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right, uint16_t
bottom);
```

Returns

Status of the bar rendering. GFX_STATUS_SUCCESS - bar rendering done. GFX_STATUS_FAILURE - bar rendering failed.

Description

This function renders a bar shape with the currently set fill style (See GFX_FillStyleGet() and GFX_FillStyleSet() for details of fill style):

- solid color - when the fill style is set to GFX_FILL_STYLE_COLOR
- alpha blended fill - when the fill style is set to GFX_FILL_STYLE_ALPHA_COLOR.

Any other selected fill style will be ignored and will assume a solid color fill will be used. The parameters left, top, right bottom will define the shape dimension.

When fill style is set to GFX_FILL_STYLE_ALPHA_COLOR, the bar can also be rendered with an option to select the type of background. GFX_BACKGROUND_NONE - the bar will be rendered with no alpha blending. GFX_BACKGROUND_COLOR - the bar will be alpha blended with the currently set background color. GFX_BACKGROUND_IMAGE - the bar will be alpha blended with the currently set background image. GFX_BACKGROUND_DISPLAY_BUFFER - the bar will be alpha blended with the current contents of the frame buffer.

The background type is set by the GFX_BackgroundTypeSet().

The rendering of this shape becomes undefined when any one of the following is true:

- Any of the following pixel locations left,top or right,bottom falls outside the frame buffer.
- Colors are not set before this function is called.
- When right < left
- When bottom < top
- When pixel locations defined by left, top and/or right, bottom are not on the frame buffer.

Preconditions

Fill style must be set by GFX_FillStyleSet() when alpha blended fill is desired. Color must be set by GFX_ColorSet().

Example

```
// assume RED is a macro that define GFX_COLOR types
GFX_STATUS status;
// assume BackGroundImage is a valid image already draw
// on the screen
GFX_RESOURCE_HDR *pMyBackgroundImage = &BackGroundImage;

// render a RED bar
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_COLOR);
GFX_ColorSet(GFX_INDEX_0, RED);
status = GFX_BarDraw(GFX_INDEX_0, 10, 110, 100, 200);

// render an alpha blended bar with
// the current contents of the frame buffer
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_ALPHA_COLOR);
GFX_BackgroundTypeSet(GFX_INDEX_0, GFX_BACKGROUND_DISPLAY_BUFFER);
GFX_ColorSet(GFX_INDEX_0, RED);
status = GFX_BarDraw(GFX_INDEX_0, 10, 110, 100, 200);

// render an alpha blended bar with a background image
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_ALPHA_COLOR);

// color value here has no effect since the background set
// is type GFX_BACKGROUND_IMAGE
GFX_BackgroundSet(GFX_INDEX_0, 0, 0, pMyBackGroundImage, 0);
```

```
GFX_BackgroundTypeSet(GFX_INDEX_0, GFX_BACKGROUND_IMAGE);
GFX_ColorSet(GFX_INDEX_0, RED);
status = GFX_BarDraw(GFX_INDEX_0, 10, 110, 100, 200);
```

Parameters

Parameters	Description
left	defines the left most pixel of the shape.
top	defines the top most pixel of the shape.
right	defines the right most pixel of the shape.
bottom	defines the bottom most pixel of the shape.

Function

GFX_STATUS GFX_BarDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom)

GFX_CircleFillDraw Function

This function renders a filled circle shape using the currently set fill style and colors.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_CircleFillDraw(SYS_MODULE_INDEX gfxIndex, uint16_t x, uint16_t y, uint16_t radius);
```

Returns

Status of the circle rendering. GFX_STATUS_SUCCESS - circle fill rendering done. GFX_STATUS_FAILURE - circle fill rendering is not done.

Description

This function renders a filled circle shape with the currently set fill style (see GFX_FILL_STYLE) with the given left, top, right and bottom parameters to define the shape dimension. The shape is rendered depending on the fill style. If a flat color is used, color must be set (see GFX_ColorSet()) before calling this function. If gradient color is used, gradient start and end color must be set (see GFX_GradientColorSet()) before calling this function. After the fill style and colors are set, multiple calls to this function can be performed.

The rendering of this shape becomes undefined when any one of the following is true:

- Any of the following pixel locations left,top or right,bottom falls outside the frame buffer.
- Fill style is not set (GFX_FillStyleSet(), before this function is called.
- Colors are not set before this function is called.
- When the center defined by x,y is not on the frame buffer.

Preconditions

Fill style must be set by GFX_FillStyleSet(). Color must be set by GFX_ColorSet().

Example

```
// assume BLUE and RED are macros that define GFX_COLOR types
GFX_STATUS status;

GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_GRADIENT_UP);
GFX_GradientColorSet(GFX_INDEX_0, BLUE, RED);
status = GFX_CircleFillDraw(GFX_INDEX_0, 50, 110, 150, 200, 20);
if (status == GFX_STATUS_SUCCESS)
    // Filled circle was drawn.
else
    // Filled circle is not drawn or not yet
    // finished rendering. To finish the rendering call the
    // function again with the same parameters.
```

Parameters

Parameters	Description
x	defines the x-coordinate position of the center of the circle.
y	defines the y-coordinate position of the center of the circle.
radius	defines the radius of the circle.

Function

GFX_STATUS GFX_CircleFillDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t x,
uint16_t y,
uint16_t radius)

GFX_RectangleFillDraw Function

This function renders a filled rectangular shape using the currently set fill style and colors.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_RectangleFillDraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right,
uint16_t bottom);
```

Returns

Status of the rectangle rendering. GFX_STATUS_SUCCESS - rectangle fill rendering done. GFX_STATUS_FAILURE - rectangle fill rendering is not done.

Description

This function renders a filled rectangular shape with the currently set fill style (see GFX_FILL_STYLE) with the given left, top, right, and bottom parameters to define the shape dimension. The shape is rendered depending on the fill style. If a flat color is used, color must be set (see GFX_ColorSet()) before calling this function. If gradient color is used, gradient start and end color must be set (see GFX_GradientColorSet()) before calling this function. After the fill style and colors are set, multiple calls to this function can be performed.

The rendering of this shape becomes undefined when any one of the following is true:

- Any of the following pixel locations left,top or right,bottom falls outside the frame buffer.
- Fill style is not set (GFX_FillStyleSet(), before this function is called.
- Colors are not set before this function is called.
- When right < left
- When bottom < top
- When pixel locations defined by left, top and/or right, bottom are not on the frame buffer.

Preconditions

Fill style must be set by GFX_FillStyleSet(). Color must be set by GFX_ColorSet().

Example

```
// assume BLUE and RED are macros that define GFX_COLOR types
GFX_STATUS status;

// render a rectangle with gradient colors from BLUE to RED
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_GRADIENT_LEFT);
GFX_GradientColorSet(GFX_INDEX_0, BLUE, RED);
status = GFX_RectangleFillDraw(GFX_INDEX_0, 10, 110, 100, 200);

// render an alpha blended rounded rectangle with
// the current contents of the frame buffer
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_ALPHA_COLOR);
GFX_AlphaBlendingValueSet(GFX_INDEX_0, 50);
GFX_BackgroundTypeSet(GFX_INDEX_0, GFX_BACKGROUND_DISPLAY_BUFFER);
GFX_ColorSet(GFX_INDEX_0, RED);
status = GFX_RectangleRoundFillDraw(GFX_INDEX_0, 10, 110, 100, 200, 10);

// render an alpha blended rectangle with an image
```

```
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_ALPHA_COLOR);
GFX_AlphaBlendingValueSet(5GFX_INDEX_0, 0);
GFX_BackgroundTypeSet(GFX_INDEX_0, GFX_BACKGROUND_IMAGE);
GFX_ColorSet(GFX_INDEX_0, RED);
status = GFX_RectangleFillDraw(GFX_INDEX_0, 10, 110, 100, 200);

// render a plain RED rounded rectangle
GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_COLOR);
GFX_ColorSet(GFX_INDEX_0, RED);
status = GFX_RectangleFillDraw(GFX_INDEX_0, 10, 110, 100, 200);
```

Parameters

Parameters	Description
left	defines the left most pixel of the shape.
top	defines the top most pixel of the shape.
right	defines the right most pixel of the shape.
bottom	defines the bottom most pixel of the shape.

Function

GFX_STATUS GFX_RectangleFillDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom)

GFX_RectangleRoundFillDraw Function

This function renders a filled rectangular shape with round corners using the currently set fill style and colors.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_RectangleRoundFillDraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t
right, uint16_t bottom, uint16_t radius);
```

Returns

Status of the rectangle rendering. GFX_STATUS_SUCCESS - rectangle fill rendering done. GFX_STATUS_FAILURE - rectangle fill rendering is not done.

Description

This function renders a filled rounded rectangular shape with the currently set fill style (see GFX_FILL_STYLE) with the given left, top, right, bottom and radius parameters to define the shape dimension. The shape is rendered depending on the fill style. If a flat color is used, color must be set (see GFX_ColorSet()) before calling this function. If gradient color is used, gradient start and end color must be set (see GFX_GradientColorSet()) before calling this function. After the fill style and colors are set, multiple calls to this function can be performed.

When left = right and top = bottom, with radius > 0, a circular object is drawn. When left < right and top < bottom and radius = 0, a rectangular object is drawn.

The rendering of this shape becomes undefined when any one of the following is true:

- Any of the following pixel locations left,top or right,bottom falls outside the frame buffer.
- Fill style is not set (GFX_FillStyleSet(), before this function is called.
- Colors are not set before this function is called.
- When right < left
- When bottom < top
- When pixel locations defined by left, top and/or right, bottom are not on the frame buffer.

Preconditions

Fill style must be set by GFX_FillStyleSet(). Color must be set by GFX_ColorSet().

Example

```
// assume BLUE and RED are macros that define GFX_COLOR types
GFX_STATUS status;
```

```
GFX_FillStyleSet(GFX_FILL_STYLE_GRADIENT_DOUBLE_VER);
GFX_GradientColorSet(BLUE, RED);
status = GFX_RectangleFillDraw(50, 110, 150, 200, 20);
if (status == GFX_STATUS_SUCCESS)
    // Filled rounded rectangle shape was drawn.
else
    // Filled rounded rectangle shape is not drawn or not yet
    // finished rendering. To finish the rendering call the
    // function again with the same parameters.
```

Parameters

Parameters	Description
left	defines the left most pixel of the shape.
top	defines the top most pixel of the shape.
right	defines the right most pixel of the shape.
bottom	defines the bottom most pixel of the shape.
radius	defines the radius of the rounded corner. A zero value will result in a rectangular shape drawn.

Function

GFX_STATUS GFX_RectangleRoundFillDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t radius)

Image Rendering Functions

Functions

	Name	Description
	GFX_ImageDraw	This function renders an image to the frame buffer.
	GFX_ImageHeaderGet	This function fills the given bitmap header with the image's header information.
	GFX_ImageHeightGet	This function returns the height of the given image.
	GFX_ImagePartialDraw	This function renders a portion of an image to the frame buffer.
	GFX_ImageWidthGet	This function returns the width of the given image.

Description

Graphics Library Primitive Layer image rendering API.

GFX_ImageDraw Function

This function renders an image to the frame buffer.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_ImageDraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, GFX_RESOURCE_HDR * pImage);
```

Returns

Status of the image rendering. GFX_STATUS_SUCCESS - image rendering done. GFX_STATUS_FAILURE - image rendering failed.

Description

This function renders an image to the frame buffer with the left-top corner of the image located at given left, top parameters.

The rendering of this shape becomes undefined when any one of the following is true:

- left, top pixel position falls outside the frame buffer.
- pointer is not properly initialized to a GFX_RESOURCE_HDR object.

Preconditions

None.

Example

```
// assume the backgroundImage has dimension of 320x240 pixels.
GFX_RESOURCE_HDR *pBackgroundImage;

pBackgroundImage = (GFX_RESOURCE_HDR *)&backgroundImage;

// Render the image starting from (10,10) x,y position

// corner of the image
GFX_ImageDraw(  GFX_INDEX_0, 10, 10,
                pBackgroundImage);
```

Parameters

Parameters	Description
left	Horizontal starting position of the image.
top	Vertical starting position of the image.
pImage	Pointer to the image to be rendered.

Function

GFX_STATUS GFX_ImageDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t left,
uint16_t top,
GFX_RESOURCE_HDR *pImage)

GFX_ImageHeaderGet Function

This function fills the given bitmap header with the image's header information.

File

gfx_primitive.h

C

```
inline void GFX_ImageHeaderGet(GFX_RESOURCE_HDR * pImage, GFX_MCHP_BITMAP_HEADER * pBitmapHdr);
```

Returns

None.

Description

This function fills the given bitmap header with the image's header information. This function results in an undefined behavior if the pointer to the image is invalid.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pImage	Pointer to the image.
pBitmap	Pointer to the destination of the header information.

Function

GFX_STATUS GFX_ImageHeaderGet(
GFX_RESOURCE_HDR *pImage,
GFX_MCHP_BITMAP_HEADER *pBitmapHdr)

GFX_ImageHeightGet Function

This function returns the height of the given image.

File

gfx_primitive.h

C

```
inline int16_t GFX_ImageHeightGet(GFX_RESOURCE_HDR * pImage);
```

Returns

The image height in pixels.

Description

This function returns the height of the given image in pixels. This function results in an undefined behavior if the pointer to the image is invalid.

This function return value is undefined if the given pointer does not point to a valid image resource.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pImage	Pointer to the image.

Function

GFX_STATUS GFX_ImageHeightGet(
GFX_RESOURCE_HDR *pImage)

GFX_ImagePartialDraw Function

This function renders a portion of an image to the frame buffer.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_ImagePartialDraw(SYS_MODULE_INDEX gfxIndex, uint16_t destination_x, uint16_t destination_y,
uint16_t source_x_offset, uint16_t source_y_offset, uint16_t source_width, uint16_t source_height,
GFX_RESOURCE_HDR * pImage);
```

Returns

Status of the image rendering. GFX_STATUS_SUCCESS - image rendering done. GFX_STATUS_FAILURE - image rendering failed.

Description

This function renders an image or a portion of an image to the frame buffer with the top, left corner of the image located at destination_x, destination_y. To render a full image, source_x_offset, source_y_offset, width and height are set to all 0. Using the actual image's width and height will also work if these are known. If they are not known, avoid the extra extra step to get the image's width and height by assigning 0 to the parameters. Another way to render a full image is to use the GFX_ImageDraw(). The image to be rendered is defined by the pointer pImage.

The rendering of this shape becomes undefined when any one of the following is true:

- destination_x, destination_y pixel position falls outside the frame buffer.
- source_x_offset, source_y_offset results in a starting position beyond the image's dimension.
- source_width and/or source_height is larger than the actual image's width and height.
- pointer is not properly initialized to a GFX_RESOURCE_HDR object.

Preconditions

None.

Example

```
// assume the backgroundImage has dimension of 320x240 pixels.
GFX_RESOURCE_HDR *pBackgroundImage;

pBackgroundImage = (GFX_RESOURCE_HDR *)&backgroundImage;

// Render only 1/4 of the image. Render the lower right
// corner of the image
GFX_ImagePartialDraw(   GFX_INDEX_0, 10, 10,
                        320/2,
                        240/2,
                        320/2,
                        240/2,
                        pBackgroundImage);
```

Parameters

Parameters	Description
destination_x	Horizontal starting position of the image.
destination_y	Vertical starting position of the image.
source_x_offset	Horizontal offset in pixels of the starting position of the partial area of the image to be rendered. Set to 0 along with source_y_offset when rendering the full image.
source_y_offset	Vertical offset in pixels of the starting position of the partial area of the image to be rendered. Set to 0 along with source_x_offset when rendering the full image.GFX_ImageDraw
source_width	The width of the partial area of the image to be rendered. Set to 0 along with source_height when rendering the full image.
source_height	The height of the partial area of the image to be rendered. Set to 0 along with source_width when rendering the full image.
pImage	Pointer to the image to be rendered.

Function

GFX_STATUS GFX_ImagePartialDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t destination_x,
uint16_t destination_y,
uint16_t source_x_offset,
uint16_t source_y_offset,
uint16_t source_width,
uint16_t source_height,
GFX_RESOURCE_HDR *pImage)

GFX_ImageWidthGet Function

This function returns the width of the given image.

File

gfx_primitive.h

C

```
inline int16_t GFX_ImageWidthGet(GFX_RESOURCE_HDR * pImage);
```

Returns

The image width in pixels.

Description

This function returns the width of the given image in pixels. This function results in an undefined behavior if the pointer to the image is invalid.

This function return value is undefined if the given pointer does not point to a valid image resource.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pImage	Pointer to the image.

Function

GFX_STATUS GFX_ImageWidthGet(
GFX_RESOURCE_HDR *pImage)

Text Rendering Functions

Functions

	Name	Description
	GFX_FontAntiAliasGet	This function returns the current font font anti-aliasing mode.
	GFX_FontAntiAliasSet	This function sets the font anti-aliasing mode.
	GFX_FontSet	This function sets the current font used when rendering strings and characters.
	GFX_FontGet	This function returns the current font used when rendering strings and characters.
	GFX_TextCharDraw	This function renders the given character using the currently set color using the currently set font.
	GFX_TextCursorPositionSet	This function sets the text cursor to a new position.
	GFX_TextCursorPositionXGet	This function returns the current x position of the text cursor.
	GFX_TextCursorPositionYGet	This function returns the current y position of the text cursor.
	GFX_TextStringBoxDraw	This function renders the given string using the currently set color and font into a rectangular area.
	GFX_TextStringDraw	This function renders the given string of character using the currently set color using the currently set font.
	GFX_TextStringHeightGet	This function returns the height of the given font.
	GFX_TextStringWidthGet	This function returns the width of the given string using the given font.
	GFX_TextCursorDraw	This is function GFX_TextCursorDraw.
	GFX_TextCursorTypeGet	This is function GFX_TextCursorTypeGet.
	GFX_TextCursorTypeSet	This is function GFX_TextCursorTypeSet.
	GFX_TextCursorWidthGet	This is function GFX_TextCursorWidthGet.

Description

Graphics Library Primitive Layer text rendering API.

GFX_FontAntiAliasGet Function

This function returns the current font font anti-aliasing mode.

File

gfx_primitive.h

C

```
GFX_FONT_ANTIALIAS_TYPE GFX_FontAntiAliasGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The font anti-aliasing mode used.

Description

This function returns the font anti-aliasing mode used when rendering anti-aliased strings and characters. Default setting at initialization is GFX_FONT_ANTIALIAS_OPAQUE.

Preconditions

None.

Example

```
extern const GFX_RESOURCE_HDR antiAliasedFont25;
uint16_t width, height;
GFX_XCHAR myString[] = "Microchip Technology Inc.";

if (GFX_FontAntiAliasGet(GFX_INDEX_0) != GFX_FONT_ANTIALIAS_OPAQUE)
    GFX_FontAntiAliasSet(GFX_INDEX_0, GFX_FONT_ANTIALIAS_OPAQUE);

GFX_FontSet(GFX_INDEX_0, (GFX_RESOURCE_HDR*) &antiAliasedFont25);
GFX_ColorSet(GFX_INDEX_0, GFX_X11_GREEN);
width = GFX_TextStringWidthGet(myString, (GFX_RESOURCE_HDR*) &Font25);
height = GFX_TextStringHeightGet((GFX_RESOURCE_HDR*) &antiAliasedFont25);

GFX_TextStringDraw( GFX_INDEX_0, (GFX_MaxXGet() - width) >> 1,
                    (GFX_MaxYGet() - height) >> 1,
                    myString,
                    0);
```

Function

GFX_FONT_ANTIALIAS_TYPE GFX_FontAntiAliasGet(SYS_MODULE_INDEX gfxIndex)

GFX_FontAntiAliasSet Function

This function sets the font anti-aliasing mode.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_FontAntiAliasSet(SYS_MODULE_INDEX gfxIndex, GFX_FONT_ANTIALIAS_TYPE type);
```

Returns

The status of the set anti-aliasing mode action.

Description

This function sets the font anti-aliasing mode used when rendering anti-aliased strings and characters.

Preconditions

None.

Example

```
extern const GFX_RESOURCE_HDR antiAliasedFont25;
uint16_t width, height;
GFX_XCHAR myString[] = "Microchip Technology Inc.";

GFX_FontAntiAliasSet(GFX_INDEX_0, GFX_FONT_ANTIALIAS_TRANSLUCENT);
GFX_FontSet(GFX_INDEX_0, (GFX_RESOURCE_HDR*) &antiAliasedFont25);
GFX_ColorSet(GFX_INDEX_0, GFX_X11_GREEN);
width = GFX_TextStringWidthGet( myString,
                                (GFX_RESOURCE_HDR*) &Font25);
height = GFX_TextStringHeightGet((GFX_RESOURCE_HDR*) &antiAliasedFont25);

GFX_TextStringDraw( GFX_INDEX_0, (GFX_MaxXGet() - width) >> 1,
                    (GFX_MaxYGet() - height) >> 1,
                    myString,
                    0);
```

Parameters

Parameters	Description
type	anti-aliasing mode selected. See GFX_FONT_ANTIALIAS_TYPE for details.

Function

GFX_STATUS GFX_FontAntiAliasSet(

SYS_MODULE_INDEX gfxIndex,

GFX_FONT_ANTIALIAS_TYPE type)

GFX_FontSet Function

This function sets the current font used when rendering strings and characters.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_FontSet(SYS_MODULE_INDEX gfxIndex, GFX_RESOURCE_HDR * pFont);
```

Returns

The status of the set font action.

Description

This function sets the current font used to render strings and characters.

Preconditions

None.

Example

```
extern const GFX_RESOURCE_HDR Font25;
uint16_t width, height;
GFX_XCHAR myString[] = "Microchip Technology Inc.";

GFX_FontSet(GFX_INDEX_0, (GFX_RESOURCE_HDR*) &Font25);
GFX_ColorSet(GFX_INDEX_0, GFX_X11_GREEN);
width = GFX_TextStringWidthGet(myString, (GFX_RESOURCE_HDR*) &Font25);
height = GFX_TextStringHeightGet((GFX_RESOURCE_HDR*) &Font25);

GFX_TextStringDraw( GFX_INDEX_0, (GFX_MaxXGet() - width) >> 1,
                    (GFX_MaxYGet() - height) >> 1,
                    myString,
                    0);
```

Parameters

Parameters	Description
pFont	pointer to the font to be used.

Function

GFX_STATUS GFX_FontSet(
SYS_MODULE_INDEX gfxIndex,
GFX_RESOURCE_HDR *pFont)

GFX_FontGet Function

This function returns the current font used when rendering strings and characters.

File

gfx_primitive.h

C

```
GFX_RESOURCE_HDR* GFX_FontGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The pointer to the current font used in rendering strings and characters.

Description

This function returns the current font used to render strings and characters.

Preconditions

GFX_FontSet() must be called prior to any call to this function. If this function is called first, the returned value is undefined.

Example

None.

Function

GFX_RESOURCE_HDR GFX_FontGet(SYS_MODULE_INDEX gfxIndex)

GFX_TextCharDraw Function

This function renders the given character using the currently set color using the currently set font.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TextCharDraw(SYS_MODULE_INDEX gfxIndex, GFX_XCHAR ch);
```

Returns

The status of the character rendering. GFX_STATUS_SUCCESS - the character was rendered GFX_STATUS_FAILURE - the character was not rendered, function must be called again to render the character. GFX_STATUS_ERROR - the character ID passed is not a valid or points to the character glyph that does does not exist on the font table.

Description

This function renders the given character using the currently set font, and color to the location defined by the text cursor position. The color is set by GFX_ColorSet() while the font is set by GFX_FontSet(). The text cursor position is set by GFX_TextCursorPositionSet()

The rendering of the character becomes undefined when any one of the following is true:

- Text cursor position is set to locations outside the frame buffer.
- Color is not set, before this function is called.
- Font is not set, before this function is called.

Preconditions

Color must be set by GFX_ColorSet(). Font must be set by GFX_FontSet(). Text cursor position must be set by GFX_TextCursorPositionSet().

Example

```
// assume textString is a string of characters
// assume WHITE is a valid GFX_COLOR data
// assume pMyFont is a valid initialized font resource pointer
static uint16_t counter = 0;
GFX_XCHAR   ch;
GFX_STATUS  status;

GFX_ColorSet(GFX_INDEX_0, WHITE);
GFX_FontSet(GFX_INDEX_0, pMyFont);

// render characters until null character
while((GFX_XCHAR)(ch = *(textString + counter)) != 0)
{
    status = GFX_TextCharDraw(GFX_INDEX_0, ch);
    if (status != GFX_STATUS_SUCCESS)
        return (GFX_STATUS_FAILURE);
    counter++;
}
```

Parameters

Parameters	Description
ch	character ID that of the character that will be rendered.

Function

GFX_STATUS GFX_TextCharDraw(
SYS_MODULE_INDEX gfxIndex,
GFX_XCHAR ch)

GFX_TextCursorPositionSet Function

This function sets the text cursor to a new position.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TextCursorPositionSet(SYS_MODULE_INDEX gfxIndex, int16_t x, int16_t y);
```

Returns

Status of the text cursor position set. GFX_STATUS_SUCCESS - text cursor position set done. GFX_STATUS_FAILURE - text cursor position set is not done.

Description

This function sets the text cursor to a new (x,y) position. Text cursor is used as a starting point of the character rendered by the GFX_TextCharDraw() function.

If x and/or y does not lie on the frame buffer, then the behavior of GFX_TextCharDraw() function is undefined.

Preconditions

None.

Example

```
GFX_TextCursorPositionSet(GFX_INDEX_0, 10,10);
// write letter 'A' starting from 10, 10 position
GFX_TextCharDraw(GFX_INDEX_0, 'A');
```

Parameters

Parameters	Description
x	new x coordinate position of the text cursor.
y	new y coordinate position of the text cursor.

Function

GFX_STATUS GFX_TextCursorPositionSet(

SYS_MODULE_INDEX gfxIndex,

int16_t x,

int16_t y)

GFX_TextCursorPositionXGet Function

This function returns the current x position of the text cursor.

File

gfx_primitive.h

C

```
int16_t GFX_TextCursorPositionXGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current text cursor x position.

Description

This function returns the current x position of the text cursor. Text cursor is used as a starting point of the line rendered by the GFX_TextCharDraw() function.

Preconditions

None.

Example

None.

Function

int16_t GFX_TextCursorPositionXGet(SYS_MODULE_INDEX gfxIndex)

GFX_TextCursorPositionYGet Function

This function returns the current y position of the text cursor.

File

gfx_primitive.h

C

```
int16_t GFX_TextCursorPositionYGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current text cursor y position.

Description

This function returns the current y position of the text cursor. Text cursor is used as a starting point of the line rendered by the GFX_TextCharDraw() function.

Preconditions

None.

Example

None.

Function

int16_t GFX_TextCursorPositionYGet(SYS_MODULE_INDEX gfxIndex)

GFX_TextStringBoxDraw Function

This function renders the given string using the currently set color and font into a rectangular area.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TextStringBoxDraw(SYS_MODULE_INDEX gfxIndex, uint16_t x, uint16_t y, uint16_t width,
uint16_t height, GFX_XCHAR * pString, uint16_t length, GFX_ALIGNMENT align);
```

Returns

The status of the string rendering. GFX_STATUS_SUCCESS - the string was rendered GFX_STATUS_FAILURE - the string was not rendered, or is not yet finished. The function must be called again to render the remaining characters or lines.

Description

This function renders the given string using the currently color and font into the rectangular area defined by the given parameters x,y, width and height. The x,y parameters defines the left, top pixel position of the rectangular area. The width and height defines the size of the rectangular area. The rectangular area will define the area where the text will be rendered. Meaning all pixels EXCLUSIVE of the defined rectangle will be the rendering area for the text. If the given string exceeds the rectangular area, any pixels falling outside and along the defined rectangle (INCLUSIVE of the rectangle) will not be drawn.

The color is set by GFX_ColorSet() while the font is set by GFX_FontSet().

Multiple lines of strings can be rendered. To define string composed of multiple lines, end each line with a new line character (character ID 0x0A). Each line will be rendered according to the text alignment (See GFX_TEXT_ALIGNMENT).

When the length parameter is set to 0, the string will be rendered until the null character is reached. When the length parameter is greater than 0, the string will be rendered until one of the following cases occurs:

- null character is reached before the total rendered characters is less than the value of length
- total rendered characters is equal to the value of length before the null character is reached.

The rendering of the character becomes undefined when any one of the following is true:

- x, y, width and height defines an area partially or fully outside outside the frame buffer.
- Color is not set, before this function is called.
- Font is not set, before this function is called.

Preconditions

Color must be set by GFX_ColorSet(). Font must be set by GFX_FontSet().

Example

```
// assume APP_DEMO_FONT to be a valid font resource
GFX_XCHAR StringMsg[] = "Hello World";

GFX_ColorSet(GFX_INDEX_0, BLACK);
GFX_ScreenClear(GFX_INDEX_0);

GFX_ColorSet(GFX_INDEX_0, WHITE);
GFX_FontSet(GFX_INDEX_0, (GFX_RESOURCE_HDR*)&APP_DEMO_FONT);

// displayed message will be centered on the screen
while(GFX_TextStringBoxDraw( GFX_INDEX_0, 0, 0,
                             GFX_MaxXGet(), GFX_MaxYGet(),
                             StringMsg,
                             0, GFX_ALIGN_CENTER) !=
        GFX_STATUS_SUCCESS);
```

Parameters

Parameters	Description
x	Horizontal starting position of the rectangular area.
y	Vertical position position of the rectangular area.
width	Defines the width of the rectangular area.
height	Defines the height of the rectangular area.
pString	Pointer to the location of the string that will be rendered. String can have multiple lines where each line is terminated by a new line character.
length	Total number of characters to be rendered. When set to 0, the function will terminate until the null character is detected.
align	The alignment of the rendered text.

Function

GFX_STATUS FX_TextStringBoxDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t x,
uint16_t y,
uint16_t width,
uint16_t height,
GFX_XCHAR *pString,
uint16_t length,
GFX_ALIGNMENT align)

GFX_TextStringDraw Function

This function renders the given string of character using the currently set color using the currently set font.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TextStringDraw(SYS_MODULE_INDEX gfxIndex, uint16_t x, uint16_t y, GFX_XCHAR * pString,
uint16_t length);
```

Returns

The status of the string rendering. GFX_STATUS_SUCCESS - the string was rendered GFX_STATUS_FAILURE - the string was not rendered, or is not yet finished. The function must be called again to render the remaining characters.

Description

This function renders the given string of character using the currently set font, and color to the location defined by the given x,y position. The color is set by GFX_ColorSet() while the font is set by GFX_FontSet().

The text rendered by this function is always left aligned. When a newline character is encountered, the next character after the newline character will be rendered left aligned on the next line. The next line position is determined by the height of the font used.

When the length parameter is set to 0, the string will be rendered until the null character is reached. When the length parameter is greater than 0, the string will be rendered until one of the following cases occurs:

- null character is reached before the total rendered characters is less than the value of length
- total rendered characters is equal to the value of length before the null character is reached.

The rendering of the character becomes undefined when any one of the following is true:

- x,y position is set to locations outside the frame buffer.
- Color is not set, before this function is called.
- Font is not set, before this function is called.

Preconditions

Color must be set by GFX_ColorSet(). Font must be set by GFX_FontSet().

Example

```
// assume WHITE is a valid GFX_COLOR data
// assume pMyFont is a valid initialized font resource pointer

static uint16_t counter = 0;
GFX_XCHAR   charArray[] = "Test String";
GFX_XCHAR   pChar = NULL;
GFX_STATUS  status;

GFX_ColorSet(GFX_INDEX_0, WHITE);
GFX_FontSet(GFX_INDEX_0, pMyFont);

// render the whole string
GFX_TextStringDraw(GFX_INDEX_0, 10, 10, charArray, 0);

// render ONLY the "Test" portion of the string
GFX_TextStringDraw(GFX_INDEX_0, 10, 30, charArray, 4);

// render ONLY the "String" portion of the string
pChar = charArray;
pChar += 5;
GFX_TextStringDraw(GFX_INDEX_0, 10, 50, pChar, 0);

// doing this also renders ONLY the "String" portion of the string
GFX_TextStringDraw(GFX_INDEX_0, 10, 70, pChar, 6);
```

Parameters

Parameters	Description
x	Horizontal starting position of the string.
y	Vertical position position of the string.
pString	Pointer to the location of the string that will be rendered.
length	Total number of characters to be rendered. When set to 0, the function will terminate until the null character is detected.

Function

GFX_STATUS GFX_TextStringDraw(
SYS_MODULE_INDEX gfxIndex,
uint16_t x,
uint16_t y,
GFX_XCHAR *pString,
uint16_t length)

GFX_TextStringHeightGet Function

This function returns the height of the given font.

File

gfx_primitive.h

C

```
inline uint16_t GFX_TextStringHeightGet(GFX_RESOURCE_HDR * pFont);
```

Returns

The height of the specified font in pixels.

Description

This function returns the height of the given font in pixels. The given font must be present in the system.

This function return value is undefined if the given pointer does not point to a valid font.

Preconditions

None.

Example

```
// assume pMyFont is a pointer initialized to a valid font
uint16_t height, x, y;

// center the text on the screen

height = GFX_TextStringHeightGet(pMyFont);
width  = GFX_TextStringWidthGet (   (GFX_XCHAR *)"Hello World",
                                     pMyFont
                                );
y = (GFX_MaxYGet() - height) >> 1;
x = (GFX_MaxXGet() - width) >> 1;
GFX_TextStringDraw(GFX_INDEX_0, x, y, (GFX_XCHAR *)"Hello World");

GFX_TextStringDraw(GFX_INDEX_0, x, y, (GFX_XCHAR *)"Hello World");
```

Parameters

Parameters	Description
pFont	Pointer to the specified font.

Function

uint16_t GFX_TextStringHeightGet(
GFX_RESOURCE_HDR *pFont);

GFX_TextStringWidthGet Function

This function returns the width of the given string using the given font.

File

gfx_primitive.h

C

```
uint16_t GFX_TextStringWidthGet(GFX_XCHAR * textString, GFX_RESOURCE_HDR * pFont);
```

Returns

The width of the specified string using the specified font in pixels.

Description

This function returns the width of the given string using the given font in pixels. The given font must be present in the system.

The pixel length is measured from the first printable character until the last printable character. This means that if the string is composed of multiple lines, the length returned is only valid for the first line of characters.

This function return value is undefined if the given pointer does not point to a valid font or one or more characters in the given string does not exist on the given font.

Preconditions

None.

Example

```
// assume pMyFont is a pointer initialized to a valid font
```

```
uint16_t height, x, y;

// center the text on the screen

height = GFX_TextStringHeightGet(GFX_INDEX_0, pMyFont);
width  = GFX_TextStringWidthGet (GFX_INDEX_0,    (GFX_XCHAR *)"Hello World",
                                    pMyFont
                                );
y = (GFX_MaxYGet() - height) >> 1;
x = (GFX_MaxXGet() - width) >> 1;
GFX_TextStringDraw(GFX_INDEX_0, x, y, (GFX_XCHAR *)"Hello World");
```

Parameters

Parameters	Description
pString	Pointer to the specified string.
pFont	Pointer to the specified font.

Function

uint16_t GFX_TextStringWidthGet(
GFX_XCHAR* pString,
GFX_RESOURCE_HDR *pFont);

GFX_TextCursorDraw Function

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TextCursorDraw(SYS_MODULE_INDEX gfxIndex, uint16_t x, uint16_t y, uint16_t cursorWidth,
uint16_t cursorHeight);
```

Description

This is function GFX_TextCursorDraw.

GFX_TextCursorTypeGet Function

File

gfx_primitive.h

C

```
GFX_TEXT_CURSOR_TYPE GFX_TextCursorTypeGet(SYS_MODULE_INDEX gfxIndex);
```

Description

This is function GFX_TextCursorTypeGet.

GFX_TextCursorTypeSet Function

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TextCursorTypeSet(SYS_MODULE_INDEX gfxIndex, GFX_TEXT_CURSOR_TYPE textCursorType);
```

Description

This is function GFX_TextCursorTypeSet.

GFX_TextCursorWidthGet Function

File

gfx_primitive.h

C

```
int16_t GFX_TextCursorWidthGet(SYS_MODULE_INDEX gfxIndex);
```

Description

This is function GFX_TextCursorWidthGet.

Rendering Style Functions

Functions

	Name	Description
	GFX_FillStyleGet	Return the fill style used when rendering filled shapes.
	GFX_FillStyleSet	Set the fill style to be used when rendering filled shapes.
	GFX_AlphaBlendingValueGet	This function returns the current alpha value set for alpha blending rendering.
	GFX_AlphaBlendingValueSet	This function sets the alpha value for alpha blending rendering.
	GFX_GradientColorSet	Sets the gradient fill start and end color.
	GFX_GradientEndColorGet	Return the gradient end color when rendering gradient filled shapes.
	GFX_GradientStartColorGet	Return the gradient start color when rendering gradient filled shapes.
	GFX_LineStyleGet	Return the line style used when rendering lines.
	GFX_LineStyleSet	Set the line style to be used when rendering lines.

Description

Graphics Library Primitive Layer rendering style API.

GFX_FillStyleGet Function

Return the fill style used when rendering filled shapes.

File

gfx_primitive.h

C

```
GFX_FILL_STYLE GFX_FillStyleGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current fill style used (see GFX_FILL_STYLE).

Description

This function returns the fill style used (see GFX_FILL_STYLE) when rendering filled shapes.

Preconditions

None.

Example

None.

Function

GFX_FILL_STYLE GFX_FillStyleGet(SYS_MODULE_INDEX gfxIndex)

GFX_FillStyleSet Function

Set the fill style to be used when rendering filled shapes.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_FillStyleSet(SYS_MODULE_INDEX gfxIndex, GFX_FILL_STYLE style);
```

Returns

Status of the fill style set. GFX_STATUS_SUCCESS - fill style set done. GFX_STATUS_FAILURE - fill style set failed.

Description

This function sets the fill style to be used (see GFX_FILL_STYLE) when rendering filled shapes. All calls to the following:

- GFX_RectangleFillDraw()
- GFX_RectangleRoundFillDraw()
- GFX_CircleFillDraw()

will use the fill style set by this function.

Preconditions

None.

Example

```
GFX_GradientColorSet(BRIGHTRED, BRIGHTBLUE);

GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_GRADIENT_DOWN);
GFX_RectangleFillDraw(GFX_INDEX_0,   10, 10,  100,  100);

GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_GRADIENT_UP);
GFX_RectangleFillDraw(GFX_INDEX_0,  110, 10,  200,  100);

GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_GRADIENT_RIGHT);
GFX_RectangleFillDraw( GFX_INDEX_0, 210, 10,  300,  100);

GFX_FillStyleSet(GFX_INDEX_0, GFX_FILL_STYLE_GRADIENT_LEFT);
GFX_RectangleFillDraw(GFX_INDEX_0,   10, 110,  100,  200);
```

Parameters

Parameters	Description
style	The specified fill style to be used.

Function

GFX_STATUS GFX_FillStyleSet(
SYS_MODULE_INDEX gfxIndex,
GFX_FILL_STYLE style)

GFX_AlphaBlendingValueGet Function

This function returns the current alpha value set for alpha blending rendering.

File

gfx_primitive.h

C

```
uint16_t GFX_AlphaBlendingValueGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The alpha blending value.

Description

This function returns the current alpha value set for alpha blending rendering. See GFX_AlphaBlendingValueSet() for details of alpha blending values.

Preconditions

None.

Example

None.

Function

uint16_t GFX_AlphaBlendingValueGet(SYS_MODULE_INDEX gfxIndex)

GFX_AlphaBlendingValueSet Function

This function sets the alpha value for alpha blending rendering.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_AlphaBlendingValueSet(SYS_MODULE_INDEX gfxIndex, uint16_t alpha);
```

Returns

Status of the alpha value set action. Return GFX_STATUS_ERROR when the alpha value set is unsupported.

Description

This function sets the alpha value for alpha blending rendering. Accepted values are dependent on the used alpha blending routines at build time. When using the software routines in the Primitive Layer, accepted values are 0, 25, 50, 75 and 100.

If using a specific implementation in the display driver used, implementation may support the full range (0-100). When this is the case, refer to the driver alpha blending solution for details.

Function operation will ignore unsupported values of alpha.

Preconditions

None.

Example

```
// render a bar with no alpha blending
GFX_AlphaBlendingValueSet(GFX_INDEX_0, 100);
while(GFX_RectangleFillDraw(GFX_INDEX_0, 10,10,30,30) != GFX_STATUS_SUCCESS);

// render a bar with 50% alpha blending to a
// background with RED color
GFX_BackgroundSet(GFX_INDEX_0, x, y, pBgImage, RED);
GFX_BackgroundTypeSet(GFX_INDEX_0, GFX_BACKGROUND_COLOR);

GFX_AlphaBlendingValueSet(GFX_INDEX_0, 50);
while(GFX_RectangleFillDraw(GFX_INDEX_0, 10,40,30,60) != GFX_STATUS_SUCCESS);
```

Parameters

Parameters	Description
alpha	Defines the alpha blending percentage to be used for alpha blending routines. Accepted values are dependent on the alpha blending routines used. For Primitive Layer
implementation accepted values are	• 100 : no alpha blending, last color set by GFX_ColorSet() will replace the pixels. • 75 : 75% of the last color set by GFX_ColorSet() will be alpha blended to the existing pixel. • 50 : 50% of the last color set by GFX_ColorSet() will be alpha blended to the existing pixel. • 25 : 25% of the last color set by GFX_ColorSet() will be alpha blended to the existing pixel. • 0 : 0% of the last color set by GFX_ColorSet() will be alpha blended to the existing pixel. This means the existing pixel color will not change.

Function

GFX_STATUS GFX_AlphaBlendingValueSet(
SYS_MODULE_INDEX gfxIndex, uint16_t alpha)

GFX_GradientColorSet Function

Sets the gradient fill start and end color.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_GradientColorSet(SYS_MODULE_INDEX gfxIndex, GFX_COLOR startColor, GFX_COLOR endColor);
```

Returns

Status of the color set. GFX_STATUS_SUCCESS - gradient color set done. GFX_STATUS_FAILURE - gradient color set failed.

Description

This function sets the gradient fill start and end colors (see GFX_FILL_STYLE) when rendering gradient filled shapes. All subsequent calls to the following using gradient fills:

- GFX_RectangleFillDraw()
- GFX_RectangleRoundFillDraw()
- GFX_CircleFillDraw()

will use the start and end colors set by this function.

Preconditions

None.

Example

See GFX_FillStyleSet().

Parameters

Parameters	Description
startColor	Gradient start color to be used.
endColor	Gradient end color to be used.

Function

GFX_STATUS GFX_GradientColorSet(
SYS_MODULE_INDEX gfxIndex,
GFX_COLOR startColor,
GFX_COLOR endColor)

GFX_GradientEndColorGet Function

Return the gradient end color when rendering gradient filled shapes.

File

gfx_primitive.h

C

```
GFX_COLOR GFX_GradientEndColorGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current gradient end color used.

Description

This function returns the end color used when rendering gradient filled shapes.

Preconditions

None.

Example

None.

Function

GFX_COLOR GFX_GradientEndColorGet(void)

GFX_GradientStartColorGet Function

Return the gradient start color when rendering gradient filled shapes.

File

gfx_primitive.h

C

```
GFX_COLOR GFX_GradientStartColorGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current gradient start color used.

Description

This function returns the start color used when rendering gradient filled shapes.

Preconditions

None.

Example

None.

Function

GFX_COLOR GFX_GradientStartColorGet(SYS_MODULE_INDEX gfxIndex)

GFX_LineStyleGet Function

Return the line style used when rendering lines.

File

gfx_primitive.h

C

```
GFX_LINE_STYLE GFX_LineStyleGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current line style used (see GFX_LINE_STYLE).

Description

This function returns the line style used (see GFX_LINE_STYLE) when rendering lines.

Preconditions

None.

Example

None.

Function

GFX_LINE_STYLE GFX_LineStyleGet(SYS_MODULE_INDEX gfxIndex)

GFX_LineStyleSet Function

Set the line style to be used when rendering lines.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_LineStyleSet(SYS_MODULE_INDEX gfxIndex, GFX_LINE_STYLE style);
```

Returns

Status of the line style set. GFX_STATUS_SUCCESS - line setting done. GFX_STATUS_FAILURE - line setting not done.

Description

This function sets the line style to be used (see GFX_LINE_STYLE) when rendering lines. All calls to the following functions

- GFX_LineDraw()
- GFX_LineToDraw()
- GFX_LineToRelativeDraw()
- GFX_CircleDraw()
- GFX_RectangleDraw()
- GFX_PolygonDraw()

will use the line style set by this function. In addition, all unfilled shapes that does specify the line style to be used will use the line style specified by this function.

Preconditions

None.

Example

```
GFX_ColorSet(GFX_INDEX_0, BRIGHTRED);
GFX_LineStyleSet(GFX_INDEX_0, GFX_LINE_STYLE_THIN_SOLID);
GFX_LineDraw(GFX_INDEX_0, 10, 10,100,  10);
```

Parameters

Parameters	Description
style	The specified line style to be used.

Function

GFX_STATUS GFX_LineStyleSet(

SYS_MODULE_INDEX gfxIndex,

GFX_LINE_STYLE style)

Color Functions

Functions

	Name	Description
	GFX_ColorGet	This function returns the color currently set to render primitive shapes and text.
	GFX_ColorSet	This function sets the color to be used in rendering primitive shapes and text.
	GFX_TransparentColorDisable	This function disables the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.
	GFX_TransparentColorEnable	This function sets the transparent color used in GFX_ImageDraw() functions and enables the transparent color feature.
	GFX_TransparentColorGet	This returns the current transparent color set for the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.
	GFX_TransparentColorStatusGet	This returns the current state of the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

Description

Graphics Library Primitive Layer color API.

GFX_ColorGet Function

This function returns the color currently set to render primitive shapes and text.

File

gfx_primitive.h

C

```
GFX_COLOR GFX_ColorGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The currently set color.

Description

This function returns the color currently set to render primitive shapes and text (See GFX_ColorSet() for more information).

Preconditions

None.

Example

None.

Function

void GFX_ColorGet(SYS_MODULE_INDEX gfxIndex)

GFX_ColorSet Function

This function sets the color to be used in rendering primitive shapes and text.

File

gfx_primitive.h

C

```
void GFX_ColorSet(SYS_MODULE_INDEX gfxIndex, GFX_COLOR newColor);
```

Returns

None.

Description

This function sets the color to be used to render primitive shapes and text. Any primitive shape that is set to any of the following line style and fill style will be rendering using the set color.

GFX_LINE_STYLE:

- GFX_LINE_STYLE_THIN_SOLID
- GFX_LINE_STYLE_THIN_DOTTED
- GFX_LINE_STYLE_THIN_DASHED
- GFX_LINE_STYLE_THICK_SOLID
- GFX_LINE_STYLE_THICK_DOTTED
- GFX_LINE_STYLE_THICK_DASHED

GFX_FILL_STYLE:

- GFX_FILL_STYLE_COLOR

Rendering text using the following text rendering functions will also use the set color:

- GFX_TextCharDraw()
- GFX_TextStringDraw()
- GFX_TextStringBoxDraw()

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the rendering color set to render primitive shapes.

Function

void GFX_ColorSet(SYS_MODULE_INDEX gfxIndex, GFX_COLOR color)

GFX_TransparentColorDisable Function

This function disables the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TransparentColorDisable(SYS_MODULE_INDEX gfxIndex);
```

Returns

The status of the transparent color disable action.

Description

This function disables the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

The transparent color feature can only be enabled when the color depth used is 24 or 16 bpp.

Preconditions

Transparent color feature must be enabled at build time. This is enabled by default and can be disabled by defining the macro GFX_CONFIG_TRANSPARENT_COLOR_DISABLE in the system.

Example

```
// assume ScreenBackground and RibbonIcon are valid
// image resources
// assume BLACK is a valid GFX_COLOR value

GFX_TransparentColorEnable(GFX_INDEX_0, BLACK);
GFX_ImageDraw(GFX_INDEX_0, 0,0, (void*)&ScreenBackground);
GFX_ImageDraw(GFX_INDEX_0, 50,50, (void*)&RibbonIcon);

// disable the transparent color feature since the
// next image to render contains black pixels that
// we want to render
GFX_TransparentColorDisable(GFX_INDEX_0);
GFX_ImageDraw(GFX_INDEX_0, 50,50, (void*)&OverlayImage);
```

Function

GFX_STATUS GFX_TransparentColorDisable(SYS_MODULE_INDEX gfxIndex)

GFX_TransparentColorEnable Function

This function sets the transparent color used in GFX_ImageDraw() functions and enables the transparent color feature.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_TransparentColorEnable(SYS_MODULE_INDEX gfxIndex, GFX_COLOR color);
```

Returns

The status of the transparent color set action.

Description

This function sets the transparent color used in GFX_ImageDraw() functions and enables the transparent color feature.

When GFX_ImageDraw() or GFX_ImagePartialDraw() is called, any pixels in the image that matches the color value will not be rendered to the frame buffer.

The transparent color feature can only be enabled when the color depth used is 24 or 16 bpp.

Preconditions

Transparent color feature must be enabled at build time. This is enabled by default and can be disabled by defining the macro GFX_CONFIG_TRANSPARENT_COLOR_DISABLE in the system.

Example

```
// assume ScreenBackground and RibbonIcon are valid
// image resources
// assume BLACK is a valid GFX_COLOR value

GFX_TransparentColorEnable(GFX_INDEX_0, BLACK);
GFX_ImageDraw(GFX_INDEX_0, 0,0, (void*)&ScreenBackground);
GFX_ImageDraw(GFX_INDEX_0, 0,0, (void*)&RibbonIcon);
```

Parameters

Parameters	Description
color	the color value selected as the transparent color.

Function

GFX_STATUS GFX_TransparentColorEnable(
SYS_MODULE_INDEX gfxIndex,
GFX_COLOR color)

GFX_TransparentColorGet Function

This returns the current transparent color set for the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

File

gfx_primitive.h

C

```
GFX_COLOR GFX_TransparentColorGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current transparent color used.

Description

This returns the current transparent color set for the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

The transparent color feature can only be enabled when the color depth used is 24 or 16 bpp.

Preconditions

Transparent color feature must be enabled at build time. This is enabled by default and can be disabled by defining the macro GFX_CONFIG_TRANSPARENT_COLOR_DISABLE in the system.

Example

```
// assume ScreenBackground and RibbonIcon are valid
// image resources
// assume BLACK is a valid GFX_COLOR value

GFX_TransparentColorEnable(GFX_INDEX_0, BLACK);
GFX_ImageDraw(GFX_INDEX_0, 0,0, (void*)&ScreenBackground);
GFX_ImageDraw(GFX_INDEX_0, 50,50, (void*)&RibbonIcon);

// disable the transparent color feature since the
// next image to render contains black pixels that
// we want to render
if (GFX_TransparentColorGet(GFX_INDEX_0) == BLACK)
    GFX_TransparentColorDisable(GFX_INDEX_0);
GFX_ImageDraw(GFX_INDEX_0, 50,50, (void*)&OverlayImage);
```

Function

GFX_COLOR GFX_TransparentColorGet(SYS_MODULE_INDEX gfxIndex)

GFX_TransparentColorStatusGet Function

This returns the current state of the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

File

gfx_primitive.h

C

```
GFX_FEATURE_STATUS GFX_TransparentColorStatusGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current transparent feature status (see GFX_FEATURE_STATUS for details).

Description

This returns the current transparent color set for the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

The transparent color feature can only be enabled when the color depth used is 24 or 16 bpp.

Preconditions

Transparent color feature must be enabled at build time. This is enabled by default and can be disabled by defining the macro GFX_CONFIG_TRANSPARENT_COLOR_DISABLE in the system.

Example

```
// assume ScreenBackground and RibbonIcon are valid
// image resources
// assume BLACK is a valid GFX_COLOR value

if (GFX_TransparentColorStatusGet(GFX_INDEX_0) == GFX_FEATURE_DISABLED)
    GFX_TransparentColorEnable(GFX_INDEX_0, BLACK);
GFX_ImageDraw(GFX_INDEX_0, 0,0, (void*)&ScreenBackground);
GFX_ImageDraw(GFX_INDEX_0, 50,50, (void*)&RibbonIcon);

// disable the transparent color feature since the
// next image to render contains black pixels that
// we want to render
if (GFX_TransparentColorGet(GFX_INDEX_0) == BLACK)
    GFX_TransparentColorDisable(GFX_INDEX_0);
GFX_ImageDraw(GFX_INDEX_0, 50,50, (void*)&OverlayImage);
```

Function

GFX_FEATURE_STATUS GFX_TransparentColorStatusGet(SYS_MODULE_INDEX gfxIndex)

Background Functions

Functions

	Name	Description
	GFX_BackgroundColorGet	This function returns the color used in the current background.
	GFX_BackgroundImageGet	This function returns the image used in the current background.
	GFX_BackgroundImageLeftGet	This function returns the horizontal starting position of the current background.
	GFX_BackgroundImageTopGet	This function returns the vertical starting position of the current background.
	GFX_BackgroundSet	This function sets the background information.
	GFX_BackgroundTypeGet	This function returns the type of the current background.
	GFX_BackgroundTypeSet	This function sets the background type.

Description

Graphics Library Primitive Layer background API.

GFX_BackgroundColorGet Function

This function returns the color used in the current background.

File

gfx_primitive.h

C

```
GFX_COLOR GFX_BackgroundColorGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The color used in the current background.

Description

This function returns the color used in the current background.

Preconditions

None.

Example

None.

Function

GFX_COLOR GFX_BackgroundColorGet(SYS_MODULE_INDEX gfxIndex)

GFX_BackgroundImageGet Function

This function returns the image used in the current background.

File

gfx_primitive.h

C

```
GFX_RESOURCE_HDR * GFX_BackgroundImageGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The pointer to the image used in the current background.

Description

This function returns the pointer to the image used in the current background.

Preconditions

None.

Example

None.

Function

GFX_RESOURCE_HDR *GFX_BackgroundImageGet(SYS_MODULE_INDEX gfxIndex)

GFX_BackgroundImageLeftGet Function

This function returns the horizontal starting position of the current background.

File

gfx_primitive.h

C

```
uint16_t GFX_BackgroundImageLeftGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The horizontal starting position of the current background.

Description

This function returns the horizontal starting position of the current background. This position defines the x position of the upper left corner of the background.

Preconditions

None.

Example

None.

Function

uint16_t GFX_BackgroundImageLeftGet(SYS_MODULE_INDEX gfxIndex)

GFX_BackgroundImageTopGet Function

This function returns the vertical starting position of the current background.

File

gfx_primitive.h

C

```
uint16_t GFX_BackgroundImageTopGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The vertical starting position of the current background.

Description

This function returns the vertical starting position of the current background. This position defines the y position of the upper left corner of the background.

Preconditions

None.

Example

None.

Function

uint16_t GFX_BackgroundImageTopGet(SYS_MODULE_INDEX gfxIndex)

GFX_BackgroundSet Function

This function sets the background information.

File

gfx_primitive.h

C

```
void GFX_BackgroundSet(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, GFX_RESOURCE_HDR * pImage,
GFX_COLOR color);
```

Returns

None.

Description

This function sets the background information. Note that the width and height of the background is only needed when the background is an image. When the background is a color (image pointer is NULL), the width and height is not needed here since the purpose of the background information is to record only the color used.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
left	Horizontal starting position of the background.
top	Vertical starting position of the background.
pImage	Pointer to the background image used. Set this to NULL if not used.
color	If the background image is NULL, background is assumed to be this color.

Function

void GFX_BackgroundSet(

SYS_MODULE_INDEX gfxIndex,

uint16_t left,

uint16_t top,

GFX_RESOURCE_HDR *pImage,

GFX_COLOR color);

GFX_BackgroundTypeGet Function

This function returns the type of the current background.

File

gfx_primitive.h

C

```
GFX_BACKGROUND_TYPE GFX_BackgroundTypeGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The type of the current background set.

Description

This function returns the type of the current background.

Preconditions

None.

Example

None.

Function

GFX_BACKGROUND_TYPE GFX_BackgroundTypeGet(SYS_MODULE_INDEX gfxIndex)

GFX_BackgroundTypeSet Function

This function sets the background type.

File

gfx_primitive.h

C

```
void GFX_BackgroundTypeSet(SYS_MODULE_INDEX gfxIndex, GFX_BACKGROUND_TYPE type);
```

Returns

None.

Description

This function sets the background type.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
type	Type of background that will be used.

Function

void GFX_BackgroundTypeSet(SYS_MODULE_INDEX gfxIndex, GFX_BACKGROUND_TYPE type)

Double Buffering Functions

Functions

	Name	Description
	GFX_DoubleBufferAreaMark	This function adds the given rectangular area as an area that will be included in the list of areas for synchronization.
	GFX_DoubleBufferEnable	This function enables the double buffering feature of the graphics library.
	GFX_DoubleBufferDisable	This function disables the double buffering feature of the graphics library.
	GFX_DoubleBufferStatusGet	This function returns the current status of the double buffering feature of the graphics library.
	GFX_DoubleBufferAreaGet	This function returns a rectangular area that needs synchronization specified by the given index.
	GFX_DoubleBufferSyncAllStatusClear	This function clears the synchronize all status.
	GFX_DoubleBufferSyncAllStatusGet	This function returns the status of the synchronize all flag.
	GFX_DoubleBufferSyncAllStatusSet	This function sets the whole draw buffer to be unsynchronized.
	GFX_DoubleBufferSyncAreaCountGet	This function returns the current count of rectangular areas that needs to be synchronized.
	GFX_DoubleBufferSyncAreaCountSet	This function sets the current count of rectangular areas that needs to be synchronized.
	GFX_DoubleBufferSynchronizeRequest	This function requests synchronization of the contents of the draw and frame buffer.
	GFX_DoubleBufferSynchronize	This function synchronizes the contents of the draw and frame buffer immediately.
	GFX_DoubleBufferSynchronizeStatusGet	This function returns the status of the synchronization request of the draw and frame buffer.
	GFX_DrawBufferGet	This function returns the index of the current draw buffer.
	GFX_DrawBufferInitialize	This function initializes the address of the draw buffer specified by the given index.
	GFX_DrawBufferSet	This function sets the draw buffer with the given index number.
	GFX_FrameBufferGet	This function returns the index of the current frame buffer.
	GFX_FrameBufferSet	This function sets the frame buffer with the given index number.
	GFX_DoubleBufferPause	This function pauses the double buffering feature of the graphics library.
	GFX_DoubleBufferResume	This function resumes the double buffering feature of the graphics library.

Description

Graphics Library Primitive Layer double buffering API.

GFX_DoubleBufferAreaMark Function

This function adds the given rectangular area as an area that will be included in the list of areas for synchronization.

File

gfx_primitive.h

C

```
void GFX_DoubleBufferAreaMark(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right,
uint16_t bottom);
```

Returns

None.

Description

When double buffering mode is enabled, this function adds the given rectangular area into the list of rectangular areas for synchronization. When this function is called, the given rectangular area is assumed to contain new pixel information and is added into the list of areas to be synchronized.

Synchronization can be scheduled using the GFX_DoubleBufferSynchronizeRequest() or immediately performed using GFX_DoubleBufferSynchronize().

Preconditions

Double buffering feature must be enabled.

Example

None.

Parameters

Parameters	Description
left	defines the left most pixel of the rectangular area.
top	defines the top most pixel of the rectangular area.
right	defines the right most pixel of the rectangular area.
bottom	defines the bottom most pixel of the rectangular area.

Function

void GFX_DoubleBufferAreaMark(
SYS_MODULE_INDEX gfxIndex,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom)

GFX_DoubleBufferEnable Function

This function enables the double buffering feature of the graphics library.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferEnable(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the double buffering feature enabling. GFX_STATUS_SUCCESS - the double buffering was successfully enabled.
GFX_STATUS_FAILURE - the double buffering was not successfully enabled.

Description

Double buffering is a feature where two buffers are utilized to perform rendering on one buffer while displaying the other buffer. The frame buffer is the buffer that is being displayed while the draw buffer is used for rendering.

When this function is called, the buffer with index number 0 is set as the draw buffer while the buffer with index number 1 is set as the frame buffer. Synchronization of the two buffers is scheduled and sets the count of unsynchronized areas to zero.

Preconditions

None.

Example

None.

Function

GFX_STATUS GFX_DoubleBufferEnable(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferDisable Function

This function disables the double buffering feature of the graphics library.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferDisable(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the double buffering feature disabling. GFX_STATUS_SUCCESS - the double buffering was successfully disabled.
GFX_STATUS_FAILURE - the double buffering was not successfully disabled.

Description

When double buffering is enabled, calling this function disables the double buffering feature of the graphics library.

After this function executes, all rendering will be performed on the frame buffer.

Preconditions

None.

Example

None.

Function

GFX_STATUS GFX_DoubleBufferDisable(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferStatusGet Function

This function returns the current status of the double buffering feature of the graphics library.

File

gfx_primitive.h

C

```
GFX_FEATURE_STATUS GFX_DoubleBufferStatusGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the double buffering feature disabling. GFX_FEATURE_ENABLED - the double buffering feature is enabled.
GFX_FEATURE_DISABLED - the double buffering feature is not enabled.

Description

If the graphics library double buffering feature is enabled, this function returns if the double buffering mode is activated or not. When activated, the function returns GFX_FEATURE_ENABLED. When deactivated, the function returns GFX_FEATURE_DISABLED.

Preconditions

None.

Example

None.

Function

GFX_FEATURE_STATUS GFX_DoubleBufferStatusGet(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferAreaGet Function

This function returns a rectangular area that needs synchronization specified by the given index.

File

gfx_primitive.h

C

```
GFX_RECTANGULAR_AREA * GFX_DoubleBufferAreaGet(SYS_MODULE_INDEX gfxIndex, uint16_t index);
```

Returns

The location of the rectangular area specified by the structure GFX_RECTANGULAR_AREA.

Description

Double buffering mode maintains an array of these GFX_RECTANGULAR_AREA that needs synchronization. The array serves as a list of pixel areas that needs to be synchronized.

This function returns a rectangular area that needs synchronization specified by the given index. The returned value is a pointer to the structure GFX_RECTANGULAR_AREA that describes the rectangular area of interest.

Preconditions

Double buffering feature must be enabled.

Example

None.

Parameters

Parameters	Description
index	the index of the rectangular area located in the array of areas that needs synchronization.

Function

GFX_RECTANGULAR_AREA GFX_DoubleBufferAreaGet(SYS_MODULE_INDEX gfxIndex, uint16_t index)

GFX_DoubleBufferSyncAllStatusClear Function

This function clears the synchronize all status.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferSyncAllStatusClear(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the synchronize all clear. GFX_STATUS_SUCCESS - the synchronization all clear was successful. GFX_STATUS_FAILURE - the synchronization all clear was not successful.

Description

This function clears the synchronize all status. This function will clear or cancel all synchronization requests. Calling this function while an ongoing synchronization is being performed, will not terminate the synchronization.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

GFX_STATUS GFX_DoubleBufferSyncAllStatusClear(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferSyncAllStatusGet Function

This function returns the status of the synchronize all flag.

File

gfx_primitive.h

C

```
GFX_FEATURE_STATUS GFX_DoubleBufferSyncAllStatusGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the synchronize all flag. GFX_FEATURE_ENABLED - the synchronization all is set. GFX_FEATURE_DISABLED - the synchronization all is not set.

Description

This function returns the status of the synchronize all flag.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

GFX_FEATURE_STATUS GFX_DoubleBufferSyncAllStatusGet(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferSyncAllStatusSet Function

This function sets the whole draw buffer to be unsynchronized.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferSyncAllStatusSet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the synchronize all set. GFX_STATUS_SUCCESS - the synchronization all was successful. GFX_STATUS_FAILURE - the synchronization all was not successful.

Description

This function sets the whole draw buffer to be unsynchronized. The next synchronization will copy all pixels of the draw buffer to the frame buffer.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

GFX_STATUS GFX_DoubleBufferSyncAllStatusSet(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferSyncAreaCountGet Function

This function returns the current count of rectangular areas that needs to be synchronized.

File

gfx_primitive.h

C

```
uint16_t GFX_DoubleBufferSyncAreaCountGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current count of rectangular areas that needs synchronization.

Description

This function returns the current count of rectangular areas that needs to be synchronized.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

uint16_t GFX_DoubleBufferSyncAreaCountGet(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferSyncAreaCountSet Function

This function sets the current count of rectangular areas that needs to be synchronized.

File

gfx_primitive.h

C

```
void GFX_DoubleBufferSyncAreaCountSet(SYS_MODULE_INDEX gfxIndex, uint16_t count);
```

Returns

None.

Description

This function sets the current count of rectangular areas that needs to be synchronized.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

void GFX_DoubleBufferSyncAreaCountSet(SYS_MODULE_INDEX gfxIndex, uint16_t count)

GFX_DoubleBufferSynchronizeRequest Function

This function requests synchronization of the contents of the draw and frame buffer.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferSynchronizeRequest(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the double buffer synchronization request. GFX_STATUS_SUCCESS - the synchronization request was successful. GFX_STATUS_FAILURE - the synchronization request was not successful.

Description

This function requests synchronization of the contents of the draw and frame buffer. The contents will be synchronized on the next vertical blanking period.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

GFX_STATUS GFX_DoubleBufferSynchronizeRequest(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferSynchronize Function

This function synchronizes the contents of the draw and frame buffer immediately.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferSynchronize(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the double buffer synchronization. GFX_STATUS_SUCCESS - the synchronization was successful. GFX_STATUS_FAILURE - the synchronization was not successful.

Description

This function synchronizes the contents of the draw and frame buffer immediately. Contents of both frame and draw buffer will be the same after the function exits.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

GFX_STATUS GFX_DoubleBufferSynchronize(SYS_MODULE_INDEX gfxIndex)

GFX_DoubleBufferSynchronizeStatusGet Function

This function returns the status of the synchronization request of the draw and frame buffer.

File

gfx_primitive.h

C

```
GFX_FEATURE_STATUS GFX_DoubleBufferSynchronizeStatusGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Status of the double buffer synchronization request. GFX_FEATURE_ENABLED - the synchronization request is enabled. GFX_FEATURE_DISABLED - the synchronization request is not enabled.

Description

This function returns the status of the synchronization request of the draw and frame buffer. The difference between GFX_DoubleBufferSynchronizeStatusGet() and GFX_DoubleBufferSyncAllStatusGet() is that the GFX_DoubleBufferSynchronizeStatusGet() returns the status of synchronization request. The size of the synchronization may or may not be the full screen synchronization. The GFX_DoubleBufferSyncAllStatusGet() on the other hand returns the status of a full screen synchronization.

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

GFX_FEATURE_STATUS GFX_DoubleBufferSynchronizeStatusGet(SYS_MODULE_INDEX gfxIndex)

GFX_DrawBufferGet Function

This function returns the index of the current draw buffer.

File

gfx_primitive.h

C

```
uint16_t GFX_DrawBufferGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The index of the current draw buffer set.

Description

This function returns the index of the current draw buffer. Draw buffer is the buffer where rendering is performed. For systems with single buffer this function will always return 0.

Preconditions

None.

Example

None.

Function

uint16_t GFX_DrawBufferGet(SYS_MODULE_INDEX gfxIndex)

GFX_DrawBufferInitialize Function

This function initializes the address of the draw buffer specified by the given index.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DrawBufferInitialize(SYS_MODULE_INDEX gfxIndex, uint16_t index, uint32_t address);
```

Returns

Status of the buffer set. GFX_STATUS_SUCCESS - the buffer was successfully set. GFX_STATUS_FAILURE - the buffer was not successfully set.

Description

**********|**********

For system with multiple buffers, this function is used to initialize the array of buffers. The address of the draw buffer will be associated with the specified index. Use this function to initialize or modify the array of frame buffers in the system at run time.

For systems with single buffer, frame buffer and draw buffer are the same buffer. Calls to this function will have no effect and will always return GFX_STATUS_SUCCESS. The size of the buffer is defined by the dimension of the screen and the color depth used.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
index	specifies the index value of the buffer in the array of buffers. For single buffer systems, the given index will be ignored.
address	specifies the location of the buffer in memory.

Function

GFX_STATUS GFX_DrawBufferInitialize(

SYS_MODULE_INDEX gfxIndex,

uint16_t index,

uint32_t address)

GFX_DrawBufferSet Function

This function sets the draw buffer with the given index number.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DrawBufferSet(SYS_MODULE_INDEX gfxIndex, uint16_t index);
```

Returns

Status of the draw buffer set. GFX_STATUS_SUCCESS - the draw buffer was successfully set. GFX_STATUS_FAILURE - the draw buffer was not successfully set.

Description

This function sets the draw buffer with the given index number. For systems with single buffer, frame buffer and draw buffer are the same buffer. Calls to this function will have no effect and will always return GFX_STATUS_SUCCESS.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
index	the index of the buffer to be set as the draw buffer.

Function

GFX_STATUS GFX_DrawBufferSet(SYS_MODULE_INDEX gfxIndex, uint16_t index)

GFX_FrameBufferGet Function

This function returns the index of the current frame buffer.

File

gfx_primitive.h

C

```
uint16_t GFX_FrameBufferGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The index of the current frame buffer set.

Description

This function returns the index of the current frame buffer. Frame buffer is the buffer that is currently displayed in the screen. For systems with single buffer this function will always return 0.

Preconditions

None.

Example

None.

Function

uint16_t GFX_FrameBufferGet(void)

GFX_FrameBufferSet Function

This function sets the frame buffer with the given index number.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_FrameBufferSet(SYS_MODULE_INDEX gfxIndex, uint16_t index);
```

Returns

Status of the draw buffer set. GFX_STATUS_SUCCESS - the draw buffer was successfully set. GFX_STATUS_FAILURE - the draw buffer was not successfully set.

Description

This function sets the frame buffer with the given index number. This is the buffer that is displayed on the screen. For systems with single buffer, frame buffer and draw buffer are the same buffer. Calls to this function will have no effect and will always return GFX_STATUS_SUCCESS.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
index	the index of the buffer to be set as the frame buffer.

Function

GFX_STATUS GFX_FrameBufferSet(SYS_MODULE_INDEX gfxIndex, uint16_t index)

GFX_DoubleBufferPause Function

This function pauses the double buffering feature of the graphics library.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferPause(SYS_MODULE_INDEX gfxIndex);
```

Description

Double buffering is a feature where two buffers are utilized to perform rendering on one buffer while displaying the other buffer. The frame buffer is the buffer that is being displayed while the draw buffer is used for rendering.

When this function is called, the feature is paused and no buffer swap will occur.

Function

GFX_STATUS GFX_DoubleBufferPause(void)

GFX_DoubleBufferResume Function

This function resumes the double buffering feature of the graphics library.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_DoubleBufferResume(SYS_MODULE_INDEX gfxIndex);
```

Description

Double buffering is a feature where two buffers are utilized to perform rendering on one buffer while displaying the other buffer. The frame buffer is the buffer that is being displayed while the draw buffer is used for rendering.

When this function is called, the feature is resumes the previous status of the double buffering feature.

Function

GFX_STATUS GFX_DoubleBufferResume(void)

External Resources Functions

Functions

	Name	Description
	GFX_ExternalResourceCallback	This function performs data fetch from external memory.

Description

Graphics Library Primitive Layer external resources API.

GFX_ExternalResourceCallback Function

This function performs data fetch from external memory.

File

gfx_primitive.h

C

```
GFX_STATUS GFX_ExternalResourceCallback(GFX_RESOURCE_HDR * pResource, uint32_t offset, uint16_t nCount,
void * pBuffer);
```

Returns

Status of the external resource callback. GFX_STATUS_SUCCESS when all the data was successfully retrieved. GFX_STATUS_FAILURE when partial or no data was retrieved.

Description

This function must be implemented in the application. The library will call this function each time when the external memory data will be required.

The application must copy requested byte quantity into the buffer provided. Data start address in external memory is a sum of the address in GFX_RESOURCE_HDR structure and offset.

An example of a situation where external memory will be accessed is when external fonts or images are used. External resources in the library are defined by the type in the GFX_RESOURCE_HDR (see GFX_RESOURCE for details).

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pResource	Pointer to the external memory resource information.
offset	Offset of the data from the location of the resource in external memory.
nCount	Number of bytes to be transferred into the buffer.
buffer	Pointer to the buffer that will hold the retrieved data.

Function

GFX_STATUS GFX_ExternalResourceCallback(
GFX_RESOURCE_HDR *pResource,
uint32_t offset,
uint16_t nCount,
void *pBuffer)

Data Types and Constants

Enumerations

	Name	Description
	GFX_PRIMITIVE_STATES	Defines the various states that can be achieved by the PRIMITIVE HAL operation.

Macros

	Name	Description
	GFX_PRIMITIVE_HANDLE_INVALID	Constant that defines the value of an Invalid Device Handle.
	GFX_TEXT_CURSOR_HEIGHT_DEFAULT	Constant that defines the default value of text cursor height.
	GFX_TEXT_CURSOR_WIDTH_DEFAULT	Constant that defines the default value of text cursor width.

Types

	Name	Description
	GFX_PRIMITIVE_HANDLE	Data type for GFX PRIMITIVE HAL handle.

Description

Graphics Library Primitive Layer data types and constants.

GFX_PRIMITIVE_HANDLE_INVALID Macro

Constant that defines the value of an Invalid Device Handle.

File

gfx_primitive.h

C

```
#define GFX_PRIMITIVE_HANDLE_INVALID
```

Description

GFX PRIMITIVE HAL Invalid Handle

This constant is returned by the GFX_PRIMITIVE_Open() function when the function fails.

Remarks

None.

GFX_PRIMITIVE_HANDLE Type

Data type for GFX PRIMITIVE HAL handle.

File

gfx_primitive.h

C

```
typedef uintptr_t GFX_PRIMITIVE_HANDLE;
```

Description

Data type for GFX PRIMITIVE HAL handle.

The data type of the handle that is returned from GFX_PRIMITIVE_Open function.

Remarks

None.

GFX_PRIMITIVE_STATES Enumeration

Defines the various states that can be achieved by the PRIMITIVE HAL operation.

File

gfx_primitive.h

C

```
typedef enum {
  GFX_PRIMITIVE_STATE_BUSY,
  GFX_PRIMITIVE_STATE_INIT
} GFX_PRIMITIVE_STATES;
```

Members

Members	Description
GFX_PRIMITIVE_STATE_BUSY	Driver state busy
GFX_PRIMITIVE_STATE_INIT	Driver state init

Description

PRIMITIVE HAL State Machine States

This enumeration defines the various states that can be achieved by the HAL operation.

Remarks

None.

GFX_TEXT_CURSOR_HEIGHT_DEFAULT Macro

Constant that defines the default value of text cursor height.

File

gfx_primitive.h

C

```
#define GFX_TEXT_CURSOR_HEIGHT_DEFAULT 10
```

Description

GFX PRIMITIVE HAL Default Text cursor height

This constant is utilized when cursor height is not defined.

Remarks

None.

GFX_TEXT_CURSOR_WIDTH_DEFAULT Macro

Constant that defines the default value of text cursor width.

File

gfx_primitive.h

C

```
#define GFX_TEXT_CURSOR_WIDTH_DEFAULT 6
```

Description

GFX PRIMITIVE HAL Default Text cursor width

This constant is utilized when cursor width is not defined.

Remarks

GFX_TextCursorWidthGet returns this value if a valid cursor width is not available.

Color Definitions

Macros

	Name	Description
	BLACK	Basic colors definitions. The basic colors defined in this section are defined for basic demo requirements. End application can define additional colors or override existing default colors. If overriding an existing default color, define the new color value before the #include of Graphics.h to avoid the compile time warning. When using palette, a different file 'PaletteColorDefines.h' has to be used instead.
	BLUE	This is macro BLUE.
	BRIGHTBLUE	This is macro BRIGHTBLUE.
	BRIGHTCYAN	This is macro BRIGHTCYAN.
	BRIGHTGREEN	This is macro BRIGHTGREEN.
	BRIGHTMAGENTA	This is macro BRIGHTMAGENTA.
	BRIGHTRED	This is macro BRIGHTRED.
	BRIGHTYELLOW	This is macro BRIGHTYELLOW.
	BROWN	This is macro BROWN.
	BURLYWOOD	This is macro BURLYWOOD.
	CYAN	This is macro CYAN.
	DARKGRAY	This is macro DARKGRAY.
	DARKORANGE	This is macro DARKORANGE.
	GFX_W3_ALICEBLUE	This is macro GFX_W3_ALICEBLUE.
	GFX_W3_ANTIQUEWHITE	This is macro GFX_W3_ANTIQUEWHITE.
	GFX_W3_AQUA	This is macro GFX_W3_AQUA.
	GFX_W3_AQUAMARINE	This is macro GFX_W3_AQUAMARINE.
	GFX_W3_AZURE	This is macro GFX_W3_AZURE.
	GFX_W3_BEIGE	This is macro GFX_W3_BEIGE.
	GFX_W3_BISQUE	This is macro GFX_W3_BISQUE.
	GFX_W3_BLACK	This is macro GFX_W3_BLACK.
	GFX_W3_BLANCHEDALMOND	This is macro GFX_W3_BLANCHEDALMOND.
	GFX_W3_BLUE	This is macro GFX_W3_BLUE.

