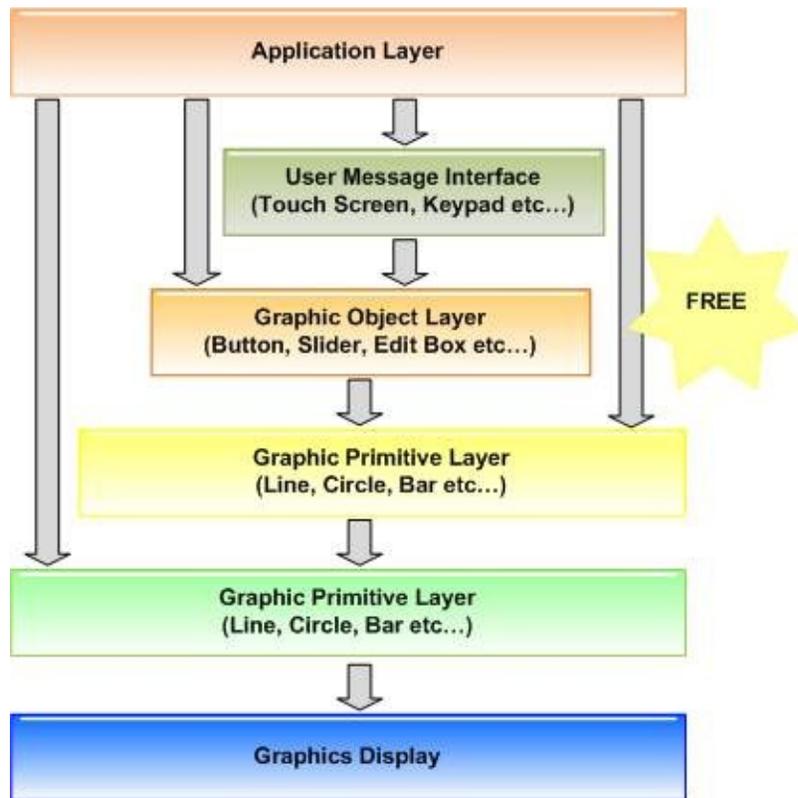


# Introduction

This document explains how to get started with Microchip Graphics Library solution. It presents the available resources and where to obtain these resources. It also includes the Application Programming Interface (API) of the Microchip Graphics Library.



The Microchip Graphics Library is highly modular and is optimized for Microchip's 16-bit microcontrollers. Additionally, the library is free for Microchip customers, easy to use, and has an open documented interface for new driver support, which requires creation of only one C file.

The Microchip Graphics Library Software Version 3.06.02 supports the following features:

- Configurable graphic resolution
- Up to 16-bit or 65K colors
- 2D objects such as line, circle, text, rectangle, polygon, bar
- 3D objects such as buttons, meter, dial, list box, edit box, check box, radio buttons, window, group box, slider.
- Image, animation
- Variety of user input device such as touch screen, keypad etc...
- Multiple fonts
- Unicode fonts
- PIC24 support, PIC32 support

---

## Introduction

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Release Notes

---

## Microchip Graphics Library Release Notes

### Version 3.06.02 (2012-10-15)

This library provides support for the display modules with built in graphics controller and displays connected to external graphics controller. Documentation for the Microchip Graphics Library can be found in this file:

- **Graphics Library Help.chm**

### Version Log

*New:*

None

*Changes:*

None.

*Fixes:*

- Fix [PutImage\(\)](#) function to support RLE compression of images with the following attributes and system level settings:
  - image has 16 bit wide palette entries
  - color depth of image is 8 bpp or 4 bpp
  - [USE\\_PALETTE](#) enabled
  - [COLOR\\_DEPTH](#) set to 8 or 4 bpp
- Fix anti-aliased fonts issue when calculating the 25% blending value of the foreground and background colors.

*Deprecated Items:*

None added on this release.

## *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 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 in V1.00 of XC16.
- When using ListBox Widget, the widget height should be greater than the height of the font used for the widget.
- When using [PutImage\(\)](#) function with RLE compressed images and [USE\\_PALETTE](#) enabled, images with 24 bit wide palette entry is not supported.
- [FillCircle\(\)](#) and [Circle\(\)](#) function output does not match when rendering with small radius.

## **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, dsPIC and PIC32

Family.

## Development Tools

This graphics library release (Version 3.06.02) was tested with IDEs MPLAB v8.85 and MPLAB X 1.20; compilers XC16 v1.00 and XC32 v1.00.

There is a known compatibility issue between the graphics library and C30 v3.25. using C30 3.25 is not recommended for graphics library development.

## Documentation of Resources and Utilities

- The Graphics Library Help File and API document is located in <install\_dir>/Microchip/Help.
  - Graphics Library Help.pdf
- "Graphics Resource Converter" documentation is located in <install\_dir>/Microchip/Graphics/bin/grc.
  - Graphics Resource Converter Help.pdf
- External Memory Programmer documentation is located in <install\_dir>/Microchip/Graphics/bin/memory\_programmer.
  - External Memory Programmer Help.pdf

where: "install\_dir" - is the location of the Microchip Application Library installation.

## Display Modules

The display driver layer of the library is organized to easily switch from one display driver to another. Use of customized display driver is allowed and in the Display Device Driver Layer section. Following graphics controllers are supported in this version:

Display Module	Interface	File Names
Microchip Graphics Display Driver - customizable driver for RGB Glass. Currently used in	RGB	mchpGfxDrv.c, mchpGfxDrv.h

PIC24FJ256DA210 device family.		
Microchip Low-Cost Controllerless (LCC) Graphics Display Driver - customizable driver for RGB Glass. Currently used for selected PIC32MX device families.	RGB	mchpGfxLCC.c, mchpGfxLCC.h
Samsung S6D0129/S6D0139	8/16 bit PMP	drvTFT001.c, drvTFT001.h
LG LGDP4531	8/16 bit PMP	drvTFT001.c, drvTFT001.h
Renesas R61505U/R61580	8/16 bit PMP	drvTFT001.c, drvTFT001.h
Orise Technology SPDF5408	8/16 bit PMP	drvTFT001.c, drvTFT001.h
Epson S1D13517	8/16 bit PMP	S1D13517.c, S1D13517.h
Epson S1D13522	8/16 bit PMP, SPI	S1D13522.c, S1D13522.h
Solomon Systech SSD1926	8/16 bit PMP	SSD1926.c, SSD1926.h
Solomon Systech SSD1289	8/16 bit PMP	drvTFT002.c, drvTFT002.h
Solomon Systech SSD1339 for OLED displays	8 bit PMP	SSD1339.c, SSD1339.h
Solomon Systech SSD1303 for OLED displays	8 bit PMP	SH1101A_SSD1303.c, SH1101A_SSD1303.h
Solomon Systech SSD1305 for	8 bit	SSD1305.c,

OLED displays	PMP	SSD1305.h
Solomon Systech SSD2119	8/16 bit PMP	drvTFT002.c, drvTFT002.h
Sino Wealth SH1101A for OLED displays	8 bit PMP	SH1101A_SSD1303.c, SH1101A_SSD1303.h
Sitronix ST7529	8 bit PMP	ST7529.c, ST7529.h
Hitech Electronics HIT1270	8 bit PMP	HIT1270.c, HIT1270.h
Ilitek ILI9320	8/16 bit PMP	drvTFT001.c, drvTFT001.h
Himax HX8347	8/16 bit PMP	HX8347.c, HX8347.h
UltraChip UC1610	8 bit PMP	UC1610.c, UC1610.h

Please refer to [Adding New Device Driver](#) section to get more information on adding support for another LCD controller.

