Graphics Library

Microchip Libraries for Applications (MLA)

Table of Contents

Graphics Library

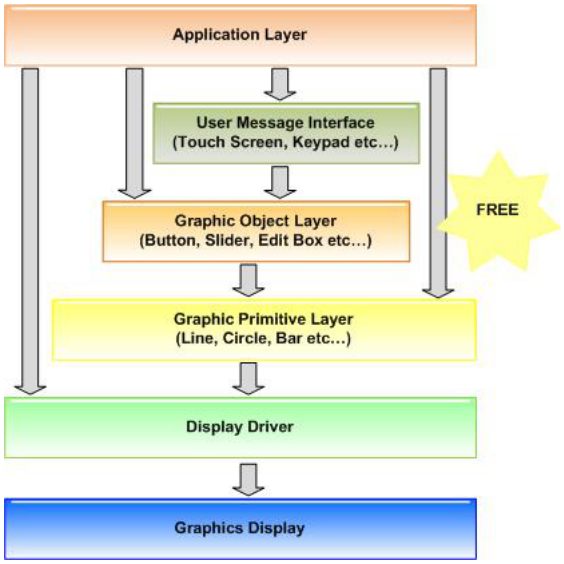
1 Graphics Library

1.1 Introduction

Microchip Graphics Library Help Documentation

Description

The Microchip Libraries for Applications (MLA) Graphics Library is a free, modular library optimized for Microchip 16-bit Microcontrollers. The library structure is shown in the following figure.



Microchip Graphics Library Architecture

1. Application Layer – This is the program that utilizes the Graphics Library.

2. User Message Interface- This is a layer should be implemented by user to provide messages for the library.

3. Graphics Object Layer – This layer renders the control objects such as button, list box, progress bar, meter and so on.

4. Graphics Primitives Layer – This layer implements the primitive rendering functions.

5. Device Display Driver – This layer is the graphics display driver component that is optimized to the actual display module used.

6. Graphics Display Module – This is the actual display module.

The library comes with 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.

1.2 Legal Information

This software distribution is controlled by the Legal Information at www.microchip.com/mla_license

1.3 Release Notes

This section describes the release notes of the Microchip Graphics Library.

Description

Microchip Graphics Library v4.00 (2013-12-20)

New This Release:

- Major revision of the API set of the library. All the API names are modified. Refer to summary of API changes in the Changes section.
- When rendering unfilled polygon, new API added and old API are renamed

NEW API Name	Old API Name
GFX_CircleDraw()	Circle()
GFX_RectangleDraw()	Rectangle()
GFX_RectangleRoundDraw()	Bevel()
GFX_PolygonDraw()	DrawPoly()

```
// example
GFX_LineStyleSet(GFX_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(BRIGHTRED);
GFX_RectangleDraw(left, top, right, bottom);
GFX_LineDraw(x1, y1, x2, y2);
GFX_CircleDraw(x, y, center);
```

- When rendering filled polygon, new API added and old API are renamed

NEW API Name	Old API Name
GFX_CircleFillDraw()	CircleFill()
GFX_RectangleFillDraw()	BarGradient() BarAlpha()
GFX_RectangleRoundFillDraw()	BevelFill()
GFX_BarDraw()	Bar()

```
// example
GFX_FillStyleSet(GFX_FILL_STYLE_ALPHA_COLOR);
GFX_ColorSet(BRIGHTRED);
GFX_RectangleFillDraw(left, top, right, bottom);
GFX_CircleFillDraw(x, y, center);
```

- When rendering filled polygons, the fill style is now a parameter (see GFX_FILL_STYLE). Alpha blending and gradient are now set as a fill style.

```
// example
GFX_FillStyleSet(GFX_FILL_STYLE_GRADIENT_DOUBLE_VER);
GFX_GradientColorSet(BLUE, RED);
GFX_RectangleFillDraw(50, 110, 150, 200, 20);

GFX_FillStyleSet(GFX_FILL_STYLE_ALPHA_COLOR);
GFX_ColorSet(GREEN);
GFX_RectangleRoundFillDraw(50, 110, 150, 200, 20);
```

- When rendering strings, you can now render the strings into a defined rectangular area and align the text.

NEW API Name	Old API Name
GFX_FontSet()	SetFont()
GFX_FontGet()	--

GFX_TextCursorPositionSet()	MoveTo()
GFX_TextCursorPositionXGet()	GetX()
GFX_TextCursorPositionYGet()	GetY()
GFX_FontAntiAliasSet()	GFX_Font_SetAntiAliasType()
GFX_FontAntiAliasGet()	GFX_Font_GetAntiAliasType()
deprecated	SetFontOrientation()
deprecated	GetFontOrientation()
GFX_TextCharDraw()	OutChar()
deprecated	OutText()
GFX_TextStringDraw()	OutTextXY()
GFX_TextStringBoxDraw()	--
GFX_TextStringHeightGet()	GetTextHeight()
GFX_TextStringWidthGet()	GetTextWidth()

```
// example
GFX_XCHAR   charArray[] = "Test String";

GFX_FontSet(myFont);
GFX_ColorSet(BRIGHTRED);

// render the whole string centered in the defined rectangular area
GFX_TextStringBoxDraw(  left,  top,
                        width, height,
                        charArray, 0,
                        GFX_ALIGN_CENTER);
```

- Added GFX_PixelArrayPut() and GFX_PixelArrayGet() to improve efficiency of functions that perform color fills functions as well as rendering of images. This leads to consolidation of image rendering functions in drivers to be implemented in primitive layer. The drivers will just need to implement these two functions.
- Combined API to mix styles. Previous implementation of line draw will need to setup the line type and line thickness separately. New API will now setup the line style which is an enumeration of line thickness and types (see GFX_LINE_STYLE).