	GFX_W3_BLUEVIOLET	This is macro GFX_W3_BLUEVIOLET.
	GFX_W3_BROWN	This is macro GFX_W3_BROWN.
	GFX_W3_BURLYWOOD	This is macro GFX_W3_BURLYWOOD.
	GFX_W3_CADETBLUE	This is macro GFX_W3_CADETBLUE.
	GFX_W3_CHARTREUSE	This is macro GFX_W3_CHARTREUSE.
	GFX_W3_CHOCOLATE	This is macro GFX_W3_CHOCOLATE.
	GFX_W3_CORAL	This is macro GFX_W3_CORAL.
	GFX_W3_CORNFLOWERBLUE	This is macro GFX_W3_CORNFLOWERBLUE.
	GFX_W3_CORNSILK	This is macro GFX_W3_CORNSILK.
	GFX_W3_CRIMSON	This is macro GFX_W3_CRIMSON.
	GFX_W3_CYAN	This is macro GFX_W3_CYAN.
	GFX_W3_DARKBLUE	This is macro GFX_W3_DARKBLUE.
	GFX_W3_DARKCYAN	This is macro GFX_W3_DARKCYAN.
	GFX_W3_darkgoldenrod	This is macro GFX_W3_darkgoldenrod.
	GFX_W3_DARKGRAY	This is macro GFX_W3_DARKGRAY.
	GFX_W3_DARKGREEN	This is macro GFX_W3_DARKGREEN.
	GFX_W3_DARKGREY	This is macro GFX_W3_DARKGREY.
	GFX_W3_DARKHAKI	This is macro GFX_W3_DARKHAKI.
	GFX_W3_DARKMAGENTA	This is macro GFX_W3_DARKMAGENTA.
	GFX_W3_DARKOLIVEGREEN	This is macro GFX_W3_DARKOLIVEGREEN.
	GFX_W3_DARKORANGE	This is macro GFX_W3_DARKORANGE.
	GFX_W3_DARKORCHID	This is macro GFX_W3_DARKORCHID.
	GFX_W3_DARKRED	This is macro GFX_W3_DARKRED.
	GFX_W3_DARKSALMON	This is macro GFX_W3_DARKSALMON.
	GFX_W3_DARKSEAGREEN	This is macro GFX_W3_DARKSEAGREEN.
	GFX_W3_DARKSLATEBLUE	This is macro GFX_W3_DARKSLATEBLUE.
	GFX_W3_DARKSLATEGRAY	This is macro GFX_W3_DARKSLATEGRAY.
	GFX_W3_DARKSLATEGREY	This is macro GFX_W3_DARKSLATEGREY.
	GFX_W3_DARKTURQUOISE	This is macro GFX_W3_DARKTURQUOISE.
	GFX_W3_DARKVIOLET	This is macro GFX_W3_DARKVIOLET.
	GFX_W3_DEEPPINK	This is macro GFX_W3_DEEPPINK.
	GFX_W3_DEEPSKYBLUE	This is macro GFX_W3_DEEPSKYBLUE.
	GFX_W3_DIMGRAY	This is macro GFX_W3_DIMGRAY.
	GFX_W3_DIMGREY	This is macro GFX_W3_DIMGREY.
	GFX_W3_DODGERBLUE	This is macro GFX_W3_DODGERBLUE.
	GFX_W3_FIREBRICK	This is macro GFX_W3_FIREBRICK.
	GFX_W3_FLORALWHITE	This is macro GFX_W3_FLORALWHITE.
	GFX_W3_FORESTGREEN	This is macro GFX_W3_FORESTGREEN.
	GFX_W3_FUCHSIA	This is macro GFX_W3_FUCHSIA.
	GFX_W3_GAINSBORO	This is macro GFX_W3_GAINSBORO.
	GFX_W3_GHOSTWHITE	This is macro GFX_W3_GHOSTWHITE.
	GFX_W3_GOLD	This is macro GFX_W3_GOLD.
	GFX_W3_GOLDENROD	This is macro GFX_W3_GOLDENROD.
	GFX_W3_GRAY	This is macro GFX_W3_GRAY.
	GFX_W3_GREEN	This is macro GFX_W3_GREEN.
	GFX_W3_GREENYELLOW	This is macro GFX_W3_GREENYELLOW.
	GFX_W3_GREY	This is macro GFX_W3_GREY.
	GFX_W3_HONEYDEW	This is macro GFX_W3_HONEYDEW.
	GFX_W3_HOTPINK	This is macro GFX_W3_HOTPINK.
	GFX_W3_INDIANRED	This is macro GFX_W3_INDIANRED.
	GFX_W3_INDIGO	This is macro GFX_W3_INDIGO.
	GFX_W3_IVORY	This is macro GFX_W3_IVORY.
	GFX_W3_LAVENDER	This is macro GFX_W3_LAVENDER.
	GFX_W3_LAVENDERBLUSH	This is macro GFX_W3_LAVENDERBLUSH.
	GFX_W3_LAWNGREEN	This is macro GFX_W3_LAWNGREEN.

	GFX_W3_LEMONCHIFFON	This is macro GFX_W3_LEMONCHIFFON.
	GFX_W3_LIGHTBLUE	This is macro GFX_W3_LIGHTBLUE.
	GFX_W3_LIGHTCORAL	This is macro GFX_W3_LIGHTCORAL.
	GFX_W3_LIGHTCYAN	This is macro GFX_W3_LIGHTCYAN.
	GFX_W3_LIGHTGOLDENRODYELLOW	This is macro GFX_W3_LIGHTGOLDENRODYELLOW.
	GFX_W3_LIGHTGRAY	This is macro GFX_W3_LIGHTGRAY.
	GFX_W3_LIGHTGREEN	This is macro GFX_W3_LIGHTGREEN.
	GFX_W3_LIGHTGREY	This is macro GFX_W3_LIGHTGREY.
	GFX_W3_LIGHTPINK	This is macro GFX_W3_LIGHTPINK.
	GFX_W3_LIGHTSALMON	This is macro GFX_W3_LIGHTSALMON.
	GFX_W3_LIGHTSKYBLUE	This is macro GFX_W3_LIGHTSKYBLUE.
	GFX_W3_LIGHTSLATEGRAY	This is macro GFX_W3_LIGHTSLATEGRAY.
	GFX_W3_LIGHTSLATEGREY	This is macro GFX_W3_LIGHTSLATEGREY.
	GFX_W3_LIGHTSTEELBLUE	This is macro GFX_W3_LIGHTSTEELBLUE.
	GFX_W3_LIGHTYELLOW	This is macro GFX_W3_LIGHTYELLOW.
	GFX_W3_LIGTHSEAGREEN	This is macro GFX_W3_LIGTHSEAGREEN.
	GFX_W3_LIME	This is macro GFX_W3_LIME.
	GFX_W3_LIMEGREEN	This is macro GFX_W3_LIMEGREEN.
	GFX_W3_LINEN	This is macro GFX_W3_LINEN.
	GFX_W3_MAGENTA	This is macro GFX_W3_MAGENTA.
	GFX_W3_MAROON	This is macro GFX_W3_MAROON.
	GFX_W3_MEDIUMAQUAMARINE	This is macro GFX_W3_MEDIUMAQUAMARINE.
	GFX_W3_MEDIUMBLUE	This is macro GFX_W3_MEDIUMBLUE.
	GFX_W3_MEDIUMORCHID	This is macro GFX_W3_MEDIUMORCHID.
	GFX_W3_MEDIUMPURPLE	This is macro GFX_W3_MEDIUMPURPLE.
	GFX_W3_MEDIUMSEAGREEN	This is macro GFX_W3_MEDIUMSEAGREEN.
	GFX_W3_MEDIUMSLATEBLUE	This is macro GFX_W3_MEDIUMSLATEBLUE.
	GFX_W3_MEDIUMSPRINGGREEN	This is macro GFX_W3_MEDIUMSPRINGGREEN.
	GFX_W3_MEDIUMTURQUOISE	This is macro GFX_W3_MEDIUMTURQUOISE.
	GFX_W3_MEDIUMVIOLETRED	This is macro GFX_W3_MEDIUMVIOLETRED.
	GFX_W3_MIDNIGHTBLUE	This is macro GFX_W3_MIDNIGHTBLUE.
	GFX_W3_MINTCREAM	This is macro GFX_W3_MINTCREAM.
	GFX_W3_MISTYROSE	This is macro GFX_W3_MISTYROSE.
	GFX_W3_MOCCASIN	This is macro GFX_W3_MOCCASIN.
	GFX_W3_NAVAJOWHITE	This is macro GFX_W3_NAVAJOWHITE.
	GFX_W3_NAVY	This is macro GFX_W3_NAVY.
	GFX_W3_OLDLACE	This is macro GFX_W3_OLDLACE.
	GFX_W3_OLIVE	This is macro GFX_W3_OLIVE.
	GFX_W3_OLIVEDRAB	This is macro GFX_W3_OLIVEDRAB.
	GFX_W3_ORANGE	This is macro GFX_W3_ORANGE.
	GFX_W3_ORANGERED	This is macro GFX_W3_ORANGERED.
	GFX_W3_ORCHID	This is macro GFX_W3_ORCHID.
	GFX_W3_PALEGOLDENROD	This is macro GFX_W3_PALEGOLDENROD.
	GFX_W3_PALEGREEN	This is macro GFX_W3_PALEGREEN.
	GFX_W3_PALETURQUOISE	This is macro GFX_W3_PALETURQUOISE.
	GFX_W3_PALEVIOLETRED	This is macro GFX_W3_PALEVIOLETRED.
	GFX_W3_PAPAYAWHIP	This is macro GFX_W3_PAPAYAWHIP.
	GFX_W3_PEACHPUFF	This is macro GFX_W3_PEACHPUFF.
	GFX_W3_PERU	This is macro GFX_W3_PERU.
	GFX_W3_PINK	This is macro GFX_W3_PINK.
	GFX_W3_PLUM	This is macro GFX_W3_PLUM.
	GFX_W3_POWDERBLUE	This is macro GFX_W3_POWDERBLUE.
	GFX_W3_PURPLE	This is macro GFX_W3_PURPLE.
	GFX_W3_RED	This is macro GFX_W3_RED.
	GFX_W3_ROSYBROWN	This is macro GFX_W3_ROSYBROWN.

	GFX_W3_ROYALBLUE	This is macro GFX_W3_ROYALBLUE.
	GFX_W3_SADDLEBROWN	This is macro GFX_W3_SADDLEBROWN.
	GFX_W3_SALMON	This is macro GFX_W3_SALMON.
	GFX_W3_SANDYGREEN	This is macro GFX_W3_SANDYGREEN.
	GFX_W3_SEAGREEN	This is macro GFX_W3_SEAGREEN.
	GFX_W3_SEASHELL	This is macro GFX_W3_SEASHELL.
	GFX_W3_SIENNA	This is macro GFX_W3_SIENNA.
	GFX_W3_SILVER	This is macro GFX_W3_SILVER.
	GFX_W3_SKYBLUE	This is macro GFX_W3_SKYBLUE.
	GFX_W3_SLATEBLUE	This is macro GFX_W3_SLATEBLUE.
	GFX_W3_SLATEGRAY	This is macro GFX_W3_SLATEGRAY.
	GFX_W3_SLATEGREY	This is macro GFX_W3_SLATEGREY.
	GFX_W3_SNOW	This is macro GFX_W3_SNOW.
	GFX_W3_SPRINGGREEN	This is macro GFX_W3_SPRINGGREEN.
	GFX_W3_STEELBLUE	This is macro GFX_W3_STEELBLUE.
	GFX_W3_TAN	This is macro GFX_W3_TAN.
	GFX_W3_TEAL	This is macro GFX_W3_TEAL.
	GFX_W3_THISTLE	This is macro GFX_W3_THISTLE.
	GFX_W3_TOMATO	This is macro GFX_W3_TOMATO.
	GFX_W3_TURQUOISE	This is macro GFX_W3_TURQUOISE.
	GFX_W3_VIOLET	This is macro GFX_W3_VIOLET.
	GFX_W3_WHEAT	This is macro GFX_W3_WHEAT.
	GFX_W3_WHITE	This is macro GFX_W3_WHITE.
	GFX_W3_WHITESMOKE	This is macro GFX_W3_WHITESMOKE.
	GFX_W3_YELLOW	This is macro GFX_W3_YELLOW.
	GFX_W3_YELLOWGREEN	This is macro GFX_W3_YELLOWGREEN.
	GFX_X11_ALICE_BLUE	This is macro GFX_X11_ALICE_BLUE.
	GFX_X11_ANTIQUE_WHITE	This is macro GFX_X11_ANTIQUE_WHITE.
	GFX_X11_AQUA	This is macro GFX_X11_AQUA.
	GFX_X11_AQUAMARINE	This is macro GFX_X11_AQUAMARINE.
	GFX_X11_AZURE	This is macro GFX_X11_AZURE.
	GFX_X11_BEIGE	This is macro GFX_X11_BEIGE.
	GFX_X11_BISQUE	This is macro GFX_X11_BISQUE.
	GFX_X11_BLACK	This is macro GFX_X11_BLACK.
	GFX_X11_BLANCHED_ALMOND	This is macro GFX_X11_BLANCHED_ALMOND.
	GFX_X11_BLUE	This is macro GFX_X11_BLUE.
	GFX_X11_BLUE_VIOLET	This is macro GFX_X11_BLUE_VIOLET.
	GFX_X11_BROWN	This is macro GFX_X11_BROWN.
	GFX_X11_BURLY_WOOD	This is macro GFX_X11_BURLY_WOOD.
	GFX_X11_CADEL_BLUE	This is macro GFX_X11_CADEL_BLUE.
	GFX_X11_CHARTEUSE	This is macro GFX_X11_CHARTEUSE.
	GFX_X11_CHOCOLATE	This is macro GFX_X11_CHOCOLATE.
	GFX_X11_CORAL	This is macro GFX_X11_CORAL.
	GFX_X11_CORNFLOWER_BLUE	This is macro GFX_X11_CORNFLOWER_BLUE.
	GFX_X11_CORNSILK	X11: Brown Colors
	GFX_X11_CRIMSON	This is macro GFX_X11_CRIMSON.
	GFX_X11_CYAN	This is macro GFX_X11_CYAN.
	GFX_X11_DARK_BLUE	This is macro GFX_X11_DARK_BLUE.
	GFX_X11_DARK_CYAN	This is macro GFX_X11_DARK_CYAN.
	GFX_X11_DARK_GOLDENROD	This is macro GFX_X11_DARK_GOLDENROD.
	GFX_X11_DARK_GREEN	This is macro GFX_X11_DARK_GREEN.
	GFX_X11_DARK_GREY	This is macro GFX_X11_DARK_GREY.
	GFX_X11_DARK_KHAKI	This is macro GFX_X11_DARK_KHAKI.
	GFX_X11_DARK_MAGENTA	This is macro GFX_X11_DARK_MAGENTA.
	GFX_X11_DARK_OLIVE_GREEN	X11: Green Colors

	GFX_X11_DARK_ORANGE	This is macro GFX_X11_DARK_ORANGE.
	GFX_X11_DARK_ORCHID	This is macro GFX_X11_DARK_ORCHID.
	GFX_X11_DARK_RED	This is macro GFX_X11_DARK_RED.
	GFX_X11_DARK_SALMON	This is macro GFX_X11_DARK_SALMON.
	GFX_X11_DARK_SEA_GREEN	This is macro GFX_X11_DARK_SEA_GREEN.
	GFX_X11_DARK_SLATE_BLUE	This is macro GFX_X11_DARK_SLATE_BLUE.
	GFX_X11_DARK_SLATE_GREY	This is macro GFX_X11_DARK_SLATE_GREY.
	GFX_X11_DARK_TURQUOISE	This is macro GFX_X11_DARK_TURQUOISE.
	GFX_X11_DARK_VIOLET	This is macro GFX_X11_DARK_VIOLET.
	GFX_X11_DEEP_PINK	This is macro GFX_X11_DEEP_PINK.
	GFX_X11_DEEP_SKY_BLUE	This is macro GFX_X11_DEEP_SKY_BLUE.
	GFX_X11_DIM_GREY	This is macro GFX_X11_DIM_GREY.
	GFX_X11_DODGER_BLUE	This is macro GFX_X11_DODGER_BLUE.
	GFX_X11_FIRE_BRICK	This is macro GFX_X11_FIRE_BRICK.
	GFX_X11_FLORAL_WHITE	This is macro GFX_X11_FLORAL_WHITE.
	GFX_X11_FOREST_GREEN	This is macro GFX_X11_FOREST_GREEN.
	GFX_X11_FUCHSIA	This is macro GFX_X11_FUCHSIA.
	GFX_X11_GAINSBORO	This is macro GFX_X11_GAINSBORO.
	GFX_X11_GHOST_WHITE	This is macro GFX_X11_GHOST_WHITE.
	GFX_X11_GOLD	This is macro GFX_X11_GOLD.
	GFX_X11_GOLDENROD	This is macro GFX_X11_GOLDENROD.
	GFX_X11_GREEN	This is macro GFX_X11_GREEN.
	GFX_X11_GREEN_YELLOW	This is macro GFX_X11_GREEN_YELLOW.
	GFX_X11_GREY	This is macro GFX_X11_GREY.
	GFX_X11_HONEYDEW	This is macro GFX_X11_HONEYDEW.
	GFX_X11_HOT_PINK	This is macro GFX_X11_HOT_PINK.
	GFX_X11_INDIAN_RED	This is macro GFX_X11_INDIAN_RED.
	GFX_X11_INDIGO	This is macro GFX_X11_INDIGO.
	GFX_X11_IVORY	This is macro GFX_X11_IVORY.
	GFX_X11_KHAKI	This is macro GFX_X11_KHAKI.
	GFX_X11_LAVENDER	X11: BLUE COLORS
	GFX_X11_LAVENDOR_BLUSH	This is macro GFX_X11_LAVENDOR_BLUSH.
	GFX_X11_LAWN_GREEN	This is macro GFX_X11_LAWN_GREEN.
	GFX_X11_LEMON_CHIFFON	This is macro GFX_X11_LEMON_CHIFFON.
	GFX_X11_LIGHT_BLUE	This is macro GFX_X11_LIGHT_BLUE.
	GFX_X11_LIGHT_CAROL	This is macro GFX_X11_LIGHT_CAROL.
	GFX_X11_LIGHT_CYAN	This is macro GFX_X11_LIGHT_CYAN.
	GFX_X11_LIGHT_GOLDENROD_YELLOW	This is macro GFX_X11_LIGHT_GOLDENROD_YELLOW.
	GFX_X11_LIGHT_GRAY	This is macro GFX_X11_LIGHT_GRAY.
	GFX_X11_LIGHT_GREEN	This is macro GFX_X11_LIGHT_GREEN.
	GFX_X11_LIGHT_PINK	This is macro GFX_X11_LIGHT_PINK.
	GFX_X11_LIGHT_SALMON	X11: Red Colors
	GFX_X11_LIGHT_SEA_GREEN	This is macro GFX_X11_LIGHT_SEA_GREEN.
	GFX_X11_LIGHT_SKY_BLUE	This is macro GFX_X11_LIGHT_SKY_BLUE.
	GFX_X11_LIGHT_SLATE_GREY	This is macro GFX_X11_LIGHT_SLATE_GREY.
	GFX_X11_LIGHT_STEEL_BLUE	X11: BLUE COLORS
	GFX_X11_LIGHT_YELLOW	This is macro GFX_X11_LIGHT_YELLOW.
	GFX_X11_LIME	This is macro GFX_X11_LIME.
	GFX_X11_LIME_GREEN	This is macro GFX_X11_LIME_GREEN.
	GFX_X11_LINEN	This is macro GFX_X11_LINEN.
	GFX_X11_MAGENTA	This is macro GFX_X11_MAGENTA.
	GFX_X11_MARRON	This is macro GFX_X11_MARRON.
	GFX_X11_MEDIUM_AQUAMARINE	X11: CYAN COLORS
	GFX_X11_MEDIUM_BLUE	This is macro GFX_X11_MEDIUM_BLUE.
	GFX_X11_MEDIUM_ORCHID	This is macro GFX_X11_MEDIUM_ORCHID.

	GFX_X11_MEDIUM_PURPLE	This is macro GFX_X11_MEDIUM_PURPLE.
	GFX_X11_MEDIUM_SEA_GREEN	This is macro GFX_X11_MEDIUM_SEA_GREEN.
	GFX_X11_MEDIUM_SLATE_BLUE	This is macro GFX_X11_MEDIUM_SLATE_BLUE.
	GFX_X11_MEDIUM_SPRING_GREEN	This is macro GFX_X11_MEDIUM_SPRING_GREEN.
	GFX_X11_MEDIUM_TURQUOISE	This is macro GFX_X11_MEDIUM_TURQUOISE.
	GFX_X11_MEDIUM_VIOLET_RED	This is macro GFX_X11_MEDIUM_VIOLET_RED.
	GFX_X11_MIDNIGHT_BLUE	This is macro GFX_X11_MIDNIGHT_BLUE.
	GFX_X11_MINT_CREAM	This is macro GFX_X11_MINT_CREAM.
	GFX_X11_MISTY_ROSE	This is macro GFX_X11_MISTY_ROSE.
	GFX_X11_MOCCASIN	This is macro GFX_X11_MOCCASIN.
	GFX_X11_NAVAJO_WHITE	This is macro GFX_X11_NAVAJO_WHITE.
	GFX_X11_NAVY	This is macro GFX_X11_NAVY.
	GFX_X11_OLD_LACE	This is macro GFX_X11_OLD_LACE.
	GFX_X11_OLIVE	This is macro GFX_X11_OLIVE.
	GFX_X11_OLIVE_DRAB	This is macro GFX_X11_OLIVE_DRAB.
	GFX_X11_ORANGE	This is macro GFX_X11_ORANGE.
	GFX_X11_ORANGE_RED	X11: Orange Colors
	GFX_X11_ORCHID	This is macro GFX_X11_ORCHID.
	GFX_X11_PALE_GOLDENROD	This is macro GFX_X11_PALE_GOLDENROD.
	GFX_X11_PALE_GREEN	This is macro GFX_X11_PALE_GREEN.
	GFX_X11_PALE_TURQUOISE	This is macro GFX_X11_PALE_TURQUOISE.
	GFX_X11_PALE_VIOLET_RED	This is macro GFX_X11_PALE_VIOLET_RED.
	GFX_X11_PAPAYA_WHIP	This is macro GFX_X11_PAPAYA_WHIP.
	GFX_X11_PEACH_PUFF	This is macro GFX_X11_PEACH_PUFF.
	GFX_X11_PERU	This is macro GFX_X11_PERU.
	GFX_X11_PINK	X11: Pink Colors
	GFX_X11_PLUM	This is macro GFX_X11_PLUM.
	GFX_X11_POWDER_BLUE	This is macro GFX_X11_POWDER_BLUE.
	GFX_X11_PURPLE	This is macro GFX_X11_PURPLE.
	GFX_X11_RED	This is macro GFX_X11_RED.
	GFX_X11_ROSY_BROWN	This is macro GFX_X11_ROSY_BROWN.
	GFX_X11_ROYAL_BLUE	This is macro GFX_X11_ROYAL_BLUE.
	GFX_X11_SADDLE_BROWN	This is macro GFX_X11_SADDLE_BROWN.
	GFX_X11_SALMON	This is macro GFX_X11_SALMON.
	GFX_X11_SANDY_BROWN	This is macro GFX_X11_SANDY_BROWN.
	GFX_X11_SEA_GREEN	This is macro GFX_X11_SEA_GREEN.
	GFX_X11_SEASHELL	This is macro GFX_X11_SEASHELL.
	GFX_X11_SIENNA	This is macro GFX_X11_SIENNA.
	GFX_X11_SILVER	This is macro GFX_X11_SILVER.
	GFX_X11_SKY_BLUE	This is macro GFX_X11_SKY_BLUE.
	GFX_X11_SLATE_BLUE	This is macro GFX_X11_SLATE_BLUE.
	GFX_X11_SLATE_GREY	This is macro GFX_X11_SLATE_GREY.
	GFX_X11_SNOW	This is macro GFX_X11_SNOW.
	GFX_X11_SPRING_GREEN	This is macro GFX_X11_SPRING_GREEN.
	GFX_X11_STEEL_BLUE	This is macro GFX_X11_STEEL_BLUE.
	GFX_X11_TAN	This is macro GFX_X11_TAN.
	GFX_X11_TEAL	This is macro GFX_X11_TEAL.
	GFX_X11_THISTLE	This is macro GFX_X11_THISTLE.
	GFX_X11_TOMATO	This is macro GFX_X11_TOMATO.
	GFX_X11_TURQUOISE	This is macro GFX_X11_TURQUOISE.
	GFX_X11_VIOLET	This is macro GFX_X11_VIOLET.
	GFX_X11_WHEAT	This is macro GFX_X11_WHEAT.
	GFX_X11_WHITE	X11: WHITE/GRAY/BLACK COLORS
	GFX_X11_WHITE_SMOKE	This is macro GFX_X11_WHITE_SMOKE.
	GFX_X11_YELLOW	X11: Yellow Colors

	GFX_X11_YELLOW_GREEN	This is macro GFX_X11_YELLOW_GREEN.
	GOLD	This is macro GOLD.
	GRAY000	This is macro GRAY000.
	GRAY001	This is macro GRAY001.
	GRAY002	This is macro GRAY002.
	GRAY003	This is macro GRAY003.
	GRAY004	This is macro GRAY004.
	GRAY005	This is macro GRAY005.
	GRAY006	This is macro GRAY006.
	GRAY007	This is macro GRAY007.
	GRAY008	This is macro GRAY008.
	GRAY009	This is macro GRAY009.
	GRAY010	This is macro GRAY010.
	GRAY011	This is macro GRAY011.
	GRAY012	This is macro GRAY012.
	GRAY013	This is macro GRAY013.
	GRAY014	This is macro GRAY014.
	GRAY015	This is macro GRAY015.
	GRAY032	This is macro GRAY032.
	GRAY096	This is macro GRAY096.
	GRAY128	This is macro GRAY128.
	GRAY160	This is macro GRAY160.
	GRAY192	This is macro GRAY192.
	GRAY204	This is macro GRAY204.
	GRAY224	This is macro GRAY224.
	GRAY229	This is macro GRAY229.
	GRAY242	This is macro GRAY242.
	GREEN	This is macro GREEN.
	KHAKI	This is macro KHAKI.
	LIGHTBLUE	This is macro LIGHTBLUE.
	LIGHTCYAN	This is macro LIGHTCYAN.
	LIGHTGRAY	This is macro LIGHTGRAY.
	LIGHTGREEN	This is macro LIGHTGREEN.
	LIGHTMAGENTA	This is macro LIGHTMAGENTA.
	LIGHTORANGE	This is macro LIGHTORANGE.
	LIGHTRED	This is macro LIGHTRED.
	LIGHTYELLOW	This is macro LIGHTYELLOW.
	MAGENTA	This is macro MAGENTA.
	ORANGE	This is macro ORANGE.
	PERU	This is macro PERU.
	RED	This is macro RED.
	SADDLEBROWN	This is macro SADDLEBROWN.
	SIENNA	This is macro SIENNA.
	TAN	This is macro TAN.
	WHEAT	This is macro WHEAT.
	WHITE	This is macro WHITE.
	YELLOW	This is macro YELLOW.

Description

BLACK Macro

File

gfx_colors.h

C

```
#define BLACK GFX_RGBConvert(0, 0, 0)
```

Description

Basic colors definitions. The basic colors defined in this section are defined for basic demo requirements. End application can define additional colors or override existing default colors. If overriding an existing default color, define the new color value before the #include of Graphics.h to avoid the compile time warning. When using palette, a different file 'PaletteColorDefines.h' has to be used instead.

BLUE Macro

File

gfx_colors.h

C

```
#define BLUE GFX_RGBConvert(0, 0, 128)
```

Description

This is macro BLUE.

BRIGHTBLUE Macro

File

gfx_colors.h

C

```
#define BRIGHTBLUE GFX_RGBConvert(0, 0, 255)
```

Description

This is macro BRIGHTBLUE.

BRIGHTCYAN Macro

File

gfx_colors.h

C

```
#define BRIGHTCYAN GFX_RGBConvert(0, 255, 255)
```

Description

This is macro BRIGHTCYAN.

BRIGHTGREEN Macro

File

gfx_colors.h

C

```
#define BRIGHTGREEN GFX_RGBConvert(0, 255, 0)
```

Description

This is macro BRIGHTGREEN.

BRIGHTMAGENTA Macro

File

gfx_colors.h

C

```
#define BRIGHTMAGENTA GFX_RGBConvert(255, 0, 255)
```

Description

This is macro BRIGHTMAGENTA.

BRIGHTRED Macro

File

gfx_colors.h

C

```
#define BRIGHTRED GFX_RGBConvert(255, 0, 0)
```

Description

This is macro BRIGHTRED.

BRIGHTYELLOW Macro

File

gfx_colors.h

C

```
#define BRIGHTYELLOW GFX_RGBConvert(255, 255, 0)
```

Description

This is macro BRIGHTYELLOW.

BROWN Macro

File

gfx_colors.h

C

```
#define BROWN GFX_RGBConvert(255, 128, 0)
```

Description

This is macro BROWN.

BURLYWOOD Macro

File

gfx_colors.h

C

```
#define BURLYWOOD GFX_RGBConvert(222, 184, 135)
```

Description

This is macro BURLYWOOD.

CYAN Macro

File

gfx_colors.h

C

```
#define CYAN GFX_RGBConvert(0, 128, 128)
```

Description

This is macro CYAN.

DARKGRAY Macro

File

gfx_colors.h

C

```
#define DARKGRAY GFX_RGBConvert(64, 64, 64)
```

Description

This is macro DARKGRAY.

DARKORANGE Macro

File

gfx_colors.h

C

```
#define DARKORANGE GFX_RGBConvert(255, 140, 0)
```

Description

This is macro DARKORANGE.

GFX_W3_ALICEBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ALICEBLUE GFX_RGBConvert(240, 248, 255)
```

Description

This is macro GFX_W3_ALICEBLUE.

GFX_W3_ANTIQUEWHITE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ANTIQUEWHITE GFX_RGBConvert(250, 235, 215)
```

Description

This is macro GFX_W3_ANTIQUEWHITE.

GFX_W3_AQUA Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_AQUA GFX_RGBConvert( 0, 255, 255)
```

Description

This is macro GFX_W3_AQUA.

GFX_W3_AQUAMARINE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_AQUAMARINE GFX_RGBConvert(127, 255, 212)
```

Description

This is macro GFX_W3_AQUAMARINE.

GFX_W3_AZURE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_AZURE GFX_RGBConvert(240, 255, 255)
```

Description

This is macro GFX_W3_AZURE.

GFX_W3_BEIGE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BEIGE GFX_RGBConvert(245, 245, 220)
```

Description

This is macro GFX_W3_BEIGE.

GFX_W3_BISQUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BISQUE GFX_RGBConvert(255, 228, 196)
```

Description

This is macro GFX_W3_BISQUE.

GFX_W3_BLACK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BLACK GFX_RGBConvert( 0, 0, 0)
```

Description

This is macro GFX_W3_BLACK.

GFX_W3_BLANCHEDALMOND Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BLANCHEDALMOND GFX_RGBConvert(255, 235, 205)
```

Description

This is macro GFX_W3_BLANCHEDALMOND.

GFX_W3_BLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BLUE GFX_RGBConvert( 0, 0, 255)
```

Description

This is macro GFX_W3_BLUE.

GFX_W3_BLUEVIOLET Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BLUEVIOLET GFX_RGBConvert(138, 43, 226)
```

Description

This is macro GFX_W3_BLUEVIOLET.

GFX_W3_BROWN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BROWN GFX_RGBConvert(165, 42, 42)
```

Description

This is macro GFX_W3_BROWN.

GFX_W3_BURLYWOOD Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_BURLYWOOD GFX_RGBConvert(222, 184, 135)
```

Description

This is macro GFX_W3_BURLYWOOD.

GFX_W3_CADETBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CADETBLUE GFX_RGBConvert( 95, 158, 160)
```

Description

This is macro GFX_W3_CADETBLUE.

GFX_W3_CHARTREUSE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CHARTREUSE GFX_RGBConvert(127, 255, 0)
```

Description

This is macro GFX_W3_CHARTREUSE.

GFX_W3_CHOCOLATE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CHOCOLATE GFX_RGBConvert(210, 105, 30)
```

Description

This is macro GFX_W3_CHOCOLATE.

GFX_W3_CORAL Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CORAL GFX_RGBConvert(255, 127, 80)
```

Description

This is macro GFX_W3_CORAL.

GFX_W3_CORNFLOWERBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CORNFLOWERBLUE GFX_RGBConvert(100, 149, 237)
```

Description

This is macro GFX_W3_CORNFLOWERBLUE.

GFX_W3_CORNSILK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CORNSILK GFX_RGBConvert(255, 248, 220)
```

Description

This is macro GFX_W3_CORNSILK.

GFX_W3_CRIMSON Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CRIMSON GFX_RGBConvert(220, 20, 60)
```

Description

This is macro GFX_W3_CRIMSON.

GFX_W3_CYAN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_CYAN GFX_RGBConvert( 0, 255, 255)
```

Description

This is macro GFX_W3_CYAN.

GFX_W3_DARKBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKBLUE GFX_RGBConvert( 0, 0, 139)
```

Description

This is macro GFX_W3_DARKBLUE.

GFX_W3_DARKCYAN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKCYAN GFX_RGBConvert( 0, 139, 139)
```

Description

This is macro GFX_W3_DARKCYAN.

GFX_W3_darkgoldenrod Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_darkgoldenrod GFX_RGBConvert(184, 134, 11)
```

Description

This is macro GFX_W3_darkgoldenrod.

GFX_W3_DARKGRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKGRAY GFX_RGBConvert(169, 169, 169)
```

Description

This is macro GFX_W3_DARKGRAY.

GFX_W3_DARKGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKGREEN GFX_RGBConvert( 0, 100, 0)
```

Description

This is macro GFX_W3_DARKGREEN.

GFX_W3_DARKGREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKGREY GFX_RGBConvert(169, 169, 169)
```

Description

This is macro GFX_W3_DARKGREY.

GFX_W3_DARKHAKI Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKHAKI GFX_RGBConvert(189, 183, 107)
```

Description

This is macro GFX_W3_DARKHAKI.

GFX_W3_DARKMAGENTA Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKMAGENTA GFX_RGBConvert(139, 0, 139)
```

Description

This is macro GFX_W3_DARKMAGENTA.

GFX_W3_DARKOLIVEGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKOLIVEGREEN GFX_RGBConvert( 85, 107, 47)
```

Description

This is macro GFX_W3_DARKOLIVEGREEN.

GFX_W3_DARKORANGE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKORANGE GFX_RGBConvert(255, 140, 0)
```

Description

This is macro GFX_W3_DARKORANGE.

GFX_W3_DARKORCHID Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKORCHID GFX_RGBConvert(153, 50, 204)
```

Description

This is macro GFX_W3_DARKORCHID.

GFX_W3_DARKRED Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKRED GFX_RGBConvert(139, 0, 0)
```

Description

This is macro GFX_W3_DARKRED.

GFX_W3_DARKSALMON Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKSALMON GFX_RGBConvert(233, 150, 122)
```

Description

This is macro GFX_W3_DARKSALMON.

GFX_W3_DARKSEAGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKSEAGREEN GFX_RGBConvert(143, 188, 143)
```

Description

This is macro GFX_W3_DARKSEAGREEN.

GFX_W3_DARKSLATEBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKSLATEBLUE GFX_RGBConvert( 72, 61, 139)
```

Description

This is macro GFX_W3_DARKSLATEBLUE.

GFX_W3_DARKSLATEGRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKSLATEGRAY GFX_RGBConvert( 47, 79, 79)
```

Description

This is macro GFX_W3_DARKSLATEGRAY.

GFX_W3_DARKSLATEGREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKSLATEGREY GFX_RGBConvert( 47, 79, 79)
```

Description

This is macro GFX_W3_DARKSLATEGREY.

GFX_W3_DARKTURQUOISE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKTURQUOISE GFX_RGBConvert( 0, 206, 209)
```

Description

This is macro GFX_W3_DARKTURQUOISE.

GFX_W3_DARKVIOLET Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DARKVIOLET GFX_RGBConvert(148, 0, 211)
```

Description

This is macro GFX_W3_DARKVIOLET.

GFX_W3_DEEPPINK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DEEPPINK GFX_RGBConvert(255, 20, 147)
```

Description

This is macro GFX_W3_DEEPPINK.

GFX_W3_DEEPSKYBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DEEPSKYBLUE GFX_RGBConvert( 0, 191, 255)
```

Description

This is macro GFX_W3_DEEPSKYBLUE.

GFX_W3_DIMGRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DIMGRAY GFX_RGBConvert(105, 105, 105)
```

Description

This is macro GFX_W3_DIMGRAY.

GFX_W3_DIMGREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DIMGREY GFX_RGBConvert(105, 105, 105)
```

Description

This is macro GFX_W3_DIMGREY.

GFX_W3_DODGERBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_DODGERBLUE GFX_RGBConvert( 30, 144, 255)
```

Description

This is macro GFX_W3_DODGERBLUE.

GFX_W3_FIREBRICK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_FIREBRICK GFX_RGBConvert(178, 34, 34)
```

Description

This is macro GFX_W3_FIREBRICK.

GFX_W3_FLORALWHITE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_FLORALWHITE GFX_RGBConvert(255, 250, 240)
```

Description

This is macro GFX_W3_FLORALWHITE.

GFX_W3_FORESTGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_FORESTGREEN GFX_RGBConvert( 34, 139, 34)
```

Description

This is macro GFX_W3_FORESTGREEN.

GFX_W3_FUCHSIA Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_FUCHSIA GFX_RGBConvert(255, 0, 255)
```

Description

This is macro GFX_W3_FUCHSIA.

GFX_W3_GAINSBORO Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GAINSBORO GFX_RGBConvert(220, 220, 220)
```

Description

This is macro GFX_W3_GAINSBORO.

GFX_W3_GHOSTWHITE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GHOSTWHITE GFX_RGBConvert(248, 248, 255)
```

Description

This is macro GFX_W3_GHOSTWHITE.

GFX_W3_GOLD Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GOLD GFX_RGBConvert(255, 215, 0)
```

Description

This is macro GFX_W3_GOLD.

GFX_W3_GOLDENROD Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GOLDENROD GFX_RGBConvert(218, 165, 32)
```

Description

This is macro GFX_W3_GOLDENROD.

GFX_W3_GRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GRAY GFX_RGBConvert(128, 128, 128)
```

Description

This is macro GFX_W3_GRAY.

GFX_W3_GREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GREEN GFX_RGBConvert( 0, 128, 0)
```

Description

This is macro GFX_W3_GREEN.

GFX_W3_GREENYELLOW Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GREENYELLOW GFX_RGBConvert(173, 255, 47)
```

Description

This is macro GFX_W3_GREENYELLOW.

GFX_W3_GREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_GREY GFX_RGBConvert(128, 128, 128)
```

Description

This is macro GFX_W3_GREY.

GFX_W3_HONEYDEW Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_HONEYDEW GFX_RGBConvert(240, 255, 240)
```

Description

This is macro GFX_W3_HONEYDEW.

GFX_W3_HOTPINK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_HOTPINK GFX_RGBConvert(255, 105, 180)
```

Description

This is macro GFX_W3_HOTPINK.

GFX_W3_INDIANRED Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_INDIANRED GFX_RGBConvert(205, 92, 92)
```

Description

This is macro GFX_W3_INDIANRED.

GFX_W3_INDIGO Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_INDIGO GFX_RGBConvert( 75, 0, 130)
```

Description

This is macro GFX_W3_INDIGO.

GFX_W3_IVORY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_IVORY GFX_RGBConvert(255, 255, 240)
```

Description

This is macro GFX_W3_IVORY.

GFX_W3_LAVENDER Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LAVENDER GFX_RGBConvert(230, 230, 250)
```

Description

This is macro GFX_W3_LAVENDER.

GFX_W3_LAVENDERBLUSH Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LAVENDERBLUSH GFX_RGBConvert(255, 240, 245)
```

Description

This is macro GFX_W3_LAVENDERBLUSH.

GFX_W3_LAWNGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LAWNGREEN GFX_RGBConvert(124, 252, 0)
```

Description

This is macro GFX_W3_LAWNGREEN.

GFX_W3_LEMONCHIFFON Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LEMONCHIFFON GFX_RGBConvert(255, 250, 205)
```

Description

This is macro GFX_W3_LEMONCHIFFON.

GFX_W3_LIGHTBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTBLUE GFX_RGBConvert(173, 216, 230)
```

Description

This is macro GFX_W3_LIGHTBLUE.

GFX_W3_LIGHTCORAL Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTCORAL GFX_RGBConvert(240, 128, 128)
```

Description

This is macro GFX_W3_LIGHTCORAL.

GFX_W3_LIGHTCYAN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTCYAN GFX_RGBConvert(224, 255, 255)
```

Description

This is macro GFX_W3_LIGHTCYAN.

GFX_W3_LIGHTGOLDENRODYELLOW Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTGOLDENRODYELLOW GFX_RGBConvert(250, 250, 210)
```

Description

This is macro GFX_W3_LIGHTGOLDENRODYELLOW.

GFX_W3_LIGHTGRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTGRAY GFX_RGBConvert(211, 211, 211)
```

Description

This is macro GFX_W3_LIGHTGRAY.

GFX_W3_LIGHTGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTGREEN GFX_RGBConvert(144, 238, 144)
```

Description

This is macro GFX_W3_LIGHTGREEN.

GFX_W3_LIGHTGREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTGREY GFX_RGBConvert(211, 211, 211)
```

Description

This is macro GFX_W3_LIGHTGREY.

GFX_W3_LIGHTPINK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTPINK GFX_RGBConvert(255, 182, 193)
```

Description

This is macro GFX_W3_LIGHTPINK.

GFX_W3_LIGHTSALMON Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTSALMON GFX_RGBConvert(255, 160, 122)
```

Description

This is macro GFX_W3_LIGHTSALMON.

GFX_W3_LIGHTSKYBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTSKYBLUE GFX_RGBConvert(135, 206, 250)
```

Description

This is macro GFX_W3_LIGHTSKYBLUE.

GFX_W3_LIGHTSLATEGRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTSLATEGRAY GFX_RGBConvert(119, 136, 153)
```

Description

This is macro GFX_W3_LIGHTSLATEGRAY.

GFX_W3_LIGHTSLATEGREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTSLATEGREY GFX_RGBConvert(119, 136, 153)
```

Description

This is macro GFX_W3_LIGHTSLATEGREY.

GFX_W3_LIGHTSTEELBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTSTEELBLUE GFX_RGBConvert(176, 196, 222)
```

Description

This is macro GFX_W3_LIGHTSTEELBLUE.

GFX_W3_LIGHTYELLOW Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGHTYELLOW GFX_RGBConvert(255, 255, 224)
```

Description

This is macro GFX_W3_LIGHTYELLOW.

GFX_W3_LIGTHSEAGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIGTHSEAGREEN GFX_RGBConvert( 32, 178, 170)
```

Description

This is macro GFX_W3_LIGTHSEAGREEN.

GFX_W3_LIME Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIME GFX_RGBConvert( 0, 255, 0)
```

Description

This is macro GFX_W3_LIME.

GFX_W3_LIMEGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LIMEGREEN GFX_RGBConvert( 50, 205, 50)
```

Description

This is macro GFX_W3_LIMEGREEN.

GFX_W3_LINEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_LINEN GFX_RGBConvert(250, 240, 230)
```

Description

This is macro GFX_W3_LINEN.

GFX_W3_MAGENTA Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MAGENTA GFX_RGBConvert(255, 0, 255)
```

Description

This is macro GFX_W3_MAGENTA.

GFX_W3_MAROON Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MAROON GFX_RGBConvert(128, 0, 0)
```

Description

This is macro GFX_W3_MAROON.

GFX_W3_MEDIUMAQUAMARINE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMAQUAMARINE GFX_RGBConvert(102, 205, 170)
```

Description

This is macro GFX_W3_MEDIUMAQUAMARINE.

GFX_W3_MEDIUMBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMBLUE GFX_RGBConvert( 0, 0, 205)
```

Description

This is macro GFX_W3_MEDIUMBLUE.

GFX_W3_MEDIUMORCHID Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMORCHID GFX_RGBConvert(186, 85, 211)
```

Description

This is macro GFX_W3_MEDIUMORCHID.

GFX_W3_MEDIUMPURPLE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMPURPLE GFX_RGBConvert(147, 112, 219)
```

Description

This is macro GFX_W3_MEDIUMPURPLE.

GFX_W3_MEDIUMSEAGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMSEAGREEN GFX_RGBConvert( 60, 179, 113)
```

Description

This is macro GFX_W3_MEDIUMSEAGREEN.

GFX_W3_MEDIUMSLATEBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMSLATEBLUE GFX_RGBConvert(123, 104, 238)
```

Description

This is macro GFX_W3_MEDIUMSLATEBLUE.

GFX_W3_MEDIUMSPRINGGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMSPRINGGREEN GFX_RGBConvert( 0, 250, 154)
```

Description

This is macro GFX_W3_MEDIUMSPRINGGREEN.

GFX_W3_MEDIUMTURQUOISE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMTURQUOISE GFX_RGBConvert( 72, 209, 204)
```

Description

This is macro GFX_W3_MEDIUMTURQUOISE.

GFX_W3_MEDIUMVIOLETRED Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MEDIUMVIOLETRED GFX_RGBConvert(199, 21, 133)
```

Description

This is macro GFX_W3_MEDIUMVIOLETRED.

GFX_W3_MIDNIGHTBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MIDNIGHTBLUE GFX_RGBConvert( 25, 25, 112)
```

Description

This is macro GFX_W3_MIDNIGHTBLUE.

GFX_W3_MINTCREAM Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MINTCREAM GFX_RGBConvert(245, 255, 250)
```

Description

This is macro GFX_W3_MINTCREAM.

GFX_W3_MISTYROSE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MISTYROSE GFX_RGBConvert(255, 228, 225)
```

Description

This is macro GFX_W3_MISTYROSE.

GFX_W3_MOCCASIN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_MOCCASIN GFX_RGBConvert(255, 228, 181)
```

Description

This is macro GFX_W3_MOCCASIN.

GFX_W3_NAVAJOWHITE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_NAVAJOWHITE GFX_RGBConvert(255, 222, 173)
```

Description

This is macro GFX_W3_NAVAJOWHITE.

GFX_W3_NAVY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_NAVY GFX_RGBConvert( 0, 0, 128)
```

Description

This is macro GFX_W3_NAVY.

GFX_W3_OLDLACE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_OLDLACE GFX_RGBConvert(253, 245, 230)
```

Description

This is macro GFX_W3_OLDLACE.

GFX_W3_OLIVE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_OLIVE GFX_RGBConvert(128, 128, 0)
```

Description

This is macro GFX_W3_OLIVE.

GFX_W3_OLIVEDRAB Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_OLIVEDRAB GFX_RGBConvert(107, 142, 35)
```

Description

This is macro GFX_W3_OLIVEDRAB.

GFX_W3_ORANGE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ORANGE GFX_RGBConvert(255, 165, 0)
```

Description

This is macro GFX_W3_ORANGE.

GFX_W3_ORANGERED Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ORANGERED GFX_RGBConvert(255, 69, 0)
```

Description

This is macro GFX_W3_ORANGERED.

GFX_W3_ORCHID Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ORCHID GFX_RGBConvert(218, 112, 214)
```

Description

This is macro GFX_W3_ORCHID.

GFX_W3_PALEGOLDENROD Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PALEGOLDENROD GFX_RGBConvert(238, 232, 170)
```

Description

This is macro GFX_W3_PALEGOLDENROD.

GFX_W3_PALEGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PALEGREEN GFX_RGBConvert(152, 251, 152)
```

Description

This is macro GFX_W3_PALEGREEN.

GFX_W3_PALETURQUOISE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PALETURQUOISE GFX_RGBConvert(175, 238, 238)
```

Description

This is macro GFX_W3_PALETURQUOISE.

GFX_W3_PALEVIOLETRED Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PALEVIOLETRED GFX_RGBConvert(219, 112, 147)
```

Description

This is macro GFX_W3_PALEVIOLETRED.

GFX_W3_PAPAYAWHIP Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PAPAYAWHIP GFX_RGBConvert(255, 239, 213)
```

Description

This is macro GFX_W3_PAPAYAWHIP.

GFX_W3_PEACHPUFF Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PEACHPUFF GFX_RGBConvert(255, 218, 185)
```

Description

This is macro GFX_W3_PEACHPUFF.

GFX_W3_PERU Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PERU GFX_RGBConvert(205, 133, 63)
```

Description

This is macro GFX_W3_PERU.

GFX_W3_PINK Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PINK GFX_RGBConvert(255, 192, 203)
```

Description

This is macro GFX_W3_PINK.

GFX_W3_PLUM Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PLUM GFX_RGBConvert(221, 160, 221)
```

Description

This is macro GFX_W3_PLUM.

GFX_W3_POWDERBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_POWDERBLUE GFX_RGBConvert(176, 224, 230)
```

Description

This is macro GFX_W3_POWDERBLUE.

GFX_W3_PURPLE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_PURPLE GFX_RGBConvert(128, 0, 128)
```

Description

This is macro GFX_W3_PURPLE.

GFX_W3_RED Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_RED GFX_RGBConvert(255, 0, 0)
```

Description

This is macro GFX_W3_RED.

GFX_W3_ROSYBROWN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ROSYBROWN GFX_RGBConvert(188, 143, 143)
```

Description

This is macro GFX_W3_ROSYBROWN.

GFX_W3_ROYALBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_ROYALBLUE GFX_RGBConvert( 65, 105, 225)
```

Description

This is macro GFX_W3_ROYALBLUE.

GFX_W3_SADDLEBROWN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SADDLEBROWN GFX_RGBConvert(139, 69, 19)
```

Description

This is macro GFX_W3_SADDLEBROWN.

GFX_W3_SALMON Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SALMON GFX_RGBConvert(250, 128, 114)
```

Description

This is macro GFX_W3_SALMON.

GFX_W3_SANDYGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SANDYGREEN GFX_RGBConvert(244, 164, 96)
```

Description

This is macro GFX_W3_SANDYGREEN.

GFX_W3_SEAGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SEAGREEN GFX_RGBConvert( 46, 139, 87)
```

Description

This is macro GFX_W3_SEAGREEN.

GFX_W3_SEASHELL Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SEASHELL GFX_RGBConvert(255, 245, 238)
```

Description

This is macro GFX_W3_SEASHELL.

GFX_W3_SIENNA Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SIENNA GFX_RGBConvert(160, 82, 45)
```

Description

This is macro GFX_W3_SIENNA.

GFX_W3_SILVER Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SILVER GFX_RGBConvert(192, 192, 192)
```

Description

This is macro GFX_W3_SILVER.

GFX_W3_SKYBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SKYBLUE GFX_RGBConvert(135, 206, 235)
```

Description

This is macro GFX_W3_SKYBLUE.

GFX_W3_SLATEBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SLATEBLUE GFX_RGBConvert(106, 90, 205)
```

Description

This is macro GFX_W3_SLATEBLUE.

GFX_W3_SLATEGRAY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SLATEGRAY GFX_RGBConvert(112, 128, 144)
```

Description

This is macro GFX_W3_SLATEGRAY.

GFX_W3_SLATEGREY Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SLATEGREY GFX_RGBConvert(112, 128, 144)
```

Description

This is macro GFX_W3_SLATEGREY.

GFX_W3_SNOW Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SNOW GFX_RGBConvert(255, 250, 250)
```

Description

This is macro GFX_W3_SNOW.

GFX_W3_SPRINGGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_SPRINGGREEN GFX_RGBConvert( 0, 255, 127)
```

Description

This is macro GFX_W3_SPRINGGREEN.

GFX_W3_STEELBLUE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_STEELBLUE GFX_RGBConvert( 70, 130, 180)
```

Description

This is macro GFX_W3_STEELBLUE.

GFX_W3_TAN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_TAN GFX_RGBConvert(210, 180, 140)
```

Description

This is macro GFX_W3_TAN.

GFX_W3_TEAL Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_TEAL GFX_RGBConvert( 0, 128, 128)
```

Description

This is macro GFX_W3_TEAL.

GFX_W3_THISTLE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_THISTLE GFX_RGBConvert(216, 191, 216)
```

Description

This is macro GFX_W3_THISTLE.

GFX_W3_TOMATO Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_TOMATO GFX_RGBConvert(255, 99, 71)
```

Description

This is macro GFX_W3_TOMATO.

GFX_W3_TURQUOISE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_TURQUOISE GFX_RGBConvert( 64, 224, 208)
```

Description

This is macro GFX_W3_TURQUOISE.

GFX_W3_VIOLET Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_VIOLET GFX_RGBConvert(238, 130, 238)
```

Description

This is macro GFX_W3_VIOLET.

GFX_W3_WHEAT Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_WHEAT GFX_RGBConvert(245, 222, 179)
```

Description

This is macro GFX_W3_WHEAT.

GFX_W3_WHITE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_WHITE GFX_RGBConvert(255, 255, 255)
```

Description

This is macro GFX_W3_WHITE.

GFX_W3_WHITESMOKE Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_WHITESMOKE GFX_RGBConvert(245, 245, 245)
```

Description

This is macro GFX_W3_WHITESMOKE.

GFX_W3_YELLOW Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_YELLOW GFX_RGBConvert(255, 255, 0)
```

Description

This is macro GFX_W3_YELLOW.

GFX_W3_YELLOWGREEN Macro

File

gfx_colors_w3.h

C

```
#define GFX_W3_YELLOWGREEN GFX_RGBConvert(154, 205, 50)
```

Description

This is macro GFX_W3_YELLOWGREEN.

GFX_X11_ALICE_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ALICE_BLUE GFX_RGBConvert(0xF0, 0xF8, 0xFF)
```

Description

This is macro GFX_X11_ALICE_BLUE.

GFX_X11_ANTIQUE_WHITE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ANTIQUE_WHITE GFX_RGBConvert(0xFA, 0xEB, 0xD7)
```

Description

This is macro GFX_X11_ANTIQUE_WHITE.

GFX_X11_AQUA Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_AQUA GFX_RGBConvert(0x00, 0xFF, 0xFF)
```

Description

This is macro GFX_X11_AQUA.

GFX_X11_AQUAMARINE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_AQUAMARINE GFX_RGBConvert(0x7F, 0xFF, 0xD4)
```

Description

This is macro GFX_X11_AQUAMARINE.

GFX_X11_AZURE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_AZURE GFX_RGBConvert(0xF0, 0xFF, 0xFF)
```

Description

This is macro GFX_X11_AZURE.

GFX_X11_BEIGE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BEIGE GFX_RGBConvert(0xF5, 0xF5, 0xDC)
```

Description

This is macro GFX_X11_BEIGE.

GFX_X11_BISQUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BISQUE GFX_RGBConvert(0xFF, 0xE4, 0xC4)
```

Description

This is macro GFX_X11_BISQUE.

GFX_X11_BLACK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BLACK GFX_RGBConvert(0x00, 0x00, 0x00)
```

Description

This is macro GFX_X11_BLACK.

GFX_X11_BLANCHED_ALMOND Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BLANCHED_ALMOND GFX_RGBConvert(0xFF, 0xEB, 0xCD)
```

Description

This is macro GFX_X11_BLANCHED_ALMOND.

GFX_X11_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BLUE GFX_RGBConvert(0x00, 0x00, 0xFF)
```

Description

This is macro GFX_X11_BLUE.

GFX_X11_BLUE_VIOLET Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BLUE_VIOLET GFX_RGBConvert(0x8A, 0x2B, 0xE2)
```

Description

This is macro GFX_X11_BLUE_VIOLET.

GFX_X11_BROWN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BROWN GFX_RGBConvert(0xA5, 0x2A, 0x2A)
```

Description

This is macro GFX_X11_BROWN.

GFX_X11_BURLY_WOOD Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_BURLY_WOOD GFX_RGBConvert(0xDE, 0xB8, 0x87)
```

Description

This is macro GFX_X11_BURLY_WOOD.

GFX_X11_CADEL_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CADEL_BLUE GFX_RGBConvert(0x5F, 0x9E, 0xA0)
```

Description

This is macro GFX_X11_CADEL_BLUE.

GFX_X11_CHARTEUSE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CHARTEUSE GFX_RGBConvert(0x7F, 0xFF, 0x00)
```

Description

This is macro GFX_X11_CHARTEUSE.

GFX_X11_CHOCOLATE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CHOCOLATE GFX_RGBConvert(0xD2, 0x69, 0x1E)
```

Description

This is macro GFX_X11_CHOCOLATE.

GFX_X11_CORAL Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CORAL GFX_RGBConvert(0xFF, 0x7F, 0x50)
```

Description

This is macro GFX_X11_CORAL.

GFX_X11_CORNFLOWER_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CORNFLOWER_BLUE GFX_RGBConvert(0x64, 0x95, 0xED)
```

Description

This is macro GFX_X11_CORNFLOWER_BLUE.

GFX_X11_CORNSILK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CORNSILK GFX_RGBConvert(0xFF, 0xF8, 0xDC)
```

Description

X11: Brown Colors

GFX_X11_CRIMSON Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CRIMSON GFX_RGBConvert(0xDC, 0x14, 0x3C)
```

Description

This is macro GFX_X11_CRIMSON.

GFX_X11_CYAN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_CYAN GFX_RGBConvert(0x00, 0xFF, 0xFF)
```

Description

This is macro GFX_X11_CYAN.

GFX_X11_DARK_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_BLUE GFX_RGBConvert(0x00, 0x00, 0x8B)
```

Description

This is macro GFX_X11_DARK_BLUE.

GFX_X11_DARK_CYAN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_CYAN GFX_RGBConvert(0x00, 0x8B, 0x8B)
```

Description

This is macro GFX_X11_DARK_CYAN.

GFX_X11_DARK_GOLDENROD Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_GOLDENROD GFX_RGBConvert(0xB8, 0x86, 0x0B)
```

Description

This is macro GFX_X11_DARK_GOLDENROD.

GFX_X11_DARK_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_GREEN GFX_RGBConvert(0x00, 0x64, 0x00)
```

Description

This is macro GFX_X11_DARK_GREEN.

GFX_X11_DARK_GREY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_GREY GFX_RGBConvert(0xA9, 0xA9, 0xA9)
```

Description

This is macro GFX_X11_DARK_GREY.

GFX_X11_DARK_KHAKI Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_KHAKI GFX_RGBConvert(0xBD, 0xB7, 0x6B)
```

Description

This is macro GFX_X11_DARK_KHAKI.

GFX_X11_DARK_MAGENTA Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_MAGENTA GFX_RGBConvert(0x8B, 0x00, 0x8B)
```

Description

This is macro GFX_X11_DARK_MAGENTA.

GFX_X11_DARK_OLIVE_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_OLIVE_GREEN GFX_RGBConvert(0x55, 0x6B, 0x2F)
```

Description

X11: Green Colors

GFX_X11_DARK_ORANGE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_ORANGE GFX_RGBConvert(0xFF, 0x8C, 0x00)
```

Description

This is macro GFX_X11_DARK_ORANGE.

GFX_X11_DARK_ORCHID Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_ORCHID GFX_RGBConvert(0x99, 0x32, 0xCC)
```

Description

This is macro GFX_X11_DARK_ORCHID.

GFX_X11_DARK_RED Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_RED GFX_RGBConvert(0x8B, 0x00, 0x00)
```

Description

This is macro GFX_X11_DARK_RED.

GFX_X11_DARK_SALMON Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_SALMON GFX_RGBConvert(0xEA, 0x96, 0x7A)
```

Description

This is macro GFX_X11_DARK_SALMON.

GFX_X11_DARK_SEA_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_SEA_GREEN GFX_RGBConvert(0x8F, 0xBC, 0x8F)
```

Description

This is macro GFX_X11_DARK_SEA_GREEN.

GFX_X11_DARK_SLATE_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_SLATE_BLUE GFX_RGBConvert(0x48, 0x3D, 0x8B)
```

Description

This is macro GFX_X11_DARK_SLATE_BLUE.

GFX_X11_DARK_SLATE_GREY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_SLATE_GREY GFX_RGBConvert(0x2F, 0x4F, 0x4F)
```

Description

This is macro GFX_X11_DARK_SLATE_GREY.

GFX_X11_DARK_TURQUOISE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_TURQUOISE GFX_RGBConvert(0x00, 0xCE, 0xD1)
```

Description

This is macro GFX_X11_DARK_TURQUOISE.

GFX_X11_DARK_VIOLET Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DARK_VIOLET GFX_RGBConvert(0x94, 0x00, 0xD3)
```

Description

This is macro GFX_X11_DARK_VIOLET.

GFX_X11_DEEP_PINK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DEEP_PINK GFX_RGBConvert(0xFF, 0x14, 0x93)
```

Description

This is macro GFX_X11_DEEP_PINK.

GFX_X11_DEEP_SKY_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DEEP_SKY_BLUE GFX_RGBConvert(0x00, 0xBF, 0xFF)
```

Description

This is macro GFX_X11_DEEP_SKY_BLUE.

GFX_X11_DIM_GREY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DIM_GREY GFX_RGBConvert(0x69, 0x69, 0x69)
```

Description

This is macro GFX_X11_DIM_GREY.

GFX_X11_DODGER_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_DODGER_BLUE GFX_RGBConvert(0x1E, 0x90, 0xFF)
```

Description

This is macro GFX_X11_DODGER_BLUE.

GFX_X11_FIRE_BRICK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_FIRE_BRICK GFX_RGBConvert(0xB2, 0x22, 0x22)
```

Description

This is macro GFX_X11_FIRE_BRICK.

GFX_X11_FLORAL_WHITE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_FLORAL_WHITE GFX_RGBConvert(0xFF, 0xFA, 0xF0)
```

Description

This is macro GFX_X11_FLORAL_WHITE.

GFX_X11_FOREST_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_FOREST_GREEN GFX_RGBConvert(0x22, 0x8B, 0x22)
```

Description

This is macro GFX_X11_FOREST_GREEN.

GFX_X11_FUCHSIA Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_FUCHSIA GFX_RGBConvert(0xFF, 0x00, 0xFF)
```

Description

This is macro GFX_X11_FUCHSIA.

GFX_X11_GAINSBORO Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GAINSBORO GFX_RGBConvert(0xDC, 0xDC, 0xDC)
```

Description

This is macro GFX_X11_GAINSBORO.

GFX_X11_GHOST_WHITE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GHOST_WHITE GFX_RGBConvert(0xF8, 0xF8, 0xFF)
```

Description

This is macro GFX_X11_GHOST_WHITE.

GFX_X11_GOLD Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GOLD GFX_RGBConvert(0xFF, 0xD7, 0x00)
```

Description

This is macro GFX_X11_GOLD.

GFX_X11_GOLDENROD Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GOLDENROD GFX_RGBConvert(0xDA, 0xA5, 0x20)
```

Description

This is macro GFX_X11_GOLDENROD.

GFX_X11_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GREEN GFX_RGBConvert(0x00, 0x80, 0x00)
```

Description

This is macro GFX_X11_GREEN.

GFX_X11_GREEN_YELLOW Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GREEN_YELLOW GFX_RGBConvert(0xAD, 0xFF, 0x2F)
```

Description

This is macro GFX_X11_GREEN_YELLOW.

GFX_X11_GREY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_GREY GFX_RGBConvert(0x80, 0x80, 0x80)
```

Description

This is macro GFX_X11_GREY.

GFX_X11_HONEYDEW Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_HONEYDEW GFX_RGBConvert(0xF0, 0xFF, 0xF0)
```

Description

This is macro GFX_X11_HONEYDEW.

GFX_X11_HOT_PINK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_HOT_PINK GFX_RGBConvert(0xFF, 0x69, 0xB4)
```

Description

This is macro GFX_X11_HOT_PINK.

GFX_X11_INDIAN_RED Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_INDIAN_RED GFX_RGBConvert(0xCD, 0x5C, 0x5C)
```

Description

This is macro GFX_X11_INDIAN_RED.

GFX_X11_INDIGO Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_INDIGO GFX_RGBConvert(0x4B, 0x00, 0x82)
```

Description

This is macro GFX_X11_INDIGO.

GFX_X11_IVORY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_IVORY GFX_RGBConvert(0xFF, 0xFF, 0xF0)
```

Description

This is macro GFX_X11_IVORY.

GFX_X11_KHAKI Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_KHAKI GFX_RGBConvert(0xF0, 0xE6, 0x8C)
```

Description

This is macro GFX_X11_KHAKI.

GFX_X11_LAVENDER Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LAVENDER GFX_RGBConvert(0xE6, 0xE6, 0xFA)
```

Description

X11: BLUE COLORS

GFX_X11_LAVENDOR_BLUSH Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LAVENDOR_BLUSH GFX_RGBConvert(0xFF, 0xF0, 0xF5)
```

Description

This is macro GFX_X11_LAVENDOR_BLUSH.

GFX_X11_LAWN_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LAWN_GREEN GFX_RGBConvert(0x7C, 0xFC, 0x00)
```

Description

This is macro GFX_X11_LAWN_GREEN.

GFX_X11_LEMON_CHIFFON Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LEMON_CHIFFON GFX_RGBConvert(0xFF, 0xFA, 0xCD)
```

Description

This is macro GFX_X11_LEMON_CHIFFON.

GFX_X11_LIGHT_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_BLUE GFX_RGBConvert(0xAD, 0xD8, 0xE6)
```

Description

This is macro GFX_X11_LIGHT_BLUE.

GFX_X11_LIGHT_CAROL Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_CAROL GFX_RGBConvert(0xF0, 0x80, 0x80)
```

Description

This is macro GFX_X11_LIGHT_CAROL.

GFX_X11_LIGHT_CYAN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_CYAN GFX_RGBConvert(0xE0, 0xFF, 0xFF)
```

Description

This is macro GFX_X11_LIGHT_CYAN.

GFX_X11_LIGHT_GOLDENROD_YELLOW Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_GOLDENROD_YELLOW GFX_RGBConvert(0xFA, 0xFA, 0xD2)
```

Description

This is macro GFX_X11_LIGHT_GOLDENROD_YELLOW.

GFX_X11_LIGHT_GRAY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_GRAY GFX_RGBConvert(0xD3, 0xD3, 0xD3)
```

Description

This is macro GFX_X11_LIGHT_GRAY.

GFX_X11_LIGHT_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_GREEN GFX_RGBConvert(0x90, 0xEE, 0x90)
```

Description

This is macro GFX_X11_LIGHT_GREEN.

GFX_X11_LIGHT_PINK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_PINK GFX_RGBConvert(0xFF, 0xB6, 0xC1)
```

Description

This is macro GFX_X11_LIGHT_PINK.

GFX_X11_LIGHT_SALMON Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_SALMON GFX_RGBConvert(0xFF, 0xA0, 0x7A)
```

Description

X11: Red Colors

GFX_X11_LIGHT_SEA_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_SEA_GREEN GFX_RGBConvert(0x20, 0xB2, 0xAA)
```

Description

This is macro GFX_X11_LIGHT_SEA_GREEN.

GFX_X11_LIGHT_SKY_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_SKY_BLUE GFX_RGBConvert(0x87, 0xCE, 0xFA)
```

Description

This is macro GFX_X11_LIGHT_SKY_BLUE.

GFX_X11_LIGHT_SLATE_GREY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_SLATE_GREY GFX_RGBConvert(0x77, 0x88, 0x99)
```

Description

This is macro GFX_X11_LIGHT_SLATE_GREY.

GFX_X11_LIGHT_STEEL_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_STEEL_BLUE GFX_RGBConvert(0xB0, 0xC4, 0xDE)
```

Description

X11: BLUE COLORS

GFX_X11_LIGHT_YELLOW Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIGHT_YELLOW GFX_RGBConvert(0xFF, 0xFF, 0xE0)
```

Description

This is macro GFX_X11_LIGHT_YELLOW.

GFX_X11_LIME Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIME GFX_RGBConvert(0x00, 0xFF, 0x00)
```

Description

This is macro GFX_X11_LIME.

GFX_X11_LIME_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LIME_GREEN GFX_RGBConvert(0x32, 0xCD, 0x32)
```

Description

This is macro GFX_X11_LIME_GREEN.

GFX_X11_LINEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_LINEN GFX_RGBConvert(0xFA, 0xF0, 0xE6)
```

Description

This is macro GFX_X11_LINEN.

GFX_X11_MAGENTA Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MAGENTA GFX_RGBConvert(0xFF, 0x00, 0xFF)
```

Description

This is macro GFX_X11_MAGENTA.

GFX_X11_MARRON Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MARRON GFX_RGBConvert(0x80, 0x00, 0x00)
```

Description

This is macro GFX_X11_MARRON.

GFX_X11_MEDIUM_AQUAMARINE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_AQUAMARINE GFX_RGBConvert(0x66, 0xCD, 0xAA)
```

Description

X11: CYAN COLORS

GFX_X11_MEDIUM_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_BLUE GFX_RGBConvert(0x00, 0x00, 0xCD)
```

Description

This is macro GFX_X11_MEDIUM_BLUE.

GFX_X11_MEDIUM_ORCHID Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_ORCHID GFX_RGBConvert(0xBA, 0x55, 0xD3)
```

Description

This is macro GFX_X11_MEDIUM_ORCHID.

GFX_X11_MEDIUM_PURPLE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_PURPLE GFX_RGBConvert(0x93, 0x70, 0xDB)
```

Description

This is macro GFX_X11_MEDIUM_PURPLE.

GFX_X11_MEDIUM_SEA_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_SEA_GREEN GFX_RGBConvert(0x3C, 0xB3, 0x71)
```

Description

This is macro GFX_X11_MEDIUM_SEA_GREEN.

GFX_X11_MEDIUM_SLATE_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_SLATE_BLUE GFX_RGBConvert(0x7B, 0x68, 0xEE)
```

Description

This is macro GFX_X11_MEDIUM_SLATE_BLUE.

GFX_X11_MEDIUM_SPRING_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_SPRING_GREEN GFX_RGBConvert(0x00, 0xFA, 0x9A)
```

Description

This is macro GFX_X11_MEDIUM_SPRING_GREEN.

GFX_X11_MEDIUM_TURQUOISE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_TURQUOISE GFX_RGBConvert(0x48, 0xD1, 0xCC)
```

Description

This is macro GFX_X11_MEDIUM_TURQUOISE.

GFX_X11_MEDIUM_VIOLET_RED Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MEDIUM_VIOLET_RED GFX_RGBConvert(0xC7, 0x15, 0x85)
```

Description

This is macro GFX_X11_MEDIUM_VIOLET_RED.

GFX_X11_MIDNIGHT_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MIDNIGHT_BLUE GFX_RGBConvert(0x19, 0x19, 0x70)
```

Description

This is macro GFX_X11_MIDNIGHT_BLUE.

GFX_X11_MINT_CREAM Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MINT_CREAM GFX_RGBConvert(0xF5, 0xFF, 0xFA)
```

Description

This is macro GFX_X11_MINT_CREAM.

GFX_X11_MISTY_ROSE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MISTY_ROSE GFX_RGBConvert(0xFF, 0xE4, 0xE1)
```

Description

This is macro GFX_X11_MISTY_ROSE.

GFX_X11_MOCCASIN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_MOCCASIN GFX_RGBConvert(0xFF, 0xE4, 0xB5)
```

Description

This is macro GFX_X11_MOCCASIN.

GFX_X11_NAVAJO_WHITE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_NAVAJO_WHITE GFX_RGBConvert(0xFF, 0xDE, 0xAD)
```

Description

This is macro GFX_X11_NAVAJO_WHITE.

GFX_X11_NAVY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_NAVY GFX_RGBConvert(0x00, 0x00, 0x80)
```

Description

This is macro GFX_X11_NAVY.

GFX_X11_OLD_LACE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_OLD_LACE GFX_RGBConvert(0xFD, 0xF5, 0xE6)
```

Description

This is macro GFX_X11_OLD_LACE.

GFX_X11_OLIVE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_OLIVE GFX_RGBConvert(0x80, 0x80, 0x00)
```

Description

This is macro GFX_X11_OLIVE.

GFX_X11_OLIVE_DRAB Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_OLIVE_DRAB GFX_RGBConvert(0x6B, 0x8E, 0x23)
```

Description

This is macro GFX_X11_OLIVE_DRAB.

GFX_X11_ORANGE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ORANGE GFX_RGBConvert(0xFF, 0xA5, 0x00)
```

Description

This is macro GFX_X11_ORANGE.

GFX_X11_ORANGE_RED Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ORANGE_RED GFX_RGBConvert(0xFF, 0x45, 0x00)
```

Description

X11: Orange Colors

GFX_X11_ORCHID Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ORCHID GFX_RGBConvert(0xDA, 0x70, 0xD6)
```

Description

This is macro GFX_X11_ORCHID.

GFX_X11_PALE_GOLDENROD Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PALE_GOLDENROD GFX_RGBConvert(0xEE, 0xE8, 0xAA)
```

Description

This is macro GFX_X11_PALE_GOLDENROD.

GFX_X11_PALE_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PALE_GREEN GFX_RGBConvert(0x98, 0xFB, 0x98)
```

Description

This is macro GFX_X11_PALE_GREEN.

GFX_X11_PALE_TURQUOISE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PALE_TURQUOISE GFX_RGBConvert(0xAF, 0xEE, 0xEE)
```

Description

This is macro GFX_X11_PALE_TURQUOISE.

GFX_X11_PALE_VIOLET_RED Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PALE_VIOLET_RED GFX_RGBConvert(0xDB, 0x70, 0x93)
```

Description

This is macro GFX_X11_PALE_VIOLET_RED.

GFX_X11_PAPAYA_WHIP Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PAPAYA_WHIP GFX_RGBConvert(0xFF, 0xEF, 0xD5)
```

Description

This is macro GFX_X11_PAPAYA_WHIP.

GFX_X11_PEACH_PUFF Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PEACH_PUFF GFX_RGBConvert(0xFF, 0xDA, 0xB9)
```

Description

This is macro GFX_X11_PEACH_PUFF.

GFX_X11_PERU Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PERU GFX_RGBConvert(0xCD, 0x85, 0x3F)
```

Description

This is macro GFX_X11_PERU.

GFX_X11_PINK Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PINK GFX_RGBConvert(0xFF, 0xC0, 0xCB)
```

Description

X11: Pink Colors

GFX_X11_PLUM Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PLUM GFX_RGBConvert(0xDD, 0xA0, 0xDD)
```

Description

This is macro GFX_X11_PLUM.

GFX_X11_POWDER_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_POWDER_BLUE GFX_RGBConvert(0xB0, 0xE0, 0xE6)
```

Description

This is macro GFX_X11_POWDER_BLUE.

GFX_X11_PURPLE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_PURPLE GFX_RGBConvert(0x80, 0x00, 0x80)
```

Description

This is macro GFX_X11_PURPLE.

GFX_X11_RED Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_RED GFX_RGBConvert(0xFF, 0x00, 0x00)
```

Description

This is macro GFX_X11_RED.

GFX_X11_ROSY_BROWN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ROSY_BROWN GFX_RGBConvert(0xBc, 0x8F, 0x8F)
```

Description

This is macro GFX_X11_ROSY_BROWN.

GFX_X11_ROYAL_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_ROYAL_BLUE GFX_RGBConvert(0x41, 0x69, 0xE1)
```

Description

This is macro GFX_X11_ROYAL_BLUE.

GFX_X11_SADDLE_BROWN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SADDLE_BROWN GFX_RGBConvert(0x8B, 0x45, 0x13)
```

Description

This is macro GFX_X11_SADDLE_BROWN.

GFX_X11_SALMON Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SALMON GFX_RGBConvert(0xFA, 0x80, 0x72)
```

Description

This is macro GFX_X11_SALMON.

GFX_X11_SANDY_BROWN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SANDY_BROWN GFX_RGBConvert(0xF4, 0xA4, 0x60)
```

Description

This is macro GFX_X11_SANDY_BROWN.

GFX_X11_SEA_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SEA_GREEN GFX_RGBConvert(0x2E, 0x8B, 0x57)
```

Description

This is macro GFX_X11_SEA_GREEN.

GFX_X11_SEASHELL Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SEASHELL GFX_RGBConvert(0xFF, 0xF5, 0xEE)
```

Description

This is macro GFX_X11_SEASHELL.

GFX_X11_SIENNA Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SIENNA GFX_RGBConvert(0xA0, 0x52, 0x2D)
```

Description

This is macro GFX_X11_SIENNA.

GFX_X11_SILVER Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SILVER GFX_RGBConvert(0xC0, 0xC0, 0xC0)
```

Description

This is macro GFX_X11_SILVER.

GFX_X11_SKY_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SKY_BLUE GFX_RGBConvert(0x87, 0xCE, 0xEB)
```

Description

This is macro GFX_X11_SKY_BLUE.

GFX_X11_SLATE_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SLATE_BLUE GFX_RGBConvert(0x6A, 0x5A, 0xCD)
```

Description

This is macro GFX_X11_SLATE_BLUE.

GFX_X11_SLATE_GREY Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SLATE_GREY GFX_RGBConvert(0x70, 0x80, 0x90)
```

Description

This is macro GFX_X11_SLATE_GREY.

GFX_X11_SNOW Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SNOW GFX_RGBConvert(0xFF, 0xFA, 0xFA)
```

Description

This is macro GFX_X11_SNOW.

GFX_X11_SPRING_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_SPRING_GREEN GFX_RGBConvert(0x00, 0xFF, 0x7F)
```

Description

This is macro GFX_X11_SPRING_GREEN.

GFX_X11_STEEL_BLUE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_STEEL_BLUE GFX_RGBConvert(0x46, 0x82, 0xB4)
```

Description

This is macro GFX_X11_STEEL_BLUE.

GFX_X11_TAN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_TAN GFX_RGBConvert(0xD2, 0xB4, 0x8C)
```

Description

This is macro GFX_X11_TAN.

GFX_X11_TEAL Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_TEAL GFX_RGBConvert(0x00, 0x80, 0x80)
```

Description

This is macro GFX_X11_TEAL.

GFX_X11_THISTLE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_THISTLE GFX_RGBConvert(0xD8, 0xBF, 0xD8)
```

Description

This is macro GFX_X11_THISTLE.

GFX_X11_TOMATO Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_TOMATO GFX_RGBConvert(0xFF, 0x63, 0x47)
```

Description

This is macro GFX_X11_TOMATO.

GFX_X11_TURQUOISE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_TURQUOISE GFX_RGBConvert(0x40, 0xE0, 0xD0)
```

Description

This is macro GFX_X11_TURQUOISE.

GFX_X11_VIOLET Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_VIOLET GFX_RGBConvert(0xEE, 0x82, 0xEE)
```

Description

This is macro GFX_X11_VIOLET.

GFX_X11_WHEAT Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_WHEAT GFX_RGBConvert(0xF5, 0xDE, 0xB3)
```

Description

This is macro GFX_X11_WHEAT.

GFX_X11_WHITE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_WHITE GFX_RGBConvert(0xFF, 0xFF, 0xFF)
```

Description

X11: WHITE/GRAY/BLACK COLORS

GFX_X11_WHITE_SMOKE Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_WHITE_SMOKE GFX_RGBConvert(0xF5, 0xF5, 0xF5)
```

Description

This is macro GFX_X11_WHITE_SMOKE.

GFX_X11_YELLOW Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_YELLOW GFX_RGBConvert(0xFF, 0xFF, 0x00)
```

Description

X11: Yellow Colors

GFX_X11_YELLOW_GREEN Macro

File

gfx_colors_x11.h

C

```
#define GFX_X11_YELLOW_GREEN GFX_RGBConvert(0x9A, 0xCD, 0x32)
```

Description

This is macro GFX_X11_YELLOW_GREEN.

GOLD Macro

File

gfx_colors.h

C

```
#define GOLD GFX_RGBConvert(255, 215, 0)
```

Description

This is macro GOLD.

GRAY000 Macro

File

gfx_colors.h

C

```
#define GRAY000 0
```

Description

This is macro GRAY000.

GRAY001 Macro

File

gfx_colors.h

C

```
#define GRAY001 1
```

Description

This is macro GRAY001.

GRAY002 Macro

File

gfx_colors.h

C

```
#define GRAY002 2
```

Description

This is macro GRAY002.

GRAY003 Macro

File

gfx_colors.h

C

```
#define GRAY003 3
```

Description

This is macro GRAY003.

GRAY004 Macro

File

gfx_colors.h

C

```
#define GRAY004 4
```

Description

This is macro GRAY004.