## Widgets

In this version the following widgets are implemented:

- [Analog Clock](#)
- [Button](#)
- [Chart](#)
- [Checkbox](#)
- [Dial](#)
- [Digital Meter](#)
- [Edit Box](#)
- [Grid](#)
- [Group Box](#)
- [List Box](#)

- [Meter](#)
- [Picture Control](#)
- [Progress Bar](#)
- [Radio Button](#)
- [Slider / Scroll Bar](#)
- [Static Text](#)
- [Text Entry](#)
- [Window](#)

This version of the library supports touch screen, side buttons and a variety of key pad configurations as a user input device.

## Required Resources

The library utilizes the following estimated MCU resources (in # of bytes):

Module	XC16 (Note 2)			XC32 (Note 2)		
	Heap	RAM	Flash	Heap	RAM	Flash
Primitives Layer	0	98	10242	0	103	8266
GOL	26 (per style scheme)	36	2547	26 (per style scheme)	54	2308
<a href="#">Button</a>	36 (per instance)	16	2262	36 (per instance)	24	1916
<a href="#">Chart</a>	56 (per instance)	128	12330	56 (per instance)	148	12132
<a href="#">Check Box</a>	30 (per instance)	4	1119	30 (per instance)	8	916

<u>Dial</u>	48 (per instance)	20	1572	48 (per instance)	24	1460
<u>Digital Meter</u>	36 (per instance)	52	1263	36 (per instance)	56	972
<u>Edit Box</u>	34 (per instance)	10	1248	34 (per instance)	16	1064
<u>Grid</u>	40 (per instance)	0	2238	40 (per instance)	0	2104
<u>Group Box</u>	32 (per instance)	8	1083	32 (per instance)	12	860
<u>List Box</u>	36 (per instance), 12 (per item)	8	2271	36 (per instance), 12 (per item)	16	1848
<u>Meter</u>	60 (per instance)	44	3087	60 (per instance)	48	2704
<u>Picture Control</u>	38 (per instance)	10	834	38 (per instance)	12	724
<u>Progress Bar</u>	34 (per instance)	28	1632	34 (per instance)	32	1264
<u>Radio Button</u>	36 (per instance)	14	1218	36 (per instance)	20	992
<u>Slider / Scroll Bar</u>	40 (per instance)	22	2481	40 (per instance)	28	2152
<u>Static Text</u>	30 (per instance)	8	912	30 (per instance)	12	740
<u>Text Entry</u>	52 (per instance), (24 per	24	2832	52 (per instance), (24 per	28	2372

	key)			key)		
<u>Window</u>	32 (per instance)	2	969	32 (per instance)	4	784

Notes:

1. Data is collected using 16-bit color depth.
2. Data is collected using 's' optimization. XC32 data is using 16-bit build mode.
3. The collected data are based on the most common used configuration of the library and may change depending on the compile time options listed in [Graphics Library Configuration](#) section of this document.
4. Data is based on Graphics Library version 3.06.02.

The heap is the dynamic memory requirement. When using a strategy of dynamically creating objects for the active screen, then the heap memory used will be released when the screen is changed. A new set of objects will be created and the required heap will be allocated. In this scenario, when calculating for the worst case heap requirement, consider the screen with maximum number of objects. If for worse case you have 6 buttons, 2 sliders and 2 edit box on screen that utilizes three style schemes then worse case dynamic memory requirement will be  $6*36 + 2*40 + 2*34 + 3*26$ ; 442 bytes. The RAM requirements (columns 3 & 6), however, is not dependent on the number of instances. If the object is included in the build, then it will use a fixed amount of RAM irrespective of its usage.

Each font, depending on the height, will require program memory. For an English font with 32-127 character IDs and a height of 18-26 pixels, the required memory will be in the range of 7 – 10KB of program memory. This requirement may change for other languages with additional characters.

Images require additional memory. The memory requirement for images depends on color depth and size.

The fonts and images can be stored in internal memory or external memory. The external memory can be anything serial EEPROM, parallel Flash, SD card etc. The application provides physical interface code for these devices.

## 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 (GraphicsConfig.h). 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.
  - $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x20000$  (bytes) - Use PMA[15:0]
  - $0x20000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x40000$ (bytes) - Use PMA[16:0]
  - $0x40000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x80000$ (bytes) - Use PMA[17:0]
  - $0x80000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x100000$ (bytes) - Use PMA[18:0]
  - $0x100000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x200000$ (bytes) - Use PMA[19:0]
  - $0x200000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x400000$ (bytes) - Use PMA[20:0]
  - $0x400000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq 0x800000$ (bytes) - Use PMA[21:0]
  - $0x800000$  (bytes) <  $\text{GFX\_EPMP\_CSx\_MEMORY\_SIZE} \leq$

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 EditText 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":

```


Copy Code

// Valid values of the first field for JPEG_FLASH a  
#define FILE_JPEG_FLASH      2    // the JPEG file is  
#define FILE_JPEG_EXTERNAL  3    // the JPEG file is
```

to

```


Copy Code

// Valid values of the first field for JPEG_FLASH a  
#define FILE_JPEG_FLASH      0    // the JPEG file is  
#define FILE_JPEG_EXTERNAL  1    // the JPEG file is
```

- 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 gfixpmp.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.