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

GFX_ColorSet(BRIGHTBLUE);
GFX_LineStyleSet(GFX_LINE_STYLE_THICK_DASHED);
GFX_CircleDraw(100, 100,  50);
```

- Removed default font implementation. Application is now responsible in generating fonts.
- Removed default style scheme implementation. Application is now responsible in initializing the style schemes used by the objects.
- Button object now has two image pointers to allow easy use of a Button with two images. One image will be assigned to the press state and the other image assigned to the release state.
- Added support for primitive layer background variable. This variable defines the background information which optimizes the refresh of areas that the background occupies.
- Folder path for utilities:
- Graphics Resource Converter is now in <mla_root_folder>/framework/gfx/utilities/grc.
- External Memory Programmer is now in <mla_root_folder>/framework/gfx/utilities/memory_programmer.
- Restructure of the directory tree of the library.
- Library files are now located in: <mla_root_folder>/framework/gfx
- Driver files are now located in: <mla_root_folder>/framework/driver/gfx

- Demo projects are now located in: <mla_root_folder>/apps/gfx
- Documentation is now located in: <mla_root_folder>/doc
- Modified SPI driver to use enhanced buffering scheme. The new driver improved performance of reading SPI devices.
- Demo projects released with this version:
- Primitive Layer Demo
- Application Notes Demo
- Objects not ported with this release (may or may not be added in future releases):
- Chart object
- Grid object
- Analog Clock

Known Issues:

- This release do not support the following features:
- Palette on PIC24FJ256DA210 family of devices.
- Image Decoder routines
- Anti-aliasing and extended glyph features are not supported when using PIC24FJ256DA210 CHRGPU.
- When using PIC24FJ256GB210 PIM with Explorer 16 board that has a 5v Lumex LCD display, the S1D13517 demo does not run correctly.
- When using XC16 Compiler V1.00, add "-fno-ivopts" compile option. This is a known issue V1.00 of XC16.
- When using ListBox Widget, the widget height should be greater than the height of the font used for the widget.
- No GDD xml template included in this GFX release.

Fixes:

- Fixed issue on extended glyph for certain font (such as Thai) when used with Static text object is clipped.

Changes:

- Distribution of Graphics Object Default Scheme is discontinued. Application is now responsible for initializing style schemes.
- Distribution of Default Font is discontinued. Application is now responsible of creating fonts for application use.
- Summary of API changes.
- ***Primitive Layer:***

NEW API Name	Old API Name
GFX_Initialize()	InitGraph()
GFX_ScreenClear()	ClearDevice()
GFX_ColorSet()	SetColor()
GFX_ColorGet()	GetColor()
GFX_TransparentColorEnable()	TransparentColorEnable()
GFX_TransparentColorDisable()	TransparentColorDisable()
GFX_TransparentColorGet()	GetTransparentColor()
GFX_LineDraw()	Line()
GFX_LinePositionRelativeSet()	MoveRel()
GFX_LinePositionSet()	MoveTo()
GFX_LinePositionXGet()	--
GFX_LinePositionYGet()	--
GFX_LineToDraw()	LineTo()
GFX_LineToRelativeDraw()	LineRel()

GFX_LineStyleSet()	SetLineThickness(), SetLineType()
GFX_LineStyleGet()	--
GFX_FillStyleSet()	--
GFX_FillStyleGet()	--
GFX_GradientColorSet()	--
GFX_GradientStartColorGet()	--
GFX_GradientEndColorGet()	--
GFX_AlphaBlendingValueSet()	SetAlpha()
GFX_AlphaBlendingValueGet()	GetAlpha()
GFX_CircleDraw()	Circle()
GFX_RectangleDraw()	Rectangle()
GFX_RectangleRoundDraw()	Bevel()
GFX_PolygonDraw()	DrawPoly()
GFX_CircleFillDraw()	CircleFill()
GFX_RectangleFillDraw()	--
GFX_RectangleRoundFillDraw()	FillBevel()
GFX_BarDraw()	Bar()
GFX_ImageDraw()	PutImage()
GFX_ImagePartialDraw()	PutImagePartial()
GFX_ImageWidthGet()	GetImageWidth()
GFX_ImageHeightGet()	GetImageHeight()
GFX_FontSet()	SetFont()
GFX_FontGet()	--
GFX_TextCursorPositionSet()	MoveTo()
GFX_TextCursorPositionXGet()	GetX()
GFX_TextCursorPositionYGet()	GetY()
GFX_FontAntiAliasSet()	GFX_Font_SetAntiAliasType()
GFX_FontAntiAliasGet()	GFX_Font_GetAntiAliasType()
deprecated	SetFontOrientation()
deprecated	GetFontOrientation()
GFX_TextCharDraw()	OutChar()
deprecated	OutText()
GFX_TextStringDraw()	OutTextXY()
GFX_TextStringBoxDraw()	--
GFX_TextStringHeightGet()	GetTextHeight()
GFX_TextStringWidthGet()	GetTextWidth()
GFX_RGBConvert()	RGBConvert()
GFX_BackgroundSet()	--
GFX_BackgroundTypeSet()	--
GFX_BackgroundTypeGet()	--
GFX_BackgroundColorGet()	--
GFX_BackgroundImageGet()	--

GFX_BackgroundImageLeftGet()	--
GFX_BackgroundImageTopGet()	--
GFX_ExternalResourceCallback()	ExternalMemoryCallback()
GFX_DoubleBufferEnable()	SwitchOnDoubleBuffering()
GFX_DoubleBufferDisable()	SwitchOffDoubleBuffering()
GFX_DoubleBufferStatusGet()	--
GFX_DoubleBufferAreaGet()	--
GFX_DoubleBufferAreaMark()	InvalidateRectangle()
GFX_DoubleBufferSyncAllStatusSet()	InvalidateAll()
GFX_DoubleBufferSyncAllStatusGet()	--
GFX_DoubleBufferSyncAllStatusClear()	--
GFX_DoubleBufferSyncAreaCountSet()	--
GFX_DoubleBufferSyncAreaCountGet()	--
GFX_DoubleBufferSynchronizeRequest()	RequestDisplayUpdate()
GFX_DoubleBufferSynchronizeStatusGet()	IsDisplayUpdatePending()
GFX_DoubleBufferSynchronize()	UpdateDisplayNow()
deprecated	GetDrawBufferAddress()
deprecated	GetFrameBufferAddress()
GFX_DrawBufferInitialize()	--
GFX_DrawBufferSet()	--
GFX_DrawBufferGet()	--
GFX_FrameBufferSet()	--
GFX_FrameBufferGet()	--

- ***Object Layer:***

NEW API Name	**Old API Name**
GFX_GOL_ObjectStateSet()	SetState()
GFX_GOL_ObjectStateGet()	GetState()
GFX_GOL_ObjectStateClear()	ClearState()
GFX_GOL_ObjectAdd()	GOLAddObject()
GFX_GOL_ObjectFind()	GOLFindObject()
GFX_GOL_ObjectIDGet()	GetObjID()
GFX_GOL_ObjectTypeGet()	GetObjType()
GFX_GOL_ObjectNextGet()	GetObjNext()
GFX_GOL_ObjectListNew()	GOLNewList()
GFX_GOL_ObjectListSet()	GOLSetList()
GFX_GOL_ObjectListGet()	GOLGetList()
GFX_GOL_ObjectListFree()	GOLFree()
GFX_GOL_ObjectDelete()	GOLDeleteObject()
GFX_GOL_ObjectByIDDelete()	GOLDeleteObjectByID()
GFX_GOL_ObjectFocusSet()	GOLSetFocus()
GFX_GOL_ObjectFocusGet()	GOLGetFocus()

GFX_GOL_ObjectFocusNextGet()	GOLGetFocusNext()
GFX_GOL_ObjectFocusPrevGet()	GOLGetFocusPrev()
GFX_GOL_ObjectCanBeFocused()	GOLCanBeFocused()
GFX_GOL_ObjectListDraw()	GOLDraw()
GFX_GOL_ObjectListHide()	--
GFX_GOL_ObjectDrawEnable()	GOLRedraw()
GFX_GOL_ObjectRectangleRedraw()	GOLRedrawRec()
GFX_GOL_ObjectIsRedrawSet()	IsObjUpdated()
GFX_GOL_ObjectDrawDisable()	GOLDrawComplete()
deprecated	GOLCreateScheme()
GFX_GOL_ObjectStyleSchemeSet()	GOLSetScheme()
GFX_GOL_ObjectStyleSchemeGet()	GOLGetScheme()
deprecated	GOLGetSchemeDefault()
GFX_GOL_DrawCallbackSet()	--
GFX_GOL_MessageCallbackSet()	--
deprecated	GOLDrawCallback()
deprecated	GOLMessageCallback()
GFX_GOL_ButtonCreate()	BtnCreate()
GFX_GOL_ButtonDraw()	BtnDraw()
GFX_GOL_ButtonPressStateImageSet()	--
GFX_GOL_ButtonPressStateImageGet()	--
GFX_GOL_ButtonReleaseStateImageSet()	--
GFX_GOL_ButtonReleaseStateImageGet()	--
GFX_GOL_ButtonTextSet()	BtnSetText()
GFX_GOL_ButtonTextGet()	BtnGetText()
GFX_GOL_ButtonTextAlignmentSet()	--
GFX_GOL_ButtonTextAlignmentGet()	--
GFX_GOL_ButtonActionSet()	BtnMsgDefault()
GFX_GOL_ButtonActionGet()	BtnTranslateMsg()
GFX_GOL_CheckBoxCreate()	CbCreate()
GFX_GOL_CheckBoxDraw()	CbDraw()
GFX_GOL_CheckBoxTextSet()	CbSetText()
GFX_GOL_CheckBoxTextGet()	CbGetText()
GFX_GOL_CheckBoxTextAlignmentSet()	--
GFX_GOL_CheckBoxTextAlignmentGet()	--
GFX_GOL_CheckBoxActionSet()	CbMsgDefault()
GFX_GOL_CheckBoxActionGet()	CbTranslateMsg()
GFX_GOL_DigitalMeterCreate()	DmCreate()
GFX_GOL_DigitalMeterDraw()	DmDraw()
GFX_GOL_DigitalMeterTextAlignmentSet()	--
GFX_GOL_DigitalMeterTextAlignmentGet()	--
GFX_GOL_DigitalMeterValueSet()	DmSetValue()

GFX_GOL_DigitalMeterValueGet()	DmGetValue()
GFX_GOL_DigitalMeterIncrement()	DmIncValue()
GFX_GOL_DigitalMeterDecrement()	DmDecValue()
GFX_GOL_DigitalMeterActionGet()	DmTranslateMsg()
GFX_GOL_EditBoxCreate()	EbCreate()
GFX_GOL_EditBoxDraw()	EbDraw()
GFX_GOL_EditBoxTextSet()	EbSetText()
GFX_GOL_EditBoxTextGet()	EbGetText()
GFX_GOL_EditBoxTextAlignmentSet()	--
GFX_GOL_EditBoxTextAlignmentGet()	--
GFX_GOL_EditBoxActionSet()	EbMsgDefault()
GFX_GOL_EditBoxActionGet()	EbTranslateMsg()
GFX_GOL_EditBoxCharAdd()	EbAddChar()
GFX_GOL_EditBoxCharRemove()	EbDeleteChar()
GFX_GOL_GroupboxCreate()	GbCreate()
GFX_GOL_GroupboxDraw()	GbDraw()
GFX_GOL_GroupboxTextSet()	GbSetText()
GFX_GOL_GroupboxTextGet()	GbGetText()
GFX_GOL_GroupboxTextAlignmentSet()	--
GFX_GOL_GroupboxTextAlignmentGet()	--
GFX_GOL_GroupboxActionGet()	GbTranslateMsg()
GFX_GOL_ListBoxCreate()	LbCreate()
GFX_GOL_ListBoxDraw()	LbDraw()
GFX_GOL_ListBoxItemAdd()	LbAddItem()
GFX_GOL_ListBoxTextAlignmentSet()	--
GFX_GOL_ListBoxTextAlignmentGet()	--
GFX_GOL_ListBoxActionSet()	LbMsgDefault()
GFX_GOL_ListBoxActionGet()	LbTranslateMsg()
GFX_GOL_ListBoxItemListGet()	LbGetItemList()
GFX_GOL_ListBoxItemListRemove()	LbDelItemsList()
GFX_GOL_ListBoxItemRemove()	LbDelItem()
GFX_GOL_ListBoxItemFocusSet()	LbSetFocusedItem()
GFX_GOL_ListBoxItemFocusGet()	LbGetFocusedItem()
GFX_GOL_ListBoxItemCountGet()	LbGetCount()
GFX_GOL_ListBoxItemImageSet()	LbSetBitmap()
GFX_GOL_ListBoxItemImageGet()	LbGetBitmap()
GFX_GOL_ListBoxItemSelectStatusSet()	LbSetSel()
GFX_GOL_ListBoxItemSelectStatusClear()	LbClrSel()
GFX_GOL_ListBoxSelectionChange()	LbChangeSel()
GFX_GOL_ListBoxSelectionGet()	LbGetSel()
GFX_GOL_ListBoxVisibleItemCountGet()	LbGetVisibleCount()
GFX_GOL_MeterCreate()	MtrCreate()

GFX_GOL_MeterDraw()	MtrDraw()
GFX_GOL_MeterMaximumValueGet()	--
GFX_GOL_MeterMinimumValueGet()	--
GFX_GOL_MeterTitleFontSet()	MtrSetTitleFont()
GFX_GOL_MeterValueFontSet()	MtrSetValueFont()
GFX_GOL_MeterTypeSet()	--
GFX_GOL_MeterActionSet()	MtrMsgDefault()
GFX_GOL_MeterActionGet()	MtrTranslateMsg()
GFX_GOL_MeterIncrement()	MtrIncVal()
GFX_GOL_MeterDecrement()	MtrDecVal()
GFX_GOL_MeterRangeSet()	--
GFX_GOL_MeterScaleColorsSet()	MtrSetScaleColors()
GFX_GOL_MeterValueSet()	MtrSetVal()
GFX_GOL_MeterValueGet()	MtrGetVal()
GFX_GOL_PictureControlCreate()	PictCreate()
GFX_GOL_PictureControlDraw()	PictDraw()
GFX_GOL_PictureControlImageSet()	PictSetBitmap()
GFX_GOL_PictureControlImageGet()	PictGetBitmap()
GFX_GOL_PictureControlActionGet()	PictTranslateMsg()
GFX_GOL_PictureControlPartialSet()	--
GFX_GOL_PictureControlScaleSet()	PictSetScale()
GFX_GOL_ProgressBarCreate()	PbCreate()
GFX_GOL_ProgressBarDraw()	PbDraw()
GFX_GOL_ProgressBarPositionSet()	PbSetPos()
GFX_GOL_ProgressBarPositionGet()	PbGetPos()
GFX_GOL_ProgressBarRangeSet()	PbSetRange()
GFX_GOL_ProgressBarRangeGet()	PbGetRange()
GFX_GOL_ProgressBarActionGet()	PbTranslateMsg()
GFX_GOL_RadioButtonCreate()	RbCreate()
GFX_GOL_RadioButtonDraw()	RbDraw()
GFX_GOL_RadioButtonTextSet()	RbSetText()
GFX_GOL_RadioButtonTextGet()	RbGetText()
GFX_GOL_RadioButtonCheckSet()	RbSetCheck()
GFX_GOL_RadioButtonCheckGet()	RbGetCheck()
GFX_GOL_RadioButtonActionSet()	RbMsgDefault()
GFX_GOL_RadioButtonActionGet()	RbTranslateMsg()
GFX_GOL_RadioButtonTextAlignmentSet()	--
GFX_GOL_RadioButtonTextAlignmentGet()	--
GFX_GOL_ScrollBarCreate()	SldCreate()
GFX_GOL_ScrollBarDraw()	SldDraw()
GFX_GOL_ScrollBarPageSet()	SldSetPage()
GFX_GOL_ScrollBarPageGet()	SldGetPage()

GFX_GOL_ScrollBarRangeSet()	SldSetRange()
GFX_GOL_ScrollBarRangeGet()	SldGetRange()
GFX_GOL_ScrollBarPositionSet()	SldSetPos()
GFX_GOL_ScrollBarPositionGet()	SldGetPos()
GFX_GOL_ScrollBarPositionIncrement()	SldIncPos()
GFX_GOL_ScrollBarPositionDecrement()	SldDecPos()
GFX_GOL_ScrollBarActionSet()	SldMsgDefault()
GFX_GOL_ScrollBarActionGet()	SldTranslateMsg()
GFX_GOL_StaticTextCreate()	StCreate()
GFX_GOL_StaticTextDraw()	StDraw()
GFX_GOL_StaticTextAlignmentSet()	--
GFX_GOL_StaticTextAlignmentGet()	--
GFX_GOL_StaticTextSet()	StSetText()
GFX_GOL_StaticTextGet()	StGetText()
GFX_GOL_StaticTextActionGet()	StTranslateMsg()
GFX_GOL_TextEntryCreate()	TeCreate()
GFX_GOL_TextEntryKeyListCreate()	TeCreateKeyMembers()
GFX_GOL_TextEntryDraw()	TeDraw()
GFX_GOL_TextEntryBufferClear()	TeClearBuffer()
GFX_GOL_TextEntryBufferSet()	TeSetBuffer()
GFX_GOL_TextEntryBufferGet()	TeGetBuffer()
GFX_GOL_TextEntryCharAdd()	TeAddChar()
GFX_GOL_TextEntryKeyCommandSet()	TeSetKeyCommand()
GFX_GOL_TextEntryKeyCommandGet()	TeGetKeyCommand()
GFX_GOL_TextEntryKeyIsPressed()	TeIsKeyPressed()
GFX_GOL_TextEntryKeyTextSet()	TeSetKeyText()
GFX_GOL_TextEntryKeyMemberListDelete()	TeDelKeyMembers()
GFX_GOL_TextEntryLastCharDelete()	--
GFX_GOL_TextEntrySpaceCharAdd()	TeSpaceChar()
GFX_GOL_TextEntryActionSet()	TeMsgDefault()
GFX_GOL_TextEntryActionGet()	TeTranslateMsg()
GFX_GOL_WindowCreate()	WndCreate()
GFX_GOL_WindowDraw()	WndDraw()
GFX_GOL_WindowImageSet()	--
GFX_GOL_WindowImageGet()	--
GFX_GOL_WindowTextSet()	WndSetText()
GFX_GOL_WindowTextGet()	WndGetText()
GFX_GOL_WindowActionGet()	WndTranslateMsg()
GFX_GOL_WindowTextAlignmentSet()	--
GFX_GOL_WindowTextAlignmentGet()	--

- ***Driver Layer:***

NEW API Name	Old API Name
GFX_PixelPut()	PutPixel()
GFX_PixelGet()	GetPixel()
GFX_MaxXGet()	GetMaxX()
GFX_MaxYGet()	GetMaxY()
GFX_DisplayBrightnessSet()	DisplayBrightness()
deprecated	SetClip()
deprecated	SetClipRgn()
deprecated	GetClipLeft()
deprecated	GetClipTop()
deprecated	GetClipRight()
deprecated	GetClipBottom()

Deprecated Items:

- Round Dial object is discontinued.
- Transitions routines are discontinued.
- See API Changes Table summary for other deprecated functions.

Application Notes

- AN1136 How to Use Widgets in Microchip Graphics Library
- AN1182 Fonts in the Microchip Graphics Library
- AN1227 Using a Keyboard with the Microchip Graphics Library
- AN1246 How to Create Widgets in Microchip Graphics Library
- AN1368 Developing Graphics Applications using PIC MCU with Integrated Graphics Controller

PIC Family

This version of the library supports PIC24 and dsPIC Families.

Development Tools

This graphics library release was tested with

- MPLAB X IDE Version 2.00
- XC16 v1.21.

Documentation of Resources and Utilities

- The Graphics Library Help File and API document is located in
- <mla_root_folder>/doc/
- "Graphics Resource Converter" documentation is located in
- <mla_root_folder>/framework/gfx/utilities/grc
- External Memory Programmer documentation is located in
- <mla_root_folder>/framework/gfx/utilities/memory_programmer

where:

- "mla_root_folder" is the location of the Microchip Libraries for Applications installation.

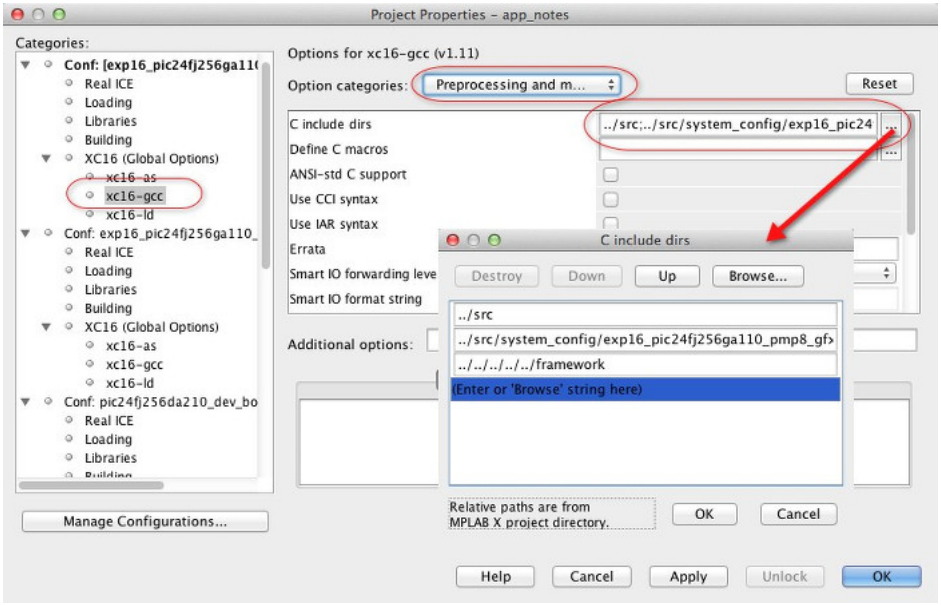
Information in Adding Resources Generated by the Graphics Resource Converter (GRC)

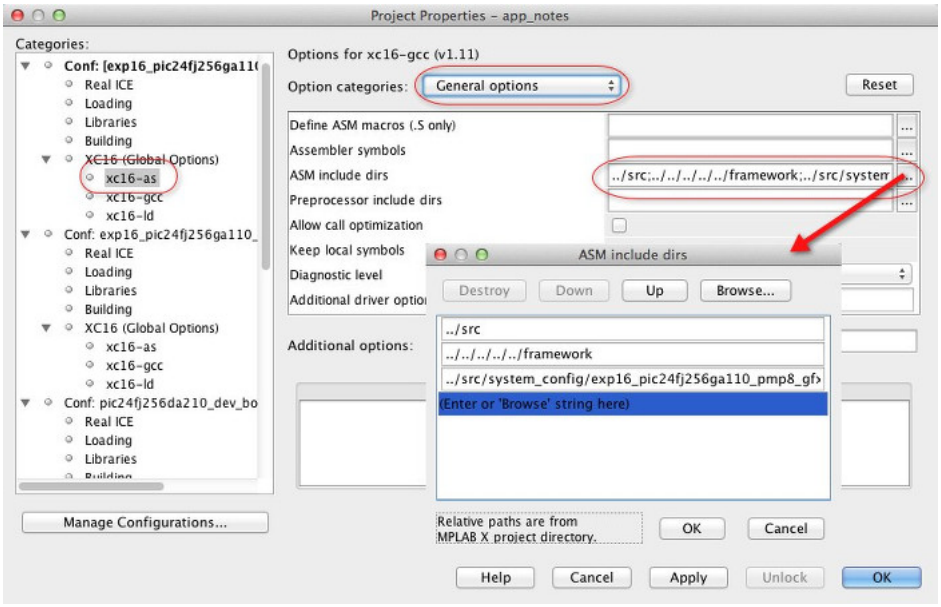
The GRC is used when modifying or adding fonts and images to applications. GRC creates multiple files for resources placed in internal flash or external memory. The following files are generated by GRC:

- Resources placed in internal flash.

- file_name_given_reference.c
- file_name_given.h
- file_name_given.S
- Resources placed in external memory.
- file_name_given_reference.c
- file_name_given.h
- file_name_given.hex

When adding resources in MPLAB X projects, it is necessary to add include paths to both C and assembler (when adding resources placed in internal flash).





Previous Versions Log

v3.06 (2012-08-22)

New:

- Partial rendering of images now supported (see PutImagePartial()).
- Double Buffering is now supported in Microchip Low-Cost Controllerless (LCC) Graphics Display Driver.

- Added dsPIC33EPXXX device family support.
- Added S1D13522 EPD Controller Driver.
- Added E-Paper Epson Demo.
- Added Alpha-Blend support for Bar() function.
- Graphics Resource Converter (GRC) now allows for padding and non-padding bitmap images. Bitmap images are padded which means that each horizontal line will start on a byte boundary. The option has been added to allow for conversion of bitmap resources to be non-padded which allows the least resource space and controllers with windowing that auto increments to use them.
- For XC16 or C30 builds, internal fonts can now be placed program memory. If the font data or a combination of font data resources exceed the 32 Kbyte limit of the data memory space, a define, USE_GFX_FONT_IN_PROGRAM_SECTION should be defined in graphics configuration header. This will place the font resource data in program memory space.

Changes:

- Graphics Object Demo now uses images that are RLE compressed.
- Address range check for GFX_EPMP_CS1_MEMORY_SIZE and GFX_EPMP_CS2_MEMORY_SIZE in Graphics Module in PIC24FJ256DA210 Display Driver file (mchpGfxDrv.c) is modified for strict checks of allocated address pins for the EPMP. See "Migration Changes" below for the address lines needed to be allocated.
- Swapped the bit orientation for the 1 BPP bitmap images. The previous versions of the library expects the left most pixel at the MSBit. This has been changed so the the left most pixel is located at the LSBit. The change was made to accommodate controllers that have windowing. This also makes the pixel orientation consistent with the pixel orientation of 4bpp images.

Fixes:

- Fix FillBevel() & FillCircle() to avoid rendering lines more than once.

Deprecated Items: The following Resistive Touch Screen macro names are replaced for readability and flexibility if use:

- TRIS_XPOS - replaced by ResistiveTouchScreen_XPlus_Config_As_Input()
- TRIS_YPOS - replaced by ResistiveTouchScreen_YPlus_Config_As_Input()
- TRIS_XNEG - replaced by ResistiveTouchScreen_XMinus_Config_As_Input()
- TRIS_YNEG - replaced by ResistiveTouchScreen_YMinus_Config_As_Output()
- LAT_XPOS - replaced by ResistiveTouchScreen_XPlus_Drive_High()
- LATS_YPOS - replaced by ResistiveTouchScreen_YPlus_Drive_High()
- LAT_XNEG - replaced by ResistiveTouchScreen_XMinus_Drive_Low()
- LAT_YNEG - replaced by ResistiveTouchScreen_YMinus_Drive_Low()

Migration Changes:

- To use the new Resistive Touchscreen macros, replace the TouchScreenResistive.c file with the version in this release. Then replace the hardware profile macros to use the new macro names. Existing hardware profile can still be used but build warnings will appear.
- If custom display driver is used and the PutImage() functions are implemented for faster rendering, the new partial image rendering feature requires these PutImage() functions to be modified. See PutImagePartial() API description and implementation in Primitive.c for details.
- Address range check for GFX_EPMP_CS1_MEMORY_SIZE and GFX_EPMP_CS2_MEMORY_SIZE that are defined in hardware profile are modified when using the Graphics Module in PIC24FJ256DA210 Device and using external memory for display buffer, the driver file (mchpGfxDrv.c). This check allocates address pins for the EPMP. Modify the GFX_EPMP_CS1_MEMORY_SIZE and GFX_EPMP_CS2_MEMORY_SIZE values set in the hardware profile to match the table shown below:

EPMP Memory Size	EPMP Address Lines
GFX_EPMP_CSx_MEMORY_SIZE <= 0x20000 (bytes)	Use PMA[15:0]
0x20000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x40000(bytes)	Use PMA[16:0]
0x40000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x80000(bytes)	Use PMA[17:0]

0x80000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x100000(bytes)	Use PMA[18:0]
0x100000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x200000(bytes)	Use PMA[19:0]
0x200000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x400000(bytes)	Use PMA[20:0]
0x400000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x800000(bytes)	Use PMA[21:0]
0x800000 (bytes) < GFX_EPMP_CSx_MEMORY_SIZE <= 0x1000000(bytes)	Use PMA[22:0]

- 1BPP images needs to be regenerated using the "Graphics Resource Converter" since the bit orientation is swapped. When rendering a 1 BPP image, the PutImage() function will expect the left most pixel to be located in the LSBit of each word.
- To utilize the new feature where the fonts can be placed in the program memory, to remove the 32 KByte limit for data space in XC16 or C30 builds, fonts must be regenerated using the "Graphics Resource Converter" and then add #define USE_GFX_FONT_IN_PROGRAM_SECTION in GraphicsConfig.h.

Known Issues:

- Extended glyph for certain font (such as Thai) when used with Static text widget is clipped. Future version will add additional vertical text alignment to the static text widget
- Anti-aliasing and extended glyph features are not supported when using PIC24FJ256DA210 CHRGPU.
- SPI flash programming on the Epson S1D13517 PICtail board Rev 1.1 is not always reliable, the S1D13517 demo no longer uses external memory flash in the example.
- When using PIC24FJ256GB210 PIM with Explorer 16 board that has a 5v Lumex LCD display, the S1D13517 demo does not run correctly.
- When using XC16 Compiler V1.00, add "-fno-ivopts" compile option. This is a known issue V1.00 of XC16.
- When using ListBox Widget, the widget height should be greater than the height of the font used for the widget.

v3.04.01 (2012-04-03)

New:

- No new items on this release.

Changes:

- Structure of this help file is modified. Section names that list APIs now has the API in the name.

Fixes:

- Fix BarGradient() and BevelGradient() when USE_NONBLOCKING_CONFIG config is enabled.
- Added missing GbSetText() function in GroupBox widget.
- Fix SetPalette() to work with palette stored in external memory.

Deprecated Items:

- TYPE_MEMORY - replaced by GFX_RESOURCE
- EXTDATA - replaced by GFX_EXTDATA
- BITMAP_FLASH - replaced by IMAGE_FLASH
- BITMAP_RAM - replaced by IMAGE_RAM
- BITMAP_EXTERNAL - replaced by GFX_EXTDATA
- EEPROM.h and EEPROM.c are replaced by the following files:
- MCHP25LC256.c - source code
- MCHP25LC256.h - header file
- In the HardwareProfile.h add #define USE_MCHP25LC256 to use the new driver.

Migration Changes:

- none

Known Issues:

- Extended glyph for certain font (such as Thai) when used with Static text widget is clipped. Future version will add

additional vertical text alignment to the static text widget.

- Anti-aliasing and extended glyph features are not supported when using PIC24FJ256DA210 CHRGPU.
- SPI flash programming on the Epson S1D13517 PICtail board Rev 1.1 is not always reliable, the S1D13517 demo no longer uses external memory flash in the example.
- When using PIC24FJ256GB210 PIM with Explorer 15 board with a 5v Lumex LCD display, the S1D13517 demo does not run correctly.

v3.04 (2012-02-15)

New:

- Font Table is updated to version 2 to accommodate anti-alias font, and extended glyph.
- Added anti-aliasing support for fonts (2bpp). See Primitive demo for an example.
- Added extended-glyph support for fonts. See demo in the AppNote demo - AN1182, This demo now includes Hindi and Thai font.
- Added 32-bit CRC code for resources to be placed in external memory. This CRC value can be used to verify if the data in external memory is valid or not. Refer to Graphics Resource Converter release notes for details. Demos that uses external memory as a resource are now checking the CRC value, and if invalid, automatically requests for external memory resource programming.
- Added new demos specific to PIC24FJ256DA210:
- Elevator Demo - This demo shows an elevator status monitor that indicates the current location of the elevator car.
- RCCGPU-IPU Demo - formerly PIC24F_DA Demo. This demo shows how RCCGPU and IPU modules are used.
- Color Depth Demo - This demos shows how 1bpp, 4bpp and 8 bpp color depths applications are implemented.
- Remote Control Demo - This demo shows how a universal remote control can be implemented using RF4CE communication protocol. This demo also integrates the MRF24J40MA (a certified 2.4 GHz IEEE 802.15.4 radio transceiver module) for sending RF4CE messages to the paired device.
- Added driver for Solomon Systech 132x64 OLED/PLED Display Controller SSD1305

Changes:

- Modified Resistive Touch Screen calibration. Now the calibration stores the 8 touch points to support large touch panels.
- Modified 4-wire Resistive Touch Screen driver to support build time setting of single samples and auto-sampling of resistive touch inputs.
- Naming of internal and external resource files (files that defined fonts and images) in most demos are now standardized to use the same naming convention.
- Support for Graphics PICTail v2 (AC164127) is now discontinued.
- Merged "JPEG" demo and "Image Decoders" demo to "Image Decoder" demo.
- Graphics Resource Converter upgrade (Version 3.17.47) - refer to "Graphics Resource Converter Help.pdf" located in <install directory>/Microchip Solutions/Microchip/Graphics/bin/grc for details.
- External Memory Programmer upgrade (Version 1.00.01) - refer to "External Memory Programmer Help.pdf" located in <install directory>/Microchip Solutions/Microchip/Graphics/bin/memory_programmer for details.

Fixes:

- Fix PushRectangle() issue where one line of pixel is not being updated.
- Fix TextEntry widget issue where the string is not displayed when the allocated string buffer length is equal to the maximum string length set in the widget.
- Fonts maximum character height is now set to 2^16.

Deprecated Items:

- TYPE_MEMORY - replaced by GFX_RESOURCE
- EXTDATA - replaced by GFX_EXTDATA
- BITMAP_FLASH - replaced by IMAGE_FLASH
- BITMAP_RAM - replaced by IMAGE_RAM
- BITMAP_EXTERNAL - replaced by GFX_EXTDATA

- EEPROM.h and EEPROM.c are replaced by the following files:
- MCHP25LC256.c - source code
- MCHP25LC256.h - header file
- in the HardwareProfile.h add #define USE_MCHP25LC256 to use the new driver.

Migration Changes:

- For existing code that wants to use the new anti-aliased fonts or extended glyph features: regenerate the font tables using the "Graphics Resource Converter" with the check box for the required feature set to be enabled. For anti-aliased fonts, add the macro #define USE_ANTIALIASED_FONTS in the GraphicsConfig.h

Known Issues:

- Extended glyph for certain font (such as Thai) when used with Static text widget is clipped. Future version will add additional vertical text alignment to the static text widget
- Anti-aliasing and extended glyph features are not supported when using PIC24FJ256DA210 CHRGPU.
- SPI flash programming on the Epson S1D13517 PICtail board Rev 1.1 is not always reliable, the S1D13517 demo no longer uses external memory flash in the example.
- When using PIC24FJ256GB210 PIM with Explorer 15 board with a 5v Lumex LCD display, the S1D13517 demo does not run correctly.

v3.03 (v3.02)

New:

- Added custom video playback from SD Card in SSD1926 Demo. Video frames are formatted to RGB565 format.
- Added support for 1bpp, 4bpp and 8 bpp color depth on Chart widget.
- Added support for Display Boards from Semitron
- Seiko 35QVW1T
- Seiko 43WVW1T

Changes:

- Maximum font height is now 256 pixels.
- Modified EditBox behavior
- Caret is now by default enabled.
- Caret can now be shown even if USE_FOCUS is disabled.
- Applications can now respond to touchscreen event on EditBoxes when USE_FOCUS is disabled.
- Modified resistive touchscreen calibration sequence.
- Graphics Resource Converter upgrade (Version 3.8.21) - refer to "Graphics Resource Converter Help.pdf" located in <install directory>/Microchip Solutions/Microchip/Graphics/bin/grc for details.
- External Memory Programmer upgrade (Version 1.00.01) - refer to "External Memory Programmer Help.pdf" located in <install directory>/Microchip Solutions/Microchip/Graphics/bin/memory_programmer for details.

Fixes:

- Fix Low Cost Controller display driver issue when run with Resistive Touch Screen driver that uses single samples.
- Fix issue on PIC24FJ256DA210 display driver PutImage()'s issue when using palette on 4 bpp and 1 bpp images.
- Fix issue on PIC24FJ256DA210 display driver PutImage()'s missing lines when the image last pixel row or column falls on the edge of the screen.
- Fix Resistive Touch Screen driver issue on rotated screens.
- Fix GetImageHeight() GetImageWidth() issues for images that are RLE compressed.
- EPMP module is now disabled when memory range defined for display buffer is located in internal memory.
- Add default color definitions in gfxcolors.h for 1bpp, 4bpp 8bpp and 16 bpp. Added back legacy colors.
- Fix HX8347 driver WritePixel() macro when using 16bit PMP mode.
- Fix PIC24FJ256DA210 display driver issue on source data (continuous and discontinuous data) when doing block copies of memory using RCCGPU.

Deprecated Items:

- TYPE_MEMORY - replaced by GFX_RESOURCE
- EXTDATA - replaced by GFX_EXTDATA
- BITMAP_FLASH - replaced by IMAGE_FLASH
- BITMAP_RAM - replaced by IMAGE_RAM
- BITMAP_EXTERNAL - replaced by GFX_EXTDATA
- EEPROM.h and EEPROM.c are replaced by the following files:
- MCHP25LC256.c - source code
- MCHP25LC256.h - header file
- in the HardwareProfile.h add #define USE_MCHP25LC256 to use the new driver.

Migration Changes:

- EditBox widget's caret behavior is now by default enabled when USE_FOCUS is set. To disable, ignore all messages for the edit box by returning a zero when in GOLMsgCallback().

Known Issues:

- PutImage() does not work when using PIC24FJ256DA210 and look up table is used on images located at EDS memory with color depth less than 8bpp.
- External Memory Programmer utility does not work with Graphics PICTail v2 (AC164127)
- When using PIC24FJ256GB210 PIM with Explorer 15 board with a 5v Lumex LCD display, the S1D13517 demo does not run correctly.
- Font tables are limited to 256 pixel character height.

v3.02

New:

- Added custom video playback from SD Card in SSD1926 Demo. Video frames are formatted to RGB565 format.
- Added support for 1bpp, 4bpp and 8 bpp color depth on Chart widget.
- Added support for Display Boards from Semitron
- Seiko 35QVW1T
- Seiko 43WVW1T

Changes:

- Maximum font height is now 256 pixels.
- Modified EditBox behavior
- Caret is now by default enabled.
- Caret can now be shown even if USE_FOCUS is disabled.
- Applications can now respond to touchscreen event on EditBoxes when USE_FOCUS is disabled.
- Modified resistive touchscreen calibration sequence.
- Graphics Resource Converter upgrade (Version 3.8.21) - refer to

"Graphics Resource Converter Help.pdf" located in <install directory>/Microchip Solutions/Microchip/Graphics/bin/grc for details.

- External Memory Programmer upgrade (Version 1.00.01) - refer to "External Memory Programmer Help.pdf" located in <install directory>/Microchip Solutions/Microchip/Graphics/bin/memory_programmer for details.

Fixes:

- Fix issue on PIC24FJ256DA210 display driver PutImage()'s missing lines when the image last pixel row or column falls on the edge of the screen.
- Fix Resistive Touch Screen driver issue on rotated screens.
- Fix GetImageHeight() GetImageWidth() issues for images that are RLE compressed.
- EPMP module is now disabled when memory range defined for display buffer is located in internal memory.

- Add default color definitions in gfxcolors.h for 1bpp, 4bpp 8bpp and 16 bpp. Added back legacy colors.
- Fix HX8347 driver WritePixel() macro when using 16bit PMP mode.
- Fix PIC24FJ256DA210 display driver issue on source data (continuous and discontinuous data) when doing block copies of memory using RCCGPU.

Deprecated Items:

- TYPE_MEMORY - replaced by GFX_RESOURCE
- EXTDATA - replaced by GFX_EXTDATA
- BITMAP_FLASH - replaced by IMAGE_FLASH
- BITMAP_RAM - replaced by IMAGE_RAM
- BITMAP_EXTERNAL - replaced by GFX_EXTDATA
- EEPROM.h and EEPROM.c are replaced by the following files:
- MCHP25LC256.c - source code
- MCHP25LC256.h - header file
- in the HardwareProfile.h add #define USE_MCHP25LC256 to use the new driver.

Migration Changes:

- EditBox widget's caret behavior is now by default enabled when USE_FOCUS is set. To disable, ignore all messages for the edit box by returning a zero when in GOLMsgCallback().

Known Issues:

- PutImage() does not work when using PIC24FJ256DA210 and look up table is used on images located at EDS memory with color depth less than 8bpp.
- External Memory Programmer utility does not work with Graphics PICTail v2 (AC164127)
- When using PIC24FJ256GB210 PIM with Explorer 15 board with a 5v Lumex LCD display, the S1D13517 demo does not run correctly.
- Font tables are limited to 256 pixel character height.

v3.01 (v3.00)

New:

- Graphics External Memory Programmer ported to java version.
- Two options to program boards:
- UART option if the board supports UART interface
- USB option if the board support USB device interface
- when installing the USB drivers for the programmer utility Use the drivers located in "<install directory>/Microchip/Utilities/USB Drivers/MPLABComm"
- For detailed usage, please refer to the External Programmer help file
- Added Analog Clock widget.
- Added new driver Epson S1D13517 display driver with additional driver features:
- Gradient
- Alpha Blending
- 16/24 bits per pixel (bpp)
- Added new Graphics PICtail Plus Epson S1D13517 Board (AC164127-7)
- Added new Graphics Display Truly 5.7" 640x480 Board (AC164127-8)
- Added new Graphics Display Truly 7" 800x480 Board (AC164127-9)
- Added 24bpp support.
- Added a specific PIC24FJ256DA210 demo, PIC24F_DA
- Graphics Resource Converter - refer to the Graphics Resource Converter help file for release note information.

- New PIC32MX Low-Cost Controllerless Graphics PICTail Board (AC164144)
- Added Run Length Encoding (RLE) compression for bitmap images.
- RLE4 - compression for 4-bit palette (16 color) images
- RLE8 - compression for 8-bit palette (256 color) images
- Added Transparency feature for PutImage() functions in Primitive Layer. For Driver Layer, this feature is enabled in the following drivers:
- mchpGfxDrv - Microchip Graphics Controller Driver
- SSD1926 - Solomon Systech Display Controller Driver
- Added new demo for Graphics PICtail Plus Epson S1D13517 Board (S1D13517 Demo)
- Added AR1020 Resistive Touch Screen Controller beta support.
- Added support for MikroElektronika "mikroMMB for PIC24" board.
- Added DisplayBrightness() function for display drivers that have an option to control the display back light.
- MPLAB X demo project support (BETA)
- Tested with MPLAB X Beta 6
- Each demo project contains multiple configuration schemes
- The graphics object layer uses a default scheme structure. If the application wishes to use a different default scheme, the application will need to define GFX_SCHEMEDEFAULT in GraphicsConfig header file.

Changes:

- Relocated all Graphics Demo projects under Graphics directory (<install directory>/Graphics).
- Works with Graphics Display Designer version 2.1
- Works with Graphics Resource Converter version 3.3
- Removed IPU decoding from the Primitive demo
- Removed ImageFileConverter application from Image Decoder demo
- Use Graphics Resource Converter to generate output for the demo.
- Shorten file name by using abbreviated names
- Refer to abbreviations.htm in the Microchip help directory for details
- Change the "Alternative Configurations" directory in the demos to "Configs"
- Changed the "Precompiled Demos" directory in the demos to "Precompiled Hex"
- Combined all application note demos (AN1136, AN1182, AN1227 and AN1246) into one demo project (AppNotes).
- Modified External Memory and JPEG demos to include USB device mode to program external flash memory.
- Moved the location of the COLOR_DEPTH setting from the HardwareProfile.h to the GraphicsConfig.h
- Removed USE_DRV_FUNCTIONNAME (example USE_DRV_LINE to implement the Line() function in the driver) option to implement Primitive Layer functions in the Driver Layer. The Primitive Layer functions are now modified to have weak attributes, so when the driver layer implements the same function, the one in the driver will be used at build time.
- Modified HardwareProfile.h for Graphics demo boards and development Platforms.
- When using Resistive Touch Screen: add macro USE_TOUCHSCREEN_RESISTIVE
- When using AR1020 as the touch screen controller: add macro USE_TOUCHSCREEN_AR1020
- When using SPI Flash Memory (SST25VF016) in Graphics Development Boards: add macro USE_SST25VF016
- When using Parallel Flash Memory (SST39LF400) in PIC24FJ256DA210 Development Board: add macro USE_SST39LF400
- When using Parallel Flash Memory (SST39LF400) in PIC24FJ256DA210 Development Board: add macro USE_SST39LF400
- Added function pointers for Non-Volatile Memories when used in the Demos.
- NVM_SECTORERASE_FUNC - function pointer to sector erase function.
- NVM_WRITE_FUNC - function pointer to write function.

- NVM_READ_FUNC - function pointer to read function.
- Display Driver Layer architecture is changed. Refer to Adding New Device Driver for new requirements.
- Modified Resistive Touch Driver calibration
- In the PIC24FJ256DA210 driver, CopyWindow() is modified to CopyBlock().

Fixes:

- Fixed issue on vertical Progress Bar rendering.
- Updated demos Google map and JPEG to use the proper GFX_RESOURCE identifiers for JPEG resources
- HX8347 driver now compiles and works with 'mikroMMB for PIC24'
- Bug fixes in the digital meet and cross hair widgets
- Removed references to the PIC24 configuration bit COE_OFF.
- Fixed issue on PutImage() when using PIC24FJ256DA210 and look up table is used on images located at internal or external SPI flash with color depth less than 8bpp.
- Fixed issue on Line() in the SSD1926 and mchpGfxDrv driver files. The stored coordinates after a successful rendering of a line will be at the end point (x2,y2).

Deprecated Items:

- TYPE_MEMORY - replaced by GFX_RESOURCE
- EXTDATA - replaced by GFX_EXTDATA
- BITMAP_FLASH - replaced by IMAGE_FLASH
- BITMAP_RAM - replaced by IMAGE_RAM
- BITMAP_EXTERNAL - replaced by GFX_EXTDATA

Migration Changes:

- DisplayDriver.c is not used to select the display driver (the file is not a part of the graphics library release package).
- The application should include the driver(s) in the project. Multiple driver files can be included in one project because the hardware profile will define the driver and only that driver's source code will be used.
- For example, a project may be designed to use the Microchip's PIC24FJ256DA210 graphics controller and the SSD1926 depending on the hardware profile used. The project will include the following source files, SSD1926.c and mchpGfxDrv.c, among the graphics source files. The hardware profile will contain the following macros, GFX_USE_DISPLAY_CONTROLLER_SSD1926 and GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210, to select the SSD1926 and Microchip's PIC24FJ256DA210 graphics controller, respectively.
- When using PIC24FJ256DA210, the driver files are now changed to:
- mchpGfxDrv.c - source code
- mchpGfxDrv.h - header file
- mchpGfxDrvBuffer.c - source code that declares display buffer, work areas and cache areas for IPU operations in EDS.
- The touch screen drivers in the "Board Support Package" directory are renamed to:
- TouchScreenResistive.c - internal resistive touch source code
- TouchScreenResistive.h - internal resistive touch header file
- TouchScreenAR1020.c - external AR1020 touch source code
- TouchScreenAR1020.h - external AR1020 touch header file
- The two original files (TouchScreen.c and TouchScreen.h) are still needed since they are defining common interfaces to resistive touch drivers.
- When using the internal resistive touch module, the project will contain TouchScreenResistive.c and TouchScreen.c. All modules that reference touch screen APIs will include TouchScreen.h header file.
- When using the external AR1020 touch module, the project will contain TouchScreenAR1020.c and TouchScreen.c. All modules that reference touch screen APIs will include TouchScreen.h header file.
- The touch screen initialization routine API has changed. The new TouchInit API has four parameters passed. Please refer to the API definition for more details.

- When using the Potentiometer on the graphics development boards, include in your project the following files found in the "Board Support Package" directory:
- TouchScreenResistive.c - source code
- TouchScreenResistive.h - header file
- Potentiometer.h - contains the APIs for the A/D interface.
- EEPROM.h and EEPROM.c are going to be deprecated. Use the two new files:
- MCHP25LC256.c - source code
- MCHP25LC256.h - header file
- in the HardwareProfile.h add #define USE_MCHP25LC256 to use the new driver.
- A SPI driver has been created to support projects with multiple devices on one SPI channel.
- Projects will need to include the source file drv_spi.c in projects that use devices on a SPI channel.
- SPI Flash initialization routines will need to pass a DRV_SPI_INIT_DATA structure. This structure defines the SPI control and bit rate used by the SPI Flash module.
- The COLOR_DEPTH macro has been moved from the hardware profile header file to the GraphicsConfig.h header file.
- For project migration please refer the graphics demos for examples.

Known Issues:

- PutImage() does not work when using PIC24FJ256DA210 and look up table is used on images located at EDS memory with color depth less than 8bpp.
- PutImage() of when using PIC24FJ256DA210 for 8bpp images is missing the last row and last column of the bitmap when the image is from external memory, look up table is used and the screen is rotated 90 degrees.
- When compiling the Analog Clock source code with C30 v3.24 the optimization setting must be set to 0 (none).
- External Memory Programmer utility does not work with Graphics PICTail v2 (AC164127)
- When using PIC24FJ256GB210 PIM with Explorer 15 board with a 5v Lumex LCD display, the S1D13517 demo does not run correctly.
- Font tables are limited to 256 pixel character height. For fonts generated for external memory, the maximum height limitation is 128 pixels.

v2.11

New:

- Graphics Resource Converter (GRC) ported to java version.
- Added support for Inflate Processing Unit (IPU) and Character Processing Unit (CHRGPU)of the Microchip Graphics Module implemented in PIC24FJ256DA210.
- Added new "Google Map Demo" for PIC32MX795F512L and PIC24FJ256DA210 device.
- Added SST39LF400 Parallel Flash Memory driver in "Board Support Package". This is the driver for the parallel flash on the "PIC24FJ256DA210 Development Board".
- Added demo support for PIC32 MultiMedia Expansion Board (DM320005).
- Added GFX_IMAGE_HEADER structure. This structure defines how the image(s) are accessed and processed by the Graphics Library.
- Added a third option (#define XCHAR unsigned char) on the XCHAR usage. This is provided as an option to use characters with IDs above 127 and below 256. With this option, European fonts that uses characters with character IDs above 127 and below 256 can now be generated and used in the library.
- Added a scheme to replace the default font GOLFontDefault in the library with any user defined fonts. Refer to "Changing the default Font" for details.

Changes:

- Added compile switches to all drivers in "Board Support Package" for options to compile out when not used in specific projects.
- Replaced TYPE_MEMORY with GFX_RESOURCE type enumeration and expanded the enumeration for graphics resources (such as images and fonts). GFX_RESOURCE type will determine the source and the data format of the resource (compressed or uncompressed).

- Changes on the macros used on the "Graphics JPEG Demo":

```
// Valid values of the first field for JPEG_FLASH and JPEG_EXTERNAL structures
#define FILE_JPEG_FLASH 2 // the JPEG file is located in internal flash
#define FILE_JPEG_EXTERNAL 3 // the JPEG file is located in external memory
```

to

```
// Valid values of the first field for JPEG_FLASH and JPEG_EXTERNAL structures
#define FILE_JPEG_FLASH 0 // the JPEG file is located in internal flash
#define FILE_JPEG_EXTERNAL 1 // the JPEG file is located in external memory
```

- Added function pointers to GOL structure OBJ_HEADER. These function pointers makes it easier to add user created objects in the Graphics Library.
- DRAW_FUNC - function pointer to object drawing function.
- FREE_FUNC - function pointer to object free function. Only for objects that needs free routines to effectively free up memory used by the object create function.
- MSG_FUNC - function pointer to object message function.
- MSG_DEFAULT_FUNC - function pointer to object default message function.
- Merged "Graphics External Memory Demo" and "Graphics External Memory Programmer" into one demo "Graphics External Memory Programmer and Demo".

Fixes:

- TouchScreen driver now checks display orientation and adjusts the touch to be aligned to the display orientation.
- Fixed GOLFocusNext() issue on list that does not contain an object that can be focused.
- Removed redundant code in GOLRedrawRec().
- Added an option in XCHAR to use unsigned char.

Deprecated Items:

- TYPE_MEMORY - replaced by GFX_RESOURCE
- EXTDATA - replaced by GFX_EXTDATA
- BITMAP_FLASH - replaced by IMAGE_FLASH
- BITMAP_RAM - replaced by IMAGE_RAM
- BITMAP_EXTERNAL - replaced by GFX_EXTDATA

Migration Changes:

- To use drivers located in "Board Support Package" directory, add the USE_DRIVERNAME macro in application code (in the demos these are added in the HardwareProfile.h) to include the drivers. Refer to the specific driver code for the correct USE_DRIVERNAME macro name.
- The new version of the Graphics Resource Converter generates graphics application resources (fonts and images) using the new GFX_IMAGE_HEADER structure for images and new GFX_RESOURCE type defines to specify location of the resources. Because of this, some structures are deprecated and replaced with more appropriate structures. To remove the deprecation warnings, regenerate the fonts and images files using the new Graphics Resource Converter.

Known Issues:

- Graphics SSD1926 JPEG and SD Card Demo does not support Graphics Display Powertip 4.3" 480x272 Board (PH480272T_005_I11Q). As is, there's not enough spare memory space to carry out the hardware JPEG decoding operation by the SSD1926. A potential work around is to reduce the active display area size to reserve more memory space for the JPEG decoding operation.
- SSD1926 hardware acceleration for eclipse is disabled due to missing pixels at Angle 0.
- PIC32MX460 PIM (not Starter Kit) does not support 16-bit PMP mode with Graphics PICtail Plus Board Version 3 (SSD1926) Board. It only supports 8-bit PMP mode. This is due to pin mapping conflicts on the boards.
- This version of Graphics Library is not compatible with Graphics Display Designer v2.0.0.9c

v2.10

New:

- Added new demo "Graphics Object Layer Palette Demo" for PIC24FJ256DA210 device.

- Added support for PIC32MX795F512L device.
- Added documentation for the Grid object.
- Added Vertical Mode to Progress Bar.
- Added MicrochipGraphicsModule display driver.
- Added "Board Support Package" directory. This contains common hardware drivers for Microchip demo boards.
- Added OBJ_MSG_PASSIVE as a translated message on the slider to detect a touch screen release message. This has no effect on the state of the slider. Applications that does not qualify for touch press and touch move event must now qualify the messages for the slider object to avoid processing messages for touch release.

Changes:

- To improve speed modified gfxpmp.c and gfxepmp.c to be inline functions in gfxpmp.h and gfxepmp.h respectively.
- Changed HX8347A.c to HX8347.c (both the D and A driver version is implemented in the new file). To select set the DISPLAY_CONTROLLER to be HX8347A or HX8347D in the hardware profile.
- Modified malloc() and free() to be defined by macros in GraphicsConfig.h file. For applications using Operating System, the macros can be redefined to match the OS malloc and free functions. The default settings are:
- #define GFX_malloc(size) malloc(size)
- #define GFX_free(ptr) free(ptr)
- Merged GOL Demo English and Chinese demo into one demo.
- Removed the macro "GRAPHICS_HARDWARE_PLATFORM". This is specific to Microchip demo boards.
- Abstracted the timer from the touch screen driver.
- Moved the following hardware drivers to the "Board Support Package" directory
- Touch screen driver: TouchScreen.c and TouchScreen.h files.
- SPI Flash driver: SST25VF016.c and SST25VF016.h files.
- Graphics PICtail Version 2 Parallel Flash driver: SST39VF040.c and SST39VF040.h files.
- Explorer 16 SPI EEPROM Flash driver: EEPROM.c and EEPROM.h files.
- Graphics PICtail Version 2 Beeper driver: Beep.c and Beep.h files.
- Revised the Seiko 3.5" 320x240 display panel schematic to revision B. Corrected the pin numbering on the hirose connector. See "Schematic for Graphics Display Seiko 3.5in 320x240 Board Rev B.pdf" file on the /Microchip Solutions/Microchip/Graphics/Documents/Schematics directory.

Fixes:

- Fixed TextEntry object issue on output string maximum length.
- Fixed Slider increment/decrement issue on Keyboard messages.
- Fixed GOLGetFocusNext() bug when none of the objects in the list can be focused.

Migration Changes:

- pmp interface files are converted to header files and functions are now inline functions to speed up pmp operations. Projects must be modified to:
- include gfxpmp.h and gfxepmp.h source files in the project.
- gfxepmp.c file is retained but will only contain the definition of the EPMP pmp_data.
- Converted the macro: #define GRAPHICS_HARDWARE_PLATFORM HARDWARE_PLATFORM where HARDWARE_PLATFORM is one of the supported hardware platforms defined in the section Graphics Hardware Platform to just simply #define HARDWARE_PLATFORM.
- Since the timer module is abstracted from the touch screen driver in the "Board Support Package", the timer or the module that calls for the sampling of the touch screen must be implemented in the application code. Call the function TouchProcessTouch() to sample the touch screen driver if the user has touched the touch screen or not. For example:

```
// to indicate the hardware platform used is the
// Graphics PICtail  Plus Board Version 3
#define GFX_PICTAIL_V3
```

- Projects which uses the following hardware drivers will need to use the latest version of the drivers located in the "Board Support Package" directory.

- Touch screen driver: TouchScreen.c and TouchScreen.h files.
- SPI Flash driver: SST25VF016.c and SST25VF016.h files.
- Graphics PICtail Version 2 Parallel Flash driver: SST39VF040.c and SST39VF040.h files.
- Explorer 16 SPI EEPROM Flash driver: EEPROM.c and EEPROM.h files.
- Graphics PICtail Version 2 Beeper driver: Beep.c and Beep.h files.
- In the TouchScreen driver, the timer initialization and timer interrupt sub-routine (ISR) are abstracted out of the driver. The initialization and the ISR should be defined in the application code. the TouchProcessTouch() function in the driver should be called in the ISR to process the touch.

Known Issues:

- Graphics SSD1926 JPEG and SD Card Demo does not support Graphics Display Powertip 4.3" 480x272 Board (PH480272T_005_I11Q). As is, there's not enough spare memory space to carry out the hardware JPEG decoding operation by the SSD1926. A potential work around is to reduce the active display area size to reserve more memory space for the JPEG decoding operation.
- SSD1926 hardware acceleration for eclipse is disabled due to missing pixels at Angle 0.
- PIC32MX460 PIM (not Starter Kit) does not support 16-bit PMP mode with Graphics PICtail Plus Board Version 3 (SSD1926) Board. It only supports 8-bit PMP mode. This is due to pin mapping conflicts on the boards.
- This version of Graphics Library is not compatible with Graphics Display Designer v2.0.0.9c

v2.01

Changes:

- Modified drivers for abstraction of pmp and epmp interfaces. they have a common header file DisplayDriverInterface.h.
- DisplayDriverInterface.h is added to the Graphics.h file.
- DelayMs() API is abstracted from the driver files. TimeDelay.c and TimeDelay.h is added.

Fixes:

- Fixed background color bug in StaticText and Digital Meter object.
- Fixed ListBox LbSetFocusedItem() bug on empty lists.
- Graphics SSD1926 JPEG and SD Card Demo is fixed to support SD card of size 2GB or bigger

Migration Changes:

- pmp interface is abstracted from the driver. Projects must be modified to:
- include gfxpmp.c and gfxepmp.c source files in the project.
- DelayMs() is abstracted from the drivers.
- Add TimeDelay.c source file in the project.
- Add TimeDelay.h header file in the project.

v2.00

Changes:

- "Graphics PICtail Board Memory Programmer" has been renamed to "Graphics External Memory Programmer".
- "Bitmap & Font Converter" utility has been renamed to "Graphics Resource Converter".
- Font format has changed. The bit order has been reversed. Necessary for cross compatibility.
- Added 2 new directories in each demo
- Precompiled Demos - this directory contains all pre-compiled demos for all hardware and PIC devices supported by the demo.
- Alternative Configurations - this directory contains all the Hardware Profiles for all hardware and PIC devices supported by the demo.
- Moved all hardware and display parameters from GraphicsConfig.h to HardwareProfile.h.HardwareProfile.h references a hardware profile file in "Alternative Configurations" directory based on the PIC device selected.

Fixes:

- Fixed BtnSetText() bug when using Multi-Line Text in Buttons.

- Fixed SDSectorWrite() function in SSD1926_SDCard.c in the "Graphics SSD1926 JPEG and SD Card Demo".

Migration Changes:

- Move all hardware and display parameters from GraphicsConfig.h to HardwareProfile.h
- panel type, display controller, vertical and horizontal resolution, front and back porches, synchronization signal timing and polarity settings etc.
- GRAPHICS_PICTAIL_VERSION,1,2,250,3 options are now deprecated, new usages are:
- #define GRAPHICS_HARDWARE_PLATFORM GFX_PICTAIL_V1
- #define GRAPHICS_HARDWARE_PLATFORM GFX_PICTAIL_V2
- #define GRAPHICS_HARDWARE_PLATFORM GFX_PICTAIL_V3
- The font format has changed, run Graphics Resource Converter to regenerate font files, the bit order is reversed. No legacy support is provided. Primitive/Driver layers now expects the new format.
- The initialization sequence of GOLInit() relative to the flash memory initialization is sensitive due to the sharing of hardware resources, i.e. SPI or PMP. Care should be taken to make sure the peripheral and I/O port settings are correct when accessing different devices.
- A number of configuration options have been moved from GraphicsConfig.h to HardwareProfile.h, this is required to maintain a more logical flow.
- HardwareProfile.h now points to one of many Alternative Configuration files, each one specific to a certain hardware board setup.
- DelayMs routine in SH1101A-SSD1303.c/h is now a private function, no public API is exposed. In future releases, DelayMS will be removed from all drivers and be replaced by an independent module.
- GenericTypeDefs.h has been updated with new definitions, this should not impact any legacy codes.

v1.75b

Changes:

- None.

Fixes:

- Fixed Line2D() bug in SSD1926.c.
- Fixed remainder error in JPEG decoding in JpegDecoder.c.
- Fixed pinout labels for reference design schematic "Schematic for Graphics Display Powertip 4.3in 480x272 Board Rev 2.pdf".

Migration Changes:

- None.

v1.75 Release (July 10, 2009)

Changes:

- Added 2D acceleration support for controllers with accelerated primitive drawing functions.
- Added Digital Meter Widget for fast display refresh using fixed width fonts.
- Added support for selected PIC24H Family of devices.
- Added support for selected dsPIC33 Family of devices.
- Updated all primitive functions to return success or fail status when executed. This is used to evaluate if the primitive function has finished rendering.
- Updated Solomon Systech SSD1926 driver to use 2D accelerated functions.
- New Display Controller Driver supported:
- Ilitek ILI9320
- Solomon Systech SSD1289
- Himax HX8347
- Renesas R61580
- New demos are added:

- PIC24F Starter Kit Demo
- PIC24H Starter Kit Demo 1
- Graphics JPEG Demo using internal and external flash memory for image storage
- Graphics SSD1926 JPEG and SD Card Demo using SD Card for image storage
- Added JPEG support to "Font and Bitmap Converter Utility".
- Modified Button Widget for new options:
- Use multi-line text (set USE_BUTTON_MULTI_LINE)
- Detect continuous touch screen press event using messaging
- Added Touch Screen event EVENT_STILLPRESS to support continuous press detection.
- New reference design schematics added:
- Schematic for Graphics Display DisplayTech 3.2in 240x320 Board.pdf
- Schematic for Graphics Display Newhaven 2.2in 240x320 with HX8347.pdf
- Schematic for Graphics Display Seiko 3.5in 320x240 Board.pdf
- Schematic for Graphics Displays DisplayTech and Truly 3.2in 240x320 with SSD1289.pdf
- Schematic for ILI9320.pdf

Fixes:

- Fixed dimension calculation for quarter sized meter.

Migration Changes:

- When using accelerated primitive functions while USE_NONBLOCKING_CONFIG is enabled, the accelerated primitive must be checked if it succesfully rendered. Refer to the coding example for details.
- Replaced LGDP4531_R61505_S6D0129_S6D0139_SPFD5408.c and LGDP4531_R61505_S6D0129_S6D0139_SPFD5408.h files with drvTFT001.c and drvTFT001.h respectively.

v1.65 Release (March 13, 2009)

Changes:

- Added support for the new Graphics PICtail Plus Daughter Board (AC164127-3). This new board comes in two components: the controller board and the display board. The display board uses RGB type displays driven by the controller board. This configuration allows easy replacement of the display glass.
- Added application note AN1246 "How to Create Widgets".
- New Display Controller Driver supported
- UltraChip UC1610
- New demos are added
- Graphics AN1246 Demo showing the TextEntry Widget.
- Graphics Multi-App Demo showing USB HID, USB MSD and SD MSD demos using the Microchip Graphics Library.
- Modified Meter Widget for new options:
- Set Title and Value Font at creation time
- Added GOLGetFocusPrev() for focus control on GOL Objects.
- Added work spaces to demo releases.
- Graphics PICtail Plus Board 1 is obsolete. All references to this board is removed from documentation.
- New reference design schematics added:
- Schematic for Graphics Display Ampire 5.7in 320x240 Board Rev A.pdf
- Schematic for Graphics Display Powertip 3.5in 320x240 Board Rev B.pdf
- Schematic for Graphics Display Powertip 4.3in 480x272 Board Rev B.pdf
- Schematic for Graphics Display Truly 3.5in 320x240 Board Rev A.pdf
- Schematic for Truly TOD9M0043.pdf

Fixes:

- Fixed drawing bug on TextEntry Widget
- Added missing documentation on Chart and Text Entry Widgets

Migration Changes:

- none

v1.60 Release (December 3, 2008)

Changes: Graphics Library Help Release Notes Graphics Library Version 3.06.04 19

- Added TextEntry Widget.
- Modified Meter Widget for new options:
- Define different fonts for value and title displayed.
- Add option for resolution of one decimal point when displaying values.
- Add color options to all six arcs of the Meter.
- Meter range is not defined by a minimum value and a maximum value.
- Added feature to a the Button Widget to allow cancelling of press by moving away the touch and releasing from the Button's face.
- Added font sizes options of 3,4,5,6 & 7 in Font & Bitmap Converter Utility when converting fonts from TTF files.
- Enhanced the architecture of the Display Device Driver Layer.

Fixes:

- Fixed Font & Bitmap Converter Utility generation of reference strings to be set to const section.
- Fixed panel rendering to always draw the panel face color even if bitmaps are present.

Migration Changes:

- Added the following files in the Display Device Driver Layer to easily switch from one display driver to another.
- DisplayDriver.h
- DisplayDriver.c
- Modified implementation of GraphicsConfig.h file to support new Display Device Driver Layer architecture.
- Moved the definitions of pins used in device drivers implemented in all the demos to HardwareProfile.h file.
- EEPROM Driver
- Touch Screen Driver
- Beeper Driver
- Flash Memory Driver
- Display Drivers
- Modified GOL.c and GOL.h to include processing of TextEntry object when enabled by application.

v1.52 Release (August 29, 2008)

Changes:

- Added Chart Widget.
- Added Property State for Window Widget. Text in Title Area can now be centered.
- Added Supplementary Library for Image Decoders.
- Added documentation of Default Actions on widgets.
- Replaced USE_MONOCHROME compile switch with COLOR_DEPTH to define color depth of the display.
- Added GOL_EMBOSS_SIZE to be user defined in GraphicsConfig.h.
- Simplified initialization code for SSD1906 driver.

Fixes:

- Fixed touch screen algorithm.

- Fixed file path error in Font & Bitmap Converter Utility.

Migration Changes:

- USE_GOL must be defined when using any Widgets.
- New include directory paths are added to demo projects:
- ../../../Your Project Directory Name
- ../../Include
- Moved all driver files to new directory
- C files from ../Microchip/Graphics to ../Microchip/Graphics/Drivers
- GOL_EMBOSS_SIZE can now be defined by the user in GraphicsConfig.h. If user does not define this in GraphicsConfig.h the default value in GOL.h is used.

v1.4 Release (April 18, 2008)

Changes:

- Added full support for PIC32 families.
- Added Application Note demo on fonts.
- Added images for end designs using PIC devices.

Fixes:

- Fixed GetPixel error in SSD1906 Driver.
- Fixed SST39VF040 parallel flash driver reading instability.
- Fixed error in 64Kbytes rollover in utility conversion of bitmaps and SST39VF040 parallel flash driver.
- Fixed milli-second delay on PIC32.
- Fixed compile time option errors for PIC32.
- Fixed Graphics Object Layer Demo error in PIC32 when time and date are set.
- Fixed Picture widget bug in detecting touchscreen in translating messages.

v1.3 Release (March 07, 2008)

Changes:

- none

Fixes:

- Fixed an inaccurate ADC reading problem with some Explorer 16 Development Boards that uses 5V LCD display.
- Editbox allocation of memory for text is corrected.
- Fix slider SetRange() bug.
- Set PIC32 configuration bits related to PLL to correct values.
- Touch screen portrait mode bug is fixed.

v1.2 Release (February 15, 2008)

Changes:

- Added support for Graphics PICtail Plus Board Version 2
- Added support for foreign language fonts
- Version 1.2 of the font and bitmap utility
- Support for multi-language scripts
- Support for generating font table from installed fonts
- Support to reduce font table size by removing unused characters
- Support to select between C30 and C32 compiler when generating bitmaps and font tables.
- Added Chinese version of Graphics Object Layer Demo.
- New Display Controller Drivers supported

- Solomon Systech SSD1906
- Orise Technology SPDF5408
- Replaced USE_UNICODE compile switch to USE_MULTIBYTECHAR compile switch to define XCHAR as 2-byte character.
- Added compile switches
- GRAPHICS_PICTAIL_VERSION - sets the PICtail board version being used.
- USE_MONOCHROME to enable monochrome mode.

Graphics Library Help Release Notes Graphics Library Version 3.06.04 21

- USE_PORTRAIT - to enable the portrait mode of the display without changing the display driver files.
- Added beta support for PIC32 device

Fixes:

- Specification changes to List Box widget. Bitmap is added to List Box items.
- Fixed List Box LbDelItemsList() error in not resetting item pointer to NULL when items are removed.
- Editbox allocation of memory for text is corrected.
- Static Text multi-byte character failure is fixed.
- Bar() function erroneous call to MoveTo() is removed since it causes the drawing cursor to be displaced.
- Removed SCREEN_HOR_SIZE and SCREEN_VER_SIZE macros from documentation. Maximum X and Y sizes are to be obtained by GetMaxX() and GetMaxY() macros.

v1.0 Release (November 1, 2007)

Changes:

- Edit Box, List Box, Meter, Dial widgets are added.
- Button is modified. New options for the object are added.
- Modified OBJ_REMOVE definition to OBJ_HIDE (example BTN_REMOVE to BTN_HIDE).
- Modified GOLPanelDraw() function to include rounded panels.
- External memory support for the fonts and bitmaps is implemented.
- SSD1339 and LGDP4531 controllers support is added.
- Bevel(), FillBevel() and Arc() functions are added.
- Modified Graphics Object Layer Demo.
- Added Graphics External Memory Demo & Graphics PICtailTM Board Memory Programmer Demo.
- Added Graphics Application Note (AN1136- How to use widgets.) Demo.

Fixes: none

v0.93 Beta release (August 29, 2007)

Changes: none Fixes:

- In demo code the bitmap images couldn't be compiled without optimization. bmp2c.exe utility output is changed to fix this bug.
- In demo code "volatile" is added for global variables used in ISRs.

v0.92 Beta release (July 25, 2007)

Changes:

- Keyboard and side buttons support is added.
- Keyboard focus support is added.
- PutImage() parameters are changed. Instead of pointer to the bitmap image the pointer to BITMAP_FLASH structure must be passed.
- GOLSuspend() and GOLResume() are removed.
- GOLMsg() doesn't check object drawing status. It should be called if GOL drawing is completed.

- GOLStartNewList() is replaced with GOLNewList().
- Line() function calls are replaced with Bar() function calls for vertical and horizontal lines.
- Parameter "change" is removed for SldIncVal() and SldDecVal() .
- Some optimization and cleanup.
- Slider API is changed:
- SldSetVal() is changed to SldSetPos()
- SldGetVal() is changed to SldGetPos()
- SldIncVal() is changed to SldIncPos()
- SldDecVal() is changed to SldDecPos()
- SldCreate() input parameter are changed:
- "delta" changed to "res"

Fixes:

- PutImage().
- Line().
- FillCircle().
- For vertical slider the relation between thumb location and slider position is changed. For position = 0 thumb will be located at the bottom. For position = range it will be at the top.

v0.9 Beta release (July 06, 2007)

Changes:

- Background color support is removed.
- Non-blocking configuration for graphics primitives is added.
- GetImageWidth(), GetImageHeight() are added.
- The following functions are terminated by control characters (< 32):
- OutText()
- OutTextXY()
- GetTextWidth()
- Graphics Objects Layer (GOL) is added.
- Button, Slider, Checkbox, Radio Button, Static Text, Picture control, Progress Bar, Window, Group Box are implemented.
- Touch screen support is added.

Fixes:

- ReadFlashByte().
- GetTextWidth().

v0.1 (June 05, 2007)

Changes:

- Initial release includes driver and graphic primitive layers only.
- Only driver for Samsung S6D0129 controller is available.

Fixes:

- None.

Known Issues

1.4 Using The Library

This topic describes the architecture of the Graphics Library and the components comprising each layers. Information on how to use the Primitive Layer and Object Layer is also presented.

1.4.1 Library Overview

This section describes the architecture of the library and its basic components.

Description

The Graphics Library is composed of the following layers:

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

1.4.1.1 Graphics Objects

This section describes the basics of the Graphics Objects.

Description

The Graphics Object Layer is composed of objects that can act as control widgets in an application. There are several pre-defined 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 an application note mentioned at the Release Notes section of this documentation.

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 function is optional. This 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 GFX_GOL_TRANSLATED_ACTION enumeration for the listing of all translated actions of all objects in the library. This list do 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 function is optional. This 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. Application should instead use the GFX_GOL_ObjectListDraw() so the management of the object rendering will be done by the Graphics Object Layer.

1.4.1.2 Object Layer Rendering

This section describes how objects are rendered.

Description

Object List Rendering

Initially, 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 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 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 will be discussed in the messaging portion of this documentation.

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

- Property State Bits - this defines the general shape of the object.
- Draw State Bits - this determines 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. The Draw state bits, on the other hand dynamically changes and usually changes 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 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 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.

```
//************************************************************************
//  Main
//************************************************************************
int main(void)
{
    // GOL message structure
    GFX_GOL_MESSAGE msg;

    //Initialize board, drivers and graphics library
    SYSTEM_InitializeBoard();
    GFX_Initialize();

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

    // set the draw callback function pointer
    GFX_GOL_DrawCallbackSet(APP_ObjectDrawCallback);
```

```
    ....

    while(1)
    {
        // Draw GOL objects
        if(GFX_GOL_ObjectListDraw() == GFX_STATUS_SUCCESS)
        {

            // Get message from touch screen
            TouchGetMsg(&msg);
            // Process message
            GFX_GOL_ObjectMessage(&msg);

        }
    }

}

//*****************************************************************
//  The draw callback is called within GFX_GOL_ObjectListDraw()
//*****************************************************************
bool APP_ObjectDrawCallback(void)
{
    // assume pScrollBar is pointer to the scrollbar

    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 pre-defined 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);
    }
```

1.4.1.3 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 to be 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 let's go of the press, 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, changes 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 example code below 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);
...

while(1)
{
    // Draw GOL objects
    if(GFX_GOL_ObjectListDraw() == GFX_STATUS_SUCCESS)
    {

        // Get message from touch screen
        TouchGetMsg(&msg);
        // Process message
        GFX_GOL_ObjectMessage(&msg);

    }
}
```

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 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 documentation of the callback function 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.

Below is an example 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 2 buttons and a scroll bar is 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 do 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 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.

The 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 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().

1.4.2 How the Library Works

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

1.4.2.1 Using the Primitive Layer

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

1.4.2.1.1 Line Rendering

This section describes how to render lines.

Description

When rendering lines, two things needs to be set:

1. Set the color (GFX_ColorSet())
2. 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 has 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.

Remarks

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

1.4.2.1.2 Polygon Rendering

This section describes how to render polygons.

1.4.2.1.2.1 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 needs to be set:

1. Set the color (GFX_ColorSet())
2. Set the line style (GFX_LineStyleSet())
3. Call the specific polygon function

1.4.2.1.2.2 Filled Polygon Rendering

This section describes how to render filled polygons.

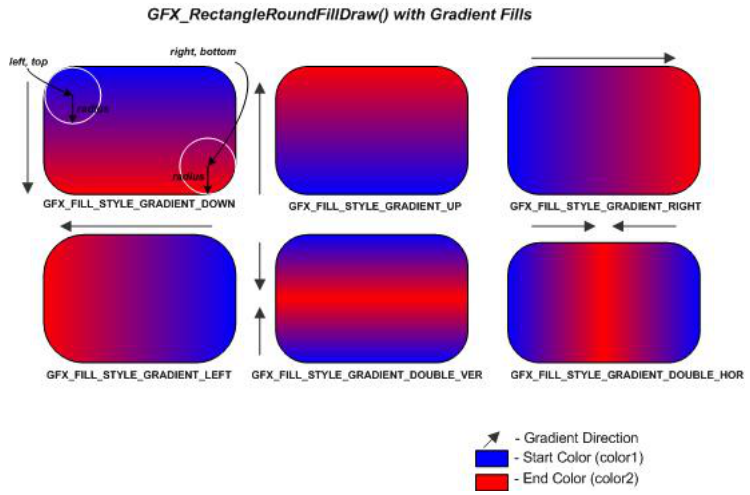
Description

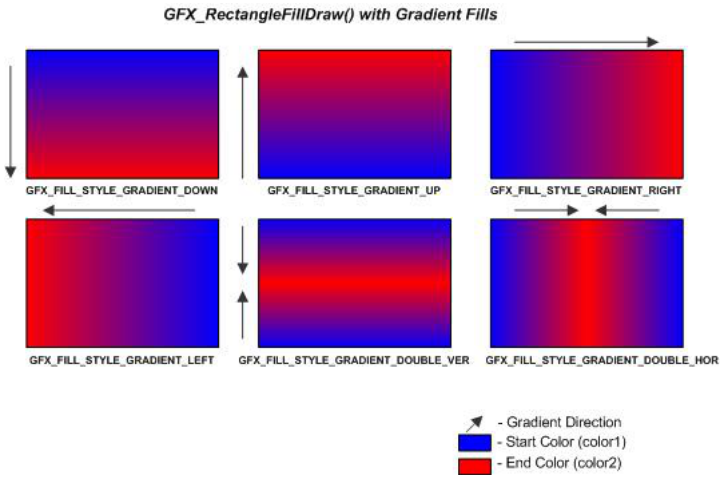
Filled polygons are filled rectangles and filled rounded rectangles. Bar is a special case of a filled rectangle and filled circle is a special case of a filled rounded rectangle.

For filled polygons, the fill styles can be set:

Fill Style	Description
GFX_FILL_STYLE_NONE	The effect of this fill type is the same as unfilled polygon.
GFX_FILL_STYLE_ALPHA_COLOR	The fill will be the color set by the last call to GFX_ColorSet() function.
GFX_FILL_STYLE_GRADIENT_DOWN	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_UP	The effect is a gradient fill in the vertical up direction.
GFX_FILL_STYLE_GRADIENT_UP	The effect is a gradient fill in the vertical down 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).

Remarks

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

1.4.2.1.3 Text Rendering and Font Features

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

1.4.2.1.3.1 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 this to render a single character.
- GFX_TextStringDraw() - Use this to render a string of characters.
- GFX_TextStringBoxDraw() - Use this 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

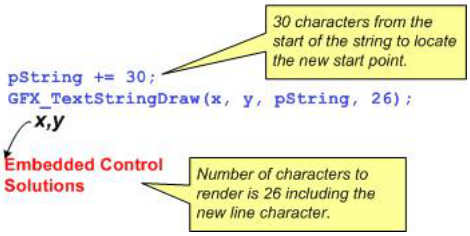
To render a string:

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);
```



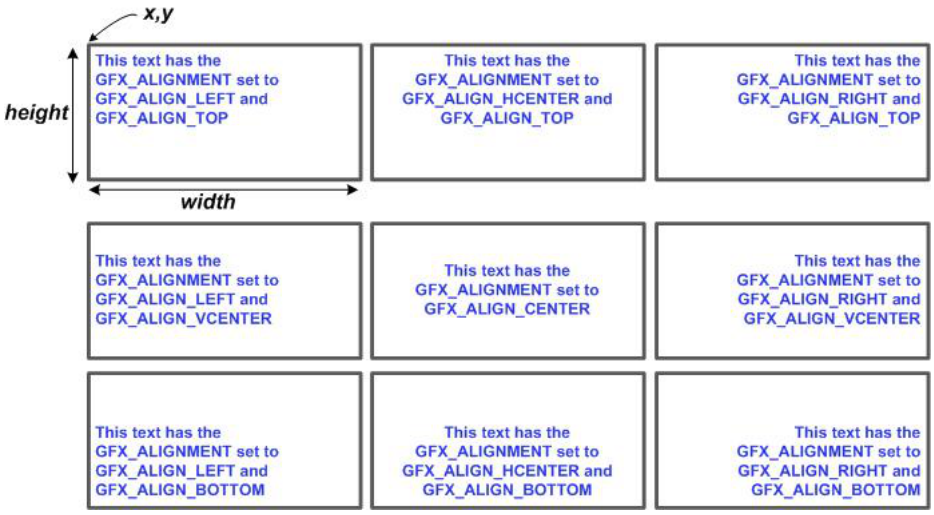
Multiple lines of strings are also supported. The string can contain the new line characters and rendering results in multiple lines of text. The use GFX_TextStringDraw() will always render strings left aligned. To render text with alignment use GFX_TextStringBoxDraw().

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 this to get the length of a string in pixels.

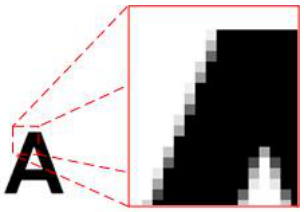
- GFX_TextStringHeightGet() - Use this to get the height of the given font. All characters in a given font resource will have a constant height.
- GFX_FontSet() - Use this to set the font resource to use.
- GFX_FontGet() - Use this to check the currently used font resource.

1.4.2.1.3.2 Anti-alised Fonts

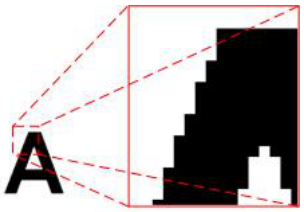
This section describes the anti-aliasing feature of the library.

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 can't 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 figure below.



When anti-aliasing is turned off, the pixels abruptly changes from background color to foreground color shown in the figure below. 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.

Remarks

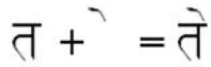
- Application may use anti-aliased and non-anti-aliased fonts at the same time.

1.4.2.1.3.3 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 set, more than one glyph overlaps each other to form a single character of that language as shown in the figure below.



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

Remarks

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

1.4.2.2 Using the Graphics Object Layer

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

1.4.2.2.1 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 the objects in the library utilizes the style scheme to define how the object will 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 so 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):

1. Text - The parameter that defines the text used in the object
- pFont - defines the font used in the style of the object.
2. 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.
3. 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.

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

- fillStyle - specified 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.

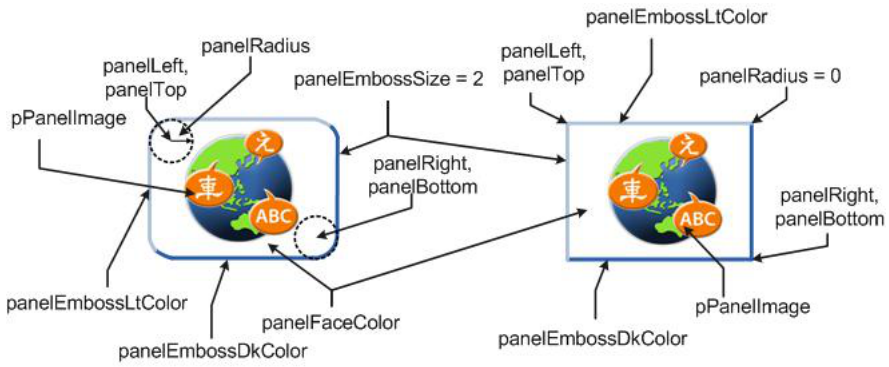
5. Style - The parameter that defines the 3-D effect of the object.

- 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 below:

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.
--	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 texts.

See the section on panel rendering 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 that 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.

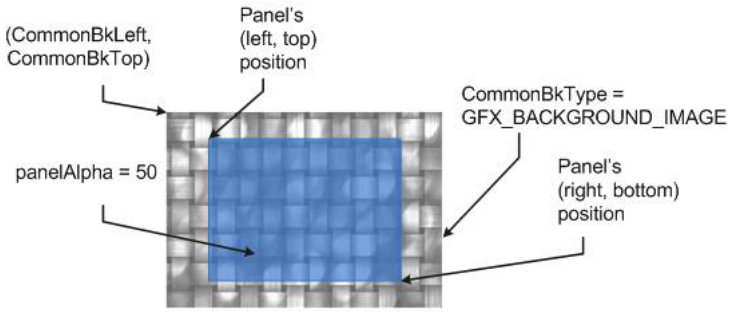
The requirement of the background feature is that the background should encompass the object. In other words, all the pixels of the object should be within the background dimension. Once this requirement is fulfilled, the parameters of the object that defines the location of the object (left, top, right 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:

1. CommonBkLeft - The left most pixel location of the background.
2. CommonBkTop - The top most pixel location of the background.
3. CommonBkType - Should be set to one of the GFX_BACKGROUND_TYPE.
4. 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 re-drawing 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:

1. fillStyle - Set to GFX_FILL_STYLE_ALPHA_COLOR.
2. 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.

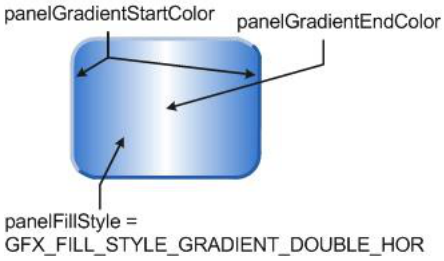
To implement an gradient filled object the following style scheme parameters should be set:

1. 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

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



Gradient Filled Panel

Remarks

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

1.5 Configuring the Library

This section describes how the library can be configured.

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.

1.5.1 Configuration Options

This section summarizes all the available configuration options of the Graphics Library.

Macros

Name	Description
GFX_CONFIG_ALPHABLEND_DISABLE	Macro that disables the Alpha Blending feature.
GFX_CONFIG_BISTABLE_DISPLAY_AUTO_REFRESH_ENABLE	Macro enables the support for auto refresh in bi-state display.
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_PROG_SPACE_ENABLE	Macro enables the relocation of fonts in the program space.
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_NONBLOCKING_DISABLE	Blocking and Non-Blocking configuration selection.
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.

Description

The following configuration options are available to the use the Graphics Library.

1.5.1.1 GFX_CONFIG_ALPHABLEND_DISABLE Macro

Macro that disables the Alpha Blending feature.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.2 GFX_CONFIG_BISTABLE_DISPLAY_AUTO_REFRESH_ENABLE Macro

Macro enables the support for auto refresh in bi-state display.

File

gfx_config_template.h

Syntax

```
#define GFX_CONFIG_BISTABLE_DISPLAY_AUTO_REFRESH_ENABLE
```

Description

Macro: GFX_CONFIG_BISTABLE_DISPLAY_AUTO_REFRESH_ENABLE

When using bi-state displays, this macro allows for widget auto-update. This feature is only enabled in objects of the Object Layer. The direct calls to primitive rendering routines will need to be manually updated.

To enable this feature, add this macro in the configuration. This macro has no effect in display drivers that do not support bistable auto refresh.

Remarks

None.

1.5.1.3 GFX_CONFIG_COLOR_DEPTH Macro

Macro that sets the color depth of the application.

File

gfx_config_template.h

Syntax

```
#define GFX_CONFIG_COLOR_DEPTH 16
```

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.

1.5.1.4 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

Syntax

```
#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.