GRAY005 Macro

File

gfx_colors.h

C

```
#define GRAY005 5
```

Description

This is macro GRAY005.

GRAY006 Macro

File

gfx_colors.h

C

```
#define GRAY006 6
```

Description

This is macro GRAY006.

GRAY007 Macro

File

gfx_colors.h

C

```
#define GRAY007 7
```

Description

This is macro GRAY007.

GRAY008 Macro

File

gfx_colors.h

C

```
#define GRAY008 8
```

Description

This is macro GRAY008.

GRAY009 Macro

File

gfx_colors.h

C

```
#define GRAY009 9
```

Description

This is macro GRAY009.

GRAY010 Macro

File

gfx_colors.h

C

```
#define GRAY010 GFX_RGBConvert(10, 10, 10)
```

Description

This is macro GRAY010.

GRAY011 Macro

File

gfx_colors.h

C

```
#define GRAY011 11
```

Description

This is macro GRAY011.

GRAY012 Macro

File

gfx_colors.h

C

```
#define GRAY012 12
```

Description

This is macro GRAY012.

GRAY013 Macro

File

gfx_colors.h

C

```
#define GRAY013 13
```

Description

This is macro GRAY013.

GRAY014 Macro

File

gfx_colors.h

C

```
#define GRAY014 14
```

Description

This is macro GRAY014.

GRAY015 Macro

File

gfx_colors.h

C

```
#define GRAY015 15
```

Description

This is macro GRAY015.

GRAY032 Macro

File

gfx_colors.h

C

```
#define GRAY032 GFX_RGBConvert(32, 32, 32)
```

Description

This is macro GRAY032.

GRAY096 Macro

File

gfx_colors.h

C

```
#define GRAY096 GFX_RGBConvert(96, 96, 96)
```

Description

This is macro GRAY096.

GRAY128 Macro

File

gfx_colors.h

C

```
#define GRAY128 GFX_RGBConvert(128, 128, 128)
```

Description

This is macro GRAY128.

GRAY160 Macro

File

gfx_colors.h

C

```
#define GRAY160 GFX_RGBConvert(160, 160, 160)
```

Description

This is macro GRAY160.

GRAY192 Macro

File

gfx_colors.h

C

```
#define GRAY192 GFX_RGBConvert(192, 192, 192)
```

Description

This is macro GRAY192.

GRAY204 Macro

File

gfx_colors.h

C

```
#define GRAY204 GFX_RGBConvert(204, 204, 204)
```

Description

This is macro GRAY204.

GRAY224 Macro

File

gfx_colors.h

C

```
#define GRAY224 GFX_RGBConvert(224, 224, 224)
```

Description

This is macro GRAY224.

GRAY229 Macro

File

gfx_colors.h

C

```
#define GRAY229 GFX_RGBConvert(229, 229, 229)
```

Description

This is macro GRAY229.

GRAY242 Macro

File

gfx_colors.h

C

```
#define GRAY242 GFX_RGBConvert(242, 242, 242)
```

Description

This is macro GRAY242.

GREEN Macro

File

gfx_colors.h

C

```
#define GREEN GFX_RGBConvert(0, 128, 0)
```

Description

This is macro GREEN.

KHAKI Macro

File

gfx_colors_w3.h

C

```
#define KHAKI GFX_RGBConvert(240, 230, 140)
```

Description

This is macro KHAKI.

LIGHTBLUE Macro

File

gfx_colors.h

C

```
#define LIGHTBLUE GFX_RGBConvert(128, 128, 255)
```

Description

This is macro LIGHTBLUE.

LIGHTCYAN Macro

File

gfx_colors.h

C

```
#define LIGHTCYAN GFX_RGBConvert(128, 255, 255)
```

Description

This is macro LIGHTCYAN.

LIGHTGRAY Macro

File

gfx_colors.h

C

```
#define LIGHTGRAY GFX_RGBConvert(128, 128, 128)
```

Description

This is macro LIGHTGRAY.

LIGHTGREEN Macro

File

gfx_colors.h

C

```
#define LIGHTGREEN GFX_RGBConvert(128, 255, 128)
```

Description

This is macro LIGHTGREEN.

LIGHTMAGENTA Macro

File

gfx_colors.h

C

```
#define LIGHTMAGENTA GFX_RGBConvert(255, 128, 255)
```

Description

This is macro LIGHTMAGENTA.

LIGHTORANGE Macro

File

gfx_colors.h

C

```
#define LIGHTORANGE GFX_RGBConvert(255, 200, 0)
```

Description

This is macro LIGHTORANGE.

LIGHTRED Macro

File

gfx_colors.h

C

```
#define LIGHTRED GFX_RGBConvert(255, 128, 128)
```

Description

This is macro LIGHTRED.

LIGHTYELLOW Macro

File

gfx_colors.h

C

```
#define LIGHTYELLOW GFX_RGBConvert(255, 255, 150)
```

Description

This is macro LIGHTYELLOW.

MAGENTA Macro

File

gfx_colors.h

C

```
#define MAGENTA GFX_RGBConvert(128, 0, 128)
```

Description

This is macro MAGENTA.

ORANGE Macro

File

gfx_colors.h

C

```
#define ORANGE GFX_RGBConvert(255, 187, 76)
```

Description

This is macro ORANGE.

PERU Macro

File

gfx_colors.h

C

```
#define PERU GFX_RGBConvert(205, 133, 63)
```

Description

This is macro PERU.

RED Macro

File

gfx_colors.h

C

```
#define RED GFX_RGBConvert(128, 0, 0)
```

Description

This is macro RED.

SADDLEBROWN Macro

File

gfx_colors.h

C

```
#define SADDLEBROWN GFX_RGBConvert(139, 69, 19)
```

Description

This is macro SADDLEBROWN.

SIENNA Macro

File

gfx_colors.h

C

```
#define SIENNA GFX_RGBConvert(160, 82, 45)
```

Description

This is macro SIENNA.

TAN Macro

File

gfx_colors.h

C

```
#define TAN GFX_RGBConvert(210, 180, 140)
```

Description

This is macro TAN.

WHEAT Macro

File

gfx_colors.h

C

```
#define WHEAT GFX_RGBConvert(245, 245, 220)
```

Description

This is macro WHEAT.

WHITE Macro

File

gfx_colors.h

C

```
#define WHITE GFX_RGBConvert(255, 255, 255)
```

Description

This is macro WHITE.

YELLOW Macro

File

gfx_colors.h

C

```
#define YELLOW GFX_RGBConvert(255, 255, 128)
```

Description

This is macro YELLOW.

Graphics Object Layer

This section describes the API and usage of the Graphics Library Object Layer.

Graphics Object Layer API

GOL Objects

Graphics Library Object Layer object API.

Button Object

Button is an object that emulates a press and release effect when operated upon.

Functions

	Name	Description
	GFX_GOL_ButtonActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ButtonActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ButtonCreate	This function creates a GFX_GOL_BUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ButtonDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ButtonTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_ButtonTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_ButtonTextSet	This function sets the address of the current text string used by the object.

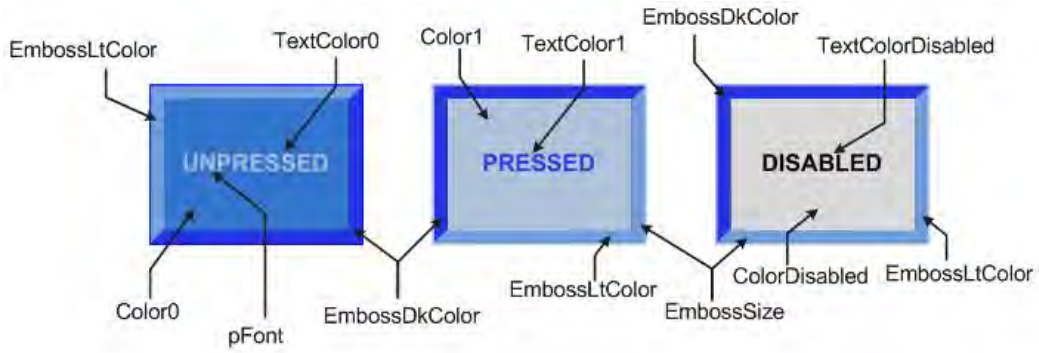
Macros

	Name	Description
	GFX_GOL_ButtonPressStateImageGet	This function gets the image used when in the pressed state.
	GFX_GOL_ButtonPressStateImageSet	This function sets the image to be used when in the pressed state.
	GFX_GOL_ButtonReleaseStateImageGet	This function gets the image used when in the released state.
	GFX_GOL_ButtonReleaseStateImageSet	This function sets the image to be used when in the released state.
	GFX_GOL_ButtonTextGet	This function returns the address of the current text string used by the object.

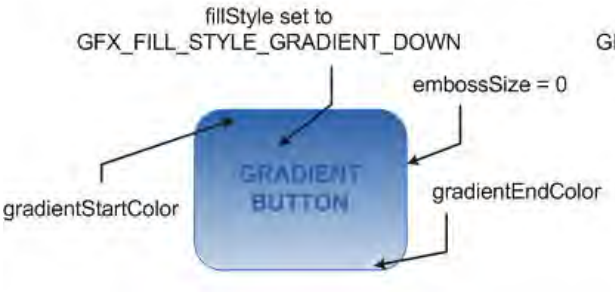
Description

Button supports Keyboard and Touchscreen inputs, replying to their events with the predefined actions (see GFX_GOL_ButtonActionGet() and GFX_GOL_ButtonActionSet() for details).

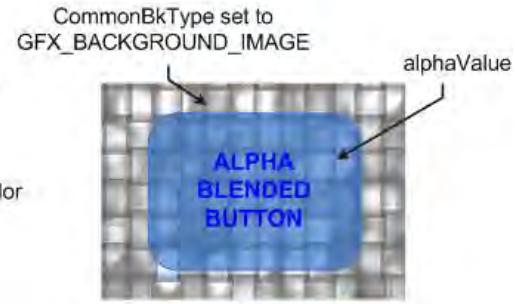
The button object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the button object.



Button Style Scheme



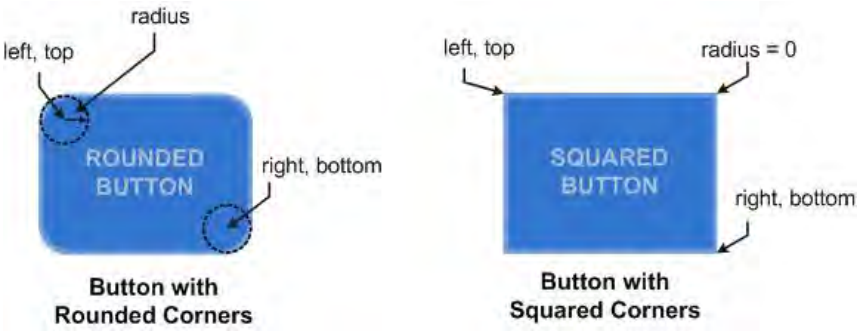
Gradient Fills in Buttons



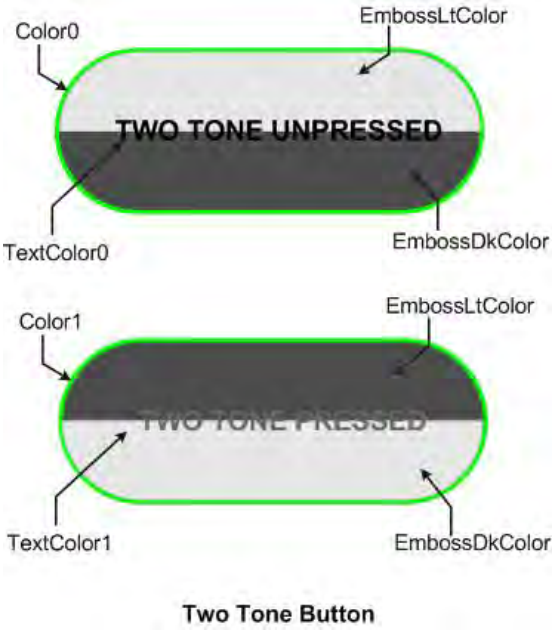
Alpha Blending in Buttons

Aside from the basic color styles, the object can also be drawn with gradient fills and alpha blended fills.

Buttons can also be drawn with rectangular or rounded edges.



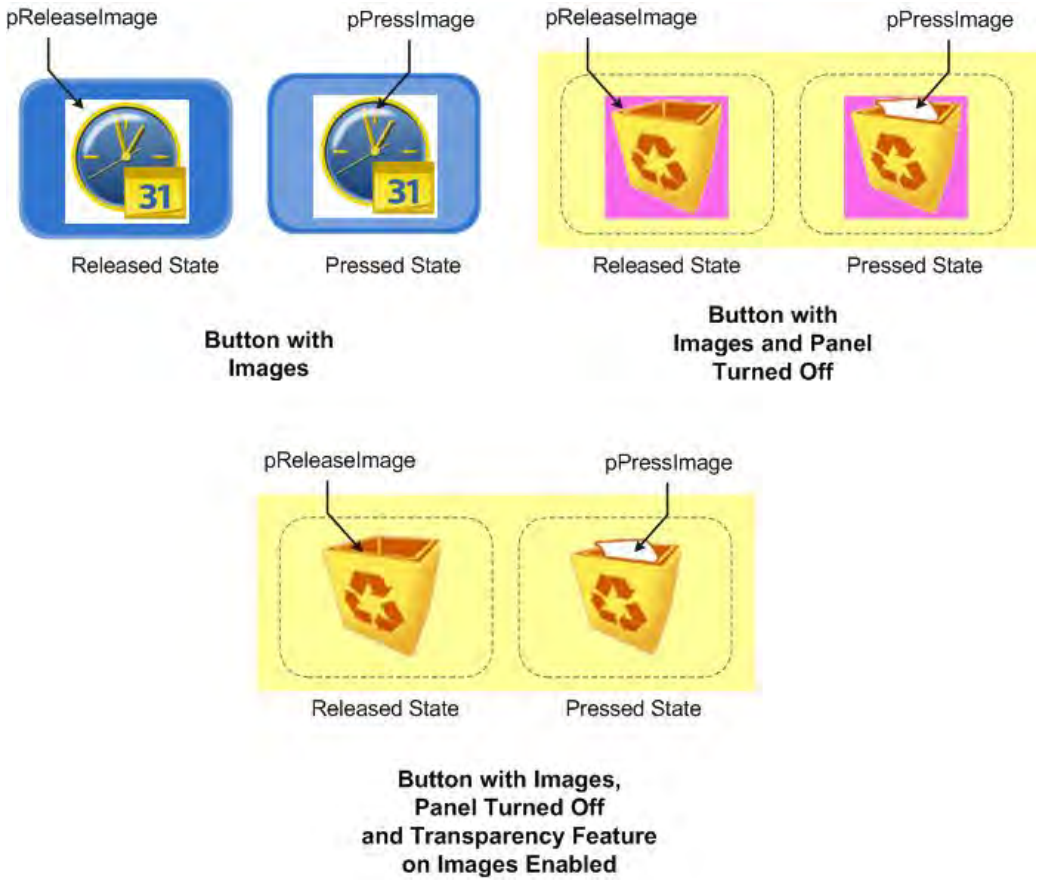
Another variation in the rendering of button object is the two-tones mode.



Buttons also allows alignment on the text used. The text alignment behavior of the object is the same as GFX_TextStringBoxDraw().

Instead of using text to name buttons, images or icons can also be used. Since there are two main states of the button (pressed and released state), the object allows usage of two different images. One is assigned to the pressed state and the other assigned to the released state.

When only one image is used, both presses and released pointers should be assigned to the same image. If one state is set to NULL, then that state will not render any image.



Another feature of the button object is to turn off the panel rendering of the object. This is useful when icons are used to define the buttons. Also to complete the feature set, the transparent color feature of images can also be enabled. This allows applications to use images with the flat background color ignored when rendering.

Note: Alpha blending of gradient fills is not supported by the object.

GFX_GOL_ButtonPressStateImageGet Macro

This function gets the image used when in the pressed state.

File

gfx_gol_button.h

C

```
#define GFX_GOL_ButtonPressStateImageGet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pPressImage)
```

Returns

Pointer to the image resource.

Description

GFX GOL button press state image get.

This function gets the image used when in the pressed state.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_RESOURCE_HDR *GFX_GOL_ButtonPressStateImageGet(
GFX_GOL_BUTTON *pObject)

GFX_GOL_ButtonPressStateImageSet Macro

This function sets the image to be used when in the pressed state.

File

gfx_gol_button.h

C

```
#define GFX_GOL_ButtonPressStateImageSet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pPressImage = pImage)
```

Returns

None.

Description

GFX GOL button press state image set.

This function sets the image to be used when in the pressed state.

Preconditions

Object must exist in memory.

Example

```
// assume ImageIcon is a valid GFX_RESOURCE_HDR

GFX_RESOURCE_HDR *pMyIcon = &ImageIcon;
GFX_GOL_BUTTON *pButton;

GFX_GOL_ButtonPressStateImageSet(pButton , pMyIcon);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pImage	pointer to the image resource.

Function

void GFX_GOL_ButtonPressStateImageSet(
GFX_GOL_BUTTON *pObject,
GFX_RESOURCE_HDR *pImage)

GFX_GOL_ButtonReleaseStateImageGet Macro

This function gets the image used when in the released state.

File

gfx_gol_button.h

C

```
#define GFX_GOL_ButtonReleaseStateImageGet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pReleaseImage)
```

Returns

Pointer to the image resource.

Description

GFX GOL button release state image get.

This function gets the image used when in the released state.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_RESOURCE_HDR *GFX_GOL_ButtonReleaseStateImageGet(
GFX_GOL_BUTTON *pObject)

GFX_GOL_ButtonReleaseStateImageSet Macro

This function sets the image to be used when in the released state.

File

gfx_gol_button.h

C

```
#define GFX_GOL_ButtonReleaseStateImageSet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pReleaseImage = pImage)
```

Returns

None.

Description

GFX GOL button release state image set.

This function sets the image to be used when in the released state.

Preconditions

Object must exist in memory.

Example

```
// assume ImageIcon is a valid GFX_RESOURCE_HDR

GFX_RESOURCE_HDR *pMyIcon = &ImageIcon;
GFX_GOL_BUTTON *pButton;

GFX_GOL_ButtonReleaseStateImageSet(pButton , pMyIcon);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pImage	pointer to the image resource.

Function

void GFX_GOL_ButtonReleaseStateImageSet(
GFX_GOL_BUTTON *pObject,
GFX_RESOURCE_HDR *pImage)

GFX_GOL_ButtonTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_button.h

C

```
#define GFX_GOL_ButtonTextGet(pObject) (((GFX_GOL_BUTTON *)pObject)->pText)
```

Returns

Pointer to text string.

Description

GFX GOL button text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR *pChar;
GFX_GOL_BUTTON GFX_GOL_BUTTON[2];

pChar = GFX_GOL_ButtonTextGet(GFX_GOL_BUTTON[0]);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_ButtonTextGet(
GFX_GOL_BUTTON *pObject)

GFX_GOL_ButtonActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_button.h

C

```
void GFX_GOL_ButtonActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL button action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_BUTTON_ACTION_PRESSED	Touch Screen,	Set GFX_GOL_BUTTON_PRESSED_STATE	Button will be redrawn in the pressed state.
	Keyboard		
GFX_GOL_BUTTON_ACTION_RELEASED	Touch Screen,	Clear GFX_GOL_BUTTON_PRESSED_STATE	Button will be redrawn in the released state.
	Keyboard		
GFX_GOL_BUTTON_ACTION_CANCELPRESS	Touch Screen,	Clear GFX_GOL_BUTTON_PRESSED_STATE	Button will be redrawn in the released state.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_ButtonActionSet(
GFX_GOL_TRANSLATED_ACTION translatedMsg,
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_ButtonActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_button.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_ButtonActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_BUTTON_ACTION_PRESSED - Object is pressed
- GFX_GOL_BUTTON_ACTION_RELEASED - Object is released
- GFX_GOL_BUTTON_ACTION_CANCELPRESS - Object will be released, user cancels press action on the GFX_GOL_BUTTON
- GFX_GOL_OBJECT_ACTION_INVALID - Object is not affected

Description

GFX GOL button action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_BUTTON_ACTION_PRESSED	Touch Screen	EVENT_PRESS, EVENT_MOVE	If events occurs and the x,y position falls in the face of the GFX_GOL_BUTTON while the GFX_GOL_BUTTON is not pressed.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_CR_PRESSED or SCAN_SPACE_PRESSED while the GFX_GOL_BUTTON is not pressed.
GFX_GOL_BUTTON_ACTION_STILLPRESSED	Touch Screen	EVENT_STILLPRESS	If event occurs and the x,y position does not change from the previous press position in the face of the GFX_GOL_BUTTON.
GFX_GOL_BUTTON_ACTION_RELEASED	Touch Screen	EVENT_RELEASE	If the event occurs and the x,y position falls in the face of the GFX_GOL_BUTTON while the GFX_GOL_BUTTON is pressed.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_CR_RELEASED or SCAN_SPACE_RELEASED while the GFX_GOL_BUTTON is pressed.
GFX_GOL_BUTTON_ACTION_CANCELPRESS	Touch Screen	EVENT_MOVE	If the event occurs outside the face of the GFX_GOL_BUTTON and the GFX_GOL_BUTTON is currently pressed.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_ButtonActionGet(

void *pObject,

GFX_GOL_MESSAGE *pMessage);

GFX_GOL_ButtonCreate Function

This function creates a GFX_GOL_BUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_button.h

C

```
GFX_GOL_BUTTON * GFX_GOL_ButtonCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t top,
uint16_t right, uint16_t bottom, uint16_t radius, uint16_t state, GFX_RESOURCE_HDR * pPressImage,
GFX_RESOURCE_HDR * pReleaseImage, GFX_XCHAR * pText, GFX_ALIGNMENT alignment, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL button create

This function creates a GFX_GOL_BUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The object allows setting two images. One for the pressed state and the other for the release state. If no image is to be used for the object set both pointers to NULL.

If only one image is used for both pressed and released state, set both pPressImage and pReleaseImage to the same image.

The behavior of GFX_GOL_ButtonCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pPressImage and pReleaseImage is not pointing to a GFX_RESOURCE_HDR.
- pText is an unterminated string

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_BUTTON *buttons[3];
GFX_GOL_BUTTON_STATE state;

// assume pScheme is initialized to a scheme in memory.
state = GFX_GOL_BUTTON_DRAW_STATE;

buttons[0] = GFX_GOL_ButtonCreate(
        gfxIndex,
        1, 20, 64, 50, 118, 0,
        state, NULL, NULL, "ON",
        GFX_ALIGN_HCENTER | GFX_ALIGN_VCENTER,
        pScheme);
// check if GFX_GOL_BUTTON 0 is created
```

```
    if (buttons[0] == NULL)
       return 0;

    buttons[1] = GFX_GOL_ButtonCreate(
            gfxIndex,
            2, 52, 64, 82, 118, 0,
            state, NULL, NULL, "OFF",
            GFX_ALIGN_LEFT | GFX_ALIGN_VCENTER,
            pScheme);
    // check if GFX_GOL_BUTTON 1 is created
    if (buttons[1] == NULL)
       return 0;

    buttons[2] = GFX_GOL_ButtonCreate(
            gfxIndex,
            3, 84, 64, 114, 118, 0,
            state, NULL, NULL, "HI",
            GFX_ALIGN_RIGHT | GFX_ALIGN_VCENTER,
            pScheme);
    // check if GFX_GOL_BUTTON 2 is created
    if (buttons[2] == NULL)
       return 0;

    return 1;
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
radius	Radius of the rounded edge. When using gradient buttons and radius != 0, emboss size <= radius. If this is not met, the the GFX_GOL_BUTTON face will not have gradient effect.
state	Sets the initial state of the object.
pPressImage	Pointer to the image used on the face of the object when it is in the pressed state.
pReleaseImage	Pointer to the image used on the face of the object when it is in the pressed state.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_BUTTON  *GFX_GOL_ButtonCreate(
SYS_MODULE_INDEX gfxIndex,
uint16_t         ID,
uint16_t         left,
uint16_t         top,
uint16_t         right,
uint16_t         bottom,
uint16_t         radius,
uint16_t         state,
                 GFX_RESOURCE_HDR    *pPressImage,
                 GFX_RESOURCE_HDR    *pReleaseImage,
                 GFX_XCHAR           *pText,
                 GFX_ALIGNMENT       alignment,
                 GFX_GOL_OBJ_SCHEME  *pScheme)
```

GFX_GOL_ButtonDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_button.h

C

```
GFX_STATUS GFX_GOL_ButtonDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL button draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the face of the GFX_GOL_BUTTON is drawn on top of the bitmap. Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_ButtonDraw(void *pObject)

GFX_GOL_ButtonTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_button.h

C

```
GFX_ALIGNMENT GFX_GOL_ButtonTextAlignmentGet(GFX_GOL_BUTTON * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL button text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_ButtonTextAlignmentGet(

GFX_GOL_BUTTON *pObject)

GFX_GOL_ButtonTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_button.h

C

```
void GFX_GOL_ButtonTextAlignmentSet(GFX_GOL_BUTTON * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

GFX GOL button text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_ButtonTextAlignmentSet(

GFX_GOL_BUTTON *pObject,

GFX_ALIGNMENT align)

GFX_GOL_ButtonTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_button.h

C

```
void GFX_GOL_ButtonTextSet(GFX_GOL_BUTTON * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

GFX GOL button text set.

This function sets the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR Label0[] = â€œONâ€?;
GFX_XCHAR Label1[] = â€œOFFâ€?;
GFX_GOL_BUTTON GFX_GOL_BUTTON[2];

GFX_GOL_ButtonTextSet(GFX_GOL_BUTTON[0], Label0);
GFX_GOL_ButtonTextSet(GFX_GOL_BUTTON[1], Label1);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be used.

Function

```
GFX_XCHAR *GFX_GOL_ButtonTextSet(
                GFX_GOL_BUTTON *pObject,
                GFX_XCHAR *pText)
```

Check Box Object

Check Box is an object that simulates a check box on paper. Usually it is used as an option setting where the checked or filled state means the option is enabled and the unfilled or unchecked state means the option is disabled.

Functions

	Name	Description
	GFX_GOL_CheckBoxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_CheckBoxActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_CheckBoxCreate	This function creates a GFX_GOL_CHECKBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_CheckBoxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_CheckBoxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_CheckBoxTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_CheckBoxTextSet	This function sets the address of the current text string used by the object.

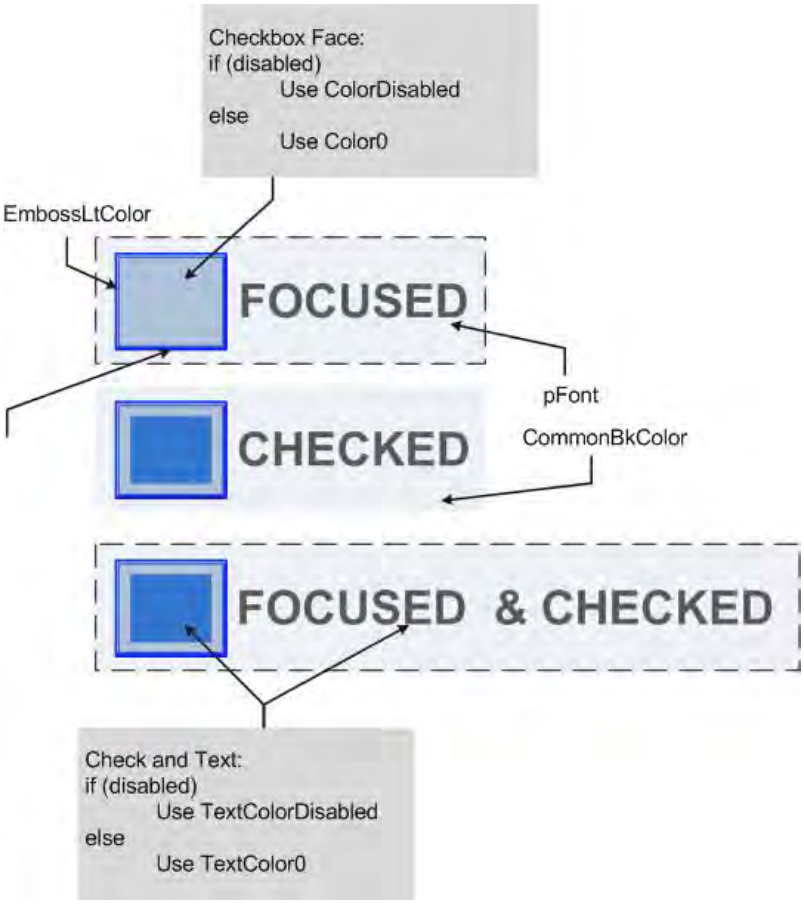
Macros

	Name	Description
	GFX_GOL_CheckBoxTextGet	This function returns the address of the current text string used by the object.

Description

Check Box supports Keyboard and Touchscreen inputs, replying to their events with the predefined actions (see GFX_GOL_CheckBoxActionGet() and GFX_GOL_CheckBoxActionSet() for details).

The Check Box object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



When creating the object, the alignment of the text of the object can be formatted with the same options that GFX_TextStringBoxDraw() allows.

GFX_GOL_CheckBoxTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_check_box.h

C

```
#define GFX_GOL_CheckBoxTextGet(pObject) (((GFX_GOL_CHECKBOX *)pObject)->pText)
```

Returns

Pointer to text string.

Description

GFX GOL check box text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
// assume CHECK_BOX_OBJECT is a radio button that exists
GFX_XCHAR *pChar;
GFX_GOL_CHECKBOX *pObject = &CHECK_BOX_OBJECT;

pChar = GFX_GOL_CheckBoxTextGet(pObject);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_CheckBoxTextGet(
GFX_GOL_CHECKBOX *pObject)

GFX_GOL_CheckBoxActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_check_box.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_CheckBoxActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_CHECKBOX_ACTION_CHECKED - Check box is checked
- GFX_GOL_CHECKBOX_ACTION_UNCHECKED - Check box is unchecked
- GFX_GOL_OBJECT_ACTION_INVALID - Object is not affected

Description

GFX GOL checkbox action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_CHECKBOX_ACTION_CHECKED	Touch Screen	EVENT_RELEASE	If events occurs and the x,y position falls in the area of the check box while the check box is unchecked.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_CR_PRESSED or SCAN_SPACE_PRESSED while the check box is unchecked.
GFX_GOL_CHECKBOX_ACTION_UNCHECKED	Touch Screen	EVENT_RELEASE	If events occurs and the x,y position falls in the area of the check box while the check box is checked.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_CR_PRESSED or SCAN_SPACE_PRESSED while the check box is checked.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_CheckBoxActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage);

GFX_GOL_CheckBoxActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_check_box.h

C

```
void GFX_GOL_CheckBoxActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL checkbox action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_CHECKBOX_ACTION_CHECKED	Touch Screen,	Set GFX_GOL_CHECKBOX_CHECKED_STATE	Check Box will be redrawn in checked state.
	Keyboard		
GFX_GOL_CHECKBOX_ACTION_UNCHECKED	Touch Screen,	Clear GFX_GOL_CHECKBOX_CHECKED_STATE	Check Box will be redrawn in un-checked state.
	Keyboard		

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_CheckBoxActionSet(

GFX_GOL_TRANSLATED_ACTION translatedMsg,

void *pObject,

GFX_GOL_MESSAGE *pMessage)

GFX_GOL_CheckBoxCreate Function

This function creates a GFX_GOL_CHECKBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_check_box.h

C

```
GFX_GOL_CHECKBOX * GFX_GOL_CheckBoxCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL check box create.

This function creates a GFX_GOL_CHECKBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_CheckBoxCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME  *pScheme;
GFX_GOL_CHECKBOX    *pCb[2];

pCb[0] = GFX_GOL_CheckBoxCreate(
                    gfxIndex,
                    ID_CHECKBOX1,               // ID
                    20,135,150,175,             // dimension
                    GFX_GOL_CHECKBOX_DRAW_STATE, // Draw the object
                    "Scale",                    // text
                    GFX_ALIGN_CENTER,           // text alignment
                    pScheme);                   // use this scheme

pCb[1] = GFX_GOL_CheckBoxCreate(
                    gfxIndex,
                    ID_CHECKBOX2,               // ID
                    170,135,300,175,            // dimension
                    GFX_GOL_CHECKBOX_DRAW_STATE, // Draw the object
                    "Animate",                  // text
                    GFX_ALIGN_LEFT | GFX_ALIGN_VCENTER, // text alignment
                    pScheme);                   // use this scheme
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_CHECKBOX  *GFX_GOL_CheckBoxCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t        ID,
uint16_t        left,
uint16_t        top,
uint16_t        right,
uint16_t        bottom,
uint16_t        state,
                GFX_XCHAR       *pText,
```

GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME *pScheme)

GFX_GOL_CheckBoxDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_check_box.h

C

```
GFX_STATUS GFX_GOL_CheckBoxDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL checkbox draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the GFX_GOL_CHECKBOX is drawn with the text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_CheckBoxDraw(void *pObject)

GFX_GOL_CheckBoxTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_check_box.h

C

```
GFX_ALIGNMENT GFX_GOL_CheckBoxTextAlignmentGet(GFX_GOL_CHECKBOX * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL check box text alignement get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_CheckBoxTextAlignmentGet(
GFX_GOL_CHECKBOX *pObject)

GFX_GOL_CheckBoxTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_check_box.h

C

```
void GFX_GOL_CheckBoxTextAlignmentSet(GFX_GOL_CHECKBOX * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

GFX GOL checkboc text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_CheckBoxTextAlignmentSet(
GFX_GOL_CHECKBOX *pObject,
GFX_ALIGNMENT align)

GFX_GOL_CheckBoxTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_check_box.h

C

```
void GFX_GOL_CheckBoxTextSet(GFX_GOL_CHECKBOX * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

GFX OL check box text set.

This function sets the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR Label0[] = â€œEnabledâ€?;
GFX_XCHAR Label1[] = â€œDisabledâ€?;
GFX_GOL_CHECKBOX GFX_GOL_CHECKBOX[2];

GFX_GOL_CheckBoxTextSet(GFX_GOL_CHECKBOX[0], Label0);
GFX_GOL_CheckBoxTextSet(GFX_GOL_CHECKBOX[1], Label1);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be used.

Function

GFX_XCHAR *GFX_GOL_CheckBoxTextSet(
GFX_GOL_CHECKBOX *pObject,
GFX_XCHAR *pText)

Custom Control Object

Functions

	Name	Description
	GFX_GOL_CustomControlActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_CustomControlActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_CustomControlCreate	This function creates a GFX_GOL_CUSTOMCONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_CustomControlDraw	This function renders the object on the screen based on the current state of the object.

Macros

	Name	Description
	GFX_GOL_CustomControlGetPos	This function returns the position of the control.
	GFX_GOL_CustomControlSetPos	This function sets the position of the control.

Description

GFX_GOL_CustomControlGetPos Macro

This function returns the position of the control.

File

gfx_gol_custom_control.h

C

```
#define GFX_GOL_CustomControlGetPos(pCc) pCc->pos
```

Returns

The position of the object.

Description

GFX GOL custom control get position

This function returns the position of the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_CustomControlGetPos(
GFX_GOL_CUSTOMCONTROL *pObject)

GFX_GOL_CustomControlSetPos Macro

This function sets the position of the control.

File

gfx_gol_custom_control.h

C

```
#define GFX_GOL_CustomControlSetPos(pCc, position) pCc->pos = position
```

Returns

none

Description

GFX GOL custom control set position.

This function sets the position of the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_CustomControlSetPos(
GFX_GOL_CUSTOMCONTROL *pObject, uint16_t pos)

GFX_GOL_CustomControlActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_custom_control.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_CustomControlActionGet(void * pObj, GFX_GOL_MESSAGE * pMsg);
```

Returns

- GFX_GOL_CUSTOMCONTROL_ACTION_SELECTED - Object is selected.
- GFX_GOL_OBJECT_ACTION_INVALID - Object is not affected

Description

GFX GOL custom control action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_CUSTOMCONTROL_ACTION_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If events occurs and the x,y position falls in the area of the Digital Meter.

GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_CustomControlActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_CustomControlActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_custom_control.h

C

```
void GFX_GOL_CustomControlActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObj, GFX_GOL_MESSAGE * pMsg);
```

Returns

None.

Description

GFX GOL custom control action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source Set/Clear State Bit Description
GFX_GOL_CUSTOMCONTROL_ACTION_SELECTED	Touch Screen, Set GFX_GOL_RADIOBUTTON_DRAW_STATE, Depending on the current value of RB_CHECKED Check Box will be redrawn.
	Keyboard Set/Clear GFX_GOL_RADIOBUTTON_CHECKED_STATE

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_CustomControlActionSet(
GFX_GOL_TRANSLATED_ACTION translatedMsg,
void *pObject,

GFX_GOL_MESSAGE *pMessage)

GFX_GOL_CustomControlCreate Function

This function creates a GFX_GOL_CUSTOMCONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_custom_control.h

C

```
GFX_GOL_CUSTOMCONTROL * GFX_GOL_CustomControlCreate(SYS_MODULE_INDEX clientIndex, uint16_t ID, uint16_t
left, uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL custom control create.

This function creates a GFX_GOL_CUSTOMCONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_CustomControlCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME  *pScheme;
GFX_GOL_CUSTOMCONTROL    *pCc[2];

pCc[0] = GFX_GOL_CustomControlCreate(
                    GFX_INDEX_0,
                    ID_CUSTOM1, // ID
                    30,
                    0,
                    150,
                    120, // dimension
                    // will be displayed after creation
                    GFX_GOL_CUSTOMCONTROL_DRAW_STATE,
                    pScheme
 );
```

Parameters

Parameters	Description
clientIndex	client user instance
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

GFX_GOL_CUSTOMCONTROL *GFX_GOL_CustomControlCreate(

SYS_MODULE_INDEX clientIndex,

uint16_t ID,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t state,
GFX_GOL_OBJ_SCHEME * pScheme)

GFX_GOL_CustomControlDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_custom_control.h

C

```
GFX_STATUS GFX_GOL_CustomControlDraw(void * pObj);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL custom control draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_CustomControlDraw(void *pObject)

Digital Meter Object

Digital Meter is an object that can be used to display a value of a sampled variable. This object is ideal when fast refresh of the value is needed. The object refreshes only the digits that needs to change.

Functions

	Name	Description
	GFX_GOL_DigitalMeterActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_DigitalMeterCreate	This function creates a GFX_GOL_DIGITALMETER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_DigitalMeterDecrement	This function decrements the meter value by the delta value set.
	GFX_GOL_DigitalMeterDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_DigitalMeterIncrement	This function increments the meter value by the delta value set.
	GFX_GOL_DigitalMeterValueSet	This function sets the value of the meter.

Macros

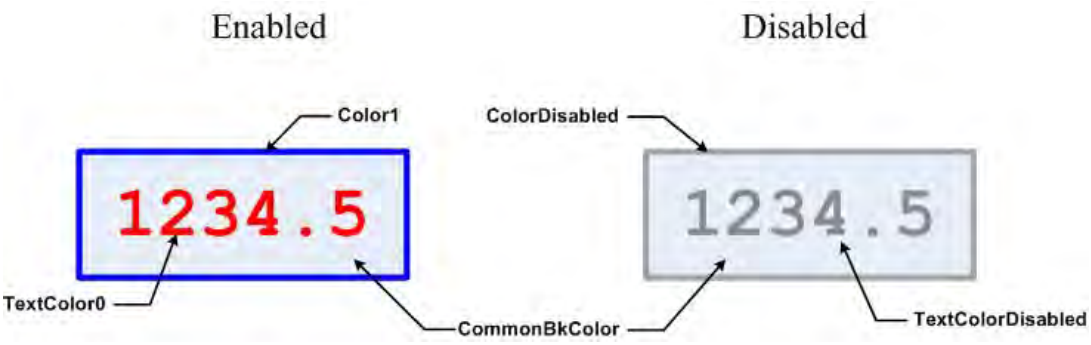
	Name	Description
	GFX_GOL_DigitalMeterTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_DigitalMeterTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_DigitalMeterValueGet	This function returns the current value of the digital meter.

Description

Digital Meter is an object that can be used to display a value of a sampled variable. This object is ideal when fast refresh of the value is needed. The object refreshes only the digits that needs to change.

Digital Meter supports only Touchscreen inputs, replying to the events with the predefined actions (see GFX_GOL_DigitalMeterActionGet() for details). There is no default action set function in this object. Application can create specific responses to the system action in the message callback function.

The DigitalMeter object is rendered using the assigned style scheme. The following figure illustrates the color assignments for the digital meter.



GFX_GOL_DigitalMeterTextAlignmentGet Macro

This function returns the text alignment of the text string used by the object.

File

gfx_gol_digital_meter.h

C

```
#define GFX_GOL_DigitalMeterTextAlignmentGet(pObject) \
                (((GFX_GOL_DIGITALMETER *)pObject)->alignment)
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL digital meter text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_DigitalMeterTextAlignmentGet(
GFX_GOL_DIGITALMETER *pObject)

GFX_GOL_DigitalMeterTextAlignmentSet Macro

This function sets the text alignment of the text string used by the object.

File

gfx_gol_digital_meter.h

C

```
#define GFX_GOL_DigitalMeterTextAlignmentSet(pObject, align) \
                (((GFX_GOL_DIGITALMETER *)pObject)->alignment = align)
```

Returns

None.

Description

GFX GOL digital meter text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_DigitalMeterTextAlignmentSet(

GFX_GOL_DIGITALMETER *pObject,

GFX_ALIGNMENT align)

GFX_GOL_DigitalMeterValueGet Macro

This function returns the current value of the digital meter.

File

gfx_gol_digital_meter.h

C

```
#define GFX_GOL_DigitalMeterValueGet(pObject) \
                                (((GFX_GOL_DIGITALMETER*)pObject)->value)
```

Returns

The current value of the object.

Description

GFX GOL digital meter value get.

This function returns the current value of the digital meter.

Preconditions

Object must exist in memory.

Example

```
#define MAXVALUE 100;

GFX_GOL_DIGITALMETER *pMeter;
uint32_t ctr = 0;

// create scroll bar here and initialize parameters
pMeter = GFX_GOL_DigitalMeterCreate(....)
GFX_GOL_ObjectStateSet(pMeter, GFX_GOL_DIGITALMETER_DRAW_STATE);

// draw the scroll bar
GFX_GOL_ObjectListDraw(gfxIndex);
```

```
// a routine that updates the position of the thumb through some
// conditions
while("some condition")
{
    GFX_GOL_DigitalMeterValueSet(pMeter, ctr);
    GFX_GOL_ObjectStateSet( pMeter,
                            GFX_GOL_DIGITALMETER_UPDATE_STATE);

    // update the screen
    GFX_GOL_ObjectListDraw(gfxIndex);

    // update ctr here
    ctr = "some source of value";
}

if (GFX_GOL_DigitalMeterValueGet(pScrollBar) > MAXVALUE)
    return 0;
else
    "do something else"
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_DigitalMeterValueGet(
GFX_GOL_DIGITALMETER *pObject)

GFX_GOL_DigitalMeterActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_digital_meter.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_DigitalMeterActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_DIGITALMETER_ACTION_SELECTED - Object is selected.
- GFX_GOL_OBJECT_ACTION_INVALID - Object is not affected

Description

GFX GOL Digital meter action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_DIGITALMETER_ACTION_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If events occurs and the x,y position falls in the area of the Digital Meter.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_DigitalMeterActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_DigitalMeterCreate Function

This function creates a GFX_GOL_DIGITALMETER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_digital_meter.h

C

```
GFX_GOL_DIGITALMETER * GFX_GOL_DigitalMeterCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left,
uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, uint32_t value, uint8_t NoOfDigits, uint8_t
DotPos, GFX_ALIGNMENT alignment, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL digital meter create.

This function creates a GFX_GOL_DIGITALMETER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_DIGITALMETER *pDm;

pScheme = GFX_GOL_SchemeCreate();
state = GFX_GOL_DIGITALMETER_DRAW_STATE |
        GFX_GOL_DIGITALMETER_FRAME_STATE;
GFX_GOL_DigitalMeterCreate(
        gfxIndex,
        ID_DIGITALMETER1,   // ID
        30,80,235,160,      // dimension
        state,              // has frame and center aligned
        789,4,1,            // to display 078.9
        GFX_ALIGN_CENTER,
        pScheme);           // use given scheme

// draw the objects
while(GFX_GOL_ObjectListDraw(gfxIndex) != GFX_STATUS_SUCCESS);
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
instance	Device instance
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.

bottom	Bottom most position of the object.
state	Sets the initial state of the object.
value	Sets the initial value to be displayed
NoOfDigits	Sets the number of digits to be displayed
DotPos	Sets the position of decimal point in the display
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme. Set to NULL if default style scheme is used.

Function

GFX_GOL_DIGITALMETER *GFX_GOL_DigitalMeterCreate(
SYS_MODULE_INDEX gfxIndex,
uint16_t ID,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t state,
uint32_t value,
uint8_t NoOfDigits,
uint8_t DotPos,
GFX_ALIGNMENT alignment
GFX_GOL_OBJ_SCHEME *pScheme)

GFX_GOL_DigitalMeterDecrement Function

This function decrements the meter value by the delta value set.

File

gfx_gol_digital_meter.h

C

```
void GFX_GOL_DigitalMeterDecrement(GFX_GOL_DIGITALMETER * pObject, uint16_t delta);
```

Returns

None.

Description

GFX GOL digital meter decrement.

This function decrements the meter value by the given delta value set. If the delta given is less than the minimum value of the meter, the value will remain to be at minimum.

Object must be redrawn after this function is called to reflect the changes to the object.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_DigitalMeterIncrement().

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_DigitalMeterDecrement(
GFX_GOL_DIGITALMETER *pObject,
uint16_t delta)

GFX_GOL_DigitalMeterDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_digital_meter.h

C

```
GFX_STATUS GFX_GOL_DigitalMeterDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL digital meter draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_DigitalMeterDraw(void *pObject)

GFX_GOL_DigitalMeterIncrement Function

This function increments the meter value by the delta value set.

File

gfx_gol_digital_meter.h

C

```
void GFX_GOL_DigitalMeterIncrement(GFX_GOL_DIGITALMETER * pObject, uint16_t delta);
```

Returns

None.

Description

GFX GOL digital meter

This function increments the scroll bar position by the given delta value set. If the delta given exceeds the maximum value of the meter, the value will remain to be at maximum.

Object must be redrawn after this function is called to reflect the changes to the object.

Preconditions

Object must exist in memory.

Example

```
void ControlSpeed(  GFX_GOL_DIGITALMETER* pObj,
                    int setSpeed,
                    int curSpeed)
{
    // set page size to 1
    GFX_GOL_DigitalMeterValueSet(pObj, 1);

    if (setSpeed < curSpeed)
```

```
        {
            while(GFX_GOL_DigitalMeterValueGet(pObj) < SetSpeed)
            {
                // increment by 1
                GFX_GOL_DigitalMeterIncrement(pObj, 1);
            }
        }
        else if (setSpeed > curSpeed)
        {
            while(GFX_GOL_DigitalMeterValueGet(pObj) > SetSpeed)
            {
                // decrement by 1
                GFX_GOL_DigitalMeterDecrement(pObj, 1);
            }
        }
    }
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_DigitalMeterIncrement(
GFX_GOL_DIGITALMETER *pObject,
uint16_t delta)

GFX_GOL_DigitalMeterValueSet Function

This function sets the value of the meter.

File

gfx_gol_digital_meter.h

C

```
void GFX_GOL_DigitalMeterValueSet(GFX_GOL_DIGITALMETER * pObject, int16_t value);
```

Returns

None.

Description

GFX GOL digital meter value set.

This function sets the value of the meter. The value used should be within the range set for the object. The new value is checked to be in the minimum and maximum value range. If the value set is less than the minimum value, the value that will be set is the minimum value. If the value set is more than the maximum value, then the value that will be set is the maximum value.

Preconditions

Object must exist in memory.

Example

```
GFX_GOL_DIGITALMETER *pMeter;
uint16_t ctr = 0;

// create slider here and initialize parameters
GFX_GOL_ObjectStateSet(pMeter, GFX_GOL_DIGITALMETER_DRAW_STATE);
GFX_GOL_ObjectListDraw(gfxIndex);

while("some condition")
{
    GFX_GOL_DigitalMeterValueSet(pMeter, ctr);
    GFX_GOL_ObjectStateSet( pMeter,
                            GFX_GOL_DIGITALMETER_UPDATE_STATE);

    // redraw the scroll bar
    GFX_GOL_ObjectListDraw(gfxIndex);

   // update ctr here
```

```
        ctr = "some source of value";
    }
```

Parameters

Parameters	Description
pObject	pointer to the object.
value	the new value of the object.

Function

void GFX_GOL_DigitalMeterValueSet(
GFX_GOL_DIGITALMETER *pObject,
int16_t value)

Edit Box Object

Edit Box is an object that emulates a cell or a text area that can be edited dynamically.

Functions

	Name	Description
	GFX_GOL_EditBoxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_EditBoxActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_EditBoxCharAdd	This function adds a character at the end of the text used by the object.
	GFX_GOL_EditBoxCharRemove	This function removes a character at the end of the text used by the object.
	GFX_GOL_EditBoxCreate	This function creates a GFX_GOL_EDITBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_EditBoxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_EditBoxTextSet	This function sets the text in the object.
	GFX_GOL_EditBoxCursorPositionDecrement	Decrement cursor position.
	GFX_GOL_EditBoxCursorPositionIncrement	Increment cursor position.

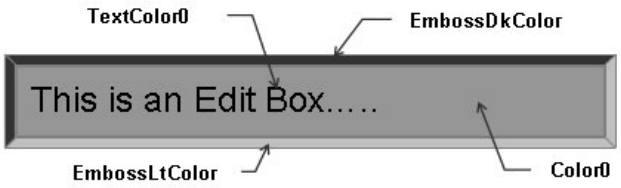
Macros

	Name	Description
	GFX_GOL_EditBoxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_EditBoxTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_EditBoxTextGet	This function returns the address of the current text string used by the object.

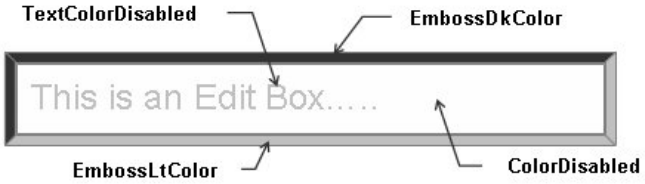
Description

Edit Box supports only keyboard inputs, replying to the events with the predefined actions (see GFX_GOL_EditBoxActionGet() and GFX_GOL_EditBoxActionSet() for details).

The Edit Box object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



Edit Box in enabled state



Edit Box in disabled state

GFX_GOL_EditBoxTextAlignmentGet Macro

This function returns the text alignment of the text string used by the object.

File

gfx_gol_edit_box.h

C

```
#define GFX_GOL_EditBoxTextAlignmentGet(pObject) \
                (((GFX_GOL_EDITBOX *)pObject)->alignment)
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL edit box text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_EditBoxTextAlignmentGet(
GFX_GOL_EDITBOX *pObject)

GFX_GOL_EditBoxTextAlignmentSet Macro

This function sets the text alignment of the text string used by the object.

File

gfx_gol_edit_box.h

C

```
#define GFX_GOL_EditBoxTextAlignmentSet(pObject, align) \
                (((GFX_GOL_EDITBOX *)pObject)->alignment = align)
```

Returns

None.

Description

GFX GOL edit box text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_EditBoxTextAlignmentSet(
GFX_GOL_EDITBOX *pObject,
GFX_ALIGNMENT align)

GFX_GOL_EditBoxTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_edit_box.h

C

```
#define GFX_GOL_EditBoxTextGet(pObject) (((GFX_GOL_EDITBOX *)pObject)->pText)
```

Returns

Pointer to text string.

Description

GFX GOL edit box text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_EditBoxTextGet(
GFX_GOL_EDITBOX *pObject)

GFX_GOL_EditBoxActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_edit_box.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_EditBoxActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_EDITBOX_ACTION_ADD_CHAR - New character should be added
- GFX_GOL_EDITBOX_ACTION_DEL_CHAR - Last character should be removed.
- GFX_GOL_EDITBOX_ACTION_TOUCHSCREEN - focus will be drawn when enabled. Caret will be drawn when enabled.
- GFX_GOL_OBJECT_ACTION_INVALID - Object is not affected

Description

GFX GOL edit box action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_EDITBOX_ACTION_ADD_CHAR	Keyboard	EVENT_CHARCODE	New character should be added.
GFX_GOL_EDITBOX_ACTION_DEL_CHAR	Keyboard	EVENT_KEYPRESS	Last character should be removed.
GFX_GOL_EDITBOX_ACTION_TOUCHSCREEN	Touch Screen	GFX_GOL_EDITBOX_FOCUSED_STATE	Focus will be drawn, caret displayed when enabled.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_EditBoxActionGet(

void *pObject,

GFX_GOL_MESSAGE *pMessage);

GFX_GOL_EditBoxActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_edit_box.h

C

```
void GFX_GOL_EditBoxActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL edit box action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the

application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_EDITBOX_ACTION_ADD_CHAR	Keyboard	Set GFX_GOL_EDITBOX_DRAW_STATE	New character is added and Edit Box will be redrawn.
GFX_GOL_EDITBOX_ACTION_DEL_CHAR	KeyBoard	Set GFX_GOL_EDITBOX_DRAW_STATE	Last character is removed and Edit Box will be redrawn.
GFX_GOL_EDITBOX_ACTION_TOUCHSCREEN	Touch Screen	Set GFX_GOL_EDITBOX_FOCUSED_STATE	When enabled, focus will be redrawn, caret will also be redrawn if enabled.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_EditBoxActionSet(
GFX_GOL_TRANSLATED_ACTION translatedMsg,
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_EditBoxCharAdd Function

This function adds a character at the end of the text used by the object.

File

gfx_gol_edit_box.h

C

```
GFX_STATUS GFX_GOL_EditBoxCharAdd(GFX_GOL_EDITBOX * pObject, GFX_XCHAR ch);
```

Returns

The status of the addition.

- GFX_STATUS_SUCCESS - when the addition was successful.
- GFX_STATUS_FAILURE - when the addition was not successful.

Description

GFX GOL edit box char add.

This function adds a character at the end of the text used by the object. When the object contains the maximum number of characters any addition call to this function will not affect the text in the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
ch	The character to be added.

Function

GFX_STATUS GFX_GOL_EditBoxCharAdd(
GFX_GOL_EDITBOX *pObject,
GFX_XCHAR ch)

GFX_GOL_EditBoxCharRemove Function

This function removes a character at the end of the text used by the object.

File

gfx_gol_edit_box.h

C

```
GFX_STATUS GFX_GOL_EditBoxCharRemove(GFX_GOL_EDITBOX * pObject);
```

Returns

The status of the addition.

- GFX_STATUS_SUCCESS - when the removal was successful.
- GFX_STATUS_FAILURE - when the removal has no effect and the buffer is empty.

Description

GFX GOL edit box char remove.

This function removes a character at the end of the text used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_EditBoxCharRemove(
GFX_GOL_EDITBOX *pObject)

GFX_GOL_EditBoxCreate Function

This function creates a GFX_GOL_EDITBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_edit_box.h

C

```
GFX_GOL_EDITBOX * GFX_GOL_EditBoxCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, uint16_t charMax, GFX_ALIGNMENT
alignment, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL edit box create.

This function creates a GFX_GOL_EDITBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_ListBoxCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom

- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
// assume pScheme to be initialized with the style scheme

#define MAX_CHAR_COUNT 15

GFX_GOL_OBJ_SCHEME      *pScheme;
GFX_GOL_EDITBOX         *pEb;
GFX_GOL_EDITBOX_STATE   state;

GFX_XCHAR               TextBuffer[MAX_CHAR_COUNT + 1];

// assume pScheme is initialized to a scheme in memory.
state = GFX_GOL_EDITBOX_DRAW_STATE;

pEb = GFX_GOL_EditBoxCreate(
        1, 20, 64, 50, 118,
        state, TextBuffer, MAX_CHAR_COUNT
        GFX_ALIGN_CENTER,
        pScheme);

// check if object is not created
if (pEb == NULL)
   return 0;
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pText	Pointer to the text of the object.
charMax	Defines the maximum number of characters in the edit box.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_EDITBOX  *GFX_GOL_EditBoxCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t        ID,
uint16_t        left,
uint16_t        top,
uint16_t        right,
uint16_t        bottom,
uint16_t        state,
                    GFX_XCHAR       *pText,
uint16_t        charMax,
                    GFX_ALIGNMENT       alignment,
                    GFX_GOL_OBJ_SCHEME  *pScheme)
```

GFX_GOL_EditBoxDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_edit_box.h

C

```
GFX_STATUS GFX_GOL_EditBoxDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL edit box draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_EditBoxDraw(void *pObject)

GFX_GOL_EditBoxTextSet Function

This function sets the text in the object.

File

gfx_gol_edit_box.h

C

```
void GFX_GOL_EditBoxTextSet(GFX_GOL_EDITBOX * pObject, GFX_XCHAR* pText);
```

Returns

None.

Description

GFX GOL edit box text set.

This function sets the text in the object. This function copies the text located in the address pointed to by pText to the object text buffer.

The string length must be less than or equal to the maximum characters allowed in the object. The object will truncate the string once it reaches the maximum length.

Preconditions

Object must exist in memory.

Example

```
#define MAX_EDITBOX_CHAR_LENGTH 20
GFX_XCHAR Label0[] = "This is really a long string";

// assume pEb is a pointer initialized to an edit box that
// as a buffer with 20 characters allowed.

GFX_GOL_EditBoxTextSet(pEb, Label0);
```

```
// at this point the edit box contains
// This is really a lon" - truncated to only 20 characters.
```

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be copied.

Function

GFX_XCHAR *GFX_GOL_EditBoxTextSet(
GFX_GOL_EDITBOX *pObject,
GFX_XCHAR *pText)

GFX_GOL_EditBoxCursorPositionDecrement Function

Decrement cursor position.

File

gfx_gol_edit_box.h

C

```
GFX_STATUS GFX_GOL_EditBoxCursorPositionDecrement(GFX_GOL_EDITBOX * pObject);
```

Returns

The status of the addition.

- GFX_STATUS_SUCCESS - Cursor position decrement successful.
- GFX_STATUS_FAILURE - Cursor position decrement failed.

Description

GFX GOL edit box cursor position decrement.

This function decrements cursor position.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_EditBoxCursorPositionDecrement(
GFX_GOL_EDITBOX *pObject)

GFX_GOL_EditBoxCursorPositionIncrement Function

Increment cursor position.

File

gfx_gol_edit_box.h

C

```
GFX_STATUS GFX_GOL_EditBoxCursorPositionIncrement(GFX_GOL_EDITBOX * pObject);
```

Returns

The status of the addition.

- GFX_STATUS_SUCCESS - Cursor position increment successful.
- GFX_STATUS_FAILURE - Cursor position increment failed.

Description

GFX GOL edit box cursor position increment.

This function increments cursor position.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_EditBoxCursorPostionIncrement(
GFX_GOL_EDITBOX *pObject)

Group Box Object

Group Box is an object that can be used to group objects together in the screen.

Functions

	Name	Description
	GFX_GOL_GroupboxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_GroupboxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_GroupboxCreate	This function creates a GFX_GOL_GROUPBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_GroupboxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_GroupboxTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_GroupboxTextSet	This function sets the address of the current text string used by the object.

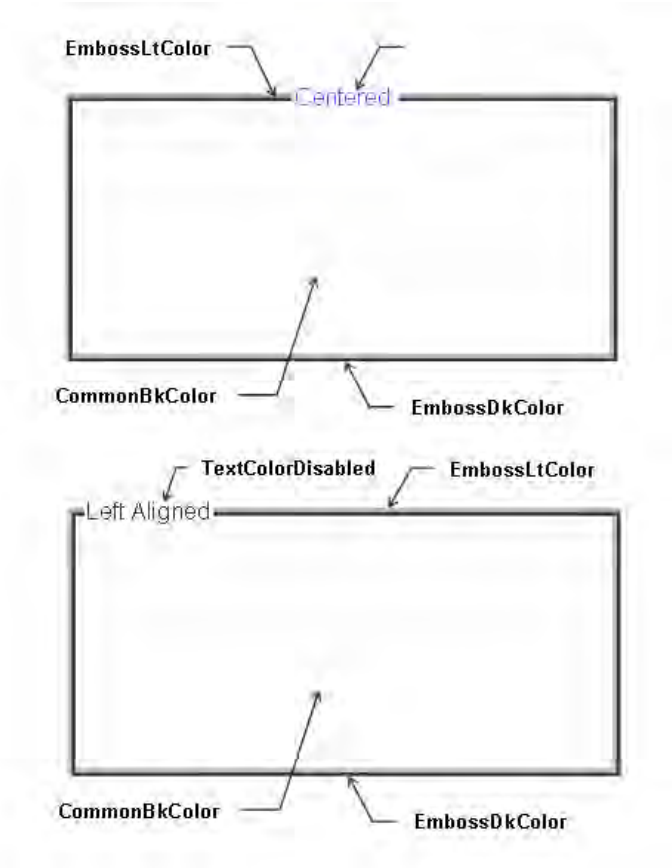
Macros

	Name	Description
	GFX_GOL_GroupboxTextGet	This function returns the address of the current text string used by the object.

Description

Group Box supports only Touchscreen inputs, replying to the events with the predefined actions (see GFX_GOL_GroupBoxActionGet() for details). There is no default action set function in this object. Application can create specific responses to the system action in the message callback function.

The Group Box object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



GFX_GOL_GroupboxTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_group_box.h

C

```
#define GFX_GOL_GroupboxTextGet(pObject) (((GFX_GOL_GROUPBOX *)pObject)->pText)
```

Returns

Pointer to text string.

Description

GFX GOL group box text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR *pChar;
GFX_GOL_GROUPBOX pGroupbox;

pChar = GFX_GOL_GroupboxTextGet(pGroupbox);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

```
GFX_XCHAR *GFX_GOL_GroupboxTextGet(
                GFX_GOL_GROUPBOX *pObject)
```

GFX_GOL_GroupboxActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_group_box.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_GroupboxActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_GROUPBOX_ACTION_SELECTED - Group Box area selected action ID.
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL group box action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
GFX_GOL_GROUPBOX_ACTION_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If events occurs and the x,y position falls in the area of the group box.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_GroupboxActionGet(

void *pObject,

GFX_GOL_MESSAGE *pMessage);

GFX_GOL_GroupboxDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_group_box.h

C

```
GFX_STATUS GFX_GOL_GroupboxDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL group box draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the face of the GFX_GOL_GROUPBOX is drawn on top of the bitmap. Text alignment based on the alignment parameter set on the

object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_GroupboxDraw(void *pObject)

GFX_GOL_GroupboxCreate Function

This function creates a GFX_GOL_GROUPBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_group_box.h

C

```
GFX_GOL_GROUPBOX * GFX_GOL_GroupboxCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL group box create.

This function creates a GFX_GOL_GROUPBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_GROUPBOX *pGroupbox;
GFX_GOL_GROUPBOX_STATE state;

// assume pScheme is initialized to a scheme in memory.
state = GFX_GOL_GROUPBOX_DRAW_STATE;

pGroupbox  = GFX_GOL_GroupboxCreate(
       gfxIndex,
       1,                              // ID
       0, 0,
       GFX_Primitive_MaxXGet(),
       GFX_Primitive_MaxYGet(),        // whole screen dimension
       state,                          // set state to draw all
       "Place Title Here.",            // text
       GFX_ALIGN_VCENTER,              // alignment of text
       NULL);                          // use default GOL scheme

if (pGroupbox == NULL)
    return 0;
```

```
    return 1;
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pImage	Pointer to the image used on the face of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_GROUPBOX  *GFX_GOL_GroupboxCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t            ID,
uint16_t            left,
uint16_t            top,
uint16_t            right,
uint16_t            bottom,
uint16_t            state,
                    GFX_RESOURCE_HDR    *pImage,
                    GFX_XCHAR           *pText,
                    GFX_ALIGNMENT       alignment,
                    GFX_GOL_OBJ_SCHEME  *pScheme)
```

GFX_GOL_GroupboxTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_group_box.h

C

```
GFX_ALIGNMENT GFX_GOL_GroupboxTextAlignmentGet(GFX_GOL_GROUPBOX * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL group box text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

```
    GFX_ALIGNMENT GFX_GOL_GroupboxTextAlignmentGet(
```

GFX_GOL_GROUPBOX *pObject)

GFX_GOL_GroupboxTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_group_box.h

C

```
void GFX_GOL_GroupboxTextAlignmentSet(GFX_GOL_GROUPBOX * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

GFX GOL group box text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_GroupboxTextAlignmentSet(
GFX_GOL_GROUPBOX *pObject,
GFX_ALIGNMENT align)

GFX_GOL_GroupboxTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_group_box.h

C

```
void GFX_GOL_GroupboxTextSet(GFX_GOL_GROUPBOX * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

GFX GOL group box text set.

This function sets the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR Label0[] = ?Group One?;
GFX_XCHAR Label1[] = ?Group Two?;
GFX_GOL_GROUPBOX pGroupbox;

GFX_GOL_GroupboxTextSet(pGroupbox, Label0);
GFX_GOL_GroupboxTextSet(pGroupbox, Label1);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be used.

Function

```
GFX_XCHAR *GFX_GOL_GroupboxTextSet(
                GFX_GOL_GROUPBOX *pObject,
                GFX_XCHAR *pText)
```

List Box Object

List Box is an object that defines a scrollable area where items are listed. User can select a single item or multiple of items.

Functions

	Name	Description
	GFX_GOL_ListBoxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ListBoxActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ListBoxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ListBoxItemAdd	This function adds an item to the list box.
	GFX_GOL_ListBoxItemFocusGet	This function returns the index of the focused item in the list box.
	GFX_GOL_ListBoxItemFocusSet	This function sets the focus of the item with the index number specified by index.
	GFX_GOL_ListBoxItemListRemove	This function removes the entire items list of the list box and free the memory used.
	GFX_GOL_ListBoxItemRemove	This function removes an item from the list box and free the memory used.
	GFX_GOL_ListBoxSelectionChange	This function changes the selection status of the given item in the list box.
	GFX_GOL_ListBoxSelectionGet	This function searches for selected items from the list box.
	GFX_GOL_ListBoxVisibleItemCountGet	This function returns the count of items visible in the list box.
	GFX_GOL_ListBoxCreate	This function creates a GFX_GOL_LISTBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

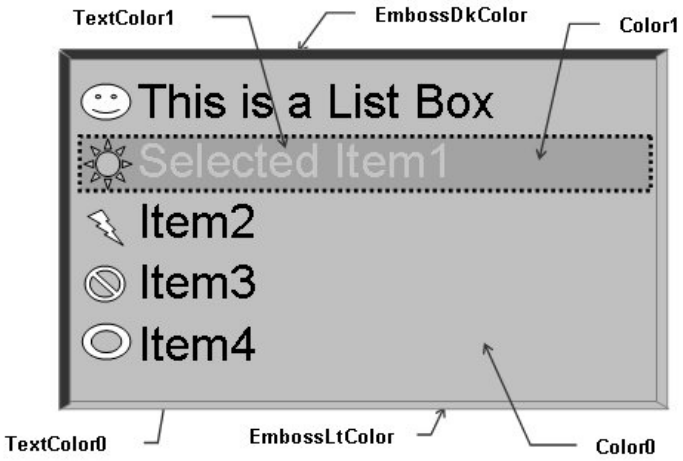
Macros

	Name	Description
	GFX_GOL_ListBoxItemCountGet	This function returns the count of items in the list box.
	GFX_GOL_ListBoxItemImageGet	This function gets the image set for a list box item.
	GFX_GOL_ListBoxItemImageSet	This function sets the image for a list box item.
	GFX_GOL_ListBoxItemListGet	This function returns the pointer to the item list of the list box.
	GFX_GOL_ListBoxItemSelectStatusClear	This function clears the selection status of the item.
	GFX_GOL_ListBoxItemSelectStatusSet	This function sets the selection status of the item to selected.
	GFX_GOL_ListBoxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_ListBoxTextAlignmentSet	This function sets the text alignment of the text string used by the object.

Description

List Box supports Keyboard and Touchscreen inputs, replying to their events with the predefined actions (see GFX_GOL_ListBoxActionGet() and GFX_GOL_ListBoxActionSet() for details).

The List Box object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the object.



GFX_GOL_ListBoxItemCountGet Macro

This function returns the count of items in the list box.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxItemCountGet(pObject) \
                (((GFX_GOL_LISTBOX *)pObject)->itemsNumber)
```

Returns

The number of items in the list box.

Description

GFX GOL list box item count get.

This function returns the count of items in the list box.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.

Function

uint16_t GFX_GOL_ListBoxItemCountGet(
GFX_GOL_LISTBOX *pObject)

GFX_GOL_ListBoxItemImageGet Macro

This function gets the image set for a list box item.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxItemImageGet(pItem) (((GFX_GOL_LISTITEM *)pItem)->pImage)
```

Returns

Pointer to the image used for the item.

Description

GFX GOL list box item image get.

This function gets the image set for a list box item.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_ListBoxItemListGet() example.

Parameters

Parameters	Description
pItem	pointer to the list box item.

Function

GFX_RESOURCE_HDR *GFX_GOL_ListBoxItemImageGet(
GFX_GOL_LISTITEM *pItem)

GFX_GOL_ListBoxItemImageSet Macro

This function sets the image for a list box item.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxItemImageSet(pItem, image) (((GFX_GOL_LISTITEM *)pItem)->pImage = image)
```

Returns

None.

Description

GFX GOL item image set.

This function sets the image for a list box item.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_ListBoxItemAdd() and GFX_GOL_ListBoxItemListGet() example.

Parameters

Parameters	Description
pItem	pointer to the list box item.
pImage	pointer to the image resource.

Function

void GFX_GOL_ListBoxItemImageSet(
GFX_GOL_LISTITEM *pItem,
GFX_RESOURCE_HDR *pImage)

GFX_GOL_ListBoxItemListGet Macro

This function returns the pointer to the item list of the list box.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxItemListGet(pObject) \
                ((GFX_GOL_LISTITEM *)((GFX_GOL_LISTBOX *)pObject)->pItemList)
```

Returns

The pointer to the item list of the list box.

Description

GFX GOL list box item list get.

This function returns the pointer to the item list of the list box.

Preconditions

Object must exist in memory.

Example

```
// assume:
// pLb is a initialized to a list box in memory
// myIcon is a valid image in memory

GFX_GOL_LISTITEM *pList, *pItem;
GFX_RESOURCE_HDR *pMyIcon = &myIcon;

pList = GFX_GOL_ListBoxItemListGet(pLb);
pItem = pList;

// set the image for all the items to myIcon
// set the last item image to NULL

do
{
    GFX_GOL_ListBoxImageSet(pItem, &myIcon);
}
while(pItem->pNextItem != NULL);

// get the last item's image and set to null if not null
if (GFX_GOL_ListBoxItemImageGet(pItem) != NULL)
    GFX_GOL_ListBoxImageSet(pItem, NULL);
```

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_GOL_LISTITEM *GFX_GOL_ListBoxItemListGet(
GFX_GOL_LISTBOX *pObject)

GFX_GOL_ListBoxItemSelectStatusClear Macro

This function clears the selection status of the item.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxItemSelectStatusClear(pObject, pItem) \
                                                             \
                if(pItem->status & GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED)    \
                    GFX_GOL_ListBoxSelectionChange((GFX_GOL_LISTBOX *)pObject, pItem);
```

Returns

None.

Description

GFX GOL box item select status clear.

This function clears the selection status of the item.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.
pItem	The pointer the item that will have the selected status cleared.

Function

void GFX_GOL_ListBoxItemSelectStatusClear(
GFX_GOL_LISTBOX *pObject,
GFX_GOL_LISTITEM *pItem)

GFX_GOL_ListBoxItemSelectStatusSet Macro

This function sets the selection status of the item to selected.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxItemSelectStatusSet(pObject, pItem) \
                                                                \
                if(!(pItem->status & GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED)) \
                    GFX_GOL_ListBoxSelectionChange((GFX_GOL_LISTBOX *)pObject, pItem);
```

Returns

None.

Description

GFX GOL list box item select status set.

This function sets the selection status of the item to selected.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.
pItem	The pointer the item that will have the selected status set to selected.

Function

void GFX_GOL_ListBoxItemSelectStatusSet(
GFX_GOL_LISTBOX *pObject,
GFX_GOL_LISTITEM *pItem)

GFX_GOL_ListBoxTextAlignmentGet Macro

This function returns the text alignment of the text string used by the object.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxTextAlignmentGet(pObject) \
                (((GFX_GOL_LISTBOX *)pObject)->alignment)
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL list box text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_ListBoxTextAlignmentGet(
GFX_GOL_LISTBOX *pObject)

GFX_GOL_ListBoxTextAlignmentSet Macro

This function sets the text alignment of the text string used by the object.

File

gfx_gol_list_box.h

C

```
#define GFX_GOL_ListBoxTextAlignmentSet(pObject, align) \
                (((GFX_GOL_LISTBOX *)pObject)->alignment = align)
```

Returns

None.

Description

GFX GOL list box text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_ListBoxTextAlignmentSet(
GFX_GOL_LISTBOX *pObject,
GFX_ALIGNMENT align)

GFX_GOL_ListBoxActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_list_box.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_ListBoxActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_LISTBOX_ACTION_TOUCHSCREEN â€“ Item is selected using touch screen
- GFX_GOL_LISTBOX_ACTION_MOVE â€“ Focus is moved to the next item depending on the key pressed (UP or DOWN key).
- GFX_GOL_LISTBOX_ACTION_SELECTED â€“ Selection is set to the currently focused item
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL list box action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_LISTBOX_ACTION_TOUCHSCREEN	Touch Screen	Any	Item is selected using touch screen.
GFX_GOL_LISTBOX_ACTION_MOVE	Keyboard	EVENT_KEYSCAN	Focus is moved to the next item depending on the key pressed (UP or DOWN key).
GFX_GOL_LISTBOX_ACTION_SELECTED	Keyboard	EVENT_KEYSCAN	Selection status (GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED) is set to the currently focused item.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_ListBoxActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage);

GFX_GOL_ListBoxActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_list_box.h

C

```
void GFX_GOL_ListBoxActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL list box action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_LISTBOX_ACTION_TOUCHSCREEN	Touch Screen	Set GFX_GOL_LISTBOX_FOCUSED_STATE,	If focus is enabled, the focus state bit GFX_GOL_LISTBOX_FOCUSED_STATE will be set.
		Set GFX_GOL_LISTBOX_DRAW_FOCUS_STATE	GFX_GOL_LISTBOX_DRAW_FOCUS_STATE draw state bit will force the List Box to be redrawn with focus.
		Set GFX_GOL_LISTBOX_DRAW_ITEMS_STATE	List Box will be redrawn with selected item(s).
GFX_GOL_LISTBOX_ACTION_MOVE	KeyBoard	Set GFX_GOL_LISTBOX_DRAW_ITEMS_STATE	List Box will be redrawn with focus on one item.
GFX_GOL_LISTBOX_ACTION_SELECTED	KeyBoard	Set GFX_GOL_LISTBOX_DRAW_ITEMS_STATE	List Box will be redrawn with selection on the current item focused.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_ListBoxActionSet(

GFX_GOL_TRANSLATED_ACTION translatedMsg,

void *pObject,

GFX_GOL_MESSAGE *pMessage)

GFX_GOL_ListBoxDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_list_box.h

C

```
GFX_STATUS GFX_GOL_ListBoxDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL list box draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_ListBoxDraw(void *pObject)

GFX_GOL_ListBoxItemAdd Function

This function adds an item to the list box.

File

gfx_gol_list_box.h

C

```
GFX_GOL_LISTITEM * GFX_GOL_ListBoxItemAdd(GFX_GOL_LISTBOX * pObject, GFX_GOL_LISTITEM * pPrevItem,
GFX_XCHAR * pText, GFX_RESOURCE_HDR * pImage, uint16_t status, uint16_t data);
```

Returns

The pointer to the created item.

Description

GFX GOL list box item add.

This function adds an item to the list box. This function allocates the memory needed for the GFX_GOL_LISTITEM and adds it to the list box. The newly created GFX_GOL_LISTITEM will store the location of pText, pImage and other parameters describing the added item.

Preconditions

Object must exist in memory.

Example

```
const GFX_XCHAR ItemList[] = "Line1n" "Line2n" "Line3n";

extern GFX_RESOURCE_HDR myIcon;
GFX_GOL_LISTBOX         *pLb;
GFX_GOL_LISTITEM        *pItem, *pItemList;
GFX_XCHAR               *pTemp;

// Assume that pLb is pointing to an existing list box in memory
// that is empty (no list).

// Create the list of the list box

// Initialize this to NULL to indicate that items will be added
// at the end of the list if the list exist on the list box or
// start a new list if the list box is empty.
pItem = NULL;
pTemp = ItemList;
pItem = GFX_GOL_ListBoxItemAdd(
                pLb,
                pItem,
                pTemp,
                NULL,
                LB_STS_SELECTED,
                1);
if(pItem == NULL)
    return 0;
GFX_GOL_ListBoxImageSet(pItem, &myIcon);

// Adjust pTemp to point to the next line
```

```
    while((uint16_t)*pTemp++ > (uint16_t)31);

    // add the next item
    pItem = GFX_GOL_ListBoxItemAdd(
                    pLb,
                    pItem,
                    pTemp,
                    NULL,
                    0,
                    2)
    if(pItem == NULL)
        return 0;
    GFX_GOL_ListBoxImageSet(pItem, &myIcon);

    // Adjust pTemp to point to the next line
    while((uint16_t)*pTemp++ > (uint16_t)31);

    // this time insert the next item after the first item on the list
    pItem = LbGetItemList(pLb);
    pItem = GFX_GOL_ListBoxItemAdd(
                    pLb,
                    pItem,
                    pTemp,
                    NULL,
                    0,
                    3)
    if(pItem == NULL)
        return 0;
    GFX_GOL_ListBoxImageSet(pItem, &myIcon);
```

Parameters

Parameters	Description
pObject	Pointer to the object.
pPrevItem	Pointer to the item after which a new item must be inserted, if this pointer is NULL, the item will be appended at the end of the items list.
pText	Pointer to the text that will be inserted. Text must persist in memory for as long as it is referenced by an item in the list box.
pImage	Pointer to the image for the item. Image must persist in memory for as long as it is referenced by the an item in the list box.
status	This parameter specifies if the item being added will be selected or redrawn (LB_STS_SELECTED or LB_STS_REDRAW). Refer to LISTITEM structure for details.
data	User assigned data associated with the item.