Example:

```

Copy Code
// 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 successfully 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:

- 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.
- 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 PICtail™ 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.
- [GetWidth\(\)](#), [GetImageHeight\(\)](#) are added.
- [OutText\(\)](#), [OutTextXY\(\)](#) and [GetTextWidth\(\)](#) functions are terminated by control characters (< 32).
- 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**

None

---

## Release Notes

---

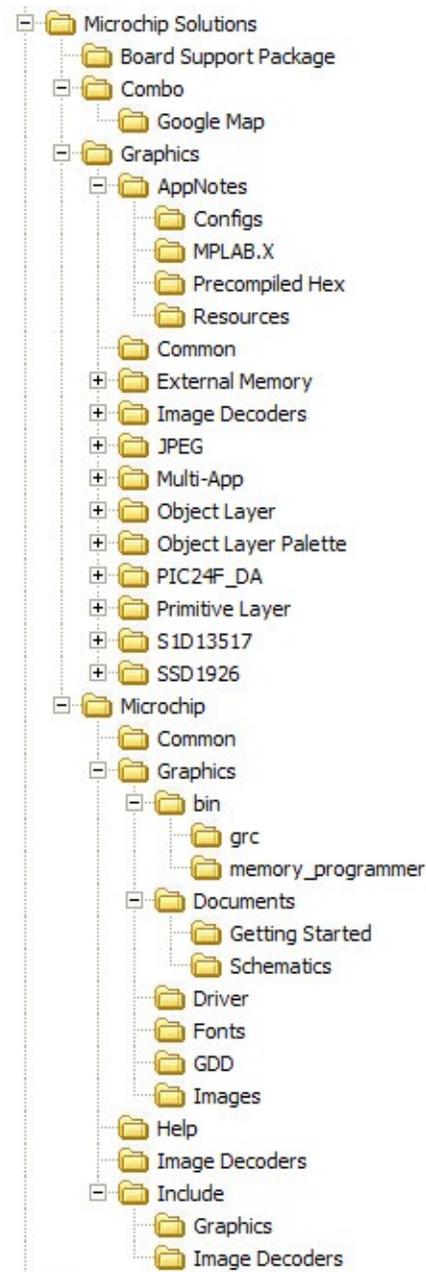
Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Getting Started

## Directory Structure

The Microchip Graphics Library installation follows the standard directory structure for all Microchip library solutions. Installing the library will give the following structure:



One of the demo subdirectories (example: Graphics or Combo) may become "*Your Applications Directory*" that will contain your application source code. You can add code and modules here that will use and interact with the library. The library specific folders are the following:

- The **Microchip** folder will contain the library components.
- The **Help** sub-folder under **Microchip** folder will contain this document (**Graphics Library Help.chm** file).
- The **Graphics** sub-folder under the **Microchip** folder is where the C files, documentation and utilities are located.
  - Inside this **Graphics** sub-folder are the directories for the **Drivers, Documents, GDD, Images** and **bin** directories. It will also contain the directory for the **Image Decoders** source files.
  - The **GDD** (Graphics Display Designer) directory contains the GDD project template. Use this to start projects using the Graphics Display Designer.
  - The **bin** directory contains the Graphics Resource Converter utility and External Memory Programmer both implemented in java.
- The **Include** sub-folder under the **Microchip** folder will contain common header files to all Microchip library solutions.
- Another **Graphics** directory is included in the **Include** sub folder. This will hold the Graphics Library header files as well as the header files for the **Image Decoders**.
- The **Board Support Package** folder will contain hardware specific drivers that are common to the Microchip Demo Boards (such as Explorer 16, display panels or PICtail™ Plus Daughter Boards).

All subdirectories and files under the **Microchip** directory should not be modified. In case your project will use more than one Microchip library solution, this directory will contain all the library files you install. Thus, it is important to maintain the files in this directory. The **Microchip Solutions** directory may become your "*MyProjects*" directory that will contain all your projects using the different Microchip solutions.

## How to Get Started

There are various ways to get started with Microchip Graphics Library:

1. Obtain Development Boards from the "Getting Started" section of the Microchip graphics website ([www.microchip.com/graphics](http://www.microchip.com/graphics)):
  1. Explorer 16 Starter Kit (DV164003) with any of the Graphics PICtail™ Plus Daughter Boards.
  2. PIC24FJ256DA210 Development Board (DV164039) with any of the individual Graphics Display Boards.
  3. A PIC32 Starter Kit and Graphics LCD Controller PICtail™ Plus SSD1926 Board (AC164127-5) with any of the individual Graphics Display Boards.
  4. A PIC32 Starter Kit and Graphics PICtail Plus Epson S1D13517 Board (AC164127-7) with any of the individual Graphics Display Boards.
  5. A PIC32 Starter Kit and Multi-Media Expansion Board (DM320005).
2. Graphics PICtail™ Plus Daughter Board available:
  - AC164127-3 - Graphics PICtail Plus Daughter Board with Truly 3.2" Display Kit



- AC164127-5 - Graphics LCD Controller PICtail Plus SSD1926 Board. This board is the same board used in AC164127-3 PICtail Plus and Display Panel combo shown above.



- AC164127-7 - Graphics PICtail Plus Epson S1D13517 Board.



- AC164144 - Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board.



### 3. Graphics Display Boards available:

- AC164127-4 - Graphics Display Truly 3.2" 240x320 Board



- AC164127-6 - Graphics Display Powertip 4.3" 480x272 Board



- AC164127-8 - Graphics Display Truly 5.7" 640x480 Board



- AC164127-9 - Graphics Display Truly 7" 800x480 Board



- AC164139 - Graphics Display Prototype Board



4. Refer to Web Seminar 4 on "Microchip Graphics Library Architecture" from the "Training and Support" section for an overview of the structure and the different layers of the library. It also gives a brief information on how to use the library.
5. Refer to Microchip's Regional Training Center class on Graphics Library:
  - [HIF 2131](#) – Designing with Microchip Graphics Library

6. For a much detailed look on the usage, you can refer to the following application notes from the "Training and Support" section.
  - [AN1136](#) How to Use Widgets in Microchip Graphics Library. This application note introduces the basic functions needed to create and manage Widgets.
  - [AN1182](#) Fonts in the Microchip Graphics Library. This application note describes the format of the Microchip Graphics Library's font image. It also tells how to reduce the number of characters in a font and automate the creation of the character arrays referring to an application's strings.
  - [AN1227](#) Using a Keyboard with the Microchip Graphics Library. This application note describes how to implement a keyboard-based GUI.
  - [AN1246](#) How to Create Widgets in Microchip Graphics Library. This application note serves as a useful guide in creating customized Widgets. The essential components of a Widget are enumerated and described in this document. This application note also outlines the process of integrating the new Widget into the Graphics Library in order to utilize the already implemented routines for processing messages and rendering Widgets.
  - [AN1368](#) Developing Graphics Applications using PIC MCUs with Integrated Graphics Controller. This application note is intended for engineers who are designing their first graphic application. It describes the basic definitions and jargons of graphics applications and it helps the engineer to understand the theory, necessary decision factors, hardware considerations, available microcontrollers and development tools.
7. Finally, you can obtain the free licensed Microchip Graphics library also from the "Getting Started" section.



## How to Build Projects for the PIC24FJ256DA210 Development Board

1. In the application specific HardwareProfile.h file of your project set the hardware platform to PIC24FJ256DA210 Development Board:

[Copy Code](#)

```
#define PIC24FJ256DA210_DEV_BOARD
```

2. In the the same application specific HardwareProfile.h file of your project, set the correct display controller and the display panel combination. Selecting the correct display panel will choose the correct parameter settings for the display. [Examples](#) of these parameters are horizontal and vertical resolution, display orientation, vertical and horizontal pulse width, and front and back porch settings.
  - When using the Truly 3.2" display on AC164127-4 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_
```