1.5.1.5 GFX_CONFIG_FOCUS_DISABLE Macro

Macro that disables the focus feature in objects.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.6 GFX_CONFIG_FONT_ANTIALIASED_DISABLE Macro

Macro that disables the use of anti-aliased fonts.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.7 GFX_CONFIG_FONT_CHAR_SIZE Macro

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

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.8 GFX_CONFIG_FONT_EXTERNAL_DISABLE Macro

Macro that disables sourcing of font resources from external sources.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.9 GFX_CONFIG_FONT_FLASH_DISABLE Macro

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

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.10 GFX_CONFIG_FONT_PROG_SPACE_ENABLE Macro

Macro enables the relocation of fonts in the program space.

File

gfx_config_template.h

Syntax

```
#define GFX_CONFIG_FONT_PROG_SPACE_ENABLE
```

Description

Macro: GFX_CONFIG_FONT_PROG_SPACE_ENABLE

For XC16 builds only: When placing fonts in internal data memory, there is a 32K limit for data space. The total data should not exceed 32K. When this is unavoidable, the macro GFX_CONFIG_FONT_PROG_SPACE_ENABLE will relocate the font in program space. This will remove the 32K restriction but at the expense of slower access.

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

Remarks

None.

1.5.1.11 GFX_CONFIG_FONT_RAM_DISABLE Macro

Macro that disables sourcing of font resources from RAM sources.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.12 GFX_CONFIG_GRADIENT_DISABLE Macro

Macro that disables the gradient fill feature.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.13 GFX_CONFIG_IMAGE_EXTERNAL_DISABLE Macro

Macro that disables sourcing of image resources from external sources.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.14 GFX_CONFIG_IMAGE_FLASH_DISABLE Macro

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

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.15 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

Syntax

```
#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.

1.5.1.16 GFX_CONFIG_IMAGE_RAM_DISABLE Macro

Macro that disables sourcing of image resources from RAM sources.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.17 GFX_CONFIG_IPU_DECODE_DISABLE Macro

Macro that disables RLE compression of images.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.18 GFX_CONFIG_NONBLOCKING_DISABLE Macro

Blocking and Non-Blocking configuration selection.

File

gfx_config_template.h

Syntax

```
#define GFX_CONFIG_NONBLOCKING_DISABLE
```

Description

Macro: GFX_CONFIG_NONBLOCKING_DISABLE

Enabling this macro will disable the non-blocking mode of the library. Non-blocking is enabled to allow

Basic rendering functions such as GFX_LineDraw(), GFX_RectangleDraw(), GFX_BarDraw() etc... are functions implemented in the Primitive Layer. These functions can also be implemented in the device driver layer if the display device supports hardware acceleration of the function. Applications that calls these functions can take advantage of the hardware accelerated primitives. How these functions are used will depend on the configuration setting for the blocking or non-blocking mode.

All primitive rendering functions returns a status.

- GFX_STATUS_FAILURE â€“ when the primitive was not successfully rendered
- GFX_STATUS_SUCCESS â€“ when the primitive was successfully rendered

When using Graphics Library non-blocking mode is by default enabled. If this mode is to be disabled, add this line in the configuration: #define GFX_CONFIG_NONBLOCKING_DISABLE

When using a display controller with hardware accelerated primitives (like SSD1926 which is on the Graphics PICtailâ„¢ Plus Board Version 3 (AC164127-3) faster primitive rendering on Line, Rectangle and Bar functions will be performed. Compiling with the Blocking or Non-Blocking mode set will still use the accelerated primitives but the application code that directly call the primitive functions will have to be coded accordingly.

To explain the two modes when directly calling the primitive functions please take a look at the example below.

Case 1: Non-Blocking disabled

```
// all primitives are blocking calls
GFX_LineDraw(a,b);
GFX_RectangleDraw(c,d,e,f);
GFX_BarDraw(c+2, d+2, e-2, f-2);
```

Case 2: Non-Blocking enabled

```
// all primitives are non-blocking calls
while(GFX_LineDraw(a,b) != GFX_STATUS_SUCCESS);
while(GFX_RectangleDraw(c,d,e,f) != GFX_STATUS_SUCCESS);
while(GFX_BarDraw(c+2, d+2, e-2, f-2) != GFX_STATUS_SUCCESS);
```

If the while check is not in place, it possible that the only primitive that you will see in the screen is the Line. For case 2, one can also be creative in the application code and implement some form of non-blocking scheme and make use of the time while waiting for the primitives to render.

Another example for case 2:

```
GFX_STATUS DrawMyFigure(
            uint16_t a,
            uint16_t b,
            uint16_t c,
            uint16_t d,
            uint16_t e,
            uint16_t f)
{
    typedef enum {
        DRAW_LINE,
        DRAW_RECT,
        DRAW_BAR,
    } DRAW_MYFIGURE_STATES;

    static DRAW_MYFIGURE_STATES state = DRAW_LINE;

    // checks if the hardware is still busy
    if (GFX_RenderStatusGet() == GFX_STATUS_BUSY_BIT)
        return (GFX_STATUS_FAILURE);

    switch(state)
    {
        case DRAW_LINE:
            if (GFX_LineDraw(a, b) != GFX_STATUS_SUCCESS)
                return GFX_STATUS_FAILURE;
            state = DRAW_RECT;
        case DRAW_RECT:
            if(GFX_RectangleDraw(c,d,e,f) != GFX_STATUS_SUCCESS)
                return GFX_STATUS_FAILURE;
            state = DRAW_BAR;
        case DRAW_BAR:
            if(GFX_BarDraw(c+2, d+2, e-2, f-2) !=
                        GFX_STATUS_SUCCESS)
                return GFX_STATUS_FAILURE;
            state = DRAW_LINE;
            return GFX_STATUS_SUCCESS;
        default:
            // this case should not occur
            return GFX_STATUS_SUCCESS;
    }
}
```

This non-blocking code can be used in the application and the application can do other tasks whenever DrawMyFigure() returns GFX_STATUS_FAILURE. Application should call DrawMyFigure() again until it return a GFX_STATUS_SUCCESS signifying that the Line, Rectangle and Bar shapes were drawn successfully.

To disable the non-blocking feature, add this macro in the configuration.

Remarks

None.

1.5.1.19 GFX_CONFIG_PALETTE_DISABLE Macro

Macro that disables the palette feature.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.20 GFX_CONFIG_PALETTE_EXTERNAL_DISABLE Macro

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

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.21 GFX_CONFIG_RLE_DECODE_DISABLE Macro

Macro that disables RLE compression of images.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.22 GFX_CONFIG_TRANSPARENT_COLOR_DISABLE Macro

Macro that disables the use of anti-aliased fonts.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.23 GFX_CONFIG_USE_KEYBOARD_DISABLE Macro

This macro disables the keyboard support in objects.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.24 GFX_CONFIG_USE_TOUCHSCREEN_DISABLE Macro

This macro disables the resistive touchscreen support in objects.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.25 GFX_EXTERNAL_FONT_RASTER_BUFFER_SIZE Macro

Macro sets the size of the external font raster buffer.

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.26 GFX_free Macro

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

File

gfx_config_template.h

Syntax

```
#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.

1.5.1.27 GFX_malloc Macro

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

File

gfx_config_template.h

Syntax

```
#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.

1.5.2 Configuration Examples

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

1.5.2.1 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
- from internal flash
- from external memory
- fonts from RAM is disabled
- font that are anti-aliased is enabled
- font sourced from external memory is allocated 51 bytes of buffer
- images are sourced from two places
- from internal flash

- from external memory
- fonts from RAM is disabled
- images that are RLE encoded is enabled
- images that are DEFLATE (for IPU module) encoded is enabled
- images can be rendered with transparent color feature
- touch screen support is enabled
- keyboard support is enabled
- malloc and 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)
```

1.5.2.2 Example 2

Configuration example 2.

Description

- all primitive functions are blocking
- the following are all 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, external font raster buffer is not needed
- Object layer is not used so malloc and 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)
```

1.6 Library Interface

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

Refer to each section for a detailed description.

1.6.1 Graphics Primitive Layer

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

1.6.1.1 Graphics Primitive Layer API

Graphics Library Primitive Layer Interface.

1.6.1.1.1 Initialization Functions

The following API are used to initialize the layer.

Functions

	Name	Description
	GFX_Initialize	Initialize the Graphics Library.
	GFX_ScreenClear	Clears the screen to the currently set color (GFX_ColorSet()).

1.6.1.1.1.1 GFX_Initialize Function

Initialize the Graphics Library.

File

gfx_primitive.h

Syntax

```
void GFX_Initialize();
```

Returns

None.

Description

This function initialize the Graphics Library primitive layer and Graphics Object Layer if it is 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 alphe 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

None.

Function

void GFX_Initialize(void)

1.6.1.1.1.2 GFX_ScreenClear Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_ScreenClear();
```

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(BLACK);
GFX_ScreenClear();
```

Function

GFX_STATUS GFX_ScreenClear(void)

1.6.1.1.2 Line Rendering Functions

The following API are used to render lines.

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()).

1.6.1.1.2.1 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

Syntax

```
GFX_STATUS GFX_LineDraw(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(BRIGHTRED);
GFX_LineStyleSet(GFX_LINE_STYLE_THIN_DOTTED);
GFX_LineDraw(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(

uint16_t x1,

uint16_t y1,

uint16_t x2,

uint16_t y2)

1.6.1.1.2.2 GFX_LinePositionRelativeSet Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_LinePositionRelativeSet(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(

int16_t dX,

int16_t dY)

1.6.1.1.2.3 GFX_LinePositionSet Function

This function sets the line cursor to a new position.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_LinePositionSet(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(

uint16_t x,

uint16_t y)

1.6.1.1.2.4 GFX_LinePositionXGet Function

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

File

gfx_primitive.h

Syntax

```
int16_t GFX_LinePositionXGet();
```

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(
                                  int16_t dX,
                                  int16_t dY)
{
    return (GFX_LineDraw(   GFX_LinePositionXGet(),
                            GFX_LinePositionYGet(),
                            GFX_LinePositionXGet() + dX,
                            GFX_LinePositionYGet() + dY));
}
```

Function

int16_t GFX_LinePositionXGet()

1.6.1.1.2.5 GFX_LinePositionYGet Function

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

File

gfx_primitive.h

Syntax

```
int16_t GFX_LinePositionYGet();
```

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()

1.6.1.1.2.6 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

Syntax

```
GFX_STATUS GFX_LineToDraw(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(BRIGHTRED);
GFX_LineStyleSet(GFX_LINE_STYLE_THICK_DOTTED);

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

int16_t x2,

int16_t y2)

1.6.1.1.2.7 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

Syntax

```
GFX_STATUS GFX_LineToRelativeDraw(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(BRIGHTRED);
GFX_LineStyleSet(GFX_LINE_STYLE_THIN_DOTTED);

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

int16_t dX,

int16_t dY)

1.6.1.1.3 Polygon Rendering Functions

The following API are used to render polygon shapes.

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.

1.6.1.1.3.1 GFX_CircleDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_CircleDraw(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 bahavior 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_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(BRIGHTRED);
GFX_CircleDraw(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(

uint16_t x,

uint16_t y,

uint16_t radius)

1.6.1.1.3.2 GFX_PolygonDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_PolygonDraw(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(WHITE);                          // set color
SetLineType(GFX_LINE_STYLE_THIN_DOTTED);      // set line style
GFX_PolygonDraw(3, OpenShapeXYPoints);  // draw an open shape
GFX_PolygonDraw(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(

uint16_t sides,

uint16_t *pPoints)

1.6.1.1.3.3 GFX_RectangleDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_RectangleDraw(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_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(BRIGHTRED);
GFX_RectangleDraw(30, 30, 88, 88, 15);

// draw a bright blue round rectangle
GFX_LineStyleSet(GFX_LINE_STYLE_THIN_DASHED);
GFX_ColorSet(BRIGHTBLUE);
GFX_RectangleRoundDraw(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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom)

1.6.1.1.3.4 GFX_RectangleRoundDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_RectangleRoundDraw(uint16_t left, uint16_t top, uint16_t right, uint16_t
```

```
bottom, uint16_t radius);
```

Returns

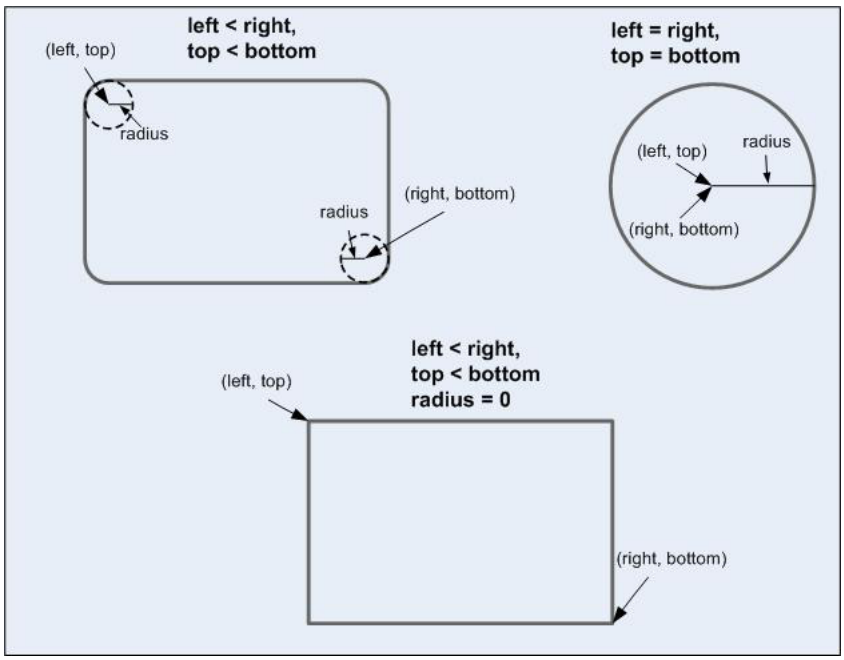
Status of the rectangle rendering. GFX_STATUS_SUCCESS - rectangle rendering done. GFX_STATUS_FAILURE - rectangle rendering is not done.

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_LINE_STYLE_THIN_SOLID);
GFX_ColorSet(BRIGHTRED);
GFX_RectangleDraw(30, 30, 88, 88, 15);

// draw a bright blue round rectangle
GFX_LineStyleSet(GFX_LINE_STYLE_THIN_DASHED);
GFX_ColorSet(BRIGHTBLUE);
GFX_RectangleRoundDraw(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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom,

uint16_t radius)

1.6.1.1.4 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

The following API are used to render filled polygon shapes.

1.6.1.1.4.1 GFX_BarDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_BarDraw(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_FILL_STYLE_COLOR);
GFX_ColorSet(RED);
status = GFX_BarDraw(10, 110, 100, 200);

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

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

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

GFX_BackgroundTypeSet(GFX_BACKGROUND_IMAGE);
GFX_ColorSet(RED);
status = GFX_BarDraw(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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom)

1.6.1.1.4.2 GFX_CircleFillDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_CircleFillDraw(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_FILL_STYLE_GRADIENT_UP);
GFX_GradientColorSet(BLUE, RED);
status = GFX_CircleFillDraw(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(

uint16_t x,

uint16_t y,

uint16_t radius)

1.6.1.1.4.3 GFX_RectangleFillDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_RectangleFillDraw(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_FILL_STYLE_GRADIENT_LEFT);
GFX_GradientColorSet(BLUE, RED);
status = GFX_RectangleFillDraw(10, 110, 100, 200);

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

// render an alpha blended rectangle with an image
GFX_FillStyleSet(GFX_FILL_STYLE_ALPHA_COLOR);
GFX_AlphaBlendingValueSet(50);
GFX_BackgroundTypeSet(GFX_BACKGROUND_IMAGE);
GFX_ColorSet(RED);
```

```
status = GFX_RectangleFillDraw(10, 110, 100, 200);

// render a plain RED rounded rectangle
GFX_FillStyleSet(GFX_FILL_STYLE_COLOR);
GFX_ColorSet(RED);
status = GFX_RectangleFillDraw(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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom)

1.6.1.1.4.4 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

Syntax

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

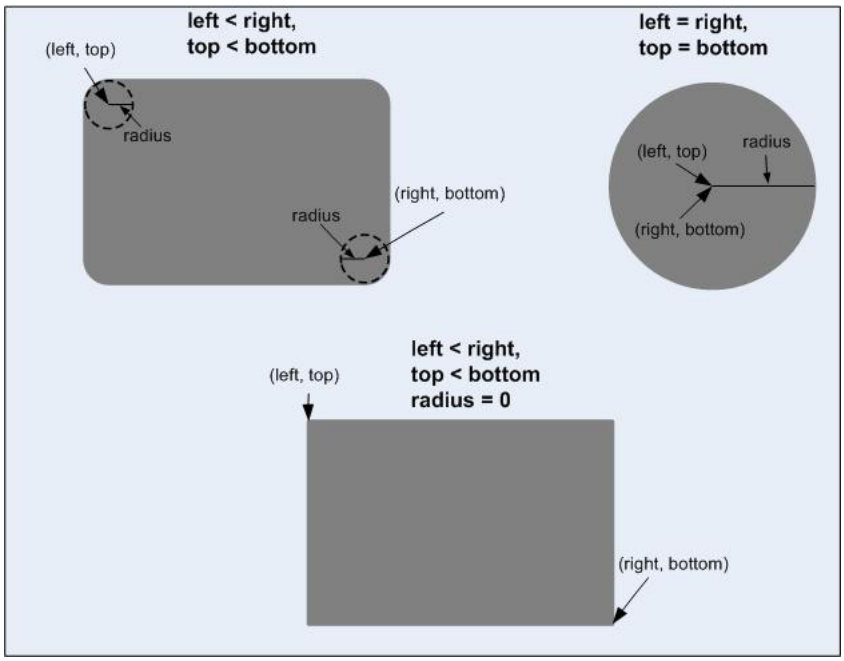
Returns

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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom,

uint16_t radius)

1.6.1.1.5 Image Rendering Functions

The following API are used to render images.

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.

1.6.1.1.5.1 GFX_ImageDraw Function

This function renders an image to the frame buffer.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_ImageDraw(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(  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(

uint16_t left,

uint16_t top,

GFX_RESOURCE_HDR *pImage)

1.6.1.1.5.2 GFX_ImageHeaderGet Function

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

File

gfx_primitive.h

Syntax

```
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)

1.6.1.1.5.3 GFX_ImageHeightGet Function

This function returns the height of the given image.

File

gfx_primitive.h

Syntax

```
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)

1.6.1.1.5.4 GFX_ImagePartialDraw Function

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

File

gfx_primitive.h

Syntax

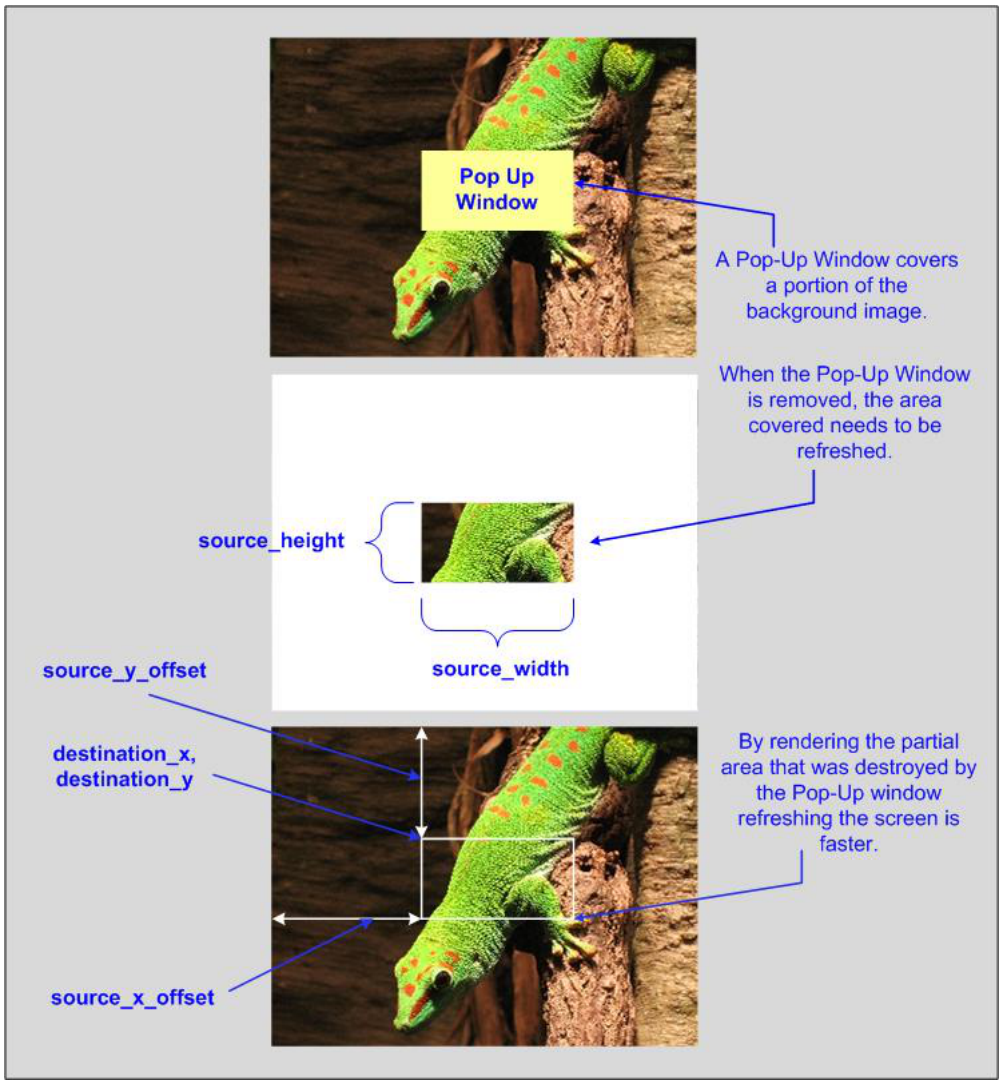
```
GFX_STATUS GFX_ImagePartialDraw(uint16_t destination_x, uint16_t destination_y, uint16_t
source_x_offset, uint16_t source_y_offset, uint16_t source_width, uint16_t source_height,
GFX_RESOURCE_HDR * pImage);
```

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(   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(

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)

1.6.1.1.5.5 GFX_ImageWidthGet Function

This function returns the width of the given image.

File

gfx_primitive.h

Syntax

```
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)
```

1.6.1.1.6 Text Rendering Functions

The following API are used to render strings and characters.

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_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_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.

1.6.1.1.6.1 GFX_FontAntiAliasGet Function

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

File

gfx_primitive.h

Syntax

```
GFX_FONT_ANTIALIAS_TYPE GFX_FontAntiAliasGet();
```

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_FONT_ANTIALIAS_OPAQUE)
    GFX_FontAntiAliasSet(GFX_FONT_ANTIALIAS_OPAQUE);

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

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

Function

GFX_FONT_ANTIALIAS_TYPE GFX_FontAntiAliasGet(void)

1.6.1.1.6.2 GFX_FontAntiAliasSet Function

This function sets the font anti-aliasing mode.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_FontAntiAliasSet(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_FONT_ANTIALIAS_TRANSLUCENT);
GFX_FontSet((GFX_RESOURCE_HDR*) &antiAliasedFont25);
GFX_ColorSet(GFX_X11_GREEN);
width = GFX_TextStringWidthGet( myString,
                                (GFX_RESOURCE_HDR*) &Font25);
height = GFX_TextStringHeightGet((GFX_RESOURCE_HDR*) &antiAliasedFont25);

GFX_TextStringDraw( (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(GFX_FONT_ANTIALIAS_TYPE type)

1.6.1.1.6.3 GFX_FontGet Function

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

File

gfx_primitive.h

Syntax

```
GFX_RESOURCE_HDR* GFX_FontGet();
```

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(void)

1.6.1.1.6.4 GFX_FontSet Function

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

File

gfx_primitive.h

Syntax

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

GFX_TextStringDraw( (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(GFX_RESOURCE_HDR *pFont)

1.6.1.1.6.5 GFX_TextCharDraw Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_TextCharDraw(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(WHITE);
GFX_FontSet(pMyFont);

// render characters until null character
while((GFX_XCHAR)(ch = *(textString + counter)) != 0)
{
    status = GFX_TextCharDraw(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(

GFX_XCHAR ch)

1.6.1.1.6.6 GFX_TextCursorPositionSet Function

This function sets the text cursor to a new position.

File

gfx_primitive.h

Syntax

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

int16_t x,

int16_t y)

1.6.1.1.6.7 GFX_TextCursorPositionXGet Function

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

File

gfx_primitive.h

Syntax

```
int16_t GFX_TextCursorPositionXGet();
```

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()

1.6.1.1.6.8 GFX_TextCursorPositionYGet Function

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

File

gfx_primitive.h

Syntax

```
int16_t GFX_TextCursorPositionYGet();
```

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()

1.6.1.1.6.9 GFX_TextStringBoxDraw Function

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

File

gfx_primitive.h

Syntax

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

Returns

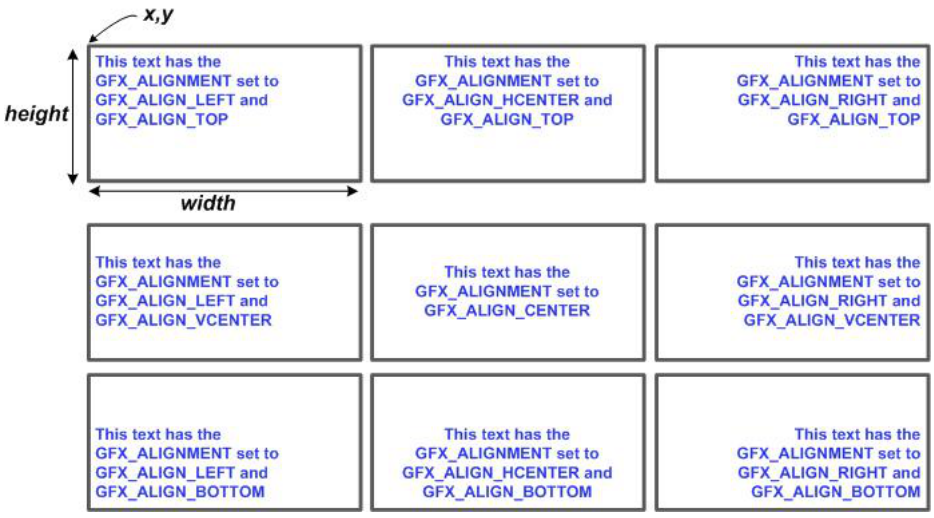
The status of the string rendering. GFX_STATUS_SUCCESS - the string was rendered GFX_STATUS_FAILURE - the string was not rendered, or is not yet finished. The function must be called again to render the remaining characters or lines.