Function

```
GFX_GOL_LISTITEM *GFX_GOL_ListBoxItemAdd(
                GFX_GOL_LISTBOX     *pObject,
                GFX_GOL_LISTITEM    *pPrevItem,
                GFX_XCHAR           *pText,
                GFX_RESOURCE_HDR    *pImage,
    uint16_t        status,
    uint16_t        data)
```

GFX_GOL_ListBoxItemFocusGet Function

This function returns the index of the focused item in the list box.

File

gfx_gol_list_box.h

C

```
int16_t GFX_GOL_ListBoxItemFocusGet(GFX_GOL_LISTBOX * pObject);
```

Returns

The index of the focused item in the list box.

Description

GFX GOL list box item focus get.

This function returns the index of the focused item in the list box. First item on the list is always indexed 0. If none of the items in the list is focused, -1 is returned.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.

Function

int16_t GFX_GOL_ListBoxItemFocusGet(
GFX_GOL_LISTBOX *pObject)

GFX_GOL_ListBoxItemFocusSet Function

This function sets the focus of the item with the index number specified by index.

File

gfx_gol_list_box.h

C

```
void GFX_GOL_ListBoxItemFocusSet(GFX_GOL_LISTBOX * pObject, uint16_t index);
```

Returns

None.

Description

GFX GOL list box item focus set.

This function sets the focus of the item with the index number specified by index. First item on the list is always indexed 0.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.
index	The index number of the item to be focused.

Function

void GFX_GOL_ListBoxItemFocusSet(
GFX_GOL_LISTBOX *pObject,
uint16_t index)

GFX_GOL_ListBoxItemListRemove Function

This function removes the entire items list of the list box and free the memory used.

File

gfx_gol_list_box.h

C

```
void GFX_GOL_ListBoxItemListRemove(void * pObject);
```

Returns

None.

Description

GFX GOL list box item list remove.

This function removes the entire items list of the list box and free the memory used. The memory freed is the memory used for the list box. The actual text or image used for the item are not removed from memory.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_ListBoxItemListRemove(

void *pObject)

GFX_GOL_ListBoxItemRemove Function

This function removes an item from the list box and free the memory used.

File

gfx_gol_list_box.h

C

```
void GFX_GOL_ListBoxItemRemove(GFX_GOL_LISTBOX * pObject, GFX_GOL_LISTITEM * pItem);
```

Returns

None.

Description

GFX GOL list box item remove.

This function removes an item from the list box and free the memory used. The memory freed is the memory used for the list box. The actual text or image used for the item are not removed from memory.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pItem	pointer to the list box item that will be removed.

Function

void GFX_GOL_ListBoxItemRemove(

GFX_GOL_LISTBOX *pObject

GFX_GOL_LISTITEM *pItem)

GFX_GOL_ListBoxSelectionChange Function

This function changes the selection status of the given item in the list box.

File

gfx_gol_list_box.h

C

```
void GFX_GOL_ListBoxSelectionChange(GFX_GOL_LISTBOX * pObject, GFX_GOL_LISTITEM * pItem);
```

Returns

None.

Description

GFX GOL list box selection change.

This function changes the selection status of the given item in the list box. There are two cases that this function checks Case 1: The list box is set to multiple selection.

- The item pointed to by pItem will toggle the selection status.

Case 2: The list box is set to single selection.

- If the currently selected item is not the item pointed to by pItem, the currently selected item will toggle to not selected. The item pointed to by pItem will then be set to selected.
- If the currently selected item is the same item pointed to by pItem, the selection status of that item will be set to not selected.

The change in the item's selection status should be redrawn to reflect the changes.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.
pItem	The pointer to the item that will have the selection status changed.

Function

void GFX_GOL_ListBoxSelectionChange(
GFX_GOL_LISTBOX *pObject,
GFX_GOL_LISTITEM *pItem)

GFX_GOL_ListBoxSelectionGet Function

This function searches for selected items from the list box.

File

gfx_gol_list_box.h

C

```
GFX_GOL_LISTITEM * GFX_GOL_ListBoxSelectionGet(GFX_GOL_LISTBOX * pObject, GFX_GOL_LISTITEM * pFromItem);
```

Returns

The pointer to the first selected item found. NULL if there are no items selected.

Description

GFX GOL list box selection get.

This function searches for selected items from the list box. A starting position can optionally be given. If starting position is set to NULL, search will begin from the first item on the item list.

The function returns the pointer to the first selected item found or NULL if there are no items selected.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.

pFromItem	The starting point of the search. If this is set to NULL, the search will start at the beginning of the item list.

Function

GFX_GOL_LISTITEM *GFX_GOL_ListBoxSelectionGet(
GFX_GOL_LISTBOX *pObject,
GFX_GOL_LISTITEM *pFromItem)

GFX_GOL_ListBoxVisibleItemCountGet Function

This function returns the count of items visible in the list box.

File

gfx_gol_list_box.h

C

```
uint16_t GFX_GOL_ListBoxVisibleItemCountGet(GFX_GOL_LISTBOX * pObject);
```

Returns

The number of visible items in the list box.

Description

GFX GOL list box visible item count get.

This function returns the count of items visible in the list box.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object.

Function

uint16_t GFX_GOL_ListBoxVisibleItemCountGet(
GFX_GOL_LISTBOX *pObject)

GFX_GOL_ListBoxCreate Function

This function creates a GFX_GOL_LISTBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_list_box.h

C

```
GFX_GOL_LISTBOX * GFX_GOL_ListBoxCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL list box create.

This function creates a GFX_GOL_LISTBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_ListBoxCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom

- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
#define LISTBOX_ID   10
const XCHAR ItemList[] = "Line1n" "Line2n";

GFX_GOL_OBJ_SCHEME  *pScheme;
GFX_GOL_LISTBOX     *pLb;
GFX_XCHAR           *pTemp;
uint16_t            state, counter;

//assume scheme is a valid style scheme in memory
pScheme = &scheme;
state = GFX_GOL_LISTBOX_DRAW_STATE;

// create an empty list box with default style scheme
pLb = GFX_GOL_ListBoxCreate(    gfxIndex,
                                LISTBOX_ID,     // ID number
                                10,10,150,200,  // dimension
                                state,          // initial state
                                NULL,           // set items to be empty
                                GFX_ALIGN_CENTER,
                                NULL);          // use default style scheme
// check if List box was created
if (pLb == NULL)
   return 0;

// create the list of items to be placed in the list box
// Add items (each line will become one item,
// lines must be separated by 'n' character)
pTemp = ItemList;
counter = 0;

while(*pTemp)
{
    // since each item is appended NULL is assigned to
    // GFX_GOL_LISTITEM pointer.
    if(NULL == GFX_GOL_ListBoxItemAdd(
                pLb,
                NULL,
                pTemp,
                NULL,
                0,
                counter))
    {
        break;
    }

    while(*pTemp++ > (unsigned GFX_XCHAR)31);

    if(*(pTemp-1) == 0)
        break;
    counter++;
}
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.

right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pText	Pointer to the text of the object. This is used for the items of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_LISTBOX *GFX_GOL_ListBoxCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t        ID,
uint16_t        left,
uint16_t        top,
uint16_t        right,
uint16_t        bottom,
uint16_t        state,
                GFX_XCHAR           *pText,
                GFX_ALIGNMENT       alignment,
                GFX_GOL_OBJ_SCHEME  *pScheme)
```

Meter Object

Meter is an object that can be used to graphically display a sampled input.

Functions

	Name	Description
	GFX_GOL_MeterActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_MeterActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_MeterCreate	This function creates a GFX_GOL_METER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_MeterDecrement	This function decrements the meter value by the delta value set.
	GFX_GOL_MeterDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_MeterIncrement	This function increments the meter value by the delta value set.
	GFX_GOL_MeterRangeSet	This function sets the range of the meter.
	GFX_GOL_MeterScaleColorsSet	This function sets the arc colors of the object.
	GFX_GOL_MeterValueSet	This function sets the value of the meter.

Macros

	Name	Description
	GFX_GOL_MeterMaximumValueGet	This function returns the maximum value set for the meter.
	GFX_GOL_MeterMinimumValueGet	This function returns the minimum value set for the meter.
	GFX_GOL_MeterTitleFontSet	This function sets the meter title font used.
	GFX_GOL_MeterTypeSet	This function sets the meter draw type.
	GFX_GOL_MeterValueFontSet	This function sets the meter's displayed value font used.
	GFX_GOL_MeterValueGet	This function returns the current value of the meter.

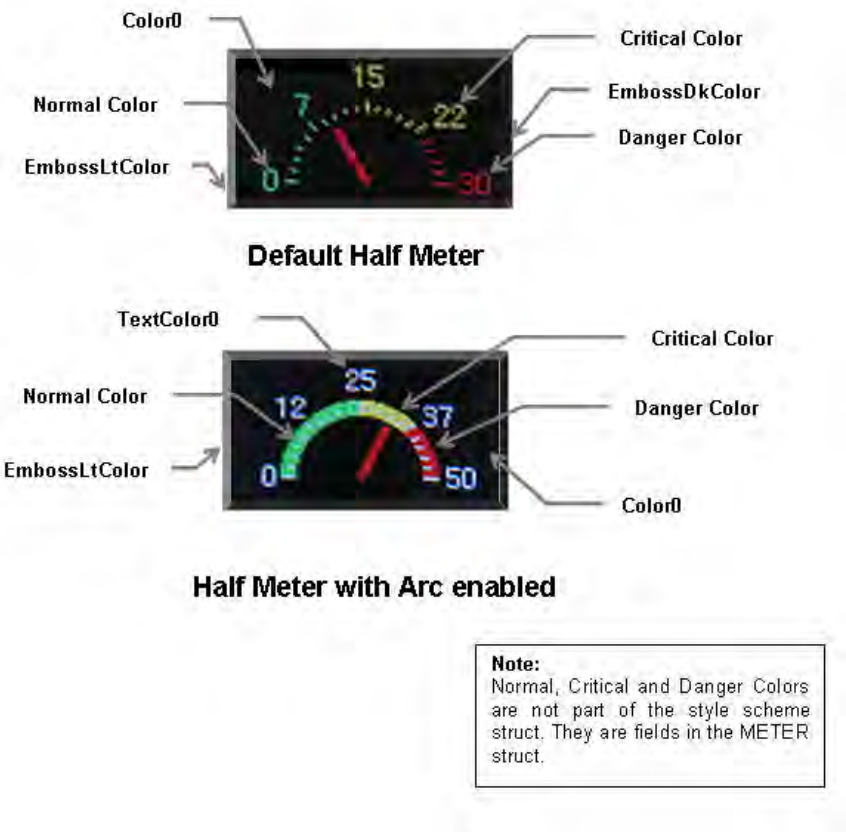
Description

There are three meter types that you can draw:

- MTR_WHOLE_TYPE
- MTR_HALF_TYPE
- MTR_QUARTER_TYPE

Meter supports system inputs, replying to their events with the predefined actions (see GFX_GOL_MeterActionGet() and GFX_GOL_MeterActionSet() for details).

The Meter object is rendered using the assigned style scheme, value range colors (see GFX_GOL_MeterScaleColorSet() for details) and type (see GFX_GOL_METER_DRAW_TYPE). The following figure illustrates the assignments for a half type meter.



Note:
Normal, Critical and Danger Colors are not part of the style scheme struct. They are fields in the METER struct.

GFX_GOL_MeterMaximumValueGet Macro

This function returns the maximum value set for the meter.

File

gfx_gol_meter.h

C

```
#define GFX_GOL_MeterMaximumValueGet(pObject) \
        (((GFX_GOL_METER *)pObject)->maxValue)
```

Returns

The current maximum value set for the object.

Description

GFX GOL meter maximum value get.

This function returns the maximum value set for the meter.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_MeterMaximumValueGet(
GFX_GOL_METER *pObject)

GFX_GOL_MeterMinimumValueGet Macro

This function returns the minimum value set for the meter.

File

gfx_gol_meter.h

C

```
#define GFX_GOL_MeterMinimumValueGet(pObject) \
        (((GFX_GOL_METER *)pObject)->minValue)
```

Returns

The current minimum value set for the object.

Description

GFX GOL meter minimum value get.

This function returns the minimum value set for the meter.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_MeterMinimumValueGet(
GFX_GOL_METER *pObject)

GFX_GOL_MeterTitleFontSet Macro

This function sets the meter title font used.

File

gfx_gol_meter.h

C

```
#define GFX_GOL_MeterTitleFontSet(pObject, pNewFont) \
        (((GFX_GOL_METER *)pObject)->pTitleFont = pNewFont)
```

Returns

None.

Description

GFX GOL meter title font set.

This function sets the meter title font used. Font pointer must be initialized to a valid GFX_RESOURCE_HDR.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pNewFont	pointer to the selected font.

Function

void GFX_GOL_MeterTitleFontSet(

GFX_GOL_METER *pObject,
GFX_RESOURCE_HDR *pNewFont)

GFX_GOL_MeterTypeSet Macro

This function sets the meter draw type.

File

gfx_gol_meter.h

C

```
#define GFX_GOL_MeterTypeSet(pObject, type) \
        (((GFX_GOL_METER *)pObject)->type = type)
```

Returns

None.

Description

GFX GOL meter type set.

This function sets the meter draw type.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
type	the draw type selected (see GFX_GOL_METER_DRAW_TYPE for more information).

Function

void GFX_GOL_MeterTypeSet(
GFX_GOL_METER *pObject,
GFX_GOL_METER_DRAW_TYPE type)

GFX_GOL_MeterValueFontSet Macro

This function sets the meter's displayed value font used.

File

gfx_gol_meter.h

C

```
#define GFX_GOL_MeterValueFontSet(pObject, pNewFont) \
        (((GFX_GOL_METER *)pObject)->pValueFont = pNewFont)
```

Returns

None.

Description

GFX GOL meter value font set.

This function sets the meter's displayed value font used. Font pointer must be initialized to a valid GFX_RESOURCE_HDR.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

pNewFont	pointer to the selected font.

Function

void GFX_GOL_MeterValueFontSet(
GFX_GOL_METER *pObject,
GFX_RESOURCE_HDR *pNewFont)

GFX_GOL_MeterValueGet Macro

This function returns the current value of the meter.

File

gfx_gol_meter.h

C

```
#define GFX_GOL_MeterValueGet(pMtr) (((GFX_GOL_METER*)pMtr)->value)
```

Returns

The current value of the object.

Description

GFX GOL meter value get.

This function returns the current value of the meter.

Preconditions

Object must exist in memory.

Example

```
#define MAXVALUE 100;

GFX_GOL_METER *pMeter;
uint32_t ctr = 0;

// create scroll bar here and initialize parameters
pMeter = GFX_GOL_MeterCreate(....)
GFX_GOL_ObjectStateSet(pMeter, GFX_GOL_METER_DRAW_STATE);

// draw the scroll bar
GFX_GOL_ObjectListDraw(gfxIndex);

// a routine that updates the position of the thumb through some
// conditions
while("some condition")
{
    GFX_GOL_MeterValueSet(pMeter, ctr);
    GFX_GOL_ObjectStateSet( pMeter,
                            GFX_GOL_METER_UPDATE_DRAW_STATE);

    // update the screen
    GFX_GOL_ObjectListDraw(gfxIndex);

    // update ctr here
    ctr = "some source of value";
}

if (GFX_GOL_MeterValueGet(pScrollBar) > MAXVALUE)
    return 0;
else
    "do something else"
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_MeterValueGet(
GFX_GOL_METER *pObject)

GFX_GOL_MeterActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_meter.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_MeterActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_METER_ACTION_SET â€“ Meter id is given in parameter 1 for a TYPE_SYSTEM message.
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL meter action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_METER_ACTION_SET	System	EVENT_SET	If event set occurs and the meter id is sent in parameter 1.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_MeterActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_MeterActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_meter.h

C

```
void GFX_GOL_MeterActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL meter action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_METER_ACTION_SET	System	Set GFX_GOL_METER_DRAW_STATE	Meter will be redrawn to update the needle position and value displayed. The new value is given in the pMessage parameter 2 (param2). While parameter 1 (param1) of the message holds the ID of the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_MeterActionSet(

GFX_GOL_TRANSLATED_ACTION translatedMsg,

void *pObject,

GFX_GOL_MESSAGE *pMessage)

GFX_GOL_MeterCreate Function

This function creates a GFX_GOL_METER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_meter.h

C

```
GFX_GOL_METER * GFX_GOL_MeterCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t top,
uint16_t right, uint16_t bottom, uint16_t state, GFX_GOL_METER_DRAW_TYPE type, int16_t value, int16_t
minValue, int16_t maxValue, GFX_RESOURCE_HDR * pTitleFont, GFX_RESOURCE_HDR * pValueFont, GFX_XCHAR *
pText, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL meter create.

This function creates a GFX_GOL_METER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_MeterCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- type is not one of the defined types
- pTitleFont and pValueFont is not defined to a valid font GFX_RESOURC_HDR
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
#define ID_METER 101

extern const FONT_FLASH GOLMediumFont;      // medium font
extern const FONT_FLASH GOLSmallFont;       // small font

GFX_GOL_OBJ_SCHEME *pMeterScheme;
GFX_GOL_METER *pMtr;

// assume pMeterScheme is initialized to a scheme in memory.

// draw object after creation
state = GFX_GOL_METER_DRAW_STATE | GFX_GOL_METER_RING_STATE;

pMtr = GFX_GOL_MeterCreate(
            gfxIndex,
            ID_METER,                   // assign ID
            30, 50, 150, 180,           // set dimension
            state,
            GFX_GOL_METER_WHOLE_TYPE,   // type of meter
            0,                          // set initial value
            0, 100,                     // set min and max value
            &GOLMediumFont,             // set title font
            &GOLSmallFont,              // set value font
            "Speed",                    // Text Label
            pMeterScheme);              // style scheme

// check if meter was created
if (pMtr == NULL)
    return 0;

// Change range colors: Normal values to WHITE
//                      Critical values to BLUE
//                      Danger values to RED
// assume that WHITE, GREEN, YELLOW and RED have been defined.
GFX_GOL_MeterScaleColorSet(pMtr, WHITE, WHITE, WHITE,
                                 GREEN, YELLOW, RED);

// use GOLDraw() to draw the meter created
while(!GOLDraw());
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
type	Specifies the type of Meter to be drawn (see GFX_GOL_METER_TYPE).
value	Initial value set to the meter.
minValue	The minimum value the meter will display.
maxValue	The maximum value the meter will display.
pTitleFont	Pointer to the font used for the Title.
pValueFont	Pointer to the font used for the value.
pText	Pointer to the text label of the meter.
pScheme	Pointer to the style scheme used.

Function

```
GFX_GOL_METER *GFX_GOL_MeterCreate(
SYS_MODULE_INDEX    gfxIndex,
```

```
uint16_t            ID,
uint16_t            left,
uint16_t            top,
uint16_t            right,
uint16_t            bottom,
uint16_t            state,
                    GFX_GOL_METER_DRAW_TYPE type,
int16_t             value,
int16_t             minValue,
int16_t             maxValue,
                    GFX_RESOURCE_HDR       *pTitleFont,
                    GFX_RESOURCE_HDR       *pValueFont,
                    GFX_XCHAR              *pText,
                    GFX_GOL_OBJ_SCHEME     *pScheme)
```

GFX_GOL_MeterDecrement Function

This function decrements the meter value by the delta value set.

File

gfx_gol_meter.h

C

```
void GFX_GOL_MeterDecrement(GFX_GOL_METER * pObject, uint16_t delta);
```

Returns

None.

Description

GFX GOL meter decrement.

This function decrements the meter value by the given delta value set. If the delta given is less than the minimum value of the meter, the value will remain to be at minimum.

Object must be redrawn after this function is called to reflect the changes to the object.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_MeterIncrement().

Parameters

Parameters	Description
pObject	pointer to the object.

Function

```
void GFX_GOL_MeterDecrement(
                    GFX_GOL_METER *pObject,
uint16_t delta)
```

GFX_GOL_MeterDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_meter.h

C

```
GFX_STATUS GFX_GOL_MeterDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished.

Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL meter draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_MeterDraw(void *pObject)

GFX_GOL_MeterIncrement Function

This function increments the meter value by the delta value set.

File

gfx_gol_meter.h

C

```
void GFX_GOL_MeterIncrement(GFX_GOL_METER * pObject, uint16_t delta);
```

Returns

None.

Description

GFX GOL meter increment

This function increments the scroll bar position by the given delta value set. If the delta given exceeds the maximum value of the meter, the value will remain to be at maximum.

Object must be redrawn after this function is called to reflect the changes to the object.

Preconditions

Object must exist in memory.

Example

```
void ControlSpeed(  GFX_GOL_METER* pObj,
                    int setSpeed,
                    int curSpeed)
{
    // set page size to 1
    GFX_GOL_MeterValueSet(pObj, 1);

    if (setSpeed < curSpeed)
    {
        while(GFX_GOL_MeterValueGet(pObj) < SetSpeed)
            GFX_GOL_MeterIncrement(pObj, 1);  // increment by 1
    }
    else if (setSpeed > curSpeed)
    {
        while(GFX_GOL_MeterValueGet(pObj) > SetSpeed)
            GFX_GOL_MeterDecrement(pObj, 1);  // decrement by 1
    }
}
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_MeterIncrement(

GFX_GOL_METER *pObject,

uint16_t delta)

GFX_GOL_MeterRangeSet Function

This function sets the range of the meter.

File

gfx_gol_meter.h

C

```
void GFX_GOL_MeterRangeSet(GFX_GOL_METER * pObject, int16_t minValue, int16_t maxValue);
```

Returns

None.

Description

GFX GOL meter range set.

This function sets the range of the meter. When the range is modified, object must be completely redrawn to reflect the change. minValue should always be less than maxValue.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
minValue	new minimum value of the object.
maxValue	new maximum value of the object.

Function

void GFX_GOL_MeterRangeSet(

GFX_GOL_METER *pObject,

int16_t minValue,

int16_t maxValue)

GFX_GOL_MeterScaleColorsSet Function

This function sets the arc colors of the object.

File

gfx_gol_meter.h

C

```
void GFX_GOL_MeterScaleColorsSet(GFX_GOL_METER * pObject, GFX_COLOR color1, GFX_COLOR color2, GFX_COLOR
color3, GFX_COLOR color4, GFX_COLOR color5, GFX_COLOR color6);
```

Returns

None.

Description

GFX GOL meter scale colors set.

After the object is created, this function must be called to set the arc colors of the object.

Scale colors can be used to highlight values of the meter. User can set these colors to define the arc colors and scale colors. This also sets the color of the meter value when displayed. The color settings are set to the following angles:

Color Boundaries	Type Whole	Type Half	Type Quarter
Arc 6	225 to 180	not used	not used
Arc 5	179 to 135	179 to 135	not used
Arc 4	134 to 90	134 to 90	not used
Arc 3	89 to 45	89 to 45	89 to 45
Arc 2	44 to 0	44 to 0	44 to 0
Arc 1	-45 to -1	not used	not used

As the meter is drawn colors are changed depending on the angle of the scale and label being drawn.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
color1	color assigned to Arc 1 and Scale 1.
color2	color assigned to Arc 2 and Scale 2.
color3	color assigned to Arc 3 and Scale 3.
color4	color assigned to Arc 4 and Scale 4.
color5	color assigned to Arc 5 and Scale 5.
color6	color assigned to Arc 6 and Scale 6.

Function

void GFX_GOL_MeterScaleColorsSet(
GFX_GOL_METER *pObject,
GFX_COLOR color1,
GFX_COLOR color2,
GFX_COLOR color3,
GFX_COLOR color4,
GFX_COLOR color5,
GFX_COLOR color6)

GFX_GOL_MeterValueSet Function

This function sets the value of the meter.

File

gfx_gol_meter.h

C

```
void GFX_GOL_MeterValueSet(GFX_GOL_METER * pObject, int16_t value);
```

Returns

None.

Description

GFX GOL meter value set.

This function sets the value of the meter. The value used should be within the range set for the object. The new value is checked to be in the minimum and maximum value range. If the value set is less than the minimum value, the value that will be set is the minimum value. If the value set is more than the maximum value, then the value that will be set is the maximum value.

Preconditions

Object must exist in memory.

Example

```
GFX_GOL_METER *pMeter;
uint16_t ctr = 0;

// create slider here and initialize parameters
GFX_GOL_ObjectStateSet(pMeter, GFX_GOL_METER_DRAW_STATE);
GFX_GOL_ObjectListDraw(gfxIndex);

while("some condition")
{
    GFX_GOL_MeterValueSet(pMeter, ctr);
    GFX_GOL_ObjectStateSet( pMeter,
                            GFX_GOL_METER_UPDATE_DRAW_STATE);

    // redraw the scroll bar
    GFX_GOL_ObjectListDraw(gfxIndex);

   // update ctr here
   ctr = "some source of value";
}
```

Parameters

Parameters	Description
pObject	pointer to the object.
value	the new value of the object.

Function

void GFX_GOL_MeterValueSet(
GFX_GOL_METER *pObject,
int16_t value)

Panel Object

Functions

	Name	Description
	GFX_GOL_PanelAlphaParameterSet	This function sets the alpha blending value when using alpha blending in panels.
	GFX_GOL_PanelBackgroundSet	This function sets panel background information.
	GFX_GOL_PanelDraw	This function renders the panel.
	GFX_GOL_PanelGradientParameterSet	This function sets the gradient fill start and end colors of a panel.
	GFX_GOL_PanelParameterSet	This function sets the parameters to draw a panel.

Description

GFX_GOL_PanelAlphaParameterSet Function

This function sets the alpha blending value when using alpha blending in panels.

File

gfx_gol.h

C

```
void GFX_GOL_PanelAlphaParameterSet(SYS_MODULE_INDEX gfxIndex, uint16_t alphaValue);
```

Returns

None.

Description

GFX GOL panel alpha parameter set.

This function sets the alpha blending value when using alpha blending in panels. This along with the parameters set by the function GFX_GOL_PanelParameterSet() will determine how the panel will be drawn. The actual drawing of the panel is performed by GFX_GOL_PanelDraw().

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
alphaVlaue	the alpha value used to render alpha blended panels.

Function

void GFX_GOL_PanelAlphaParameterSet(SYS_MODULE_INDEX gfxIndex,
uint16_t alphaValue)

GFX_GOL_PanelBackgroundSet Function

This function sets panel background information.

File

gfx_gol.h

C

```
void GFX_GOL_PanelBackgroundSet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObjectHeader);
```

Returns

None.

Description

GFX GOL panel background set.

This function sets panel background information. This is an internal function and should not be called by the application.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObjectHeader	the object header of the object that needs to draw a panel with background.

Function

void GFX_GOL_PanelBackgroundSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObjectHeader)

GFX_GOL_PanelDraw Function

This function renders the panel.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_PanelDraw(SYS_MODULE_INDEX gfxIndex);
```

Returns

GFX_STATUS_SUCCESS - when the panel rendering is done. GFX_STATUS_FAILURE - when the panel rendering is not yet done.

Description

GFX GOL panel draw.

This function renders the panel. Panel parameters are set by the GFX_GOL_PanelParameterSet() and GFX_GOL_PanelGradientParameterSet() or GFX_GOL_PanelAlphaParameterSet(). The function returns success (GFX_STATUS_SUCCESS) when the panel is rendered. If the function returned not success this function must be called again until success is returned.

Preconditions

Panel parameters must be set first using GFX_GOL_PanelParameterSet() and GFX_GOL_PanelGradientParameterSet() or GFX_GOL_PanelAlphaParameterSet().

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_STATUS GFX_GOL_PanelDraw(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_PanelGradientParameterSet Function

This function sets the gradient fill start and end colors of a panel.

File

gfx_gol.h

C

```
void GFX_GOL_PanelGradientParameterSet(SYS_MODULE_INDEX gfxIndex, GFX_COLOR startColor, GFX_COLOR endColor);
```

Returns

None.

Description

GFX GOL panel gradient parameter set.

This function sets the gradient fill start and end colors of a panel. This along with the parameters set by the function GFX_GOL_PanelParameterSet() will determine how the panel will be drawn. The actual drawing of the panel is performed by GFX_GOL_PanelDraw().

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
startColor	the gradient fill start color.
endColor	the gradient fill end color.

Function

void GFX_GOL_PanelGradientParameterSet(SYS_MODULE_INDEX gfxIndex,

GFX_COLOR startColor,

GFX_COLOR endColor)

GFX_GOL_PanelParameterSet Function

This function sets the parameters to draw a panel.

File

gfx_gol.h

C

```
void GFX_GOL_PanelParameterSet(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right,
uint16_t bottom, uint16_t radius, GFX_COLOR faceClr, GFX_COLOR embossLtClr, GFX_COLOR embossDkClr,
```

```
GFX_RESOURCE_HDR * pBitmap, GFX_FILL_STYLE fillStyle, uint16_t embossSize);
```

Returns

None.

Description

GFX GOL panel parameter set.

This function sets the parameters to draw a panel. Panel is not an object. It is a routine to draw a basic component of objects. The actual drawing of the panel is performed by the GFX_GOL_PanelDraw(). After the parameters are set, call GFX_GOL_PanelDraw() to render the panel. The panel is drawn using the following:

1. Panel width is determined by right - left.
2. Panel height is determined by top - bottom.
3. Panel radius - specifies if the panel will have a rounded edge. If zero then the panel will have sharp (cornered) edge.
4. If 2*radius = height = width, the panel is circular.
5. If the panel is drawn with an image, pBitmap should point to an image resource.
6. If the panel face is drawn with the fill style specified by fillStyle. When gradient fill is used, set the gradient colors using GFX_GOL_PanelGradientParameterSet(). When alpha blending fill is used, set the alpha blending value using GFX_GOL_PanelAlphaParameterSet().

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
left	defines the left most pixel of the panel.
top	defines the top most pixel of the panel.
right	defines the right most pixel of the panel.
bottom	defines the bottom most pixel of the panel.
radius	defines the radius of the rounded corner. A zero value will result in a rectangular panel drawn.
faceClr	the color used for the face of the panel.
embossLtClr	the color used for the light emboss color for 3D effect.
embossDkClr	the color used for the dark emboss color for 3D effect.
pBitmap	pointer to the image resource of the panel.
fillStyle	fill style use for the face of the panel.
embossSize	when this is not zero, the embossLtClr and embossDkClr are used to draw the 3D effect. When this is set to zero, there will be no 3D effect.

Function

void GFX_GOL_PanelParameterSet(
SYS_MODULE_INDEX gfxIndex
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t radius,
GFX_COLOR faceClr,
GFX_COLOR embossLtClr,
GFX_COLOR embossDkClr,
GFX_RESOURCE_HDR *pBitmap,
GFX_FILL_STYLE fillStyle,
uint16_t embossSize)

Picture Control Object

Picture Control is an object that can be used to transform an image to be an object in the screen and have control on the rendering. This object

can be used to create animation using a series of bitmaps.

Functions

	Name	Description
	GFX_GOL_PictureControlActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_PictureControlCreate	This function creates a GFX_GOL_PICTURECONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_PictureControlDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_PictureControlPartialSet	This function sets the partial image parameters to be in the object.
	GFX_GOL_PictureControlScaleSet	Sets the scale factor used to render the image used in the object.

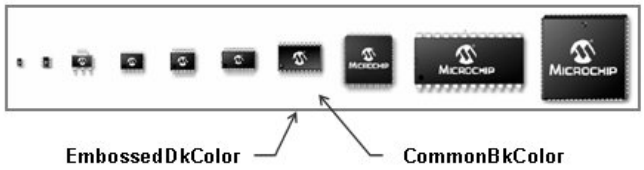
Macros

	Name	Description
	GFX_GOL_PictureControlImageGet	This function gets the image used when in the pressed state.
	GFX_GOL_PictureControlImageSet	This function sets the image to be in the object.

Description

Picture Control object supports Touchscreen inputs, replying to the events with the predefined actions (see GFX_GOL_PictureActionGet() for details).

The Picture object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the object.



GFX_GOL_PictureControlImageGet Macro

This function gets the image used when in the pressed state.

File

gfx_gol_picture.h

C

```
#define GFX_GOL_PictureControlImageGet(pObject) \
                    (((GFX_GOL_PICTURECONTROL*)pObject)->pImage)
```

Returns

Pointer to the image resource.

Description

GFX GOL picture control image get.

This function gets the image used when in the pressed state.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_RESOURCE_HDR *GFX_GOL_PictureControlImageGet(
GFX_GOL_PICTURECONTROL *pObject)

GFX_GOL_PictureControlImageSet Macro

This function sets the image to be in the object.

File

gfx_gol_picture.h

C

```
#define GFX_GOL_PictureControlImageSet(pObject, image) \
                    (((GFX_GOL_PICTURECONTROL*)pObject)->pImage = (GFX_RESOURCE_HDR *)image)
```

Returns

None.

Description

GFX GOL picture control image set.

This function sets the image to be in the object.

Preconditions

Object must exist in memory.

Example

```
// assume OrigImage and NewImage are valid GFX_RESOURCE_HDR
// pointers for images

GFX_RESOURCE_HDR *pOrigIcon = &OrigImage;
GFX_RESOURCE_HDR *pNewIcon = &NewImage;
GFX_GOL_PICTURECONTROL  *pPicture;

pPicture = GFX_GOL_PictureControlCreate(
                    gfxIndex,
                    10,
                    0, 0,
                    GFX_MaxXGet(), GFX_MaxYGet(),
                    GFX_GOL_PICTURECONTROL_DRAW_STATE,
                    pOrigIcon,
                    NULL);

// change the image used
GFX_GOL_PictureControlImageSet(pPicture , pNewIcon);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pImage	pointer to the image resource.

Function

void GFX_GOL_PictureControlImageSet(
GFX_GOL_PICTURECONTROL *pObject,
GFX_RESOURCE_HDR *pImage)

GFX_GOL_PictureControlActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_picture.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_PictureControlActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_PICTURECONTROL_ACTION_SELECTED â€“ Object is selected
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL picture control action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_PICTURECONTROL_ACTION_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If events occurs and the x,y position falls in the area of the picture.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_PictureControlActionGet(

void *pObject,

GFX_GOL_MESSAGE *pMessage);

GFX_GOL_PictureControlCreate Function

This function creates a GFX_GOL_PICTURECONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_picture.h

C

```
GFX_GOL_PICTURECONTROL * GFX_GOL_PictureControlCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t
left, uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, int8_t scaleFactor, GFX_RESOURCE_HDR *
pImage, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL picture control create.

This function creates a GFX_GOL_PICTURECONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The object allows creation with the image set to NULL. In this case, nothing will be drawn when the object is set to be drawn and the frame is not enabled (See GFX_GOL_PICTURECONTROL_FRAME_STATE). If the frame is enabled, then only the frame will be drawn.

When the assigned image's dimension is larger than the dimension of the object, partial image parameters will be set in such a way that the upper left most corner of the image that has the same dimension as the object will be used in the object. This is the default behavior.

The partial parameters can be modified by calling the GFX_GOL_PictureControlPartialSet() function with the desired partial image parameters. See GFX_ImagePartialDraw() for details on the partial image rendering.

The behavior of GFX_GOL_PictureControlCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pImage is not pointing to a GFX_RESOURCE_HDR.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
scaleFactor	Sets the scaling factor when the image is rendered. This feature is only available in certain builds.
pImage	Pointer to the image by the object
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_PICTURECONTROL  *GFX_GOL_PictureControlCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t            ID,
uint16_t            left,
uint16_t            top,
uint16_t            right,
uint16_t            bottom,
uint16_t            state,
int8_t              scaleFactor,
                    GFX_RESOURCE_HDR    *pImage,
                    GFX_GOL_OBJ_SCHEME  *pScheme)
```

GFX_GOL_PictureControlDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_picture.h

C

```
GFX_STATUS GFX_GOL_PictureControlDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL picture control draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When the image size The text on the face of the GFX_GOL_PICTURECONTROL is drawn on top of the bitmap. Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to

avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_PictureControlDraw(void *pObject)

GFX_GOL_PictureControlPartialSet Function

This function sets the partial image parameters to be in the object.

File

gfx_gol_picture.h

C

```
void GFX_GOL_PictureControlPartialSet(GFX_GOL_PICTURECONTROL * pObject, uint16_t xOffset, uint16_t yOffset,
uint16_t partialWidth, uint16_t partialHeight);
```

Returns

None.

Description

GFX GOL picture control partial set.

This function sets the partial image parameters to be used in the object. This function allows usage of the object to specify a rectangular area of an image to be drawn as part of the object. This is useful when an image is already included in a larger image. To save memory, a separate image is not necessary for the picture object. The location of the smaller image in the larger image can be specified to show up in the picture object.

This function will result in an undefined behavior when one of the following is true:

- xOffset - value must not be greater than the image width.
- yOffset - value must not be greater than the image height.
- partialWidth - value must not be greater than image width - xoffset + 1. Value must also be less than the actual image width.
- partialHeight - value must not be greater than image height - yoffset + 1. Value must also be less than the actual image height.

Preconditions

Object must exist in memory. The image pointer of the object must be initialized properly.

Example

```
// assume pLargeImage is a valid GFX_RESOURCE_HDR
// assume BigImage has a height and width of 100 pixels.

GFX_RESOURCE_HDR *pLargeImage = &BigImage;
GFX_GOL_PICTURECONTROL  *pPicture;
uint16_t         width, height;
uint16_t         xOffset, yOffset;
uint16_t         objectWidth, objectHeight;

objectWidth  = 60  - 50; // 10 pixels
objectHeight = 120 - 90; // 30 pixels

// -1 is needed since the object dimension is inclusive
pPicture = GFX_GOL_PictureControlCreate(
                    gfxIndex,
                    10,
                    50, 90,
                    50 + objectWidth - 1,
                    90 + objectHeight - 1,
```

```
                        GFX_GOL_PICTURECONTROL_DRAW_STATE,
                        largeImage,
                        NULL);

    // set the parameters of the partial image to be
    // shown on the image

    // get the large image dimensions
    width  = GFX_ImageWidthGet(pLargeImage);
    height = GFX_ImageHeightGet(pLargeImage);

    // get the offset so the middle of the large image with
    // the width and height matching the object will be used.
    xOffset = (width  - objectWidth)  >> 1;
    yOffset = (height - objectHeight) >> 1;

    GFX_GOL_PictureControlPartialSet(  pPicture,
                              xOffset, yOffset,
                              objectWidth, objectHeight);
```

Parameters

Parameters	Description
xOffset	x offset of the smaller portion of a large image
yOffset	y offset of the smaller portion of a large image
partialWidth	width of the selected portion of the image
partialHeight	height of the selected portion of the image

Function

void GFX_GOL_PictureControlPartialSet(

GFX_GOL_PICTURECONTROL *pObject,

uint16_t xOffset,

uint16_t yOffset,

uint16_t partialWidth,

uint16_t partialHeight)

GFX_GOL_PictureControlScaleSet Function

Sets the scale factor used to render the image used in the object.

File

gfx_gol_picture.h

C

```
void GFX_GOL_PictureControlScaleSet(GFX_GOL_PICTURECONTROL * pObject, int8_t scale);
```

Returns

None.

Description

GFX GOL picture control scale set.

This function sets the scale factor used to render the image used in the object using scale.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object
scale	Scale factor that will be used to display the image.

Function

void GFX_GOL_PictureControlScaleSet(
GFX_GOL_PICTURECONTROL pObject,
int8_t scale)

Progress Bar Object

Progress Bar is an object that can be used to display the progress of a task such as a data download or transfer.

Functions

	Name	Description
	GFX_GOL_ProgressBarActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ProgressBarCreate	This function creates a GFX_GOL_PROGRESSBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ProgressBarDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ProgressBarPositionSet	This function sets the position of the progress bar.
	GFX_GOL_ProgressBarRangeSet	This function sets the range of the progress bar.

Macros

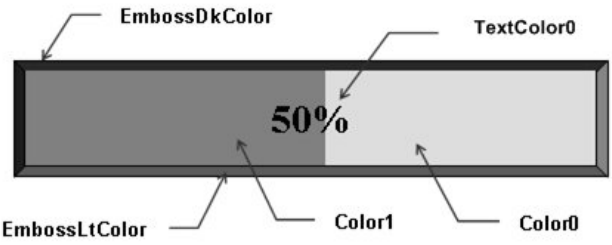
	Name	Description
	GFX_GOL_ProgressBarPositionGet	This function returns the current position of the progress bar.
	GFX_GOL_ProgressBarRangeGet	This function returns the range of the progress bar.

Description

Progress Bar is an object that can be used to display the progress of a task such as a data download or transfer.

Progress Bar supports only Touchscreen inputs, replying to the events with the predefined actions (see GFX_GOL_ProgressBarActionGet() for details).

The Progress Bar object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the object.



CommonBkColor – used to hide/remove the progress bar from the screen.

GFX_GOL_ProgressBarPositionGet Macro

This function returns the current position of the progress bar.

File

gfx_gol_progress_bar.h

C

```
#define GFX_GOL_ProgressBarPositionGet(pObject) \
                                       (((GFX_GOL_PROGRESSBAR*)pObject)->pos)
```

Returns

The current position of the scroll bar thumb.

Description

GFX GOL progress bar position get.

This function returns the current position of the progress bar.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_ProgressBarPositionSet() example.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_ProgressBarPositionGet(
GFX_GOL_PROGRESSBAR *pObject)

GFX_GOL_ProgressBarRangeGet Macro

This function returns the range of the progress bar.

File

gfx_gol_progress_bar.h

C

```
#define GFX_GOL_ProgressBarRangeGet(pObject) (pObject->range)
```

Returns

The current range used by the object.

Description

GFX GOL progress bar range get.

This function returns the range of the progress bar.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_ProgressBarRangeGet(
GFX_GOL_PROGRESSBAR *pObject)

GFX_GOL_ProgressBarActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_progress_bar.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_ProgressBarActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_PROGRESSBAR_ACTION_SELECTED â€“ Object is selected
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL progress bar action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit Description
GFX_GOL_PROGRESSBAR_ACTION_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE, If events occurs and the x,y position falls in the area of the object. EVENT_MOVE
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_ProgressBarActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage);

GFX_GOL_ProgressBarCreate Function

This function creates a GFX_GOL_PROGRESSBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_progress_bar.h

C

```
GFX_GOL_PROGRESSBAR * GFX_GOL_ProgressBarCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left,
uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, uint16_t pos, uint16_t range,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL progress bar create.

This function creates a GFX_GOL_PROGRESSBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_RadioButtonCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pos > range
- range = 0
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
GFX_GOL_PROGRESSBAR *pPBar;
void CreateProgressBar()
{
    pPBar = GFX_GOL_ProgressBarCreate(
                    gfxIndex,
                    ID_PROGRESSBAR1,   // ID
                    50,90,270,140,      // dimension
                    PB_DRAW,            // Draw the object
                    25,                 // position
                    50,                 // set the range
                    NULL);              // use default GOL scheme

}
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pos	Defines the initial position of the progress.
range	This specifies the maximum value of the progress bar when the progress bar is at 100% position.
pScheme	Pointer to the style scheme used.

Function

GFX_GOL_PROGRESSBAR *GFX_GOL_ProgressBarCreate(
SYS_MODULE_INDEX gfxIndex,
uint16_t ID,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t state,
uint16_t pos,
uint16_t range,
GFX_GOL_OBJ_SCHEME *pScheme)

GFX_GOL_ProgressBarDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_progress_bar.h

C

```
GFX_STATUS GFX_GOL_ProgressBarDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL progress bar draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_ProgressBarDraw(void *pObject)

GFX_GOL_ProgressBarPositionSet Function

This function sets the position of the progress bar.

File

gfx_gol_progress_bar.h

C

```
void GFX_GOL_ProgressBarPositionSet(GFX_GOL_PROGRESSBAR * pObject, uint16_t position);
```

Returns

None.

Description

GFX GOL progress bar position set.

This function sets the position of the progress bar.

The value used for the position should be within the range set for the object.

Function will have an undefined behavior if the position is outside the range.

Preconditions

Object must exist in memory.

Example

```
        GFX_GOL_PROGRESSBAR *pPb;
        uint8_t  direction = 1;

// this code increments and decrements the progress bar by 1
// assume progress bar was created and initialized before
while (1)
        {
            if(direction)
            {
                if(pPb ->pos == pPb ->range)
                    direction = 0;
                else
                    GFX_GOL_ProgressBarPositionSet(
                        pPb,
                        GFX_GOL_ProgressBarPositionGet(pPb)+1);
            }
            else
            {
                if(pPb ->pos == 0)
                    direction = 1;
                else
                    GFX_GOL_ProgressBarPositionSet(
```

```
                        pPb,
                        GFX_GOL_ProgressBarPositionGet(pPb)-1);
            }
        }
```

Parameters

Parameters	Description
pObject	pointer to the object.
position	the new position of the scroll bar thumb.

Function

void GFX_GOL_ProgressBarPositionSet(
GFX_GOL_PROGRESSBAR *pObject,
uint16_t position)

GFX_GOL_ProgressBarRangeSet Function

This function sets the range of the progress bar.

File

gfx_gol_progress_bar.h

C

```
void GFX_GOL_ProgressBarRangeSet(GFX_GOL_PROGRESSBAR * pObject, uint16_t range);
```

Returns

None.

Description

GFX GOL progress bar range set.

This function sets the range of the progress bar. When the range is modified, object must be completely redrawn to reflect the change.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
range	new range of the object.

Function

void GFX_GOL_ProgressBarRangeSet(
GFX_GOL_PROGRESSBAR *pObject,
uint16_t range)

Radio Button Object

Radio Button is an object that can be used to offer set of choices to the user. Only one of the choices is selectable. Changing selection automatically removes the selection on the previous option.

Functions

	Name	Description
	GFX_GOL_RadioButtonCheckGet	This function returns the ID of the currently checked radio button in the group.
	GFX_GOL_RadioButtonActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_RadioButtonActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_RadioButtonCheckSet	This function sets the ID of the currently checked radio button in the group.
	GFX_GOL_RadioButtonDraw	This function renders the object on the screen based on the current state of the object.

	GFX_GOL_RadioButtonTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_RadioButtonTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_RadioButtonTextSet	This function sets the address of the current text string used by the object.
	GFX_GOL_RadioButtonCreate	This function creates a GFX_GOL_RADIOBUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_RadioButtonListCreate	This is function GFX_GOL_RadioButtonListCreate.

Macros

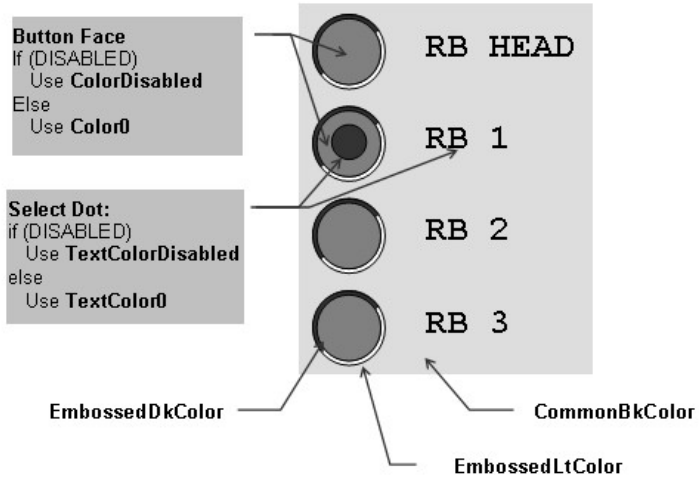
	Name	Description
	GFX_GOL_RadioButtonTextGet	This function returns the address of the current text string used by the object.

Description

Radio Button is an object that can be used to offer set of choices to the user. Only one of the choices is selectable. Changing selection automatically removes the selection on the previous option.

Radio Button supports Keyboard and Touchscreen inputs, replying to their events with the predefined actions (see GFX_GOL_RadioButtonActionGet() and GFX_GOL_RadioButtonActionSet() for details).

The Radio Button object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the object.



GFX_GOL_RadioButtonTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_radio_button.h

C

```
#define GFX_GOL_RadioButtonTextGet(pObject) \
                                    (((GFX_GOL_RADIOBUTTON *)pObject)->pText)
```

Returns

Pointer to text string.

Description

GFX GOL radio button text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
// assume RADIO_BUTTON_OBJECT is a radio button that exists
GFX_XCHAR *pChar;
```

```
GFX_GOL_RADIOBUTTON *pRadioButton = &RADIO_BUTTON_OBJECT;

pChar = GFX_GOL_ButtonRadioTextGet(pRadioButton);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_RadioButtonTextGet(
GFX_GOL_RADIOBUTTON *pObject)

GFX_GOL_RadioButtonCheckGet Function

This function returns the ID of the currently checked radio button in the group.

File

gfx_gol_radio_button.h

C

```
uint16_t GFX_GOL_RadioButtonCheckGet(GFX_GOL_RADIOBUTTON * pObject);
```

Returns

The ID of the member of the group with the check.

Description

GFX GOL radio button check get.

This function returns the ID of the currently checked radio button in the group. When there is only one member of the group, then that member will have the check.

When no member of the group is checked, then the id returned is (-1 or 0xFFFF).

Preconditions

Object must exist in memory.

Example

```
static  GFX_XCHAR   label0[] = "RB1";
static  GFX_XCHAR   label1[] = "RB2";
static  GFX_XCHAR   label2[] = "RB3";
        uint16_t    state;
        GFX_GOL_OBJ_SCHEME  *pScheme;
        RADIOBUTTON *pRb[3];
        uint16_t    ID;

pScheme = GFX_GOL_ObjectSchemeCreate();

// Object will be drawn after creation
// Object will be first button in the group
state = GFX_GOL_RADIOBUTTON_DRAW_STATE |
        GFX_GOL_RADIOBUTTON_CHECKED_STATE;

pRb[0] = GFX_GOL_RadioButtonCreate(
                  gfxIndex,
                  ID_RADIOBUTTON1,
                  255,40,310,80,
                  state,
                  label0,
                  GFX_ALIGN_CENTER,
                  pScheme);

// Object will be drawn after creation
state = GFX_GOL_RADIOBUTTON_DRAW_STATE;

pRb[1] = GFX_GOL_RadioButtonCreate(
                  gfxIndex
                  ID_RADIOBUTTON2,
```

```
                    255,85,310,125,
                    state,
                    label1,
                    GFX_ALIGN_CENTER,
                    pScheme);

    // Object will be drawn after creation
    state = GFX_GOL_RADIOBUTTON_DRAW_STATE;

    pRb[2] = GFX_GOL_RadioButtonCreate(
                    gfxIndex,
                    ID_RADIOBUTTON3,
                    255,130,310,170,
                    state,
                    label2,
                    GFX_ALIGN_CENTER,
                    pScheme);

    // draw the objects
    while(GFX_GOL_ObjectListDraw(gfxIndex) != GFX_STATUS_SUCCESS);

    // can also use pRb[1] or pRb[0] to search the checked
    // radio button of the group. ID here should be ID_RADIOBUTTON1
    ID = GFX_GOL_RadioButtonCheckGet(pRb[2]);

    if (ID == ID_RADIOBUTTON1)
    {
        // do something here then clear the check
GFX_GOL_ObjectStateClear(pRb[0], RB_CHECKED);
// Change the checked object. Pointer used is
        // any of the three. The ID used will find the
        // correct object to be checked
GFX_GOL_RadioButtonCheckSet(pRb[3], ID_RADIOBUTTON2);
    }
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_RadioButtonCheckGet(
GFX_GOL_RADIOBUTTON *pObject)

GFX_GOL_RadioButtonActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_radio_button.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_RadioButtonActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_RADIOBUTTON_ACTION_CHECKED - Radio Button is checked
- GFX_GOL_OBJECT_ACTION_INVALID - object is not affected

Description

GFX GOL radio button action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_RADIOBUTTON_ACTION_CHECKED	Touch Screen	EVENT_PRESS	If event occurs and the x,y position falls in the area of the Radio Button while the Radio Button is not checked.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_CR_PRESSED or SCAN_SPACE_PRESSED while the Radio Button is not checked.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_RadioButtonActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_RadioButtonActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_radio_button.h

C

```
void GFX_GOL_RadioButtonActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

GFX GOL radio button action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit Description
GFX_GOL_RADIOBUTTON_ACTION_CHECKED	Touch Screen,	Set GFX_GOL_RADIOBUTTON_DRAW_STATE, Depending on the current value of RB_CHECKED Check Box will be redrawn.
	Keyboard	Set/Clear GFX_GOL_RADIOBUTTON_CHECKED_STATE

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_RadioButtonActionSet(
GFX_GOL_TRANSLATED_ACTION translatedMsg,
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_RadioButtonCheckSet Function

This function sets the ID of the currently checked radio button in the group.

File

gfx_gol_radio_button.h

C

```
void GFX_GOL_RadioButtonCheckSet(GFX_GOL_RADIOBUTTON * pObject, uint16_t id);
```

Returns

None.

Description

GFX GOL radio button check set.

This function sets the ID of the currently checked radio button in the group. When there is only one member of the group, then that member will have the check.

When the id given does not exist in the group, the function will do nothing.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_RadioButtonCheckGet() example.

Parameters

Parameters	Description
pObject	pointer to the object.
id	id of the member of the group.

Function

GFX_XCHAR GFX_GOL_RadioButtonCheckSet(
GFX_GOL_RADIOBUTTON *pObject,
uint16_t id)

GFX_GOL_RadioButtonDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_radio_button.h

C

```
GFX_STATUS GFX_GOL_RadioButtonDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL radio button draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the GFX_GOL_RADIOBUTTON is drawn with the text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_RadioButtonDraw(void *pObject)

GFX_GOL_RadioButtonTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_radio_button.h

C

```
GFX_ALIGNMENT GFX_GOL_RadioButtonTextAlignmentGet(GFX_GOL_RADIOBUTTON * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL radio button text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_RadioButtonTextAlignmentGet(
GFX_GOL_RADIOBUTTON *pObject)

GFX_GOL_RadioButtonTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_radio_button.h

C

```
void GFX_GOL_RadioButtonTextAlignmentSet(GFX_GOL_RADIOBUTTON * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

GFX GOL radio button text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_RadioButtonTextAlignmentSet(

GFX_GOL_RADIOBUTTON *pObject,

GFX_ALIGNMENT align)

GFX_GOL_RadioButtonTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_radio_button.h

C

```
void GFX_GOL_RadioButtonTextSet(GFX_GOL_RADIOBUTTON * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

GFX GOL radio button text set.

This function sets the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR Label0[] = â€œONâ€?;
GFX_XCHAR Label1[] = â€œOFFâ€?;
GFX_GOL_RADIOBUTTON GFX_GOL_RADIOBUTTON[2];

GFX_GOL_RadioButtonTextSet(GFX_GOL_RADIOBUTTON[0], Label0);
GFX_GOL_RadioButtonTextSet(GFX_GOL_RADIOBUTTON[1], Label1);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be used.

Function

GFX_XCHAR *GFX_GOL_RadioButtonTextSet(

GFX_GOL_RADIOBUTTON *pObject,
GFX_XCHAR *pText)

GFX_GOL_RadioButtonCreate Function

This function creates a GFX_GOL_RADIOBUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_radio_button.h

C

```
GFX_GOL_RADIOBUTTON * GFX_GOL_RadioButtonCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left,
uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL Radio button create.

This function creates a GFX_GOL_RADIOBUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_RadioButtonCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- pText is an unterminated string

Preconditions

None.

Example

```
static  GFX_XCHAR   label0[] = "RB1";
static  GFX_XCHAR   label1[] = "RB2";
static  GFX_XCHAR   label2[] = "RB3";
        uint16_t    state;
        GFX_GOL_OBJ_SCHEME  *pScheme;
        RADIOBUTTON *pRb[3];

pScheme = GFX_GOL_ObjectSchemeCreate();

// Object will be drawn after creation
// Object will be first button in the group
state = GFX_GOL_RADIOBUTTON_DRAW_STATE |
        GFX_GOL_RADIOBUTTON_CHECKED_STATE;

pRb[0] = GFX_GOL_RadioButtonCreate(
                  gfxIndex,
                  ID_RADIOBUTTON1,
                  255,40,310,80,
                  state,
                  label0,
                  GFX_ALIGN_CENTER,
                  pScheme);

// Object will be drawn after creation
state = GFX_GOL_RADIOBUTTON_DRAW_STATE;

pRb[1] = GFX_GOL_RadioButtonCreate(
                  gfxIndex,
                  ID_RADIOBUTTON2,
                  255,85,310,125,
                  state,
                  label1,
```

```
                    GFX_ALIGN_CENTER,
                    pScheme);

    // Object will be drawn after creation
    state = GFX_GOL_RADIOBUTTON_DRAW_STATE;

    pRb[2] = GFX_GOL_RadioButtonCreate(
                    gfxIndex,
                    ID_RADIOBUTTON3,
                    255,130,310,170,
                    state,
                    label2,
                    GFX_ALIGN_CENTER,
                    pScheme);

    // draw the objects
    while(GFX_GOL_ObjectListDraw(gfxIndex) != GFX_STATUS_SUCCESS);
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_RADIOBUTTON *GFX_GOL_RadioButtonCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t        ID,
uint16_t        left,
uint16_t        top,
uint16_t        right,
uint16_t        bottom,
uint16_t        state,
                GFX_XCHAR       *pText,
                GFX_ALIGNMENT       alignment,
                GFX_GOL_OBJ_SCHEME  *pScheme)
```

GFX_GOL_RadioButtonListCreate Function

File

gfx_gol_radio_button.h

C

```
GFX_GOL_RADIOBUTTON * GFX_GOL_RadioButtonListCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left,
uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME * pScheme, uint16_t groupNo, uint16_t numOfGroups);
```

Description

This is function GFX_GOL_RadioButtonListCreate.

Scheme Object

Functions

	Name	Description
	GFX_GOL_SchemeBackgroundColorSet	This is function GFX_GOL_SchemeBackgroundColorSet.
	GFX_GOL_SchemeAlphaPrecentSet	This is function GFX_GOL_SchemeAlphaPrecentSet.
	GFX_GOL_SchemeBackgroundColorGet	This is function GFX_GOL_SchemeBackgroundColorGet.
	GFX_GOL_SchemeBackgroundImageSet	This is function GFX_GOL_SchemeBackgroundImageSet.
	GFX_GOL_SchemeBackgroundTypeGet	This is function GFX_GOL_SchemeBackgroundTypeGet.
	GFX_GOL_SchemeBackgroundTypeSet	This is function GFX_GOL_SchemeBackgroundTypeSet.
	GFX_GOL_SchemeColor0Get	This is function GFX_GOL_SchemeColor0Get.
	GFX_GOL_SchemeColor0Set	This is function GFX_GOL_SchemeColor0Set.
	GFX_GOL_SchemeColor1Get	This is function GFX_GOL_SchemeColor1Get.
	GFX_GOL_SchemeColor1Set	This is function GFX_GOL_SchemeColor1Set.
	GFX_GOL_SchemeColorDisabledGet	This is function GFX_GOL_SchemeColorDisabledGet.
	GFX_GOL_SchemeColorDisabledSet	This is function GFX_GOL_SchemeColorDisabledSet.
	GFX_GOL_SchemeColorSet	This is function GFX_GOL_SchemeColorSet.
	GFX_GOL_SchemeEmbossDarkColorGet	This is function GFX_GOL_SchemeEmbossDarkColorGet.
	GFX_GOL_SchemeEmbossDarkColorSet	This is function GFX_GOL_SchemeEmbossDarkColorSet.
	GFX_GOL_SchemeEmbossLightColorGet	This is function GFX_GOL_SchemeEmbossLightColorGet.
	GFX_GOL_SchemeEmbossLightColorSet	This is function GFX_GOL_SchemeEmbossLightColorSet.
	GFX_GOL_SchemeEmbossSet	This is function GFX_GOL_SchemeEmbossSet.
	GFX_GOL_SchemeEmbossSizeGet	This is function GFX_GOL_SchemeEmbossSizeGet.
	GFX_GOL_SchemeEmbossSizeSet	This is function GFX_GOL_SchemeEmbossSizeSet.
	GFX_GOL_SchemeFillStyleGet	This is function GFX_GOL_SchemeFillStyleGet.
	GFX_GOL_SchemeFillStyleSet	This is function GFX_GOL_SchemeFillStyleSet.
	GFX_GOL_SchemeFontGet	This is function GFX_GOL_SchemeFontGet.
	GFX_GOL_SchemeFontSet	This is function GFX_GOL_SchemeFontSet.
	GFX_GOL_SchemeGradientColorSet	This is function GFX_GOL_SchemeGradientColorSet.
	GFX_GOL_SchemeGradientEndColorGet	This is function GFX_GOL_SchemeGradientEndColorGet.
	GFX_GOL_SchemeGradientEndColorSet	This is function GFX_GOL_SchemeGradientEndColorSet.
	GFX_GOL_SchemeGradientStartColorGet	This is function GFX_GOL_SchemeGradientStartColorGet.
	GFX_GOL_SchemeGradientStartColorSet	This is function GFX_GOL_SchemeGradientStartColorSet.
	GFX_GOL_SchemeTextColor0Get	This is function GFX_GOL_SchemeTextColor0Get.
	GFX_GOL_SchemeTextColor0Set	This is function GFX_GOL_SchemeTextColor0Set.
	GFX_GOL_SchemeTextColor1Get	This is function GFX_GOL_SchemeTextColor1Get.
	GFX_GOL_SchemeTextColor1Set	This is function GFX_GOL_SchemeTextColor1Set.
	GFX_GOL_SchemeTextColorDisableGet	This is function GFX_GOL_SchemeTextColorDisableGet.
	GFX_GOL_SchemeTextColorDisableSet	This is function GFX_GOL_SchemeTextColorDisableSet.
	GFX_GOL_SchemeTextColorSet	This is function GFX_GOL_SchemeTextColorSet.

Description

GFX_GOL_SchemeBackgroundColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeBackgroundColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR color);
```

Description

This is function GFX_GOL_SchemeBackgroundColorSet.

GFX_GOL_SchemeAlphaPrecentSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeAlphaPrecentSet(GFX_GOL_OBJ_SCHEME * pScheme, uint16_t percent);
```

Description

This is function GFX_GOL_SchemeAlphaPrecentSet.

GFX_GOL_SchemeBackgroundColorGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeBackgroundColorGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeBackgroundColorGet.

GFX_GOL_SchemeBackgroundImageSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeBackgroundImageSet(GFX_GOL_OBJ_SCHEME * pScheme, uint16_t left, uint16_t top,
GFX_RESOURCE_HDR * image);
```

Description

This is function GFX_GOL_SchemeBackgroundImageSet.

GFX_GOL_SchemeBackgroundTypeGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_BACKGROUND_TYPE GFX_GOL_SchemeBackgroundTypeGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeBackgroundTypeGet.

GFX_GOL_SchemeBackgroundTypeSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeBackgroundTypeSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_BACKGROUND_TYPE type);
```

Description

This is function GFX_GOL_SchemeBackgroundTypeSet.

GFX_GOL_SchemeColor0Get Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeColor0Get(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeColor0Get.

GFX_GOL_SchemeColor0Set Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeColor0Set(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR color);
```

Description

This is function GFX_GOL_SchemeColor0Set.

GFX_GOL_SchemeColor1Get Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeColor1Get(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeColor1Get.

GFX_GOL_SchemeColor1Set Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeColor1Set(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR color);
```

Description

This is function GFX_GOL_SchemeColor1Set.

GFX_GOL_SchemeColorDisabledGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeColorDisabledGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeColorDisabledGet.

GFX_GOL_SchemeColorDisabledSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeColorDisabledSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR disableColor);
```

Description

This is function GFX_GOL_SchemeColorDisabledSet.

GFX_GOL_SchemeColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR color0, GFX_COLOR color1,
```

```
GFX_COLOR disableColor);
```

Description

This is function GFX_GOL_SchemeColorSet.

GFX_GOL_SchemeEmbossDarkColorGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeEmbossDarkColorGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeEmbossDarkColorGet.

GFX_GOL_SchemeEmbossDarkColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeEmbossDarkColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR darkColor);
```

Description

This is function GFX_GOL_SchemeEmbossDarkColorSet.

GFX_GOL_SchemeEmbossLightColorGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeEmbossLightColorGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeEmbossLightColorGet.

GFX_GOL_SchemeEmbossLightColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeEmbossLightColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR lightColor);
```

Description

This is function GFX_GOL_SchemeEmbossLightColorSet.

GFX_GOL_SchemeEmbossSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeEmbossSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR darkColor, GFX_COLOR
lightColor, uint16_t size);
```

Description

This is function GFX_GOL_SchemeEmbossSet.

GFX_GOL_SchemeEmbossSizeGet Function

File

gfx_gol_scheme.h

C

```
inline uint16_t GFX_GOL_SchemeEmbossSizeGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeEmbossSizeGet.

GFX_GOL_SchemeEmbossSizeSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeEmbossSizeSet(GFX_GOL_OBJ_SCHEME * pScheme, uint16_t size);
```

Description

This is function GFX_GOL_SchemeEmbossSizeSet.

GFX_GOL_SchemeFillStyleGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_FILL_STYLE GFX_GOL_SchemeFillStyleGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeFillStyleGet.

GFX_GOL_SchemeFillStyleSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeFillStyleSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_FILL_STYLE style);
```

Description

This is function GFX_GOL_SchemeFillStyleSet.

GFX_GOL_SchemeFontGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_RESOURCE_HDR* GFX_GOL_SchemeFontGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeFontGet.

GFX_GOL_SchemeFontSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeFontSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_RESOURCE_HDR * font);
```

Description

This is function GFX_GOL_SchemeFontSet.

GFX_GOL_SchemeGradientColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeGradientColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR startColor, GFX_COLOR endColor);
```

Description

This is function GFX_GOL_SchemeGradientColorSet.

GFX_GOL_SchemeGradientEndColorGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeGradientEndColorGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeGradientEndColorGet.

GFX_GOL_SchemeGradientEndColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeGradientEndColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR endColor);
```

Description

This is function GFX_GOL_SchemeGradientEndColorSet.

GFX_GOL_SchemeGradientStartColorGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeGradientStartColorGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeGradientStartColorGet.

GFX_GOL_SchemeGradientStartColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeGradientStartColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR startColor);
```

Description

This is function GFX_GOL_SchemeGradientStartColorSet.

GFX_GOL_SchemeTextColor0Get Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeTextColor0Get(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeTextColor0Get.

GFX_GOL_SchemeTextColor0Set Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeTextColor0Set(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR textColor);
```

Description

This is function GFX_GOL_SchemeTextColor0Set.

GFX_GOL_SchemeTextColor1Get Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeTextColor1Get(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeTextColor1Get.

GFX_GOL_SchemeTextColor1Set Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeTextColor1Set(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR textColor);
```

Description

This is function GFX_GOL_SchemeTextColor1Set.

GFX_GOL_SchemeTextColorDisableGet Function

File

gfx_gol_scheme.h

C

```
inline GFX_COLOR GFX_GOL_SchemeTextColorDisableGet(GFX_GOL_OBJ_SCHEME * pScheme);
```

Description

This is function GFX_GOL_SchemeTextColorDisableGet.

GFX_GOL_SchemeTextColorDisableSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeTextColorDisableSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR disableColor);
```

Description

This is function GFX_GOL_SchemeTextColorDisableSet.

GFX_GOL_SchemeTextColorSet Function

File

gfx_gol_scheme.h

C

```
inline void GFX_GOL_SchemeTextColorSet(GFX_GOL_OBJ_SCHEME * pScheme, GFX_COLOR textColor0, GFX_COLOR
textColor1, GFX_COLOR disableColor);
```

Description

This is function GFX_GOL_SchemeTextColorSet.

Scroll Bar Object

Scroll Bar is an object that can be used to display a value or scrolling location in a predefined area.

Functions

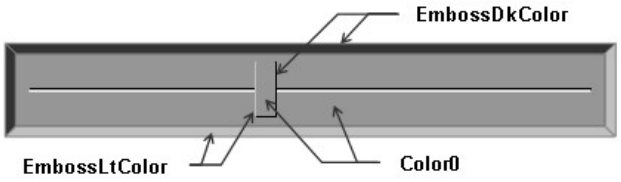
	Name	Description
	GFX_GOL_ScrollBarActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ScrollBarActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ScrollBarCreate	This function creates a GFX_GOL_SCROLLBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ScrollBarDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ScrollBarPositionDecrement	This function decrements the scroll bar position by the delta change (page) value set.
	GFX_GOL_ScrollBarPageGet	This function returns the page size of the object.
	GFX_GOL_ScrollBarPageSet	This function sets the page size of the object.
	GFX_GOL_ScrollBarPositionGet	This function returns the current position of the scroll bar thumb.
	GFX_GOL_ScrollBarPositionIncrement	This function increments the scroll bar position by the delta change (page) value set.
	GFX_GOL_ScrollBarPositionSet	This function sets the position of the scroll bar thumb.
	GFX_GOL_ScrollBarRangeGet	This function returns the range of the thumb of the scroll bar.
	GFX_GOL_ScrollBarRangeSet	This function sets the range of the thumb of the scroll bar.

Description

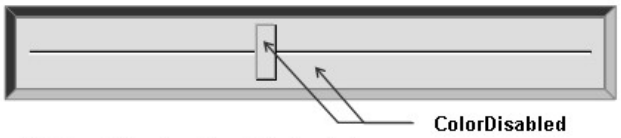
Scroll Bar is an object that can be used to display a value or scrolling location in a predefined area.

Scroll Bar supports Keyboard and Touchscreen inputs, replying to their events with the predefined actions (see GFX_GOL_ScrollBarActionGet() and GFX_GOL_ScrollBarActionSet() for details).

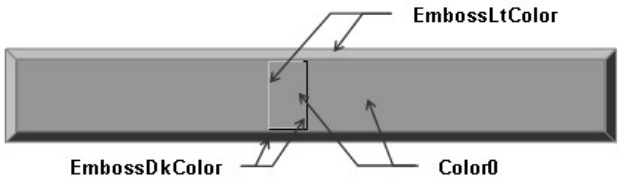
The Scroll Bar object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the object.



Slider Mode enabled state



Slider Mode disabled state



Scrollbar Mode enabled state

GFX_GOL_ScrollBarActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_scroll_bar.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_ScrollBarActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_SCROLLBAR_ACTION_INC â€“ Increment scroll bar position.
- GFX_GOL_SCROLLBAR_ACTION_DEC â€“ Decrement scroll bar position.
- GFX_GOL_OBJECT_ACTION_PASSIVE â€“ Object bar is not affected
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL scroll bar action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_SCROLLBAR_ACTION_INC	Touch Screen	EVENT_PRESS, EVENT_MOVE	If events occurs and the x,y position falls in the area of the slider and the slider position is to the LEFT of the x,y position for a horizontal slider or BELOW the x,y position for a vertical slider.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_UP_PRESSED or SCAN_LEFT_PRESSED.
GFX_GOL_SCROLLBAR_ACTION_DEC	Touch Screen	EVENT_PRESS, EVENT_MOVE	If events occurs and the x,y position falls in the area of the slider and the slider position is to the RIGHT of the x,y position for a horizontal slider or ABOVE the x,y position for a vertical slider.
	Keyboard	EVENT_KEYSCAN	If event occurs and parameter1 passed matches the object's ID and parameter 2 passed matches SCAN_DOWN_PRESSED or SCAN_RIGHT_PRESSED.
GFX_GOL_OBJECT_ACTION_PASSIVE	Touch Screen	EVENT_RELEASE	If events occurs and the x,y position falls in the area of the slider.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_ScrollBarActionGet(
void *pObject, GFX_GOL_MESSAGE *pMessage)

GFX_GOL_ScrollBarActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_scroll_bar.h

C

```
void GFX_GOL_ScrollBarActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL scroll bar action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_SCROLLBAR_ACTION_INC	Touch Screen,	Set GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE	Scroll Bar will be redrawn with thumb in the incremented position.
	Keyboard		
GFX_GOL_SCROLLBAR_ACTION_DEC	Touch Screen,	Set GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE	Scroll Bar will be redrawn with thumb in the decremented position.
	Keyboard		

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_ScrollBarActionSet(

GFX_GOL_TRANSLATED_ACTION translatedMsg,

void *pObject,

GFX_GOL_MESSAGE *pMessage)

GFX_GOL_ScrollBarCreate Function

This function creates a GFX_GOL_SCROLLBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_scroll_bar.h

C

```
GFX_GOL_SCROLLBAR * GFX_GOL_ScrollBarCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, uint16_t range, uint16_t page, uint16_t pos,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL scroll bar create.

This function creates a GFX_GOL_SCROLLBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The object can be configured as a scroll bar or a slider. Use the state bit GFX_GOL_SCROLLBAR_SLIDER_MODE_STATE to enable the usage of the object as a slider. If this state bit is not enabled, the object is set up as a scroll bar.

The object can also be configured with vertical orientation. Use the state bit GFX_GOL_SCROLLBAR_VERTICAL_STATE to set up the object with vertical orientation. If this state bit is not set, the object is used with horizontal orientation.

The behavior of GFX_GOL_ScrollBarCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- page is set to zero.
- range is set to zero.
- page > range.

Preconditions

None.

Example

```
// assume pScheme is initialized

GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_SCROLLBAR  *srollBarArray[3];
uint16_t           state;

// create a slider with
// range = [0 - 50000]
// delta = 500
// initial position = 25000

state = GFX_GOL_SCROLLBAR_DRAW_STATE;
srollBarArray[0] = GFX_GOL_ScrollBarCreate(
                        gfxIndex, 5,
                        150, 145, 285, 181,
                        state,
                        50000, 500, 25000,
                        pScheme);
if (slider[0] == NULL)
    return 0;

// create a slider with
// range = [0 - 100]
// delta = 20
// initial position = 0

state =  GFX_GOL_SCROLLBAR_DRAW_STATE |
         GFX_GOL_SCROLLBAR_SLIDER_MODE_STATE;
srollBarArray[1] = GFX_GOL_ScrollBarCreate(
                        gfxIndex, 6,
                        150, 190, 285, 220,
                        state,
                        100, 20, 0,
                        pScheme);
if (slider[1] == NULL)
    return 0;

// create a vertical scroll bars with
// range = [0 - 30]
// delta = 2
// initial position = 20

state =  GFX_GOL_SCROLLBAR_DRAW_STATE |
         GFX_GOL_SCROLLBAR_VERTICAL_STATE;
srollBarArray[2] = GFX_GOL_ScrollBarCreate(
                        gfxIndex, 7,
                        120, 145, 140, 220,
                        state,
                        30, 2, 20,
```

```
                            pScheme);
    if (slider[2] == NULL)
        return 0;

    // draw the sliders
    while (GFX_GOL_ObjectListDraw() == 0);

    return 1;
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
range	This specifies the maximum value of the slider when the thumb is on the rightmost position for a horizontal orientation and bottom most position for the vertical orientation. Minimum value is always at zero.
page	This is the incremental change of the slider when user action requests to move the slider thumb. This value is added or subtracted to the current position of the thumb.
pos	This defines the initial position of the thumb.
pScheme	The pointer to the style scheme used for the Object. Set to NULL if default style scheme is used.

Function

GFX_GOL_SCROLLBAR *GFX_GOL_ScrollBarCreate(
SYS_MODULE_INDEX gfxIndex,
uint16_t ID,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t state,
uint16_t range,
uint16_t page,
uint16_t pos,
GFX_GOL_OBJ_SCHEME *pScheme);

GFX_GOL_ScrollBarDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_scroll_bar.h

C

```
GFX_STATUS GFX_GOL_ScrollBarDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL scroll bar draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the face of the GFX_GOL_SCROLLBAR is drawn on top of the bitmap. Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_ScrollBarDraw(void *pObject)

GFX_GOL_ScrollBarPositionDecrement Function

This function decrements the scroll bar position by the delta change (page) value set.

File

gfx_gol_scroll_bar.h

C

```
void GFX_GOL_ScrollBarPositionDecrement(GFX_GOL_SCROLLBAR * pObject);
```

Returns

None.

Description

GFX GOL scroll bar position decrement.

This function decrements the scroll bar position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.

Preconditions

Object must exist in memory.

Example

```
void ControlSpeed(  GFX_GOL_SCROLLBAR* pObj,
                    int setSpeed,
                    int curSpeed)
{
    // set page size to 1
    GFX_GOL_ScrollBarPageSet(pObj, 1);

    if (setSpeed < curSpeed)
    {
        while(GFX_GOL_ScrollBarPositionGet(pObj) < SetSpeed)
        {
            // increment by 1
            GFX_GOL_ScrollBarPositionIncrement(pObj);
        }

    }
    else if (setSpeed > curSpeed)
    {
        while(GFX_GOL_ScrollBarPositionGet(pObj) > SetSpeed)
        {
            // decrement by 1
            GFX_GOL_ScrollBarPositionDecrement(pObj);
        }
    }
}
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_ScrollBarPositionDecrement(
GFX_GOL_SCROLLBAR *pObject)

GFX_GOL_ScrollBarPageGet Function

This function returns the page size of the object.

File

gfx_gol_scroll_bar.h

C

```
uint16_t GFX_GOL_ScrollBarPageGet(GFX_GOL_SCROLLBAR * pObject);
```

Returns

The page size of the object.

Description

GFX GOL scroll bar page get.

This function returns the page size of the object. Page size defines the delta change of the thumb position when incremented via GFX_GOL_ScrollBarPositionIncrement() or decremented via GFX_GOL_ScrollBarPositionDecrement(). Page size minimum value is 1. Maximum value is range/2.

Preconditions

Object must exist in memory.

Example

```
uint16_t page;
GFX_GOL_SCROLLBAR *pScrollBar;

// assume pScrollBar is initialized to a scroll bar in memory
 page = GFX_GOL_ScrollBarPageGet(pScrollBar);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_ScrollBarPageGet(
GFX_GOL_SCROLLBAR *pObject)

GFX_GOL_ScrollBarPageSet Function

This function sets the page size of the object.

File

gfx_gol_scroll_bar.h

C

```
void GFX_GOL_ScrollBarPageSet(GFX_GOL_SCROLLBAR * pObject, uint16_t page);
```

Returns

None.

Description

GFX GOL scroll bar page set.

This function sets the page size of the object. Page size defines the delta change of the thumb position when incremented via GFX_GOL_ScrollBarPositionIncrement() or decremented via GFX_GOL_ScrollBarPositionDecrement(). Page size minimum value is 1. Maximum value is range/2.

Modifying the page size at run time may require a redraw of the object to show the visual effect of the change.

Preconditions

Object must exist in memory.

Example

See GFX_GOL_ScrollBarPositionIncrement() for code example.

Parameters

Parameters	Description
pObject	pointer to the object.
page	value of the page size of the object.

Function

void GFX_GOL_ScrollBarPageSet(
GFX_GOL_SCROLLBAR *pObject,
uint16_t page)

GFX_GOL_ScrollBarPositionGet Function

This function returns the current position of the scroll bar thumb.