- When using the Powertip 4.3" display on AC164127-6 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210  
// set the display panel
```

```
#define GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q
```

3. In the the same application specific HardwareProfile.h file of your project, set following (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the PMP interface  
#define USE_16BIT_PMP  
// set the Graphics Clock Divider that generates th  
// Refer to display data sheet for pixel clock freq  
// and Family Reference Manual - Oscillator for det  
#define GFX_GCLK_DIVIDER 38  
// set the display buffer start address.  
#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00020000  
// set the EPMP CS1 base address if using external  
#define GFX_EPMP_CS1_BASE_ADDRESS 0x00020000u1  
#define GFX_EPMP_CS1_MEMORY_SIZE 0x40000u1  
// set the EPMP CS2 base address if using external  
#define GFX_EPMP_CS2_BASE_ADDRESS 0x00080000u1  
#define GFX_EPMP_CS2_MEMORY_SIZE 0x80000u1
```

4. In the the project's GraphicsConfig.h set the color depth to the desired bpp value (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the color depth used  
#define COLOR_DEPTH 16
```



## How to Build Projects for Graphics PICtail™ Plus Board

## Version 3:

1. In the application specific HardwareProfile.h file of your project set the hardware platform to Graphics PICtail™ Plus Board Version 3:

[Copy Code](#)

```
#define GFX_PICTAIL_V3
```

2. In the the same application specific HardwareProfile.h file of your project, set the correct display controller and the display panel combination. Selecting the correct display panel will choose the correct parameter settings for the display. [Examples](#) of these parameters are horizontal and vertical resolution, display orientation, vertical and horizontal pulse width, and front and back porch settings.
  - When using the Truly 3.2" display on AC164127-4 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_SSD1926  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_
```

- When using the Powertip 4.3" display on AC164127-6 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_SSD1926  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q
```

3. In the the same application specific HardwareProfile.h file of your project, set following (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the hardware platform  
#define EXPLORER_16
```

```
// set the PMP interface  
#define USE_8BIT_PMP
```

4. In the the project's GraphicsConfig.h set the color depth to the desired bpp value (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the color depth used  
#define COLOR_DEPTH 16
```



## How to Build Projects for Graphics PICtail™ Plus Epson S1D13517 Board

1. In the application specific HardwareProfile.h file of your project set the hardware platform to Graphics PICtail™ Plus Epson S1D13517 Board:

[Copy Code](#)

```
#define GFX_PICTAIL_V3E
```

2. In the the same application specific HardwareProfile.h file of your project, set the correct display controller and the display panel combination. Selecting the correct display panel will choose the correct parameter settings for the display. [Examples](#) of these parameters are horizontal and vertical resolution, display orientation, vertical and horizontal pulse width, and front and back porch settings.
  - When using the Truly 3.2" display on AC164127-4 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_S1D13517  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_
```

- When using the Powertip 4.3" display on AC164127-6 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_S1D13517  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q
```

3. In the the same application specific HardwareProfile.h file of your project, set following (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the hardware platform  
#define EXPLORER_16  
// set the PMP interface  
#define USE_8BIT_PMP  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_
```

4. In the the project's GraphicsConfig.h set the color depth to the desired bpp value (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the color depth used  
#define COLOR_DEPTH 16
```



## Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board:

1. In the application specific HardwareProfile.h file of your project set the hardware platform to Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board:

[Copy Code](#)

```
#define GFX_PICTAIL_LCC
```

2. In the the same application specific HardwareProfile.h file of your project, set the correct display controller and the display panel combination. Selecting the correct display panel will choose the correct parameter settings for the display. [Examples](#) of these parameters are horizontal and vertical resolution, display orientation, vertical and horizontal pulse width, and front and back porch settings.
  - When using the Truly 3.2" display on AC164127-4 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_DMA  
// set the display panel  
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_
```

- When using the Powertip 4.3" display on AC164127-6 board

[Copy Code](#)

```
// set the display controller  
#define GFX_USE_DISPLAY_CONTROLLER_DMA  
// set the display panel
```

```
#define GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q
```

3. In the the same application specific HardwareProfile.h file of your project, set following (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the hardware platform  
#define EXPLORER_16 // or you can use PIC_SK if us  
// set the PMP interface  
#define USE_8BIT_PMP
```

4. In the the project's GraphicsConfig.h set the color depth to the desired bpp value (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the color depth used  
#define COLOR_DEPTH 16
```



## How to Build Projects for the Multimedia Expansion Board

1. In the application specific HardwareProfile.h file of your project set the hardware platform to Graphics PICtail™ Plus Epson S1D13517 Board:

[Copy Code](#)

```
#define MEB_BOARD
```

2. In the application specific HardwareProfile.h file of your project, set the correct display controller.

[Copy Code](#)

```
#define GFX_USE_DISPLAY_CONTROLLER_SSD1926
```

3. In the the same application specific HardwareProfile.h file of your project, set following (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the PMP interface  
#define USE_8BIT_PMP  
// set the Starter Kit used  
#define PIC32_GP_SK // use generic PIC32 Starter Ki  
or  
#define PIC32_USB_SK // use PIC32 USB Starter Kit  
or  
#define PIC32_ETH_SK // use PIC32 Ethernet Starter
```

4. In the the project's GraphicsConfig.h set the color depth to the desired bpp value (Refer to each demo hardware profiles for examples):

[Copy Code](#)