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(BLACK);
GFX_ScreenClear();

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

// displayed message will be centered on the screen
while(GFX_TextStringBoxDraw( 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(

uint16_t x,

uint16_t y,

uint16_t width,

uint16_t height,

GFX_XCHAR *pString,

uint16_t length,

GFX_ALIGNMENT align)

1.6.1.1.6.10 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

Syntax

```
GFX_STATUS GFX_TextStringDraw(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(WHITE);
GFX_FontSet(pMyFont);

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

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

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

// doing this also renders ONLY the "String" portion of the string
GFX_TextStringDraw(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(

uint16_t x,

uint16_t y,

GFX_XCHAR *pString,

uint16_t length)

1.6.1.1.6.11 GFX_TextStringHeightGet Function

This function returns the height of the given font.

File

gfx_primitive.h

Syntax

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

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

Parameters

Parameters	Description
pFont	Pointer to the specified font.

Function

uint16_t GFX_TextStringHeightGet(

GFX_RESOURCE_HDR *pFont);

1.6.1.1.6.12 GFX_TextStringWidthGet Function

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

File

gfx_primitive.h

Syntax

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

1.6.1.1.7 Rendering Style Functions

The following API are used to set and get rendering styles.

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

1.6.1.1.7.1 GFX_AlphaBlendingValueGet Function

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

File

gfx_primitive.h

Syntax

```
uint16_t GFX_AlphaBlendingValueGet();
```

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(void)

1.6.1.1.7.2 GFX_AlphaBlendingValueSet Function

This function sets the alpha value for alpha blending rendering.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_AlphaBlendingValueSet(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(100);
while(GFX_RectangleFillDraw(10,10,30,30) != GFX_STATUS_SUCCESS);

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

GFX_AlphaBlendingValueSet(50);
while(GFX_RectangleFillDraw(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(uint16_t alpha)

1.6.1.1.7.3 GFX_FillStyleGet Function

Return the fill style used when rendering filled shapes.

File

gfx_primitive.h

Syntax

```
GFX_FILL_STYLE GFX_FillStyleGet();
```

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(void)

1.6.1.1.7.4 GFX_FillStyleSet Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_FillStyleSet(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_FILL_STYLE_GRADIENT_DOWN);
GFX_RectangleFillDraw(  10, 10,  100,  100);

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

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

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

Parameters

Parameters	Description
style	The specified fill style to be used.

Function

GFX_STATUS GFX_FillStyleSet(

GFX_FILL_STYLE style)

1.6.1.1.7.5 GFX_GradientColorSet Function

Sets the gradient fill start and end color.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_GradientColorSet(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(

GFX_COLOR startColor,

GFX_COLOR endColor)

1.6.1.1.7.6 GFX_GradientEndColorGet Function

Return the gradient end color when rendering gradient filled shapes.

File

gfx_primitive.h

Syntax

```
GFX_COLOR GFX_GradientEndColorGet();
```

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)

1.6.1.1.7.7 GFX_GradientStartColorGet Function

Return the gradient start color when rendering gradient filled shapes.

File

gfx_primitive.h

Syntax

```
GFX_COLOR GFX_GradientStartColorGet();
```

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(void)

1.6.1.1.7.8 GFX_LineStyleGet Function

Return the line style used when rendering lines.

File

gfx_primitive.h

Syntax

```
GFX_LINE_STYLE GFX_LineStyleGet();
```

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(void)

1.6.1.1.7.9 GFX_LineStyleSet Function

Set the line style to be used when rendering lines.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_LineStyleSet(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(BRIGHTRED);
GFX_LineStyleSet(GFX_LINE_STYLE_THIN_SOLID);
GFX_LineDraw(10, 10,100,  10);
```

Parameters

Parameters	Description
style	The specified line style to be used.

Function

GFX_STATUS GFX_LineStyleSet(

GFX_LINE_STYLE style)

1.6.1.1.8 Color Functions

The following API are used to set and get colors.

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.

Macros

Name	Description
GFX_Color25Convert	Function to convert the 25% equivalent of the given color.
GFX_Color50Convert	Function to convert the 50% equivalent of the given color.
GFX_Color75Convert	Function to convert the 75% equivalent of the given color.
GFX_ComponentBlueGet	Function to extract the blue component of the given color.
GFX_ComponentGreenGet	Function to extract the green component of the given color.
GFX_ComponentRedGet	Function to extract the red component of the given color.
GFX_RGBConvert	Function to convert 24 bpp color data to X bpp color data.

1.6.1.1.8.1 GFX_Color25Convert Macro

Function to convert the 25% equivalent of the given color.

File

gfx_types_macros.h

Syntax

```
#define GFX_Color25Convert(color) (GFX_COLOR)((color & (0b1110011110011100))>>2)
```

Returns

The converted color.

Description

Function to convert the 25% equivalent of the given color.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the color value that will be converted.

Function

GFX_COLOR GFX_Color25Convert(

GFX_COLOR color)

1.6.1.1.8.2 GFX_Color50Convert Macro

Function to convert the 50% equivalent of the given color.

File

gfx_types_macros.h

Syntax

```
#define GFX_Color50Convert(color) (GFX_COLOR)((color & (0b1111011111011110))>>1)
```

Returns

The converted color.

Description

Function to convert the 50% equivalent of the given color.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the color value that will be converted.

Function

GFX_COLOR GFX_Color50Convert(

GFX_COLOR color)

1.6.1.1.8.3 GFX_Color75Convert Macro

Function to convert the 75% equivalent of the given color.

File

gfx_types_macros.h

Syntax

```
#define GFX_Color75Convert(color) (GFX_COLOR)(GFX_Color50Convert(color) +
GFX_Color25Convert(color))
```

Returns

The converted color.

Description

Function to convert the 75% equivalent of the given color.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the color value that will be converted.

Function

GFX_COLOR GFX_Color75Convert(

GFX_COLOR color)

1.6.1.1.8.4 GFX_ColorGet Function

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

File

gfx_primitive.h

Syntax

```
GFX_COLOR GFX_ColorGet();
```

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(GFX_COLOR color)

1.6.1.1.8.5 GFX_ColorSet Function

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

File

gfx_primitive.h

Syntax

```
void GFX_ColorSet(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(GFX_COLOR color)

1.6.1.1.8.6 GFX_ComponentBlueGet Macro

Function to extract the blue component of the given color.

File

gfx_types_macros.h

Syntax

```
#define GFX_ComponentBlueGet(color) ((color) & 0x0000FF)
```

Returns

The blue component of the given color.

Description

Function to extract the blue component of the given color.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the color value that will be used to extract the blue component.

Function

GFX_COLOR GFX_ComponentBlueGet(

GFX_COLOR color)

1.6.1.1.8.7 GFX_ComponentGreenGet Macro

Function to extract the green component of the given color.

File

gfx_types_macros.h

Syntax

```
#define GFX_ComponentGreenGet(color) (((color) & 0x00FF00) >> 8)
```

Returns

The green component of the given color.

Description

Function to extract the green component of the given color.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the color value that will be used to extract the green component.

Function

GFX_COLOR GFX_ComponentGreenGet(

GFX_COLOR color)

1.6.1.1.8.8 GFX_ComponentRedGet Macro

Function to extract the red component of the given color.

File

gfx_types_macros.h

Syntax

```
#define GFX_ComponentRedGet(color) (((color) & 0xFF0000) >> 16)
```

Returns

The red component of the given color.

Description

Function to extract the red component of the given color.

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
color	the color value that will be used to extract the red component.

Function

GFX_COLOR GFX_ComponentRedGet(

GFX_COLOR color)

1.6.1.1.8.9 GFX_RGBConvert Macro

Function to convert 24 bpp color data to X bpp color data.

File

gfx_types_macros.h

Syntax

```
#define GFX_RGBConvert(red, green, blue) (GFX_COLOR) (((GFX_COLOR)(red) & 0xE0) |
(((GFX_COLOR)(green) & 0xE0) >> 3) | (((GFX_COLOR)(blue)) >> 6))
```

Returns

The converted color data. Size is dependent on the COLOR_DEPTH.

Description

RGB Conversion macro. 24 BPP color to x BPP color conversion. Color depths of 1, 2 or 4 are usually for monochrome/grayscale format. These are not supported in RGB conversion.

COLOR_DEPTH	Conversion
8	8-8-8 to 3-3-2 conversion
16	8-8-8 to to 5-6-5 conversion
24	8-8-8 to 8-8-8 conversion or no conversion

Remarks

None.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
red	red component of the color.
green	green component of the color.
blue	blue component of the color.

Function

GFX_COLOR GFX_RGBConvert(

uint8_t red,

uint8_t blue,

uint8_t green)

1.6.1.1.8.10 GFX_TransparentColorDisable Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_TransparentColorDisable();
```

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(BLACK);
GFX_ImageDraw(0,0, (void*)&ScreenBackground);
GFX_ImageDraw(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_ImageDraw(50,50, (void*)&OverlayImage);
```

Function

GFX_STATUS GFX_TransparentColorDisable(void)

1.6.1.1.8.11 GFX_TransparentColorEnable Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_TransparentColorEnable(GFX_COLOR color);
```

Returns

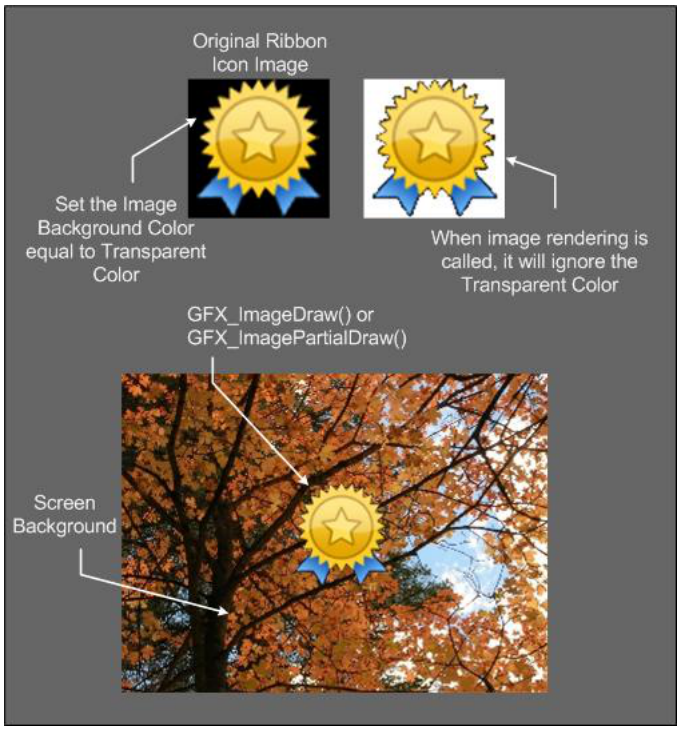
The status of the transparent color set action.

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

Parameters

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

Function

GFX_STATUS GFX_TransparentColorEnable(GFX_COLOR color)

1.6.1.1.8.12 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

Syntax

```
GFX_COLOR GFX_TransparentColorGet();
```

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(BLACK);
GFX_ImageDraw(0,0, (void*)&ScreenBackground);
GFX_ImageDraw(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 == BLACK)
    GFX_TransparentColorDisable();
GFX_ImageDraw(50,50, (void*)&OverlayImage);
```

Function

GFX_COLOR GFX_TransparentColorGet(void)

1.6.1.1.8.13 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

Syntax

```
GFX_FEATURE_STATUS GFX_TransparentColorStatusGet();
```

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_FEATURE_DISABLED)
    GFX_TransparentColorEnable(BLACK);
GFX_ImageDraw(0,0, (void*)&ScreenBackground);
GFX_ImageDraw(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 == BLACK)
    GFX_TransparentColorDisable();
GFX_ImageDraw(50,50, (void*)&OverlayImage);
```

Function

GFX_FEATURE_STATUS GFX_TransparentColorStatusGet(void)

1.6.1.1.9 Background Functions

The following API are used to set and manipulate the background.

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.

1.6.1.1.9.1 GFX_BackgroundColorGet Function

This function returns the color used in the current background.

File

gfx_primitive.h

Syntax

```
GFX_COLOR GFX_BackgroundColorGet();
```

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(void)

1.6.1.1.9.2 GFX_BackgroundImageGet Function

This function returns the image used in the current background.

File

gfx_primitive.h

Syntax

```
GFX_RESOURCE_HDR * GFX_BackgroundImageGet();
```

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(void)

1.6.1.1.9.3 GFX_BackgroundImageLeftGet Function

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

File

gfx_primitive.h

Syntax

```
uint16_t GFX_BackgroundImageLeftGet();
```

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(void)

1.6.1.1.9.4 GFX_BackgroundImageTopGet Function

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

File

gfx_primitive.h

Syntax

```
uint16_t GFX_BackgroundImageTopGet();
```

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(void)

1.6.1.1.9.5 GFX_BackgroundSet Function

This function sets the background information.

File

gfx_primitive.h

Syntax

```
void GFX_BackgroundSet(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(uint16_t left,

uint16_t top,

GFX_RESOURCE_HDR *pImage,

GFX_COLOR color);

1.6.1.1.9.6 GFX_BackgroundTypeGet Function

This function returns the type of the current background.

File

gfx_primitive.h

Syntax

```
GFX_BACKGROUND_TYPE GFX_BackgroundTypeGet();
```

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(void)

1.6.1.1.9.7 GFX_BackgroundTypeSet Function

This function sets the background type.

File

gfx_primitive.h

Syntax

```
void GFX_BackgroundTypeSet(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(GFX_BACKGROUND_TYPE type)

1.6.1.1.10 Double Buffering Functions

The following API are used to manage double buffering.

Functions

	Name	Description
	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_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_FrameBufferGet	This function returns the index of the current frame buffer.
	GFX_FrameBufferSet	This function sets the frame buffer with the given index number.

1.6.1.1.10.1 GFX_DoubleBufferAreaGet Function

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

File

gfx_primitive.h

Syntax

```
GFX_RECTANGULAR_AREA * GFX_DoubleBufferAreaGet(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(uint16_t index)

1.6.1.1.10.2 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

Syntax

```
void GFX_DoubleBufferAreaMark(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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom)

1.6.1.1.10.3 GFX_DoubleBufferDisable Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DoubleBufferDisable();
```

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(void)

1.6.1.1.10.4 GFX_DoubleBufferEnable Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DoubleBufferEnable();
```

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(void)

1.6.1.1.10.5 GFX_DoubleBufferStatusGet Function

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

File

gfx_primitive.h

Syntax

```
GFX_FEATURE_STATUS GFX_DoubleBufferStatusGet();
```

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(void)

1.6.1.1.10.6 GFX_DoubleBufferSyncAllStatusClear Function

This function clears the synchronize all status.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DoubleBufferSyncAllStatusClear();
```

Returns

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

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(void)

1.6.1.1.10.7 GFX_DoubleBufferSyncAllStatusGet Function

This function returns the status of the synchronize all flag.

File

gfx_primitive.h

Syntax

```
GFX_FEATURE_STATUS GFX_DoubleBufferSyncAllStatusGet();
```

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(void)

1.6.1.1.10.8 GFX_DoubleBufferSyncAllStatusSet Function

This function sets the whole draw buffer to be unsynchronized.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DoubleBufferSyncAllStatusSet();
```

Returns

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

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(void)

1.6.1.1.10.9 GFX_DoubleBufferSyncAreaCountGet Function

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

File

gfx_primitive.h

Syntax

```
uint16_t GFX_DoubleBufferSyncAreaCountGet();
```

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(void)

1.6.1.1.10.10 GFX_DoubleBufferSyncAreaCountSet Function

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

File

gfx_primitive.h

Syntax

```
void GFX_DoubleBufferSyncAreaCountSet(uint16_t count);
```

Returns

None.

Description

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

Preconditions

Double buffering feature must be enabled.

Example

None.

Function

void GFX_DoubleBufferSyncAreaCountSet(uint16_t count)

1.6.1.1.10.11 GFX_DoubleBufferSynchronize Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DoubleBufferSynchronize();
```

Returns

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

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(void)

1.6.1.1.10.12 GFX_DoubleBufferSynchronizeRequest Function

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

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DoubleBufferSynchronizeRequest();
```

Returns

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

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(void)

1.6.1.1.10.13 GFX_DoubleBufferSynchronizeStatusGet Function

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

File

gfx_primitive.h

Syntax

```
GFX_FEATURE_STATUS GFX_DoubleBufferSynchronizeStatusGet();
```

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(void)

1.6.1.1.10.14 GFX_DrawBufferGet Function

This function returns the index of the current draw buffer.

File

gfx_primitive.h

Syntax

```
uint16_t GFX_DrawBufferGet();
```

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(void)

1.6.1.1.10.15 GFX_DrawBufferInitialize Function

This function initializes the address of the draw buffer specified by the given index.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DrawBufferInitialize(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(

uint16_t index,

uint32_t address)

1.6.1.1.10.16 GFX_DrawBufferSet Function

This function sets the draw buffer with the given index number.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_DrawBufferSet(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(uint16_t index)

1.6.1.1.10.17 GFX_FrameBufferGet Function

This function returns the index of the current frame buffer.

File

gfx_primitive.h

Syntax

```
uint16_t GFX_FrameBufferGet();
```

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)

1.6.1.1.10.18 GFX_FrameBufferSet Function

This function sets the frame buffer with the given index number.

File

gfx_primitive.h

Syntax

```
GFX_STATUS GFX_FrameBufferSet(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(uint16_t index)

1.6.1.1.11 External Resources Functions

The following API are used to access external resources in the Graphics Library.

Functions

	Name	Description
	GFX_ExternalResourceCallback	This function performs data fetch from external memory.

1.6.1.1.11.1 GFX_ExternalResourceCallback Function

This function performs data fetch from external memory.

File

gfx_primitive.h

Syntax

```
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 succesfully 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)
```

1.6.1.2 Data Types and Constants

The following are the types and constants used in the Graphics Primitive Layer.

Types

Name	Description
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_GLYPH_ENTRY	The structure describing the glyph entry in fonts.
GFX_FONT_GLYPH_ENTRY_EXTENDED	The structure describing the entended glyph entry in fonts.
GFX_FONT_HEADER	The structure used to define the font header.
GFX_LINE_STYLE	Specifies the different line styles.
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.

1.6.1.2.1 GFX_ALIGNMENT Type

Summary of the different text alignment supported in the library.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.2 GFX_BACKGROUND Type

Background information structure.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.3 GFX_BACKGROUND_TYPE Type

Specifies the different background fill types.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.4 GFX_DOUBLE_BUFFERING_MODE Type

Structure used for double buffering management.

File

gfx_types_macros.h

Syntax

```
typedef struct {
  GFX_FEATURE_STATUS gfxDoubleBufferFeature;
  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 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.

1.6.1.2.5 GFX_FEATURE_STATUS Type

States of a feature that can be enabled/disabled at run time.

File

gfx_types_macros.h

Syntax

```
typedef enum {
  GFX_FEATURE_ENABLED,
  GFX_FEATURE_DISABLED
} GFX_FEATURE_STATUS;
```

Members

Members	Description
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 nbe enabled/disabled at run time.

Remarks

None.

1.6.1.2.6 GFX_FILL_STYLE Type

Specifies the available fill styles.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.7 GFX_FONT_ANTIALIAS_TYPE Type

Summary of the transparency types in text anti-aliasing.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.8 GFX_FONT_GLYPH_ENTRY Type

The structure describing the glyph entry in fonts.

File

gfx_types_font.h

Syntax

```
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.

1.6.1.2.9 GFX_FONT_GLYPH_ENTRY_EXTENDED Type

The structure describing the entended glyph entry in fonts.

File

gfx_types_font.h

Syntax

```
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 entended 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.

1.6.1.2.10 GFX_FONT_HEADER Type

The structure used to define the font header.

File

gfx_types_font.h

Syntax

```
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.

1.6.1.2.11 GFX_LINE_STYLE Type

Specifies the different line styles.

File

gfx_types_macros.h

Syntax

```
typedef enum {
  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_LINE_STYLE;
```

Members

Members	Description
GFX_LINE_STYLE_THIN_SOLID	solid line, 1 pixel wide (default)
GFX_LINE_STYLE_THIN_DOTTED	dotted line, 1 pixel wide
GFX_LINE_STYLE_THIN_DASHED	dashed line, , 1 pixel wide
GFX_LINE_STYLE_THICK_SOLID	solid line, 3 pixel wide

GFX_LINE_STYLE_THICK_DOTTED	dotted line, 3 pixel wide
GFX_LINE_STYLE_THICK_DASHED	dashed line, 3 pixel wide

Description

Typedef: GFX_LINE_STYLE

This enumeration specifies the available line styles used in the library.

Remarks

None.

1.6.1.2.12 GFX_MCHP_BITMAP_HEADER Type

The structure used to define the Microchip bitmap header.

File

gfx_types_image.h

Syntax

```
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.

1.6.1.2.13 GFX_PARTIAL_IMAGE_PARAM Type

Partial Image information structure.

File

gfx_types_macros.h

Syntax

```
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 tha partial image (with respect to the image horizontal origin).
uint16_t yoffset;	yoffset of tha 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.

1.6.1.2.14 GFX_RECTANGULAR_AREA Type

A generic rectangular area structure.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.15 GFX_RESOURCE Type

Specifies the different resource types in the library.

File

gfx_types_resource.h

Syntax

```
typedef enum {
  GFX_RESOURCE_MEMORY_FLASH,
```

```
    GFX_RESOURCE_MEMORY_EXTERNAL,
    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_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_EDS_EPMP_NONE = (GFX_RESOURCE_TYPE_JPEG | 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_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_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_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_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

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.

1.6.1.2.16 GFX_RESOURCE_BINARY Type

Defines the structure used for the binary type resource.

File

gfx_types_resource.h

Syntax

```
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.

1.6.1.2.17 GFX_RESOURCE_FONT Type

Defines the structure used for the font type resource.

File

gfx_types_resource.h

Syntax

```
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.

1.6.1.2.18 GFX_RESOURCE_HDR Type

Defines the structure used for the resource types.

File

gfx_types_resource.h

Syntax

```
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.

1.6.1.2.19 GFX_RESOURCE_IMAGE Type

Defines the structure used for the image type resource.

File

gfx_types_resource.h

Syntax

```
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.

1.6.1.2.20 GFX_RESOURCE_PALETTE Type

Defines the structure used for the palette type resource.

File

gfx_types_resource.h

Syntax

```
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.

1.6.1.2.21 GFX_STATUS Type

Rendering status.

File

gfx_types_macros.h

Syntax

```
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.

1.6.1.2.22 GFX_STATUS_BIT Type

Additional rendering status.

File

gfx_types_macros.h

Syntax

```
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.

1.6.2 Graphics Object Layer

Graphics Object Layer Interface.

1.6.2.1 GOL Objects

The Graphics Object Layer (GOL) contains the Advanced Graphics Objects or commonly known as widgets.

1.6.2.1.1 Button Object

Button is an object that emulates a press and release effect when operated upon.

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.

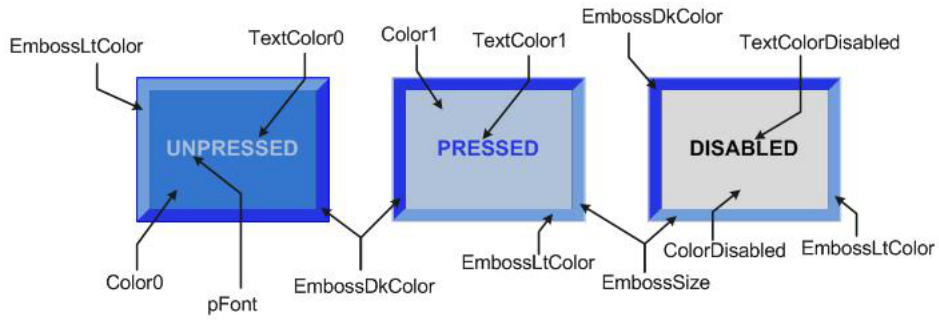
Macros

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.

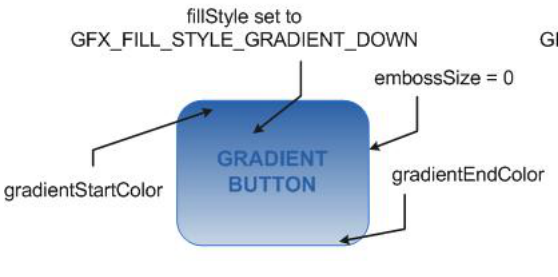
Description

Button supports Keyboard and Touchscreen inputs, replying to their events with the pre-defined actions (see GFX_GOL_ButtonActionGet() and GFX_GOL_ButtonActionSet() for details).

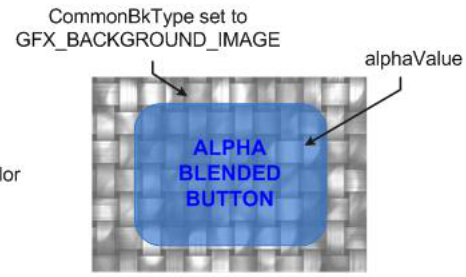
The button object is rendered using the assigned style scheme. The following figure illustrates the usage of the style schemes in the button object.



Button Style Scheme



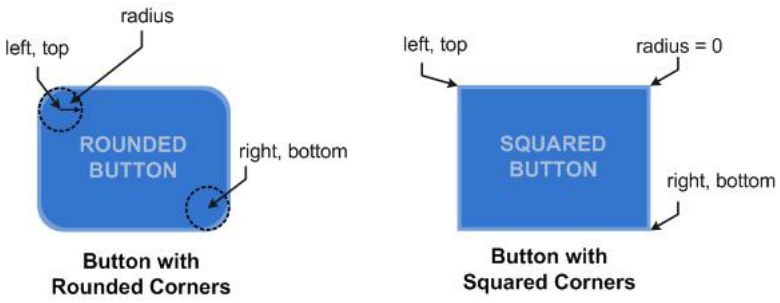
Gradient Fills in Buttons



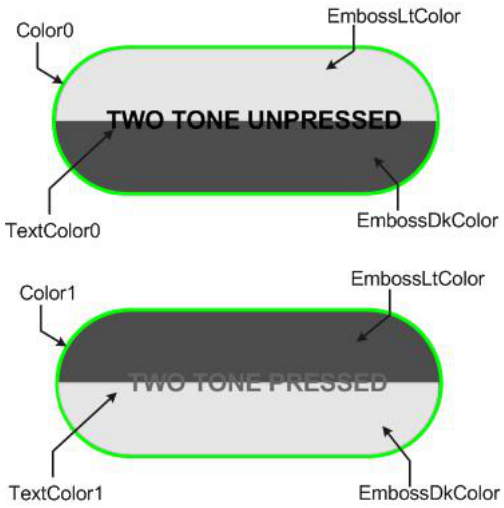
Alpha Blending in Buttons

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.

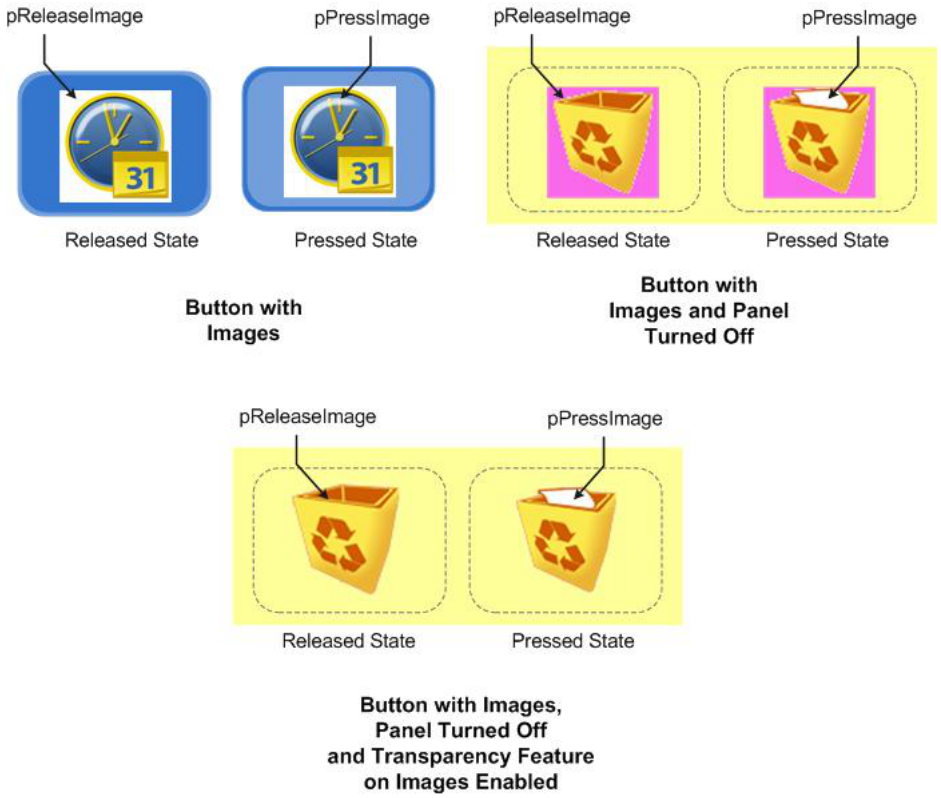


Two Tone Button

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.

Remarks

- Alpha blending of gradient fills is not supported by the object.

1.6.2.1.1.1 GFX_GOL_ButtonPressStateImageGet Macro

This function gets the image used when in the pressed state.

File

gfx_gol_button.h

Syntax

```
#define GFX_GOL_ButtonPressStateImageGet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pPressImage)
```

Returns

Pointer to the image resource.

Description

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)

1.6.2.1.1.2 GFX_GOL_ButtonPressStateImageSet Macro

This function sets the image to be used when in the pressed state.

File

gfx_gol_button.h

Syntax

```
#define GFX_GOL_ButtonPressStateImageSet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pPressImage = pImage)
```

Returns

None.

Description

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)

1.6.2.1.1.3 GFX_GOL_ButtonReleaseStateImageGet Macro

This function gets the image used when in the released state.

File

gfx_gol_button.h

Syntax

```
#define GFX_GOL_ButtonReleaseStateImageGet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pReleaseImage)
```

Returns

Pointer to the image resource.

Description

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)

1.6.2.1.1.4 GFX_GOL_ButtonReleaseStateImageSet Macro

This function sets the image to be used when in the released state.

File

gfx_gol_button.h

Syntax

```
#define GFX_GOL_ButtonReleaseStateImageSet(pObject, pImage) \
            (((GFX_GOL_BUTTON *)pObject)->pReleaseImage = pImage)
```

Returns

None.

Description

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)

1.6.2.1.1.5 GFX_GOL_ButtonTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_button.h

Syntax

```
#define GFX_GOL_ButtonTextGet(pObject) (((GFX_GOL_BUTTON *)pObject)->pText)
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.1.6 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

Syntax

```
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

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);

1.6.2.1.1.7 GFX_GOL_ButtonActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_button.h

Syntax

```
void GFX_GOL_ButtonActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.1.8 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

Syntax

```
GFX_GOL_BUTTON * GFX_GOL_ButtonCreate(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

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(
        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(
        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(
        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
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(
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)
```

1.6.2.1.1.9 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

Syntax

```
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

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)

1.6.2.1.1.10 GFX_GOL_ButtonTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_button.h

Syntax

```
GFX_ALIGNMENT GFX_GOL_ButtonTextAlignmentGet(GFX_GOL_BUTTON * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

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)

1.6.2.1.1.11 GFX_GOL_ButtonTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_button.h

Syntax

```
void GFX_GOL_ButtonTextAlignmentSet(GFX_GOL_BUTTON * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

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)

1.6.2.1.1.12 GFX_GOL_ButtonTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_button.h

Syntax

```
void GFX_GOL_ButtonTextSet(GFX_GOL_BUTTON * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

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)

1.6.2.1.2 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.