File

gfx_gol_scroll_bar.h

C

```
uint16_t GFX_GOL_ScrollBarPositionGet(GFX_GOL_SCROLLBAR * pObject);
```

Returns

The current position of the scroll bar thumb.

Description

GFX GOL scroll bar position get.

This function returns the current position of the scroll bar thumb. The thumb is the rectangular area that slides left or right (for horizontal orientation) or slides up or down (for vertical orientation).

Preconditions

Object must exist in memory.

Example

```
#define MAXVALUE 100;

GFX_GOL_SCROLLBAR *pScrollBar;
uint32_t ctr = 0;

// create scroll bar here and initialize parameters
pScrollBar = GFX_GOL_ScrollBarCreate(....)
GFX_GOL_ObjectStateSet(pScrollBar, GFX_GOL_SCROLLBAR_DRAW_STATE);

// draw the scroll bar
GFX_GOL_ObjectListDraw(gfxIndex);

// a routine that updates the position of the thumb through some
// conditions
while("some condition")
{
    GFX_GOL_ScrollBarPositionSet(pScrollBar, ctr);
    GFX_GOL_ObjectStateSet( pScrollBar,
                            GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE);

    // update the screen
    GFX_GOL_ObjectListDraw(gfxIndex);

    // update ctr here
    ctr = "some source of value";
```

```
}

if (GFX_GOL_ScrollBarPositionGet(pScrollBar) > MAXVALUE)
    return 0;
else
    "do something else"
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_ScrollBarPositionGet(
GFX_GOL_SCROLLBAR *pObject)

GFX_GOL_ScrollBarPositionIncrement Function

This function increments the scroll bar position by the delta change (page) value set.

File

gfx_gol_scroll_bar.h

C

```
void GFX_GOL_ScrollBarPositionIncrement(GFX_GOL_SCROLLBAR * pObject);
```

Returns

None.

Description

GFX GOL scroll bar position increment.

This function increments the scroll bar position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.

Preconditions

Object must exist in memory.

Example

```
void ControlSpeed(  GFX_GOL_SCROLLBAR* pObj,
                    int setSpeed,
                    int curSpeed)
{
    // set page size to 1
    GFX_GOL_ScrollBarPageSet(pObj, 1);

    if (setSpeed < curSpeed)
    {
        while(GFX_GOL_ScrollBarPositionGet(pObj) < SetSpeed)
            GFX_GOL_ScrollBarPositionIncrement(pObj);  // increment by 1
    }
    else if (setSpeed > curSpeed)
    {
        while(GFX_GOL_ScrollBarPositionGet(pObj) > SetSpeed)
            GFX_GOL_ScrollBarPositionDecrement(pObj);  // decrement by 1
    }
}
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_ScrollBarPositionIncrement(
GFX_GOL_SCROLLBAR *pObject)

GFX_GOL_ScrollBarPositionSet Function

This function sets the position of the scroll bar thumb.

File

gfx_gol_scroll_bar.h

C

```
void GFX_GOL_ScrollBarPositionSet(GFX_GOL_SCROLLBAR * pObject, uint16_t position);
```

Returns

None.

Description

GFX GOL scroll bar position set.

This function sets the position of the scroll bar thumb. The thumb is the rectangular area that slides left or right (for horizontal orientation) or slides up or down (for vertical orientation).

The value used for the position should be within the range set for the object.

Function will have an undefined behavior if the position is outside the range.

Preconditions

Object must exist in memory.

Example

```
GFX_GOL_SCROLLBAR *pScrollBar;
uint16_t ctr = 0;

// create slider here and initialize parameters
GFX_GOL_ObjectStateSet(pScrollBar, GFX_GOL_SCROLLBAR_DRAW_STATE);
GFX_GOL_ObjectListDraw(gfxIndex);

while("some condition")
{
    GFX_GOL_ScrollBarPositionSet(pScrollBar, ctr);
    GFX_GOL_ObjectStateSet( pScrollBar,
                            GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE);

    // redraw the scroll bar
    GFX_GOL_ObjectListDraw(gfxIndex);

   // update ctr here
   ctr = "some source of value";
}
```

Parameters

Parameters	Description
pObject	pointer to the object.
position	the new position of the scroll bar thumb.

Function

void GFX_GOL_ScrollBarPositionSet(

GFX_GOL_SCROLLBAR *pObject,

uint16_t position)

GFX_GOL_ScrollBarRangeGet Function

This function returns the range of the thumb of the scroll bar.

File

gfx_gol_scroll_bar.h

C

```
uint16_t GFX_GOL_ScrollBarRangeGet(GFX_GOL_SCROLLBAR * pObject);
```

Returns

The range of the scroll bar.

Description

GFX GOL scroll bar range get.

This function returns the range of the thumb of the scroll bar.

Preconditions

Object must exist in memory.

Example

```
GFX_GOL_SCROLLBAR *pSld;
uint16_t range;

range = GFX_GOL_ScrollBarRangeGet(pSld);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

uint16_t GFX_GOL_ScrollBarRangeGet(
GFX_GOL_SCROLLBAR *pObject)

GFX_GOL_ScrollBarRangeSet Function

This function sets the range of the thumb of the scroll bar.

File

gfx_gol_scroll_bar.h

C

```
void GFX_GOL_ScrollBarRangeSet(GFX_GOL_SCROLLBAR * pObject, uint16_t range);
```

Returns

None.

Description

GFX GOL scroll bar range set.

This function sets the range of the thumb of the scroll bar. When the range is modified, object must be completely redrawn to reflect the change.

Preconditions

Object must exist in memory.

Example

```
GFX_GOL_SCROLLBAR *pSld;

GFX_GOL_ScrollBarRangeSet(pSld, 100);

// to completely redraw the object when
// GFX_GOL_ObjectListDraw(gfxIndex) is executed.
GFX_GOL_ObjectStateSet(pSld, SLD_DRAW);
```

Parameters

Parameters	Description
pObject	pointer to the object.
range	new range of the scroll bar.

Function

void GFX_GOL_ScrollBarRangeSet(
GFX_GOL_SCROLLBAR *pObject,
uint16_t range)

Static Text Object

Static Text is an object that can be used to display a single or multi-line string of text in a defined area.

Functions

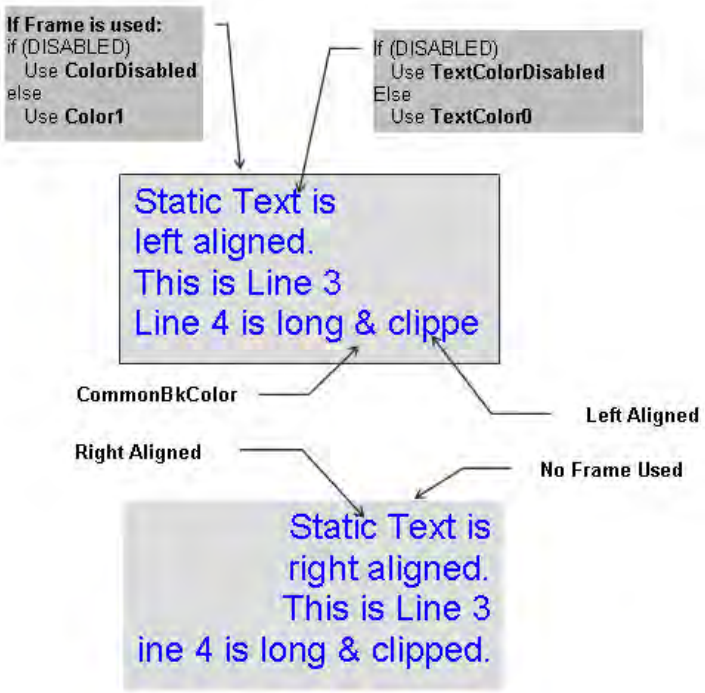
	Name	Description
	GFX_GOL_StaticTextActionGet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_StaticTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_StaticTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_StaticTextCreate	This function creates a GFX_GOL_STATICTEXT object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_StaticTextDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_StaticTextGet	This function returns the address of the current text string used by the object.
	GFX_GOL_StaticTextSet	This function sets the address of the current text string used by the object.

Description

Static Text is an object that can be used to display a single or multi-line string of text in a rectangular area defined by the dimension of the object. The area defined will also serve as the writable region, where any pixels that exceeds the area's dimension will be clipped.

Static Text supports Touchscreen inputs only, replying to the events with the predefined actions (see GFX_GOL_StaticTextActionGet() for details).

The Static Text object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the object.



GFX_GOL_StaticTextActionGet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_static_text.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_StaticTextActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_STATICTEXT_ACTION_SELECTED â€“ Object is selected
- GFX_GOL_OBJECT_ACTION_INVALID â€“ Object is not affected

Description

GFX GOL static text action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen inputs.

Translated Message	Input Source	Events	Description
GFX_GOL_STATICTEXT_ACTION_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If events occurs and the x,y position falls in the area of the static text.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_StaticTextActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage);

GFX_GOL_StaticTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_static_text.h

C

```
GFX_ALIGNMENT GFX_GOL_StaticTextAlignmentGet(GFX_GOL_STATICTEXT * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL static text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_StaticTextAlignmentGet(
GFX_GOL_STATICTEXT *pObject)

GFX_GOL_StaticTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_static_text.h

C

```
void GFX_GOL_StaticTextAlignmentSet(GFX_GOL_STATICTEXT * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

GFX GOL static text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
align	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_StaticTextAlignmentSet(

GFX_GOL_STATICTEXT *pObject,

GFX_ALIGNMENT align)

GFX_GOL_StaticTextCreate Function

This function creates a GFX_GOL_STATICTEXT object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_static_text.h

C

```
GFX_GOL_STATICTEXT * GFX_GOL_StaticTextCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left,
uint16_t top, uint16_t right, uint16_t bottom, uint16_t state, GFX_XCHAR * pText, GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL static text create.

This function creates a GFX_GOL_STATICTEXT object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The text can be configured to have text aligned. See GFX_ALIGNMENT for details. Text can also have multiple lines by inserting the new line character to the text string supplied to the object. Any string that exceeds the dimension of the object will be clipped.

When the object is used with no background, application must manage the object when text is modified and redrawn. i.e. the previous text must be removed. Use GFX_GOL_STATICTEXT_NOBACKGROUND_STATE state bit to disable the background.

The behavior of GFX_GOL_StaticTextCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME

- pText is an unterminated string

Preconditions

None.

Example

```
#define ID_STATICTEXT1 0x10

// assume pScheme is initialized
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_STATICTEXT *pSt;

pSt = GFX_GOL_StaticTextCreate(
    gfxIndex,
    ID_STATICTEXT1,                 // ID
    30,80,235,160,                  // dimension
    GFX_GOL_STATICTEXT_DRAW_STATE,  // draw the object
    "Static Textn Example",        // 2 lines of text
    GFX_ALIGN_CENTER,               // align text on the center
    pScheme);                       // use given scheme
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_STATICTEXT  *GFX_GOL_StaticTextCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t        ID,
uint16_t        left,
uint16_t        top,
uint16_t        right,
uint16_t        bottom,
uint16_t        state,
                GFX_XCHAR       *pText,
                GFX_ALIGNMENT       alignment,
                GFX_GOL_OBJ_SCHEME  *pScheme)
```

GFX_GOL_StaticTextDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_static_text.h

C

```
GFX_STATUS GFX_GOL_StaticTextDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL static text draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_StaticTextDraw(void *pObject)

GFX_GOL_StaticTextGet Function

This function returns the address of the current text string used by the object.

File

gfx_gol_static_text.h

C

```
GFX_XCHAR * GFX_GOL_StaticTextGet(GFX_GOL_STATICTEXT * pObject);
```

Returns

Pointer to text string.

Description

GFX GOL static text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR *pChar;
GFX_GOL_STATICTEXT OBJECT_ARRAY[2];

pChar = GFX_GOL_StaticTextGet(OBJECT_ARRAY[0]);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_StaticTextGet(
GFX_GOL_STATICTEXT *pObject)

GFX_GOL_StaticTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_static_text.h

C

```
void GFX_GOL_StaticTextSet(GFX_GOL_STATICTEXT * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

GFX GOL static text set.

This function sets the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be used.

Function

GFX_XCHAR *GFX_GOL_StaticTextSet(
GFX_GOL_STATICTEXT *pObject,
GFX_XCHAR *pText)

Surface Object

Functions

	Name	Description
	GFX_GOL_SurfaceActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_SurfaceCreate	This function creates a GFX_GOL_SURFACE object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_SurfaceDraw	This function renders the object on the screen based on the current state of the object.

Description

GFX_GOL_SurfaceActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_surface.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_SurfaceActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_SURFACE_ACTION_CLIENT - Surface client area selected action ID.
- GFX_GOL_SURFACE_ACTION_TITLE - Surface title bar selected action ID.

Description

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
GFX_GOL_SURFACE_ACTION_TITLE	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If events occurs and the x,y position falls in the TITLE area of the surface
GFX_GOL_SURFACE_ACTION_CLIENT	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If events occurs and the x,y position falls in the CLIENT area of the surface
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_SurfaceActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage);

GFX_GOL_SurfaceCreate Function

This function creates a GFX_GOL_SURFACE object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_surface.h

C

```
GFX_GOL_SURFACE * GFX_GOL_SurfaceCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, GFX_GOL_SURFACE_CALLBACK_FUNC callback,
GFX_RESOURCE_HDR * pImage, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

This function creates a GFX_GOL_SURFACE object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_SURFACE *pSurface;
GFX_GOL_SURFACE_STATE state;

// assume pScheme is initialized to a scheme in memory.
state = GFX_GOL_SURFACE_DRAW_STATE;


pSurface  = GFX_GOL_SurfaceCreate(
       gfxIndex,
       1,           // ID
```

```
            0,0,GFX_Primitive_MaxXGet(),GFX_Primitive_MaxYGet(), // whole screen dimension
            state,                  // set state to draw all
            (char*)myIcon,              // icon
            "Place Title Here.",                    // text
            NULL);                  // use default GOL scheme

    if (pSurface == NULL)
        return 0;

    return 1;
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pImage	Pointer to the image used on the face of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

GFX_GOL_SURFACE *GFX_GOL_SurfaceCreate(
SYS_MODULE_INDEX gfxIndex,
uint16_t ID,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t state,
GFX_RESOURCE_HDR *pImage,
GFX_XCHAR *pText,
GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME *pScheme)

GFX_GOL_SurfaceDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_surface.h

C

```
GFX_STATUS GFX_GOL_SurfaceDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the face of the GFX_GOL_SURFACE is drawn on top of the bitmap. Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See

GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_SurfaceDraw(void *pObject)

Text Entry Object

Text Entry is an object that can be used to emulate a key pad entry with a display area for the entered characters.

Functions

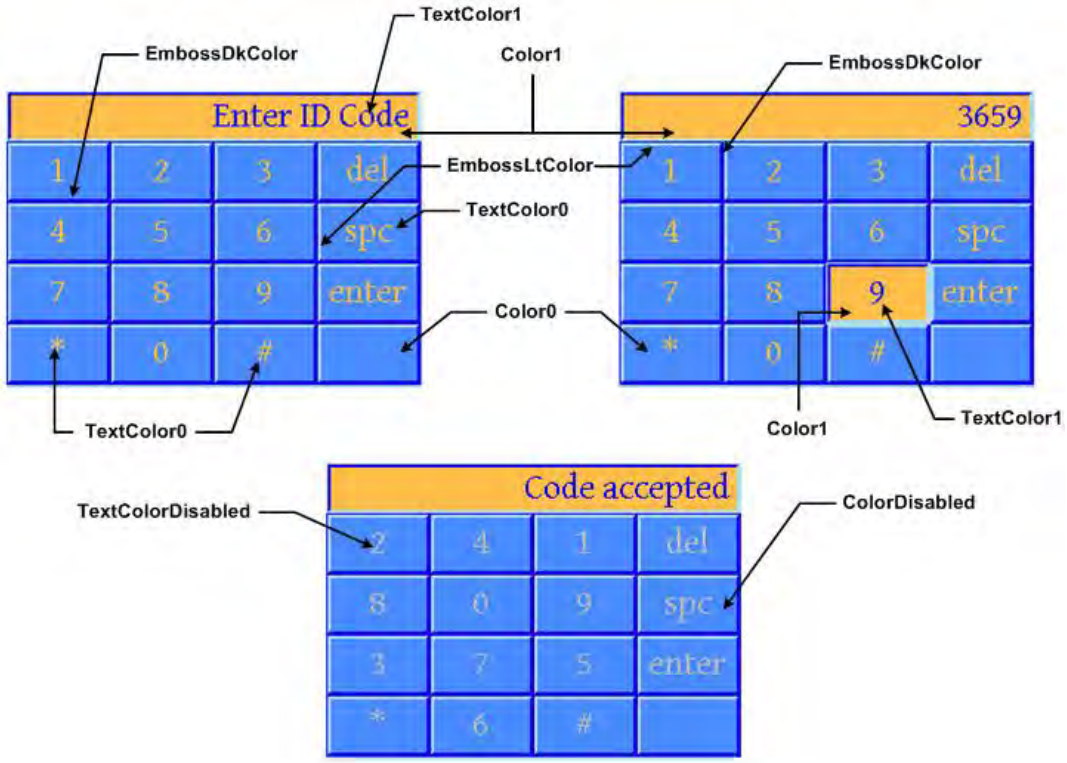
	Name	Description
	GFX_GOL_TextEntryActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_TextEntryActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_TextEntryBufferGet	This function returns the buffer used to display text.
	GFX_GOL_TextEntryBufferClear	This function will clear the data in the display.
	GFX_GOL_TextEntryBufferSet	This function sets the buffer used to display text.
	GFX_GOL_TextEntryCharAdd	This function will insert a character to the end of the buffer.
	GFX_GOL_TextEntryCreate	This function creates a GFX_GOL_TEXTENTRY object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_TextEntryDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_TextEntryKeyCommandGet	This function will return the currently assigned command to the key with the given index.
	GFX_GOL_TextEntryKeyCommandSet	This function will assign a command to a key with the given index.
	GFX_GOL_TextEntryKeyIsPressed	This function will test if a key given by its index in the object is pressed.
	GFX_GOL_TextEntryKeyListCreate	This function will create the list of key members that holds the information on each key.
	GFX_GOL_TextEntryKeyMemberListDelete	This function will delete the key member list assigned to the object.
	GFX_GOL_TextEntryKeyTextSet	This function will set the text assigned to a key with the given index.
	GFX_GOL_TextEntryLastCharDelete	This function will remove the last character of the buffer and replace it with a string terminator.
	GFX_GOL_TextEntrySpaceCharAdd	This function will insert a space character to the end of the buffer.

Description

Text Entry is an object that can be used to emulate a key pad entry with a display area for the entered characters. The object has a feature where you can define a key to reply with a translated message that signifies a command key was pressed. A command key example can be your enter or carriage return key or an escape key. Multiple keys can be assigned command keys. Application can utilize the command key to define the behavior of the program based on a command key press.

Static Text supports Touchscreen inputs only, replying to the events with the predefined actions (see GFX_GOL_TextEntryActionGet() and GFX_GOL_TextEntryActionSet() for details).

The Text Entry object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



When creating the object, the alignment of the text of the object can be formatted with the same options that GFX_TextStringBoxDraw() allows.

GFX_GOL_TextEntryActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_text_entry.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_TextEntryActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_TEXTENTRY_ACTION_PRESS - A key is pressed
- GFX_GOL_TEXTENTRY_ACTION_RELEASED - A key was released (generic for keys with no commands or characters assigned)
- GFX_GOL_TEXTENTRY_ACTION_ADD_CHAR - A key was released with character assigned
- GFX_GOL_TEXTENTRY_ACTION_DELETE - A key was released with delete command assigned
- GFX_GOL_TEXTENTRY_ACTION_SPACE - A key was released with space command assigned
- GFX_GOL_TEXTENTRY_ACTION_ENTER - A key was released with enter command assigned
- GFX_GOL_OBJECT_ACTION_INVALID - Text Entry is not affected

Description

GFX GOL text entry action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit	Description
GFX_GOL_TEXTENTRY_ACTION_PRESS	Touch Screen	EVENT_PRESS, EVENT_MOVE	If the event occurs and the x,y position falls in the face of one of the keys of the object while the key is unpressed.
GFX_GOL_TEXTENTRY_ACTION_RELEASED	Touch Screen	EVENT_MOVE	If the event occurs and the x,y position falls outside the face of one of the keys of the object while the key is pressed.
GFX_GOL_TEXTENTRY_ACTION_RELEASED	Touch Screen	EVENT_RELEASE	If the event occurs and the x,y position falls does not falls inside any of the faces of the keys of the object.

GFX_GOL_TEXTENTRY_ACTION_ADD_CHAR	Touch Screen	EVENT_RELEASE, EVENT_MOVE	If the event occurs and the x,y position falls in the face of one of the keys of the object while the key is unpressed and the key is associated with no commands.
GFX_GOL_TEXTENTRY_ACTION_DELETE	Touch Screen	EVENT_RELEASE, EVENT_MOVE	If the event occurs and the x,y position falls in the face of one of the keys of the object while the key is unpressed and the key is associated with delete command.
GFX_GOL_TEXTENTRY_ACTION_SPACE	Touch Screen	EVENT_RELEASE, EVENT_MOVE	If the event occurs and the x,y position falls in the face of one of the keys of the object while the key is unpressed and the key is associated with space command.
GFX_GOL_TEXTENTRY_ACTION_ENTER	Touch Screen	EVENT_RELEASE, EVENT_MOVE	If the event occurs and the x,y position falls in the face of one of the keys of the object while the key is unpressed and the key is associated with enter command.
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_TextEntryActionGet(
void *pObject,
GFX_GOL_MESSAGE *pMessage);

GFX_GOL_TextEntryActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntryActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject, GFX_GOL_MESSAGE *
pMessage);
```

Returns

None.

Description

GFX GOL text entry action set.

This function performs the state change of the object based on the translated action. This change can be overridden by the application using the application defined GFX_GOL_MESSAGE_CALLBACK_FUNC. When the user message is determined to affect the object, application can perform the state change in the message callback function. The following state changes are supported:

Translated Message Input Source	Set/Clear State Bit	Description
GFX_GOL_TEXTENTRY_ACTION_ADD_CHAR Touch Screen,	Set GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE,	Add a character in the buffer and update the text displayed.
GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE,	Clear GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	
GFX_GOL_TEXTENTRY_ACTION_SPACE Touch Screen,	Set GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE,	Insert a space character in the buffer and update the text displayed.
GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE,	Clear GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	

GFX_GOL_TEXTENTRY_ACTION_DELETE Touch Screen,	Set GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE,	Delete the most recent character in the buffer and update the text displayed.
GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE,	Clear GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	
GFX_GOL_TEXTENTRY_ACTION_ENTER Touch Screen,	Set GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE,	User can define the use of this event in the message callback. Object will just update the key.
GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE,	Clear GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	
GFX_GOL_TEXTENTRY_ACTION_RELEASED Touch Screen,	Clear GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	A Key in the object will be redrawn in the unpressed state.
	Set GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE	
GFX_GOL_TEXTENTRY_ACTION_PRESSED Touch Screen,	Set GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	A Key in the object will be redrawn in the pressed state.
GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE		

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
translatedMsg	The action of the object based on the message.
pObject	The pointer to the object whose state will be modified.
pMessage	The pointer to the original message.

Function

void GFX_GOL_TextEntryActionSet(
GFX_GOL_TRANSLATED_ACTION translatedMsg,
void *pObject,
GFX_GOL_MESSAGE *pMessage)

GFX_GOL_TextEntryBufferGet Function

This function returns the buffer used to display text.

File

gfx_gol_text_entry.h

C

```
GFX_XCHAR * GFX_GOL_TextEntryBufferGet(GFX_GOL_TEXTENTRY * pObject);
```

Returns

The pointer to the text buffer used to display text.

Description

GFX GOL text entry buffer get.

This function returns the buffer used to display text.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_TextEntryBufferGet(
GFX_GOL_TEXTENTRY *pObject)

GFX_GOL_TextEntryBufferClear Function

This function will clear the data in the display.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntryBufferClear(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

GFX GOL text entry buffer clear.

This function will clear the data in the display. Object must be redrawn to reflect the change in the buffer. Use the drawing state bit GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE to update the text on the screen.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_TextEntryBufferClear(
GFX_GOL_TEXTENTRY *pObject)

GFX_GOL_TextEntryBufferSet Function

This function sets the buffer used to display text.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntryBufferSet(GFX_GOL_TEXTENTRY * pObject, GFX_XCHAR * pText, uint16_t MaxSize);
```

Returns

None.

Description

GFX GOL text entry buffer set.

This function sets the buffer used to display text. If the buffer is initialized with a string, the string must be a null terminated string. If the string length is greater than MaxSize, string will be truncated to MaxSize. pText must point to a valid memory location with size equal to MaxSize. The total number of characters that will be displayed on the object will be MaxSize - 1 since the last character will be the string terminator character.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the new text buffer to be displayed.
maxSize	maximum length of the new buffer to be used.

Function

void GFX_GOL_TextEntryBufferSet(
GFX_GOL_TEXTENTRY *pObject,
GFX_XCHAR *pText,
uint16_t MaxSize)

GFX_GOL_TextEntryCharAdd Function

This function will insert a character to the end of the buffer.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntryCharAdd(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

GFX GOL text entry char add.

This function will insert a character to the end of the buffer. Drawing states GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE or GFX_GOL_TEXTENTRY_DRAW_STATE must be set to see the effect of the addition.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_TextEntryCharAdd(
GFX_GOL_TEXTENTRY *pObject)

GFX_GOL_TextEntryCreate Function

This function creates a GFX_GOL_TEXTENTRY object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_text_entry.h

C

```
GFX_GOL_TEXTENTRY * GFX_GOL_TextEntryCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t
top, uint16_t right, uint16_t bottom, uint16_t state, uint16_t horizontalKeys, uint16_t verticalKeys,
GFX_XCHAR * pText[], GFX_XCHAR * pBuffer, GFX_ALIGNMENT alignment, uint16_t bufferLength, GFX_RESOURCE_HDR
* pDisplayFont, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL text entry create.

This function creates a GFX_GOL_TEXTENTRY object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

The behavior of GFX_GOL_TextEntryCreate() will be undefined if one of the following is true:

- left >= right
- top >= bottom
- pScheme is not pointing to a GFX_GOL_OBJ_SCHEME
- horizontal key or vertical key count is 0
- pText is an unterminated string
- pBuffer is initialized to an allocated memory.

Preconditions

None.

Example

```
#define ID_TEXTENTRY 0x20
#define TEBUFFERSIZE 20    // 20 characters

GFX_XCHAR delKey[] = {'d','e','l',0};
GFX_XCHAR spaceKey[] = {'s','p','c',0};
GFX_XCHAR enterKey[] = {'e','n','t','e','r',0};
GFX_XCHAR exitKey[] = {'M','a','i','n',0};
GFX_XCHAR key1[] = {'1',0};
GFX_XCHAR key2[] = {'2',0};
GFX_XCHAR key3[] = {'3',0};
GFX_XCHAR key4[] = {'4',0};
GFX_XCHAR key5[] = {'5',0};
GFX_XCHAR key6[] = {'6',0};
GFX_XCHAR key7[] = {'7',0};
GFX_XCHAR key8[] = {'8',0};
GFX_XCHAR key9[] = {'9',0};
GFX_XCHAR key0[] = {'0',0};
GFX_XCHAR keystar[] = {'*',0};
GFX_XCHAR keypound[] = {'#',0};

GFX_XCHAR *pKeyNames[] = { key1,    key2, key3,     delKey,
                           key4,    key5, key6,     spaceKey,
                           key7,    key8, key9,     enterKey,
                           keystar, key0, keypound, exitKey
                         };

// assume pScheme is initialized
// myFont is a font in memory
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_TEXTENTRY  *pTe;
GFX_RESOURCE_HDR   *pFont = &myFont;

pTe = GFX_GOL_TextEntryCreate(
    gfxIndex,
    ID_TEXTENTRY,                   // ID
    30,80,235,160,                  // dimension
    GFX_GOL_TEXTENTRY_DRAW_STATE,   // draw the object
    4,                              // number of horizontal keys
    4,                              // number of vertical keys
    pKeyNames,                      // pointer to the array of key names
    "Enter Code",                   // initial text
    GFX_ALIGN_CENTER,               // align text on the center
    TEBUFFERSIZE,                   // size of the buffer for text
    pFont,                          // pointer to the font of the
                                    // displayed text
    pScheme);                       // use given scheme
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
radius	Radius of the rounded edge. When using gradient buttons and radius != 0, emboss size <= radius. If this is not met, the the GFX_GOL_BUTTON face will not have gradient effect.
state	Sets the initial state of the object.
horizontalKeys	Number of horizontal keys
verticalKeys	Number of vertical keys
pText	array of pointer to the custom "text" assigned by the user.
pBuffer	pointer to the buffer that holds the text to be displayed.
alignment	text alignment of the text used in the object.
bufferLength	length of the buffer assigned by the user. The choice of the length should include the string null terminator.
For example	if the bufferLength is set to 3, only two characters can be shown on the object since the last character will be the string terminator character.
pDisplayFont	pointer to the font image to be used on the edit box section of the object.
pScheme	Pointer to the style scheme used.

Function

```
    GFX_GOL_TEXTENTRY *GFX_GOL_TextEntryCreate(
SYS_MODULE_INDEX    gfxIndex,
uint16_t        ID,
uint16_t        left,
uint16_t        top,
uint16_t        right,
uint16_t        bottom,
uint16_t        state,
uint16_t        horizontalKeys,
uint16_t        verticalKeys,
                    GFX_XCHAR       *pText[],
                    GFX_XCHAR       *pBuffer,
                    GFX_ALIGNMENT       alignment,
uint16_t        bufferLength,
                    GFX_RESOURCE_HDR    *pDisplayFont,
                    GFX_GOL_OBJ_SCHEME  *pScheme);
```

GFX_GOL_TextEntryDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_text_entry.h

C

```
GFX_STATUS GFX_GOL_TextEntryDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL text entry draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right

and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_TextEntryDraw(void *pObject)

GFX_GOL_TextEntryKeyCommandGet Function

This function will return the currently assigned command to the key with the given index.

File

gfx_gol_text_entry.h

C

```
GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE GFX_GOL_TextEntryKeyCommandGet(GFX_GOL_TEXTENTRY * pObject, uint16_t
index);
```

Returns

Command assigned to the key (See GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE).

Description

GFX GOL text entry key command set.

This function will return the currently assigned command to the key with the given index. (See GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE)

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
index	index or position of the key.

Function

GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE GFX_GOL_TextEntryKeyCommandSet(
GFX_GOL_TEXTENTRY *pObject,
uint16_t index,
GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE command)

GFX_GOL_TextEntryKeyCommandSet Function

This function will assign a command to a key with the given index.

File

gfx_gol_text_entry.h

C

```
bool GFX_GOL_TextEntryKeyCommandSet(GFX_GOL_TEXTENTRY * pObject, uint16_t index,
GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE command);
```

Returns

TRUE - if the assignment was a success. FALSE - if the assignment was not successful.

Description

GFX GOL text entry key command set.

This function will assign a command to a key with the given index. (See GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE)

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
index	index or position of the key.
command	command assigned for the key.

Function

bool GFX_GOL_TextEntryKeyCommandSet(
GFX_GOL_TEXTENTRY *pObject,
uint16_t index,
GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE command)

GFX_GOL_TextEntryKeyIsPressed Function

This function will test if a key given by its index in the object is pressed.

File

gfx_gol_text_entry.h

C

```
bool GFX_GOL_TextEntryKeyIsPressed(GFX_GOL_TEXTENTRY * pObject, uint16_t index);
```

Returns

TRUE - if the key is pressed. FALSE - if the key is not pressed.

Description

GFX GOL text entry key is pressed.

This function will test if a key given by its index in the object is pressed.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
index	index or position of key that is being tested.

Function

bool GFX_GOL_TextEntryKeyIsPressed(
GFX_GOL_TEXTENTRY *pObject,
uint16_t index)

GFX_GOL_TextEntryKeyListCreate Function

This function will create the list of key members that holds the information on each key.

File

gfx_gol_text_entry.h

C

```
GFX_GOL_TEXTENTRY_KEYMEMBER * GFX_GOL_TextEntryKeyListCreate(GFX_GOL_TEXTENTRY * pObject, GFX_XCHAR *
pText[]);
```

Returns

Returns the pointer to the newly created key member list. A NULL is returned if the list is not created successfully.

Description

GFX GOL text entry key list create.

This function will create the list of key members that holds the information on each key. The number of keys is determined by the equation (keycount = verticalKeys*horizontalKeys). The object creates the information holder for each key automatically and assigns each entry in the *pText[] array with the first entry automatically assigned to the key with an index of 1.

The number of entries to *pText[] must be at least equal to keycount. The last key is assigned with an index of keycount-1.

No checking is performed on the length of *pText[] entries to match (verticalKeys*horizontalKeys).

The function behavior is undefined if the pText[] array is less than the keycount value.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text defined by the application.

Function

```
GFX_GOL_TEXTENTRY_KEYMEMBER *GFX_GOL_TextEntryKeyListCreate(
                GFX_GOL_TEXTENTRY *pObject,
                GFX_XCHAR *pText[])
```

GFX_GOL_TextEntryKeyMemberListDelete Function

This function will delete the key member list assigned to the object.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntryKeyMemberListDelete(void * pObject);
```

Returns

None.

Description

GFX GOL text entry key member list delete.

This function will delete the key member list assigned to the object from memory. Pointer to the key member list is then initialized to NULL. All memory resources allocated to the key member list is freed.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_TextEntryKeyMemberListDelete(
void *pObject)

GFX_GOL_TextEntryKeyTextSet Function

This function will set the text assigned to a key with the given index.

File

gfx_gol_text_entry.h

C

```
bool GFX_GOL_TextEntryKeyTextSet(GFX_GOL_TEXTENTRY * pObject, uint16_t index, GFX_XCHAR * pText);
```

Returns

TRUE - if the assignment was a success. FALSE - if the assignment was not successful.

Description

GFX GOL text entry key text set.

This function will set the text assigned to a key with the given index.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
index	index or position of the key.
pText	pointer to the text that will be assigned to the key.

Function

bool GFX_GOL_TextEntryKeyTextSet(
GFX_GOL_TEXTENTRY *pObject,
uint16_t index,
GFX_XCHAR *pText)

GFX_GOL_TextEntryLastCharDelete Function

This function will remove the last character of the buffer and replace it with a string terminator.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntryLastCharDelete(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

GFX GOL text entry last char delete.

This function will remove the last character of the buffer and replace it with a string terminator. Drawing states GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE or GFX_GOL_TEXTENTRY_DRAW_STATE must be set to see the effect of the addition.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_TextEntryLastCharDelete(
GFX_GOL_TEXTENTRY *pObject)

GFX_GOL_TextEntrySpaceCharAdd Function

This function will insert a space character to the end of the buffer.

File

gfx_gol_text_entry.h

C

```
void GFX_GOL_TextEntrySpaceCharAdd(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

GFX GOL text entry space char add.

This function will insert a space character to the end of the buffer. Drawing states GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE or GFX_GOL_TEXTENTRY_DRAW_STATE must be set to see the effect of the addition.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

void GFX_GOL_TextEntrySpaceCharAdd(
GFX_GOL_TEXTENTRY *pObject)

Window Object

Window is an object that can be used to encapsulate objects into a group.

Functions

	Name	Description
	GFX_GOL_WindowActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_WindowCreate	This function creates a GFX_GOL_WINDOW object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_WindowDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_WindowTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_WindowTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_WindowTextSet	This function sets the address of the current text string used by the object.

Macros

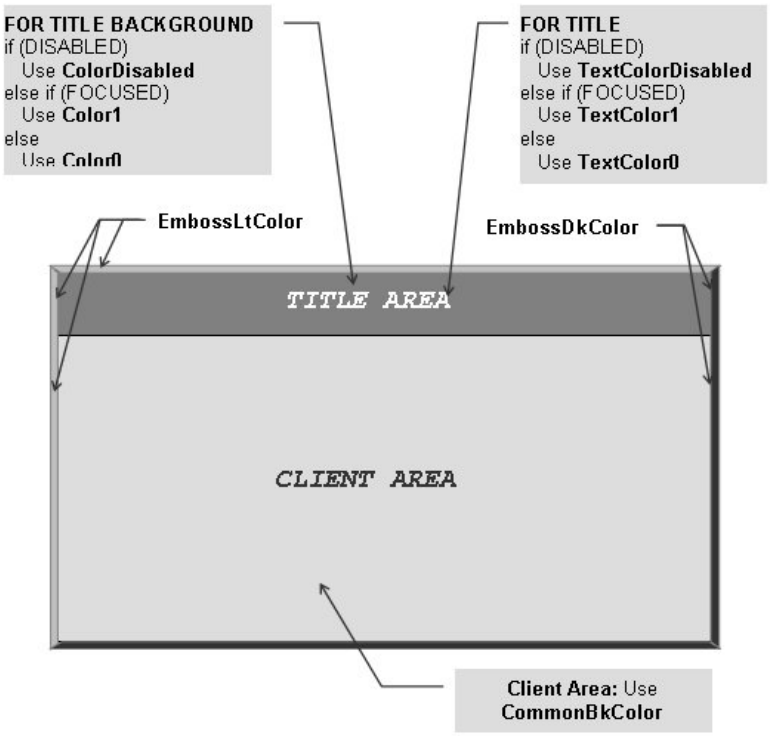
	Name	Description
	GFX_GOL_WindowImageGet	This function gets the image used.
	GFX_GOL_WindowImageSet	This function sets the image used in the object.
	GFX_GOL_WindowTextGet	This function returns the address of the current text string used by the object.

Description

Window is an object that can be used to encapsulate objects into a group. Unlike the Group Box object, the Window object has additional features such as displaying an icon or a small bitmap on its Title Bar. It also has additional controls for both Title Bar and Client Area.

Window supports Touchscreen inputs only, replying to the events with the predefined actions (see GFX_GOL_WindowActionGet() for details).

The Window object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



CommonBkColor – used to hide the window from the screen.

GFX_GOL_WindowImageGet Macro

This function gets the image used.

File

gfx_gol_window.h

C

```
#define GFX_GOL_WindowImageGet(pObject) \
                                (((GFX_GOL_WINDOW *)pObject)->pImage)
```

Returns

Pointer to the image resource.

Description

GFX GOL window image get.

This function gets the image used.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_RESOURCE_HDR *GFX_GOL_WindowImageGet(
GFX_GOL_WINDOW *pObject)

GFX_GOL_WindowImageSet Macro

This function sets the image used in the object.

File

gfx_gol_window.h

C

```
#define GFX_GOL_WindowImageSet(pObject, pImg) \
                                (((GFX_GOL_WINDOW *)pObject)->pImage = pImg)
```

Returns

None.

Description

GFX GOL window image set.

This function sets the image used in the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pImage	pointer to the image to be set.

Function

void GFX_GOL_WindowImageSet(
GFX_GOL_WINDOW *pObject,
GFX_RESOURCE_HDR *pImage)

GFX_GOL_WindowTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_window.h

C

```
#define GFX_GOL_WindowTextGet(pObject) (((GFX_GOL_WINDOW *)pObject)->pText)
```

Returns

Pointer to text string.

Description

GFX GOL window text get.

This function returns the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR *pChar;
GFX_GOL_WINDOW pWindow;

pChar = GFX_GOL_WindowTextGet(pWindow);
```

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_XCHAR *GFX_GOL_WindowTextGet(
GFX_GOL_WINDOW *pObject)

GFX_GOL_WindowActionGet Function

This function evaluates the message from a user if the message will affect the object or not.

File

gfx_gol_window.h

C

```
GFX_GOL_TRANSLATED_ACTION GFX_GOL_WindowActionGet(void * pObject, GFX_GOL_MESSAGE * pMessage);
```

Returns

- GFX_GOL_WINDOW_ACTION_CLIENT - Window client area selected action ID.
- GFX_GOL_WINDOW_ACTION_TITLE - Window title bar selected action ID.

Description

GFX GOL window action get.

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the action for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
GFX_GOL_WINDOW_ACTION_TITLE	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If events occurs and the x,y position falls in the TITLE area of the window
GFX_GOL_WINDOW_ACTION_CLIENT	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If events occurs and the x,y position falls in the CLIENT area of the window
GFX_GOL_OBJECT_ACTION_INVALID	Any	Any	If the message did not affect the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	The pointer to the object where the message will be evaluated to check if the message will affect the object.
pMessage	Pointer to the the message from the user interface.

Function

GFX_GOL_TRANSLATED_ACTION GFX_GOL_WindowActionGet(

void *pObject,

GFX_GOL_MESSAGE *pMessage);

GFX_GOL_WindowCreate Function

This function creates a GFX_GOL_WINDOW object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

File

gfx_gol_window.h

C

```
GFX_GOL_WINDOW * GFX_GOL_WindowCreate(SYS_MODULE_INDEX gfxIndex, uint16_t ID, uint16_t left, uint16_t top,
uint16_t right, uint16_t bottom, uint16_t state, GFX_RESOURCE_HDR * pImage, GFX_XCHAR * pText,
GFX_ALIGNMENT alignment, GFX_GOL_OBJ_SCHEME * pScheme);
```

Returns

Pointer to the newly created object.

Description

GFX GOL window create.

This function creates a GFX_GOL_WINDOW object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

This function returns the pointer to the newly created object. If the object is not successfully created, it returns NULL.

Preconditions

None.

Example

```
GFX_GOL_OBJ_SCHEME *pScheme;
GFX_GOL_WINDOW *pWindow;
GFX_GOL_WINDOW_STATE state;

// assume pScheme is initialized to a scheme in memory.
state = GFX_GOL_WINDOW_DRAW_STATE;


pWindow  = GFX_GOL_WindowCreate(
       gfxIndex,
       1,           // ID
       0,0,GFX_Primitive_MaxXGet(),GFX_Primitive_MaxYGet(), // whole screen dimension
       state,                   // set state to draw all
       (char*)myIcon,               // icon
       "Place Title Here.",                       // text
       NULL);                   // use default GOL scheme

if (pWindow == NULL)
    return 0;

return 1;
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
ID	Unique user defined ID for the object instance.
left	Left most position of the object.
top	Top most position of the object.
right	Right most position of the object.
bottom	Bottom most position of the object.
state	Sets the initial state of the object.
pImage	Pointer to the image used on the face of the object.
pText	Pointer to the text of the object.
alignment	text alignment of the text used in the object.
pScheme	Pointer to the style scheme used.

Function

GFX_GOL_WINDOW *GFX_GOL_WindowCreate(
SYS_MODULE_INDEX gfxIndex,
uint16_t ID,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom,
uint16_t state,
GFX_RESOURCE_HDR *pImage,
GFX_XCHAR *pText,
GFX_ALIGNMENT alignment,
GFX_GOL_OBJ_SCHEME *pScheme)

GFX_GOL_WindowDraw Function

This function renders the object on the screen based on the current state of the object.

File

gfx_gol_window.h

C

```
GFX_STATUS GFX_GOL_WindowDraw(void * pObject);
```

Returns

GFX_STATUS_SUCCESS - When the object rendering is finished. GFX_STATUS_FAILURE - When the object rendering is not yet finished. Application needs to call this rendering function again to continue the rendering.

Description

GFX GOL window draw.

This function renders the object on the screen based on the current state of the object. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the object. The font used is determined by the style scheme set.

The text on the face of the GFX_GOL_WINDOW is drawn on top of the bitmap. Text alignment based on the alignment parameter set on the object.

When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is started. This is to avoid incomplete object rendering.

Normally, application will just call GFX_GOL_ObjectListDraw() to allow the Graphics Library to manage all object rendering. See GFX_GOL_ObjectListDraw() for more information on object rendering.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.

Function

GFX_STATUS GFX_GOL_WindowDraw(void *pObject)

GFX_GOL_WindowTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_window.h

C

```
GFX_ALIGNMENT GFX_GOL_WindowTextAlignmentGet(GFX_GOL_WINDOW * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

GFX GOL window text alignment get.

This function returns the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_ALIGNMENT GFX_GOL_WindowTextAlignmentGet(
GFX_GOL_WINDOW *pObject)

GFX_GOL_WindowTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_window.h

C

```
void GFX_GOL_WindowTextAlignmentSet(GFX_GOL_WINDOW * pObject, GFX_ALIGNMENT alignment);
```

Returns

None.

Description

GFX GOL window text alignment set.

This function sets the text alignment of the text string used by the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	Pointer to the object.
alignment	The alignment set for the text in the object. See GFX_ALIGNMENT for more details.

Function

void GFX_GOL_WindowTextAlignmentSet(
GFX_GOL_WINDOW *pObject,
GFX_ALIGNMENT alignment)

GFX_GOL_WindowTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_window.h

C

```
void GFX_GOL_WindowTextSet(GFX_GOL_WINDOW * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

GFX GOL window text set.

This function sets the address of the current text string used by the object.

Preconditions

Object must exist in memory.

Example

```
GFX_XCHAR Label0[] = ?ON?;
GFX_XCHAR Label1[] = ?OFF?;
GFX_GOL_WINDOW pWindow;

GFX_GOL_WindowTextSet(pWindow, Label0);
GFX_GOL_WindowTextSet(pWindow, Label1);
```

Parameters

Parameters	Description
pObject	pointer to the object.
pText	pointer to the text string to be used.

Function

void GFX_GOL_WindowTextSet(
GFX_GOL_WINDOW *pObject,
GFX_XCHAR *pText)

GOL Object States

Macros

	Name	Description
	GFX_GOL_ObjectStateClear	This function clears the state bits of the given object.
	GFX_GOL_ObjectStateGet	This function retrieves the current value of the state bits of an object.
	GFX_GOL_ObjectStateSet	This function sets the state bits of the given object.

Description

Graphics Library Object Layer object state API.

GFX_GOL_ObjectStateClear Macro

This function clears the state bits of the given object.

File

gfx_gol.h

C

```
#define GFX_GOL_ObjectStateClear(pObject, stateBits) \
                ((((GFX_GOL_OBJ_HEADER*)pObject)->state) &= (~(stateBits)))
```

Returns

GFX_STATUS_SUCCESS - is returned if the clear was successful. GFX_STATUS_FAILURE - is returned if the clear was not successful.

Description

GFX GOL Object state clear.

This function clears the state bits of the given object. Object must be redrawn to display the changes. It is possible to set several state bits with this function.

Preconditions

None.

Example

See GFX_GOL_ObjectStateSet() for code example.

Parameters

Parameters	Description
pObject	Pointer to the object.
stateBits	Defines which state bits are to be cleared. Please refer to specific objects for object state bits definition for details

Function

GFX_STATUS GFX_GOL_ObjectStateClear(
GFX_GOL_OBJ_HEADER *pObject,
uint16_t stateBits);

GFX_GOL_ObjectStateGet Macro

This function retrieves the current value of the state bits of an object.

File

gfx_gol.h

C

```
#define GFX_GOL_ObjectStateGet(pObject, stateBits) \
                (((GFX_GOL_OBJ_HEADER*)pObject)->state & stateBits)
```

Returns

The current status of the specified state bits.

Description

GFX GOL object state get

This function retrieves the current value of the state bits of an object. It is possible to get several state bits.

Preconditions

None.

Example

```
#define BTN_HIDE 0x8000

GFX_GOL_BUTTON *pB;
// pB is created and initialized
// do something here to set state

// Hide the button (remove from screen)
if (GFX_GOL_ObjectStateGet(pB, GFX_GOL_BUTTON_HIDE_STATE))
{
    GFX_ColorSet(pB->pGolScheme->CommonBkColor);
    GFX_BarDraw(pB->left, pB->top,pB->right,pB->bottom);
}
```

Parameters

Parameters	Description
pObject	Pointer to the object.
stateBits	Defines which state bits are to be retrieved. Please refer to specific objects for object state bits definition for details

Function

uint16_t GFX_GOL_ObjectStateGet(
GFX_GOL_OBJ_HEADER *pObject,

uint16_t stateBits);

GFX_GOL_ObjectStateSet Macro

This function sets the state bits of the given object.

File

gfx_gol.h

C

```
#define GFX_GOL_ObjectStateSet(pObject, stateBits) \
                ((((GFX_GOL_OBJ_HEADER*)pObject)->state) |= stateBits)
```

Returns

GFX_STATUS_SUCCESS - is returned if the set was successful. GFX_STATUS_FAILURE - is returned if the set was not successful.

Description

GFX GOL object state set.

This function sets the state bits of the given object. Object must be redrawn to display the changes. It is possible to set several state bits with this function.

Preconditions

None.

Example

```
void SetMessage(uint16_t msg, GFX_GOL_BUTTON* pB)
{
    switch(msg)
    {
        case GFX_GOL_BUTTON_ACTION_PRESSED:
            // set pressed and redraw
            GFX_GOL_ObjectStateSet(pB, BTN_PRESSED|BTN_DRAW);
            break;
        case GFX_GOL_BUTTON_ACTION_RELEASED:
            // reset pressed
            GFX_GOL_ObjectStateClear(pB, BTN_PRESSED);
            // redraw
            GFX_GOL_ObjectStateSet(pB, BTN_DRAW);
            break;
        default:
            break;
    }
}
```

Parameters

Parameters	Description
pObject	Pointer to the object.
stateBits	Defines which state bits are to be cleared. Please refer to specific objects for object state bits definition for details

Function

GFX_STATUS GFX_GOL_ObjectStateSet(
GFX_GOL_OBJ_HEADER *pObject,
uint16_t stateBits);

GOL Object Management

Functions

	Name	Description
	GFX_GOL_ObjectAdd	This function adds an object to the tail of the currently active list.
	GFX_GOL_ObjectByIDDelete	This function removes an object with the given user defined ID from the currently active list.
	GFX_GOL_ObjectCanBeFocused	Checks if the object can be focused.

	GFX_GOL_ObjectDelete	This function removes an object from the currently active list.
	GFX_GOL_ObjectFind	This function returns the pointer to object in the list with the user defined ID assigned to it.
	GFX_GOL_ObjectFocusGet	This function returns the pointer to the object that is currently receiving keyboard input (or focused).
	GFX_GOL_ObjectFocusNextGet	This function returns the pointer to the next object in the active list of objects which can be focused.
	GFX_GOL_ObjectFocusPrevGet	This function returns the pointer to the previous object in the active list of objects which can be focused.
	GFX_GOL_ObjectFocusSet	This function sets the object to be focused.
	GFX_GOL_ObjectIDGet	This function returns the object ID.
	GFX_GOL_ObjectListFree	This function sets the active list to the new list.
	GFX_GOL_ObjectListGet	This function returns the current active list.
	GFX_GOL_ObjectListNew	This function removes an object with the given user defined ID from the currently active list.
	GFX_GOL_ObjectListSet	This function sets the active list to the new list.
	GFX_GOL_ObjectNextGet	This function returns the pointer to next object in the list after the specified object.
	GFX_GOL_ObjectTypeGet	This function returns the object type.

Description

Graphics Library Object Layer object management API.

GFX_GOL_ObjectAdd Function

This function adds an object to the tail of the currently active list.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectAdd(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

None.

Description

GFX GOL object add.

This function adds an object to the tail of the currently active list. The new list tail is set to point to NULL after the new object is added.

Preconditions

None.

Example

```
void MoveObject(    GFX_GOL_OBJ_HEADER *pSrcList,
                    GFX_GOL_OBJ_HEADER *pDstList,
                    GFX_GOL_OBJ_HEADER *pObjtoMove)
{
    GFX_GOL_OBJ_HEADER *pTemp = pSrcList;

    if(pTemp != pObjtoMove)
    {
        while(pTemp->pNxtObj != pObjtoMove)
            pTemp = pTemp->pNxtObj;
    }

    pTemp->pNxtObj = pObjtoMove->pNxt;  // remove object from list
    GFX_GOL_ObjectListSet(pDstList);    // destination as active list
    GFX_GOL_ObjectAdd(gfxIndex, pObjtoMove);      // add object to active list
}
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. Pointer to the object that will be added to the list.

Function

void GFX_GOL_ObjectAdd(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectByIDDelete Function

This function removes an object with the given user defined ID from the currently active list.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectByIDDelete(SYS_MODULE_INDEX gfxIndex, uint16_t id);
```

Returns

GFX_STATUS_SUCCESS - is returned if the removal was successful. GFX_STATUS_FAILURE - is returned if the removal was not successful.

Description

GFX GOL object by ID delete.

This function removes an object with the given user defined ID from the currently active list. Aside from the removal of the object from the list, the RAM resources consumed by the object is also freed.

If there is no object with the given ID, the function exits with GFX_STATUS_FAILURE.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. Pointer to the object that will removed from the list.

Function

GFX_STATUS GFX_GOL_ObjectByIDDelete(SYS_MODULE_INDEX gfxIndex,
uint16_t id)

GFX_GOL_ObjectCanBeFocused Function

Checks if the object can be focused.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectCanBeFocused(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

GFX_STATUS_SUCCESS - when the object can be focused GFX_STATUS_FAILURE - when the object cannot be focused or do not support the focus feature.

Description

GFX GOL object can be focused.

This function checks if the object can be focused or not. If the object can be focused, it returns GFX_STATUS_SUCCESS. If it cannot be focused,

it returns GFX_STATUS_FAILURE. Selected objects have the focus feature. Refer to the object documentation for details.

Objects that do not support focus feature will ignore any focus settings.

If the object is disabled it cannot be set to focused state.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObject	pointer to the object.

Function

GFX_STATUS GFX_GOL_ObjectCanBeFocused(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectDelete Function

This function removes an object from the currently active list.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectDelete(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

GFX_STATUS_SUCCESS - is returned if the removal was successful. GFX_STATUS_FAILURE - is returned if the removal was not successful.

Description

GFX GOL object delete.

This function removes an object from the currently active list. Aside from the removal of the object from the list, the RAM resources consumed by the object is also freed.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. Pointer to the object that will removed from the list.

Function

GFX_STATUS GFX_GOL_ObjectDelete(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectFind Function

This function returns the pointer to object in the list with the user defined ID assigned to it.

File

gfx_gol.h

C

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFind(SYS_MODULE_INDEX gfxIndex, uint16_t ID);
```

Returns

The pointer to the object in the list with the given ID.

Description

GFX GOL object find.

This function returns the pointer to object in the list with the user defined ID assigned to it.

Preconditions

None.

Example

```
void CopyObject(GFX_GOL_OBJ_HEADER *pSrcList,
                GFX_GOL_OBJ_HEADER *pDstList,
                uint16_t ID)
{
    GFX_GOL_OBJ_HEADER *pTemp;

    // find the object
    pTemp = GFX_GOL_ObjectFind(gfxIndex, ID);

    if (pTemp != NULL)
    {
        // destination as active list

        GFX_GOL_ObjectListSet(gfxIndex, pDstList);

        // add object to active list
        GFX_GOL_ObjectAdd(gfxIndex, pTemp);
    }
}
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. User defined ID of the object of interest.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFind(SYS_MODULE_INDEX gfxIndex, uint16_t ID)

GFX_GOL_ObjectFocusGet Function

This function returns the pointer to the object that is currently receiving keyboard input (or focused).

File

gfx_gol.h

C

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFocusGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The pointer the currently focused object. Returns NULL if there is no object currently set.

Description

GFX GOL object focus get.

This function returns the pointer to the object that is currently receiving keyboard input (or focused).

If there are no object that can accept keyboard messages, then the function will return NULL.

Objects that can be focused are those objects that can receive keyboard inputs.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFocusGet(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectFocusNextGet Function

This function returns the pointer to the next object in the active list of objects which can be focused.

File

gfx_gol.h

C

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFocusNextGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The pointer to the object that can be focused.

Description

GFX GOL object focus next get.

This function returns the pointer to the next object in the active list of objects which can be focused.

The reference point is the currently focused object. If there is no currently focused object, the searched starts from the beginning of the active list of objects.

Objects that can be focused are those objects that can receive keyboard inputs.

If there is no object capable of receiving keyboard inputs (i.e. none can be focused) NULL is returned.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFocusNextGet(
SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectFocusPrevGet Function

This function returns the pointer to the previous object in the active list of objects which can be focused.

File

gfx_gol.h

C

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFocusPrevGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The pointer to the object that can be focused.

Description

GFX GOL object focus previous get.

This function returns the pointer to the previous object in the active list of objects which can be focused.

The reference point is the currently focused object. If there is no currently focused object, the searched starts from the beginning of the active list of objects.

Objects that can be focused are those objects that can receive keyboard inputs.

If there is no object capable of receiving keyboard inputs (i.e. none can be focused) NULL is returned.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFocusPrevGet(
SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectFocusSet Function

This function sets the object to be focused.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectFocusSet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

GFX_STATUS_SUCCESS - when the object can be focused GFX_STATUS_FAILURE - when the object cannot be focused or do not support the focus feature.

Description

GFX GOL object focus set.

This function sets the specified object to be focused.

If the object cannot accept keyboard messages, the object will not be set to focused state. If the object can accept keyboard messages, then the focus state will be set and will be marked to be redrawn to show the focus when the focus feature is enabled.

Objects that can be focused are those objects that can receive keyboard inputs.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObject	pointer to the object.

Function

GFX_STATUS *GFX_GOL_ObjectFocusSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectIDGet Function

This function returns the object ID.

File

gfx_gol.h

C

```
uint16_t GFX_GOL_ObjectIDGet(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

The user defined ID of the object.

Description

GFX GOL object ID get.

This function returns the user defined ID assigned to the object.

Preconditions

None.

Example

```
void ExampleUsageOfGettingID(GFX_GOL_OBJ_HEADER *pObject)
{
    uint16_t id;

    switch(id = GFX_GOL_ObjectIDGet(pObject))
    {
        case ID_WINDOW1:
            // do something
        case ID_WINDOW2:
            // do something else
        case ID_WINDOW3:
            // do something else
        default:
            // do something else
    }
}
```

Function

uint16_t GFX_GOL_ObjectIDGet(GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectListFree Function

This function sets the active list to the new list.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectListFree(SYS_MODULE_INDEX gfxIndex);
```

Returns

GFX_STATUS_SUCCESS - is returned if the free was successful. GFX_STATUS_FAILURE - is returned if the free was not successful.

Description

GFX GOL Object List free.

This function frees all the memory used by objects in the active list and initializes the active list pointer to NULL to start a new empty list. This function must be called only inside the GFX_GOL_ObjectDrawCallback() function when using GFX_GOL_ObjectListDraw() and GFX_GOL_ObjectMessage() functions. This requirement assures that primitive rendering settings are not altered by the rendering state machines of the objects.

Preconditions

None.

Example

```
void DeletePage(GFX_GOL_OBJ_HEADER *pPage)
{
    GFX_GOL_OBJ_HEADER *pTemp;

    // assuming pPage is different from the current active list
```

```
        // save the active list
pTemp = GFX_GOL_ObjectListGet(gfxIndex);

        // set list as active list
        GFX_GOL_ObjectListSet(gfxIndex, pPage);

        // pPage objects are deleted
GFX_GOL_ObjectListFree(gfxIndex);

        // restore the active list
GFX_GOL_ObjectListSet(gfxIndex, pTemp);
    }
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_STATUS GFX_GOL_ObjectListFree(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectListGet Function

This function returns the current active list.

File

gfx_gol.h

C

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectListGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

Pointer (type GFX_GOL_OBJ_HEADER) to the current active list.

Description

GFX GOL object list get

This function returns the pointer to the current active.

Preconditions

None.

Example

See GFX_GOL_ObjectLlistNew() for example code.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectListGet(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectListNew Function

This function removes an object with the given user defined ID from the currently active list.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectListNew(SYS_MODULE_INDEX gfxIndex);
```

Returns

GFX_STATUS_SUCCESS - is returned if the new list start was successful. GFX_STATUS_FAILURE - is returned if the new list start was not successful.

Description

GFX GOL object list new

This function starts a new linked list of objects and resets the keyboard focus to none. This function assigns the current active list and current focused object (receiving keyboard inputs) object pointers to NULL. Any keyboard inputs at this point will be ignored.

This function does not erase the objects in the previous list. Application must save the previous list to another pointer if to be referenced later. If not needed anymore, memory used by that list should be freed by GFX_GOL_ObjectListFree() function. In this case, freeing the list with GFX_GOL_ObjectListFree() function has the same effect as GFX_GOL_ObjectListNew() where the current active list is empty.

Preconditions

None.

Example

```
// assume pointers to objects (pButton, pWindow and pSlider
// are initialized to objects already created
// GFX_GOL_OBJ_HEADER *pButton;
// GFX_GOL_OBJ_HEADER *pWindow;
// GFX_GOL_OBJ_HEADER *pSlider;

GFX_GOL_OBJ_HEADER *pSave;

// save current list
pSave = GFX_GOL_ObjectListGet(gfxIndex);

// start the new list, after the start of the list, the
// current active list is empty.
GFX_GOL_ObjectListNew(gfxIndex);

// assume that objects are already created
// you can now add objects to the new list
GFX_GOL_ObjectAdd(gfxIndex, pButton);
GFX_GOL_ObjectAdd(gfxIndex, pWindow);
GFX_GOL_ObjectAdd(gfxIndex, pSlider);
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_STATUS GFX_GOL_ObjectListNew(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectListSet Function

This function sets the active list to the new list.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectListSet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pList);
```

Returns

GFX_STATUS_SUCCESS - is returned if the set was successful. GFX_STATUS_FAILURE - is returned if the set was not successful.

Description

GFX GOL object list set.

This function sets the active list to the new list. The previous list will still exist in memory. Application must save the previous list before the set is called if the previous list will be referenced later. If the previous list is not needed anymore, then the list must be removed from memory by GFX_GOL_ObjectListFree() function.

Setting the active list to the new list will reset the focused pointer object to NULL.

Preconditions

None.

Example

```
GFX_GOL_OBJ_HEADER *pSave;

// save current list
pSave = GFX_GOL_ObjectListSet(gfxIndex);

// start the new list
GFX_GOL_ObjectListNew(gfxIndex);

// you can now add objects to the current list
// assume that objects are already created
GFX_GOL_ObjectAdd(gfxIndex, pButton);
GFX_GOL_ObjectAdd(gfxIndex, pWindow);
GFX_GOL_ObjectAdd(gfxIndex, pSlider);

// do something here on the new list

// return the old list
GFX_GOL_ObjectListSet(gfxIndex, pSave);
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. Pointer to the new list.

Function

GFX_STATUS *GFX_GOL_ObjectListSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pList)

GFX_GOL_ObjectNextGet Function

This function returns the pointer to next object in the list after the specified object.

File

gfx_gol.h

C

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectNextGet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

The pointer to the next object in the list.

Description

GFX GOL object next get

This function returns the pointer to next object in the list after the specified object.

Preconditions

None.

Example

```
void RedrawButtons(void)
{

    GFX_GOL_OBJ_HEADER *pCurr;

    // get active list
    pCurr = GFX_GOL_ObjectListGet(gfxIndex);

    while(pCurr->pNxtObj != NULL)
    {
        // just select button objects and redraw them
        if (GFX_GOL_ObjectTypeGet(gfxIndex, pCurr) == BUTTON)
        {
            // set to be redrawn
            pCurr->state = BTN_DRAW;
```

```
            }
            pCurr = GFX_GOL_ObjectNextGet(gfxIndex, pCurr);
        }
        // redraw all buttons in the active list
        GFX_GOL_ObjectListDraw(gfxIndex);
    }
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. Pointer to the object of interest.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectNextGet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectTypeGet Function

This function returns the object type.

File

gfx_gol.h

C

```
GFX_GOL_OBJ_TYPE GFX_GOL_ObjectTypeGet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

The type of the object. The type is one of the defined enumerated types of GFX_GOL_OBJ_TYPE.

Description

GFX GOL object type get.

This function returns the object type. The object type is one of the defined enumerated types of GFX_GOL_OBJ_TYPE.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance. Pointer to the object of interest.

Function

GFX_GOL_OBJ_TYPE GFX_GOL_ObjectTypeGet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GOL Object Rendering

Functions

	Name	Description
	GFX_GOL_DrawCallbackSet	This function sets the draw callback function that the application will use to render application specific rendering.
	GFX_GOL_ObjectDrawEnable	This function sets the object to be redraw.
	GFX_GOL_ObjectDrawDisable	This function resets the drawing state bits of the object.
	GFX_GOL_ObjectIsRedrawSet	This function checks if the object needs to be redrawn or not.
	GFX_GOL_ObjectListDraw	This function redraws all objects in the current active list that has the rendering state bits set.
	GFX_GOL_ObjectListHide	This function marks all objects in the active list to be hidden.
	GFX_GOL_ObjectRectangleRedraw	This function marks all objects in the active list intersected by the given rectangular area to be redrawn.
	GFX_GOL_ObjectHideDraw	This function performs the hiding of an object from the screen.

	GFX_GOL_TwoTonePanelDraw	This function renders the two-tone panel.
	GFX_GOL_ObjectBackGroundSet	This function sets the background information.

Description

Graphics Library Object Layer object rendering API.

GFX_GOL_DrawCallbackSet Function

This function sets the draw callback function that the application will use to render application specific rendering.

File

gfx_gol.h

C

```
void GFX_GOL_DrawCallbackSet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_DRAW_CALLBACK_FUNC pFunc);
```

Returns

None.

Description

GFX GOL draw call back set.

This function sets the draw callback function that the application will use to call primitive function to implement application specific shapes.See GFX_GOL_DRAW_CALLBACK_FUNC definition for details on the draw callback function.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pFunc	pointer to the draw callback function.

Function

void GFX_GOL_DrawCallbackSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_DRAW_CALLBACK_FUNC pFunc)

GFX_GOL_ObjectDrawEnable Function

This function sets the object to be redraw.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectDrawEnable(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

None.

Description

GFX GOL object draw enable.

This function sets the object to be redrawn. For the redraw to be effective, the object must be in the current active list. If not, the redraw action will not be performed until the list where the object is currently inserted will be set as the active list.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObject	pointer to the object that will be redrawn.

Function

void GFX_GOL_ObjectDrawEnable(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectDrawDisable Function

This function resets the drawing state bits of the object.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectDrawDisable(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

None.

Description

GFX GOL object draw disable.

This function resets the drawing state bits of the object. This function can be called to cancel any drawing state bits that has been set or clears all the drawing state bits after the object has been redrawn.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObject	pointer to the object.

Function

void GFX_GOL_ObjectDrawDisable(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectIsRedrawSet Function

This function checks if the object needs to be redrawn or not.

File

gfx_gol.h

C

```
bool GFX_GOL_ObjectIsRedrawSet(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

true - when the object needs to be redrawn. false - when the object does not need to be redrawn.

Description

GFX GOL object is redraw set.

This function checks if the object needs to be redrawn or not. The function returns true if it is to be redrawn or false if it is not to be redrawn.

Preconditions

None.

Example

```
int DrawButtonWindowOnly()
{
    static GFX_GOL_OBJ_HEADER *pCurrentObj = NULL;
    uint16_t done = 0;

    if (pCurrentObj == NULL)
    {
        // get current list
       pCurrentObj = GFX_GOL_ObjectListGet(gfxIndex);
    }

    while(pCurrentObj != NULL)
    {
        if(GFX_GOL_ObjectIsRedrawSet(gfxIndex, pCurrentObj) == true)
        {
            done = pCurrentObj->draw(gfxIndex, pCurrentObj);

            // reset state of object if done
            if (done)
                GOLDrawComplete(pCurrentObj)
                // Return if not done. This means that Button Draw function
                // was not able to finish redrawing the object
                // and must be called again to finish rendering of
                // objects in the list that have new states.
            else
                return 0;
        }
        // go to the next object in the list
        pCurrentObj = pCurrentObj->pNxtObj;
    }
    return 1;
}
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObject	pointer to the object that will be checked.

Function

bool GFX_GOL_ObjectIsRedrawSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectListDraw Function

This function redraws all objects in the current active list that has the rendering state bits set.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectListDraw(SYS_MODULE_INDEX gfxIndex);
```

Returns

GFX_STATUS_SUCCESS - is returned when the active list is completely parsed and redrawn. GFX_STATUS_BUSY - is returned when the active list is not completely parsed and redrawn.

Description

GFX GOL object list draw.

This function loops through the active list and redraws objects that need to be redrawn. Partial redrawing or full redraw is performed depending on

the drawing states of the objects.

GFX_GOL_ObjectDrawCallback() function is called by GFX_GOL_ObjectListDraw() when drawing of objects in the active list is completed. GFX_GOL_ObjectDrawCallback() is an application implemented function that allows the application the opportunity to insert application specific rendering using Primitive Layer rendering functions.

The GFX_GOL_ObjectListDraw() function can return with GFX_STATUS_BUSY. In this case, it indicates that the currently redrawn object is not able to continue. Application needs to call GFX_GOL_ObjectListDraw() again to continue the redraw of the objects in the list.

Preconditions

None.

Example

```
    // Assume objects are created & states are set to draw objects
    while(1)
    {
        // parse active list and redraw objects
        // that needs to be redrawn
if( GFX_GOL_ObjectListDraw(gfxIndex) == GFX_STATUS_SUCCESS)
        {
            // at this point drawing is completed
            // it is safe to modify objects states and linked list

            // evaluate messages from touch screen device
            TouchGetMsg(&msg);

            // evaluate each object is affected by the message
            GFX_GOL_ObjectMessage(gfxIndex, &msg);
        }
    }
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_STATUS GFX_GOL_ObjectListDraw(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectListHide Function

This function marks all objects in the active list to be hidden.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectListHide(SYS_MODULE_INDEX gfxIndex);
```

Returns

None.

Description

GFX GOL object list hide.

This function marks all objects in the active list to be hidden.

After calling this function, the next call to GFX_GOL_ObjectListDraw() will hide all objects.

Preconditions

The objects must be using the same background information.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

void GFX_GOL_ObjectListHide(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectRectangleRedraw Function

This function marks all objects in the active list intersected by the given rectangular area to be redrawn.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectRectangleRedraw(SYS_MODULE_INDEX gfxIndex, uint16_t left, uint16_t top, uint16_t right,
uint16_t bottom);
```

Returns

None.

Description

GFX GOL object rectangle redraw.

This function marks all objects in the active list intersected by the given rectangular area to be redrawn.

After calling this function, the next call to GFX_GOL_ObjectListDraw() will redraw all objects that are marked for redraw.

Preconditions

None.

Example

```
GFX_GOL_OBJ_HEADER *pTemp;
GFX_GOL_OBJ_HEADER *pAllObjects;

// assume *pAllObjects points to a list of all existing objects
// created and initialized

// mark all objects inside the rectangle to be redrawn
GFX_GOL_ObjectRectangleRedraw(gfxIndex,10,10,100,100);

// save the current active list
pTemp = pAllObjects;

// reset active list
GFX_GOL_ObjectListNew(gfxIndex);

// build the new active list with only those objects that
// are marked to be redrawn
while(pTemp->pNxtObj != NULL)
{
    if (pTemp->state&0x7C00)
        GFX_GOL_ObjectAdd(gfxIndex, pTemp);

    pTemp = pTemp->pNxtObj;
}

 // redraw active list
GFX_GOL_ObjectListDraw(gfxIndex);
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
left	Defines the left most border of the rectangle area.
top	Defines the top most border of the rectangle area.
right	Defines the right most border of the rectangle area.
bottom	Defines the bottom most border of the rectangle area.

Function

void GFX_GOL_ObjectRectangleRedraw(SYS_MODULE_INDEX gfxIndex,
uint16_t left,
uint16_t top,
uint16_t right,
uint16_t bottom)

GFX_GOL_ObjectHideDraw Function

This function performs the hiding of an object from the screen.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_ObjectHideDraw(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObject);
```

Returns

GFX_STATUS_SUCCESS - when the hiding is done. GFX_STATUS_FAILURE - when the hiding is not yet done.

Description

GFX GOL object hide draw.

This function performs the hiding of an object from the screen. If the object's style scheme is set to have a background, the background is taken into account.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObject	pointer to the object that will be checked.

Function

GFX_STATUS GFX_GOL_ObjectHideDraw(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_TwoTonePanelDraw Function

This function renders the two-tone panel.

File

gfx_gol.h

C

```
GFX_STATUS GFX_GOL_TwoTonePanelDraw(SYS_MODULE_INDEX gfxIndex);
```

Returns

GFX_STATUS_SUCCESS - when the panel rendering is done. GFX_STATUS_FAILURE - when the panel rendering is not yet done.

Description

GFX GOL two tone panel draw.

This function renders the two-tone panel. Panel parameters are set by the GFX_GOL_PanelParameterSet(). The function returns success (GFX_STATUS_SUCCESS) when the panel is rendered. If the function returned not success this function must be called again until success is returned.

Preconditions

Panel parameters must be set first using GFX_GOL_PanelParameterSet().