```
// set the color depth used  
#define COLOR_DEPTH 16
```

## Demo Projects

The Microchip Graphics Library documentation has several components that covers installation, customization and usage of the library. Several demo projects are included in the installation to help you get started. Detailed information on each demo project is available from the "Getting Started" help file located in each of the demo folders.

## Schematics

The library installation also includes schematics of currently supported controllers and glass. These can be found in the `../<install directory>/Microchip/Graphics/Documents/Schematics` directory.

- 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 3.2in 240x320 Board Rev 4.pdf
- Schematic for Graphics Display Truly 3.5in 320x240 Board Rev A.pdf
- Schematic for Graphics LCD Controller PICtail SSD1926 Board Rev 2.pdf
- Schematic for Solomon Systech SSD1906.pdf
- Schematic for Truly GG1N1291UTSW-W-TP-E.pdf
- Schematic for Truly TFT-G240320UTSW-92W-TP.pdf
- Schematic for Truly TOD9M0043.pdf
- Schematic for Microtips MTF-T022BHNLP.pdf
- Schematic for Densitron TSR67802.pdf
- 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
- Schematic for Graphics Display Prototype Board Rev 1.pdf
- Schematic for Graphics Display Truly 5.7in 640x480 Board Rev 2.pdf
- Schematic for Graphics Display Truly 7in 800x480 Board Rev 2.pdf

- Schematic for Graphics LCD Controller PICTail Plus S1D13517 Rev 1.1.pdf
- Schematic for Low-Cost Controllerless (LCC) Graphics Board Rev 1.pdf
- Schematic for PIC24FJ256DA210 Development Board Rev 1.1.pdf

## Images

The library allows displaying 1bpp, 4bpp, 8bpp, 16bpp and 24bpp images. They can be located in program flash space or external memory. To convert the bitmap format file (BMP extension) or JPEG format file into source C file containing data array for internal memory or Intel hex file for external memory the “**Graphics Resource Converter**” included in the library installation can be used. Refer to the utility help file for details on usage.

## Fonts

The library operates with 8-bit character encoded strings. It covers languages defined in ISO 8859 standards. East Asian and any other languages support is available for UNICODE encoded fonts. Font can be stored in internal flash as an array in const section (this limits font image size by 32Kbytes) or can be located in external memory. To convert the font file into source C file containing data array for internal memory or Intel hex file for external memory the “**Graphics Resource Converter**” utility can be used. The utility allows importing raster font files (FNT extension) or true font files (TTF extension). Refer to the help built in the utility for details. Raster font files can be extracted from MS Windows bitmap font package file (FNT extension) or converted from true type font file (TTF extension) with a third party font editor. One such freeware editor Fony is available at <http://hukka.furtopia.org/>. Another example is [↗](#)

<http://fontforge.sourceforge.net>.

The utility also allows reducing the generated fonts to include only the characters that the application will use. This can be done by using font filtering. Please refer to application note "AN1182: Fonts in the Microchip Graphics Library" from the "Training and Support" section of the Microchip graphics website for details of implementing reduced fonts.

## How to use the API Documentation

This help file includes the API description of the library. The way the API is structured is similar to the library layers.

1. The Device Driver Layer presents all the API included to initialize and use the display controller and the glass. This section also contains information on how to add new Device Driver.