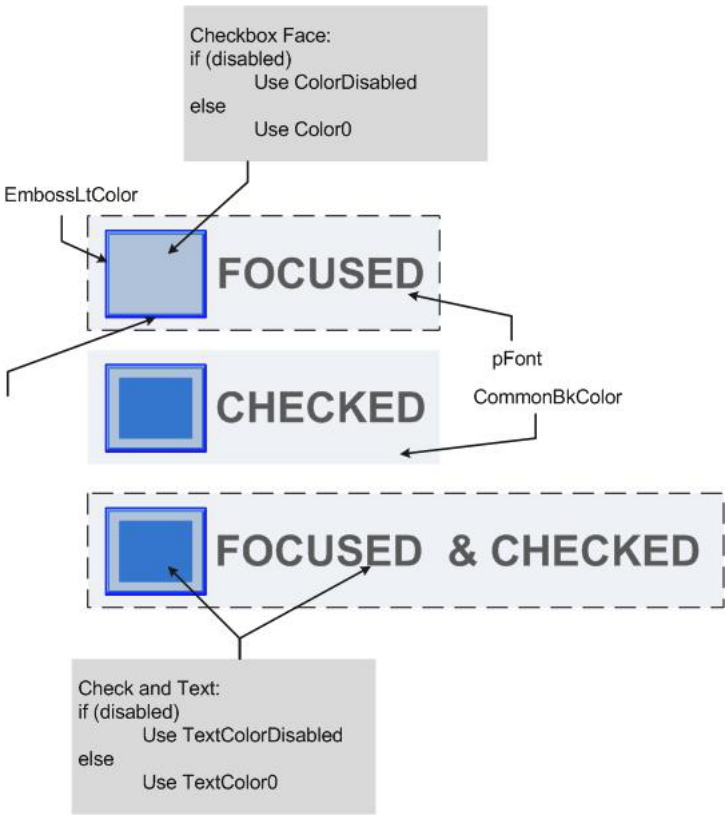
Macros

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 pre-defined 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.

1.6.2.1.2.1 GFX_GOL_CheckBoxTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_check_box.h

Syntax

```
#define GFX_GOL_CheckBoxTextGet(pObject) (((GFX_GOL_CHECKBOX *)pObject)->pText)
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.2.2 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

Syntax

```
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

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);

1.6.2.1.2.3 GFX_GOL_CheckBoxActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_check_box.h

Syntax

```
void GFX_GOL_CheckBoxActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.2.4 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

Syntax

```
GFX_GOL_CHECKBOX * GFX_GOL_CheckBoxCreate(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

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(
                    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(
                    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
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(

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)

1.6.2.1.2.5 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

Syntax

```
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

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)

1.6.2.1.2.6 GFX_GOL_CheckBoxTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_check_box.h

Syntax

```
GFX_ALIGNMENT GFX_GOL_CheckBoxTextAlignmentGet(GFX_GOL_CHECKBOX * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

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)

1.6.2.1.2.7 GFX_GOL_CheckBoxTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_check_box.h

Syntax

```
void GFX_GOL_CheckBoxTextAlignmentSet(GFX_GOL_CHECKBOX * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

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)

1.6.2.1.2.8 GFX_GOL_CheckBoxTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_check_box.h

Syntax

```
void GFX_GOL_CheckBoxTextSet(GFX_GOL_CHECKBOX * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

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)

1.6.2.1.3 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

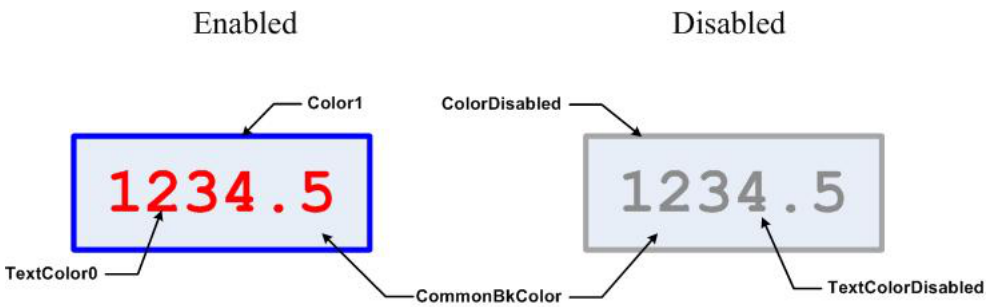
Name	Description
GFX_GOL_DigitalMeterTextAlignmentGet	This function returns the text alignment of the text string used by the object.
GFX_GOL_DigitalMeterTextAlignmentSet	This function sets the text alignment of the text string used by the object.
GFX_GOL_DigitalMeterValueGet	This function returns the 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 pre-defined 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.



1.6.2.1.3.1 GFX_GOL_DigitalMeterTextAlignmentGet Macro

This function returns the text alignment of the text string used by the object.

File

gfx_gol_digital_meter.h

Syntax

```
#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

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)

1.6.2.1.3.2 GFX_GOL_DigitalMeterTextAlignmentSet Macro

This function sets the text alignment of the text string used by the object.

File

gfx_gol_digital_meter.h

Syntax

```
#define GFX_GOL_DigitalMeterTextAlignmentSet(pObject, align) \
                (((GFX_GOL_DIGITALMETER *)pObject)->alignment = align)
```

Returns

None.

Description

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)

1.6.2.1.3.3 GFX_GOL_DigitalMeterValueGet Macro

This function returns the current value of the digital meter.

File

gfx_gol_digital_meter.h

Syntax

```
#define GFX_GOL_DigitalMeterValueGet(pObject) \
                                (((GFX_GOL_DIGITALMETER*)pObject)->value)
```

Returns

The current value of the object.

Description

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();

// 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();

    // 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)

1.6.2.1.3.4 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

Syntax

```
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

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)

1.6.2.1.3.5 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

Syntax

```
GFX_GOL_DIGITALMETER * GFX_GOL_DigitalMeterCreate(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

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(
        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() != GFX_STATUS_SUCCESS);
```

Parameters

Parameters	Description
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(

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)

1.6.2.1.3.6 GFX_GOL_DigitalMeterDecrement Function

This function decrements the meter value by the delta value set.

File

gfx_gol_digital_meter.h

Syntax

```
void GFX_GOL_DigitalMeterDecrement(GFX_GOL_DIGITALMETER * pObject, uint16_t delta);
```

Returns

None.

Description

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)

1.6.2.1.3.7 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

Syntax

```
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

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)

1.6.2.1.3.8 GFX_GOL_DigitalMeterIncrement Function

This function increments the meter value by the delta value set.

File

gfx_gol_digital_meter.h

Syntax

```
void GFX_GOL_DigitalMeterIncrement(GFX_GOL_DIGITALMETER * pObject, uint16_t delta);
```

Returns

None.

Description

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)

1.6.2.1.3.9 GFX_GOL_DigitalMeterValueSet Function

This function sets the value of the meter.

File

gfx_gol_digital_meter.h

Syntax

```
void GFX_GOL_DigitalMeterValueSet(GFX_GOL_DIGITALMETER * pObject, int16_t value);
```

Returns

None.

Description

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();

while("some condition")
{
    GFX_GOL_DigitalMeterValueSet(pMeter, ctr);
    GFX_GOL_ObjectStateSet( pMeter,
                            GFX_GOL_DIGITALMETER_UPDATE_STATE);

    // redraw the scroll bar
    GFX_GOL_ObjectListDraw();

   // 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)

1.6.2.1.4 Edit Box Object

Edit Box is 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.

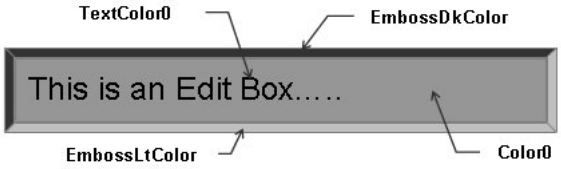
Macros

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.

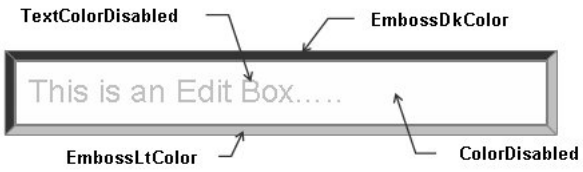
Description

Edit Box supports only keyboard inputs, replying to the events with the pre-defined 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

When creating the object, the alignment of the text of the object can be formatted with the same options that GFX_TextStringBoxDraw() allows.

1.6.2.1.4.1 GFX_GOL_EditBoxTextAlignmentGet Macro

This function returns the text alignment of the text string used by the object.

File

gfx_gol_edit_box.h

Syntax

```
#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

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)

1.6.2.1.4.2 GFX_GOL_EditBoxTextAlignmentSet Macro

This function sets the text alignment of the text string used by the object.

File

gfx_gol_edit_box.h

Syntax

```
#define GFX_GOL_EditBoxTextAlignmentSet(pObject, align) \
                (((GFX_GOL_EDITBOX *)pObject)->alignment = align)
```

Returns

None.

Description

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)

1.6.2.1.4.3 GFX_GOL_EditBoxTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_edit_box.h

Syntax

```
#define GFX_GOL_EditBoxTextGet(pObject) (((GFX_GOL_EDITBOX *)pObject)->pText)
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.4.4 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

Syntax

```
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. Careet will be drawn when enabled.
- GFX_GOL_OBJECT_ACTION_INVALID - Object is not affected

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	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);

1.6.2.1.4.5 GFX_GOL_EditBoxActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_edit_box.h

Syntax

```
void GFX_GOL_EditBoxActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.4.6 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

Syntax

```
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

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)

1.6.2.1.4.7 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

Syntax

```
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

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)

1.6.2.1.4.8 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

Syntax

```
GFX_GOL_EDITBOX * GFX_GOL_EditBoxCreate(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

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
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(
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)
```

1.6.2.1.4.9 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

Syntax

```
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

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)

1.6.2.1.4.10 GFX_GOL_EditBoxTextSet Function

This function sets the text in the object.

File

gfx_gol_edit_box.h

Syntax

```
void GFX_GOL_EditBoxTextSet(GFX_GOL_EDITBOX * pObject, GFX_XCHAR* pText);
```

Returns

None.

Description

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)
```

1.6.2.1.5 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_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.

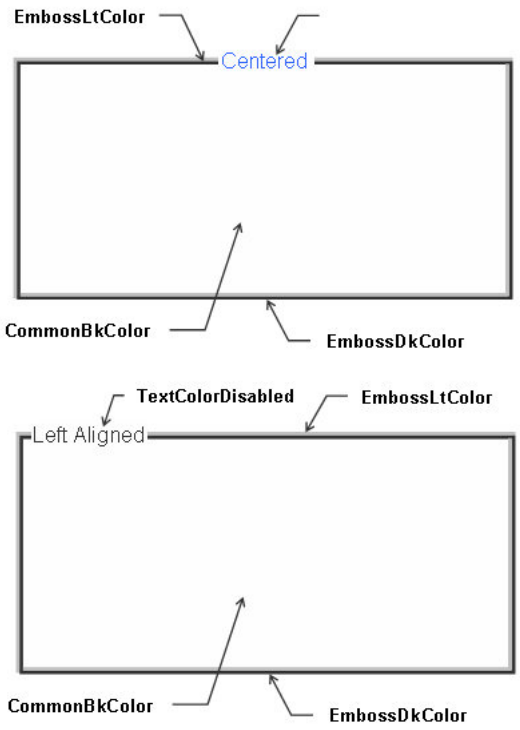
Macros

Name	Description
GFX_GOL_GroupboxTextGet	This function returns the address of the current text string used by the object.

Description

Group Box supports only Touchscreen inputs, replying to the events with the pre-defined 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.



1.6.2.1.5.1 GFX_GOL_GroupboxTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_group_box.h

Syntax

```
#define GFX_GOL_GroupboxTextGet(pObject) (((GFX_GOL_GROUPBOX *)pObject)->pText)
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.5.2 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

Syntax

```
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

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);

1.6.2.1.5.3 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

Syntax

```
GFX_GOL_GROUPBOX * GFX_GOL_GroupboxCreate(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

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(
       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
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(

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)

1.6.2.1.5.4 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

Syntax

```
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

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)

1.6.2.1.5.5 GFX_GOL_GroupboxTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_group_box.h

Syntax

```
GFX_ALIGNMENT GFX_GOL_GroupboxTextAlignmentGet(GFX_GOL_GROUPBOX * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

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)

1.6.2.1.5.6 GFX_GOL_GroupboxTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_group_box.h

Syntax

```
void GFX_GOL_GroupboxTextAlignmentSet(GFX_GOL_GROUPBOX * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

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)

1.6.2.1.5.7 GFX_GOL_GroupboxTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_group_box.h

Syntax

```
void GFX_GOL_GroupboxTextSet(GFX_GOL_GROUPBOX * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

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)

1.6.2.1.6 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_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.

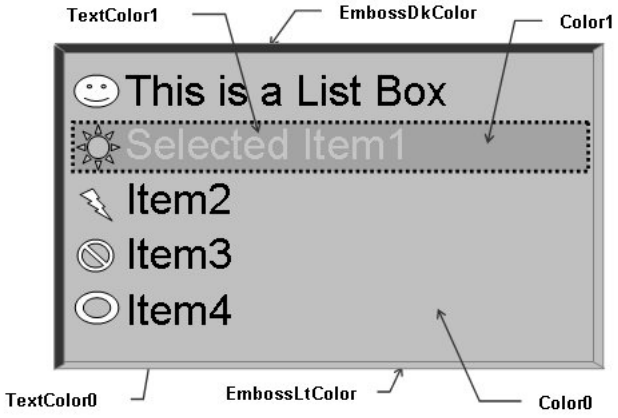
Macros

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.

Description

List Box supports Keyboard and Touchscreen inputs, replying to their events with the pre-defined 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.



When creating the object, the alignment of the text of the object can be formatted with the same options that GFX_TextStringBoxDraw() allows.

1.6.2.1.6.1 GFX_GOL_ListBoxItemCountGet Macro

This function returns the count of items in the list box.

File

gfx_gol_list_box.h

Syntax

```
#define GFX_GOL_ListBoxItemCountGet(pObject) \
                (((GFX_GOL_LISTBOX *)pObject)->itemsNumber)
```

Returns

The number of items in the list box.

Description

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)

1.6.2.1.6.2 GFX_GOL_ListBoxItemImageGet Macro

This function gets the image set for a list box item.

File

gfx_gol_list_box.h

Syntax

```
#define GFX_GOL_ListBoxItemImageGet(pItem) (((GFX_GOL_LISTITEM *)pItem)->pImage)
```

Returns

Pointer to the image used for the item.

Description

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)

1.6.2.1.6.3 GFX_GOL_ListBoxItemImageSet Macro

This function sets the image for a list box item.

File

gfx_gol_list_box.h

Syntax

```
#define GFX_GOL_ListBoxItemImageSet(pItem, pImage) (((GFX_GOL_LISTITEM *)pItem)->pImage =
pImage)
```

Returns

None.

Description

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)

1.6.2.1.6.4 GFX_GOL_ListBoxItemListGet Macro

This function returns the pointer to the item list of the list box.

File

gfx_gol_list_box.h

Syntax

```
#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

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)

1.6.2.1.6.5 GFX_GOL_ListBoxItemSelectStatusClear Macro

This function clears the selection status of the item.

File

gfx_gol_list_box.h

Syntax

```
#define GFX_GOL_ListBoxItemSelectStatusClear(pObject, pItem) \
                                                              \
                if(pItem->status & GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED)    \
                    GFX_GOL_ListBoxSelectionChange((LISTBOX *)pObject, pItem);
```

Returns

None.

Description

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)

1.6.2.1.6.6 GFX_GOL_ListBoxItemSelectStatusSet Macro

This function sets the selection status of the item to selected.

File

gfx_gol_list_box.h

Syntax

```
#define GFX_GOL_ListBoxItemSelectStatusSet(pObject, pItem) \
                                                                \
                if(!(pItem->status & GFX_GOL_LISTBOX_ITEM_STATUS_SELECTED)) \
                    GFX_GOL_ListBoxSelectionChange((LISTBOX *)pObject, pItem);
```

Returns

None.

Description

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)

1.6.2.1.6.7 GFX_GOL_ListBoxTextAlignmentGet Macro

This function returns the text alignment of the text string used by the object.

File

gfx_gol_list_box.h

Syntax

```
#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

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)

1.6.2.1.6.8 GFX_GOL_ListBoxTextAlignmentSet Macro

This function sets the text alignment of the text string used by the object.

File

gfx_gol_list_box.h

Syntax

```
#define GFX_GOL_ListBoxTextAlignmentSet(pObject, align) \
                (((GFX_GOL_LISTBOX *)pObject)->alignment = align)
```

Returns

None.

Description

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)

1.6.2.1.6.9 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

Syntax

```
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

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);

1.6.2.1.6.10 GFX_GOL_ListBoxActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_list_box.h

Syntax

```
void GFX_GOL_ListBoxActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.6.11 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

Syntax

```
GFX_GOL_LISTBOX * GFX_GOL_ListBoxCreate(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

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 listbox with default style scheme
pLb = GFX_GOL_ListBoxCreate(    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 Listbox was created
if (pLb == NULL)
   return 0;
```

```
// create the list of items to be placed in the listbox
// 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
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(
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)
```

1.6.2.1.6.12 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

Syntax

```
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

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)

1.6.2.1.6.13 GFX_GOL_ListBoxItemAdd Function

This function adds an item to the list box.

File

gfx_gol_list_box.h

Syntax

```
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

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)
```

1.6.2.1.6.14 GFX_GOL_ListBoxItemFocusGet Function

This function returns the index of the focused item in the list box.

File

gfx_gol_list_box.h

Syntax

```
int16_t GFX_GOL_ListBoxItemFocusGet(GFX_GOL_LISTBOX * pObject);
```

Returns

The index of the focused item in the list box.

Description

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)
```

1.6.2.1.6.15 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

Syntax

```
void GFX_GOL_ListBoxItemFocusSet(GFX_GOL_LISTBOX * pObject, uint16_t index);
```

Returns

None.

Description

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)

1.6.2.1.6.16 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

Syntax

```
void GFX_GOL_ListBoxItemListRemove(void * pObject);
```

Returns

None.

Description

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)

1.6.2.1.6.17 GFX_GOL_ListBoxItemRemove Function

This function removes an item from the list box and free the memory used.

File

gfx_gol_list_box.h

Syntax

```
void GFX_GOL_ListBoxItemRemove(GFX_GOL_LISTBOX * pObject, GFX_GOL_LISTITEM * pItem);
```

Returns

None.

Description

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)

1.6.2.1.6.18 GFX_GOL_ListBoxSelectionChange Function

This function changes the selection status of the given item in the list box.

File

gfx_gol_list_box.h

Syntax

```
void GFX_GOL_ListBoxSelectionChange(GFX_GOL_LISTBOX * pObject, GFX_GOL_LISTITEM * pItem);
```

Returns

None.

Description

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)

1.6.2.1.6.19 GFX_GOL_ListBoxSelectionGet Function

This function searches for selected items from the list box.

File

gfx_gol_list_box.h

Syntax

```
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

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)
```

1.6.2.1.6.20 GFX_GOL_ListBoxVisibleItemCountGet Function

This function returns the count of items visible in the list box.

File

gfx_gol_list_box.h

Syntax

```
uint16_t GFX_GOL_ListBoxVisibleItemCountGet(GFX_GOL_LISTBOX * pObject);
```

Returns

The number of visible items in the list box.

Description

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)
```

1.6.2.1.7 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:

1. MTR_WHOLE_TYPE
2. MTR_HALF_TYPE
3. MTR_QUARTER_TYPE

Meter supports system inputs, replying to their events with the pre-defined 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.

Default Half Meter

Half Meter with Arc enabled

Note:
Normal, Critical and Danger Colors are not part of the style scheme struct. They are fields in the METER struct.

1.6.2.1.7.1 GFX_GOL_MeterMaximumValueGet Macro

This function returns the maximum value set for the meter.

File

gfx_gol_meter.h

Syntax

```
#define GFX_GOL_MeterMaximumValueGet(pObject) (((GFX_GOL_METER *)pObject)->maxValue)
```

Returns

The current maximum value set for the object.

Description

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)

1.6.2.1.7.2 GFX_GOL_MeterMinimumValueGet Macro

This function returns the minimum value set for the meter.

File

gfx_gol_meter.h

Syntax

```
#define GFX_GOL_MeterMinimumValueGet(pObject) (((GFX_GOL_METER *)pObject)->minValue)
```

Returns

The current minimum value set for the object.

Description

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)

1.6.2.1.7.3 GFX_GOL_MeterTitleFontSet Macro

This function sets the meter title font used.

File

gfx_gol_meter.h

Syntax

```
#define GFX_GOL_MeterTitleFontSet(pObject, pNewFont) (((GFX_GOL_METER
*)pObject)->pTitleFont = pNewFont)
```

Returns

None.

Description

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)

1.6.2.1.7.4 GFX_GOL_MeterTypeSet Macro

This function sets the meter draw type.

File

gfx_gol_meter.h

Syntax

```
#define GFX_GOL_MeterTypeSet(pObject, type) (((GFX_GOL_METER *)pObject)->type = type)
```

Returns

None.

Description

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)

1.6.2.1.7.5 GFX_GOL_MeterValueFontSet Macro

This function sets the meter's displayed value font used.

File

gfx_gol_meter.h

Syntax

```
#define GFX_GOL_MeterValueFontSet(pObject, pNewFont) (((GFX_GOL_METER
*)pObject)->pValueFont = pNewFont)
```

Returns

None.

Description

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)

1.6.2.1.7.6 GFX_GOL_MeterValueGet Macro

This function returns the current value of the meter.

File

gfx_gol_meter.h

Syntax

```
#define GFX_GOL_MeterValueGet(pMtr) (((GFX_GOL_METER*)pMtr)->value)
```

Returns

The current value of the object.

Description

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();

// 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();

    // 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)

1.6.2.1.7.7 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

Syntax

```
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

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)

1.6.2.1.7.8 GFX_GOL_MeterActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_meter.h

Syntax

```
void GFX_GOL_MeterActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.7.9 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

Syntax

```
GFX_GOL_METER * GFX_GOL_MeterCreate(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

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(
            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
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(
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)
```

1.6.2.1.7.10 GFX_GOL_MeterDecrement Function

This function decrements the meter value by the delta value set.

File

gfx_gol_meter.h

Syntax

```
void GFX_GOL_MeterDecrement(GFX_GOL_METER * pObject, uint16_t delta);
```

Returns

None.

Description

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)

1.6.2.1.7.11 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

Syntax

```
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

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)

1.6.2.1.7.12 GFX_GOL_MeterIncrement Function

This function increments the meter value by the delta value set.

File

gfx_gol_meter.h

Syntax

```
void GFX_GOL_MeterIncrement(GFX_GOL_METER * pObject, uint16_t delta);
```

Returns

None.

Description

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)

1.6.2.1.7.13 GFX_GOL_MeterRangeSet Function

This function sets the range of the meter.

File

gfx_gol_meter.h

Syntax

```
void GFX_GOL_MeterRangeSet(GFX_GOL_METER * pObject, int16_t minValue, int16_t maxValue);
```

Returns

None.

Description

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)

1.6.2.1.7.14 GFX_GOL_MeterScaleColorsSet Function

This function sets the arc colors of the object.

File

gfx_gol_meter.h

Syntax

```
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

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)

1.6.2.1.7.15 GFX_GOL_MeterValueSet Function

This function sets the value of the meter.

File

gfx_gol_meter.h

Syntax

```
void GFX_GOL_MeterValueSet(GFX_GOL_METER * pObject, int16_t value);
```

Returns

None.

Description

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();

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();

   // 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)

1.6.2.1.8 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.

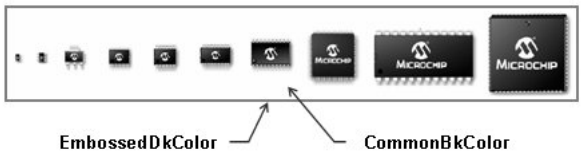
Macros

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 pre-defined 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.



1.6.2.1.8.1 GFX_GOL_PictureControlImageGet Macro

This function gets the image used when in the pressed state.

File

gfx_gol_picture.h

Syntax

```
#define GFX_GOL_PictureControlImageGet(pObject) \
                    (((GFX_GOL_PICTURECONTROL*)pObject)->pImage)
```

Returns

Pointer to the image resource.

Description

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)

1.6.2.1.8.2 GFX_GOL_PictureControlImageSet Macro

This function sets the image to be in the object.

File

gfx_gol_picture.h

Syntax

```
#define GFX_GOL_PictureControlImageSet(pObject, pImage) \
                    (((GFX_GOL_PICTURECONTROL*)pObject)->pImage = pImage)
```

Returns

None.

Description

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(
                    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)

1.6.2.1.8.3 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

Syntax

```
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

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);

1.6.2.1.8.4 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

Syntax

```
GFX_GOL_PICTURECONTROL * GFX_GOL_PictureControlCreate(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

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 parial 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
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(

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)

1.6.2.1.8.5 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

Syntax

```
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

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)

1.6.2.1.8.6 GFX_GOL_PictureControlPartialSet Function

This function sets the partial image parameters to be in the object.

File

gfx_gol_picture.h

Syntax

```
void GFX_GOL_PictureControlPartialSet(GFX_GOL_PICTURECONTROL * pObject, uint16_t xOffset,
uint16_t yOffset, uint16_t partialWidth, uint16_t partialHeight);
```

Returns

None.

Description

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(
                    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)

1.6.2.1.8.7 GFX_GOL_PictureControlScaleSet Function

Sets the scale factor used to render the image used in the object.

File

gfx_gol_picture.h

Syntax

```
void GFX_GOL_PictureControlScaleSet(GFX_GOL_PICTURECONTROL pObject, int8_t scale);
```

Returns

None.

Description

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)

1.6.2.1.9 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

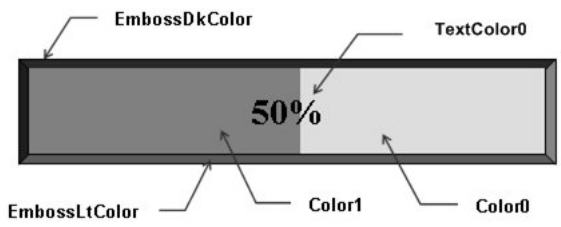
Name	Description
GFX_GOL_ProgressBarPositionGet	This function returns the current position of the progress bar.
GFX_GOL_ProgressBarRangeGet	This function returns the range of the progress bar.

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 pre-defined 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.

1.6.2.1.9.1 GFX_GOL_ProgressBarPositionGet Macro

This function returns the current position of the progress bar.

File

gfx_gol_progress_bar.h

Syntax

```
#define GFX_GOL_ProgressBarPositionGet(pObject) (((GFX_GOL_PROGRESSBAR*)pObject)->pos)
```

Returns

The current position of the scroll bar thumb.

Description

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)

1.6.2.1.9.2 GFX_GOL_ProgressBarRangeGet Macro

This function returns the range of the progress bar.

File

gfx_gol_progress_bar.h

Syntax

```
#define GFX_GOL_ProgressBarRangeGet(pObject) (pObject->range)
```

Returns

The current range used by the object.

Description

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)

1.6.2.1.9.3 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

Syntax

```
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

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);

1.6.2.1.9.4 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

Syntax

```
GFX_GOL_PROGRESSBAR * GFX_GOL_ProgressBarCreate(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

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 = PbCreate(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
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(

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)

1.6.2.1.9.5 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

Syntax

```
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

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)

1.6.2.1.9.6 GFX_GOL_ProgressBarPositionSet Function

This function sets the position of the progress bar.

File

gfx_gol_progress_bar.h

Syntax

```
void GFX_GOL_ProgressBarPositionSet(GFX_GOL_PROGRESSBAR * pObject, uint16_t position);
```

Returns

None.

Description

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)

1.6.2.1.9.7 GFX_GOL_ProgressBarRangeSet Function

This function sets the range of the progress bar.

File

gfx_gol_progress_bar.h

Syntax

```
void GFX_GOL_ProgressBarRangeSet(GFX_GOL_PROGRESSBAR * pObject, uint16_t range);
```

Returns

None.

Description

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)

1.6.2.1.10 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_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_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

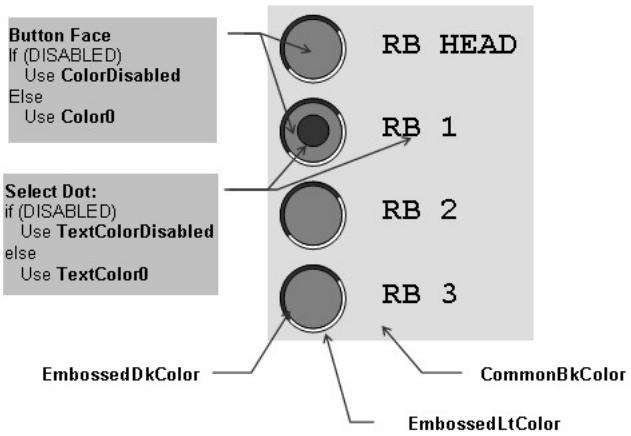
Name	Description
GFX_GOL_RadioButtonTextGet	This function returns the address of the current text string used by the object.

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 pre-defined 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.



When creating the object, the alignment of the text of the object can be formatted with the same options that GFX_TextStringBoxDraw() allows.

1.6.2.1.10.1 GFX_GOL_RadioButtonTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_radio_button.h

Syntax

```
#define GFX_GOL_RadioButtonTextGet(pObject) (((GFX_GOL_RADIOBUTTON *)pObject)->pText)
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.10.2 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

Syntax

```
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

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 objectï¿½s 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)

1.6.2.1.10.3 GFX_GOL_RadioButtonActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_radio_button.h

Syntax

```
void GFX_GOL_RadioButtonActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.10.4 GFX_GOL_RadioButtonCheckGet Function

This function returns the ID of the currently checked radio button in the group.

File

gfx_gol_radio_button.h

Syntax

```
uint16_t GFX_GOL_RadioButtonCheckGet(GFX_GOL_RADIOBUTTON * pObject);
```

Returns

The ID of the member of the group with the check.

Description

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(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(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(ID_RADIOBUTTON3,
                  255,130,310,170,
                  state,
                  label2,
                  GFX_ALIGN_CENTER,
                  pScheme);

// draw the objects
while(GFX_GOL_ObjectListDraw() != 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)

1.6.2.1.10.5 GFX_GOL_RadioButtonCheckSet Function

This function sets the ID of the currently checked radio button in the group.

File

gfx_gol_radio_button.h

Syntax

```
void GFX_GOL_RadioButtonCheckSet(GFX_GOL_RADIOBUTTON * pObject, uint16_t id);
```

Returns

None.