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_STATUS GFX_GOL_TwoTonePanelDraw(SYS_MODULE_INDEX gfxIndex)

GFX_GOL_ObjectBackGroundSet Function

This function sets the background information.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectBackGroundSet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_OBJ_HEADER * pObjectHeader);
```

Returns

None.

Description

GFX GOL object back ground set.

This function sets the background information. This is an internal function and should not be called by the application. This function is used by object's drawing functions to set the background information.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pObjectHeader	the object header of the object that needs to draw a panel with background.

Function

GFX_STATUS GFX_GOL_ObjectBackGroundSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_OBJ_HEADER *pObjectHeader)

GOL Object Messaging

Functions

	Name	Description
	GFX_GOL_ObjectMessage	This function process the received message from the user to determine the affected objects. Depending on the message and the affected objects, object states are modified based on the default or user-defined behavior.
	GFX_GOL_MessageCallbackSet	This function sets the message callback function that the application will use to evaluate user inputs that affects the objects and application behavior.

Description

Graphics Library Object Layer object messaging API.

GFX_GOL_ObjectMessage Function

This function process the received message from the user to determine the affected objects. Depending on the message and the affected objects, object states are modified based on the default or user-defined behavior.

File

gfx_gol.h

C

```
void GFX_GOL_ObjectMessage(SYS_MODULE_INDEX gfxIndex, GFX_GOL_MESSAGE * pMsg);
```

Returns

None.

Description

GFX GOL object message.

This function receives a GFX_GOL_MESSAGE message from user and loops through the active list of objects to check which object is affected by the message. For affected objects the message is translated and GFX_GOL_ObjectMessageCallback() is called. In the call back function, user has the ability to implement action for the message. If the call back function returns non-zero, OBJMsgDefault() is called to process message for the object by default. If zero is returned OBJMsgDefault() is not called. Please refer to GOL Messages section for details.

Preconditions

None.

Example

```
// Assume objects are created & states are set to draw objects
while (1)
{
    if(GOLDraw())
    {
        // GOL drawing is completed here
        // it is safe to change objects

        // from user interface module
        TouchGetMsg(&msg);
        // process the message
        GFX_GOL_ObjectMessage(gfxIndex, &msg);
    }
}
```

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pMessage	Pointer to the message from user.

Function

void GFX_GOL_ObjectMessage(SYS_MODULE_INDEX gfxIndex, GFX_GOL_MESSAGE *pMessage)

GFX_GOL_MessageCallbackSet Function

This function sets the message callback function that the application will use to evaluate user inputs that affects the objects and application behavior.

File

gfx_gol.h

C

```
void GFX_GOL_MessageCallbackSet(SYS_MODULE_INDEX gfxIndex, GFX_GOL_MESSAGE_CALLBACK_FUNC pFunc);
```

Returns

None.

Description

GFX GOL message call back set.

This function sets the message callback function that the application will use to evaluate user inputs that affects the objects and application behavior. The callback function location is specified by the function pointer supplied in the call. See GFX_GOL_MESSAGE_CALLBACK_FUNC definition for details on the message callback function.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.
pFunc	pointer to the message callback function.

Function

void GFX_GOL_MessageCallbackSet(SYS_MODULE_INDEX gfxIndex,
GFX_GOL_MESSAGE_CALLBACK_FUNC pFunc)

GOL Object Style Scheme

Macros

	Name	Description
	GFX_GOL_ObjectStyleSchemeGet	This function returns the style scheme currently set for the object.
	GFX_GOL_ObjectStyleSchemeSet	This function sets the style scheme of the object.

Description

Graphics Library Object Layer object style scheme API.

GFX_GOL_ObjectStyleSchemeGet Macro

This function returns the style scheme currently set for the object.

File

gfx_gol.h

C

```
#define GFX_GOL_ObjectStyleSchemeGet(pObject) \
                (((GFX_GOL_OBJ_HEADER*)pObject)->pGolScheme)
```

Returns

The pointer to the currently set style scheme.

Description

GFX GOL object style scheme get.

This function returns the style scheme currently used by the object. The object must exist when this function is called. This function do not check if the object is valid.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_GOL_OBJ_SCHEME *GFX_GOL_ObjectStyleSchemeGet(
GFX_GOL_OBJ_HEADER *pObject)

GFX_GOL_ObjectStyleSchemeSet Macro

This function sets the style scheme of the object.

File

gfx_gol.h

C

```
#define GFX_GOL_ObjectStyleSchemeSet(pObject, pStyle) \
                ((((GFX_GOL_OBJ_HEADER*)pObject)->pGolScheme) = pStyle)
```

Returns

None.

Description

GFX GOL object style scheme set.

This function sets the style scheme of the object to the given style scheme. The object and style scheme must exist at the time of the assignment. This function do not check if the object or the style is valid.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pStyle	pointer to the style scheme.

Function

void GFX_GOL_ObjectStyleSchemeSet(
GFX_GOL_OBJ_HEADER *pObject,
GFX_GOL_OBJ_SCHEME *pStyle)

Data Types and Constants

Enumerations

	Name	Description
	GFX_GOL_BUTTON_STATE	Specifies the different states of the Button object.
	GFX_GOL_CHECKBOX_STATE	Specifies the different states of the Check Box object.
	GFX_GOL_DIGITALMETER_STATE	Specifies the different states of the Digital Meter object.
	GFX_GOL_GROUPBOX_STATE	Specifies the different states of the Group Box object.
	GFX_GOL_LISTBOX_ITEM_STATUS	Defines the types used to indicate the status of an item.
	GFX_GOL_LISTBOX_STATE	Specifies the different states of the List Box object.
	GFX_GOL_METER_DRAW_TYPE	Specifies the different types of Meter object.
	GFX_GOL_METER_STATE	Specifies the different states of the Meter object.
	GFX_GOL_OBJ_TYPE	Specifies the different object types used in the library.
	GFX_GOL_PROGRESSBAR_STATE	Specifies the different states of the Progress Bar object.
	GFX_GOL_RADIOBUTTON_STATE	Specifies the different states of the Radio Button object.
	GFX_GOL_SCROLLBAR_STATE	Specifies the different states of the Scroll Bar object.
	GFX_GOL_STATICTEXT_STATE	Specifies the different states of the Static Text object.
	GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE	Specifies the different commands available for command keys of the Text Entry object.

	GFX_GOL_TEXTENTRY_STATE	Specifies the different states of the Text Entry object.
	GFX_GOL_TRANSLATED_ACTION	Specifies the different object actions supported in the library.
	GFX_GOL_WINDOW_STATE	Specifies the different states of the Window object.
	GFX_GOL_EDITBOX_STATE	Specifies the different states of the Edit Box object.
	GFX_GOL_PICTURECONTROLCONTROL_STATE	Specifies the different states of the Picture Control object.
	GFX_GOL_COMMON_STATE_BITS	Common Object States.
	GFX_GOL_CUSTOMCONTROL_STATE	Specifies the different states of the Custom Control object.
	GFX_GOL_CLIENT_STATUS	Enumerated data type that identifies the GFX Module Client Status.
	GFX_GOL_OBJECT_TASK	Lists the different states that GFX GOL task routine can have.
	GFX_GOL_STATES	Defines the various states that can be achieved by the GFX GOL operation.
	GFX_AT_SCAN_CODES	The following macros are the supported AT Keyboard scan codes.
	GFX_GOL_SURFACE_STATE	Specifies the different states of the Surface object.

Macros

	Name	Description
	GFX_GOL_SurfaceImageGet	This function gets the image used.
	GFX_GOL_SurfaceImageSet	This function sets the image used in the object.

Structures

	Name	Description
	GFX_GOL_BUTTON	Defines the structure used for the Button object.
	GFX_GOL_CHECKBOX	Defines the structure used for the Check Box object.
	GFX_GOL_CUSTOMCONTROL	Defines the structure used for the Custom Control object.
	GFX_GOL_DIGITALMETER	Defines the structure used for the Digital Meter object.
	GFX_GOL_EDITBOX	Defines the structure used for the Edit Box object.
	GFX_GOL_GROUPBOX	Defines the structure used for the Group Box object.
	GFX_GOL_LISTBOX	Defines the structure used for the List Box object.
	GFX_GOL_LISTITEM	The structure that defines each item in the list box.
	GFX_GOL_MESSAGE	Specifies message structure used in the library.
	GFX_GOL_METER	Defines the structure used for the Meter object.
	GFX_GOL_OBJ_HEADER	Specifies Graphics Object Layer structure used in objects.
	GFX_GOL_PROGRESSBAR	Defines the structure used for the Progress Bar object.
	GFX_GOL_RADIOBUTTON	Defines the structure used for the Radio Button object.
	GFX_GOL_SCROLLBAR	Defines the structure used for the Scroll Bar object.
	GFX_GOL_STATICTEXT	Defines the structure used for the Static Text object.
	GFX_GOL_TEXTENTRY	Defines the structure used for the Text Entry object.
	GFX_GOL_WINDOW	Defines the structure used for the Window object.
	GOL_PANEL_PARAM	Specifies panel parameters.
	GFX_GOL_PICTURECONTROL	Defines the structure used for the Picture Control object.
	GFX_GOL_TEXTENTRY_KEYMEMBER	Defines the structure used to describe a key in the Text Entry object.
	_GFX_OBJ	Defines the object required for interaction with the GFX Library.
	GFX_GOL_SURFACE	Defines the structure used for the Surface object.
	GFX_GOL_OBJ_SCHEME	This structure specifies the style scheme components of an object.

Types

	Name	Description
	GFX_GOL_DRAW_CALLBACK_FUNC	Draw callback function definition. This application defined function allows the application to perform application specific rendering.
	GFX_GOL_MESSAGE_CALLBACK_FUNC	Message callback function definition. This application defined function allows the application to perform application specific processing of user messages.
	GFX_GOL_SURFACE_CALLBACK_FUNC	Surface callback function definition. This application defined function allows the application to perform application specific rendering.

Description

Graphics Library Object Layer data types and constants.

GFX_GOL_BUTTON Structure

Defines the structure used for the Button object.

File

gfx_gol_button.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t radius;
  uint16_t textWidth;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
  GFX_RESOURCE_HDR * pPressImage;
  GFX_RESOURCE_HDR * pReleaseImage;
} GFX_GOL_BUTTON;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
uint16_t radius;	Radius for rounded buttons.
uint16_t textWidth;	Computed text width, done at creation.
uint16_t textHeight;	Computed text height, done at creation.
GFX_XCHAR * pText;	Pointer to the text used.
GFX_ALIGNMENT alignment;	text alignment
GFX_RESOURCE_HDR * pPressImage;	Pointer to bitmap used.
GFX_RESOURCE_HDR * pReleaseImage;	Pointer to bitmap used.

Description

GFX GOL button

Typedef: GFX_GOL_BUTTON

Defines the structure used for the Button object.

1. Width is determined by right - left parameter in GFX_GOL_OBJ_HEADER.
2. Height is determined by top - bottom parameter in GFX_GOL_OBJ_HEADER.
3. radius - specifies if the GFX_GOL_BUTTON will have a rounded edge. If zero then the GFX_GOL_BUTTON will have sharp (cornered) edge.
4. If 2*radius = height = width, the GFX_GOL_BUTTON is a circular GFX_GOL_BUTTON.

Remarks

None.

GFX_GOL_BUTTON_STATE Enumeration

Specifies the different states of the Button object.

File

gfx_gol_button.h

C

```
typedef enum {
  GFX_GOL_BUTTON_FOCUSED_STATE,
  GFX_GOL_BUTTON_DISABLED_STATE,
  GFX_GOL_BUTTON_PRESSED_STATE,
  GFX_GOL_BUTTON_TOGGLE_STATE,
  GFX_GOL_BUTTON_TWOTONE_STATE,
  GFX_GOL_BUTTON_NOPANEL_STATE,
  GFX_GOL_BUTTON_DRAW_FOCUS_STATE,
  GFX_GOL_BUTTON_DRAW_STATE,
  GFX_GOL_BUTTON_HIDE_STATE
} GFX_GOL_BUTTON_STATE;
```

Members

Members	Description
GFX_GOL_BUTTON_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_BUTTON_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_BUTTON_PRESSED_STATE	Property bit for press state.
GFX_GOL_BUTTON_TOGGLE_STATE	Property bit to indicate object will have a toggle behavior.
GFX_GOL_BUTTON_TWOTONE_STATE	Property bit to indicate the object is a two tone type.
GFX_GOL_BUTTON_NOPANEL_STATE	Property bit to indicate the object will be drawn without a panel (for faster drawing when the object image used is larger than the object's panel).
GFX_GOL_BUTTON_DRAW_FOCUS_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_BUTTON_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_BUTTON_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL button state.

Typedef: GFX_GOL_BUTTON_STATE

This enumeration specifies the different states of the Button object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_CHECKBOX Structure

Defines the structure used for the Check Box object.

File

gfx_gol_check_box.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
} GFX_GOL_CHECKBOX;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
uint16_t textHeight;	Pre-computed text height
GFX_XCHAR * pText;	Pointer to text
GFX_ALIGNMENT alignment;	text alignment

Description

GFX GOL checkbox

Typedef: GFX_GOL_CHECKBOX

Defines the structure used for the Check Box object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_CHECKBOX_STATE Enumeration

Specifies the different states of the Check Box object.

File

gfx_gol_check_box.h

C

```
typedef enum {
  GFX_GOL_CHECKBOX_FOCUSED_STATE,
  GFX_GOL_CHECKBOX_DISABLED_STATE,
  GFX_GOL_CHECKBOX_CHECKED_STATE,
  GFX_GOL_CHECKBOX_DRAW_CHECK_STATE,
  GFX_GOL_CHECKBOX_DRAW_FOCUS_STATE,
  GFX_GOL_CHECKBOX_DRAW_STATE,
  GFX_GOL_CHECKBOX_HIDE_STATE
} GFX_GOL_CHECKBOX_STATE;
```

Members

Members	Description
GFX_GOL_CHECKBOX_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_CHECKBOX_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_CHECKBOX_CHECKED_STATE	Property bit to indicate object is checked.
GFX_GOL_CHECKBOX_DRAW_CHECK_STATE	Draw bit to indicate the check must be redrawn.
GFX_GOL_CHECKBOX_DRAW_FOCUS_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_CHECKBOX_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_CHECKBOX_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL checkbox state.

Typedef: GFX_GOL_CHECKBOX_STATE

This enumeration specifies the different states of the Check Box object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_CUSTOMCONTROL Structure

Defines the structure used for the Custom Control object.

File

gfx_gol_custom_control.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint8_t instance;
  uint16_t ID;
  void * pNxtObj;
  GFX_GOL_OBJ_TYPE type;
  uint16_t state;
  uint16_t left;
  uint16_t top;
  uint16_t right;
  uint16_t bottom;
  GFX_GOL_OBJ_SCHEME * pGolScheme;
  DRAW_FUNC draw;
  FREE_FUNC FreeObj;
  ACTIONGET_FUNC actionGet;
  ACTIONSET_FUNC actionSet;
  uint16_t pos;
  uint16_t prevPos;
} GFX_GOL_CUSTOMCONTROL;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_OBJ_HEADER).
uint8_t instance;	instance index
uint16_t ID;	instance unique ID
void * pNxtObj;	a pointer to the next object in the linked list
GFX_GOL_OBJ_TYPE type;	must be set to OBJ_CUSTOM
uint16_t state;	state
uint16_t left;	left border
uint16_t top;	top border
uint16_t right;	right border
uint16_t bottom;	bottom border
GFX_GOL_OBJ_SCHEME * pGolScheme;	the style scheme used
DRAW_FUNC draw;	custom control draw function
FREE_FUNC FreeObj;	function pointer to the object free function
ACTIONGET_FUNC actionGet;	function pointer to the object message function
ACTIONSET_FUNC actionSet;	function pointer to the object default message function
uint16_t pos;	current position
uint16_t prevPos;	previous position

Description

GFX GOL custom control.

Typedef: GFX_GOL_CUSTOMCONTROL

Defines the structure used for the Custom Control object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_DIGITALMETER Structure

Defines the structure used for the Digital Meter object.

File

gfx_gol_digital_meter.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textHeight;
  uint32_t Cvalue;
  uint32_t Pvalue;
  uint8_t NoOfDigits;
  uint8_t DotPos;
  GFX_ALIGNMENT alignment;
} GFX_GOL_DIGITALMETER;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_OBJ_HEADER).
uint16_t textHeight;	Precomputed text height
uint32_t Cvalue;	Current value
uint32_t Pvalue;	Previous value
uint8_t NoOfDigits;	Number of digits to be displayed
uint8_t DotPos;	Position of decimal point
GFX_ALIGNMENT alignment;	text alignment

Description

GFX GOL digital meter.

Typedef: GFX_GOL_DIGITALMETER

Defines the parameters required for a Digital Meter Object. Depending on the type selected the meter is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_DIGITALMETER_STATE Enumeration

Specifies the different states of the Digital Meter object.

File

gfx_gol_digital_meter.h

C

```
typedef enum {
  GFX_GOL_DIGITALMETER_DISABLED_STATE,
  GFX_GOL_DIGITALMETER_FRAME_STATE,
  GFX_GOL_DIGITALMETER_DRAW_STATE,
  GFX_GOL_DIGITALMETER_UPDATE_STATE,
  GFX_GOL_DIGITALMETER_HIDE_STATE
} GFX_GOL_DIGITALMETER_STATE;
```

Members

Members	Description
GFX_GOL_DIGITALMETER_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_DIGITALMETER_FRAME_STATE	Property bit to indicate frame is displayed.
GFX_GOL_DIGITALMETER_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_DIGITALMETER_UPDATE_STATE	Draw bit to indicate that only text must be redrawn.
GFX_GOL_DIGITALMETER_HIDE_STATE	Draw bit to indicate the object must be removed from the screen.

Description

GFX GOL digital meter state.

Typedef: GFX_GOL_DIGITALMETER_STATE

This enumeration specifies the different states of the Digital Meter object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_DRAW_CALLBACK_FUNC Type

Draw callback function definition. This application defined function allows the application to perform application specific rendering.

File

gfx_gol.h

C

```
typedef bool (* GFX_GOL_DRAW_CALLBACK_FUNC)(void);
```

Returns

true - is returned when application rendering is done. false - is returned when application rendering is not yet finished.

Description

GFX GOL draw callback function pointer

This callback function is implemented by the application. This is called inside the GFX_GOL_ObjectListDraw() function when the drawing of objects in the active list is completed.

Any application specific rendering must be performed on this callback function so no object rendering will be affected by the application calls to

primitive rendering functions. Application setting the drawing color, line style, fill style, text string cursor position and current font will not affect the object rendering. This is also the safe place to modify the active list.

When the application has performed its own primitive rendering calls, this function must return true to inform the GFX_GOL_ObjectListDraw() that it is done rendering and checking for object drawing or redrawing can continue.

Preconditions

None.

Example

None.

Function

typedef bool (*GFX_GOL_DRAW_CALLBACK_FUNC) (void);

GFX_GOL_EDITBOX Structure

Defines the structure used for the Edit Box object.

File

gfx_gol_edit_box.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
  uint16_t charMax;
  uint16_t length;
  uint16_t cursorPosition;
} GFX_GOL_EDITBOX;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
uint16_t textHeight;	Precomputed text height.
GFX_XCHAR * pText;	Pointer to text buffer.
GFX_ALIGNMENT alignment;	text alignment
uint16_t charMax;	Maximum number of characters in the edit box.
uint16_t length;	Current text length.
uint16_t cursorPosition;	cursor position

Description

GFX GOL edit box.

Typedef: GFX_GOL_EDITBOX

Defines the parameters required for a Edit Box Object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_GROUPBOX Structure

Defines the structure used for the Group Box object.

File

gfx_gol_group_box.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textWidth;
  uint16_t textHeight;
  GFX_XCHAR * pText;
```

```
  GFX_ALIGNMENT alignment;
} GFX_GOL_GROUPBOX;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_OBJ_HEADER).
uint16_t textWidth;	Precomputed text width.
uint16_t textHeight;	Precomputed text height.
GFX_XCHAR * pText;	Text string used.
GFX_ALIGNMENT alignment;	Text alignment

Description

GFX GOL group box struct.

Typedef: GFX_GOL_GROUPBOX

Defines the parameters required for a Group Box Object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_GROUPBOX_STATE Enumeration

Specifies the different states of the Group Box object.

File

gfx_gol_group_box.h

C

```
typedef enum {
  GFX_GOL_GROUPBOX_DISABLED_STATE,
  GFX_GOL_GROUPBOX_DRAW_STATE,
  GFX_GOL_GROUPBOX_HIDE_STATE
} GFX_GOL_GROUPBOX_STATE;
```

Members

Members	Description
GFX_GOL_GROUPBOX_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_GROUPBOX_DRAW_STATE	Draw bit to indicate group box must be redrawn.
GFX_GOL_GROUPBOX_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL group box state.

Typedef: GFX_GOL_GROUPBOX_STATE

This enumeration specifies the different states of the Group Box object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_LISTBOX Structure

Defines the structure used for the List Box object.

File

gfx_gol_list_box.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
```

```
  GFX_GOL_LISTITEM * pItemList;
  GFX_GOL_LISTITEM * pFocusItem;
  uint16_t itemsNumber;
  int16_t scrollY;
  int16_t textHeight;
  GFX_ALIGNMENT alignment;
} GFX_GOL_LISTBOX;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
GFX_GOL_LISTITEM * pItemList;	Pointer to the list of items.
GFX_GOL_LISTITEM * pFocusItem;	Pointer to the focused item.
uint16_t itemsNumber;	Number of items in the list box.
int16_t scrollY;	Scroll displacement for the list.
int16_t textHeight;	Precomputed text height.
GFX_ALIGNMENT alignment;	items alignment.

Description

GFX GOL list box structure.

Typedef: GFX_GOL_LISTBOX

Defines the parameters required for a List Box Object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_LISTBOX_ITEM_STATUS Enumeration

Defines the types used to indicate the status of an item.

File

gfx_gol_list_box.h

C

```
typedef enum {
  GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED,
  GFX_GOL_LISTBOX_ITEM_STATUS_REDRAW
} GFX_GOL_LISTBOX_ITEM_STATUS;
```

Members

Members	Description
GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED	Item is selected.
GFX_GOL_LISTBOX_ITEM_STATUS_REDRAW	Item is to be redrawn.

Description

GFX GOL list box item status.

Typedef: GFX_GOL_LISTBOX_ITEM_STATUS

Defines the types used to indicate the status of an item in the List Box.

Remarks

None.

GFX_GOL_LISTBOX_STATE Enumeration

Specifies the different states of the List Box object.

File

gfx_gol_list_box.h

C

```
typedef enum {
  GFX_GOL_LISTBOX_FOCUSED_STATE,
```

```
  GFX_GOL_LISTBOX_DISABLED_STATE,
  GFX_GOL_LISTBOX_SINGLE_SELECT_STATE,
  GFX_GOL_LISTBOX_DRAW_ITEMS_STATE,
  GFX_GOL_LISTBOX_DRAW_FOCUS_STATE,
  GFX_GOL_LISTBOX_DRAW_STATE,
  GFX_GOL_LISTBOX_HIDE_STATE
} GFX_GOL_LISTBOX_STATE;
```

Members

Members	Description
GFX_GOL_LISTBOX_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_LISTBOX_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_LISTBOX_SINGLE_SELECT_STATE	Property bit to indicate only one item can be selected.
GFX_GOL_LISTBOX_DRAW_ITEMS_STATE	Draw bit to indicate selected items of the object must be redrawn.
GFX_GOL_LISTBOX_DRAW_FOCUS_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_LISTBOX_DRAW_STATE	Draw Bit to indicate object must be redrawn.
GFX_GOL_LISTBOX_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL list box state.

Typedef: GFX_GOL_LISTBOX_STATE

This enumeration specifies the different states of the List Box object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_LISTITEM Structure

The structure that defines each item in the list box.

File

gfx_gol_list_box.h

C

```
typedef struct {
  void * pPrevItem;
  void * pNextItem;
  GFX_GOL_LISTBOX_ITEM_STATUS status;
  GFX_XCHAR * pText;
  GFX_RESOURCE_HDR * pImage;
  uint16_t data;
} GFX_GOL_LISTITEM;
```

Members

Members	Description
void * pPrevItem;	Pointer to the next item
void * pNextItem;	Pointer to the next item
GFX_GOL_LISTBOX_ITEM_STATUS status;	Specifies the status of the item.
GFX_XCHAR * pText;	Pointer to the text for the item
GFX_RESOURCE_HDR * pImage;	Pointer to the image used
uint16_t data;	Some data associated with the item

Description

GFX GOL list item structure.

Typedef: GFX_GOL_LISTITEM

Each item in the list box is described by this structure. The items are arranged as a linked list of this type.

Remarks

None.

GFX_GOL_MESSAGE Structure

Specifies message structure used in the library.

File

gfx_gol.h

C

```
typedef struct {
  uint8_t type;
  uint8_t uiEvent;
  int16_t param1;
  int16_t param2;
} GFX_GOL_MESSAGE;
```

Members

Members	Description
uint8_t type;	Specifies the type of input device.
uint8_t uiEvent;	An event that occurred in the input device.
int16_t param1;	Parameter 1, definition and usage is dependent on the type of input device.
int16_t param2;	Parameter 2, definition and usage is dependent on the type of input device.

Description

GFX GOL Message

Typedef: GFX_GOL_MESSAGE

Specifies message structure used in the library.

- The types must be one of the INPUT_DEVICE_TYPE:
  - TYPE_UNKNOWN
  - TYPE_KEYBOARD
  - TYPE_TOUCHSCREEN
  - TYPE_MOUSE
- uiEvent must be one of the INPUT_DEVICE_EVENT.
  - for touch screen:
    - EVENT_INVALID
    - EVENT_MOVE
    - EVENT_PRESS
    - EVENT_STILLPRESS
    - EVENT_RELEASE
  - for keyboard:
    - EVENT_KEYSCAN (param2 contains scan code)
    - EVENT_KEYCODE (param2 contains character code)
- param1:
  - for touch screen is the x position
  - for keyboard ID of object receiving the message
- param2
  - for touch screen y position
  - for keyboard scan or key code

Remarks

None.

GFX_GOL_METER Structure

Defines the structure used for the Meter object.

File

gfx_gol_meter.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  GFX_XCHAR * pText;
  GFX_GOL_METER_DRAW_TYPE type;
  int16_t value;
  int16_t minValue;
  int16_t maxValue;
  int16_t xCenter;
  int16_t yCenter;
  int16_t radius;
  int16_t xPos;
  int16_t yPos;
  GFX_COLOR color1;
  GFX_COLOR color2;
  GFX_COLOR color3;
  GFX_COLOR color4;
  GFX_COLOR color5;
  GFX_COLOR color6;
  GFX_RESOURCE_HDR * pTitleFont;
  GFX_RESOURCE_HDR * pValueFont;
} GFX_GOL_METER;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see OBJ_HEADER).
GFX_XCHAR * pText;	The text label of the meter.
GFX_GOL_METER_DRAW_TYPE type;	sets the type of the meter
int16_t value;	Current value of the meter.
int16_t minValue;	minimum value the meter can display
int16_t maxValue;	maximum value the meter can display (range is maxValue - minValue)
int16_t xCenter;	The x coordinate center position. This is computed automatically.
int16_t yCenter;	The y coordinate center position. This is computed automatically.
int16_t radius;	Radius of the meter, also defines the needle length.
int16_t xPos;	The current x position of the needle. This is computed automatically.
int16_t yPos;	The current y position of the needle. This is computed automatically.
GFX_COLOR color1;	Arc1 and scale1 color parameter.
GFX_COLOR color2;	Arc2 and scale2 color parameter
GFX_COLOR color3;	Arc3 and scale3 color parameter
GFX_COLOR color4;	Arc4 and scale4 color parameter
GFX_COLOR color5;	Arc5 and scale5 color parameter
GFX_COLOR color6;	Arc6 and scale6 color parameter
GFX_RESOURCE_HDR * pTitleFont;	Pointer to the font used in the title of the meter
GFX_RESOURCE_HDR * pValueFont;	Pointer to the font used in the current reading (if displayed) of the meter

Description

GFX GOL meter structure.

Typedef: GFX_GOL_METER

Defines the parameters required for a Meter Object. Depending on the type selected the meter is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_METER_DRAW_TYPE Enumeration

Specifies the different types of Meter object.

File

gfx_gol_meter.h

C

```
typedef enum {
  GFX_GOL_METER_WHOLE_TYPE,
  GFX_GOL_METER_HALF_TYPE,
  GFX_GOL_METER_QUARTER_TYPE
} GFX_GOL_METER_DRAW_TYPE;
```

Members

Members	Description
GFX_GOL_METER_WHOLE_TYPE	draw circular meter
GFX_GOL_METER_HALF_TYPE	draw semi circle meter
GFX_GOL_METER_QUARTER_TYPE	draw quarter of a circle meter

Description

GFX GOL meter draw type.

Typedef: GFX_GOL_METER_DRAW_TYPE

This enumeration specifies the different typer of the Meter object used in the library.

- GFX_GOL_METER_WHOLE_TYPE - The meter will be drawn using all the arc colors (color1 to color6).
- GFX_GOL_METER_HALF_TYPE - The meter will be drawn using arc colors (color5, color4, color3, color2).
- GFX_GOL_METER_QUARTER_TYPE - The meter will be drawn using arc colors (color3, color2).

Remarks

None.

GFX_GOL_METER_STATE Enumeration

Specifies the different states of the Meter object.

File

gfx_gol_meter.h

C

```
typedef enum {
  GFX_GOL_METER_DISABLED_STATE,
  GFX_GOL_METER_RING_STATE,
  GFX_GOL_METER_ACCURACY_STATE,
  GFX_GOL_METER_UPDATE_DRAW_STATE,
  GFX_GOL_METER_DRAW_STATE,
  GFX_GOL_METER_HIDE_STATE
} GFX_GOL_METER_STATE;
```

Members

Members	Description
GFX_GOL_METER_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_METER_RING_STATE	Property bit for ring type, scales are drawn over the ring. The default state of this state bit is disabled. Only scales are drawn.
GFX_GOL_METER_ACCURACY_STATE	Property bit when set, the values will have accuracy for 1 decimal place.
GFX_GOL_METER_UPDATE_DRAW_STATE	Draw bit to indicate update of the meter-hand only.
GFX_GOL_METER_DRAW_STATE	Draw bit to indicate the whole object must be drawn.
GFX_GOL_METER_HIDE_STATE	Draw bit to indicate the object must be removed from the screen.

Description

GFX GOL meter state.

Typedef: GFX_GOL_METER_STATE

This enumeration specifies the different states of the Meter object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_OBJ_HEADER Structure

Specifies Graphics Object Layer structure used in objects.

File

gfx_gol.h

C

```
typedef struct {
  SYS_MODULE_INDEX gfxIndex;
  uint16_t ID;
  void * pNxtObj;
  GFX_GOL_OBJ_TYPE type;
  uint16_t state;
  uint16_t left;
  uint16_t top;
  uint16_t right;
  uint16_t bottom;
  GFX_GOL_OBJ_SCHEME * pGolScheme;
  DRAW_FUNC DrawObj;
  FREE_FUNC FreeObj;
  ACTIONGET_FUNC actionGet;
  ACTIONSET_FUNC actionSet;
  uint16_t groupNo;
} GFX_GOL_OBJ_HEADER;
```

Members

Members	Description
SYS_MODULE_INDEX gfxIndex;	GFX instance index
uint16_t ID;	Unique id assigned for referencing.
void * pNxtObj;	A pointer to the next object.
GFX_GOL_OBJ_TYPE type;	Identifies the type of GOL object.
uint16_t state;	State of object.
uint16_t left;	Left position of the Object.
uint16_t top;	Top position of the Object.
uint16_t right;	Right position of the Object.
uint16_t bottom;	Bottom position of the Object.
GFX_GOL_OBJ_SCHEME * pGolScheme;	Pointer to the scheme used.
DRAW_FUNC DrawObj;	function pointer to the object's draw function.
FREE_FUNC FreeObj;	function pointer to the object's free function.
ACTIONGET_FUNC actionGet;	function pointer to the object's action get function.
ACTIONSET_FUNC actionSet;	function pointer to the object's action set function.
uint16_t groupNo;	GroupNo to keep track of group for radiobutton.

Description

GFX GOL object header

Typedef: GFX_GOL_OBJ_HEADER

This structure defines the Graphics Object Layer header used in all objects in the Graphics Library.

Remarks

None.

GFX_GOL_OBJ_TYPE Enumeration

Specifies the different object types used in the library.

File

gfx_gol.h

C

```
typedef enum {
  GFX_GOL_ANALOGCLOCK_TYPE,
  GFX_GOL_BUTTON_TYPE,
  GFX_GOL_CHART_TYPE,
  GFX_GOL_CHECKBOX_TYPE,
  GFX_GOL_DIGITALMETER_TYPE,
  GFX_GOL_EDITBOX_TYPE,
  GFX_GOL_GRID_TYPE,
  GFX_GOL_GROUPBOX_TYPE,
  GFX_GOL_LISTBOX_TYPE,
  GFX_GOL_METER_TYPE,
  GFX_GOL_PICTURECONTROL_TYPE,
  GFX_GOL_PROGRESSBAR_TYPE,
  GFX_GOL_RADIOBUTTON_TYPE,
  GFX_GOL_SCROLLBAR_TYPE,
  GFX_GOL_STATICTEXT_TYPE,
  GFX_GOL_TEXTENTRY_TYPE,
  GFX_GOL_WINDOW_TYPE,
  GFX_GOL_CUSTOM_TYPE,
  GFX_GOL_SURFACE_TYPE,
  GFX_GOL_UNKNOWN_TYPE
} GFX_GOL_OBJ_TYPE;
```

Members

Members	Description
GFX_GOL_ANALOGCLOCK_TYPE	Type defined for Analog Clock Object.
GFX_GOL_BUTTON_TYPE	Type defined for Button Object.
GFX_GOL_CHART_TYPE	Type defined for Chart Object.
GFX_GOL_CHECKBOX_TYPE	Type defined for Check Box Object.
GFX_GOL_DIGITALMETER_TYPE	Type defined for Digital Meter Object.
GFX_GOL_EDITBOX_TYPE	Type defined for Edit Box Object.
GFX_GOL_GRID_TYPE	Type defined for Grid Object.
GFX_GOL_GROUPBOX_TYPE	Type defined for Group Box Object.
GFX_GOL_LISTBOX_TYPE	Type defined for List Box Object.
GFX_GOL_METER_TYPE	Type defined for Meter Object.
GFX_GOL_PICTURECONTROL_TYPE	Type defined for Picture Control Object.
GFX_GOL_PROGRESSBAR_TYPE	Type defined for Progress Bar Object.
GFX_GOL_RADIOBUTTON_TYPE	Type defined for Radio Button Object.
GFX_GOL_SCROLLBAR_TYPE	Type defined for Slider or Scroll Bar Object.
GFX_GOL_STATICTEXT_TYPE	Type defined for Static Text Object.
GFX_GOL_TEXTENTRY_TYPE	Type defined for Text-Entry Object.
GFX_GOL_WINDOW_TYPE	Type defined for Window Object.
GFX_GOL_CUSTOM_TYPE	Type defined for Custom Object.
GFX_GOL_UNKNOWN_TYPE	Type is undefined and not supported by the library.

Description

GFX GOL object type

Typedef: GFX_GOL_OBJ_TYPE

This enumeration specifies the different object types used in the library.

Remarks

None.

GFX_GOL_PROGRESSBAR Structure

Defines the structure used for the Progress Bar object.

File

gfx_gol_progress_bar.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t pos;
  uint16_t prevPos;
  uint16_t range;
} GFX_GOL_PROGRESSBAR;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
uint16_t pos;	Current progress position.
uint16_t prevPos;	Previous progress position.
uint16_t range;	Sets the range of the object.

Description

GFX GOL progress bar structure.

Typedef: GFX_GOL_PROGRESSBAR

Defines the parameters required for a Progress Bar object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_PROGRESSBAR_STATE Enumeration

Specifies the different states of the Progress Bar object.

File

gfx_gol_progress_bar.h

C

```
typedef enum {
  GFX_GOL_PROGRESSBAR_DISABLED_STATE,
  GFX_GOL_PROGRESSBAR_VERTICAL_STATE,
  GFX_GOL_PROGRESSBAR_NOPROGRESS_STATE,
  GFX_GOL_PROGRESSBAR_DRAW_BAR_STATE,
  GFX_GOL_PROGRESSBAR_DRAW_STATE,
  GFX_GOL_PROGRESSBAR_HIDE_STATE
} GFX_GOL_PROGRESSBAR_STATE;
```

Members

Members	Description
GFX_GOL_PROGRESSBAR_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_PROGRESSBAR_VERTICAL_STATE	Property bit for vertical orientation. • When this state bit is 0 - object is rendered with horizontal orientation. • When this state bit is 1 - object is rendered with vertical orientation.
GFX_GOL_PROGRESSBAR_NOPROGRESS_STATE	Property bit that will suppress rendering of progress in text.
GFX_GOL_PROGRESSBAR_DRAW_BAR_STATE	Draw bit to indicate that the progress bar portion must be redrawn.
GFX_GOL_PROGRESSBAR_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_PROGRESSBAR_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL progress bar state.

Typedef: GFX_GOL_PROGRESSBAR_STATE

This enumeration specifies the different states of the Progress Bar object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_RADIOBUTTON Structure

Defines the structure used for the Radio Button object.

File

gfx_gol_radio_button.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  GFX_GOL_OBJ_HEADER * pHead;
  GFX_GOL_OBJ_HEADER * pNext;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
} GFX_GOL_RADIOBUTTON;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see OBJ_HEADER).
GFX_GOL_OBJ_HEADER * pHead;	Pointer to the first Radio Button in the group
GFX_GOL_OBJ_HEADER * pNext;	Pointer to the next Radio Button in the group
uint16_t textHeight;	Precomputed text height
GFX_XCHAR * pText;	Pointer to the text
GFX_ALIGNMENT alignment;	text alignment

Description

GFX GOL Radio button structure.

Typedef: GFX_GOL_RADIOBUTTON

Defines the parameters required for a Radio Button object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_RADIOBUTTON_STATE Enumeration

Specifies the different states of the Radio Button object.

File

gfx_gol_radio_button.h

C

```
typedef enum {
  GFX_GOL_RADIOBUTTON_FOCUSED_STATE,
  GFX_GOL_RADIOBUTTON_DISABLED_STATE,
  GFX_GOL_RADIOBUTTON_CHECKED_STATE,
  GFX_GOL_RADIOBUTTON_GROUP_STATE,
  GFX_GOL_RADIOBUTTON_DRAW_CHECK_STATE,
  GFX_GOL_RADIOBUTTON_DRAW_FOCUS_STATE,
  GFX_GOL_RADIOBUTTON_DRAW_STATE,
  GFX_GOL_RADIOBUTTON_HIDE_STATE
} GFX_GOL_RADIOBUTTON_STATE;
```

Members

Members	Description
GFX_GOL_RADIOBUTTON_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_RADIOBUTTON_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_RADIOBUTTON_CHECKED_STATE	Property bit to indicate Radio Button is checked.

GFX_GOL_RADIOBUTTON_GROUP_STATE	Bit to indicate the first Radio Button in the group. Each group MUST have this bit set for its first member even for a single member group. This means that any independent or stand alone Radio Button, the GFX_GOL_RADIOBUTTON_GROUP_STATE bit must be always set.
GFX_GOL_RADIOBUTTON_DRAW_CHECK_STATE	Draw bit to indicate check mark should be redrawn.
GFX_GOL_RADIOBUTTON_DRAW_FOCUS_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_RADIOBUTTON_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_RADIOBUTTON_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL radio button state.

Typedef: GFX_GOL_RADIOBUTTON_STATE

This enumeration specifies the different states of the Radio Button object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_SCROLLBAR Structure

Defines the structure used for the Scroll Bar object.

File

gfx_gol_scroll_bar.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  int16_t currPos;
  uint16_t prevPos;
  uint16_t range;
  int16_t pos;
  uint16_t page;
  uint16_t thWidth;
  uint16_t thHeight;
} GFX_GOL_SCROLLBAR;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
int16_t currPos;	Position of the slider relative to minimum.
uint16_t prevPos;	Previous position of the slider relative to minimum.
uint16_t range;	User defined range of the slider. Minimum value at 0 and maximum at 0x7FFF.
int16_t pos;	Position of the slider in range domain.
uint16_t page;	User specified resolution to incrementally change the position in range domain.
uint16_t thWidth;	Thumb width. This is computed internally. User must not change this value.
uint16_t thHeight;	Thumb width. This is computed internally. User must not change this value.

Description

GFX GOL scroll bar struct.

Typedef: GFX_GOL_SCROLLBAR

Defines the parameters required for a Scroll Bar object. Object is drawn with the defined shape parameters and values set on the given fields. Depending on the GFX_GOL_SCROLLBAR_SLIDER_MODE_STATE state bit slider or scrollbar mode is set. If GFX_GOL_SCROLLBAR_SLIDER_MODE_STATE is set, mode is slider; if not set mode is scroll bar. For scrollbar mode, focus rectangle is not drawn.

Remarks

None.

GFX_GOL_SCROLLBAR_STATE Enumeration

Specifies the different states of the Scroll Bar object.

File

gfx_gol_scroll_bar.h

C

```
typedef enum {
  GFX_GOL_SCROLLBAR_FOCUSED_STATE,
  GFX_GOL_SCROLLBAR_DISABLED_STATE,
  GFX_GOL_SCROLLBAR_VERTICAL_STATE,
  GFX_GOL_SCROLLBAR_SLIDER_MODE_STATE,
  GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE,
  GFX_GOL_SCROLLBAR_DRAW_FOCUS_STATE,
  GFX_GOL_SCROLLBAR_DRAW_STATE,
  GFX_GOL_SCROLLBAR_HIDE_STATE
} GFX_GOL_SCROLLBAR_STATE;
```

Members

Members	Description
GFX_GOL_SCROLLBAR_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_SCROLLBAR_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_SCROLLBAR_VERTICAL_STATE	Property bit to indicate the scroll bar is drawn with vertical orientation
GFX_GOL_SCROLLBAR_SLIDER_MODE_STATE	Property bit to indicate the scroll bar is in slider mode.
GFX_GOL_SCROLLBAR_DRAW_THUMB_STATE	Draw bit to indicate that only the thumb area will be redrawn.
GFX_GOL_SCROLLBAR_DRAW_FOCUS_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_SCROLLBAR_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_SCROLLBAR_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL scroll bar state.

Typedef: GFX_GOL_SCROLLBAR_STATE

This enumeration specifies the different states of the Scroll Bar object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_STATICTEXT Structure

Defines the structure used for the Static Text object.

File

gfx_gol_static_text.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
} GFX_GOL_STATICTEXT;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see OBJ_HEADER).
GFX_XCHAR * pText;	The pointer to text used.
GFX_ALIGNMENT alignment;	text alignment

Description

GFX GOL static text structure.

Typedef: GFX_GOL_STATICTEXT

Defines the parameters required for a Static Text object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_STATICTEXT_STATE Enumeration

Specifies the different states of the Static Text object.

File

gfx_gol_static_text.h

C

```
typedef enum {
  GFX_GOL_STATICTEXT_DISABLED_STATE,
  GFX_GOL_STATICTEXT_FRAME_STATE,
  GFX_GOL_STATICTEXT_NOBACKGROUND_STATE,
  GFX_GOL_STATICTEXT_DRAW_STATE,
  GFX_GOL_STATICTEXT_HIDE_STATE
} GFX_GOL_STATICTEXT_STATE;
```

Members

Members	Description
GFX_GOL_STATICTEXT_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_STATICTEXT_FRAME_STATE	Property bit to indicate frame is enabled.
GFX_GOL_STATICTEXT_NOBACKGROUND_STATE	Property bit to indicate background is enabled.
GFX_GOL_STATICTEXT_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_STATICTEXT_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL static text state.

Typedef: GFX_GOL_STATICTEXT_STATE

This enumeration specifies the different states of the Static Text object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_TEXTENTRY Structure

Defines the structure used for the Text Entry object.

File

gfx_gol_text_entry.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t horizontalKeys;
  uint16_t verticalKeys;
  GFX_XCHAR * pTeOutput;
  GFX_ALIGNMENT alignment;
  uint16_t CurrentLength;
  uint16_t outputLenMax;
  GFX_GOL_TEXTENTRY_KEYMEMBER * pActiveKey;
  GFX_GOL_TEXTENTRY_KEYMEMBER * pHeadOfList;
```

```
  GFX_RESOURCE_HDR * pDisplayFont;
} GFX_GOL_TEXTENTRY;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all objects (see GFX_GOL_OBJ_HEADER).
uint16_t horizontalKeys;	Number of horizontal keys.
uint16_t verticalKeys;	Number of vertical keys.
GFX_XCHAR * pTeOutput;	Pointer to the buffer assigned by the user which holds the text shown in the edit box.
GFX_ALIGNMENT alignment;	Defines the text alignment.
uint16_t CurrentLength;	Current length of the string in the buffer. The maximum value of this is equal to outputLenMax. User creates and manages the buffer. Buffer can also be managed using the APIs provided to add a character, delete the last character or clear the buffer.
uint16_t outputLenMax;	Maximum expected length of output buffer. Object will update this parameter when adding, removing characters or clearing the buffer and switching buffers.
GFX_GOL_TEXTENTRY_KEYMEMBER * pActiveKey;	Pointer to the active key. This is only used by the object. User must not change the value of this parameter directly.
GFX_GOL_TEXTENTRY_KEYMEMBER * pHeadOfList;	Pointer to head of the list
GFX_RESOURCE_HDR * pDisplayFont;	Pointer to the font used in displaying the text.

Description

GFX GOL text entry structure.

Typedef: GFX_GOL_TEXTENTRY

Defines the parameters required for a Text Entry object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE Enumeration

Specifies the different commands available for command keys of the Text Entry object.

File

gfx_gol_text_entry.h

C

```
typedef enum {
  GFX_GOL_TEXTENTRY_NONE_COM,
  GFX_GOL_TEXTENTRY_DELETE_COM,
  GFX_GOL_TEXTENTRY_SPACE_COM,
  GFX_GOL_TEXTENTRY_ENTER_COM
} GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE;
```

Members

Members	Description
GFX_GOL_TEXTENTRY_NONE_COM	This type is used when the key is not assigned to any command.
GFX_GOL_TEXTENTRY_DELETE_COM	This type is used to assign a "delete" command on a key.
GFX_GOL_TEXTENTRY_SPACE_COM	This type is used to assign an insert "space" command on a key.
GFX_GOL_TEXTENTRY_ENTER_COM	This type is used to assign an "enter" carriage return) command on a key.

Description

GFX GOL text entry key command type.

Typedef: GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE

This enumeration specifies the different commands available for the command keys of the object.

The GFX_GOL_TEXTENTRY_ENTER_COM command can be used to customize application code in the message callback function. Use the returned translated GFX_GOL_TEXTENTRY_ACTION_ENTER to detect the key pressed was assigned the enter command. Refer to GFX_GOL_TextEntryActionGet() for details.

Remarks

None.

GFX_GOL_TEXTENTRY_STATE Enumeration

Specifies the different states of the Text Entry object.

File

gfx_gol_text_entry.h

C

```
typedef enum {
  GFX_GOL_TEXTENTRY_DISABLED_STATE,
  GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE,
  GFX_GOL_TEXTENTRY_ECHO_HIDE_STATE,
  GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE,
  GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE,
  GFX_GOL_TEXTENTRY_DRAW_STATE,
  GFX_GOL_TEXTENTRY_HIDE_STATE
} GFX_GOL_TEXTENTRY_STATE;
```

Members

Members	Description
GFX_GOL_TEXTENTRY_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE	Property bit for press state.
GFX_GOL_TEXTENTRY_ECHO_HIDE_STATE	Bit to hide the entered characters and instead echo "*" characters to the display.
GFX_GOL_TEXTENTRY_UPDATE_TEXT_STATE	Draw bit to indicate redraw of the text displayed is needed.
GFX_GOL_TEXTENTRY_UPDATE_KEY_STATE	Draw bit to indicate redraw of a key is needed.
GFX_GOL_TEXTENTRY_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_TEXTENTRY_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL text entry state.

Typedef: GFX_GOL_TEXTENTRY_STATE

This enumeration specifies the different states of the Text Entry object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_TRANSLATED_ACTION Enumeration

Specifies the different object actions supported in the library.

File

gfx_gol.h

C

```
typedef enum {
  GFX_GOL_OBJECT_ACTION_INVALID = 0x3500,
  GFX_GOL_OBJECT_ACTION_PASSIVE,
  GFX_GOL_ANALOGCLOCK_ACTION_PRESSED,
  GFX_GOL_ANALOGCLOCK_ACTION_RELEASED,
  GFX_GOL_BUTTON_ACTION_PRESSED,
  GFX_GOL_BUTTON_ACTION_STILLPRESSED,
  GFX_GOL_BUTTON_ACTION_RELEASED,
  GFX_GOL_BUTTON_ACTION_CANCELPRESS,
  GFX_GOL_CHART_ACTION_SELECTED,
  GFX_GOL_CHECKBOX_ACTION_CHECKED,
  GFX_GOL_CHECKBOX_ACTION_UNCHECKED,
  GFX_GOL_CUSTOMCONTROL_ACTION_SELECTED,
  GFX_GOL_DIGITALMETER_ACTION_SELECTED,
  GFX_GOL_EDITBOX_ACTION_ADD_CHAR,
```

```
  GFX_GOL_EDITBOX_ACTION_DEL_CHAR,
  GFX_GOL_EDITBOX_ACTION_TOUCHSCREEN,
  GFX_GOL_GRID_ACTION_TOUCHED,
  GFX_GOL_GRID_ACTION_ITEM_SELECTED,
  GFX_GOL_GRID_ACTION_UP,
  GFX_GOL_GRID_ACTION_DOWN,
  GFX_GOL_GRID_ACTION_LEFT,
  GFX_GOL_GRID_ACTION_RIGHT,
  GFX_GOL_GROUPBOX_ACTION_SELECTED,
  GFX_GOL_LISTBOX_ACTION_SELECTED,
  GFX_GOL_LISTBOX_ACTION_MOVE,
  GFX_GOL_LISTBOX_ACTION_TOUCHSCREEN,
  GFX_GOL_METER_ACTION_SET,
  GFX_GOL_PICTURECONTROL_ACTION_SELECTED,
  GFX_GOL_PROGRESSBAR_ACTION_SELECTED,
  GFX_GOL_RADIOBUTTON_ACTION_CHECKED,
  GFX_GOL_SCROLLBAR_ACTION_INC,
  GFX_GOL_SCROLLBAR_ACTION_DEC,
  GFX_GOL_STATICTEXT_ACTION_SELECTED,
  GFX_GOL_TEXTENTRY_ACTION_RELEASED,
  GFX_GOL_TEXTENTRY_ACTION_PRESSED,
  GFX_GOL_TEXTENTRY_ACTION_ADD_CHAR,
  GFX_GOL_TEXTENTRY_ACTION_DELETE,
  GFX_GOL_TEXTENTRY_ACTION_SPACE,
  GFX_GOL_TEXTENTRY_ACTION_ENTER,
  GFX_GOL_WINDOW_ACTION_CLIENT,
  GFX_GOL_WINDOW_ACTION_TITLE
} GFX_GOL_TRANSLATED_ACTION;
```

Members

Members	Description
GFX_GOL_OBJECT_ACTION_INVALID = 0x3500	Invalid message response.
GFX_GOL_OBJECT_ACTION_PASSIVE	Passive message response. No change in object needed.
GFX_GOL_ANALOGCLOCK_ACTION_PRESSED	Analog Clock Pressed Action
GFX_GOL_ANALOGCLOCK_ACTION_RELEASED	Analog Clock Released Action
GFX_GOL_BUTTON_ACTION_PRESSED	Button pressed action ID.
GFX_GOL_BUTTON_ACTION_STILLPRESSED	Button is continuously pressed ID.
GFX_GOL_BUTTON_ACTION_RELEASED	Button released action ID.
GFX_GOL_BUTTON_ACTION_CANCELPRESS	Button released action ID with button press canceled.
GFX_GOL_CHART_ACTION_SELECTED	Chart selected action ID
GFX_GOL_CHECKBOX_ACTION_CHECKED	Check Box check action ID.
GFX_GOL_CHECKBOX_ACTION_UNCHECKED	Check Box uncheck action ID.
GFX_GOL_CUSTOMCONTROL_ACTION_SELECTED	Custom Control selected action ID.
GFX_GOL_DIGITALMETER_ACTION_SELECTED	Digital Meter selected action ID.
GFX_GOL_EDITBOX_ACTION_ADD_CHAR	Edit Box insert character action ID.
GFX_GOL_EDITBOX_ACTION_DEL_CHAR	Edit Box remove character action ID.
GFX_GOL_EDITBOX_ACTION_TOUCHSCREEN	Edit Box touchscreen selected action ID.
GFX_GOL_GRID_ACTION_TOUCHED	Grid item touched action ID.
GFX_GOL_GRID_ACTION_ITEM_SELECTED	Grid item selected action ID.
GFX_GOL_GRID_ACTION_UP	Grid up action ID.
GFX_GOL_GRID_ACTION_DOWN	Grid down action ID.
GFX_GOL_GRID_ACTION_LEFT	Grid left action ID.
GFX_GOL_GRID_ACTION_RIGHT	Grid right action ID.
GFX_GOL_GROUPBOX_ACTION_SELECTED	Group Box selected action ID.
GFX_GOL_LISTBOX_ACTION_SELECTED	List Box item select action ID.
GFX_GOL_LISTBOX_ACTION_MOVE	List Box item move action ID.
GFX_GOL_LISTBOX_ACTION_TOUCHSCREEN	List Box touchscreen selected action ID.
GFX_GOL_METER_ACTION_SET	Meter set value action ID.
GFX_GOL_PICTURECONTROL_ACTION_SELECTED	Picture selected action ID.
GFX_GOL_PROGRESSBAR_ACTION_SELECTED	Progress Bar selected action ID.
GFX_GOL_RADIOBUTTON_ACTION_CHECKED	Radio Button check action ID.
GFX_GOL_SCROLLBAR_ACTION_INC	Slider or Scroll Bar increment action ID.

GFX_GOL_SCROLLBAR_ACTION_DEC	Slider or Scroll Bar decrement action ID.
GFX_GOL_STATICTEXT_ACTION_SELECTED	Static Text selected action ID.
GFX_GOL_TEXTENTRY_ACTION_RELEASED	TextEntry released action ID
GFX_GOL_TEXTENTRY_ACTION_PRESSED	TextEntry pressed action ID
GFX_GOL_TEXTENTRY_ACTION_ADD_CHAR	TextEntry add character action ID
GFX_GOL_TEXTENTRY_ACTION_DELETE	TextEntry delete character action ID
GFX_GOL_TEXTENTRY_ACTION_SPACE	TextEntry add space character action ID
GFX_GOL_TEXTENTRY_ACTION_ENTER	TextEntry enter action ID
GFX_GOL_WINDOW_ACTION_CLIENT	Window client area selected action ID.
GFX_GOL_WINDOW_ACTION_TITLE	Window title bar selected action ID.

Description

GFX GOL Translated Action

Typedef: GFX_GOL_TRANSLATED_ACTION

This enumeration specifies the different object actions supported in the library.

Remarks

None.

GFX_GOL_WINDOW Structure

Defines the structure used for the Window object.

File

gfx_gol_window.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
  GFX_RESOURCE_HDR * pImage;
} GFX_GOL_WINDOW;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all objects (see GFX_GOL_OBJ_HEADER).
uint16_t textHeight;	Precomputed text height
GFX_XCHAR * pText;	Pointer to the title text
GFX_ALIGNMENT alignment;	text alignment
GFX_RESOURCE_HDR * pImage;	Pointer to the image for the title bar

Description

GFX GOL window structure.

Typedef: GFX_GOL_WINDOW

Defines the parameters required for a Window object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_WINDOW_STATE Enumeration

Specifies the different states of the Window object.

File

gfx_gol_window.h

C

```
typedef enum {
```

```
  GFX_GOL_WINDOW_FOCUSED_STATE,
  GFX_GOL_WINDOW_DISABLED_STATE,
  GFX_GOL_WINDOW_DRAW_TITLE_STATE,
  GFX_GOL_WINDOW_DRAW_CLIENT_STATE,
  GFX_GOL_WINDOW_DRAW_STATE,
  GFX_GOL_WINDOW_HIDE_STATE
} GFX_GOL_WINDOW_STATE;
```

Members

Members	Description
GFX_GOL_WINDOW_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_WINDOW_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_WINDOW_DRAW_TITLE_STATE	Draw bit to indicate title area must be redrawn.
GFX_GOL_WINDOW_DRAW_CLIENT_STATE	Draw bit to indicate client area must be redrawn.
GFX_GOL_WINDOW_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_WINDOW_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL window state.

Typedef: GFX_GOL_WINDOW_STATE

This enumeration specifies the different states of the Window object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GOL_PANEL_PARAM Structure

Specifies panel parameters.

File

gfx_gol.h

C

```
typedef struct {
  int16_t panelLeft;
  int16_t panelTop;
  int16_t panelRight;
  int16_t panelBottom;
  GFX_COLOR panelFaceColor;
  GFX_COLOR panelEmbossLtColor;
  GFX_COLOR panelEmbossDkColor;
  uint16_t panelEmbossSize;
  uint16_t panelRadius;
  GFX_RESOURCE_HDR * pPanelImage;
  GFX_FILL_STYLE panelFillStyle;
  GFX_COLOR panelGradientStartColor;
  GFX_COLOR panelGradientEndColor;
  uint16_t panelAlpha;
} GOL_PANEL_PARAM;
```

Members

Members	Description
int16_t panelLeft;	Panel left coordinate.
int16_t panelTop;	Panel top coordinate.
int16_t panelRight;	Panel right coordinate.
int16_t panelBottom;	Panel Bottom coordinate.
GFX_COLOR panelFaceColor;	Panel Face color.
GFX_COLOR panelEmbossLtColor;	Panel Emboss light color.
GFX_COLOR panelEmbossDkColor;	Panel Emboss dark color.

uint16_t panelEmbossSize;	Panel Emboss size.
uint16_t panelRadius;	Panel radius.
GFX_RESOURCE_HDR * pPanelImage;	Panel image resource.
GFX_FILL_STYLE panelFillStyle;	Panel fill style.
GFX_COLOR panelGradientStartColor;	Panel gradient start color.
GFX_COLOR panelGradientEndColor;	Panel gradient end color .
uint16_t panelAlpha;	Panel alpha

Description

GOL panel parameter

Typedef: GOL_PANEL_PARAM

This structure defines the panel parameters when rendering a panel.

Remarks

None.

GFX_GOL_EDITBOX_STATE Enumeration

Specifies the different states of the Edit Box object.

File

gfx_gol_edit_box.h

C

```
typedef enum {
  GFX_GOL_EDITBOX_FOCUSED_STATE,
  GFX_GOL_EDITBOX_DISABLED_STATE,
  GFX_GOL_EDITBOX_ENABLE_CARET_STATE,
  GFX_GOL_EDITBOX_ENABLE_BAR_CARET_STATE,
  GFX_GOL_EDITBOX_ENABLE_BLOCK_CARET_STATE,
  GFX_GOL_EDITBOX_ENABLE_UNDERSCORE_CARET_STATE,
  GFX_GOL_EDITBOX_DRAW_CARET_STATE,
  GFX_GOL_EDITBOX_DRAW_FOCUS_STATE,
  GFX_GOL_EDITBOX_DRAW_STATE,
  GFX_GOL_EDITBOX_HIDE_STATE
} GFX_GOL_EDITBOX_STATE;
```

Members

Members	Description
GFX_GOL_EDITBOX_FOCUSED_STATE	Property bit for focus state. Cursor caret will be drawn when GFX_GOL_EDITBOX_ENABLE_CARET_STATE is also set.
GFX_GOL_EDITBOX_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_EDITBOX_ENABLE_CARET_STATE	Property bit to indicate cursor caret will always be shown.
GFX_GOL_EDITBOX_ENABLE_BAR_CARET_STATE	property bit to indicate bar type caret
GFX_GOL_EDITBOX_ENABLE_BLOCK_CARET_STATE	property bit to indicate block type caret
GFX_GOL_EDITBOX_ENABLE_UNDERSCORE_CARET_STATE	property bit to indicate underscore type caret
GFX_GOL_EDITBOX_DRAW_CARET_STATE	Draw bit to indicate the cursor caret will be drawn if GFX_GOL_EDITBOX_FOCUSED_STATE state bit is set and erase when GFX_GOL_EDITBOX_FOCUSED_STATE state bit is not set.
GFX_GOL_EDITBOX_DRAW_FOCUS_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_EDITBOX_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_EDITBOX_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL edit box state.

Typedef: GFX_GOL_EDITBOX_STATE

This enumeration specifies the different states of the Edit Box object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_PICTURECONTROL Structure

Defines the structure used for the Picture Control object.

File

gfx_gol_picture.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  GFX_RESOURCE_HDR * pImage;
  int8_t scaleFactor;
  uint16_t imageLeft;
  uint16_t imageTop;
  uint16_t imageRight;
  uint32_t * stream;
  uint8_t count;
  uint8_t delay;
  uint16_t imageBottom;
  GFX_PARTIAL_IMAGE_PARAM partial;
} GFX_GOL_PICTURECONTROL;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
GFX_RESOURCE_HDR * pImage;	Pointer to the image
int8_t scaleFactor;	Scale factor for the bitmap
uint16_t imageLeft;	image left position when drawn
uint16_t imageTop;	image top position when drawn
uint16_t imageRight;	image right position when drawn
uint8_t count;	Count for the number of bitmaps to be streamed
uint8_t delay;	Delay in between the streaming of bitmaps
uint16_t imageBottom;	image bottom position when drawn
GFX_PARTIAL_IMAGE_PARAM partial;	structure containing partial image data

Description

GFX GOL picture control structure.

Typedef: GFX_GOL_PICTURECONTROL

Defines the parameters required for a Picture Control object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_PICTURECONTROLCONTROL_STATE Enumeration

Specifies the different states of the Picture Control object.

File

gfx_gol_picture.h

C

```
typedef enum {
  GFX_GOL_PICTURECONTROL_DISABLED_STATE,
  GFX_GOL_PICTURECONTROL_FRAME_STATE,
  GFX_GOL_PICTURECONTROL_STREAM_STATE,
  GFX_GOL_PICTURECONTROL_DRAW_STATE,
  GFX_GOL_PICTURECONTROL_HIDE_STATE
} GFX_GOL_PICTURECONTROLCONTROL_STATE;
```

Members

Members	Description
GFX_GOL_PICTURECONTROL_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_PICTURECONTROL_FRAME_STATE	Property bit to indicate that the object will have a frame.
GFX_GOL_PICTURECONTROL_STREAM_STATE	Property bit to indicate Picture is streaming. This feature is available only when the hardware can support streaming of images.
GFX_GOL_PICTURECONTROL_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_PICTURECONTROL_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL picture control, control state.

Typedef: GFX_GOL_PICTURECONTROLCONTROL_STATE

This enumeration specifies the different states of the Picture Control object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_COMMON_STATE_BITS Enumeration

Common Object States.

File

gfx_gol.h

C

```
typedef enum {
  GFX_GOL_FOCUSED,
  GFX_GOL_DISABLED,
  GFX_GOL_HIDE,
  GFX_GOL_DRAW,
  GFX_GOL_DRAW_FOCUS,
  GFX_GOL_DRAW_UPDATE
} GFX_GOL_COMMON_STATE_BITS;
```

Members

Members	Description
GFX_GOL_FOCUSED	Focus state bit
GFX_GOL_DISABLED	Disabled state bit.
GFX_GOL_HIDE	Object hide state bit. Object will be hidden from the screen by drawing over it the common background color.
GFX_GOL_DRAW	Object redraw state bits. The whole Object must be redrawn.
GFX_GOL_DRAW_FOCUS	Focus redraw state bit. The focus rectangle must be redrawn.
GFX_GOL_DRAW_UPDATE	Partial Object redraw state bits. A part or parts of the Object must be redrawn to show updated state.

Description

GFX GOL common object states

Typedef: Common Object States

The following macros defines the common Object State bits.

Remarks

None.

GFX_GOL_MESSAGE_CALLBACK_FUNC Type

Message callback function definition. This application defined function allows the application to perform application specific processing of user

messages.

File

gfx_gol.h

C

```
typedef bool (* GFX_GOL_MESSAGE_CALLBACK_FUNC)(GFX_GOL_TRANSLATED_ACTION, GFX_GOL_OBJ_HEADER *,
GFX_GOL_MESSAGE *);
```

Returns

true - When true is returned, the object will set its state depending on the translated messages. false - When false is returned, the object will not process the translated message and will assume the application has performed necessary action on the message.

Description

GFX GOL message callback function pointer

This application defined function is called by the GFX_GOL_ObjectMessage() function allowing the application the opportunity to process the user messages and customize object behavior as well as application controlled functions.

GFX_GOL_ObjectMessage() calls this function when a valid message for an object in the active list is received. Application implements any action for the message in this callback function. If this callback function returns true, the message for the object will be processed using the default action of the object. If false is returned, the default action will not be performed. In this case, it is assumed that this callback function has performed the appropriate changes to the states of the objects.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
GFX_GOL_TRANSLATED_ACTION	Translated message for the object
GFX_GOL_OBJ_HEADER *	Pointer to the object that processed the message.
GFX_GOL_MESSAGE *	Pointer to the message from user.

Function

typedef bool (*GFX_GOL_MESSAGE_CALLBACK_FUNC)
(
GFX_GOL_TRANSLATED_ACTION,
GFX_GOL_OBJ_HEADER *,
GFX_GOL_MESSAGE *
);

GFX_GOL_TEXTENTRY_KEYMEMBER Structure

Defines the structure used to describe a key in the Text Entry object.

File

gfx_gol_text_entry.h

C

```
typedef struct {
  uint16_t left;
  uint16_t top;
  uint16_t right;
  uint16_t bottom;
  uint16_t index;
  uint16_t state;
  bool update;
  GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE command;
  GFX_XCHAR * pKeyName;
  int16_t textWidth;
  int16_t textHeight;
  void * pNextKey;
} GFX_GOL_TEXTENTRY_KEYMEMBER;
```

Members

Members	Description
uint16_t left;	Left position of the key
uint16_t top;	Top position of the key
uint16_t right;	Right position of the key
uint16_t bottom;	Bottom position of the key
uint16_t index;	Index of the key in the list
uint16_t state;	State of the key. Either Pressed (GFX_GOL_TEXTENTRY_KEY_PRESSED_STATE) or Released (0)
bool update;	flag to indicate key is to be redrawn with the current state
GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE command;	Command of the key. Either GFX_GOL_TEXTENTRY_DELETE_COM, GFX_GOL_TEXTENTRY_SPACE_COM or GFX_GOL_TEXTENTRY_ENTER_COM.
GFX_XCHAR * pKeyName;	Pointer to the custom text assigned to the key. This is displayed over the face of the key.
int16_t textWidth;	Computed text width, done at creation. Used to predict size and position of text on the key face.
int16_t textHeight;	Computed text height, done at creation. Used to predict size and position of text on the key face.
void * pNextKey;	Pointer to the next key parameters.

Description

GFX GOL text entry key member structure.

Typedef: GFX_GOL_TEXTENTRY_KEYMEMBER

Defines the structure used to describe a key in the Text Entry object. Strings displayed on each key is assigned here as well as the commands if key is assigned a command key.

Remarks

None.

GFX_GOL_CUSTOMCONTROL_STATE Enumeration

Specifies the different states of the Custom Control object.

File

gfx_gol_custom_control.h

C

```
typedef enum {
  GFX_GOL_CUSTOMCONTROL_DISABLED_STATE,
  GFX_GOL_CUSTOMCONTROL_DRAW_BAR_STATE,
  GFX_GOL_CUSTOMCONTROL_DRAW_STATE,
  GFX_GOL_CUSTOMCONTROL_HIDE_STATE
} GFX_GOL_CUSTOMCONTROL_STATE;
```

Members

Members	Description
GFX_GOL_CUSTOMCONTROL_DISABLED_STATE	Property bit to indicate object is disabled.
GFX_GOL_CUSTOMCONTROL_DRAW_BAR_STATE	Draw bit to indicate focus must be redrawn.
GFX_GOL_CUSTOMCONTROL_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_CUSTOMCONTROL_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

GFX GOL custom control state.

Typedef: GFX_GOL_CUSTOMCONTROL_STATE

This enumeration specifies the different states of the Custom Control object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_CLIENT_STATUS Enumeration

Enumerated data type that identifies the GFX Module Client Status.

File

gfx_gol.h

C

```
typedef enum {
  GFX_GOL_CLIENT_STATUS_CLOSED,
  GFX_GOL_CLIENT_STATUS_READY
} GFX_GOL_CLIENT_STATUS;
```

Members

Members	Description
GFX_GOL_CLIENT_STATUS_CLOSED	Client is closed or the specified handle is invalid
GFX_GOL_CLIENT_STATUS_READY	Client is ready

Description

GFX GOL Client Status

This enumeration defines the possible status of the GFX Module Client.

Remarks

None.

GFX_GOL_OBJECT_TASK Enumeration

Lists the different states that GFX GOL task routine can have.

File

gfx_gol.h

C

```
typedef enum {
  GFX_GOL_TASK_STATE_OPEN_MODULE,
  GFX_GOL_TASK_STATE_RUNNING
} GFX_GOL_OBJECT_TASK;
```

Members

Members	Description
GFX_GOL_TASK_STATE_OPEN_MODULE	open module for client
GFX_GOL_TASK_STATE_RUNNING	run module for client

Description

GFX GOL task states

This enumeration lists the different states that GFX GOL task routine can have.

Remarks

None.

GFX_GOL_STATES Enumeration

Defines the various states that can be achieved by the GFX GOL operation.

File

gfx_gol.h

C

```
typedef enum {
  GFX_GOL_STATE_BUSY,
  GFX_GOL_STATE_INIT,
  GFX_GOL_STATE_INITIALIZED
} GFX_GOL_STATES;
```

Members

Members	Description
GFX_GOL_STATE_BUSY	Module state busy
GFX_GOL_STATE_INIT	Module state init
GFX_GOL_STATE_INITIALIZED	Module state initialized

Description

GFX GOL Machine States

This enumeration defines the various states that can be achieved by the GOL operation.

Remarks

None.

GFX_AT_SCAN_CODES Enumeration

The following macros are the supported AT Keyboard scan codes.

File

gfx_gol_scan_codes.h

C

```
typedef enum {
  SCAN_CR_PRESSED = 0x1C,
  SCAN_CR_RELEASED = 0x9C,
  SCAN_CRA_PRESSED = 0x2C,
  SCAN_CRA_RELEASED = 0xAC,
  SCAN_DEL_PRESSED = 0x53,
  SCAN_DEL_RELEASED = 0xD3,
  SCAN_BS_PRESSED = 0x0E,
  SCAN_BS_RELEASED = 0x8E,
  SCAN_TAB_PRESSED = 0x0F,
  SCAN_TAB_RELEASED = 0x8F,
  SCAN_HOME_PRESSED = 0x47,
  SCAN_HOME_RELEASED = 0xC7,
  SCAN_END_PRESSED = 0x4F,
  SCAN_END_RELEASED = 0xCF,
  SCAN_PGUP_PRESSED = 0x49,
  SCAN_PGUP_RELEASED = 0xC9,
  SCAN_PGDOWN_PRESSED = 0x51,
  SCAN_PGDOWN_RELEASED = 0xD1,
  SCAN_UP_PRESSED = 0x48,
  SCAN_UP_RELEASED = 0xC8,
  SCAN_DOWN_PRESSED = 0x50,
  SCAN_DOWN_RELEASED = 0xD0,
  SCAN_LEFT_PRESSED = 0x4B,
  SCAN_LEFT_RELEASED = 0xCB,
  SCAN_RIGHT_PRESSED = 0x4D,
  SCAN_RIGHT_RELEASED = 0xCD,
  SCAN_SPACE_PRESSED = 0x39,
  SCAN_SPACE_RELEASED = 0xB9
} GFX_AT_SCAN_CODES;
```

Members

Members	Description
SCAN_CR_PRESSED = 0x1C	Carriage return pressed.
SCAN_CR_RELEASED = 0x9C	Carriage return released.
SCAN_CRA_PRESSED = 0x2C	Carriage return alternate pressed.
SCAN_CRA_RELEASED = 0xAC	Carriage return alternate released.

SCAN_DEL_PRESSED = 0x53	Delete key pressed.
SCAN_DEL_RELEASED = 0xD3	Delete key released.
SCAN_BS_PRESSED = 0x0E	Back space key pressed.
SCAN_BS_RELEASED = 0x8E	Back space key released.
SCAN_TAB_PRESSED = 0x0F	Tab key pressed.
SCAN_TAB_RELEASED = 0x8F	Tab key released.
SCAN_HOME_PRESSED = 0x47	Home key pressed.
SCAN_HOME_RELEASED = 0xC7	Home key released.
SCAN_END_PRESSED = 0x4F	End key pressed.
SCAN_END_RELEASED = 0xCF	End key released.
SCAN_PGUP_PRESSED = 0x49	Page up key pressed.
SCAN_PGUP_RELEASED = 0xC9	Page up key released.
SCAN_PGDOWN_PRESSED = 0x51	Page down key pressed.
SCAN_PGDOWN_RELEASED = 0xD1	Page down key released.
SCAN_UP_PRESSED = 0x48	Up key pressed.
SCAN_UP_RELEASED = 0xC8	Up key released.
SCAN_DOWN_PRESSED = 0x50	Down key pressed.
SCAN_DOWN_RELEASED = 0xD0	Down key released.
SCAN_LEFT_PRESSED = 0x4B	Left key pressed.
SCAN_LEFT_RELEASED = 0xCB	Left key released.
SCAN_RIGHT_PRESSED = 0x4D	Right key pressed.
SCAN_RIGHT_RELEASED = 0xCD	Right key released.
SCAN_SPACE_PRESSED = 0x39	Space key pressed.
SCAN_SPACE_RELEASED = 0xB9	Space key released.

Description

GFX AT keyboard scan codes.

Typedef: GFX_AT_SCAN_CODES

The following macros are the supported AT Keyboard scan codes.

Remarks

None.

GFX_GOL_SURFACE Structure

Defines the structure used for the Surface object.

File

gfx_gol_surface.h

C

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_GOL_SURFACE_CALLBACK_FUNC callback;
  GFX_RESOURCE_HDR * pImage;
} GFX_GOL_SURFACE;
```

Members

Members	Description
GFX_GOL_OBJ_HEADER hdr;	Generic header for all Objects (see GFX_GOL_OBJ_HEADER).
uint16_t textHeight;	Precomputed text height
GFX_XCHAR * pText;	Pointer to the title text
GFX_GOL_SURFACE_CALLBACK_FUNC callback;	text alignment
GFX_RESOURCE_HDR * pImage;	Pointer to the image for the title bar

Description

Typedef: GFX_GOL_SURFACE

Defines the parameters required for a Surface object. Object is drawn with the defined shape parameters and values set on the given fields.

Remarks

None.

GFX_GOL_SURFACE_CALLBACK_FUNC Type

Surface callback function definition. This application defined function allows the application to perform application specific rendering.

File

gfx_gol_surface.h

C

```
typedef GFX_STATUS (* GFX_GOL_SURFACE_CALLBACK_FUNC)(void);
```

Returns

true - is returned when application rendering is done. false - is returned when application rendering is not yet finished.

Description

This callback function is implemented by the application. This is called inside the GFX_GOL_ObjectListDraw() function when the drawing of objects in the active list is completed.

Any application specific rendering must be performed on this callback function so no object rendering will be affected by the application calls to primitive rendering functions. Application setting the drawing color, line style, fill style, text string cursor position and current font will not affect the object rendering. This is also the safe place to modify the active list.

When the application has performed its own primitive rendering calls, this function must return true to inform the GFX_GOL_ObjectListDraw() that it is done rendering and checking for object drawing or redrawing can continue.

Preconditions

None.

Example

None.

Function

typedef GFX_STATUS (*GFX_GOL_SURFACE_CALLBACK_FUNC) (void);

GFX_GOL_SURFACE_STATE Enumeration

Specifies the different states of the Surface object.

File

gfx_gol_surface.h

C

```
typedef enum {
  GFX_GOL_SURFACE_FOCUSED_STATE,
  GFX_GOL_SURFACE_DISABLED_STATE,
  GFX_GOL_SURFACE_DRAW_APP_STATE,
  GFX_GOL_SURFACE_DRAW_CLIENT_STATE,
  GFX_GOL_SURFACE_DRAW_STATE,
  GFX_GOL_SURFACE_HIDE_STATE
} GFX_GOL_SURFACE_STATE;
```

Members

Members	Description
GFX_GOL_SURFACE_FOCUSED_STATE	Property bit for focus state.
GFX_GOL_SURFACE_DISABLED_STATE	Property bit for disabled state.
GFX_GOL_SURFACE_DRAW_APP_STATE	Draw bit to indicate title area must be redrawn.
GFX_GOL_SURFACE_DRAW_CLIENT_STATE	Draw bit to indicate client area must be redrawn.

GFX_GOL_SURFACE_DRAW_STATE	Draw bit to indicate object must be redrawn.
GFX_GOL_SURFACE_HIDE_STATE	Draw bit to indicate object must be removed from screen.

Description

Typedef: GFX_GOL_SURFACE_STATE

This enumeration specifies the different states of the Surface object used in the library.

For the Property State bits, more than one of these values may be OR'd together to create a complete property state.

For the Draw State bits, hide draw bit has higher priority than the draw bit. Any of these two can be combined with other draw bits to create a complete draw state.

To test a value of any of the state types, the bit of interest must be AND'ed with value and checked to see if the result is non-zero.

Remarks

None.

GFX_GOL_OBJ_SCHEME Structure

This structure specifies the style scheme components of an object.

File

gfx_gol_scheme.h

C

```
typedef struct {
  GFX_COLOR EmbossDkColor;
  GFX_COLOR EmbossLtColor;
  GFX_COLOR TextColor0;
  GFX_COLOR TextColor1;
  GFX_COLOR TextColorDisabled;
  GFX_COLOR Color0;
  GFX_COLOR Color1;
  GFX_COLOR ColorDisabled;
  GFX_RESOURCE_HDR * pFont;
  GFX_FILL_STYLE fillStyle;
  GFX_COLOR CommonBkColor;
  uint16_t CommonBkLeft;
  uint16_t CommonBkTop;
  GFX_BACKGROUND_TYPE CommonBkType;
  GFX_RESOURCE_HDR * pCommonBkImage;
  uint16_t AlphaValue;
  GFX_COLOR gradientStartColor;
  GFX_COLOR gradientEndColor;
  uint16_t EmbossSize;
} GFX_GOL_OBJ_SCHEME;
```

Members

Members	Description
GFX_COLOR EmbossDkColor;	Emboss dark color used for 3d effect.
GFX_COLOR EmbossLtColor;	Emboss light color used for 3d effect.
GFX_COLOR TextColor0;	Character color 0 used for objects that supports text.
GFX_COLOR TextColor1;	Character color 1 used for objects that supports text.
GFX_COLOR TextColorDisabled;	Character color used when object is in a disabled state.
GFX_COLOR Color0;	Color 0 usually assigned to an Object state.
GFX_COLOR Color1;	Color 1 usually assigned to an Object state.
GFX_COLOR ColorDisabled;	Color used when an Object is in a disabled state.
GFX_RESOURCE_HDR * pFont;	Font selected for the scheme.
GFX_FILL_STYLE fillStyle;	must be set to a gradient type when using gradient
GFX_COLOR CommonBkColor;	Background color used to hide Objects.
uint16_t CommonBkLeft;	Horizontal starting position of the background.
uint16_t CommonBkTop;	Vertical starting position of the background.
GFX_BACKGROUND_TYPE CommonBkType;	Specifies the type of background to use.
GFX_RESOURCE_HDR * pCommonBkImage;	Pointer to the background image used. Set this
uint16_t AlphaValue;	Alpha value used for alpha blending

GFX_COLOR gradientStartColor;	start color of the gradient fill
GFX_COLOR gradientEndColor;	end color of the gradient fill
uint16_t EmbossSize;	Emboss size of the panel for 3-D effect. Set to zero if not used.

Description

Typedef: GFX_GOL_OBJ_SCHEME

This structure specifies the style scheme components of an object. All objects will use the style scheme when rendering. Refer to specific object documentation on how the style scheme colors are utilized by the object.

The style scheme allows objects to show 3D effects as well as feedback to users. For example, in Button objects, a press and release effect on the Buttons are easily shown by manipulating the colors when the object is drawn with a pressed state and released state. The style scheme also allows effects such as gradients and alpha blending. When using alpha blending, the style scheme requires objects to be associated with background information.

A background can be a flat color background or an image. The usage of a background requires the background dimension to be larger than the object. In other words, the object should be drawn within the background dimension. Multiple objects can share a common background. As long as all the objects are drawn within the dimension of the background they can share a common background.

The supported background types are (See GFX_BACKGROUND_TYPE):

- GFX_BACKGROUND_COLOR - this type will set the common background to be a flat color. The color used is specified by CommonBkType.
- GFX_BACKGROUND_IMAGE - this type will set the common background to be an image. The image used is specified by pCommonBkImage.

When an object is associated with a background, it can be easily hidden or redrawn with some effects (for example alpha blending with the background). The background information allows the redrawing of the background with the object without the need to manually refreshing the background using primitive calls by the application.

Remarks

None.

GFX_GOL_SurfaceImageGet Macro

This function gets the image used.

File

gfx_gol_surface.h

C

```
#define GFX_GOL_SurfaceImageGet(pObject, pImage) \
                                (((GFX_GOL_SURFACE *)pObject)->pImage)
```

Returns

Pointer to the image resource.

Description

This function gets the image used.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.

Function

GFX_RESOURCE_HDR *GFX_GOL_SurfaceImageGet(
GFX_GOL_SURFACE *pObject)

GFX_GOL_SurfaceImageSet Macro

This function sets the image used in the object.

File

gfx_gol_surface.h

C

```
#define GFX_GOL_SurfaceImageSet(pObject, image) \
                                (((GFX_GOL_SURFACE *)pObject)->pImage = image)
```

Returns

None.

Description

This function sets the image used in the object.

Preconditions

Object must exist in memory.

Example

None.

Parameters

Parameters	Description
pObject	pointer to the object.
pImage	pointer to the image to be set.

Function

void GFX_GOL_SurfaceImageSet(
GFX_GOL_SURFACE *pObject,
GFX_RESOURCE_HDR *pImage)

System and Client Interfaces

Functions

	Name	Description
	GFX_Open	Opens the specified GFX library instance and returns a handle to it.
	GFX_Close	Closes a client instance of the GFX library.
	GFX_Initialize	Initialize the GFX Library.
	GFX_Deinitialize	Deinitializes the specified instance of the GFX Module.
	GFX_Status	Gets the current status of the GFX Module.
	GFX_Tasks	Maintains the GFX module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events.
	GFX_GOL_Deinitialize	Deinitializes the specified instance of the GFX Module.
	GFX_GOL_Initialize	Initialize the GFX GOL Library.
	GFX_GOL_Open	opens an instance of the GFX GOL Library.
	GFX_GOL_Tasks	Maintains the GFX GOL module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events.
	GFX_GOL_PreemptionLevelGet	This function returns the current preemption level.
	ImageDecoderInit	This function initializes the global variables to 0 and then initializes the driver. This must be called once before any other function of the library is called
	ImageDecodeTask	This function completes one small part of the image decode function
	ImageDecode	This function decodes and displays the image on the screen
	ImageLoopCallbackRegister	This function registers the loop callback function so that the decoder calls this function in every decoding loop. This can be used by the application program to do maintenance activities such as fetching data, updating the display, etc...
	IMG_vSetboundaries	
	ImageDecoderParceHeader	This is function ImageDecoderParceHeader.
	ImagePixelIntoArray	This is function ImagePixelIntoArray.
	ImagePixelOutput	This is function ImagePixelOutput.
	ImageRegisterFrameCompleteCallback	This is function ImageRegisterFrameCompleteCallback.

	ImageRegisterImageCompleteCallback	Function prototypes
	ImageRegisterImageFailCallback	This is function ImageRegisterImageFailCallback.