2. The Graphics Primitive Layer is a hardware independent layer that contains the API for basic rendering functions. Use this section to render basic shapes like lines, rectangles, filled circle etc.
3. The Graphics Object Layer contains the API specific to each Widget type. Use this section to create and manage [Widgets](#) as well as pages or screens of different [Widgets](#). Messaging and rendering of [Widgets](#) are also included in this section.

## Updates and News

Refer to the Microchip graphics website [↗](#)  
[www.microchip.com/graphics](http://www.microchip.com/graphics) for the latest version of the Microchip Graphics Library, webinars, application notes, FAQs and latest news and updates.

---

## [Getting Started](#)

[Contents](#) | [Index](#) | [Home](#)

# Demo Projects

## Topics

Summary of demo projects that comes with the installation of the Microchip Graphics Library.

## Topics

Name	Description
<a href="#">Demo Summary</a>	This is the current list of demo projects released with the Graphics Library.
<a href="#">Microchip Application Library Abbreviations</a>	Summary of Microchip Applications Library Abbreviations used.
<a href="#">Demo Compatibility Matrix</a>	Refer to the Demo Compatibility matrix located in the <install directory>/Microchip/Graphics/Documents/Getting Started/Getting Started - Demo Compatibility Matrix.htm for details.

## Links

### [Topics](#)

### [Demo Projects](#)

## Demo Summary

Demo Name	Description (see Note)
Primitive Layer	Shows how primitive functions are used.
Object Layer	Shows how objects are used with user messages interacting with objects.
Object Layer Palette	The same as the Object Layer demo but uses the Color Look Up Table of the Microchip Graphics Module.
External Memory	A simple demo showing how images and fonts from external memory are used in a graphics application.
Multi-App	A demo showing USB Framework, Memory Disk Drive, Image Decoders and Graphics libraries and stacks are integrated into one application.
PIC32 LCC	A demo showing how the Graphics LCD Controller PICtail™ Plus LCC Board (AC164144) can be connected to either the Explorer 16 board ( with a PIC32 PIM) or connected to a PIC32 Starter Kits.
SSD1926	A demo showing the Solomon Systech Controller JPEG decoder module on the Graphics LCD Controller PICtail™ Plus SSD1926 Board (AC164127-5) and Multi-Media Expansion Board (DM320005) displaying JPEG images from an SD Card.
S1D13517	A demo showing the different features of the Epson S1D13517 Controller in the Graphics PICtail Plus Epson S1D13517 Board (AC164127-7)
E-Paper Epson	A demo showing how the E-paper Display PICtail™ Plus Board with Epson Controller (Epson P/N

	S5U13522C100S00) can be used with the Microchip Graphics Library.
Color Depth	Demo showing how the graphics module in PIC24FJ256DA210 is used in applications using color depths of 1, 4 and 8 BPP. (Location: <install_dir>/Graphics/PIC24F DA/Color Depth)
Elevator	A mock up of Elevator Monitor showing the location of the elevator car and the direction it is moving. (Location: <install_dir>/Graphics/PIC24F DA/Elevator)
RCCGPU-IPU	A demo showing how <a href="#">Rectangle</a> Copy Graphical Processing Unit (RCCGPU) and Inflate Processing Unit (IPU) of the Graphics Module in PIC24FJ256DA10 is used. (Location: <install_dir>/Graphics/PIC24F DA/RCCGPU-IPU)
Remote Control	This is a demo showing how to create a universal remote control device with Graphical Interface using RF4CE protocol. (Location: <install_dir>/Combo/Remote Control)
AppNotes	A collection of application notes demo <ul style="list-style-type: none"> <li>- Demo for Application Note AN1136</li> <li>- Demo for Application Note AN1182</li> <li>- Demo for Application Note AN1227</li> <li>- Demo for Application Note AN1246</li> </ul>