Description

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)

1.6.2.1.10.6 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

Syntax

```
GFX_GOL_RADIOBUTTON * GFX_GOL_RadioButtonCreate(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

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(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(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(ID_RADIOBUTTON3,
                  255,130,310,170,
                  state,
                  label2,
                  GFX_ALIGN_CENTER,
                  pScheme);
```

```
// draw the objects
while(GFX_GOL_ObjectListDraw() != GFX_STATUS_SUCCESS);
```

Parameters

Parameters	Description
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(
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)
```

1.6.2.1.10.7 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

Syntax

```
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

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)

1.6.2.1.10.8 GFX_GOL_RadioButtonTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_radio_button.h

Syntax

```
GFX_ALIGNMENT GFX_GOL_RadioButtonTextAlignmentGet(GFX_GOL_RADIOBUTTON * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

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)

1.6.2.1.10.9 GFX_GOL_RadioButtonTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_radio_button.h

Syntax

```
void GFX_GOL_RadioButtonTextAlignmentSet(GFX_GOL_RADIOBUTTON * pObject, GFX_ALIGNMENT
align);
```

Returns

None.

Description

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)

1.6.2.1.10.10 GFX_GOL_RadioButtonTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_radio_button.h

Syntax

```
void GFX_GOL_RadioButtonTextSet(GFX_GOL_RADIOBUTTON * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

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)

1.6.2.1.11 Scroll Bar Object

Scroll Bar is an object that can be used to display a value or scrolling location in a predefined area.

Functions

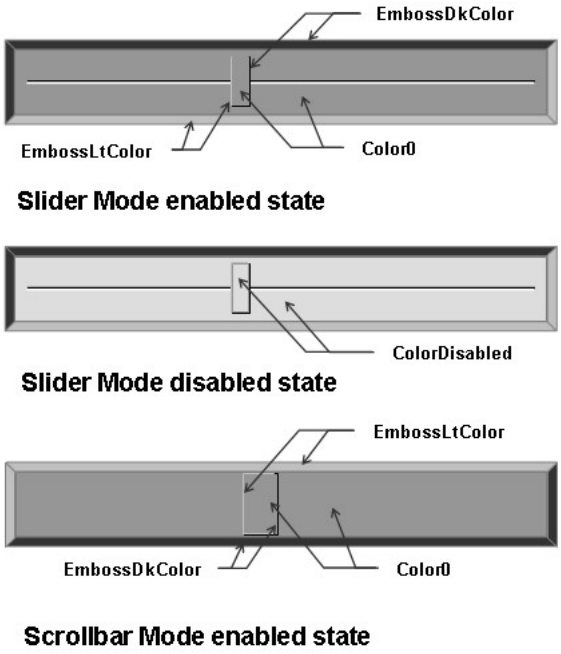
	Name	Description
	GFX_GOL_ScrollBarActionGet	This function evaluates the message from a user if the message will affect the object or not.
	GFX_GOL_ScrollBarActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_ScrollBarCreate	This function creates a 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_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.

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 pre-defined 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.



1.6.2.1.11.1 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

Syntax

```
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

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 objectï¿½s 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 objectï¿½s 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)

1.6.2.1.11.2 GFX_GOL_ScrollBarActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_scroll_bar.h

Syntax

```
void GFX_GOL_ScrollBarActionSet(GFX_GOL_TRANSLATED_ACTION translatedMsg, void * pObject,
GFX_GOL_MESSAGE * pMessage);
```

Returns

None.

Description

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)

1.6.2.1.11.3 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

Syntax

```
GFX_GOL_SCROLLBAR * GFX_GOL_ScrollBarCreate(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

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(  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(  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(  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
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(
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);
```

1.6.2.1.11.4 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

Syntax

```
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

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)

1.6.2.1.11.5 GFX_GOL_ScrollBarPageGet Function

This function returns the page size of the object.

File

gfx_gol_scroll_bar.h

Syntax

```
uint16_t GFX_GOL_ScrollBarPageGet(GFX_GOL_SCROLLBAR * pObject);
```

Returns

The page size of the object.

Description

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)

1.6.2.1.11.6 GFX_GOL_ScrollBarPageSet Function

This function sets the page size of the object.

File

gfx_gol_scroll_bar.h

Syntax

```
void GFX_GOL_ScrollBarPageSet(GFX_GOL_SCROLLBAR * pObject, uint16_t page);
```

Returns

None.

Description

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)

1.6.2.1.11.7 GFX_GOL_ScrollBarPositionDecrement Function

This function decrements the scroll bar position by the delta change (page) value set.

File

gfx_gol_scroll_bar.h

Syntax

```
void GFX_GOL_ScrollBarPositionDecrement(GFX_GOL_SCROLLBAR * pObject);
```

Returns

None.

Description

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)

1.6.2.1.11.8 GFX_GOL_ScrollBarPositionGet Function

This function returns the current position of the scroll bar thumb.

File

gfx_gol_scroll_bar.h

Syntax

```
uint16_t GFX_GOL_ScrollBarPositionGet(GFX_GOL_SCROLLBAR * pObject);
```

Returns

The current position of the scroll bar thumb.

Description

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();

// 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();

    // 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)

1.6.2.1.11.9 GFX_GOL_ScrollBarPositionIncrement Function

This function increments the scroll bar position by the delta change (page) value set.

File

gfx_gol_scroll_bar.h

Syntax

```
void GFX_GOL_ScrollBarPositionIncrement(GFX_GOL_SCROLLBAR * pObject);
```

Returns

None.

Description

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)

1.6.2.1.11.10 GFX_GOL_ScrollBarPositionSet Function

This function sets the position of the scroll bar thumb.

File

gfx_gol_scroll_bar.h

Syntax

```
void GFX_GOL_ScrollBarPositionSet(GFX_GOL_SCROLLBAR * pObject, uint16_t position);
```

Returns

None.

Description

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();

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();

       // 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)

1.6.2.1.11.11 GFX_GOL_ScrollBarRangeGet Function

This function returns the range of the thumb of the scroll bar.

File

gfx_gol_scroll_bar.h

Syntax

```
uint16_t GFX_GOL_ScrollBarRangeGet(GFX_GOL_SCROLLBAR * pObject);
```

Returns

The range of the scroll bar.

Description

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)

1.6.2.1.11.12 GFX_GOL_ScrollBarRangeSet Function

This function sets the range of the thumb of the scroll bar.

File

gfx_gol_scroll_bar.h

Syntax

```
void GFX_GOL_ScrollBarRangeSet(GFX_GOL_SCROLLBAR * pObject, uint16_t range);
```

Returns

None.

Description

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() 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)

1.6.2.1.12 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

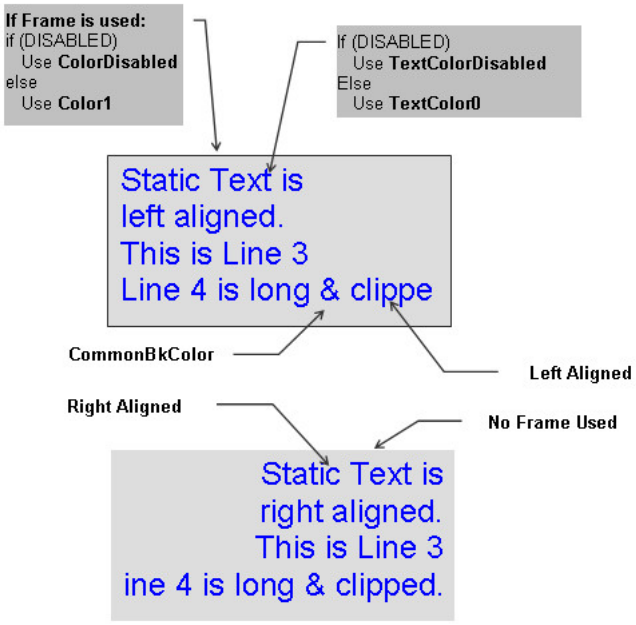
	Name	Description
	GFX_GOL_StaticTextActionGet	This function sets the text alignment of the text string used by the object.
	GFX_GOL_StaticTextAlignmentGet	This function returns the text alignment of the text string used by the object.
	GFX_GOL_StaticTextAlignmentSet	This function sets the text alignment 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 pre-defined 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.



Static Text can be rendered with an option to have a frame that will show the boundaries of the defined region (see GFX_GOL_STATICTEXT_FRAME_STATE for details).

Another option for the object is to render the object with a background. The types of background that can be used is described in the style scheme used in the object (see GFX_BACKGROUND_TYPE for types of available background).

When creating the object, the alignment of the text of the object can be formatted with the same options that GFX_TextStringBoxDraw() allows.

1.6.2.1.12.1 GFX_GOL_StaticTextActionGet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_static_text.h

Syntax

```
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

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);

1.6.2.1.12.2 GFX_GOL_StaticTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_static_text.h

Syntax

```
GFX_ALIGNMENT GFX_GOL_StaticTextAlignmentGet(GFX_GOL_STATICTEXT * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

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)

1.6.2.1.12.3 GFX_GOL_StaticTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_static_text.h

Syntax

```
void GFX_GOL_StaticTextAlignmentSet(GFX_GOL_STATICTEXT * pObject, GFX_ALIGNMENT align);
```

Returns

None.

Description

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)

1.6.2.1.12.4 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

Syntax

```
GFX_GOL_STATICTEXT * GFX_GOL_StaticTextCreate(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

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(
    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
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(
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)
```

1.6.2.1.12.5 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

Syntax

```
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

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)

1.6.2.1.12.6 GFX_GOL_StaticTextGet Function

This function returns the address of the current text string used by the object.

File

gfx_gol_static_text.h

Syntax

```
GFX_XCHAR * GFX_GOL_StaticTextGet(GFX_GOL_STATICTEXT * pObject);
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.12.7 GFX_GOL_StaticTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_static_text.h

Syntax

```
void GFX_GOL_StaticTextSet(GFX_GOL_STATICTEXT * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

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)

1.6.2.1.13 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.

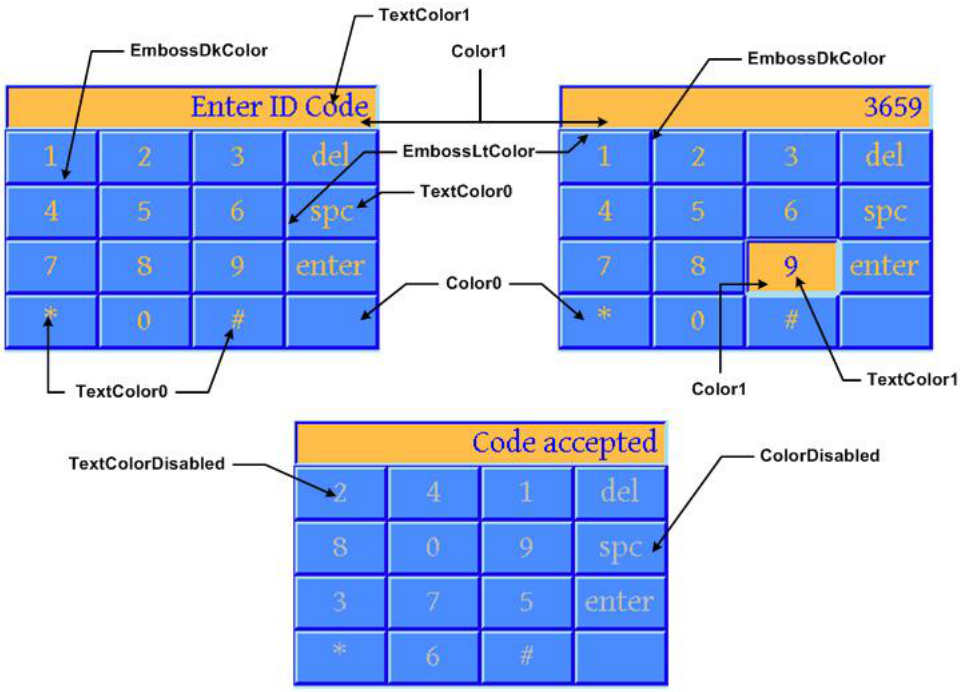
	GFX_GOL_TextEntryActionSet	This function performs the state change of the object based on the translated action.
	GFX_GOL_TextEntryBufferClear	This function will clear the data in the display.
	GFX_GOL_TextEntryBufferGet	This function returns the buffer used to display text.
	GFX_GOL_TextEntryBufferSet	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.

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 pre-defined 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.

1.6.2.1.13.1 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

Syntax

```
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

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);

1.6.2.1.13.2 GFX_GOL_TextEntryActionSet Function

This function performs the state change of the object based on the translated action.

File

gfx_gol_text_entry.h

Syntax

```
void GFX_GOL_TextEntryActionSet(uint16_t translatedMsg, void * pObject, GFX_GOL_MESSAGE *
```

```
pMessage);
```

Returns

None.

Description

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)

1.6.2.1.13.3 GFX_GOL_TextEntryBufferClear Function

This function will clear the data in the display.

File

gfx_gol_text_entry.h

Syntax

```
void GFX_GOL_TextEntryBufferClear(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

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)

1.6.2.1.13.4 GFX_GOL_TextEntryBufferGet Function

This function returns the buffer used to display text.

File

gfx_gol_text_entry.h

Syntax

```
GFX_XCHAR * GFX_GOL_TextEntryBufferGet(GFX_GOL_TEXTENTRY * pObject);
```

Returns

The pointer to the text buffer used to display text.

Description

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)

1.6.2.1.13.5 GFX_GOL_TextEntryBufferSet Function

This function sets the buffer used to display text.

File

gfx_gol_text_entry.h

Syntax

```
void GFX_GOL_TextEntryBufferSet(GFX_GOL_TEXTENTRY * pObject, GFX_XCHAR * pText, uint16_t
MaxSize);
```

Returns

None.

Description

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)

1.6.2.1.13.6 GFX_GOL_TextEntryCharAdd Function

This function will insert a character to the end of the buffer.

File

gfx_gol_text_entry.h

Syntax

```
void GFX_GOL_TextEntryCharAdd(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

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)

1.6.2.1.13.7 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

Syntax

```
GFX_GOL_TEXTENTRY * GFX_GOL_TextEntryCreate(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

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(
    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
                                    // diplayed text
    pScheme);                       // use given scheme
```

Parameters

Parameters	Description
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(
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);
```

1.6.2.1.13.8 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

Syntax

```
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

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)

1.6.2.1.13.9 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

Syntax

```
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

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)

1.6.2.1.13.10 GFX_GOL_TextEntryKeyCommandSet Function

This function will assign a command to a key with the given index.

File

gfx_gol_text_entry.h

Syntax

```
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

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)

1.6.2.1.13.11 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

Syntax

```
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

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)

1.6.2.1.13.12 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

Syntax

```
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 succesfully.

Description

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[])

1.6.2.1.13.13 GFX_GOL_TextEntryKeyMemberListDelete Function

This function will delete the key member list assigned to the object.

File

gfx_gol_text_entry.h

Syntax

```
void GFX_GOL_TextEntryKeyMemberListDelete(void * pObject);
```

Returns

None.

Description

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)

1.6.2.1.13.14 GFX_GOL_TextEntryKeyTextSet Function

This function will set the text assigned to a key with the given index.

File

gfx_gol_text_entry.h

Syntax

```
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

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)

1.6.2.1.13.15 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

Syntax

```
void GFX_GOL_TextEntryLastCharDelete(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

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)

1.6.2.1.13.16 GFX_GOL_TextEntrySpaceCharAdd Function

This function will insert a space character to the end of the buffer.

File

gfx_gol_text_entry.h

Syntax

```
void GFX_GOL_TextEntrySpaceCharAdd(GFX_GOL_TEXTENTRY * pObject);
```

Returns

None.

Description

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)

1.6.2.1.14 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

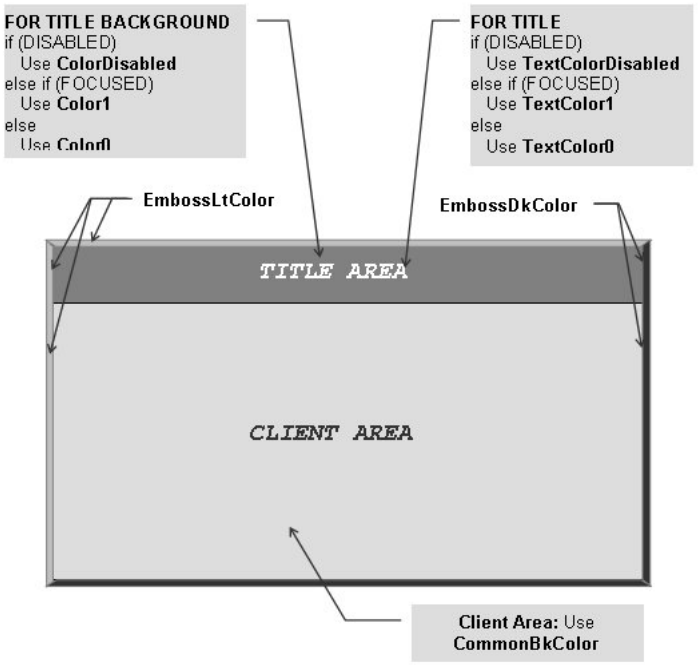
Name	Description
GFX_GOL_WindowImageGet	This function gets the image used.
GFX_GOL_WindowImageSet	This function sets the image used in the object.
GFX_GOL_WindowTextGet	This function returns the address of the current text string used by the object.

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 pre-defined 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.

When creating the object, the alignment of the text of the object in the title area can be formatted with the same options that GFX_TextStringBoxDraw() allows.

1.6.2.1.14.1 GFX_GOL_WindowImageGet Macro

This function gets the image used.

File

gfx_gol_window.h

Syntax

```
#define GFX_GOL_WindowImageGet(pObject, pImage) \
                                (((GFX_GOL_WINDOW *)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_WindowImageGet(
                    GFX_GOL_WINDOW *pObject)
```

1.6.2.1.14.2 GFX_GOL_WindowImageSet Macro

This function sets the image used in the object.

File

gfx_gol_window.h

Syntax

```
#define GFX_GOL_WindowImageSet(pObject, pImage) \
                                    (((GFX_GOL_WINDOW *)pObject)->pImage = pImage)
```

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_WindowImageSet(

GFX_GOL_WINDOW *pObject,

GFX_RESOURCE_HDR *pImage)

1.6.2.1.14.3 GFX_GOL_WindowTextGet Macro

This function returns the address of the current text string used by the object.

File

gfx_gol_window.h

Syntax

```
#define GFX_GOL_WindowTextGet(pObject) (((GFX_GOL_WINDOW *)pObject)->pText)
```

Returns

Pointer to text string.

Description

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)

1.6.2.1.14.4 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

Syntax

```
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

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);

1.6.2.1.14.5 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

Syntax

```
GFX_GOL_WINDOW * GFX_GOL_WindowCreate(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

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(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
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(

```
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)
```

1.6.2.1.14.6 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

Syntax

```
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

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)

1.6.2.1.14.7 GFX_GOL_WindowTextAlignmentGet Function

This function returns the text alignment of the text string used by the object.

File

gfx_gol_window.h

Syntax

```
GFX_ALIGNMENT GFX_GOL_WindowTextAlignmentGet(GFX_GOL_WINDOW * pObject);
```

Returns

The text alignment set in the object. See GFX_ALIGNMENT for more details.

Description

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)
```

1.6.2.1.14.8 GFX_GOL_WindowTextAlignmentSet Function

This function sets the text alignment of the text string used by the object.

File

gfx_gol_window.h

Syntax

```
void GFX_GOL_WindowTextAlignmentSet(GFX_GOL_WINDOW * pObject, GFX_ALIGNMENT alignment);
```

Returns

None.

Description

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)

1.6.2.1.14.9 GFX_GOL_WindowTextSet Function

This function sets the address of the current text string used by the object.

File

gfx_gol_window.h

Syntax

```
void GFX_GOL_WindowTextSet(GFX_GOL_WINDOW * pObject, GFX_XCHAR * pText);
```

Returns

None.

Description

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)

1.6.2.2 GOL Object States

Objects rendered on the display are based on their current Property States and the Drawing 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.

1.6.2.2.1 GFX_GOL_ObjectStateClear Macro

This function clears the state bits of the given object.

File

gfx_gol.h

Syntax

```
#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

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);

1.6.2.2.2 GFX_GOL_ObjectStateGet Macro

This function retrieves the current value of the state bits of an object.

File

gfx_gol.h

Syntax

```
#define GFX_GOL_ObjectStateGet(pObject, stateBits) \
                (((GFX_GOL_OBJ_HEADER*)pObject)->state & stateBits)
```

Returns

The current status of the specified state bits.

Description

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);

1.6.2.2.3 GFX_GOL_ObjectStateSet Macro

This function sets the state bits of the given object.

File

gfx_gol.h

Syntax

```
#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

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);
```

1.6.2.3 GOL Object Management

API for managing objects and list of objects.

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.

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.

1.6.2.3.1 GFX_GOL_ObjectAdd Function

This function adds an object to the tail of the currently active list.

File

gfx_gol.h

Syntax

```
void GFX_GOL_ObjectAdd(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

None.

Description

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(pObjtoMove);      // add object to active list
}
```

Function

void GFX_GOL_ObjectAdd(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.3.2 GFX_GOL_ObjectByIDDelete Function

This function removes an object with the given user defined ID from the currently active list.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectByIDDelete(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

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.

Function

GFX_STATUS GFX_GOL_ObjectByIDDelete(uint16_t id)

1.6.2.3.3 GFX_GOL_ObjectCanBeFocused Function

Checks if the object can be focused.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectCanBeFocused(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

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
pObject	pointer to the object.

Function

GFX_STATUS GFX_GOL_ObjectCanBeFocused(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.3.4 GFX_GOL_ObjectDelete Function

This function removes an object from the currently active list.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectDelete(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

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.

Function

GFX_STATUS GFX_GOL_ObjectDelete(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.3.5 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

Syntax

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFind(uint16_t ID);
```

Returns

The pointer to the object in the list with the given ID.

Description

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(ID);

    if (pTemp != NULL)
    {
        // destination as active list
        GFX_GOL_ObjectSetList(pDstList);

        // add object to active list
        GFX_GOL_ObjectAdd(pTemp);
    }
}
```

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFind(uint16_t ID)

1.6.2.3.6 GFX_GOL_ObjectFocusGet Function

This function returns the pointer to the object that is currently receiving keyboard input (or focused).

File

gfx_gol.h

Syntax

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFocusGet();
```

Returns

The pointer the currently focused object. Returns NULL if there is no object currently set.

Description

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.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFocusGet(void)

1.6.2.3.7 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

Syntax

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFocusNextGet();
```

Returns

The pointer to the object that can be focused.

Description

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.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFocusNextGet(void)

1.6.2.3.8 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

Syntax

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectFocusPrevGet();
```

Returns

The pointer to the object that can be focused.

Description

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.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectFocusPrevGet(void)

1.6.2.3.9 GFX_GOL_ObjectFocusSet Function

This function sets the object to be focused.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectFocusSet(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

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
pObject	pointer to the object.

Function

GFX_STATUS *GFX_GOL_ObjectFocusSet(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.3.10 GFX_GOL_ObjectIDGet Function

This function returns the object ID.

File

gfx_gol.h

Syntax

```
uint16_t GFX_GOL_ObjectIDGet(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

The user defined ID of the object.

Description

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)

1.6.2.3.11 GFX_GOL_ObjectListFree Function

This function sets the active list to the new list.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectListFree();
```

Returns

GFX_STATUS_SUCCESS - is returned if the free was successful. GFX_STATUS_FAILURE - is returned if the free was not successful.

Description

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();

        // set list as active list
        GFX_GOL_ObjectListSet(pPage);

        // pPage objects are deleted
GFX_GOL_ObjectListFree();

        // restore the active list
GFX_GOL_ObjectListSet(pTemp);
    }
```

Function

GFX_STATUS GFX_GOL_ObjectListFree(void)

1.6.2.3.12 GFX_GOL_ObjectListGet Function

This function returns the current active list.

File

gfx_gol.h

Syntax

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectListGet();
```

Returns

Pointer (type GFX_GOL_OBJ_HEADER) to the current active list.

Description

This function returns the pointer to the current active.

Preconditions

None.

Example

See GFX_GOL_ObjectListNew() for example code.

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectListGet(void)

1.6.2.3.13 GFX_GOL_ObjectListNew Function

This function removes an object with the given user defined ID from the currently active list.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectListNew();
```

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

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();

// start the new list, after the start of the list, the
// current active list is empty.
GFX_GOL_ObjectListNew();

// assume that objects are already created
// you can now add objects to the new list
GFX_GOL_ObjectAdd(pButton);
GFX_GOL_ObjectAdd(pWindow);
GFX_GOL_ObjectAdd(pSlider);
```

Function

GFX_STATUS GFX_GOL_ObjectListNew(void)

1.6.2.3.14 GFX_GOL_ObjectListSet Function

This function sets the active list to the new list.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectListSet(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

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();

// start the new list
GFX_GOL_ObjectListNew();

// you can now add objects to the current list
// assume that objects are already created
GFX_GOL_ObjectAdd(pButton);
GFX_GOL_ObjectAdd(pWindow);
GFX_GOL_ObjectAdd(pSlider);

// do something here on the new list

// return the old list
GOLSetList(pSave);
```

Function

GFX_STATUS *GFX_GOL_ObjectListSet(GFX_GOL_OBJ_HEADER *pList)

1.6.2.3.15 GFX_GOL_ObjectNextGet Function

This function returns the pointer to next object in the list after the specified object.

File

gfx_gol.h

Syntax

```
GFX_GOL_OBJ_HEADER * GFX_GOL_ObjectNextGet(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

The pointer to the next object in the list.

Description

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();

        while(pCurr->pNxtObj != NULL)
        {
            // just select button objects and redraw them
            if (GFX_GOL_ObjectTypeGet(pCurr) == BUTTON)
            {
                // set to be redrawn
                pCurr->state = BTN_DRAW;
            }
            pCurr = GFX_GOL_ObjectNextGet(pCurr);
        }
        // redraw all buttons in the active list
        GFX_GOL_ObjectListDraw();
    }
```

Function

GFX_GOL_OBJ_HEADER *GFX_GOL_ObjectNextGet(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.3.16 GFX_GOL_ObjectStyleSchemeGet Macro

This function returns the style scheme currently set for the object.

File

gfx_gol.h

Syntax

```
#define GFX_GOL_ObjectStyleSchemeGet(pObject) \
                (((GFX_GOL_OBJ_HEADER*)pObject)->pGolScheme)
```

Returns

The pointer to the currently set style scheme.

Description

This function returns the style scheme surrently 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)

1.6.2.3.17 GFX_GOL_ObjectStyleSchemeSet Macro

This function sets the style scheme of the object.

File

gfx_gol.h

Syntax

```
#define GFX_GOL_ObjectStyleSchemeSet(pObject, pStyle) \
                ((((GFX_GOL_OBJ_HEADER*)pObject)->pGolScheme) = pStyle)
```

Returns

None.

Description

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)

1.6.2.3.18 GFX_GOL_ObjectTypeGet Function

This function returns the object type.

File

gfx_gol.h

Syntax

```
GFX_GOL_OBJ_TYPE GFX_GOL_ObjectTypeGet(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

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.

Function

GFX_GOL_OBJ_TYPE GFX_GOL_ObjectTypeGet(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.4 GOL Object Rendering

API to render objects in the frame buffer.

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_ObjectDrawDisable	This function resets the drawing state bits of the object.
	GFX_GOL_ObjectDrawEnable	This function sets the object to be redraw.
	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.

Types

Name	Description
GFX_GOL_DRAW_CALLBACK_FUNC	Draw callback function definition. This application defined function allows the application to perform application specific rendering.

1.6.2.4.1 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

Syntax

```
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

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);

1.6.2.4.2 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

Syntax

```
void GFX_GOL_DrawCallbackSet(GFX_GOL_DRAW_CALLBACK_FUNC pFunc);
```

Returns

None.

Description

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
pFunc	pointer to the draw callback function.

Function

void GFX_GOL_DrawCallbackSet(

GFX_GOL_DRAW_CALLBACK_FUNC pFunc)

1.6.2.4.3 GFX_GOL_ObjectDrawDisable Function

This function resets the drawing state bits of the object.

File

gfx_gol.h

Syntax

```
void GFX_GOL_ObjectDrawDisable(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

None.

Description

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
pObject	pointer to the object.

Function

void GFX_GOL_ObjectDrawDisable(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.4.4 GFX_GOL_ObjectDrawEnable Function

This function sets the object to be redraw.

File

gfx_gol.h

Syntax

```
void GFX_GOL_ObjectDrawEnable(GFX_GOL_OBJ_HEADER * pObject);
```

Returns

None.

Description

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

```
void GOLRedrawRec(uint16_t left, uint16_t top,
                  uint16_t right, uint16_t bottom)
{

    GFX_GOL_OBJ_HEADER  *pCurrentObj;
    int                 overlapX, overlapY;

    pCurrentObj = _pGolObjects;

    while(pCurrentObj != NULL)
    {
        overlapX = overlapY = 0;

        // check overlaps in x direction
        if (((pCurrentObj->left <= left)  && (pCurrentObj->right >= left))  ||
            ((pCurrentObj->left <= right) && (pCurrentObj->right >= right)) ||
            ((pCurrentObj->left <= left)  && (pCurrentObj->right >= right)) ||
            ((pCurrentObj->left >= left)  && (pCurrentObj->right <= right))
          )
            overlapX = 1;

        // check overlaps in y direction
        if (((pCurrentObj->top <= top)    && (pCurrentObj->bottom >= top))    ||
            ((pCurrentObj->top <= bottom) && (pCurrentObj->bottom >= bottom)) ||
            ((pCurrentObj->top <= top)    && (pCurrentObj->bottom >= bottom)) ||
            ((pCurrentObj->top >= top)    && (pCurrentObj->bottom <= bottom))
          )
            overlapY = 1;

        // when any portion of the widget is touched by the defined rectangle the
        // x and y overlaps will exist.
        if (overlapX & overlapY) {
            GFX_GOL_ObjectRedraw(pCurrentObj);
        }

        pCurrentObj = (GFX_GOL_OBJ_HEADER *)pCurrentObj->pNxtObj;
    }   //end of while

}
```

Parameters

Parameters	Description
pObject	pointer to the object that will be redrawn.

Function

void GFX_GOL_ObjectDrawEnable(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.4.5 GFX_GOL_ObjectIsRedrawSet Function

This function checks if the object needs to be redrawn or not.

File

gfx_gol.h

Syntax

```
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

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();
    }

    while(pCurrentObj != NULL)
    {
        if(GFX_GOL_ObjectIsRedrawSet(pCurrentObj) == true)
        {
            done = pCurrentObj->draw(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
pObject	pointer to the object that will be checked.

Function

bool GFX_GOL_ObjectIsRedrawSet(

GFX_GOL_OBJ_HEADER *pObject)

1.6.2.4.6 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

Syntax

```
GFX_STATUS GFX_GOL_ObjectListDraw();
```

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

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() == 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(&msg);
        }
    }
```

Function

GFX_STATUS GFX_GOL_ObjectListDraw(void)

1.6.2.4.7 GFX_GOL_ObjectListHide Function

This function marks all objects in the active list to be hidden.