Data Types and Constants

	Name	Description
	GFX_FONT_GLYPH_ENTRY	The structure describing the glyph entry in fonts.
	GFX_FONT_GLYPH_ENTRY_EXTENDED	The structure describing the intended glyph entry in fonts.
	GFX_FONT_HEADER	The structure used to define the font header.
	GFX_MCHP_BITMAP_HEADER	The structure used to define the Microchip bitmap header.
	GFX_PARTIAL_IMAGE_PARAM	Partial Image information structure.
	GFX_RECTANGULAR_AREA	A generic rectangular area structure.
	GFX_RESOURCE	Specifies the different resource types in the library.
	GFX_RESOURCE_BINARY	Defines the structure used for the binary type resource.
	GFX_RESOURCE_FONT	Defines the structure used for the font type resource.
	GFX_RESOURCE_HDR	Defines the structure used for the resource types.
	GFX_RESOURCE_IMAGE	Defines the structure used for the image type resource.
	GFX_RESOURCE_PALETTE	Defines the structure used for the palette type resource.
	GFX_STATUS	Rendering status.
	GFX_STATUS_BIT	Additional rendering status.
	int16_gfx_image_prog	This is type int16_gfx_image_prog.
	int32_gfx_image_prog	This is type int32_gfx_image_prog.
	int8_gfx_image_prog	This is type int8_gfx_image_prog.
	uint16_gfx_image_prog	This is type uint16_gfx_image_prog.
	uint32_gfx_image_prog	This is type uint32_gfx_image_prog.
	uint8_gfx_image_prog	GFX Image data Types
	GFX_ALIGNMENT	Summary of the different text alignment supported in the library.
	GFX_BACKGROUND	Background information structure.
	GFX_BACKGROUND_TYPE	Specifies the different background fill types.
	GFX_DOUBLE_BUFFERING_MODE	Structure used for double buffering management.
	GFX_FEATURE_STATUS	States of a feature that can be enabled/disabled at run time.
	GFX_FILL_STYLE	Specifies the available fill styles.
	GFX_FONT_ANTIALIAS_TYPE	Summary of the transparency types in text anti-aliasing.
	GFX_FONT_SPACE	Font space section. The fonts can be located in psv (constant) or program space in PIC24/dsPIC MCUs. This define allows for switching of the pointer type used to access the font structure in memory See: GraphicsConfig.h for the application define.
	GFX_UXCHAR	This is macro GFX_UXCHAR.
	GFX_XCHAR	Configuration for the text character size.
	GRAPHICS_LIBRARY_VERSION	////////////////// GRAPHICS_LIBRARY_VERSION ////////////////////// MSB is version, LSB is subversion
	GFX_COMP_MASK	This is macro GFX_COMP_MASK.
	GFX_MEM_MASK	This is macro GFX_MEM_MASK.
	GFX_TYPE_MASK	This is macro GFX_TYPE_MASK.
	MCHP_BITMAP_NORMAL	no compression, palette is present for color depth = 8, 4 and 1 BPP
	MCHP_BITMAP_PALETTE_STR	palette is provided as a separate object (see PALETTE_HEADER) for color depth = 8, 4, and 1 BPP, ID to the palette is embedded in the bitmap.
	int16_prog	This is type int16_prog.
	int16_prog_pack	This is type int16_prog_pack.
	int32_prog	This is type int32_prog.
	int32_prog_pack	This is type int32_prog_pack.
	int8_prog	This is type int8_prog.
	int8_prog_pack	This is type int8_prog_pack.
	uint16_prog	This is type uint16_prog.
	uint16_prog_pack	This is type uint16_prog_pack.
	uint32_prog	This is type uint32_prog.
	uint32_prog_pack	This is type uint32_prog_pack.
	uint8_prog	General program space data types

	uint8_prog_pack	This is type uint8_prog_pack.
	GFX_CLIENT_OBJ	GFX client object structure.
	GFX_CLIENT_STATUS	Enumerated data type that identifies the GFX Module Client Status.
	GFX_OBJ	Defines the object required for interaction with the GFX Library.
	GFX_OBJECT_TASK	Lists the different states that GFX task routine can have.
	GFX_STATES	Defines the various states that can be achieved by the GFX operation.
	_IMG_FILE_FORMAT	IMG_ImageFileFormat specifies all the supported image file formats
	_IMG_FILE_SYSTEM_API	IMG_FileSystemAPI holds function pointers to the used File System APIs
	_IMG_PIXEL_XY_RGB_888	IMG_PixelRgb holds the RGB information of a pixel in BYTES
	FileFeof	This is type FileFeof.
	FileRead	Typedefs of the used File System APIs
	FileSeek	This is type FileSeek.
	FileTell	This is type FileTell.
	IMG_FILE_FORMAT	IMG_ImageFileFormat specifies all the supported image file formats
	IMG_FILE_SYSTEM_API	IMG_FileSystemAPI holds function pointers to the used File System APIs
	IMG_LOOP_CALLBACK	IMG_LoopCallback is a callback function which is called in every loop of the decoding cycle so that user application can do some maintenance activities
	IMG_PIXEL_OUTPUT	IMG_PixelOutput is a callback function which receives the color information of the output pixel
	IMG_PIXEL_XY_RGB_888	IMG_PixelRgb holds the RGB information of a pixel in BYTES
	__IMAGEDECODER_H__	This is macro __IMAGEDECODER_H__.
	_PutPixel	This is macro _PutPixel.
	ImageAbort	This function sets the Image Decoder's Abort flag so that decoding aborts in the next decoding loop.
	ImageFullScreenDecode	This function decodes and displays the image on the screen in fullscreen mode with center aligned and downscaled if required
	IMG_ALIGN_CENTER	Flags
	IMG_DECODE_ABORTED	This is macro IMG_DECODE_ABORTED.
	IMG_DOWN_SCALE	This is macro IMG_DOWN_SCALE.
	IMG_FEOF	This is macro IMG_FEOF.
	IMG_FILE	This is macro IMG_FILE.
	IMG_FREAD	This is macro IMG_FREAD.
	IMG_FSEEK	This is macro IMG_FSEEK.
	IMG_FTELL	This is macro IMG_FTELL.
	IMG_SCREEN_HEIGHT	This is macro IMG_SCREEN_HEIGHT.
	IMG_SCREEN_WIDTH	The individual image decoders use these defines instead of directly using those provided in the Display driver file
	IMG_vCheckAndAbort	This is macro IMG_vCheckAndAbort.
	IMG_vLoopCallback	This is macro IMG_vLoopCallback.
	IMG_vPutPixel	This is macro IMG_vPutPixel.
	GFX_PALETTE_ENTRY	Defines the union used for each entry in the palette table.
	GFX_TEXT_CURSOR_TYPE	Enumeration defines types of text cursor supported.
	IMG_LOOP_FRAMECOMPLETE	IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities
	IMG_LOOP_IMAGECOMPLETE	IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities
	IMG_LOOP_IMAGEFAIL	IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities
	IMG_vFrameComplete	This is macro IMG_vFrameComplete.
	IMG_vImageComplete	This is macro IMG_vImageComplete.
	IMG_vImageFail	This is macro IMG_vImageFail.
	IMG_vPixelArray	This is macro IMG_vPixelArray.
	IMG_vSetColor	This is macro IMG_vSetColor.

Description

This section describes the Graphics Library System and Client API.

Functions

GFX_Open Function

Opens the specified GFX library instance and returns a handle to it.

File

gfx.h

C

```
GFX_HANDLE GFX_Open(SYS_MODULE_OBJ gfxObject);
```

Returns

If successful, the function returns a valid open instance handle (a number identifying both the caller and the module instance).

If an error occurs, the return value is GFX_HANDLE_INVALID.

Description

This function opens the specified GFX library instance and provides a handle that must be provided to all other client-level operations to identify the caller and the instance of the driver.

Remarks

The handle returned is valid until the GFX_Close function is called.

Preconditions

The GFX_Initialize function must have been called before calling this function.

Example

```
GFX_HANDLE  handle;

handle = GFX_Open ( gfxObject );

if ( GFX_HANDLE_INVALID == handle )
{
    // Unable to open the library
}
```

Parameters

Parameters	Description
object	Identifier for the object instance to be opened

Function

GFX_HANDLE GFX_Open(SYS_MODULE_OBJ gfxObject)

GFX_Close Function

Closes a client instance of the GFX library.

File

gfx.h

C

```
void GFX_Close(GFX_HANDLE gfxHandle);
```

Returns

None

Description

This routine closes a client instance of the GFX library.

Preconditions

The GFX_Open routine must have been called for the specified handle.

Example

```
GFX_HANDLE handle;  // Returned from GFX_Open

GFX_Close(handle);
```

Parameters

Parameters	Description
handle	A valid handle, returned from the open routine

Function

void GFX_Close(GFX_HANDLE handle)

GFX_Initialize Function

Initialize the GFX Library.

File

gfx.h

C

```
SYS_MODULE_OBJ GFX_Initialize(const SYS_MODULE_INDEX moduleIndex, const SYS_MODULE_INIT * const moduleInit);
```

Returns

If successful, returns a valid handle to a device layer object. Otherwise, it returns SYS_MODULE_OBJ_INVALID.

Description

This function initialize the GFX Library primitive layer and Object Layer if enabled. The following default settings are set when this function is called.

1. font - Set to NULL. GFX_FontSet() must be called prior to any text rendering.
2. line type - Set to GFX_LINE_TYPE_THIN_SOLID (see GFX_LINE_STYLE).
3. fill type - Set to GFX_FILL_TYPE_COLOR (see GFX_FILL_STYLE).
4. text anti-alias type - Set to GFX_FONT_ANTIALIAS_OPAQUE (see GFX_FONT_ANTIALIAS_TYPE). This only affects fonts with anti-aliasing enabled.
5. Set transparent color feature in image draw functions to be disabled.
6. Set alpha blending value to 100 (or no alpha blending) if alpha blending feature is enabled.
7. Set background information to no background.

This function does not clear the screen and does not assign any color to the currently set color. Application should set the color and clear the screen.

Remarks

This routine must be called before other GFX library functions.

Preconditions

None.

Example

```
GFX_INIT gfxInit;
SYS_MODULE_OBJ gfxObj;

// GFX Module initialization
gfxObj = GFX_Initialize(GFX_INDEX_0, &gfxInit);

if (SYS_MODULE_OBJ_INVALID == gfxObj)
{
    // Handle error
}
```

Parameters

Parameters	Description
moduleIndex	client instance request.
moduleInit	initialization data for the instance.

Function

void GFX_Initialize (const SYS_MODULE_INDEX moduleIndex,
const SYS_MODULE_INIT * const moduleInit);

GFX_Deinitialize Function

Deinitializes the specified instance of the GFX Module.

File

gfx.h

C

```
void GFX_Deinitialize(SYS_MODULE_OBJ moduleObject);
```

Returns

None.

Description

Deinitializes the specified instance of the GFX Module. This function will also deinitialize the GOL, PRIMITIVE, and driver and conclude all GFX related operations. All internal data structures will be reset.

Remarks

Once the Initialize operation has been called, the Deinitialize operation must be called before the Initialize operation can be called again. This routine will NEVER block waiting for hardware.

Preconditions

Function GFX_Deinitialize should have been called before calling this function.

Example

```
SYS_MODULE_OBJ      object;     //  Returned from GFX_Initialize
SYS_STATUS          status;


GFX_Deinitialize(object);

status = GFX_Status(object);
if (SYS_MODULE_DEINITIALIZED != status)
{
    // Check again later if you need to know
    // when the driver is deinitialized.
}
```

Parameters

Parameters	Description
object	GFX layer object handle, returned from the GFX_Initialize routine

Function

void GFX_Deinitialize(SYS_MODULE_OBJ object)

GFX_Status Function

Gets the current status of the GFX Module.

File

gfx.h

C

```
SYS_STATUS GFX_Status(SYS_MODULE_OBJ gfxObject);
```

Returns

SYS_STATUS_READY - Indicates that the GFX layer is ready for operations.

SYS_STATUS_DEINITIALIZED - Indicates that the driver has been deinitialized.

Description

This routine provides the current status of the GFX Module.

Remarks

None.

Preconditions

Function GFX_Initialize should have been called before calling this function.

Example

```
SYS_MODULE_INDEX    index;     // Returned from GFX_Initialize
SYS_STATUS          status;

status = GFX_Status (object);
if (SYS_STATUS_READY == status)
{
    // This means the library can be used, using for example
    // GFX_DrawLine() function.
}
```

Parameters

Parameters	Description
object	GFX object index, returned from the GFX_Initialize routine

Function

SYS_STATUS GFX_Status (SYS_MODULE_OBJ gfxObject)

GFX_Tasks Function

Maintains the GFX module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events.

File

gfx.h

C

```
void GFX_Tasks(SYS_MODULE_OBJ gfxObject);
```

Returns

None.

Description

Maintains the GFX module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events. This function should be called from the SYS_Tasks() function.

Remarks

This routine is normally not called directly by an application.

Preconditions

None.

Example

```
while (true)
{
    GFX_Tasks (GFX_INDEX_0);
```

```
    // Do other tasks
}
```

Parameters

Parameters	Description
index	Object index for the specified module instance.

Function

void GFX_Tasks (SYS_MODULE_INDEX index);

GFX_GOL_Deinitialize Function

Deinitializes the specified instance of the GFX Module.

File

gfx_gol.h

C

```
void GFX_GOL_Deinitialize(SYS_MODULE_OBJ moduleObject);
```

Returns

None.

Description

Deinitializes the specified instance of the GFX Module. All internal data structures will be reset.

Remarks

Once the Initialize operation has been called, the Deinitialize operation must be called before the Initialize operation can be called again. This routine will NEVER block waiting for hardware.

Preconditions

Function GFX_GOL_Initialize should have been called before calling this function.

Example

```
SYS_MODULE_OBJ      object;     //  Returned from GFX_GOL_Initialize
SYS_STATUS          status;


GFX_GOL_Deinitialize(object);

status = GFX_Status(object);
if (SYS_MODULE_DEINITIALIZED != status)
{
    // Check again later if you need to know
    // when the driver is deinitialized.
}
```

Parameters

Parameters	Description
object	GFX layer object handle, returned from the GFX_GOL_Initialize routine

Function

void GFX_GOL_Deinitialize(SYS_MODULE_OBJ object)

GFX_GOL_Initialize Function

Initialize the GFX GOL Library.

File

gfx_gol.h

C

```
SYS_MODULE_OBJ GFX_GOL_Initialize(const SYS_MODULE_INDEX moduleIndex, const SYS_MODULE_INIT * const
```

```
moduleInit);
```

Returns

If successful, returns a valid handle to a device layer object. Otherwise, it returns SYS_MODULE_OBJ_INVALID.

Description

This function initialize the GFX Library GOL layer. The following default settings are set when this function is called.

1. font -

Remarks

This routine must be called before other GFX library functions.

Preconditions

None.

Example

```
GFX_INIT gfxInit;
SYS_MODULE_OBJ gfxObj;

// GFX Module initialization
gfxObj = GFX_GOL_Initialize(GFX_INDEX_0, &gfxInit);

if (SYS_MODULE_OBJ_INVALID == gfxObj)
{
    // Handle error
}
```

Parameters

Parameters	Description
moduleIndex	client instance request.
moduleInit	initialization data for the instance.

Function

void GFX_GOL_Initialize (const SYS_MODULE_INDEX moduleIndex,

const SYS_MODULE_INIT * const moduleInit);

GFX_GOL_Open Function

opens an instance of the GFX GOL Library.

File

gfx_gol.h

C

```
GFX_HANDLE GFX_GOL_Open(const SYS_MODULE_INDEX index);
```

Function

GFX_HANDLE GFX_GOL_Open(const SYS_MODULE_INDEX index)

GFX_GOL_Tasks Function

Maintains the GFX GOL module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events.

File

gfx_gol.h

C

```
void GFX_GOL_Tasks(SYS_MODULE_INDEX index);
```

Returns

None.

Description

Maintains the GFX module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events. This function should be called from the SYS_Tasks() function.

Remarks

This routine is normally not called directly by an application.

Preconditions

None.

Example

```
while (true)
{
    GFX_GOL_Tasks (GFX_INDEX_0);

    // Do other tasks
}
```

Parameters

Parameters	Description
index	Object index for the specified module instance.

Function

void GFX_GOL_Tasks (SYS_MODULE_INDEX index);

GFX_GOL_PreemptionLevelGet Function

This function returns the current preemption level.

File

gfx_gol.h

C

```
GFX_PREEMPTION_LEVEL GFX_GOL_PreemptionLevelGet(SYS_MODULE_INDEX gfxIndex);
```

Returns

The current preemption level.

Description

GFX GOL Preemption level get

This function returns the current preemption level.

Preconditions

None.

Parameters

Parameters	Description
gfxIndex	Object index for the specified module instance.

Function

GFX_PREEMPTION_LEVEL GFX_GOL_PreemptionLevelGet (

SYS_MODULE_INDEX gfxIndex)

ImageDecoderInit Function

File

gfx_image_decoder.h

C

```
void ImageDecoderInit();
```

Side Effects

The graphics driver will be reset

Returns

None

Description

This function initializes the global variables to 0 and then initializes the driver. This must be called once before any other function of the library is called

Example

```
  void main(void)
{
    ImageInit();
    ...
}
```

Function

void ImageDecoderInit(void)

ImageDecodeTask Function

File

gfx_image_decoder.h

C

```
uint8_t ImageDecodeTask();
```

Side Effects

None

Returns

Status code - '1' means decoding is completed

- '0' means decoding is not yet completed, call this function again

Description

This function completes one small part of the image decode function

Example

```
      IMG_bFullScreenDecode(pImageFile, IMG_JPEG, NULL, NULL);
    while(!ImageDecodeTask());
```

Function

uint8_t ImageDecodeTask(void)

ImageDecode Function

File

gfx_image_decoder.h

C

```
uint8_t ImageDecode(IMG_FILE * pImageFile, IMG_FILE_FORMAT eImgFormat, uint16_t wStartx, uint16_t wStarty,
uint16_t wWidth, uint16_t wHeight, uint16_t wFlags, IMG_FILE_SYSTEM_API * pFileAPIs, IMG_PIXEL_OUTPUT
pPixelOutput);
```

Side Effects

None

Returns

Error code -> 0 means no error

Description

This function decodes and displays the image on the screen

Example

```
void main(void)
{
   IMG_FILE pImageFile;
   IMG_vInitialize();
   pImageFile = IMG_FOPEN("Image.jpg", "r");
   if(pImageFile == NULL)
   {
         <- Error handling ->
   }
   IMG_bDecode(pImageFile, IMG_JPEG, 0, 0, 320, 240, 0, NULL, NULL);
   IMG_FCLOSE(pImageFile);
   while(1);
}
```

Function

uint8_t ImageDecode(IMG_FILE *pImageFile, IMG_FILE_FORMAT eImgFormat, uint16_t wStartx, uint16_t wStarty, uint16_t wWidth, uint16_t wHeight, uint16_t wFlags, IMG_FILE_SYSTEM_API *pFileAPIs, IMG_PIXEL_OUTPUT pPixelOutput)

ImageLoopCallbackRegister Function

File

gfx_image_decoder.h

C

```
void ImageLoopCallbackRegister(IMG_LOOP_CALLBACK pFn);
```

Side Effects

The graphics driver will be reset

Returns

None

Description

This function registers the loop callback function so that the decoder calls this function in every decoding loop. This can be used by the application program to do maintenance activities such as fetching data, updating the display, etc...

Example

```
  void Mainantance(void)
{
    ...
}

void main(void)
{
    ImageInit();
    ImageLoopCallbackRegister(Mainantance);
    ...
}
```

Function

void ImageLoopCallbackRegister(IMG_LOOP_CALLBACK pLoopCallbackFn)

IMG_vSetboundaries Function

File

gfx_image_decoder.h

C

```
void IMG_vSetboundaries();
```

Description

******* This is not for the user *******

This is used by the individual decoders

ImageDecoderParceHeader Function

File

gfx_image_decoder.h

C

```
int8_t ImageDecoderParceHeader(IMG_FILE * pImageFile, GFX_RESOURCE_HDR* header, GFX_RESOURCE source);
```

Description

This is function ImageDecoderParceHeader.

ImagePixelIntoArray Function

File

gfx_image_decoder.h

C

```
void ImagePixelIntoArray(IMG_PIXEL_XY_RGB_888 * pPix);
```

Description

This is function ImagePixelIntoArray.

ImagePixelOutput Function

File

gfx_image_decoder.h

C

```
void ImagePixelOutput(IMG_PIXEL_XY_RGB_888 * pPix);
```

Description

This is function ImagePixelOutput.

ImageRegisterFrameCompleteCallback Function

File

gfx_image_decoder.h

C

```
void ImageRegisterFrameCompleteCallback(void* callbackFn);
```

Description

This is function ImageRegisterFrameCompleteCallback.

ImageRegisterImageCompleteCallback Function

File

gfx_image_decoder.h

C

```
void ImageRegisterImageCompleteCallback(void* callbackFn);
```

Description

Function prototypes

ImageRegisterImageFailCallback Function

File

gfx_image_decoder.h

C

```
void ImageRegisterImageFailCallback(void* callbackFn);
```

Description

This is function ImageRegisterImageFailCallback.

Data Types and Constants

GFX_FONT_GLYPH_ENTRY Structure

The structure describing the glyph entry in fonts.

File

gfx_types_font.h

C

```
typedef struct {
  uint8_t width;
  uint8_t offsetLSB;
  uint16_t offsetMSB;
} GFX_FONT_GLYPH_ENTRY;
```

Members

Members	Description
uint8_t width;	The width of the glyph.
uint8_t offsetLSB;	The least Significant byte of the glyph location offset.
uint16_t offsetMSB;	The most Significant (2) bytes of the glyph location offset.

Description

Typedef: GFX_FONT_GLYPH_ENTRY

Each character's bitmap (or glyph) in a font is described by its glyph entry structure. The entry will define the width and the offset of the character glyph from the header of the font table.

Remarks

None.

GFX_FONT_GLYPH_ENTRY_EXTENDED Structure

The structure describing the intended glyph entry in fonts.

File

gfx_types_font.h

C

```
typedef struct {
  uint32_t offset;
  uint16_t cursorAdvance;
  uint16_t glyphWidth;
  int16_t xAdjust;
  int16_t yAdjust;
} GFX_FONT_GLYPH_ENTRY_EXTENDED;
```

Members

Members	Description
uint32_t offset;	The offset of the glyph. The offset order is: LSW_LSB LSW_MSB MSW_MSB MSW_MSB.
uint16_t cursorAdvance;	The x-value by which cursor has to advance after rendering the glyph.
uint16_t glyphWidth;	The width of the glyph.
int16_t xAdjust;	The value that adjusts the x-position.
int16_t yAdjust;	The value that adjusts the y-position.

Description

Typedef: GFX_FONT_GLYPH_ENTRY_EXTENDED

Similar to the GFX_FONT_GLYPH_ENTRY, each character's glyph in a font that supports intended characters is also described by its glyph entry structure. The difference is that the extended glyph entry will contain additional information on how the characters are overlapped.

Remarks

None.

GFX_FONT_HEADER Structure

The structure used to define the font header.

File

gfx_types_font.h

C

```
typedef struct {
  uint8_t fontID;
  uint8_t extendedGlyphEntry : 1;
  uint8_t res1 : 1;
  uint8_t bpp : 2;
  uint8_t orientation : 2;
  uint8_t res2 : 2;
  uint16_t firstChar;
  uint16_t lastChar;
  uint16_t height;
} GFX_FONT_HEADER;
```

Members

Members	Description
uint8_t fontID;	User assigned value
uint8_t extendedGlyphEntry : 1;	Extended Glyph entry flag. When set font has extended glyph feature enabled.
uint8_t res1 : 1;	Reserved for future use (must be set to 0)
uint8_t bpp : 2;	Actual BPP = 2^{bpp} • 0 - 1 BPP • 1 - 2 BPP • 2 - 4 BPP • 3 - 8 BPP
uint8_t orientation : 2;	Orientation of the character glyphs (0,90,180,270 degrees) • 00 - Normal • 01 - Characters rotated 270 degrees clockwise • 10 - Characters rotated 180 degrees • 11 - Characters rotated 90 degrees clockwise
uint8_t res2 : 2;	Reserved for future use (must be set to 0).
uint16_t firstChar;	Character code of first character (e.g. 32).

uint16_t lastChar;	Character code of last character in font (e.g. 3006).
uint16_t height;	Font characters height in pixels.

Description

Typedef: GFX_FONT_HEADER

The structure used to define the font header.

Remarks

None.

GFX_MCHP_BITMAP_HEADER Structure

The structure used to define the Microchip bitmap header.

File

gfx_types_image.h

C

```
typedef struct {
  uint8_t bitmapType;
  uint8_t colorDepth;
  int16_t height;
  int16_t width;
} GFX_MCHP_BITMAP_HEADER;
```

Members

Members	Description
uint8_t bitmapType;	type of image, information on how to render the image 0 - no compression, palette is present for color depth = 8, 4 and 1 BPP 1 - palette is provided as a separate object (see PALETTE_HEADER) for color depth = 8, 4, and 1 BPP, ID to the palette is embedded in the bitmap.
uint8_t colorDepth;	Color depth used
int16_t height;	Image height
int16_t width;	Image width

Description

Typedef: GFX_MCHP_BITMAP_HEADER

The structure used to define the Microchip bitmap header.

Remarks

None.

GFX_PARTIAL_IMAGE_PARAM Structure

Partial Image information structure.

File

gfx_types_macros.h

C

```
typedef struct {
  uint16_t width;
  uint16_t height;
  uint16_t xoffset;
  uint16_t yoffset;
} GFX_PARTIAL_IMAGE_PARAM;
```

Members

Members	Description
uint16_t width;	Partial Image width.
uint16_t height;	Partial Image height.
uint16_t xoffset;	xoffset of the partial image (with respect to the image horizontal origin).

uint16_t yoffset;	yoffset of the partial image (with respect to the image vertical origin).

Description

Typedef: GFX_PARTIAL_IMAGE_PARAM

Structure describing the partial image area to be rendered.

Remarks

None.

GFX_RECTANGULAR_AREA Structure

A generic rectangular area structure.

File

gfx_types_macros.h

C

```
typedef struct {
  uint16_t left;
  uint16_t top;
  uint16_t right;
  uint16_t bottom;
} GFX_RECTANGULAR_AREA;
```

Members

Members	Description
uint16_t left;	left most pixel of the area.
uint16_t top;	top most pixel of the area.
uint16_t right;	right most pixel of the area.
uint16_t bottom;	bottom most pixel of the area.

Description

Typedef: GFX_RECTANGULAR_AREA

Structure describing a generic rectangular area defined by the left,top and right,bottom points on the frame buffer.

Remarks

None.

GFX_RESOURCE Enumeration

Specifies the different resource types in the library.

File

gfx_types_resource.h

C

```
typedef enum {
  GFX_RESOURCE_MEMORY_FLASH,
  GFX_RESOURCE_MEMORY_EXTERNAL,
  GFX_RESOURCE_MEMORY_FS,
  GFX_RESOURCE_MEMORY_RAM,
  GFX_RESOURCE_MEMORY_EDS_EPMP,
  GFX_RESOURCE_TYPE_MCHP_MBITMAP,
  GFX_RESOURCE_TYPE_JPEG,
  GFX_RESOURCE_TYPE_BINARY,
  GFX_RESOURCE_TYPE_FONT,
  GFX_RESOURCE_TYPE_PALETTE,
  GFX_RESOURCE_TYPE_BMP,
  GFX_RESOURCE_TYPE_GIF,
  GFX_RESOURCE_COMP_NONE,
  GFX_RESOURCE_COMP_RLE,
  GFX_RESOURCE_COMP_IPU,
  GFX_RESOURCE_MCHP_MBITMAP_FLASH_NONE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_MCHP_MBITMAP_FLASH_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_FLASH |
```

```
GFX_RESOURCE_COMP_RLE),
  GFX_RESOURCE_MCHP_MBITMAP_FLASH_IPU = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_IPU),
  GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL
| GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_RLE),
  GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_IPU = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_IPU),
  GFX_RESOURCE_MCHP_MBITMAP_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EDS_EPMP
| GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_MCHP_MBITMAP_EDS_EPMP_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_RLE),
  GFX_RESOURCE_MCHP_MBITMAP_EDS_EPMP_IPU = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_IPU),
  GFX_RESOURCE_JPEG_FLASH_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_JPEG_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_JPEG_FS_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_FS | GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_JPEG_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BMP_FLASH_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BMP_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BMP_FS_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_FS | GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BMP_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_GIF_FLASH_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_GIF_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_GIF_FS_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_FS | GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_GIF_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BINARY_FLASH_NONE = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BINARY_FLASH_IPU = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_IPU),
  GFX_RESOURCE_BINARY_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BINARY_EXTERNAL_IPU = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_IPU),
  GFX_RESOURCE_BINARY_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_BINARY_EDS_EPMP_IPU = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_IPU),
  GFX_RESOURCE_FONT_FLASH_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_FONT_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_FONT_RAM_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_RAM | GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_FONT_EDS_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_EDS_EPMP |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_PALETTE_FLASH_NONE = (GFX_RESOURCE_TYPE_PALETTE | GFX_RESOURCE_MEMORY_FLASH |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_PALETTE_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_PALETTE | GFX_RESOURCE_MEMORY_EXTERNAL |
GFX_RESOURCE_COMP_NONE),
  GFX_RESOURCE_PALETTE_RAM_NONE = (GFX_RESOURCE_TYPE_PALETTE | GFX_RESOURCE_MEMORY_RAM |
GFX_RESOURCE_COMP_NONE)
} GFX_RESOURCE;
```

Members

Members	Description
GFX_RESOURCE_MEMORY_FLASH	A location type: Internal flash memory.
GFX_RESOURCE_MEMORY_EXTERNAL	A location type: External memory.
GFX_RESOURCE_MEMORY_FS	A location type: External memory vis File System.
GFX_RESOURCE_MEMORY_RAM	A location type: Random Access Memory (RAM).
GFX_RESOURCE_MEMORY_EDS_EPMP	A location type: Resource is external and accessed through Enhanced Parallel Master Port (EPMP), memory size and base addresses are are set in the configuration.

GFX_RESOURCE_TYPE_MCHP_MBITMAP	A data type: Microchip bitmap type of resource.
GFX_RESOURCE_TYPE_JPEG	A data type: Image of type JPEG.
GFX_RESOURCE_TYPE_BINARY	A data type: Binary data type.
GFX_RESOURCE_TYPE_FONT	A data type: Font type of data
GFX_RESOURCE_TYPE_PALETTE	A data type: Palette type of data
GFX_RESOURCE_TYPE_BMP	A data type: Image of type BMP.
GFX_RESOURCE_COMP_NONE	A compression type: Data resource has no compression.
GFX_RESOURCE_COMP_RLE	A compression type: Data resource is compressed with RLE.
GFX_RESOURCE_COMP_IPU	A compression type: Data resource compressed with DEFLATE algorithm (for IPU).
GFX_RESOURCE_MCHP_MBITMAP_FLASH_NONE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	These are common resource combinations used by the graphics library Microchip bitmap image type, located in flash and no compression.
GFX_RESOURCE_MCHP_MBITMAP_FLASH_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_RLE)	Microchip bitmap image type, located in flash and RLE compressed.
GFX_RESOURCE_MCHP_MBITMAP_FLASH_IPU = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_IPU)	Microchip bitmap image type, located in flash and compressed for IPU
GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	Microchip bitmap image type, located in external memory and no compression.
GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_RLE)	Microchip bitmap image type, located in external memory and RLE compressed.
GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_IPU = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_IPU)	Microchip bitmap image type, located in external memory and compressed for IPU.
GFX_RESOURCE_MCHP_MBITMAP_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_NONE)	Microchip bitmap image type, located in EDS memory and no compression.
GFX_RESOURCE_MCHP_MBITMAP_EDS_EPMP_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_RLE)	Microchip bitmap image type, located in EDS memory and RLE compressed.
GFX_RESOURCE_MCHP_MBITMAP_EDS_EPMP_IPU = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_IPU)	Microchip bitmap image type, located in EDS memory and compressed for IPU.
GFX_RESOURCE_JPEG_FLASH_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	JPEG image type, located in flash and no compression.
GFX_RESOURCE_JPEG_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	JPEG image type, located in external memory no compression.
GFX_RESOURCE_JPEG_FS_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_FS | GFX_RESOURCE_COMP_NONE)	JPEG image type, located in file system no compression.
GFX_RESOURCE_JPEG_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_JPEG | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_NONE)	JPEG image type, located in EDS memory and no compression.
GFX_RESOURCE_BMP_FLASH_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	BMP image type, located in flash and no compression.

GFX_RESOURCE_BMP_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	BMP image type, located in external memory no compression.
GFX_RESOURCE_BMP_FS_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_FS | GFX_RESOURCE_COMP_NONE)	BMP image type, located in file system no compression.
GFX_RESOURCE_BMP_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_BMP | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_NONE)	BMP image type, located in EDS memory and no compression.
GFX_RESOURCE_GIF_FLASH_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	GIF image type, located in flash and no compression.
GFX_RESOURCE_GIF_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	GIF image type, located in external memory no compression.
GFX_RESOURCE_GIF_FS_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_FS | GFX_RESOURCE_COMP_NONE)	GIF image type, located in file system no compression.
GFX_RESOURCE_GIF_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_GIF | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_NONE)	GIF image type, located in EDS memory and no compression.
GFX_RESOURCE_BINARY_FLASH_NONE = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	Binary image type, located in flash and no compression.
GFX_RESOURCE_BINARY_FLASH_IPU = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_IPU)	Binary image type, located in flash memory and compressed for IPU.
GFX_RESOURCE_BINARY_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	Binary image type, located in external memory and no compression.
GFX_RESOURCE_BINARY_EXTERNAL_IPU = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_IPU)	Binary image type, located in external memory and compressed for IPU.
GFX_RESOURCE_BINARY_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_NONE)	Binary image type, located in EDS memory and no compression.
GFX_RESOURCE_BINARY_EDS_EPMP_IPU = (GFX_RESOURCE_TYPE_BINARY | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_IPU)	Binary image type, located in EDS memory and compressed for IPU.
GFX_RESOURCE_FONT_FLASH_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	Font type, located in flash and no compression.
GFX_RESOURCE_FONT_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	Font type, located in external memory and no compression.
GFX_RESOURCE_FONT_RAM_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_RAM | GFX_RESOURCE_COMP_NONE)	Font type, located in RAM and no compression.
GFX_RESOURCE_FONT_EDS_NONE = (GFX_RESOURCE_TYPE_FONT | GFX_RESOURCE_MEMORY_EDS_EPMP | GFX_RESOURCE_COMP_NONE)	Font type, located in EDS memory and no compression.

GFX_RESOURCE_PALETTE_FLASH_NONE = (GFX_RESOURCE_TYPE_PALETTE | GFX_RESOURCE_MEMORY_FLASH | GFX_RESOURCE_COMP_NONE)	Palette type, located in flash and no compression.
GFX_RESOURCE_PALETTE_EXTERNAL_NONE = (GFX_RESOURCE_TYPE_PALETTE | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_NONE)	Palette type, located in external memory and no compression.
GFX_RESOURCE_PALETTE_RAM_NONE = (GFX_RESOURCE_TYPE_PALETTE | GFX_RESOURCE_MEMORY_RAM | GFX_RESOURCE_COMP_NONE)	Palette type, located in RAM and no compression.

Description

Typedef: GFX_RESOURCE

This enumeration lists the different types of resources, how they are accessed and how the data of the resource will be interpreted when rendered.

A resource in the library has three major characteristics:

1. Location or source of the resource - The following are the recognized sources of resources
   - internal flash memory type
   - internal RAM type
   - external memory type
   - external EDS memory type
- - external via File System Service
2. Data types of the resource - The following are the supported types of resources:
   - image
   - font
   - palette
   - binary data
3. Data compression type of the resource - the following are the supported compression of data:
   - RLE - Run length encoded compression. This type of compression is only available for 8 and 4 bpp bitmaps.
   - IPU - Compressed data is encoded using the DEFLATE algorithm with fixed Huffman codes; dynamic Huffman codes are not supported.

The first four types indicates the location of the resource. The next five types indicates which kind of resource and the next 3 types indicates if the resource data is compressed or not.

By combining these three groups, a resource can be described fully and accessed appropriately in the library. For example: By combining the location, type and compression into one type, the library can pass the resource type parameter when rendering the resource. GFX_RESOURCE_MCHP_MBITMAP_EXTERNAL_RLE = (GFX_RESOURCE_TYPE_MCHP_MBITMAP | GFX_RESOURCE_MEMORY_EXTERNAL | GFX_RESOURCE_COMP_RLE)

Each type will determine how the library will access the resource when rendering.

Remarks

None.

GFX_RESOURCE_BINARY Structure

Defines the structure used for the binary type resource.

File

gfx_types_resource.h

C

```
typedef struct {
  union {
    uint32_t extAddress;
    uint8_gfx_image_prog * progByteAddress;
    uint16_gfx_image_prog * progWordAddress;
    const char * constAddress;
    char * ramAddress;
    __eds__ char * edsAddress;
  } location;
  uint32_t size;
  uint32_t param1;
  uint32_t param2;
} GFX_RESOURCE_BINARY;
```

Members

Members	Description
union { uint32_t extAddress; uint8_gfx_image_prog * progByteAddress; uint16_gfx_image_prog * progWordAddress; const char * constAddress; char * ramAddress; __eds__ char * edsAddress; } location;	This defines the location of the binary resource in memory. Depending on the type, the address location is interpreted accordingly.
uint32_t extAddress;	An external address.
uint8_gfx_image_prog * progByteAddress;	An 8-bit addresses in the program section.
uint16_gfx_image_prog * progWordAddress;	A 16-bit addresses in the program section.
const char * constAddress;	An addresses in constant space in flash.
char * ramAddress;	for addresses in RAM.
__eds__ char * edsAddress;	for addresses in EDS.
uint32_t size;	The size of the binary data in bytes.
uint32_t param1;	Parameters used for the GFX_RESOURCE. Depending on the GFX_RESOURCE type definition of param1 can change. For IPU and RLE compressed images, param1 indicates the compressed size of the image.
uint32_t param2;	Parameters used for the GFX_RESOURCE. Depending on the GFX_RESOURCE type

Description

Typedef: GFX_RESOURCE_BINARY

Defines the structure used for the binary type resource.

Remarks

None.

GFX_RESOURCE_FONT Structure

Defines the structure used for the font type resource.

File

gfx_types_resource.h

C

```
typedef struct {
  union {
    uint32_t extAddress;
    GFX_FONT_SPACE char * progByteAddress;
    char * ramAddress;
    __eds__ char * edsAddress;
  } location;
  GFX_FONT_HEADER header;
} GFX_RESOURCE_FONT;
```

Members

Members	Description
union { uint32_t extAddress; GFX_FONT_SPACE char * progByteAddress; char * ramAddress; __eds__ char * edsAddress; } location;	This defines the location of the font resource in memory. Depending on the type, the address location is interpreted accordingly.
uint32_t extAddress;	An external address.
GFX_FONT_SPACE char * progByteAddress;	An 8-bit addresses in the program section.
char * ramAddress;	An addresses in RAM.
__eds__ char * edsAddress;	An addresses in EDS.
GFX_FONT_HEADER header;	The header that describes the font resource.

Description

Typedef: GFX_RESOURCE_FONT

Defines the structure used for the font type resource.

Remarks

None.

GFX_RESOURCE_HDR Structure

Defines the structure used for the resource types.

File

gfx_types_resource.h

C

```
typedef struct {
  GFX_RESOURCE type;
  uint16_t ID;
  union {
    GFX_RESOURCE_IMAGE image;
    GFX_RESOURCE_FONT font;
    GFX_RESOURCE_BINARY binary;
    GFX_RESOURCE_PALETTE palette;
  } resource;
} GFX_RESOURCE_HDR;
```

Members

Members	Description
GFX_RESOURCE type;	Graphics resource type, determines the type and location of data
uint16_t ID;	memory ID, user defined value to differentiate between graphics resources of the same type When using EDS_EPMP the following ID values are reserved and used by the Microchip display driver 0 - reserved (do not use) 1 - reserved for base address of EPMP CS1 2 - reserved for base address of EPMP CS2
union { GFX_RESOURCE_IMAGE image; GFX_RESOURCE_FONT font; GFX_RESOURCE_BINARY binary; GFX_RESOURCE_PALETTE palette; } resource;	This defines the type of the resource. Depending on the type, the resource is accessed and rendered accordingly.
GFX_RESOURCE_IMAGE image;	Resource is an image.
GFX_RESOURCE_FONT font;	Resource is font.
GFX_RESOURCE_BINARY binary;	Resource is binary.
GFX_RESOURCE_PALETTE palette;	Resource is palette.

Description

Typedef: GFX_RESOURCE_HDR

Defines the common structure used for all the graphics resources.

Remarks

None.

GFX_RESOURCE_IMAGE Structure

Defines the structure used for the image type resource.

File

gfx_types_resource.h

C

```
typedef struct {
  union {
    uint32_t extAddress;
```

```
    uint8_gfx_image_prog * progByteAddress;
    uint16_gfx_image_prog * progWordAddress;
    const char * constAddress;
    char * ramAddress;
    __eds__ char * edsAddress;
  } location;
  uint16_t width;
  uint16_t height;
  union {
    uint32_t compressedSize;
    uint32_t reserved;
  } parameter1;
  union {
    uint32_t rawSize;
    uint32_t reserved;
  } parameter2;
  uint8_t colorDepth;
  uint8_t type;
  uint16_t paletteID;
} GFX_RESOURCE_IMAGE;
```

Members

Members	Description
union { uint32_t extAddress; uint8_gfx_image_prog * progByteAddress; uint16_gfx_image_prog * progWordAddress; const char * constAddress; char * ramAddress; __eds__ char * edsAddress; } location;	This defines the location of the image resource in memory. Depending on the type, the address location is interpreted accordingly.
uint32_t extAddress;	An external address.
uint8_gfx_image_prog * progByteAddress;	An 8-bit addresses in the program section.
uint16_gfx_image_prog * progWordAddress;	A 16-bit addresses in the program section.
const char * constAddress;	An addresses in constant space in flash.
char * ramAddress;	An addresses in RAM.
__eds__ char * edsAddress;	An addresses in EDS.
uint16_t width;	The width of the image.
uint16_t height;	The height of the image.
union { uint32_t compressedSize; uint32_t reserved; } parameter1;	A generic parameter 1 that changes in usage depending on the type of the resource.
uint32_t compressedSize;	Parameters used for the GFX_RESOURCE. Depending on the GFX_RESOURCE type definition of param1 can change. For IPU and RLE compressed images, param1 indicates the compressed size of the image.
union { uint32_t rawSize; uint32_t reserved; } parameter2;	A generic parameter 2 that changes in usage depending on the type of the resource.
uint32_t rawSize;	Parameters used for the GFX_RESOURCE. Depending on the GFX_RESOURCE type definition of param2 can change. For IPU and RLE compressed images, param2 indicates the uncompressed size of the image.
uint8_t colorDepth;	The color depth of the image.
uint8_t type;	The type of image, information on how to render the image 0x00 - no compression, palette is present for color depth = 8, 4 and 1 BPP 0x10 - palette is provided as a separate object (see PALETTE_HEADER) for color depth = 8, 4, and 1 BPP, ID to the palette is embedded color depth = 8, 4, and 1 BPP, in the bitmap. 0xYY - reserved
uint16_t paletteID;	The palette ID, if type == MCHP_BITMAP_PALETTE_STR (0x10), this represents the unique ID of the palette being used

Description

Typedef: GFX_RESOURCE_IMAGE

Defines the structure used for the image type resource.

Remarks

None.

GFX_RESOURCE_PALETTE Structure

Defines the structure used for the palette type resource.

File

gfx_types_resource.h

C

```
typedef struct {
  union {
    uint32_t extAddress;
    const GFX_PALETTE_ENTRY * constByteAddress;
    GFX_PALETTE_ENTRY * ramAddress;
  } location;
  uint16_t numberOfEntries;
  uint16_t paletteID;
} GFX_RESOURCE_PALETTE;
```

Members

Members	Description
union { uint32_t extAddress; const GFX_PALETTE_ENTRY * constByteAddress; GFX_PALETTE_ENTRY * ramAddress; } location;	This defines the location of the palette resource in memory. Depending on the type, the address location is interpreted accordingly.
uint32_t extAddress;	An external address.
const GFX_PALETTE_ENTRY * constByteAddress;	An addresses in constant space in flash.
GFX_PALETTE_ENTRY * ramAddress;	An addresses in RAM.
uint16_t numberOfEntries;	The number of color entries in the palette resource.
uint16_t paletteID;	Unique ID of the palette that will match the ID in the GFX_RESOURCE_HDR if the image has palette removed and supplied as a separate object.

Description

Typedef: GFX_RESOURCE_PALETTE

Defines the structure used for the palette type resource.

Remarks

None.

GFX_STATUS Enumeration

Rendering status.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_STATUS_FAILURE,
  GFX_STATUS_SUCCESS
} GFX_STATUS;
```

Members

Members	Description
GFX_STATUS_FAILURE	Rendering is not yet performed, or has started but not yet finished.
GFX_STATUS_SUCCESS	Rendering has been performed.

Description

Typedef: GFX_STATUS

When rendering into the frame buffer, it is sometimes necessary to return the status of the rendering. This is especially true when using hardware to render primitive shapes or when a FIFO is used to render primitive shapes.

Remarks

None.

GFX_STATUS_BIT Enumeration

Additional rendering status.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_STATUS_FAILURE_BIT,
  GFX_STATUS_SUCCESS_BIT,
  GFX_STATUS_ERROR_BIT,
  GFX_STATUS_BUSY_BIT,
  GFX_STATUS_READY_BIT
} GFX_STATUS_BIT;
```

Members

Members	Description
GFX_STATUS_FAILURE_BIT	Rendering failed. Depending on the driver used, hardware may or may not recover.
GFX_STATUS_SUCCESS_BIT	Rendering is successful.
GFX_STATUS_ERROR_BIT	Rendering resulted in an error and cannot recover.
GFX_STATUS_BUSY_BIT	Rendering cannot proceed due to a common resource is busy.
GFX_STATUS_READY_BIT	Rendering can proceed.

Description

Typedef: GFX_STATUS_BIT

The following rendering status types are available for a detailed description of the status of rendering.

Remarks

None.

int16_gfx_image_prog Type

File

gfx_types_image.h

C

```
typedef int16_prog_pack int16_gfx_image_prog;
```

Description

This is type int16_gfx_image_prog.

int32_gfx_image_prog Type

File

gfx_types_image.h

C

```
typedef int32_prog_pack int32_gfx_image_prog;
```

Description

This is type int32_gfx_image_prog.

int8_gfx_image_prog Type

File

gfx_types_image.h

C

```
typedef int8_prog_pack int8_gfx_image_prog;
```

Description

This is type int8_gfx_image_prog.

uint16_gfx_image_prog Type

File

gfx_types_image.h

C

```
typedef uint16_prog_pack uint16_gfx_image_prog;
```

Description

This is type uint16_gfx_image_prog.

uint32_gfx_image_prog Type

File

gfx_types_image.h

C

```
typedef uint32_prog_pack uint32_gfx_image_prog;
```

Description

This is type uint32_gfx_image_prog.

uint8_gfx_image_prog Type

File

gfx_types_image.h

C

```
typedef uint8_prog_pack uint8_gfx_image_prog;
```

Description

GFX Image data Types

GFX_ALIGNMENT Enumeration

Summary of the different text alignment supported in the library.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_ALIGN_LEFT,
  GFX_ALIGN_HCENTER,
  GFX_ALIGN_RIGHT,
  GFX_ALIGN_TOP,
  GFX_ALIGN_VCENTER,
  GFX_ALIGN_BOTTOM,
```

```
  GFX_ALIGN_CENTER
} GFX_ALIGNMENT;
```

Members

Members	Description
GFX_ALIGN_LEFT	left aligned text
GFX_ALIGN_HCENTER	horizontal center aligned text
GFX_ALIGN_RIGHT	right aligned text
GFX_ALIGN_TOP	top aligned text
GFX_ALIGN_VCENTER	vertical center aligned text
GFX_ALIGN_BOTTOM	bottom aligned text
GFX_ALIGN_CENTER	vertical & horizontal center aligned text

Description

Typedef: GFX_ALIGNMENT

This lists all the different text alignment supported in the library. The text alignment is used in GFX_TextStringBoxDraw().

Remarks

None.

GFX_BACKGROUND Structure

Background information structure.

File

gfx_types_macros.h

C

```
typedef struct {
  uint16_t left;
  uint16_t top;
  GFX_BACKGROUND_TYPE type;
  GFX_RESOURCE_HDR * pImage;
  GFX_COLOR color;
} GFX_BACKGROUND;
```

Members

Members	Description
uint16_t left;	Horizontal starting position of the background.
uint16_t top;	Horizontal starting position of the background.
GFX_BACKGROUND_TYPE type;	Specifies the type of background to use.
GFX_RESOURCE_HDR * pImage;	Pointer to the background image used. Set this to NULL if not used.
GFX_COLOR color;	If the background image is NULL, background is assumed to be this color.

Description

Typedef: GFX_BACKGROUND

Structure describing the background information. Useful in refreshing the whole screen or an area of the screen. The background information, when used, can be one of the enumerated types in GFX_BACKGROUND_TYPE.

- pBackGroundImage - pointer to a GFX_RESOURCE_HEADER with the type GFX_RESOURCE_TYPE_MCHP_MBITMAP.
- backGroundColor - value of the background color used when pBackgroundImage is NULL.

The pBackGroundImage has the high priority and will assume an image is used as a background when the pointer is initialized. If this is NULL the backgroundColor is assumed to be the background.

Remarks

None.

GFX_BACKGROUND_TYPE Enumeration

Specifies the different background fill types.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_BACKGROUND_NONE,
  GFX_BACKGROUND_COLOR,
  GFX_BACKGROUND_IMAGE,
  GFX_BACKGROUND_DISPLAY_BUFFER
} GFX_BACKGROUND_TYPE;
```

Members

Members	Description
GFX_BACKGROUND_NONE	No background information set.
GFX_BACKGROUND_COLOR	Background is set to a color.
GFX_BACKGROUND_IMAGE	Background is set to an image.
GFX_BACKGROUND_DISPLAY_BUFFER	Background is set to the current content of the display buffer. This requires support of GFX_PixelArrayGet().

Description

Typedef: GFX_FILL_STYLE

Defines the types of background information. Knowing the background information allows easier refresh of an area on the display buffer.

Remarks

None.

GFX_DOUBLE_BUFFERING_MODE Structure

Structure used for double buffering management.

File

gfx_types_macros.h

C

```
typedef struct {
  GFX_FEATURE_STATUS gfxDoubleBufferFeature;
  GFX_FEATURE_STATUS gfxDoubleBufferFeatureRecentStatus;
  GFX_FEATURE_STATUS gfxDoubleBufferRequestSync;
  GFX_FEATURE_STATUS gfxDoubleBufferFullSync;
  GFX_RECTANGULAR_AREA gfxDoubleBufferAreas[GFX_MAX_INVALIDATE_AREAS];
  uint16_t gfxUnsyncedAreaCount;
} GFX_DOUBLE_BUFFERING_MODE;
```

Members

Members	Description
GFX_FEATURE_STATUS gfxDoubleBufferFeature;	The status of the double buffering feature.
GFX_FEATURE_STATUS gfxDoubleBufferFeatureRecentStatus;	The status of the double buffering feature.
GFX_FEATURE_STATUS gfxDoubleBufferRequestSync;	The status of the request for synchronization.
GFX_FEATURE_STATUS gfxDoubleBufferFullSync;	The status of the full synchronization request.
GFX_RECTANGULAR_AREA gfxDoubleBufferAreas[GFX_MAX_INVALIDATE_AREAS];	The array of rectangular areas that needs synchronization.
uint16_t gfxUnsyncedAreaCount;	The current count of unsynchronized areas.

Description

Typedef: GFX_RECTANGULAR_AREA

Structure describing the double buffering states of the frame and draw buffer. This saves the current states of the feature for management by the graphics library. Double buffering is only available to selected configurations.

Remarks

None.

GFX_FEATURE_STATUS Enumeration

States of a feature that can be enabled/disabled at run time.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_FEATURE_PAUSED,
  GFX_FEATURE_ENABLED,
  GFX_FEATURE_DISABLED
} GFX_FEATURE_STATUS;
```

Members

Members	Description
GFX_FEATURE_PAUSED	The feature is paused.
GFX_FEATURE_ENABLED	The feature is enabled.
GFX_FEATURE_DISABLED	The feature is disabled.

Description

Typedef: GFX_FEATURE_STATUS

Defines the states of a Graphics Library feature that can be enabled/disabled at run time.

Remarks

None.

GFX_FILL_STYLE Enumeration

Specifies the available fill styles.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_FILL_STYLE_NONE,
  GFX_FILL_STYLE_COLOR,
  GFX_FILL_STYLE_ALPHA_COLOR,
  GFX_FILL_STYLE_GRADIENT_DOWN,
  GFX_FILL_STYLE_GRADIENT_UP,
  GFX_FILL_STYLE_GRADIENT_RIGHT,
  GFX_FILL_STYLE_GRADIENT_LEFT,
  GFX_FILL_STYLE_GRADIENT_DOUBLE_VER,
  GFX_FILL_STYLE_GRADIENT_DOUBLE_HOR
} GFX_FILL_STYLE;
```

Members

Members	Description
GFX_FILL_STYLE_NONE	no fill
GFX_FILL_STYLE_COLOR	color fill
GFX_FILL_STYLE_ALPHA_COLOR	color fill with alpha blending
GFX_FILL_STYLE_GRADIENT_DOWN	gradient, vertical-down direction
GFX_FILL_STYLE_GRADIENT_UP	gradient, vertical-up direction
GFX_FILL_STYLE_GRADIENT_RIGHT	gradient, horizontal-right direction
GFX_FILL_STYLE_GRADIENT_LEFT	gradient, horizontal-left direction
GFX_FILL_STYLE_GRADIENT_DOUBLE_VER	gradient, vertical-up/down direction
GFX_FILL_STYLE_GRADIENT_DOUBLE_HOR	gradient, horizontal-left/right direction

Description

Typedef: GFX_FILL_STYLE

This enumeration specifies the available fill styles used in the library.

Remarks

None.

GFX_FONT_ANTIALIAS_TYPE Enumeration

Summary of the transparency types in text anti-aliasing.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_FONT_ANTIALIAS_OPAQUE,
  GFX_FONT_ANTIALIAS_TRANSLUCENT
} GFX_FONT_ANTIALIAS_TYPE;
```

Members

Members	Description
GFX_FONT_ANTIALIAS_OPAQUE	Mid colors are calculated only once while rendering each character. This is ideal for rendering text over a constant background.
GFX_FONT_ANTIALIAS_TRANSLUCENT	Mid colors are calculated for every necessary pixel. This feature is useful when rendering text over an image or when the background is not one flat color.

Description

Typedef: GFX_FONT_ANTIALIAS_TYPE

Transparency type when rendering characters with anti-aliasing.

Remarks

None.

GFX_FONT_SPACE Macro

File

gfx_types_font.h

C

```
#define GFX_FONT_SPACE const
```

Description

Font space section. The fonts can be located in psv (constant) or program space in PIC24/dsPIC MCUs. This define allows for switching of the pointer type used to access the font structure in memory See: GraphicsConfig.h for the application define.

GFX_UXCHAR Macro

File

gfx_types_font.h

C

```
#define GFX_UXCHAR uint8_t
```

Description

This is macro GFX_UXCHAR.

GFX_XCHAR Macro

Configuration for the text character size.

File

gfx_types_font.h

C

```
#define GFX_XCHAR char
```

Description

GFX_CONFIG_FONT_CHAR_SIZE

This configuration sets the text character size used in the library.

To enable support for unicode fonts, GFX_CONFIG_FONT_CHAR_SIZE must be defined as 16 (size is 16 bits). For standard ASCII fonts, this can be defined as 8. This changes the GFX_XCHAR definition. See GFX_XCHAR for details.

Set in configuration	GFX_XCHAR	Description
#define GFX_CONFIG_FONT_CHAR_SIZE 16	#define GFX_XCHAR uint16_t	Use multi-byte characters (0-2^16 range).
#define GFX_CONFIG_FONT_CHAR_SIZE 8	#define GFX_XCHAR uint8_t	Use byte characters (0-255 range).

Remarks

None.

GRAPHICS_LIBRARY_VERSION Macro

File

gfx.h

C

```
#define GRAPHICS_LIBRARY_VERSION 0x0400
```

Description

////////////////////// GRAPHICS_LIBRARY_VERSION ////////////////////// MSB is version, LSB is subversion

GFX_COMP_MASK Macro

File

gfx_types_resource.h

C

```
#define GFX_COMP_MASK 0xF000
```

Description

This is macro GFX_COMP_MASK.

GFX_MEM_MASK Macro

File

gfx_types_resource.h

C

```
#define GFX_MEM_MASK 0x000F
```

Description

This is macro GFX_MEM_MASK.

GFX_TYPE_MASK Macro

File

gfx_types_resource.h

C

```
#define GFX_TYPE_MASK 0x0F00
```

Description

This is macro GFX_TYPE_MASK.

MCHP_BITMAP_NORMAL Macro

File

gfx_types_resource.h

C

```
#define MCHP_BITMAP_NORMAL 0x00                // no compression, palette is present
```

Description

no compression, palette is present for color depth = 8, 4 and 1 BPP

MCHP_BITMAP_PALETTE_STR Macro

File

gfx_types_resource.h

C

```
#define MCHP_BITMAP_PALETTE_STR 0x10                // palette is provided as a separate
```

Description

palette is provided as a separate object (see PALETTE_HEADER) for color depth = 8, 4, and 1 BPP, ID to the palette is embedded in the bitmap.

int16_prog Type

File

gfx_types_image.h

C

```
typedef __prog__ int16_t int16_prog;
```

Description

This is type int16_prog.

int16_prog_pack Type

File

gfx_types_image.h

C

```
typedef __pack_upper_byte int16_t int16_prog_pack;
```

Description

This is type int16_prog_pack.

int32_prog Type

File

gfx_types_image.h

C

```
typedef __prog__ int32_t int32_prog;
```

Description

This is type int32_prog.

int32_prog_pack Type

File

gfx_types_image.h

C

```
typedef __pack_upper_byte int32_t int32_prog_pack;
```

Description

This is type int32_prog_pack.

int8_prog Type

File

gfx_types_image.h

C

```
typedef __prog__ int8_t int8_prog;
```

Description

This is type int8_prog.

int8_prog_pack Type

File

gfx_types_image.h

C

```
typedef __pack_upper_byte int8_t int8_prog_pack;
```

Description

This is type int8_prog_pack.

uint16_prog Type

File

gfx_types_image.h

C

```
typedef __prog__ uint16_t uint16_prog;
```

Description

This is type uint16_prog.

uint16_prog_pack Type

File

gfx_types_image.h

C

```
typedef __pack_upper_byte uint16_t uint16_prog_pack;
```

Description

This is type uint16_prog_pack.

uint32_prog Type

File

gfx_types_image.h

C

```
typedef __prog__ uint32_t uint32_prog;
```

Description

This is type uint32_prog.

uint32_prog_pack Type

File

gfx_types_image.h

C

```
typedef __pack_upper_byte uint32_t uint32_prog_pack;
```

Description

This is type uint32_prog_pack.

uint8_prog Type

File

gfx_types_image.h

C

```
typedef __prog__ uint8_t uint8_prog;
```

Description

General program space data types

uint8_prog_pack Type

File

gfx_types_image.h

C

```
typedef __pack_upper_byte uint8_t uint8_prog_pack;
```

Description

This is type uint8_prog_pack.

GFX_CLIENT_OBJ Structure

GFX client object structure.

File

gfx.h

C

```
typedef struct {
  GFX_CLIENT_STATUS clientState;
  bool inUse;
  void * obj;
  SYS_MODULE_INDEX halIndex;
  GFX_HANDLE primitiveHandle;
  GFX_HANDLE golHandle;
```

```
} GFX_CLIENT_OBJ;
```

Members

Members	Description
GFX_CLIENT_STATUS clientState;	Client status
bool inUse;	Set to true if this object is in use
void * obj;	Object from which the client is derived
SYS_MODULE_INDEX halIndex;	Save the index of the driver

Description

GFX client object structure

GFX client object structure.

Remarks

None.

GFX_CLIENT_STATUS Enumeration

Enumerated data type that identifies the GFX Module Client Status.

File

gfx.h

C

```
typedef enum {
  GFX_CLIENT_STATUS_CLOSED,
  GFX_CLIENT_STATUS_READY
} GFX_CLIENT_STATUS;
```

Members

Members	Description
GFX_CLIENT_STATUS_CLOSED	Client is closed or the specified handle is invalid
GFX_CLIENT_STATUS_READY	Client is ready

Description

GFX Client Status

This enumeration defines the possible status of the GFX Module Client. It is returned by the () function.

Remarks

None.

GFX_OBJ Structure

Defines the object required for interaction with the GFX Library.

File

gfx.h

C

```
typedef struct _GFX_OBJ {
  SYS_STATUS status;
  SYS_MODULE_INDEX index;
  bool inUse;
  GFX_STATES state;
  bool IsExclusive;
  uint8_t numClients;
  GFX_OBJECT_TASK task;
  GFX_INIT * initData;
  GFX_HANDLE gfxHandle;
  SYS_MODULE_OBJ primitiveObject;
  SYS_MODULE_OBJ golObject;
} GFX_OBJ;
```

Members

Members	Description
SYS_STATUS status;	Required : The status of the module
SYS_MODULE_INDEX index;	Object Index
bool inUse;	Required : Flag to indicate in use
GFX_STATES state;	GFX machine state
bool IsExclusive;	Flag to indicate that PMP is used in exclusive access mode
uint8_t numClients;	Optional: number of clients possible with the software instance
GFX_OBJECT_TASK task;	State of the task

Description

GFX Library Instance Object

This defines the object required for interaction with the GFX Library. This object exists once per application instance of the library.

Remarks

None.

GFX_OBJECT_TASK Enumeration

Lists the different states that GFX task routine can have.

File

gfx.h

C

```
typedef enum {
  GFX_TASK_STATE_OPEN_MODULE,
  GFX_TASK_STATE_RUNNING
} GFX_OBJECT_TASK;
```

Members

Members	Description
GFX_TASK_STATE_OPEN_MODULE	open module for client

Description

GFX task states

This enumeration lists the different states that GFX task routine can have.

Remarks

None.

GFX_STATES Enumeration

Defines the various states that can be achieved by the GFX operation.

File

gfx.h

C

```
typedef enum {
  GFX_STATE_BUSY,
  GFX_STATE_INIT,
  GFX_STATE_INITIALIZED
} GFX_STATES;
```

Members

Members	Description
GFX_STATE_BUSY	Module state busy
GFX_STATE_INIT	Module state init

Description

GFX Machine States

This enumeration defines the various states that can be achieved by the HAL operation.

Remarks

None.

FileFeof Type

File

gfx_image_decoder.h

C

```
typedef bool (* FileFeof)(void *stream);
```

Description

This is type FileFeof.

FileRead Type

File

gfx_image_decoder.h

C

```
typedef size_t (* FileRead)(void *ptr, size_t size, size_t n, void *stream);
```

Description

Typedefs of the used File System APIs

FileSeek Type

File

gfx_image_decoder.h

C

```
typedef bool (* FileSeek)(void *stream, long offset, int whence);
```

Description

This is type FileSeek.

FileTell Type

File

gfx_image_decoder.h

C

```
typedef uint32_t (* FileTell)(void *fo);
```

Description

This is type FileTell.

IMG_FILE_FORMAT Enumeration

File

gfx_image_decoder.h

C

```
typedef enum _IMG_FILE_FORMAT {
```

```
  IMG_NONE = 0,
  IMG_BMP,
  IMG_JPEG,
  IMG_GIF
} IMG_FILE_FORMAT;
```

Members

Members	Description
IMG_BMP	Bitmap image
IMG_JPEG	JPEG image
IMG_GIF	GIF image

Description

IMG_ImageFileFormat specifies all the supported image file formats

IMG_FILE_SYSTEM_API Structure

File

gfx_image_decoder.h

C

```
typedef struct _IMG_FILE_SYSTEM_API {
  FileRead pFread;
  FileSeek pFseek;
  FileTell pFtell;
  FileFeof pFeof;
} IMG_FILE_SYSTEM_API;
```

Description

IMG_FileSystemAPI holds function pointers to the used File System APIs

IMG_LOOP_CALLBACK Type

File

gfx_image_decoder.h

C

```
typedef void (* IMG_LOOP_CALLBACK)(void);
```

Description

IMG_LoopCallback is a callback function which is called in every loop of the decoding cycle so that user application can do some maintenance activities

IMG_PIXEL_OUTPUT Type

File

gfx_image_decoder.h

C

```
typedef void (* IMG_PIXEL_OUTPUT)(IMG_PIXEL_XY_RGB_888 *pPix);
```

Description

IMG_PixelOutput is a callback function which receives the color information of the output pixel

IMG_PIXEL_XY_RGB_888 Structure

File

gfx_image_decoder.h

C

```
typedef struct _IMG_PIXEL_XY_RGB_888 {
```

```
    uint16_t X;
    uint16_t Y;
    uint8_t R;
    uint8_t G;
    uint8_t B;
    uint16_t POS;
} IMG_PIXEL_XY_RGB_888;
```

Description

IMG_PixelRgb holds the RGB information of a pixel in BYTES

__IMAGEDECODER_H__ Macro

File

gfx_image_decoder.h

C

```
#define __IMAGEDECODER_H__
```

Description

This is macro __IMAGEDECODER_H__.

_PutPixel Macro

File

gfx_image_decoder.h

C

```
#define _PutPixel(x, y) if(IMG_pPixelOutput != NULL) { IMG_PixelXYColor.X = x; IMG_PixelXYColor.Y = y;
IMG_pPixelOutput(&IMG_PixelXYColor); }
```

Description

This is macro _PutPixel.

ImageAbort Macro

File

gfx_image_decoder.h

C

```
#define ImageAbort IMG_blAbortImageDecoding = 1;
```

Side Effects

None

Returns

None

Description

This function sets the Image Decoder's Abort flag so that decoding aborts in the next decoding loop.

Example

```
void callback(void);
void main(void)
{
    IMG_FILE pImageFile;
    IMG_vInitialize();
    ImageLoopCallbackRegister(callback);
    pImageFile = IMG_FOPEN("Image.jpg", "r");
    if(pImageFile == NULL)
    {
              <- Error handling ->
```

```
    }
    IMG_bFullScreenDecode(pImageFile, IMG_JPEG, NULL, NULL);
    IMG_FCLOSE(pImageFile);
    while(1);
}

void callback(void)
{
    if(<- check for abort condition ->)
    {
        ImageAbort();
    }
}
```

Function

void ImageAbort(void)

ImageFullScreenDecode Macro

File

gfx_image_decoder.h

C

```
#define ImageFullScreenDecode(pImageFile, eImgFormat, pFileAPIs, pPixelOutput) \
        ImageDecode(pImageFile, eImgFormat, 0, 0, IMG_SCREEN_WIDTH, IMG_SCREEN_HEIGHT, (IMG_ALIGN_CENTER |
IMG_DOWN_SCALE), pFileAPIs, pPixelOutput)
```

Side Effects

None

Returns

Error code -> 0 means no error

Description

This function decodes and displays the image on the screen in fullscreen mode with center aligned and downscaled if required

Example

```
  void main(void)
{
    IMG_FILE pImageFile;
    IMG_vInitialize();
    pImageFile = IMG_FOPEN("Image.jpg", "r");
    if(pImageFile == NULL)
    {
              <- Error handling ->
    }
    IMG_bFullScreenDecode(pImageFile, IMG_JPEG, NULL, NULL);
    IMG_FCLOSE(pImageFile);
    while(1);
}
```

Function

uint8_t ImageFullScreenDecode(IMG_FILE *pImageFile, IMG_FILE_FORMAT eImgFormat, IMG_FILE_SYSTEM_API pFileAPIs, IMG_PIXEL_OUTPUT pPixelOutput)

IMG_ALIGN_CENTER Macro

File

gfx_image_decoder.h

C

```
#define IMG_ALIGN_CENTER 0x0001
```

Description

Flags

IMG_DECODE_ABORTED Macro

File

gfx_image_decoder.h

C

```
#define IMG_DECODE_ABORTED 0xFF
```

Description

This is macro IMG_DECODE_ABORTED.

IMG_DOWN_SCALE Macro

File

gfx_image_decoder.h

C

```
#define IMG_DOWN_SCALE 0x0002
```

Description

This is macro IMG_DOWN_SCALE.

IMG_FEOF Macro

File

gfx_image_decoder.h

C

```
#define IMG_FEOF IMG_pFileAPIs->pFeof
```

Description

This is macro IMG_FEOF.

IMG_FILE Macro

File

gfx_image_decoder.h

C

```
#define IMG_FILE void
```

Description

This is macro IMG_FILE.

IMG_FREAD Macro

File

gfx_image_decoder.h

C

```
#define IMG_FREAD IMG_pFileAPIs->pFread
```

Description

This is macro IMG_FREAD.

IMG_FSEEK Macro

File

gfx_image_decoder.h

C

```
#define IMG_FSEEK IMG_pFileAPIs->pFseek
```

Description

This is macro IMG_FSEEK.

IMG_FTELL Macro

File

gfx_image_decoder.h

C

```
#define IMG_FTELL IMG_pFileAPIs->pFtell
```

Description

This is macro IMG_FTELL.

IMG_SCREEN_HEIGHT Macro

File

gfx_image_decoder.h

C

```
#define IMG_SCREEN_HEIGHT (GetMaxY()+1)
```

Description

This is macro IMG_SCREEN_HEIGHT.

IMG_SCREEN_WIDTH Macro

File

gfx_image_decoder.h

C

```
#define IMG_SCREEN_WIDTH (GetMaxX()+1)
```

Description

The individual image decoders use these defines instead of directly using those provided in the Display driver file

IMG_vCheckAndAbort Macro

File

gfx_image_decoder.h

C

```
#define IMG_vCheckAndAbort \
        if(IMG_blAbortImageDecoding == 1)   \
        {                                   \
              IMG_blAbortImageDecoding = 0; \
              return IMG_DECODE_ABORTED;    \
        }
```

Description

This is macro IMG_vCheckAndAbort.

IMG_vLoopCallback Macro

File

gfx_image_decoder.h

C

```
#define IMG_vLoopCallback
```

Description

This is macro IMG_vLoopCallback.

IMG_vPutPixel Macro

File

gfx_image_decoder.h

C

```
#define IMG_vPutPixel(x, y) if(IMG_bDownScalingFactor <= 1)\
                                {                                \
                                    _PutPixel(IMG_wStartX + (x), IMG_wStartY + (y));\
                                }                                \
                                else if((x)%IMG_bDownScalingFactor == 0 && (y)%IMG_bDownScalingFactor ==
0)\
                                {                                \
                                    _PutPixel(IMG_wStartX + ((x)/IMG_bDownScalingFactor), IMG_wStartY +
((y)/IMG_bDownScalingFactor)); \
                                }
```

Description

This is macro IMG_vPutPixel.

GFX_PALETTE_ENTRY Union

Defines the union used for each entry in the palette table.

File

gfx_types_palette.h

C

```
typedef union {
  uint16_t value;
  struct {
    uint16_t r : 5;
    uint16_t g : 6;
    uint16_t b : 5;
  } color;
  struct {
    uint16_t reserved : 12;
    uint16_t luma : 4;
  } monochrome;
} GFX_PALETTE_ENTRY;
```

Members

Members	Description
uint16_t value;	a 16-bit value representing a color or palette entry
struct { uint16_t r : 5; uint16_t g : 6; uint16_t b : 5; } color;	color value in 5-6-5 RGB format

uint16_t r : 5;	represents the RED component
uint16_t g : 6;	represents the GREEN component
uint16_t b : 5;	represents the BLUE component
struct { uint16_t reserved : 12; uint16_t luma : 4; } monochrome;	monochrome LUMA value
uint16_t reserved : 12;	reserved, used as a filler
uint16_t luma : 4;	monochrome LUMA value

Description

Typedef: GFX_PALETTE_ENTRY

Defines the union used for each entry in the palette table specified by GFX_RESOURCE_PALETTE structure.

There are two types of palette:

- For TFT: color is defined as 5-6-5 RGB format where 5-bits for RED, 6-bits for GREEN and 5-bits for BLUE.
- For Monochrome: 4 bits are used to represent the luma.

Remarks

None.

GFX_TEXT_CURSOR_TYPE Enumeration

Enumeration defines types of text cursor supported.

File

gfx_types_macros.h

C

```
typedef enum {
  GFX_TEXT_CURSOR_TYPE_UNDERSCORE
} GFX_TEXT_CURSOR_TYPE;
```

Members

Members	Description
GFX_TEXT_CURSOR_TYPE_UNDERSCORE	Text cursor of type underscore

Description

Typedef: GFX_TEXT_CURSOR_TYPE

This enumeration defines types of cursor can be drawn.

Remarks

None.

IMG_LOOP_FRAMECOMPLETE Type

File

gfx_image_decoder.h

C

```
typedef bool (* IMG_LOOP_FRAMECOMPLETE)(uint16_t frame);
```

Description

IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities

IMG_LOOP_IMAGECOMPLETE Type

File

gfx_image_decoder.h

C

```
typedef void (* IMG_LOOP_IMAGECOMPLETE)(void);
```

Description

IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities

IMG_LOOP_IMAGEFAIL Type

File

gfx_image_decoder.h

C

```
typedef void (* IMG_LOOP_IMAGEFAIL)(void);
```

Description

IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities

IMG_vFrameComplete Macro

File

gfx_image_decoder.h

C

```
#define IMG_vFrameComplete if (IMG_pFrameCompleteCallbackFn != NULL) { IMG_pImageExitRequested =
IMG_pFrameCompleteCallbackFn(++IMG_pCurrentFrame); }
```

Description

This is macro IMG_vFrameComplete.

IMG_vImageComplete Macro

File

gfx_image_decoder.h

C

```
#define IMG_vImageComplete if (IMG_pImageCompleteCallbackFn != NULL) { IMG_pImageCompleteCallbackFn(); }
```

Description

This is macro IMG_vImageComplete.

IMG_vImageFail Macro

File

gfx_image_decoder.h

C

```
#define IMG_vImageFail if (IMG_pImageFailCallbackFn != NULL) { IMG_pImageFailCallbackFn(); }
```

Description

This is macro IMG_vImageFail.

IMG_vPixelArray Macro

File

gfx_image_decoder.h

C

```
#define IMG_vPixelArray(pos) IMG_PixelXYColor.POS = pos;
```

Description

This is macro IMG_vPixelArray.

IMG_vSetColor Macro

File

gfx_image_decoder.h

C