Image Decoders	A demo showing the Image Decoder library rendering bitmaps and jpeg images.
PIC24F Starter Kit	Demo for the PIC24F Starter Kit. (Location: <install_dir>/PIC24F Starter Kit)
PIC24H Starter Kit	Demo for the PIC24H Starter Kit. (Location: <install_dir>/PIC24H Starter Kit)

Google  
Map

A demo showing the TCPIP Stack and Graphics Library combined in an application.  
(Location: <install\_dir>/Combo/Google Map)

## Note:

Unless otherwise specified, the demos are located in <install\_dir>/Graphics.

where: install\_dir - is the directory location of the MLA installation.

## Links

---

[Demo Projects](#)

---

[Demo Projects](#) > [Demo Summary](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Microchip Application Library Abbreviations

---

Microchip Application Library Configuration File and Project Name Abbreviations. A summary of the abbreviations used can be found at <Install Directory>/Microchip/Help/Abbreviations.htm

## Links

---

[Demo Projects](#)

---

[Demo Projects](#) > [Microchip Application Library Abbreviations](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Demo Compatibility Matrix

---

Refer to the Demo Compatibility matrix located in the <install directory>/Microchip/Graphics/Documents/Getting Started/Getting Started - Demo Compatibility Matrix.htm for details.

### Links

---

[Demo Projects](#)

---

[Demo Projects](#) > [Demo Compatibility Matrix](#)

---

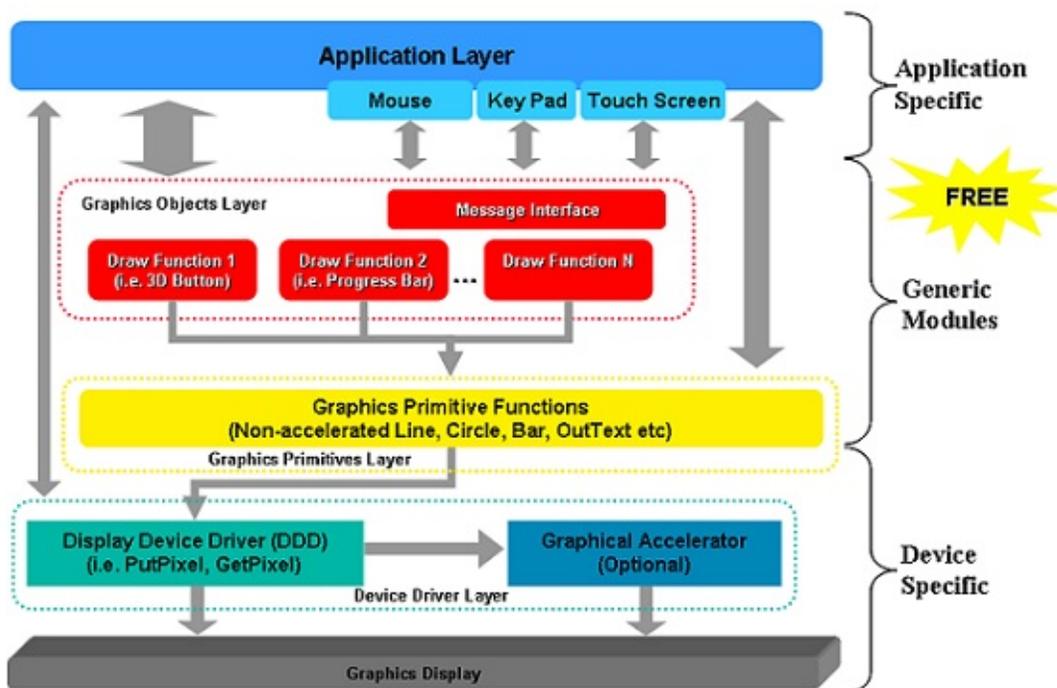
Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Library Architecture

## Topics

The Microchip Graphics 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 layer should be implemented by user to provide messages for the library.
3. Graphics Object Layer – This layer renders the widgets controls such as button, slider, window and so on.
4. Graphics Primitives Layer – This layer implements the primitive drawing functions.
5. Device Display Driver – This layer is dependent on the display device being used.

6. Graphics Display Module – This is the display device being used. The library provides two configurations (Blocking and Non-Blocking).

For Blocking configuration, all draw functions are blocking calls that delay the execution of program until rendering is done. For Non-Blocking configuration, draw functions do not wait for the drawing completion and release control to the program. In this configuration, a draw function should be called repeatedly until the rendering of that particular draw function is complete. This allows efficient use of microcontroller CPU time since it can perform other tasks if the rendering is not yet done.

## Topics

---

Name	Description
<a href="#">Graphics Object Layer</a>	Describes the Graphics Object Layer (GOL) structure and its components.
<a href="#">Graphics Primitive Layer</a>	Describes the Graphics Primitive Layer structure and its components.
<a href="#">Display Device Driver Layer</a>	Describes the Display Driver Layer structure and its components.

## Links

---

[Topics](#)

---

[Library Architecture](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

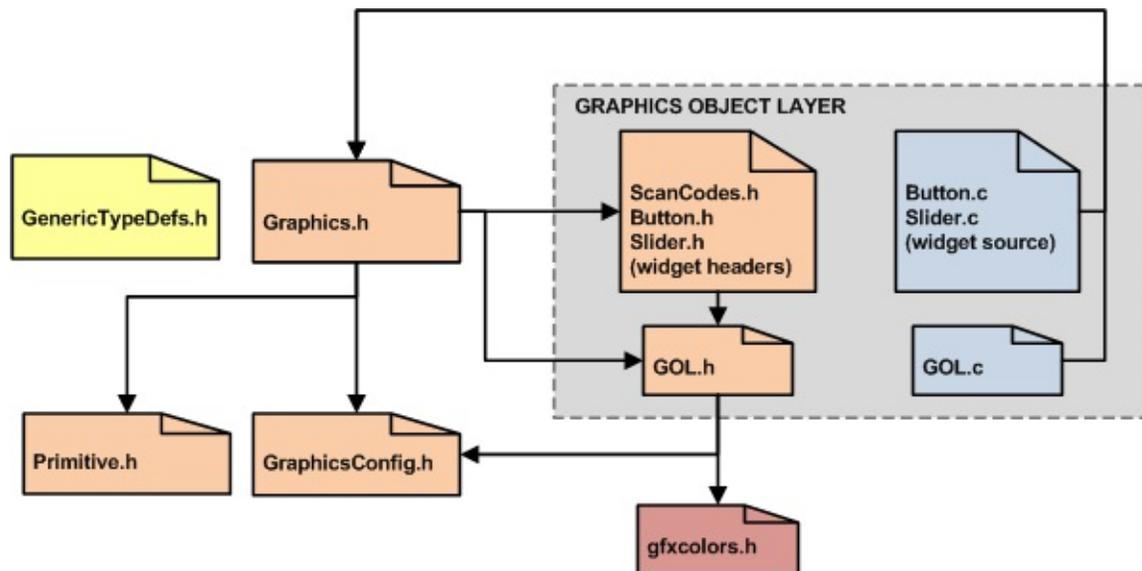
[Contents](#) | [Index](#) | [Home](#)

# Graphics Object Layer

## Topics

The Graphics Object Layer (GOL) implements the Widgets and the Graphics Library managed messaging and rendering. All Widget drawing are based on the Primitive Layer rendering functions.

The Graphics Object Layer organization is shown on the figure below:



## Topics

Name	Description
<a href="#">Object Rendering</a>	Describes the difference between the Blocking or Non-Blocking configuration when rendering Objects.

## Links

## Library Architecture, Topics

---

### Library Architecture > Graphics Object Layer

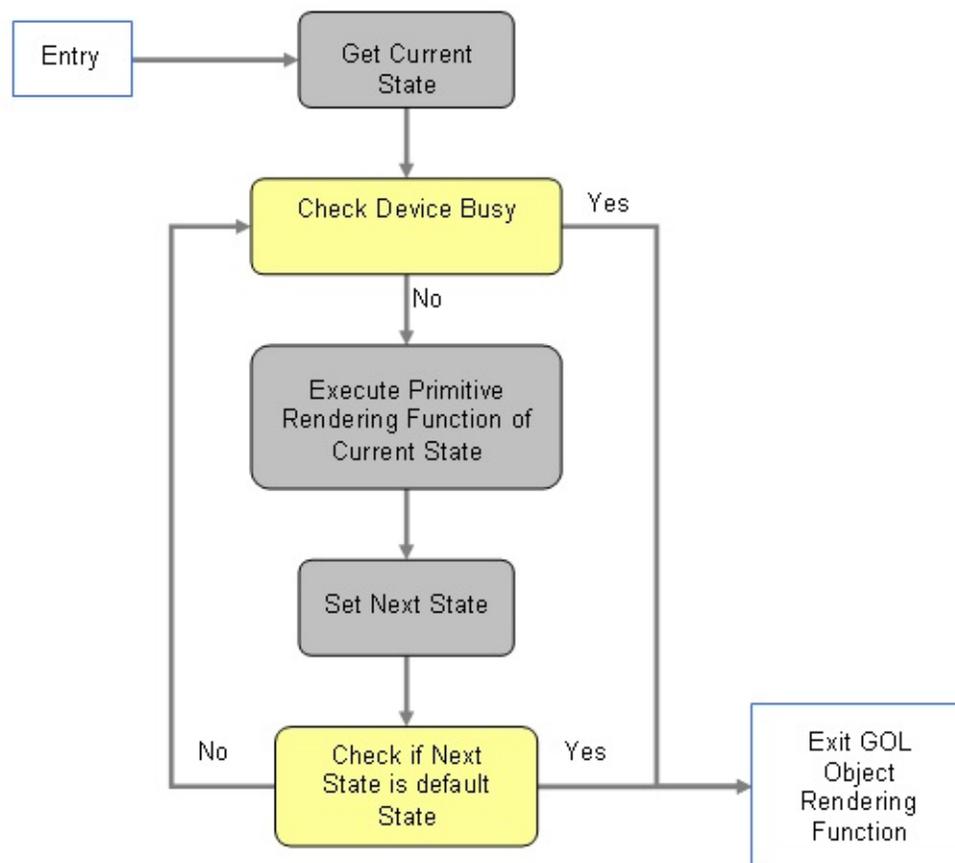