File

gfx_gol.h

Syntax

```
void GFX_GOL_ObjectListHide();
```

Returns

None.

Description

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.

Function

void GFX_GOL_ObjectListHide(void)

1.6.2.4.8 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

Syntax

```
void GFX_GOL_ObjectRectangleRedraw(uint16_t left, uint16_t top, uint16_t right, uint16_t
bottom);
```

Returns

None.

Description

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
GOLRedrawRec(10,10,100,100);

// save the current active list
pTemp = pAllObjects;

// reset active list
GFX_GOL_ObjectListNew();

// 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(pTemp);

        pTemp = pTemp->pNxtObj;
    }

     // redraw active list
    GFX_GOL_ObjectListDraw();
```

Parameters

Parameters	Description
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(

uint16_t left,

uint16_t top,

uint16_t right,

uint16_t bottom)

1.6.2.5 GOL Object Panel Rendering

These are the API to render panels of objects in the frame buffer. These functions are used internally by the object rendering functions. Application should not call these functions directly. For those application that creates their own custom objects that uses panels, these function should be used to render the panels.

Functions

	Name	Description
	GFX_GOL_ObjectBackGroundSet	This function sets the background information.
	GFX_GOL_ObjectHideDraw	This function performs the hiding of an object from the screen.
	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_TwoTonePanelDraw	This function renders the two-tone panel.

1.6.2.5.1 GFX_GOL_ObjectBackGroundSet Function

This function sets the background information.

File

gfx_gol.h

Syntax

```
void GFX_GOL_ObjectBackGroundSet(GFX_GOL_OBJ_HEADER * pObjectHeader);
```

Returns

None.

Description

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.

Function

GFX_STATUS GFX_GOL_ObjectBackGroundSet(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.5.2 GFX_GOL_ObjectHideDraw Function

This function performs the hiding of an object from the screen.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_ObjectHideDraw(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

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.

Function

GFX_STATUS GFX_GOL_ObjectHideDraw(GFX_GOL_OBJ_HEADER *pObject)

1.6.2.5.3 GFX_GOL_PanelAlphaParameterSet Function

This function sets the alpha blending value when using alpha blending in panels.

File

gfx_gol.h

Syntax

```
void GFX_GOL_PanelAlphaParameterSet(uint16_t alphaValue);
```

Returns

None.

Description

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
alphaVlaue	the alpha value used to render alpha blended panels.

Function

void GFX_GOL_PanelAlphaParameterSet(

uint16_t alphaValue)

1.6.2.5.4 GFX_GOL_PanelBackgroundSet Function

This function sets panel background information.

File

gfx_gol.h

Syntax

```
void GFX_GOL_PanelBackgroundSet(GFX_GOL_OBJ_HEADER * pObjectHeader);
```

Returns

None.

Description

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
pObjectHeader	the object header of the object that needs to draw a panel with background.

Function

void GFX_GOL_PanelBackgroundSet(GFX_GOL_OBJ_HEADER *pObjectHeader)

1.6.2.5.5 GFX_GOL_PanelDraw Function

This function renders the panel.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_PanelDraw();
```

Returns

GFX_STATUS_SUCCESS - when the panel rendering is done. GFX_STATUS_FAILURE - when the panel rendering is not

yet done.

Description

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.

Function

GFX_STATUS GFX_GOL_PanelDraw(void)

1.6.2.5.6 GFX_GOL_PanelGradientParameterSet Function

This function sets the gradient fill start and end colors of a panel.

File

gfx_gol.h

Syntax

```
void GFX_GOL_PanelGradientParameterSet(GFX_COLOR startColor, GFX_COLOR endColor);
```

Returns

None.

Description

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
startColor	the gradient fill start color.
endColor	the gradient fill end color.

Function

void GFX_GOL_PanelGradientParameterSet(

GFX_COLOR startColor,

GFX_COLOR endColor)

1.6.2.5.7 GFX_GOL_PanelParameterSet Function

This function sets the parameters to draw a panel.

File

gfx_gol.h

Syntax

```
void GFX_GOL_PanelParameterSet(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

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
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(

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)

1.6.2.5.8 GFX_GOL_TwoTonePanelDraw Function

This function renders the two-tone panel.

File

gfx_gol.h

Syntax

```
GFX_STATUS GFX_GOL_TwoTonePanelDraw();
```

Returns

GFX_STATUS_SUCCESS - when the panel rendering is done. GFX_STATUS_FAILURE - when the panel rendering is not yet done.

Description

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.

Function

GFX_STATUS GFX_GOL_TwoTonePanelDraw(void)

1.6.2.6 GOL Object Messaging

The library provides an interface to accept messages from the input devices.

Functions

	Name	Description
	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_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 behaviour or user defined behaviour.

Types

Name	Description
GFX_GOL_MESSAGE_CALLBACK_FUNC	Message callback function definition. This application defined function allows the application to perform application specific processing of user messsages.

1.6.2.6.1 GFX_GOL_MESSAGE_CALLBACK_FUNC Type

Message callback function definition. This application defined function allows the application to perform application specific processing of user messsages.

File

gfx_gol.h

Syntax

```
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

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 *

);

1.6.2.6.2 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

Syntax

```
void GFX_GOL_MessageCallbackSet(GFX_GOL_MESSAGE_CALLBACK_FUNC pFunc);
```

Returns

None.

Description

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
pFunc	pointer to the message callback function.

Function

void GFX_GOL_MessageCallbackSet(

GFX_GOL_MESSAGE_CALLBACK_FUNC pFunc)

1.6.2.6.3 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 behaviour or user defined behaviour.

File

gfx_gol.h

Syntax

```
void GFX_GOL_ObjectMessage(GFX_GOL_MESSAGE * pMsg);
```

Returns

None.

Description

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(&msg);
        }
    }
```

Parameters

Parameters	Description
pMessage	Pointer to the message from user.

Function

void GFX_GOL_ObjectMessage(GFX_GOL_MESSAGE *pMessage)

1.6.2.7 Data Types and Constants

The following are the types and constants used in the Graphics Object Layer.

Types

Name	Description
GFX_GOL_BUTTON	Defines the structure used for the Button object.
GFX_GOL_BUTTON_STATE	Specifies the different states of the Button object.
GFX_GOL_CHECKBOX	Defines the structure used for the Check Box object.
GFX_GOL_CHECKBOX_STATE	Specifies the different states of the Check Box object.
GFX_GOL_COMMON_STATE_BITS	Common Object States.
GFX_GOL_DIGITALMETER	Defines the structure used for the Digital Meter object.
GFX_GOL_DIGITALMETER_STATE	Specifies the different states of the Digital Meter object.
GFX_GOL_EDITBOX	Defines the structure used for the Edit Box object.
GFX_GOL_EDITBOX_STATE	Specifies the different states of the Edit Box object.
GFX_GOL_GROUPBOX	Defines the structure used for the Group Box object.
GFX_GOL_GROUPBOX_STATE	Specifies the different states of the Group Box object.
GFX_GOL_LISTBOX	Defines the structure used for the List 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_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_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_HEADER	Specifies Graphics Object Layer structure used in objects.
GFX_GOL_OBJ_TYPE	Specifies the different object types used in the library.
GFX_GOL_PICTURECONTROL	Defines the structure used for the Picture Control object.
GFX_GOL_PICTURECONTROLCONTROL_STATE	Specifies the different states of the Picture Control object.
GFX_GOL_PROGRESSBAR	Defines the structure used for the Progress Bar object.
GFX_GOL_PROGRESSBAR_STATE	Specifies the different states of the Progress Bar object.
GFX_GOL_RADIOBUTTON	Defines the structure used for the Radio Button object.
GFX_GOL_RADIOBUTTON_STATE	Specifies the different states of the Radio Button object.
GFX_GOL_SCROLLBAR	Defines the structure used for the Scroll Bar object.

GFX_GOL_SCROLLBAR_STATE	Specifies the different states of the Scroll Bar object.
GFX_GOL_STATICTEXT	Defines the structure used for the Static Text object.
GFX_GOL_STATICTEXT_STATE	Specifies the different states of the Static Text object.
GFX_GOL_TEXTENTRY	Defines the structure used for the Text Entry object.
GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE	Specifies the different commands available for command keys of the Text Entry object.
GFX_GOL_TEXTENTRY_KEYMEMBER	Defines the structure used to describe a key in 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	Defines the structure used for the Window object.
GFX_GOL_WINDOW_STATE	Specifies the different states of the Window object.
GOL_PANEL_PARAM	Specifies panel parameters.
INPUT_DEVICE_EVENT	Specifies the different user input device events supported in the library.
INPUT_DEVICE_TYPE	Specifies the different user input devices supported in the library.

1.6.2.7.1 GFX_GOL_BUTTON Type

Defines the structure used for the Button object.

File

gfx_gol_button.h

Syntax

```
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

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.

1.6.2.7.2 GFX_GOL_BUTTON_STATE Type

Specifies the different states of the Button object.

File

gfx_gol_button.h

Syntax

```
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

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.

1.6.2.7.3 GFX_GOL_CHECKBOX Type

Defines the structure used for the Check Box object.

File

gfx_gol_check_box.h

Syntax

```
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

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.

1.6.2.7.4 GFX_GOL_CHECKBOX_STATE Type

Specifies the different states of the Check Box object.

File

gfx_gol_check_box.h

Syntax

```
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

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.

1.6.2.7.5 GFX_GOL_COMMON_STATE_BITS Type

Common Object States.

File

gfx_gol.h

Syntax

```
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

Typedef: Common Object States

The following macros defines the common Object State bits.

Remarks

None.

1.6.2.7.6 GFX_GOL_DIGITALMETER Type

Defines the structure used for the Digital Meter object.

File

gfx_gol_digital_meter.h

Syntax

```
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;	Pre-computed 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

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.

1.6.2.7.7 GFX_GOL_DIGITALMETER_STATE Type

Specifies the different states of the Digital Meter object.

File

gfx_gol_digital_meter.h

Syntax

```
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

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.

1.6.2.7.8 GFX_GOL_EDITBOX Type

Defines the structure used for the Edit Box object.

File

gfx_gol_edit_box.h

Syntax

```
typedef struct {
  GFX_GOL_OBJ_HEADER hdr;
  uint16_t textHeight;
  GFX_XCHAR * pText;
  GFX_ALIGNMENT alignment;
  uint16_t charMax;
  uint16_t length;
} GFX_GOL_EDITBOX;
```

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 buffer.
GFX_ALIGNMENT alignment;	text alignment
uint16_t charMax;	Maximum number of characters in the edit box.
uint16_t length;	Current text length.

Description

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.

1.6.2.7.9 GFX_GOL_EDITBOX_STATE Type

Specifies the different states of the Edit Box object.

File

gfx_gol_edit_box.h

Syntax

```
typedef enum {
  GFX_GOL_EDITBOX_FOCUSED_STATE,
  GFX_GOL_EDITBOX_DISABLED_STATE,
  GFX_GOL_EDITBOX_ENABLE_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_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

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.

1.6.2.7.10 GFX_GOL_GROUPBOX Type

Defines the structure used for the Group Box object.

File

gfx_gol_group_box.h

Syntax

```
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;	Pre-computed text width.
uint16_t textHeight;	Pre-computed text height.
GFX_XCHAR * pText;	Text string used.
GFX_ALIGNMENT alignment;	text alignment

Description

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.

1.6.2.7.11 GFX_GOL_GROUPBOX_STATE Type

Specifies the different states of the Group Box object.

File

gfx_gol_group_box.h

Syntax

```
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

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.

1.6.2.7.12 GFX_GOL_LISTBOX Type

Defines the structure used for the List Box object.

File

gfx_gol_list_box.h

Syntax

```
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;	Pre-computed text height.
GFX_ALIGNMENT alignment;	items alignment

Description

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.

1.6.2.7.13 GFX_GOL_LISTBOX_ITEM_STATUS Type

Defines the types used to indicate the status of an item.

File

gfx_gol_list_box.h

Syntax

```
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

Typedef: GFX_GOL_LISTBOX_ITEM_STATUS

Defines the types used to indicate the status of an item in the List Box.

Remarks

None.

1.6.2.7.14 GFX_GOL_LISTBOX_STATE Type

Specifies the different states of the List Box object.

File

gfx_gol_list_box.h

Syntax

```
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

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.

1.6.2.7.15 GFX_GOL_LISTITEM Type

The structure that defines each item in the list box.

File

gfx_gol_list_box.h

Syntax

```
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

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.

1.6.2.7.16 GFX_GOL_MESSAGE Type

Specifies message structure used in the library.

File

gfx_gol.h

Syntax

```
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

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.

1.6.2.7.17 GFX_GOL_METER Type

Defines the structure used for the Meter object.

File

gfx_gol_meter.h

Syntax

```
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

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.

1.6.2.7.18 GFX_GOL_METER_DRAW_TYPE Type

Specifies the different types of Meter object.

File

gfx_gol_meter.h

Syntax

```
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

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.

1.6.2.7.19 GFX_GOL_METER_STATE Type

Specifies the different states of the Meter object.

File

gfx_gol_meter.h

Syntax

```
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

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.

1.6.2.7.20 GFX_GOL_OBJ_HEADER Type

Specifies Graphics Object Layer structure used in objects.

File

gfx_gol.h

Syntax

```
typedef struct {
  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;
} GFX_GOL_OBJ_HEADER;
```

Members

Members	Description
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.

Description

Typedef: GFX_GOL_OBJ_HEADER

This structure defines the Graphics Object Layer header used in all objects in the Graphics Library.

Remarks

None.

1.6.2.7.21 GFX_GOL_OBJ_TYPE Type

Specifies the different object types used in the library.

File

gfx_gol.h

Syntax

```
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_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

Typedef: GFX_GOL_OBJ_TYPE

This enumeration specifies the different object types used in the library.

Remarks

None.

1.6.2.7.22 GFX_GOL_PICTURECONTROL Type

Defines the structure used for the Picture Control object.

File

gfx_gol_picture.h

Syntax

```
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

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.

1.6.2.7.23 GFX_GOL_PICTURECONTROLCONTROL_STATE Type

Specifies the different states of the Picture Control object.

File

gfx_gol_picture.h

Syntax

```
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

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.

1.6.2.7.24 GFX_GOL_PROGRESSBAR Type

Defines the structure used for the Progress Bar object.

File

gfx_gol_progress_bar.h

Syntax

```
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

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.

1.6.2.7.25 GFX_GOL_PROGRESSBAR_STATE Type

Specifies the different states of the Progress Bar object.

File

gfx_gol_progress_bar.h

Syntax

```
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

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.

1.6.2.7.26 GFX_GOL_RADIOBUTTON Type

Defines the structure used for the Radio Button object.

File

gfx_gol_radio_button.h

Syntax

```
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;	Pre-computed text height
GFX_XCHAR * pText;	Pointer to the text
GFX_ALIGNMENT alignment;	text alignment

Description

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.

1.6.2.7.27 GFX_GOL_RADIOBUTTON_STATE Type

Specifies the different states of the Radio Button object.

File

gfx_gol_radio_button.h

Syntax

```
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

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.

1.6.2.7.28 GFX_GOL_SCROLLBAR Type

Defines the structure used for the Scroll Bar object.

File

gfx_gol_scroll_bar.h

Syntax

```
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

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.

1.6.2.7.29 GFX_GOL_SCROLLBAR_STATE Type

Specifies the different states of the Scroll Bar object.

File

gfx_gol_scroll_bar.h

Syntax

```
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

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.

1.6.2.7.30 GFX_GOL_STATICTEXT Type

Defines the structure used for the Static Text object.

File

gfx_gol_static_text.h

Syntax

```
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

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.

1.6.2.7.31 GFX_GOL_STATICTEXT_STATE Type

Specifies the different states of the Static Text object.

File

gfx_gol_static_text.h

Syntax

```
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

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.

1.6.2.7.32 GFX_GOL_TEXTENTRY Type

Defines the structure used for the Text Entry object.

File

gfx_gol_text_entry.h

Syntax

```
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 editbox.
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

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.

1.6.2.7.33 GFX_GOL_TEXTENTRY_KEY_COMMAND_TYPE Type

Specifies the different commands available for command keys of the Text Entry object.

File

gfx_gol_text_entry.h

Syntax

```
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

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.

1.6.2.7.34 GFX_GOL_TEXTENTRY_KEYMEMBER Type

Defines the structure used to describe a key in the Text Entry object.

File

gfx_gol_text_entry.h

Syntax

```
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

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.

1.6.2.7.35 GFX_GOL_TEXTENTRY_STATE Type

Specifies the different states of the Text Entry object.

File

gfx_gol_text_entry.h

Syntax

```
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

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.

1.6.2.7.36 GFX_GOL_TRANSLATED_ACTION Type

Specifies the different object actions supported in the library.

File

gfx_gol.h

Syntax

```
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 un-check 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

Typedef: GFX_GOL_TRANSLATED_ACTION

This enumeration specifies the different object actions supported in the library.

Remarks

None.

1.6.2.7.37 GFX_GOL_WINDOW Type

Defines the structure used for the Window object.

File

gfx_gol_window.h

Syntax

```
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;	Pre-computed 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

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.

1.6.2.7.38 GFX_GOL_WINDOW_STATE Type

Specifies the different states of the Window object.

File

gfx_gol_window.h

Syntax

```
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

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.

1.6.2.7.39 GOL_PANEL_PARAM Type

Specifies panel parameters.

File

gfx_gol.h

Syntax

```
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;
```

Description

Typedef: GOL_PANEL_PARAM

This structure defines the panel parameters when rendering a panel.

Remarks

None.

1.6.2.7.40 INPUT_DEVICE_EVENT Type

Specifies the different user input device events supported in the library.

File

gfx_gol.h

Syntax

```
typedef enum {
  EVENT_INVALID = 0,
  EVENT_MOVE,
  EVENT_PRESS,
  EVENT_STILLPRESS,
  EVENT_RELEASE,
  EVENT_KEYSCAN,
  EVENT_CHARCODE,
  EVENT_SET,
  EVENT_SET_STATE,
  EVENT_CLR_STATE
} INPUT_DEVICE_EVENT;
```

Members

Members	Description
EVENT_INVALID = 0	Invalid event.
EVENT_MOVE	Move event.
EVENT_PRESS	Press event.
EVENT_STILLPRESS	Continuous press event.
EVENT_RELEASE	Release event.
EVENT_KEYSCAN	Key scan event, parameters for the object ID and keyboard scan code will be sent with this event in the GFX_GOL_MESSAGE as parameter.
EVENT_CHARCODE	Character code event. The actual character code will be sent with this event in the GFX_GOL_MESSAGE as parameter.
EVENT_SET	Generic set event.
EVENT_SET_STATE	Generic set state event.
EVENT_CLR_STATE	Generic clear state event.

Description

Typedef: INPUT_DEVICE_EVENT

This enumeration specifies the different user input device events supported in the graphics library.

Remarks

None.

1.6.2.7.41 INPUT_DEVICE_TYPE Type

Specifies the different user input devices supported in the library.

File

gfx_gol.h

Syntax

```
typedef enum {
  TYPE_UNKNOWN = 0,
  TYPE_KEYBOARD,
  TYPE_TOUCHSCREEN
} INPUT_DEVICE_TYPE;
```

Members

Members	Description
TYPE_UNKNOWN = 0	Unknown device.
TYPE_KEYBOARD	Keyboard.
TYPE_TOUCHSCREEN	Touchscreen.

Description

Typedef: INPUT_DEVICE_TYPE

This enumeration specifies the different user input devices supported in the library.

Remarks

None.

1.6.3 Graphics Driver Layer

Graphics Driver Layer Interface.

1.6.3.1 Graphics Driver Layer API

Graphics Library Driver Layer Interface.

1.6.3.1.1 Driver Layer Initialization Functions

The following API are used to initialize the layer.

Functions

	Name	Description
	DRV_GFX_Initialize	Initialize the graphics display driver.

1.6.3.1.1.1 DRV_GFX_Initialize Function

Initialize the graphics display driver.

File

drv_gfx_display.h

Syntax

```
void DRV_GFX_Initialize();
```

Returns

None.

Description

This function initializes the graphics display driver. This function will be called by the application.

Preconditions

None.

Example

None.

Function

void DRV_GFX_Initialize(void)

1.6.3.1.2 Driver Layer Configuration Functions

The following API are used to set the configuration of the layer.

Macros

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

1.6.3.1.2.1 GFX_MaxXGet Macro

This function returns the maximum horizontal pixel coordinate.

File

drv_gfx_display.h

Syntax

```
#define GFX_MaxXGet (DISP_HOR_RESOLUTION - 1)
```

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.

Function

uint16_t GFX_MaxXGet(void)

1.6.3.1.2.2 GFX_MaxYGet Macro

This function returns the maximum vertical pixel coordinate.

File

drv_gfx_display.h

Syntax

```
#define GFX_MaxYGet (DISP_VER_RESOLUTION - 1)
```

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.

Function

uint16_t GFX_MaxYGet(void)

1.6.3.1.3 Driver Layer Rendering Functions

The following API are the basic driver layer rendering functions.

Functions

	Name	Description
	DRV_GFX_CompleteDrawUpdate	This function completes the partial update for bi-stable displays.
	DRV_GFX_ImageDraw	This function is a driver specific image draw function.
	DRV_GFX_SetupDrawUpdate	This function sets up the partial update for bi-stable displays.
	GFX_PixelArrayGet	Reads an array of pixels from the frame buffer.
	GFX_PixelArrayPut	Renders an array of pixels to the frame buffer.
	GFX_PixelGet	Gets color of the pixel on the given position.
	GFX_PixelPut	Draw the pixel on the given position.
	GFX_RenderStatusGet	This function returns the driver's status on rendering.

1.6.3.1.3.1 DRV_GFX_CompleteDrawUpdate Function

This function completes the partial update for bi-stable displays.

File

drv_gfx_display.h

Syntax

```
void DRV_GFX_CompleteDrawUpdate(uint16_t startx, uint16_t starty, uint16_t endx, uint16_t
endy);
```

Returns

None.

Description

This function completes the partial update for bi-stable displays.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
startx	the horizontal starting position of the partital area that is being updated.
starty	the vertical starting position of the partital area that is being updated.
endx	the horizontal ending position of the partital area that is being updated.
endy	the vertical ending position of the partital area that is being updated.

Function

void DRV_GFX_CompleteDrawUpdate(

uint16_t startx,

uint16_t starty,

uint16_t endx,

uint16_t endy)

1.6.3.1.3.2 DRV_GFX_ImageDraw Function

This function is a driver specific image draw function.

File

drv_gfx_display.h

Syntax

```
uint16_t DRV_GFX_ImageDraw(int16_t left, int16_t top, GFX_RESOURCE_HDR * image,
GFX_PARTIAL_IMAGE_PARAM * pPartialImageData);
```

Returns

The status of the image draw.

- Returns 0 when device is busy and the image is not yet completely drawn. Function must be called again to complete the rendering.
- Returns 1 when the image is completely drawn.

Description

This function is a driver specific image draw function. This function is implemented in drivers that supports accelerated rendering. Execution of image draw functions is faster when implemented in driver layer.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
left	horizontal start position of the image

top	vertical start position of the image
image	pointer to the image resource
pPartialImageData	pointer to the partial image parameters

Function

uint16_t DRV_GFX_ImageDraw(

int16_t left,

int16_t top,

GFX_RESOURCE_HDR *image,

GFX_PARTIAL_IMAGE_PARAM *pPartialImageData)

1.6.3.1.3.3 DRV_GFX_SetupDrawUpdate Function

This function sets up the partial update for bi-stable displays.

File

drv_gfx_display.h

Syntax

```
void DRV_GFX_SetupDrawUpdate(uint16_t startx, uint16_t starty, uint16_t endx, uint16_t endy);
```

Returns

None.

Description

This function sets up the partial update for bi-stable displays.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
startx	the horizontal starting position of the partital area that is being updated.
starty	the vertical starting position of the partital area that is being updated.
endx	the horizontal ending position of the partital area that is being updated.
endy	the vertical ending position of the partital area that is being updated.

Function

void DRV_GFX_SetupDrawUpdate(

uint16_t startx,

uint16_t starty,

uint16_t endx,

uint16_t endy)

1.6.3.1.3.4 GFX_PixelArrayGet Function

Reads an array of pixels from the frame buffer.

File

drv_gfx_display.h

Syntax

```
uint16_t GFX_PixelArrayGet(uint16_t x, uint16_t y, GFX_COLOR * pPixel, uint16_t numPixels);
```

Returns

Returns the actual number of pixels retrieved.

Description

This retrieves an array of pixels from the display buffer starting from the location defined by x and y with the length defined by numPixels. All the retrieved pixels must be inside the display buffer.

The function behavoir will be undefined is one of the following is true:

1. x and y location is outside the frame buffer.
2. x + numPixels exceeds the frame buffer.
3. Depending on how the frame buffer memory is arranged x + numPixels exceeds the width of the frame buffer.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
x	Horizontal starting position of the array of pixels.
y	Vertical starting position of the array of pixels.
pPixel	pointer to the retrieved array of pixel data.
numPixels	length of pixels to be retrieved.

Function

uint16_t GFX_PixelArrayGet(

uint16_t x,

uint16_t y,

GFX_COLOR *pPixel,

uint16_t numPixels)

1.6.3.1.3.5 GFX_PixelArrayPut Function

Renders an array of pixels to the frame buffer.

File

drv_gfx_display.h

Syntax

```
uint16_t GFX_PixelArrayPut(uint16_t x, uint16_t y, GFX_COLOR * pPixel, uint16_t numPixels);
```

Returns

Returns the number of pixels rendered.

Description

This renders an array of pixels starting from the location defined by x and y with the length defined by numPixels. The rendering will be performed in the increasing x direction. If the length is more than 1 line of the screen, the rendering may continue to the next line depending on the way the memory of the display controller is arranged. For example, in some display controller, if the parameters are GFX_PixelArrayPut(0, 0, ptrToArray, (320*240)); The array is rendered on all the pixels of a QVGA screen.

This function also supports transparent color feature. When the feature is enabled the pixel with the transparent color will not be rendered and will be skipped. x and y must define a location within the display buffer.

Preconditions

None.

Example

```
void RedBarDraw(
                int left,
                int top,
                int right,
                int bottom)
{
    int          x, y, length;
    GFX_COLOR    array;

    // assume RED is a red color

    length = (right - left + 1);

    // prepare a line of RED pixels
    for(x = 0; x < length; x++)
    {
        array[x] = RED;
    }

    for(y = top; y < bottom + 1; y++)
    {
        GFX_PixelArrayPut(left, y, array, length);
    }
}
```

Parameters

Parameters	Description
x	Horizontal starting position of the array of pixels.
y	Vertical starting position of the array of pixels.
pPixel	pointer to the array of pixel data.
numPixels	length of pixel array to render.

Function

uint16_t GFX_PixelArrayPut(

uint16_t x,

uint16_t y,

GFX_COLOR *pPixel,

uint16_t numPixels)

1.6.3.1.3.6 GFX_PixelGet Function

Gets color of the pixel on the given position.

File

drv_gfx_display.h

Syntax

```
GFX_COLOR GFX_PixelGet(uint16_t x, uint16_t y);
```

Returns

The color of the specified pixel.

Description

This routine gets the pixel on the given position.

If position is not on the frame buffer, then the behavior is undefined.

Preconditions

None.

Example

```
    void FillAreaWithBlue(
                    int left,
                    int top,
                    int right,
                    int bottom,
                    GFX_COLOR maskColor)
    {
        int   x, y;

        // assume BLUE is color blue
        GFX_ColorSet(BLUE);

        for(y = top; y < bottom + 1; y++)
            for(x = left; x < right + 1; x++)
            {
                // change only the pixel when pixel color is
                // maskColor
                if (GFX_PixelGet(x, y) == maskColor)
                    GFX_PixelPut(x, y);
            }
    }
```

Parameters

Parameters	Description
x	Horizontal position of the pixel.
y	Vertical position of the pixel.

Function

GFX_COLOR GFX_PixelGet(

uint16_t x,

uint16_t y)

1.6.3.1.3.7 GFX_PixelPut Function

Draw the pixel on the given position.

File

drv_gfx_display.h

Syntax

```
GFX_STATUS GFX_PixelPut(uint16_t x, uint16_t y);
```

Returns

Status of the character rendering. (see GFX_STATUS).

Description

This routine draws the pixel on the given position. The color used is the color set by the last call to GFX_ColorSet().

If position 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().

Example

```
void RedBarDraw(
                int left,
                int top,
                int right,
                int bottom)
{
    int   x, y;

    // assume RED is a red color
    GFX_ColorSet(RED);

    for(y = top; y < bottom + 1; y++)
        for(x = left; x < right + 1; x++)
            GFX_PixelPut(x, y);
}
```

Parameters

Parameters	Description
x	Horizontal position of the pixel.
y	Vertical position of the pixel.

Function

GFX_STATUS GFX_PixelPut(

uint16_t x,

uint16_t y)

1.6.3.1.3.8 GFX_RenderStatusGet Function

This function returns the driver's status on rendering.

File

drv_gfx_display.h

Syntax

```
GFX_STATUS_BIT GFX_RenderStatusGet();
```

Returns

Returns GFX_STATUS_BITS.

Description

This function returns the driver's rendering status. Valid values returned are GFX_STATUS_BUSY_BIT or GFX_STATUS_READY_BIT (see GFX_STATUS_BITS).

Preconditions

None.

Example

None.

Function

GFX_STATUS_BITS GFX_RenderStatusGet()

1.6.3.1.4 Driver Layer Hardware Functions

The following API are used for hardware settings of the display module.

Functions

	Name	Description
	DRV_GFX_DisplayBrightness	This function sets the display brightness.

1.6.3.1.4.1 DRV_GFX_DisplayBrightness Function

This function sets the display brightness.

File

drv_gfx_display.h

Syntax

```
void DRV_GFX_DisplayBrightness(uint16_t level);
```

Returns

None.

Description

This function sets the display brightness. The levels are 0-100 where 0 means that the display is turned off and 100 means that the display is at its brightest mode. This usually controls the backlight of the display.

In cases where the backlight is hooked up to a PWM signal the level will determine the frequency and period of the PWM signal. In cases there control is only on or off, 0 means off and level greater than 0 means on.

Preconditions

None.

Example

None.

Parameters

Parameters	Description
level	0-100 where 0 is the darkest mode and 100 is the brightest mode.

Function

void DRV_GFX_DisplayBrightness(uint16_t level)

1.6.3.2 Data Types and Constants

The following are the types and constants used in the Graphics Driver Layer.

Index

S

T

U

W