```
#define IMG_vSetColor(r, g, b) IMG_PixelXYColor.R = r; IMG_PixelXYColor.G = g; IMG_PixelXYColor.B = b;
```

Description

This is macro IMG_vSetColor.

Files

Files

Name	Description
gfx.h	The header file joins all header files used in the graphics library and contains compile options and defaults.
gfx_colors.h	The header file contains default color definitions for each color depth.
gfx_common.h	This is file gfx_common.h.
gfx_colors_w3.h	The header file contains W3 color definitions.
gfx_colors_x11.h	The header file contains X11 color definitions.
gfx_gol.h	Graphics Object Layer of the Microchip Graphics Library.
gfx_gol_button.h	This is the header file for the button object of the GOL.
gfx_gol_check_box.h	This is the header file for the check box object of the GOL.
gfx_gol_custom_control.h	This is the header file for the custom control object of the GOL.
gfx_gol_digital_meter.h	This is the header file for the digital meter object of the GOL.
gfx_gol_edit_box.h	This is the header file for the edit box object of the GOL.
gfx_gol_group_box.h	This is the header file for the group box object of the GOL.
gfx_gol_list_box.h	This is the header file for the list box object of the GOL.
gfx_gol_meter.h	This is the header file for the meter object of the GOL.
gfx_gol_picture.h	This is the header file for the picture object of the GOL.
gfx_gol_progress_bar.h	This is the header file for the progress bar object of the GOL.
gfx_gol_radio_button.h	This is the header file for the radio button object of the GOL.
gfx_gol_scan_codes.h	This is the header file for the AT keyboard codes.
gfx_gol_scheme.h	Graphics Object Layer Scheme routines/macros of the Microchip Graphics Library.
gfx_gol_scroll_bar.h	This is the header file for the scroll bar object of the GOL.
gfx_gol_static_text.h	This is the header file for the static text object of the GOL.
gfx_gol_surface.h	This is the header file for the surface object of the GOL.
gfx_gol_text_entry.h	This is the header file for the text entry object of the GOL.
gfx_gol_window.h	This is the header file for the window object of the GOL.
gfx_image_decoder.h	This implements the image decoding of the primitive layer.
gfx_primitive.h	Primitive Layer of the Microchip Graphics Library.
gfx_types_font.h	This header file defines the types used in fonts for the Microchip Graphics Library.
gfx_types_image.h	This header file defines the types used in images for the Microchip Graphics Library.
gfx_types_macros.h	Data types used in Microchip Graphics Library.
gfx_types_palette.h	This header file defines the types used in palettes for the Microchip Graphics Library.
gfx_types_resource.h	This is the definition of the Graphics Resource Types used with the Microchip Graphics Library.
gfx_config_template.h	This header file template defines all the Graphics Library configurations available.

Description

This section lists the source and header files used by the library.

gfx.h

The header file joins all header files used in the graphics library and contains compile options and defaults.

Enumerations

	Name	Description
	GFX_CLIENT_STATUS	Enumerated data type that identifies the GFX Module Client Status.
	GFX_OBJECT_TASK	Lists the different states that GFX task routine can have.
	GFX_STATES	Defines the various states that can be achieved by the GFX operation.

Functions

	Name	Description
	GFX_Close	Closes a client instance of the GFX library.
	GFX_Deinitialize	Deinitializes the specified instance of the GFX Module.
	GFX_Initialize	Initialize the GFX Library.
	GFX_Open	Opens the specified GFX library instance and returns a handle to it.
	GFX_Status	Gets the current status of the GFX Module.
	GFX_Tasks	Maintains the GFX module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events.

Macros

	Name	Description
	GRAPHICS_LIBRARY_VERSION	////////////////// GRAPHICS_LIBRARY_VERSION ////////////////////// MSB is version, LSB is subversion

Structures

	Name	Description
	_GFX_OBJ	Defines the object required for interaction with the GFX Library.
	GFX_CLIENT_OBJ	GFX client object structure.
	GFX_OBJ	Defines the object required for interaction with the GFX Library.

Description

Module for Microchip Graphics Library

This header file includes all the header files required to use the Microchip Graphics Library. Library features and options defined in the Graphics Library configurations will be included in each build.

File Name

gfx.h

Company

Microchip Technology Inc.

gfx_colors.h

The header file contains default color definitions for each color depth.

Macros

	Name	Description
	BLACK	Basic colors definitions. The basic colors defined in this section are defined for basic demo requirements. End application can define additional colors or override existing default colors. If overriding an existing default color, define the new color value before the #include of Graphics.h to avoid the compile time warning. When using palette, a different file 'PaletteColorDefines.h' has to be used instead.
	BLUE	This is macro BLUE.
	BRIGHTBLUE	This is macro BRIGHTBLUE.

	BRIGHTCYAN	This is macro BRIGHTCYAN.
	BRIGHTGREEN	This is macro BRIGHTGREEN.
	BRIGHTMAGENTA	This is macro BRIGHTMAGENTA.
	BRIGHTRED	This is macro BRIGHTRED.
	BRIGHTYELLOW	This is macro BRIGHTYELLOW.
	BROWN	This is macro BROWN.
	BURLYWOOD	This is macro BURLYWOOD.
	CYAN	This is macro CYAN.
	DARKGRAY	This is macro DARKGRAY.
	DARKORANGE	This is macro DARKORANGE.
	GOLD	This is macro GOLD.
	GRAY000	This is macro GRAY000.
	GRAY001	This is macro GRAY001.
	GRAY002	This is macro GRAY002.
	GRAY003	This is macro GRAY003.
	GRAY004	This is macro GRAY004.
	GRAY005	This is macro GRAY005.
	GRAY006	This is macro GRAY006.
	GRAY007	This is macro GRAY007.
	GRAY008	This is macro GRAY008.
	GRAY009	This is macro GRAY009.
	GRAY010	This is macro GRAY010.
	GRAY011	This is macro GRAY011.
	GRAY012	This is macro GRAY012.
	GRAY013	This is macro GRAY013.
	GRAY014	This is macro GRAY014.
	GRAY015	This is macro GRAY015.
	GRAY032	This is macro GRAY032.
	GRAY096	This is macro GRAY096.
	GRAY128	This is macro GRAY128.
	GRAY160	This is macro GRAY160.
	GRAY192	This is macro GRAY192.
	GRAY204	This is macro GRAY204.
	GRAY224	This is macro GRAY224.
	GRAY229	This is macro GRAY229.
	GRAY242	This is macro GRAY242.
	GREEN	This is macro GREEN.
	LIGHTBLUE	This is macro LIGHTBLUE.
	LIGHTCYAN	This is macro LIGHTCYAN.
	LIGHTGRAY	This is macro LIGHTGRAY.
	LIGHTGREEN	This is macro LIGHTGREEN.
	LIGHTMAGENTA	This is macro LIGHTMAGENTA.
	LIGHTORANGE	This is macro LIGHTORANGE.
	LIGHTRED	This is macro LIGHTRED.
	LIGHTYELLOW	This is macro LIGHTYELLOW.
	MAGENTA	This is macro MAGENTA.
	ORANGE	This is macro ORANGE.
	PERU	This is macro PERU.
	RED	This is macro RED.
	SADDLEBROWN	This is macro SADDLEBROWN.
	SIENNA	This is macro SIENNA.
	TAN	This is macro TAN.
	WHEAT	This is macro WHEAT.
	WHITE	This is macro WHITE.
	YELLOW	This is macro YELLOW.

Description

Module for Microchip Graphics Library

This header file provides default color definitions as well as color conversion macros. Application may or may not use this file. Application may replace this file with new color definitions that fits the application needs.

File Name

gfx_colors.h

Company

Microchip Technology Inc.

gfx_common.h

This is file gfx_common.h.

gfx_colors_w3.h

The header file contains W3 color definitions.

Macros

	Name	Description
	GFX_W3_ALICEBLUE	This is macro GFX_W3_ALICEBLUE.
	GFX_W3_ANTIQUEWHITE	This is macro GFX_W3_ANTIQUEWHITE.
	GFX_W3_AQUA	This is macro GFX_W3_AQUA.
	GFX_W3_AQUAMARINE	This is macro GFX_W3_AQUAMARINE.
	GFX_W3_AZURE	This is macro GFX_W3_AZURE.
	GFX_W3_BEIGE	This is macro GFX_W3_BEIGE.
	GFX_W3_BISQUE	This is macro GFX_W3_BISQUE.
	GFX_W3_BLACK	This is macro GFX_W3_BLACK.
	GFX_W3_BLANCHEDALMOND	This is macro GFX_W3_BLANCHEDALMOND.
	GFX_W3_BLUE	This is macro GFX_W3_BLUE.
	GFX_W3_BLUEVIOLET	This is macro GFX_W3_BLUEVIOLET.
	GFX_W3_BROWN	This is macro GFX_W3_BROWN.
	GFX_W3_BURLYWOOD	This is macro GFX_W3_BURLYWOOD.
	GFX_W3_CADETBLUE	This is macro GFX_W3_CADETBLUE.
	GFX_W3_CHARTREUSE	This is macro GFX_W3_CHARTREUSE.
	GFX_W3_CHOCOLATE	This is macro GFX_W3_CHOCOLATE.
	GFX_W3_CORAL	This is macro GFX_W3_CORAL.
	GFX_W3_CORNFLOWERBLUE	This is macro GFX_W3_CORNFLOWERBLUE.
	GFX_W3_CORNSILK	This is macro GFX_W3_CORNSILK.
	GFX_W3_CRIMSON	This is macro GFX_W3_CRIMSON.
	GFX_W3_CYAN	This is macro GFX_W3_CYAN.
	GFX_W3_DARKBLUE	This is macro GFX_W3_DARKBLUE.
	GFX_W3_DARKCYAN	This is macro GFX_W3_DARKCYAN.
	GFX_W3_darkgoldenrod	This is macro GFX_W3_darkgoldenrod.
	GFX_W3_DARKGRAY	This is macro GFX_W3_DARKGRAY.
	GFX_W3_DARKGREEN	This is macro GFX_W3_DARKGREEN.
	GFX_W3_DARKGREY	This is macro GFX_W3_DARKGREY.
	GFX_W3_DARKHAKI	This is macro GFX_W3_DARKHAKI.
	GFX_W3_DARKMAGENTA	This is macro GFX_W3_DARKMAGENTA.
	GFX_W3_DARKOLIVEGREEN	This is macro GFX_W3_DARKOLIVEGREEN.
	GFX_W3_DARKORANGE	This is macro GFX_W3_DARKORANGE.
	GFX_W3_DARKORCHID	This is macro GFX_W3_DARKORCHID.
	GFX_W3_DARKRED	This is macro GFX_W3_DARKRED.
	GFX_W3_DARKSALMON	This is macro GFX_W3_DARKSALMON.
	GFX_W3_DARKSEAGREEN	This is macro GFX_W3_DARKSEAGREEN.

	GFX_W3_DARKSLATEBLUE	This is macro GFX_W3_DARKSLATEBLUE.
	GFX_W3_DARKSLATEGRAY	This is macro GFX_W3_DARKSLATEGRAY.
	GFX_W3_DARKSLATEGREY	This is macro GFX_W3_DARKSLATEGREY.
	GFX_W3_DARKTURQUOISE	This is macro GFX_W3_DARKTURQUOISE.
	GFX_W3_DARKVIOLET	This is macro GFX_W3_DARKVIOLET.
	GFX_W3_DEEPPINK	This is macro GFX_W3_DEEPPINK.
	GFX_W3_DEEPSKYBLUE	This is macro GFX_W3_DEEPSKYBLUE.
	GFX_W3_DIMGRAY	This is macro GFX_W3_DIMGRAY.
	GFX_W3_DIMGREY	This is macro GFX_W3_DIMGREY.
	GFX_W3_DODGERBLUE	This is macro GFX_W3_DODGERBLUE.
	GFX_W3_FIREBRICK	This is macro GFX_W3_FIREBRICK.
	GFX_W3_FLORALWHITE	This is macro GFX_W3_FLORALWHITE.
	GFX_W3_FORESTGREEN	This is macro GFX_W3_FORESTGREEN.
	GFX_W3_FUCHSIA	This is macro GFX_W3_FUCHSIA.
	GFX_W3_GAINSBORO	This is macro GFX_W3_GAINSBORO.
	GFX_W3_GHOSTWHITE	This is macro GFX_W3_GHOSTWHITE.
	GFX_W3_GOLD	This is macro GFX_W3_GOLD.
	GFX_W3_GOLDENROD	This is macro GFX_W3_GOLDENROD.
	GFX_W3_GRAY	This is macro GFX_W3_GRAY.
	GFX_W3_GREEN	This is macro GFX_W3_GREEN.
	GFX_W3_GREENYELLOW	This is macro GFX_W3_GREENYELLOW.
	GFX_W3_GREY	This is macro GFX_W3_GREY.
	GFX_W3_HONEYDEW	This is macro GFX_W3_HONEYDEW.
	GFX_W3_HOTPINK	This is macro GFX_W3_HOTPINK.
	GFX_W3_INDIANRED	This is macro GFX_W3_INDIANRED.
	GFX_W3_INDIGO	This is macro GFX_W3_INDIGO.
	GFX_W3_IVORY	This is macro GFX_W3_IVORY.
	GFX_W3_LAVENDER	This is macro GFX_W3_LAVENDER.
	GFX_W3_LAVENDERBLUSH	This is macro GFX_W3_LAVENDERBLUSH.
	GFX_W3_LAWNGREEN	This is macro GFX_W3_LAWNGREEN.
	GFX_W3_LEMONCHIFFON	This is macro GFX_W3_LEMONCHIFFON.
	GFX_W3_LIGHTBLUE	This is macro GFX_W3_LIGHTBLUE.
	GFX_W3_LIGHTCORAL	This is macro GFX_W3_LIGHTCORAL.
	GFX_W3_LIGHTCYAN	This is macro GFX_W3_LIGHTCYAN.
	GFX_W3_LIGHTGOLDENRODYELLOW	This is macro GFX_W3_LIGHTGOLDENRODYELLOW.
	GFX_W3_LIGHTGRAY	This is macro GFX_W3_LIGHTGRAY.
	GFX_W3_LIGHTGREEN	This is macro GFX_W3_LIGHTGREEN.
	GFX_W3_LIGHTGREY	This is macro GFX_W3_LIGHTGREY.
	GFX_W3_LIGHTPINK	This is macro GFX_W3_LIGHTPINK.
	GFX_W3_LIGHTSALMON	This is macro GFX_W3_LIGHTSALMON.
	GFX_W3_LIGHTSKYBLUE	This is macro GFX_W3_LIGHTSKYBLUE.
	GFX_W3_LIGHTSLATEGRAY	This is macro GFX_W3_LIGHTSLATEGRAY.
	GFX_W3_LIGHTSLATEGREY	This is macro GFX_W3_LIGHTSLATEGREY.
	GFX_W3_LIGHTSTEELBLUE	This is macro GFX_W3_LIGHTSTEELBLUE.
	GFX_W3_LIGHTYELLOW	This is macro GFX_W3_LIGHTYELLOW.
	GFX_W3_LIGTHSEAGREEN	This is macro GFX_W3_LIGTHSEAGREEN.
	GFX_W3_LIME	This is macro GFX_W3_LIME.
	GFX_W3_LIMEGREEN	This is macro GFX_W3_LIMEGREEN.
	GFX_W3_LINEN	This is macro GFX_W3_LINEN.
	GFX_W3_MAGENTA	This is macro GFX_W3_MAGENTA.
	GFX_W3_MAROON	This is macro GFX_W3_MAROON.
	GFX_W3_MEDIUMAQUAMARINE	This is macro GFX_W3_MEDIUMAQUAMARINE.
	GFX_W3_MEDIUMBLUE	This is macro GFX_W3_MEDIUMBLUE.
	GFX_W3_MEDIUMORCHID	This is macro GFX_W3_MEDIUMORCHID.
	GFX_W3_MEDIUMPURPLE	This is macro GFX_W3_MEDIUMPURPLE.

	GFX_W3_MEDIUMSEAGREEN	This is macro GFX_W3_MEDIUMSEAGREEN.
	GFX_W3_MEDIUMSLATEBLUE	This is macro GFX_W3_MEDIUMSLATEBLUE.
	GFX_W3_MEDIUMSPRINGGREEN	This is macro GFX_W3_MEDIUMSPRINGGREEN.
	GFX_W3_MEDIUMTURQUOISE	This is macro GFX_W3_MEDIUMTURQUOISE.
	GFX_W3_MEDIUMVIOLETRED	This is macro GFX_W3_MEDIUMVIOLETRED.
	GFX_W3_MIDNIGHTBLUE	This is macro GFX_W3_MIDNIGHTBLUE.
	GFX_W3_MINTCREAM	This is macro GFX_W3_MINTCREAM.
	GFX_W3_MISTYROSE	This is macro GFX_W3_MISTYROSE.
	GFX_W3_MOCCASIN	This is macro GFX_W3_MOCCASIN.
	GFX_W3_NAVAJOWHITE	This is macro GFX_W3_NAVAJOWHITE.
	GFX_W3_NAVY	This is macro GFX_W3_NAVY.
	GFX_W3_OLDLACE	This is macro GFX_W3_OLDLACE.
	GFX_W3_OLIVE	This is macro GFX_W3_OLIVE.
	GFX_W3_OLIVEDRAB	This is macro GFX_W3_OLIVEDRAB.
	GFX_W3_ORANGE	This is macro GFX_W3_ORANGE.
	GFX_W3_ORANGERED	This is macro GFX_W3_ORANGERED.
	GFX_W3_ORCHID	This is macro GFX_W3_ORCHID.
	GFX_W3_PALEGOLDENROD	This is macro GFX_W3_PALEGOLDENROD.
	GFX_W3_PALEGREEN	This is macro GFX_W3_PALEGREEN.
	GFX_W3_PALETURQUOISE	This is macro GFX_W3_PALETURQUOISE.
	GFX_W3_PALEVIOLETRED	This is macro GFX_W3_PALEVIOLETRED.
	GFX_W3_PAPAYAWHIP	This is macro GFX_W3_PAPAYAWHIP.
	GFX_W3_PEACHPUFF	This is macro GFX_W3_PEACHPUFF.
	GFX_W3_PERU	This is macro GFX_W3_PERU.
	GFX_W3_PINK	This is macro GFX_W3_PINK.
	GFX_W3_PLUM	This is macro GFX_W3_PLUM.
	GFX_W3_POWDERBLUE	This is macro GFX_W3_POWDERBLUE.
	GFX_W3_PURPLE	This is macro GFX_W3_PURPLE.
	GFX_W3_RED	This is macro GFX_W3_RED.
	GFX_W3_ROSYBROWN	This is macro GFX_W3_ROSYBROWN.
	GFX_W3_ROYALBLUE	This is macro GFX_W3_ROYALBLUE.
	GFX_W3_SADDLEBROWN	This is macro GFX_W3_SADDLEBROWN.
	GFX_W3_SALMON	This is macro GFX_W3_SALMON.
	GFX_W3_SANDYGREEN	This is macro GFX_W3_SANDYGREEN.
	GFX_W3_SEAGREEN	This is macro GFX_W3_SEAGREEN.
	GFX_W3_SEASHELL	This is macro GFX_W3_SEASHELL.
	GFX_W3_SIENNA	This is macro GFX_W3_SIENNA.
	GFX_W3_SILVER	This is macro GFX_W3_SILVER.
	GFX_W3_SKYBLUE	This is macro GFX_W3_SKYBLUE.
	GFX_W3_SLATEBLUE	This is macro GFX_W3_SLATEBLUE.
	GFX_W3_SLATEGRAY	This is macro GFX_W3_SLATEGRAY.
	GFX_W3_SLATEGREY	This is macro GFX_W3_SLATEGREY.
	GFX_W3_SNOW	This is macro GFX_W3_SNOW.
	GFX_W3_SPRINGGREEN	This is macro GFX_W3_SPRINGGREEN.
	GFX_W3_STEELBLUE	This is macro GFX_W3_STEELBLUE.
	GFX_W3_TAN	This is macro GFX_W3_TAN.
	GFX_W3_TEAL	This is macro GFX_W3_TEAL.
	GFX_W3_THISTLE	This is macro GFX_W3_THISTLE.
	GFX_W3_TOMATO	This is macro GFX_W3_TOMATO.
	GFX_W3_TURQUOISE	This is macro GFX_W3_TURQUOISE.
	GFX_W3_VIOLET	This is macro GFX_W3_VIOLET.
	GFX_W3_WHEAT	This is macro GFX_W3_WHEAT.
	GFX_W3_WHITE	This is macro GFX_W3_WHITE.
	GFX_W3_WHITESMOKE	This is macro GFX_W3_WHITESMOKE.
	GFX_W3_YELLOW	This is macro GFX_W3_YELLOW.

	GFX_W3_YELLOWGREEN	This is macro GFX_W3_YELLOWGREEN.
	KHAKI	This is macro KHAKI.

Description

Module for Microchip Graphics Library

This header file provides W3 color definitions as an option of color definitions to use.

Remarks

http://www.w3.org/TR/SVG/types.html#ColorKeywords

File Name

gfx_colors_w3.h

Company

Microchip Technology Inc.

gfx_colors_x11.h

The header file contains X11 color definitions.

Macros

	Name	Description
	GFX_X11_ALICE_BLUE	This is macro GFX_X11_ALICE_BLUE.
	GFX_X11_ANTIQUE_WHITE	This is macro GFX_X11_ANTIQUE_WHITE.
	GFX_X11_AQUA	This is macro GFX_X11_AQUA.
	GFX_X11_AQUAMARINE	This is macro GFX_X11_AQUAMARINE.
	GFX_X11_AZURE	This is macro GFX_X11_AZURE.
	GFX_X11_BEIGE	This is macro GFX_X11_BEIGE.
	GFX_X11_BISQUE	This is macro GFX_X11_BISQUE.
	GFX_X11_BLACK	This is macro GFX_X11_BLACK.
	GFX_X11_BLANCHED_ALMOND	This is macro GFX_X11_BLANCHED_ALMOND.
	GFX_X11_BLUE	This is macro GFX_X11_BLUE.
	GFX_X11_BLUE_VIOLET	This is macro GFX_X11_BLUE_VIOLET.
	GFX_X11_BROWN	This is macro GFX_X11_BROWN.
	GFX_X11_BURLY_WOOD	This is macro GFX_X11_BURLY_WOOD.
	GFX_X11_CADEL_BLUE	This is macro GFX_X11_CADEL_BLUE.
	GFX_X11_CHARTEUSE	This is macro GFX_X11_CHARTEUSE.
	GFX_X11_CHOCOLATE	This is macro GFX_X11_CHOCOLATE.
	GFX_X11_CORAL	This is macro GFX_X11_CORAL.
	GFX_X11_CORNFLOWER_BLUE	This is macro GFX_X11_CORNFLOWER_BLUE.
	GFX_X11_CORNSILK	X11: Brown Colors
	GFX_X11_CRIMSON	This is macro GFX_X11_CRIMSON.
	GFX_X11_CYAN	This is macro GFX_X11_CYAN.
	GFX_X11_DARK_BLUE	This is macro GFX_X11_DARK_BLUE.
	GFX_X11_DARK_CYAN	This is macro GFX_X11_DARK_CYAN.
	GFX_X11_DARK_GOLDENROD	This is macro GFX_X11_DARK_GOLDENROD.
	GFX_X11_DARK_GREEN	This is macro GFX_X11_DARK_GREEN.
	GFX_X11_DARK_GREY	This is macro GFX_X11_DARK_GREY.
	GFX_X11_DARK_KHAKI	This is macro GFX_X11_DARK_KHAKI.
	GFX_X11_DARK_MAGENTA	This is macro GFX_X11_DARK_MAGENTA.
	GFX_X11_DARK_OLIVE_GREEN	X11: Green Colors
	GFX_X11_DARK_ORANGE	This is macro GFX_X11_DARK_ORANGE.
	GFX_X11_DARK_ORCHID	This is macro GFX_X11_DARK_ORCHID.
	GFX_X11_DARK_RED	This is macro GFX_X11_DARK_RED.
	GFX_X11_DARK_SALMON	This is macro GFX_X11_DARK_SALMON.
	GFX_X11_DARK_SEA_GREEN	This is macro GFX_X11_DARK_SEA_GREEN.
	GFX_X11_DARK_SLATE_BLUE	This is macro GFX_X11_DARK_SLATE_BLUE.

	GFX_X11_DARK_SLATE_GREY	This is macro GFX_X11_DARK_SLATE_GREY.
	GFX_X11_DARK_TURQUOISE	This is macro GFX_X11_DARK_TURQUOISE.
	GFX_X11_DARK_VIOLET	This is macro GFX_X11_DARK_VIOLET.
	GFX_X11_DEEP_PINK	This is macro GFX_X11_DEEP_PINK.
	GFX_X11_DEEP_SKY_BLUE	This is macro GFX_X11_DEEP_SKY_BLUE.
	GFX_X11_DIM_GREY	This is macro GFX_X11_DIM_GREY.
	GFX_X11_DODGER_BLUE	This is macro GFX_X11_DODGER_BLUE.
	GFX_X11_FIRE_BRICK	This is macro GFX_X11_FIRE_BRICK.
	GFX_X11_FLORAL_WHITE	This is macro GFX_X11_FLORAL_WHITE.
	GFX_X11_FOREST_GREEN	This is macro GFX_X11_FOREST_GREEN.
	GFX_X11_FUCHSIA	This is macro GFX_X11_FUCHSIA.
	GFX_X11_GAINSBORO	This is macro GFX_X11_GAINSBORO.
	GFX_X11_GHOST_WHITE	This is macro GFX_X11_GHOST_WHITE.
	GFX_X11_GOLD	This is macro GFX_X11_GOLD.
	GFX_X11_GOLDENROD	This is macro GFX_X11_GOLDENROD.
	GFX_X11_GREEN	This is macro GFX_X11_GREEN.
	GFX_X11_GREEN_YELLOW	This is macro GFX_X11_GREEN_YELLOW.
	GFX_X11_GREY	This is macro GFX_X11_GREY.
	GFX_X11_HONEYDEW	This is macro GFX_X11_HONEYDEW.
	GFX_X11_HOT_PINK	This is macro GFX_X11_HOT_PINK.
	GFX_X11_INDIAN_RED	This is macro GFX_X11_INDIAN_RED.
	GFX_X11_INDIGO	This is macro GFX_X11_INDIGO.
	GFX_X11_IVORY	This is macro GFX_X11_IVORY.
	GFX_X11_KHAKI	This is macro GFX_X11_KHAKI.
	GFX_X11_LAVENDER	X11: BLUE COLORS
	GFX_X11_LAVENDOR_BLUSH	This is macro GFX_X11_LAVENDOR_BLUSH.
	GFX_X11_LAWN_GREEN	This is macro GFX_X11_LAWN_GREEN.
	GFX_X11_LEMON_CHIFFON	This is macro GFX_X11_LEMON_CHIFFON.
	GFX_X11_LIGHT_BLUE	This is macro GFX_X11_LIGHT_BLUE.
	GFX_X11_LIGHT_CAROL	This is macro GFX_X11_LIGHT_CAROL.
	GFX_X11_LIGHT_CYAN	This is macro GFX_X11_LIGHT_CYAN.
	GFX_X11_LIGHT_GOLDENROD_YELLOW	This is macro GFX_X11_LIGHT_GOLDENROD_YELLOW.
	GFX_X11_LIGHT_GRAY	This is macro GFX_X11_LIGHT_GRAY.
	GFX_X11_LIGHT_GREEN	This is macro GFX_X11_LIGHT_GREEN.
	GFX_X11_LIGHT_PINK	This is macro GFX_X11_LIGHT_PINK.
	GFX_X11_LIGHT_SALMON	X11: Red Colors
	GFX_X11_LIGHT_SEA_GREEN	This is macro GFX_X11_LIGHT_SEA_GREEN.
	GFX_X11_LIGHT_SKY_BLUE	This is macro GFX_X11_LIGHT_SKY_BLUE.
	GFX_X11_LIGHT_SLATE_GREY	This is macro GFX_X11_LIGHT_SLATE_GREY.
	GFX_X11_LIGHT_STEEL_BLUE	X11: BLUE COLORS
	GFX_X11_LIGHT_YELLOW	This is macro GFX_X11_LIGHT_YELLOW.
	GFX_X11_LIME	This is macro GFX_X11_LIME.
	GFX_X11_LIME_GREEN	This is macro GFX_X11_LIME_GREEN.
	GFX_X11_LINEN	This is macro GFX_X11_LINEN.
	GFX_X11_MAGENTA	This is macro GFX_X11_MAGENTA.
	GFX_X11_MARRON	This is macro GFX_X11_MARRON.
	GFX_X11_MEDIUM_AQUAMARINE	X11: CYAN COLORS
	GFX_X11_MEDIUM_BLUE	This is macro GFX_X11_MEDIUM_BLUE.
	GFX_X11_MEDIUM_ORCHID	This is macro GFX_X11_MEDIUM_ORCHID.
	GFX_X11_MEDIUM_PURPLE	This is macro GFX_X11_MEDIUM_PURPLE.
	GFX_X11_MEDIUM_SEA_GREEN	This is macro GFX_X11_MEDIUM_SEA_GREEN.
	GFX_X11_MEDIUM_SLATE_BLUE	This is macro GFX_X11_MEDIUM_SLATE_BLUE.
	GFX_X11_MEDIUM_SPRING_GREEN	This is macro GFX_X11_MEDIUM_SPRING_GREEN.
	GFX_X11_MEDIUM_TURQUOISE	This is macro GFX_X11_MEDIUM_TURQUOISE.
	GFX_X11_MEDIUM_VIOLET_RED	This is macro GFX_X11_MEDIUM_VIOLET_RED.

	GFX_X11_MIDNIGHT_BLUE	This is macro GFX_X11_MIDNIGHT_BLUE.
	GFX_X11_MINT_CREAM	This is macro GFX_X11_MINT_CREAM.
	GFX_X11_MISTY_ROSE	This is macro GFX_X11_MISTY_ROSE.
	GFX_X11_MOCCASIN	This is macro GFX_X11_MOCCASIN.
	GFX_X11_NAVAJO_WHITE	This is macro GFX_X11_NAVAJO_WHITE.
	GFX_X11_NAVY	This is macro GFX_X11_NAVY.
	GFX_X11_OLD_LACE	This is macro GFX_X11_OLD_LACE.
	GFX_X11_OLIVE	This is macro GFX_X11_OLIVE.
	GFX_X11_OLIVE_DRAB	This is macro GFX_X11_OLIVE_DRAB.
	GFX_X11_ORANGE	This is macro GFX_X11_ORANGE.
	GFX_X11_ORANGE_RED	X11: Orange Colors
	GFX_X11_ORCHID	This is macro GFX_X11_ORCHID.
	GFX_X11_PALE_GOLDENROD	This is macro GFX_X11_PALE_GOLDENROD.
	GFX_X11_PALE_GREEN	This is macro GFX_X11_PALE_GREEN.
	GFX_X11_PALE_TURQUOISE	This is macro GFX_X11_PALE_TURQUOISE.
	GFX_X11_PALE_VIOLET_RED	This is macro GFX_X11_PALE_VIOLET_RED.
	GFX_X11_PAPAYA_WHIP	This is macro GFX_X11_PAPAYA_WHIP.
	GFX_X11_PEACH_PUFF	This is macro GFX_X11_PEACH_PUFF.
	GFX_X11_PERU	This is macro GFX_X11_PERU.
	GFX_X11_PINK	X11: Pink Colors
	GFX_X11_PLUM	This is macro GFX_X11_PLUM.
	GFX_X11_POWDER_BLUE	This is macro GFX_X11_POWDER_BLUE.
	GFX_X11_PURPLE	This is macro GFX_X11_PURPLE.
	GFX_X11_RED	This is macro GFX_X11_RED.
	GFX_X11_ROSY_BROWN	This is macro GFX_X11_ROSY_BROWN.
	GFX_X11_ROYAL_BLUE	This is macro GFX_X11_ROYAL_BLUE.
	GFX_X11_SADDLE_BROWN	This is macro GFX_X11_SADDLE_BROWN.
	GFX_X11_SALMON	This is macro GFX_X11_SALMON.
	GFX_X11_SANDY_BROWN	This is macro GFX_X11_SANDY_BROWN.
	GFX_X11_SEA_GREEN	This is macro GFX_X11_SEA_GREEN.
	GFX_X11_SEASHELL	This is macro GFX_X11_SEASHELL.
	GFX_X11_SIENNA	This is macro GFX_X11_SIENNA.
	GFX_X11_SILVER	This is macro GFX_X11_SILVER.
	GFX_X11_SKY_BLUE	This is macro GFX_X11_SKY_BLUE.
	GFX_X11_SLATE_BLUE	This is macro GFX_X11_SLATE_BLUE.
	GFX_X11_SLATE_GREY	This is macro GFX_X11_SLATE_GREY.
	GFX_X11_SNOW	This is macro GFX_X11_SNOW.
	GFX_X11_SPRING_GREEN	This is macro GFX_X11_SPRING_GREEN.
	GFX_X11_STEEL_BLUE	This is macro GFX_X11_STEEL_BLUE.
	GFX_X11_TAN	This is macro GFX_X11_TAN.
	GFX_X11_TEAL	This is macro GFX_X11_TEAL.
	GFX_X11_THISTLE	This is macro GFX_X11_THISTLE.
	GFX_X11_TOMATO	This is macro GFX_X11_TOMATO.
	GFX_X11_TURQUOISE	This is macro GFX_X11_TURQUOISE.
	GFX_X11_VIOLET	This is macro GFX_X11_VIOLET.
	GFX_X11_WHEAT	This is macro GFX_X11_WHEAT.
	GFX_X11_WHITE	X11: WHITE/GRAY/BLACK COLORS
	GFX_X11_WHITE_SMOKE	This is macro GFX_X11_WHITE_SMOKE.
	GFX_X11_YELLOW	X11: Yellow Colors
	GFX_X11_YELLOW_GREEN	This is macro GFX_X11_YELLOW_GREEN.

Description

Module for Microchip Graphics Library

This header file provides X11 color definitions as an option of color definitions to use.

File Name

gfx_colors_x11.h

Company

Microchip Technology Inc.

gfx_gol.h

Graphics Object Layer of the Microchip Graphics Library.

Enumerations

	Name	Description
	GFX_GOL_CLIENT_STATUS	Enumerated data type that identifies the GFX Module Client Status.
	GFX_GOL_COMMON_STATE_BITS	Common Object States.
	GFX_GOL_OBJ_TYPE	Specifies the different object types used in the library.
	GFX_GOL_OBJECT_TASK	Lists the different states that GFX GOL task routine can have.
	GFX_GOL_STATES	Defines the various states that can be achieved by the GFX GOL operation.
	GFX_GOL_TRANSLATED_ACTION	Specifies the different object actions supported in the library.

Functions

	Name	Description
	GFX_GOL_Deinitialize	Deinitializes the specified instance of the GFX Module.
	GFX_GOL_DrawCallbackSet	This function sets the draw callback function that the application will use to render application specific rendering.
	GFX_GOL_Initialize	Initialize the GFX GOL Library.
	GFX_GOL_MessageCallbackSet	This function sets the message callback function that the application will use to evaluate user inputs that affects the objects and application behavior.
	GFX_GOL_ObjectAdd	This function adds an object to the tail of the currently active list.
	GFX_GOL_ObjectBackGroundSet	This function sets the background information.
	GFX_GOL_ObjectByIDDelete	This function removes an object with the given user defined ID from the currently active list.
	GFX_GOL_ObjectCanBeFocused	Checks if the object can be focused.
	GFX_GOL_ObjectDelete	This function removes an object from the currently active list.
	GFX_GOL_ObjectDrawDisable	This function resets the drawing state bits of the object.
	GFX_GOL_ObjectDrawEnable	This function sets the object to be redraw.
	GFX_GOL_ObjectFind	This function returns the pointer to object in the list with the user defined ID assigned to it.
	GFX_GOL_ObjectFocusGet	This function returns the pointer to the object that is currently receiving keyboard input (or focused).
	GFX_GOL_ObjectFocusNextGet	This function returns the pointer to the next object in the active list of objects which can be focused.
	GFX_GOL_ObjectFocusPrevGet	This function returns the pointer to the previous object in the active list of objects which can be focused.
	GFX_GOL_ObjectFocusSet	This function sets the object to be focused.
	GFX_GOL_ObjectHideDraw	This function performs the hiding of an object from the screen.
	GFX_GOL_ObjectIDGet	This function returns the object ID.
	GFX_GOL_ObjectIsRedrawSet	This function checks if the object needs to be redrawn or not.
	GFX_GOL_ObjectListDraw	This function redraws all objects in the current active list that has the rendering state bits set.
	GFX_GOL_ObjectListFree	This function sets the active list to the new list.
	GFX_GOL_ObjectListGet	This function returns the current active list.
	GFX_GOL_ObjectListHide	This function marks all objects in the active list to be hidden.
	GFX_GOL_ObjectListNew	This function removes an object with the given user defined ID from the currently active list.
	GFX_GOL_ObjectListSet	This function sets the active list to the new list.
	GFX_GOL_ObjectMessage	This function process the received message from the user to determine the affected objects. Depending on the message and the affected objects, object states are modified based on the default or user-defined behavior.
	GFX_GOL_ObjectNextGet	This function returns the pointer to next object in the list after the specified object.

	GFX_GOL_ObjectRectangleRedraw	This function marks all objects in the active list intersected by the given rectangular area to be redrawn.
	GFX_GOL_ObjectTypeGet	This function returns the object type.
	GFX_GOL_Open	opens an instance of the GFX GOL Library.
	GFX_GOL_PanelAlphaParameterSet	This function sets the alpha blending value when using alpha blending in panels.
	GFX_GOL_PanelBackgroundSet	This function sets panel background information.
	GFX_GOL_PanelDraw	This function renders the panel.
	GFX_GOL_PanelGradientParameterSet	This function sets the gradient fill start and end colors of a panel.
	GFX_GOL_PanelParameterSet	This function sets the parameters to draw a panel.
	GFX_GOL_PreemptionLevelGet	This function returns the current preemption level.
	GFX_GOL_Tasks	Maintains the GFX GOL module state machine. It manages the GFX Module object list items and responds to GFX Module primitive events.
	GFX_GOL_TwoTonePanelDraw	This function renders the two-tone panel.

Macros

	Name	Description
	GFX_GOL_ObjectStateClear	This function clears the state bits of the given object.
	GFX_GOL_ObjectStateGet	This function retrieves the current value of the state bits of an object.
	GFX_GOL_ObjectStateSet	This function sets the state bits of the given object.
	GFX_GOL_ObjectStyleSchemeGet	This function returns the style scheme currently set for the object.
	GFX_GOL_ObjectStyleSchemeSet	This function sets the style scheme of the object.

Structures

	Name	Description
	GFX_GOL_MESSAGE	Specifies message structure used in the library.
	GFX_GOL_OBJ_HEADER	Specifies Graphics Object Layer structure used in objects.
	GOL_PANEL_PARAM	Specifies panel parameters.

Types

	Name	Description
	GFX_GOL_DRAW_CALLBACK_FUNC	Draw callback function definition. This application defined function allows the application to perform application specific rendering.
	GFX_GOL_MESSAGE_CALLBACK_FUNC	Message callback function definition. This application defined function allows the application to perform application specific processing of user messages.

Description

Module for Microchip Graphics Library - Graphic Object Layer

This module implements the common routines for the Graphics Object Layer of the Microchip Graphics Library. The routines are independent of the Display Driver Layer and should be compatible with any Display Driver that is compliant with the requirements of the Display Driver Layer of the Graphics Library. The module utilizes the Graphics Primitive Layer to render the objects.

File Name

gfx_gol.h

Company

Microchip Technology Inc.

gfx_gol_button.h

This is the header file for the button object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_BUTTON_STATE	Specifies the different states of the Button object.

Functions

	Name	Description
	GFX_GOL_ButtonActionGet	This function evaluates the message from a user if the message will affect the object or not.

	GFX_GOL_ButtonActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ButtonCreate	This function creates a GFX_GOL_BUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ButtonDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ButtonTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_ButtonTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_ButtonTextSet	This function sets the address of the current text string used by the object.

Macros

	Name	Description
	GFX_GOL_ButtonPressStateImageGet	This function gets the image used when in the pressed state.
	GFX_GOL_ButtonPressStateImageSet	This function sets the image to be used when in the pressed state.
	GFX_GOL_ButtonReleaseStateImageGet	This function gets the image used when in the released state.
	GFX_GOL_ButtonReleaseStateImageSet	This function sets the image to be used when in the released state.
	GFX_GOL_ButtonTextGet	This function returns the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_BUTTON	Defines the structure used for the Button object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the GFX_GOL_BUTTON Object.

File Name

gfx_gol_button.h

Company

Microchip Technology Inc.

gfx_gol_check_box.h

This is the header file for the check box object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_CHECKBOX_STATE	Specifies the different states of the Check Box object.

Functions

	Name	Description
	GFX_GOL_CheckBoxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_CheckBoxActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_CheckBoxCreate	This function creates a GFX_GOL_CHECKBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_CheckBoxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_CheckBoxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_CheckBoxTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_CheckBoxTextSet	This function sets the address of the current text string used by the object.

Macros

	Name	Description
	GFX_GOL_CheckBoxTextGet	This function returns the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_CHECKBOX	Defines the structure used for the Check Box object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the CheckBox Object.

File Name

gfx_gol_check_box.h

Company

Microchip Technology Inc.

gfx_gol_custom_control.h

This is the header file for the custom control object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_CUSTOMCONTROL_STATE	Specifies the different states of the Custom Control object.

Functions

	Name	Description
	GFX_GOL_CustomControlActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_CustomControlActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_CustomControlCreate	This function creates a GFX_GOL_CUSTOMCONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_CustomControlDraw	This function renders the object on the screen based on the current state of the object.

Macros

	Name	Description
	GFX_GOL_CustomControlGetPos	This function returns the position of the control.
	GFX_GOL_CustomControlSetPos	This function sets the position of the control.

Structures

	Name	Description
	GFX_GOL_CUSTOMCONTROL	Defines the structure used for the Custom Control object.

Description

Graphics Library Implementation

Refer to Microchip Graphics Library for complete documentation of the CustomControl Object.

File Name

gfx_gol_custom_control.c

Company

Microchip Technology Inc.

gfx_gol_digital_meter.h

This is the header file for the digital meter object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_DIGITALMETER_STATE	Specifies the different states of the Digital Meter object.

Functions

	Name	Description
	GFX_GOL_DigitalMeterActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_DigitalMeterCreate	This function creates a GFX_GOL_DIGITALMETER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_DigitalMeterDecrement	This function decrements the meter value by the delta value set.
	GFX_GOL_DigitalMeterDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_DigitalMeterIncrement	This function increments the meter value by the delta value set.
	GFX_GOL_DigitalMeterValueSet	This function sets the value of the meter.

Macros

	Name	Description
	GFX_GOL_DigitalMeterTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_DigitalMeterTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_DigitalMeterValueGet	This function returns the current value of the digital meter.

Structures

	Name	Description
	GFX_GOL_DIGITALMETER	Defines the structure used for the Digital Meter object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Digital Meter Object.

File Name

gfx_gol_digital_meter.h

Company

Microchip Technology Inc.

gfx_gol_edit_box.h

This is the header file for the edit box object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_EDITBOX_STATE	Specifies the different states of the Edit Box object.

Functions

	Name	Description
	GFX_GOL_EditBoxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_EditBoxActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_EditBoxCharAdd	This function adds a character at the end of the text used by the object.
	GFX_GOL_EditBoxCharRemove	This function removes a character at the end of the text used by the object.
	GFX_GOL_EditBoxCreate	This function creates a GFX_GOL_EDITBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_EditBoxCursorPositionDecrement	Decrement cursor position.
	GFX_GOL_EditBoxCursorPositionIncrement	Increment cursor position.
	GFX_GOL_EditBoxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_EditBoxTextSet	This function sets the text in the object.

Macros

	Name	Description
	GFX_GOL_EditBoxTextAlignmentGet	This function returns the text alignment of the text string used by the object.

	GFX_GOL_EditBoxTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_EditBoxTextGet	This function returns the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_EDITBOX	Defines the structure used for the Edit Box object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Edit Box Object.

File Name

gfx_gol_edit_box.h

Company

Microchip Technology Inc.

gfx_gol_group_box.h

This is the header file for the group box object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_GROUPBOX_STATE	Specifies the different states of the Group Box object.

Functions

	Name	Description
	GFX_GOL_GroupboxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_GroupboxCreate	This function creates a GFX_GOL_GROUPBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_GroupboxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_GroupboxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_GroupboxTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_GroupboxTextSet	This function sets the address of the current text string used by the object.

Macros

	Name	Description
	GFX_GOL_GroupboxTextGet	This function returns the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_GROUPBOX	Defines the structure used for the Group Box object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Group Box Object.

File Name

gfx_gol_group_box.h

Company

Microchip Technology Inc.

gfx_gol_list_box.h

This is the header file for the list box object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_LISTBOX_ITEM_STATUS	Defines the types used to indicate the status of an item.
	GFX_GOL_LISTBOX_STATE	Specifies the different states of the List Box object.

Functions

	Name	Description
	GFX_GOL_ListBoxActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ListBoxActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ListBoxCreate	This function creates a GFX_GOL_LISTBOX object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ListBoxDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ListBoxItemAdd	This function adds an item to the list box.
	GFX_GOL_ListBoxItemFocusGet	This function returns the index of the focused item in the list box.
	GFX_GOL_ListBoxItemFocusSet	This function sets the focus of the item with the index number specified by index.
	GFX_GOL_ListBoxItemListRemove	This function removes the entire items list of the list box and free the memory used.
	GFX_GOL_ListBoxItemRemove	This function removes an item from the list box and free the memory used.
	GFX_GOL_ListBoxSelectionChange	This function changes the selection status of the given item in the list box.
	GFX_GOL_ListBoxSelectionGet	This function searches for selected items from the list box.
	GFX_GOL_ListBoxVisibleItemCountGet	This function returns the count of items visible in the list box.

Macros

	Name	Description
	GFX_GOL_ListBoxItemCountGet	This function returns the count of items in the list box.
	GFX_GOL_ListBoxItemImageGet	This function gets the image set for a list box item.
	GFX_GOL_ListBoxItemImageSet	This function sets the image for a list box item.
	GFX_GOL_ListBoxItemListGet	This function returns the pointer to the item list of the list box.
	GFX_GOL_ListBoxItemSelectStatusClear	This function clears the selection status of the item.
	GFX_GOL_ListBoxItemSelectStatusSet	This function sets the selection status of the item to selected.
	GFX_GOL_ListBoxTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_ListBoxTextAlignmentSet	This function sets the text alignment of the text string used by the object.

Structures

	Name	Description
	GFX_GOL_LISTBOX	Defines the structure used for the List Box object.
	GFX_GOL_LISTITEM	The structure that defines each item in the list box.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the ListBox Object.

File Name

gfx_gol_list_box.h

Company

Microchip Technology Inc.

gfx_gol_meter.h

This is the header file for the meter object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_METER_DRAW_TYPE	Specifies the different types of Meter object.
	GFX_GOL_METER_STATE	Specifies the different states of the Meter object.

Functions

	Name	Description
	GFX_GOL_MeterActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_MeterActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_MeterCreate	This function creates a GFX_GOL_METER object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_MeterDecrement	This function decrements the meter value by the delta value set.
	GFX_GOL_MeterDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_MeterIncrement	This function increments the meter value by the delta value set.
	GFX_GOL_MeterRangeSet	This function sets the range of the meter.
	GFX_GOL_MeterScaleColorsSet	This function sets the arc colors of the object.
	GFX_GOL_MeterValueSet	This function sets the value of the meter.

Macros

	Name	Description
	GFX_GOL_MeterMaximumValueGet	This function returns the maximum value set for the meter.
	GFX_GOL_MeterMinimumValueGet	This function returns the minimum value set for the meter.
	GFX_GOL_MeterTitleFontSet	This function sets the meter title font used.
	GFX_GOL_MeterTypeSet	This function sets the meter draw type.
	GFX_GOL_MeterValueFontSet	This function sets the meter's displayed value font used.
	GFX_GOL_MeterValueGet	This function returns the current value of the meter.

Structures

	Name	Description
	GFX_GOL_METER	Defines the structure used for the Meter object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Meter Object.

File Name

gfx_gol_meter.h

Company

Microchip Technology Inc.

gfx_gol_picture.h

This is the header file for the picture object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_PICTURECONTROLCONTROL_STATE	Specifies the different states of the Picture Control object.

Functions

	Name	Description
	GFX_GOL_PictureControlActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_PictureControlCreate	This function creates a GFX_GOL_PICTURECONTROL object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_PictureControlDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_PictureControlPartialSet	This function sets the partial image parameters to be in the object.
	GFX_GOL_PictureControlScaleSet	Sets the scale factor used to render the image used in the object.

Macros

	Name	Description
	GFX_GOL_PictureControlImageGet	This function gets the image used when in the pressed state.
	GFX_GOL_PictureControlImageSet	This function sets the image to be in the object.

Structures

	Name	Description
	GFX_GOL_PICTURECONTROL	Defines the structure used for the Picture Control object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Picture Object.

File Name

gfx_gol_picture.h

Company

Microchip Technology Inc.

gfx_gol_progress_bar.h

This is the header file for the progress bar object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_PROGRESSBAR_STATE	Specifies the different states of the Progress Bar object.

Functions

	Name	Description
	GFX_GOL_ProgressBarActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ProgressBarCreate	This function creates a GFX_GOL_PROGRESSBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ProgressBarDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ProgressBarPositionSet	This function sets the position of the progress bar.
	GFX_GOL_ProgressBarRangeSet	This function sets the range of the progress bar.

Macros

	Name	Description
	GFX_GOL_ProgressBarPositionGet	This function returns the current position of the progress bar.
	GFX_GOL_ProgressBarRangeGet	This function returns the range of the progress bar.

Structures

	Name	Description
	GFX_GOL_PROGRESSBAR	Defines the structure used for the Progress Bar object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the ProgressBar Object.

File Name

gfx_gol_progress_bar.h

Company

Microchip Technology Inc.

gfx_gol_radio_button.h

This is the header file for the radio button object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_RADIOBUTTON_STATE	Specifies the different states of the Radio Button object.

Functions

	Name	Description
	GFX_GOL_RadioButtonActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_RadioButtonActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_RadioButtonCheckGet	This function returns the ID of the currently checked radio button in the group.
	GFX_GOL_RadioButtonCheckSet	This function sets the ID of the currently checked radio button in the group.
	GFX_GOL_RadioButtonCreate	This function creates a GFX_GOL_RADIOBUTTON object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_RadioButtonDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_RadioButtonListCreate	This is function GFX_GOL_RadioButtonListCreate.
	GFX_GOL_RadioButtonTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_RadioButtonTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_RadioButtonTextSet	This function sets the address of the current text string used by the object.

Macros

	Name	Description
	GFX_GOL_RadioButtonTextGet	This function returns the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_RADIOBUTTON	Defines the structure used for the Radio Button object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the RadioButton Object.

File Name

gfx_gol_radio_button.h

Company

Microchip Technology Inc.

gfx_gol_scan_codes.h

This is the header file for the AT keyboard codes.

Enumerations

	Name	Description
	GFX_AT_SCAN_CODES	The following macros are the supported AT Keyboard scan codes.

Description

Module for Microchip Graphics Library - Graphic Object Layer

This header file contains scan codes for the standard AT keyboard that can be used to process user action in GOL messages.

File Name

gfx_gol_scan_codes.h

Company

Microchip Technology Inc.

gfx_gol_scheme.h

Graphics Object Layer Scheme routines/macros of the Microchip Graphics Library.

Functions

	Name	Description
	GFX_GOL_SchemeAlphaPrecentSet	This is function GFX_GOL_SchemeAlphaPrecentSet.
	GFX_GOL_SchemeBackgroundColorGet	This is function GFX_GOL_SchemeBackgroundColorGet.
	GFX_GOL_SchemeBackgroundColorSet	This is function GFX_GOL_SchemeBackgroundColorSet.
	GFX_GOL_SchemeBackgroundImageSet	This is function GFX_GOL_SchemeBackgroundImageSet.
	GFX_GOL_SchemeBackgroundTypeGet	This is function GFX_GOL_SchemeBackgroundTypeGet.
	GFX_GOL_SchemeBackgroundTypeSet	This is function GFX_GOL_SchemeBackgroundTypeSet.
	GFX_GOL_SchemeColor0Get	This is function GFX_GOL_SchemeColor0Get.
	GFX_GOL_SchemeColor0Set	This is function GFX_GOL_SchemeColor0Set.
	GFX_GOL_SchemeColor1Get	This is function GFX_GOL_SchemeColor1Get.
	GFX_GOL_SchemeColor1Set	This is function GFX_GOL_SchemeColor1Set.
	GFX_GOL_SchemeColorDisabledGet	This is function GFX_GOL_SchemeColorDisabledGet.
	GFX_GOL_SchemeColorDisabledSet	This is function GFX_GOL_SchemeColorDisabledSet.
	GFX_GOL_SchemeColorSet	This is function GFX_GOL_SchemeColorSet.
	GFX_GOL_SchemeEmbossDarkColorGet	This is function GFX_GOL_SchemeEmbossDarkColorGet.
	GFX_GOL_SchemeEmbossDarkColorSet	This is function GFX_GOL_SchemeEmbossDarkColorSet.
	GFX_GOL_SchemeEmbossLightColorGet	This is function GFX_GOL_SchemeEmbossLightColorGet.
	GFX_GOL_SchemeEmbossLightColorSet	This is function GFX_GOL_SchemeEmbossLightColorSet.
	GFX_GOL_SchemeEmbossSet	This is function GFX_GOL_SchemeEmbossSet.
	GFX_GOL_SchemeEmbossSizeGet	This is function GFX_GOL_SchemeEmbossSizeGet.
	GFX_GOL_SchemeEmbossSizeSet	This is function GFX_GOL_SchemeEmbossSizeSet.
	GFX_GOL_SchemeFillStyleGet	This is function GFX_GOL_SchemeFillStyleGet.
	GFX_GOL_SchemeFillStyleSet	This is function GFX_GOL_SchemeFillStyleSet.
	GFX_GOL_SchemeFontGet	This is function GFX_GOL_SchemeFontGet.
	GFX_GOL_SchemeFontSet	This is function GFX_GOL_SchemeFontSet.
	GFX_GOL_SchemeGradientColorSet	This is function GFX_GOL_SchemeGradientColorSet.
	GFX_GOL_SchemeGradientEndColorGet	This is function GFX_GOL_SchemeGradientEndColorGet.
	GFX_GOL_SchemeGradientEndColorSet	This is function GFX_GOL_SchemeGradientEndColorSet.
	GFX_GOL_SchemeGradientStartColorGet	This is function GFX_GOL_SchemeGradientStartColorGet.
	GFX_GOL_SchemeGradientStartColorSet	This is function GFX_GOL_SchemeGradientStartColorSet.
	GFX_GOL_SchemeTextColor0Get	This is function GFX_GOL_SchemeTextColor0Get.
	GFX_GOL_SchemeTextColor0Set	This is function GFX_GOL_SchemeTextColor0Set.
	GFX_GOL_SchemeTextColor1Get	This is function GFX_GOL_SchemeTextColor1Get.
	GFX_GOL_SchemeTextColor1Set	This is function GFX_GOL_SchemeTextColor1Set.
	GFX_GOL_SchemeTextColorDisableGet	This is function GFX_GOL_SchemeTextColorDisableGet.
	GFX_GOL_SchemeTextColorDisableSet	This is function GFX_GOL_SchemeTextColorDisableSet.
	GFX_GOL_SchemeTextColorSet	This is function GFX_GOL_SchemeTextColorSet.

Structures

	Name	Description
	GFX_GOL_OBJ_SCHEME	This structure specifies the style scheme components of an object.

Description

Module for Microchip Graphics Library - Graphic Object Layer Scheme

This module implements the common routines for the Graphics Object Layer of the Microchip Graphics Library. The routines are independent of the Display Driver Layer and should be compatible with any Display Driver that is compliant with the requirements of the Display Driver Layer of the Graphics Library. The module utilizes the Graphics Primitive Layer to render the objects.

File Name

gfx_gol_scheme.h

Company

Microchip Technology Inc.

gfx_gol_scroll_bar.h

This is the header file for the scroll bar object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_SCROLLBAR_STATE	Specifies the different states of the Scroll Bar object.

Functions

	Name	Description
	GFX_GOL_ScrollBarActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ScrollBarActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ScrollBarCreate	This function creates a GFX_GOL_SCROLLBAR object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_ScrollBarDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_ScrollBarPageGet	This function returns the page size of the object.
	GFX_GOL_ScrollBarPageSet	This function sets the page size of the object.
	GFX_GOL_ScrollBarPositionDecrement	This function decrements the scroll bar position by the delta change (page) value set.
	GFX_GOL_ScrollBarPositionGet	This function returns the current position of the scroll bar thumb.
	GFX_GOL_ScrollBarPositionIncrement	This function increments the scroll bar position by the delta change (page) value set.
	GFX_GOL_ScrollBarPositionSet	This function sets the position of the scroll bar thumb.
	GFX_GOL_ScrollBarRangeGet	This function returns the range of the thumb of the scroll bar.
	GFX_GOL_ScrollBarRangeSet	This function sets the range of the thumb of the scroll bar.

Structures

	Name	Description
	GFX_GOL_SCROLLBAR	Defines the structure used for the Scroll Bar object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Slider Object.

File Name

gfx_gol_scroll_bar.h

Company

Microchip Technology Inc.

gfx_gol_static_text.h

This is the header file for the static text object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_STATICTEXT_STATE	Specifies the different states of the Static Text object.

Functions

	Name	Description
	GFX_GOL_StaticTextActionGet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_StaticTextAlignmentGet	This function returns the text alignment of the text string used by the object.

	GFX_GOL_StaticTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_StaticTextCreate	This function creates a GFX_GOL_STATICTEXT object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_StaticTextDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_StaticTextGet	This function returns the address of the current text string used by the object.
	GFX_GOL_StaticTextSet	This function sets the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_STATICTEXT	Defines the structure used for the Static Text object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Static Text Object.

File Name

gfx_gol_static_text.h

Company

Microchip Technology Inc.

gfx_gol_surface.h

This is the header file for the surface object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_SURFACE_STATE	Specifies the different states of the Surface object.

Functions

	Name	Description
	GFX_GOL_SurfaceActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_SurfaceCreate	This function creates a GFX_GOL_SURFACE object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_SurfaceDraw	This function renders the object on the screen based on the current state of the object.

Macros

	Name	Description
	GFX_GOL_SurfaceImageGet	This function gets the image used.
	GFX_GOL_SurfaceImageSet	This function sets the image used in the object.

Structures

	Name	Description
	GFX_GOL_SURFACE	Defines the structure used for the Surface object.

Types

	Name	Description
	GFX_GOL_SURFACE_CALLBACK_FUNC	Surface callback function definition. This application defined function allows the application to perform application specific rendering.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Surface Object.

File Name

gfx_gol_surface.h

Company

Microchip Technology Inc.

gfx_gol_text_entry.h

This is the header file for the text entry object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE	Specifies the different commands available for command keys of the Text Entry object.
	GFX_GOL_TEXTENTRY_STATE	Specifies the different states of the Text Entry object.

Functions

	Name	Description
	GFX_GOL_TextEntryActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_TextEntryActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_TextEntryBufferClear	This function will clear the data in the display.
	GFX_GOL_TextEntryBufferGet	This function returns the buffer used to display text.
	GFX_GOL_TextEntryBufferSet	This function sets the buffer used to display text.
	GFX_GOL_TextEntryCharAdd	This function will insert a character to the end of the buffer.
	GFX_GOL_TextEntryCreate	This function creates a GFX_GOL_TEXTENTRY object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_TextEntryDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_TextEntryKeyCommandGet	This function will return the currently assigned command to the key with the given index.
	GFX_GOL_TextEntryKeyCommandSet	This function will assign a command to a key with the given index.
	GFX_GOL_TextEntryKeyIsPressed	This function will test if a key given by its index in the object is pressed.
	GFX_GOL_TextEntryKeyListCreate	This function will create the list of key members that holds the information on each key.
	GFX_GOL_TextEntryKeyMemberListDelete	This function will delete the key member list assigned to the object.
	GFX_GOL_TextEntryKeyTextSet	This function will set the text assigned to a key with the given index.
	GFX_GOL_TextEntryLastCharDelete	This function will remove the last character of the buffer and replace it with a string terminator.
	GFX_GOL_TextEntrySpaceCharAdd	This function will insert a space character to the end of the buffer.

Structures

	Name	Description
	GFX_GOL_TEXTENTRY	Defines the structure used for the Text Entry object.
	GFX_GOL_TEXTENTRY_KEYMEMBER	Defines the structure used to describe a key in the Text Entry object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Text Entry Object.

File Name

gfx_gol_text_entry.h

Company

Microchip Technology Inc.

gfx_gol_window.h

This is the header file for the window object of the GOL.

Enumerations

	Name	Description
	GFX_GOL_WINDOW_STATE	Specifies the different states of the Window object.

Functions

	Name	Description
	GFX_GOL_WindowActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_WindowCreate	This function creates a GFX_GOL_WINDOW object with the parameters given. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	GFX_GOL_WindowDraw	This function renders the object on the screen based on the current state of the object.
	GFX_GOL_WindowTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_WindowTextAlignmentSet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_WindowTextSet	This function sets the address of the current text string used by the object.

Macros

	Name	Description
	GFX_GOL_WindowImageGet	This function gets the image used.
	GFX_GOL_WindowImageSet	This function sets the image used in the object.
	GFX_GOL_WindowTextGet	This function returns the address of the current text string used by the object.

Structures

	Name	Description
	GFX_GOL_WINDOW	Defines the structure used for the Window object.

Description

Module for Microchip Graphics Library - Graphic Object Layer

Refer to Microchip Graphics Library for complete documentation of the Window Object.

File Name

gfx_gol_window.h

Company

Microchip Technology Inc.

gfx_image_decoder.h

This implements the image decoding of the primitive layer.

Enumerations

	Name	Description
	_IMG_FILE_FORMAT	IMG_ImageFileFormat specifies all the supported image file formats
	IMG_FILE_FORMAT	IMG_ImageFileFormat specifies all the supported image file formats

Functions

	Name	Description
	ImageDecode	This function decodes and displays the image on the screen
	ImageDecoderInit	This function initializes the global variables to 0 and then initializes the driver. This must be called once before any other function of the library is called
	ImageDecoderParceHeader	This is function ImageDecoderParceHeader.
	ImageDecodeTask	This function completes one small part of the image decode function
	ImageLoopCallbackRegister	This function registers the loop callback function so that the decoder calls this function in every decoding loop. This can be used by the application program to do maintenance activities such as fetching data, updating the display, etc...
	ImagePixelIntoArray	This is function ImagePixelIntoArray.
	ImagePixelOutput	This is function ImagePixelOutput.
	ImageRegisterFrameCompleteCallback	This is function ImageRegisterFrameCompleteCallback.

	ImageRegisterImageCompleteCallback	Function prototypes
	ImageRegisterImageFailCallback	This is function ImageRegisterImageFailCallback.
	IMG_vSetboundaries	

Macros

	Name	Description
	__IMAGEDECODER_H__	This is macro __IMAGEDECODER_H__.
	_PutPixel	This is macro _PutPixel.
	ImageAbort	This function sets the Image Decoder's Abort flag so that decoding aborts in the next decoding loop.
	ImageFullScreenDecode	This function decodes and displays the image on the screen in fullscreen mode with center aligned and downscaled if required
	IMG_ALIGN_CENTER	Flags
	IMG_DECODE_ABORTED	This is macro IMG_DECODE_ABORTED.
	IMG_DOWN_SCALE	This is macro IMG_DOWN_SCALE.
	IMG_FEOF	This is macro IMG_FEOF.
	IMG_FILE	This is macro IMG_FILE.
	IMG_FREAD	This is macro IMG_FREAD.
	IMG_FSEEK	This is macro IMG_FSEEK.
	IMG_FTELL	This is macro IMG_FTELL.
	IMG_SCREEN_HEIGHT	This is macro IMG_SCREEN_HEIGHT.
	IMG_SCREEN_WIDTH	The individual image decoders use these defines instead of directly using those provided in the Display driver file
	IMG_vCheckAndAbort	This is macro IMG_vCheckAndAbort.
	IMG_vFrameComplete	This is macro IMG_vFrameComplete.
	IMG_vImageComplete	This is macro IMG_vImageComplete.
	IMG_vImageFail	This is macro IMG_vImageFail.
	IMG_vLoopCallback	This is macro IMG_vLoopCallback.
	IMG_vPixelArray	This is macro IMG_vPixelArray.
	IMG_vPutPixel	This is macro IMG_vPutPixel.
	IMG_vSetColor	This is macro IMG_vSetColor.

Structures

	Name	Description
	_IMG_FILE_SYSTEM_API	IMG_FileSystemAPI holds function pointers to the used File System APIs
	_IMG_PIXEL_XY_RGB_888	IMG_PixelRgb holds the RGB information of a pixel in BYTES
	IMG_FILE_SYSTEM_API	IMG_FileSystemAPI holds function pointers to the used File System APIs
	IMG_PIXEL_XY_RGB_888	IMG_PixelRgb holds the RGB information of a pixel in BYTES

Types

	Name	Description
	FileFeof	This is type FileFeof.
	FileRead	Typedefs of the used File System APIs
	FileSeek	This is type FileSeek.
	FileTell	This is type FileTell.
	IMG_LOOP_CALLBACK	IMG_LoopCallback is a callback function which is called in every loop of the decoding cycle so that user application can do some maintenance activities
	IMG_LOOP_FRAMECOMPLETE	IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities
	IMG_LOOP_IMAGECOMPLETE	IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities
	IMG_LOOP_IMAGEFAIL	IMG_FrameCompleteCallback is a callback function which is called at completion decoding current frame so that user application can do some activities
	IMG_PIXEL_OUTPUT	IMG_PixelOutput is a callback function which receives the color information of the output pixel

Description

Graphics Image Decoder

Refer to Microchip Graphics Library for complete documentation of the Image Decoder.

File Name

gfx_image_decoder.h

Company

Microchip Technology Inc.

gfx_primitive.h

Primitive Layer of the Microchip Graphics Library.

Enumerations

	Name	Description
	GFX_PRIMITIVE_STATES	Defines the various states that can be achieved by the PRIMITIVE HAL operation.

Functions

	Name	Description
	GFX_AlphaBlendingValueGet	This function returns the current alpha value set for alpha blending rendering.
	GFX_AlphaBlendingValueSet	This function sets the alpha value for alpha blending rendering.
	GFX_BackgroundColorGet	This function returns the color used in the current background.
	GFX_BackgroundImageGet	This function returns the image used in the current background.
	GFX_BackgroundImageLeftGet	This function returns the horizontal starting position of the current background.
	GFX_BackgroundImageTopGet	This function returns the vertical starting position of the current background.
	GFX_BackgroundSet	This function sets the background information.
	GFX_BackgroundTypeGet	This function returns the type of the current background.
	GFX_BackgroundTypeSet	This function sets the background type.
	GFX_BarDraw	This function renders a bar shape using the currently set fill style and color.
	GFX_CircleDraw	This function renders a circular shape using the currently set line style and color.
	GFX_CircleFillDraw	This function renders a filled circle shape using the currently set fill style and colors.
	GFX_ColorGet	This function returns the color currently set to render primitive shapes and text.
	GFX_ColorSet	This function sets the color to be used in rendering primitive shapes and text.
	GFX_DoubleBufferAreaGet	This function returns a rectangular area that needs synchronization specified by the given index.
	GFX_DoubleBufferAreaMark	This function adds the given rectangular area as an area that will be included in the list of areas for synchronization.
	GFX_DoubleBufferDisable	This function disables the double buffering feature of the graphics library.
	GFX_DoubleBufferEnable	This function enables the double buffering feature of the graphics library.
	GFX_DoubleBufferPause	This function pauses the double buffering feature of the graphics library.
	GFX_DoubleBufferResume	This function resumes the double buffering feature of the graphics library.
	GFX_DoubleBufferStatusGet	This function returns the current status of the double buffering feature of the graphics library.
	GFX_DoubleBufferSyncAllStatusClear	This function clears the synchronize all status.
	GFX_DoubleBufferSyncAllStatusGet	This function returns the status of the synchronize all flag.
	GFX_DoubleBufferSyncAllStatusSet	This function sets the whole draw buffer to be unsynchronized.
	GFX_DoubleBufferSyncAreaCountGet	This function returns the current count of rectangular areas that needs to be synchronized.
	GFX_DoubleBufferSyncAreaCountSet	This function sets the current count of rectangular areas that needs to be synchronized.
	GFX_DoubleBufferSynchronize	This function synchronizes the contents of the draw and frame buffer immediately.
	GFX_DoubleBufferSynchronizeRequest	This function requests synchronization of the contents of the draw and frame buffer.
	GFX_DoubleBufferSynchronizeStatusGet	This function returns the status of the synchronization request of the draw and frame buffer.
	GFX_DrawBufferGet	This function returns the index of the current draw buffer.
	GFX_DrawBufferInitialize	This function initializes the address of the draw buffer specified by the given index.
	GFX_DrawBufferSet	This function sets the draw buffer with the given index number.
	GFX_ExternalResourceCallback	This function performs data fetch from external memory.
	GFX_FillStyleGet	Return the fill style used when rendering filled shapes.
	GFX_FillStyleSet	Set the fill style to be used when rendering filled shapes.

	GFX_FontAntiAliasGet	This function returns the current font font anti-aliasing mode.
	GFX_FontAntiAliasSet	This function sets the font anti-aliasing mode.
	GFX_FontGet	This function returns the current font used when rendering strings and characters.
	GFX_FontSet	This function sets the current font used when rendering strings and characters.
	GFX_FrameBufferGet	This function returns the index of the current frame buffer.
	GFX_FrameBufferSet	This function sets the frame buffer with the given index number.
	GFX_GradientColorSet	Sets the gradient fill start and end color.
	GFX_GradientEndColorGet	Return the gradient end color when rendering gradient filled shapes.
	GFX_GradientStartColorGet	Return the gradient start color when rendering gradient filled shapes.
	GFX_ImageDraw	This function renders an image to the frame buffer.
	GFX_ImageHeaderGet	This function fills the given bitmap header with the image's header information.
	GFX_ImageHeightGet	This function returns the height of the given image.
	GFX_ImagePartialDraw	This function renders a portion of an image to the frame buffer.
	GFX_ImageWidthGet	This function returns the width of the given image.
	GFX_LineDraw	This function renders a line from x1,y1 to x2,y2 using the currently set line style (see GFX_LineStyleSet()).
	GFX_LinePositionRelativeSet	This function sets the line cursor to a new position relative to the current position.
	GFX_LinePositionSet	This function sets the line cursor to a new position.
	GFX_LinePositionXGet	This function returns the current x position of the line cursor.
	GFX_LinePositionYGet	This function returns the current y position of the line cursor.
	GFX_LineStyleGet	Return the line style used when rendering lines.
	GFX_LineStyleSet	Set the line style to be used when rendering lines.
	GFX_LineToDraw	This function renders a line from current line cursor position (x,y) to (x2,y2) using the currently set line style (see GFX_LineStyleSet()).
	GFX_LineToRelativeDraw	This function renders a line from current line cursor position (x,y) to (x+dX,y+dY) using the currently set line style (see GFX_LineStyleSet()).
	GFX_MaxXGet	This function returns the maximum horizontal pixel coordinate.
	GFX_MaxYGet	This function returns the maximum vertical pixel coordinate.
	GFX_PolygonDraw	This function renders a polygon using the currently set line style and color.
	GFX_PRIMITIVE_Close	Closes an instance of the primitive layer specified by handle.
	GFX_PRIMITIVE_Deinitialize	Deinitializes the specified instance of the GFX Module.
	GFX_PRIMITIVE_Initialize	Initialize the GFX PRIMITIVE HAL Library.
	GFX_PRIMITIVE_Open	Opens a client instance of the primitive layer specified by index.
	GFX_RectangleDraw	This function renders a rectangular shape using the currently set line style and color.
	GFX_RectangleFillDraw	This function renders a filled rectangular shape using the currently set fill style and colors.
	GFX_RectangleRoundDraw	This function renders a rounded corner rectangular shape using the currently set line style and color.
	GFX_RectangleRoundFillDraw	This function renders a filled rectangular shape with round corners using the currently set fill style and colors.
	GFX_ScreenClear	Clears the screen to the currently set color (GFX_ColorSet()).
	GFX_TextCharDraw	This function renders the given character using the currently set color using the currently set font.
	GFX_TextCursorDraw	This is function GFX_TextCursorDraw.
	GFX_TextCursorPositionSet	This function sets the text cursor to a new position.
	GFX_TextCursorPositionXGet	This function returns the current x position of the text cursor.
	GFX_TextCursorPositionYGet	This function returns the current y position of the text cursor.
	GFX_TextCursorTypeGet	This is function GFX_TextCursorTypeGet.
	GFX_TextCursorTypeSet	This is function GFX_TextCursorTypeSet.
	GFX_TextCursorWidthGet	This is function GFX_TextCursorWidthGet.
	GFX_TextStringBoxDraw	This function renders the given string using the currently set color and font into a rectangular area.
	GFX_TextStringDraw	This function renders the given string of character using the currently set color using the currently set font.
	GFX_TextStringHeightGet	This function returns the height of the given font.
	GFX_TextStringWidthGet	This function returns the width of the given string using the given font.

	GFX_TransparentColorDisable	This function disables the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.
	GFX_TransparentColorEnable	This function sets the transparent color used in GFX_ImageDraw() functions and enables the transparent color feature.
	GFX_TransparentColorGet	This returns the current transparent color set for the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.
	GFX_TransparentColorStatusGet	This returns the current state of the transparent color feature used in GFX_ImageDraw() and GFX_ImagePartialDraw() functions.

Macros

	Name	Description
	GFX_PRIMITIVE_HANDLE_INVALID	Constant that defines the value of an Invalid Device Handle.
	GFX_TEXT_CURSOR_HEIGHT_DEFAULT	Constant that defines the default value of text cursor height.
	GFX_TEXT_CURSOR_WIDTH_DEFAULT	Constant that defines the default value of text cursor width.

Types

	Name	Description
	GFX_PRIMITIVE_HANDLE	Data type for GFX PRIMITIVE HAL handle.

Description

Module for Microchip Graphics Library - Graphic Primitive Layer

This module implements the primitive rendering routines of the Microchip Graphics Library. The routines are software fallbacks for accelerated functions of the display driver and display controller. The API is compatible with any Display Driver that is compliant with the requirements of the Display Driver interface of the Graphics Library.

File Name

gfx_primitive.h

Company

Microchip Technology Inc.

gfx_types_font.h

This header file defines the types used in fonts for the Microchip Graphics Library.

Macros

	Name	Description
	GFX_FONT_SPACE	Font space section. The fonts can be located in psv (constant) or program space in PIC24/dsPIC MCUs. This define allows for switching of the pointer type used to access the font structure in memory See: GraphicsConfig.h for the application define.
	GFX_UXCHAR	This is macro GFX_UXCHAR.
	GFX_XCHAR	Configuration for the text character size.

Structures

	Name	Description
	GFX_FONT_GLYPH_ENTRY	The structure describing the glyph entry in fonts.
	GFX_FONT_GLYPH_ENTRY_EXTENDED	The structure describing the intended glyph entry in fonts.
	GFX_FONT_HEADER	The structure used to define the font header.

Description

Graphics Resource Types Header for Microchip Graphics Library

This header defines the different structures used for fonts resources in the Microchip Graphics Library.

File Name

gfx_types_font.h

Company

Microchip Technology Inc.

gfx_types_image.h

This header file defines the types used in images for the Microchip Graphics Library.

Structures

	Name	Description
	GFX_MCHP_BITMAP_HEADER	The structure used to define the Microchip bitmap header.

Types

	Name	Description
	int16_gfx_image_prog	This is type int16_gfx_image_prog.
	int16_prog	This is type int16_prog.
	int16_prog_pack	This is type int16_prog_pack.
	int32_gfx_image_prog	This is type int32_gfx_image_prog.
	int32_prog	This is type int32_prog.
	int32_prog_pack	This is type int32_prog_pack.
	int8_gfx_image_prog	This is type int8_gfx_image_prog.
	int8_prog	This is type int8_prog.
	int8_prog_pack	This is type int8_prog_pack.
	uint16_gfx_image_prog	This is type uint16_gfx_image_prog.
	uint16_prog	This is type uint16_prog.
	uint16_prog_pack	This is type uint16_prog_pack.
	uint32_gfx_image_prog	This is type uint32_gfx_image_prog.
	uint32_prog	This is type uint32_prog.
	uint32_prog_pack	This is type uint32_prog_pack.
	uint8_gfx_image_prog	GFX Image data Types
	uint8_prog	General program space data types
	uint8_prog_pack	This is type uint8_prog_pack.

Description

Graphics Resource Types Header for Microchip Graphics Library

This header defines the different structures used for images resources in the Microchip Graphics Library.

File Name

gfx_types_image.h

Company

Microchip Technology Inc.

gfx_types_macros.h

Data types used in Microchip Graphics Library.

Enumerations

	Name	Description
	GFX_ALIGNMENT	Summary of the different text alignment supported in the library.
	GFX_BACKGROUND_TYPE	Specifies the different background fill types.
	GFX_FEATURE_STATUS	States of a feature that can be enabled/disabled at run time.
	GFX_FILL_STYLE	Specifies the available fill styles.
	GFX_FONT_ANTIALIAS_TYPE	Summary of the transparency types in text anti-aliasing.
	GFX_STATUS	Rendering status.
	GFX_STATUS_BIT	Additional rendering status.
	GFX_TEXT_CURSOR_TYPE	Enumeration defines types of text cursor supported.

Structures

	Name	Description
	GFX_BACKGROUND	Background information structure.
	GFX_DOUBLE_BUFFERING_MODE	Structure used for double buffering management.
	GFX_PARTIAL_IMAGE_PARAM	Partial Image information structure.
	GFX_RECTANGULAR_AREA	A generic rectangular area structure.

Description

Module for Microchip Graphics Library

This module defines the data types used in the Microchip Graphics Library for use of Primitive and Driver Layers.

File Name

gfx_types_macros.h

Company

Microchip Technology Inc.

gfx_types_palette.h

This header file defines the types used in palettes for the Microchip Graphics Library.

Unions

	Name	Description
	GFX_PALETTE_ENTRY	Defines the union used for each entry in the palette table.

Description

Graphics Resource Types Header for Microchip Graphics Library

This header defines the different structures used for palette resources in the Microchip Graphics Library.

File Name

gfx_types_palette.h

Company

Microchip Technology Inc.

gfx_types_resource.h

This is the definition of the Graphics Resource Types used with the Microchip Graphics Library.

Enumerations

	Name	Description
	GFX_RESOURCE	Specifies the different resource types in the library.

Macros

	Name	Description
	GFX_COMP_MASK	This is macro GFX_COMP_MASK.
	GFX_MEM_MASK	This is macro GFX_MEM_MASK.
	GFX_TYPE_MASK	This is macro GFX_TYPE_MASK.
	MCHP_BITMAP_NORMAL	no compression, palette is present for color depth = 8, 4 and 1 BPP
	MCHP_BITMAP_PALETTE_STR	palette is provided as a separate object (see PALETTE_HEADER) for color depth = 8, 4, and 1 BPP, ID to the palette is embedded in the bitmap.

Structures

	Name	Description
	GFX_RESOURCE_BINARY	Defines the structure used for the binary type resource.
	GFX_RESOURCE_FONT	Defines the structure used for the font type resource.
	GFX_RESOURCE_HDR	Defines the structure used for the resource types.
	GFX_RESOURCE_IMAGE	Defines the structure used for the image type resource.

	GFX_RESOURCE_PALETTE	Defines the structure used for the palette type resource.

Description

Graphics Resource Types Header for Microchip Graphics Library

This header defines the different structures used for the Graphics Resources used in the Graphics Library. This also contains helpful macros as well as data types used.

File Name

gfx_types_resource.h

Company

Microchip Technology Inc.

gfx_config_template.h

This header file template defines all the Graphics Library configurations available.

Macros

	Name	Description
	GFX_CONFIG_ALPHABLEND_DISABLE	Macro that disables the Alpha Blending feature.
	GFX_CONFIG_COLOR_DEPTH	Macro that sets the color depth of the application.
	GFX_CONFIG_DOUBLE_BUFFERING_DISABLE	Macro disables the support for double buffering in rendering of pixels to the frame buffer.
	GFX_CONFIG_FOCUS_DISABLE	Macro that disables the focus feature in objects.
	GFX_CONFIG_FONT_ANTIALIASED_DISABLE	Macro that disables the use of anti-aliased fonts.
	GFX_CONFIG_FONT_CHAR_SIZE	Macro that sets the size of the characters used in the fonts.
	GFX_CONFIG_FONT_EXTERNAL_DISABLE	Macro that disables sourcing of font resources from external sources.
	GFX_CONFIG_FONT_FLASH_DISABLE	Macro that disables sourcing of font resources from internal flash memory.
	GFX_CONFIG_FONT_RAM_DISABLE	Macro that disables sourcing of font resources from RAM sources.
	GFX_CONFIG_GRADIENT_DISABLE	Macro that disables the gradient fill feature.
	GFX_CONFIG_IMAGE_EXTERNAL_DISABLE	Macro that disables sourcing of image resources from external sources.
	GFX_CONFIG_IMAGE_FLASH_DISABLE	Macro that disables sourcing of image resources from internal flash memory.
	GFX_CONFIG_IMAGE_PADDING_DISABLE	Macro disables the padding of bits on images converted by the Graphics Resource Converter (GRC).
	GFX_CONFIG_IMAGE_RAM_DISABLE	Macro that disables sourcing of image resources from RAM sources.
	GFX_CONFIG_IPU_DECODE_DISABLE	Macro that disables RLE compression of images.
	GFX_CONFIG_PALETTE_DISABLE	Macro that disables the palette feature.
	GFX_CONFIG_PALETTE_EXTERNAL_DISABLE	Macro that disables the palette feature that are sourced in external resources.
	GFX_CONFIG_RLE_DECODE_DISABLE	Macro that disables RLE compression of images.
	GFX_CONFIG_TRANSPARENT_COLOR_DISABLE	Macro that disables the use of anti-aliased fonts.
	GFX_CONFIG_USE_KEYBOARD_DISABLE	This macro disables the keyboard support in objects.
	GFX_CONFIG_USE_TOUCHSCREEN_DISABLE	This macro disables the resistive touchscreen support in objects.
	GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE	Macro sets the size of the external font raster buffer.
	GFX_free	Macro that defines the free function for versatility when using Operating Systems.
	GFX_INSTANCES_NUMBER	Macro to specify the number of possible client modules.
	GFX_malloc	Macro that defines the malloc function for versatility when using Operating Systems.

Description

Module for Microchip Graphics Library

File Name

graphics_config.h

Company

Microchip Technology Inc.

Index

H

I