---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Object Rendering

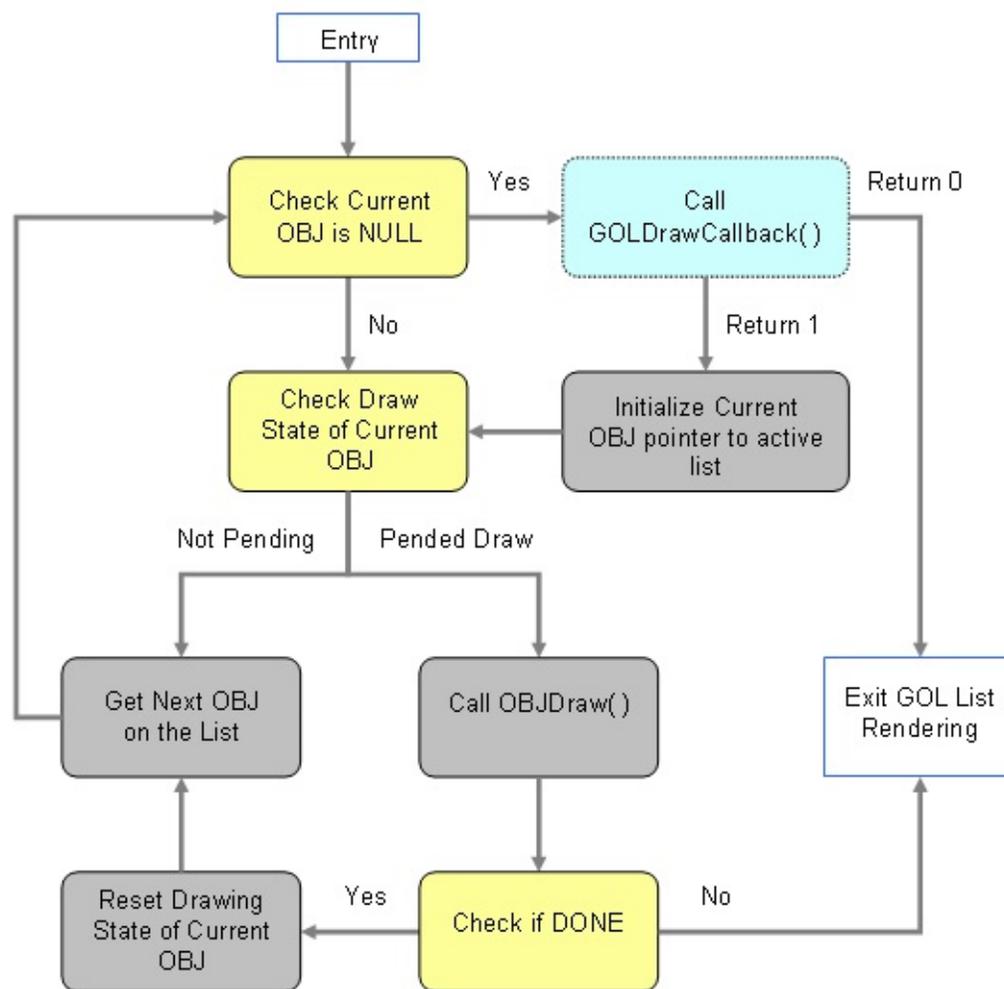
The library can render objects in a Blocking or a Non-Blocking manner. The Non-Blocking configuration is implemented by the use of drawing state machine. Each drawing functions groups the rendering steps into states. Every time a rendering step is executed, the drawing state is updated. Before each step is executed, the display device is checked if it is still busy with the previous rendering operation. If it is busy it returns a non-zero value. This indicates that the draw function must be called again to complete the rendering. The drawing function can be called several times until rendering is completed.



State Machine Controlled Rendering

For Blocking configuration linear flow of rendering is executed. Display device always return a non-busy status.

The GOL level uses the active object linked list for drawing of objects. Each object's state in the list is parsed to determine if the object needs to be redrawn or not. The drawing order is from the head to the tail of the list. This sequence is executed by GOLDraw() function. The figure below explains the rendering loop of the GOLDraw() function.



## GOL Object Rendering Loop

The loop shows two exit points in the sequence. First is when the end of the list is reached and the second is when an OBJDraw() returns a NOT DONE status. Reaching the end of the list is a normal exit. This means that all the state machines of the draw functions of each object have reset to default. Exiting with a NOT DONE status means that the latest executed draw function was pended and the object is not yet fully rendered. To complete the rendering, [GOLDDraw\(\)](#) function should be called again. The next call to [GOLDDraw\(\)](#) will pickup the rendering on the last object that returned a not DONE status. This operation makes the rendering functions non-blocking and gives opportunity to release control to program without waiting for the rendering completion.

When all objects in the active object linked list are drawn [GOLDDraw\(\)](#) calls user defined [GOLDDrawCallback\(\)](#) function. User drawing can be done in this callback function. If the function returns a zero, drawing of GOL objects in the active list is suspended. In this case color, clipping region, line type and graphic cursor will not be modified by GOL. If it returns a 1 drawing control is returned to GOL. [GOLDDraw\(\)](#) resume rendering of objects in the current active list. Inside the [GOLDDrawCallback\(\)](#) function, the active object list is not used by [GOLDDraw\(\)](#). It is safe to perform modification of the list. Please refer to [Configuration Settings](#) to set Blocking or Non-Blocking configuration.

### Links

---

[Graphics Object Layer](#)

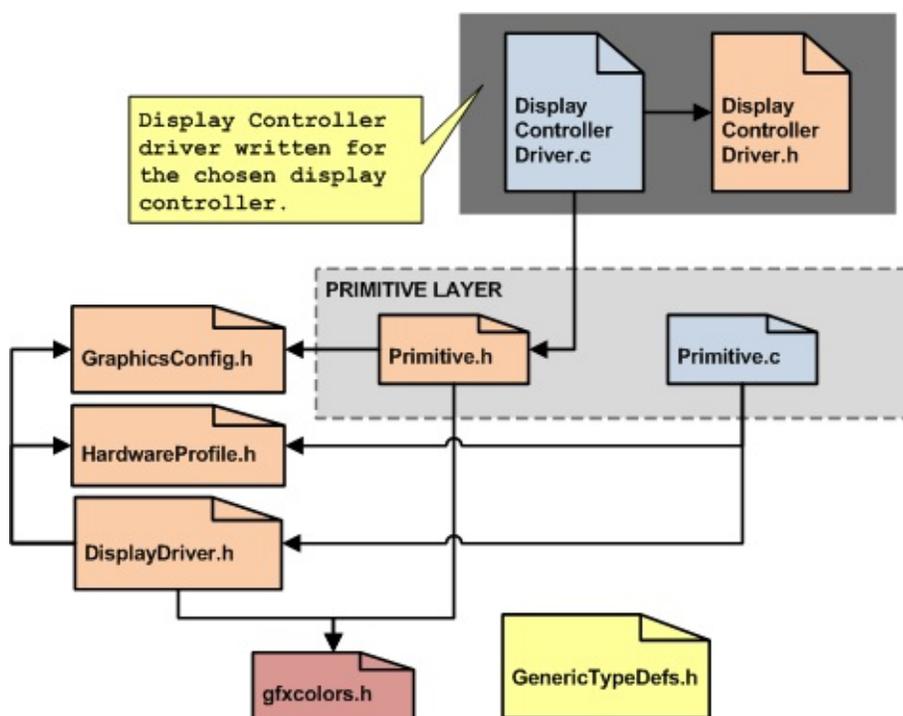
---

[Library Architecture](#) > [Graphics Object Layer](#) > [Object Rendering](#)

## Graphics Primitive Layer

This is a hardware independent layer that contains basic rendering functions. These functions can be implemented in the device driver layer if the display device supports hardware acceleration of the function.

The Primitive Layer organization is shown on the figure below:



## Links

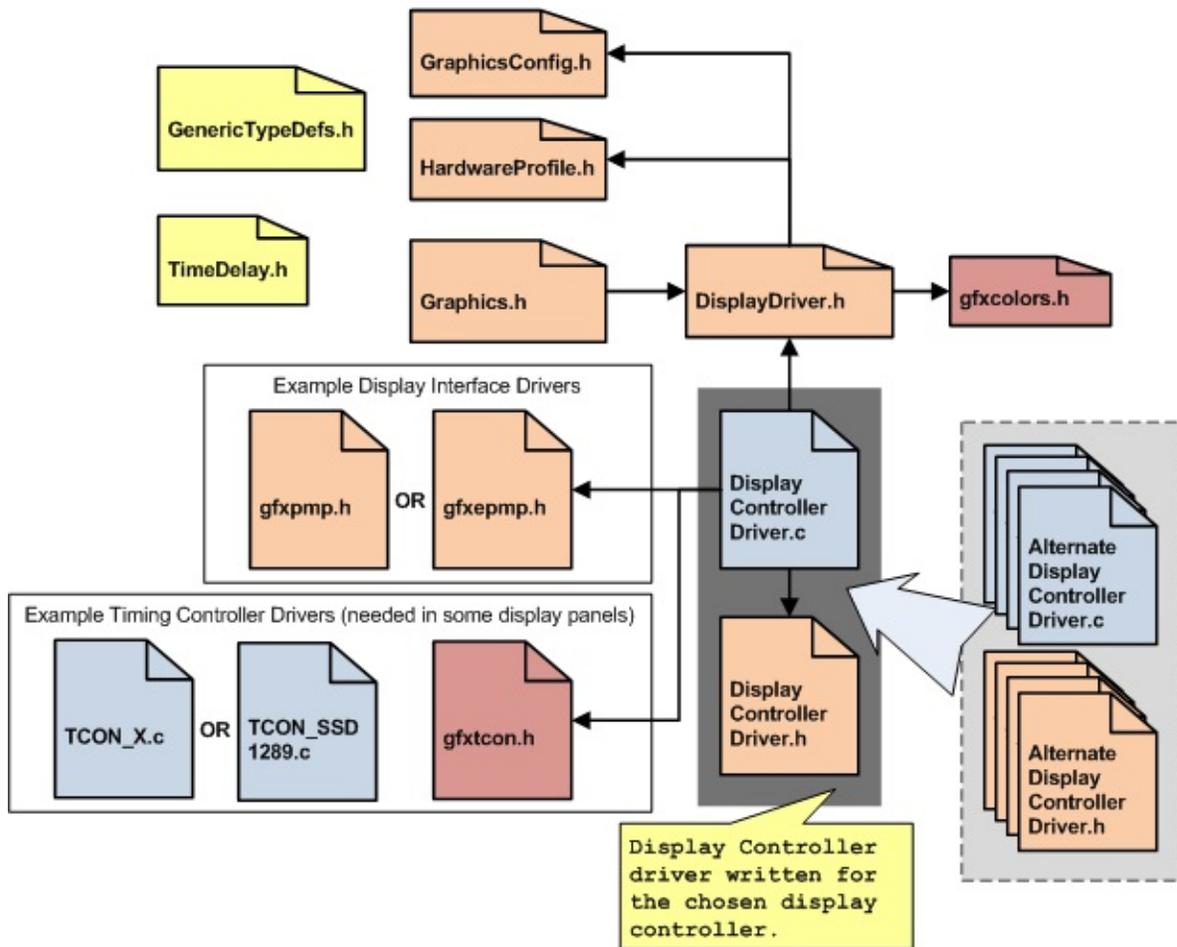
### [Library Architecture](#)

### [Library Architecture](#) > [Graphics Primitive Layer](#)

## Display Device Driver Layer

The Device Driver Layer is the layer that comprises the selection of the display driver file based on the settings specified in the HardwareProfile.h file implemented on the application layer.

The Device Driver Layer organization is shown on the figure below:



An option to use a customized driver is also supported by this scheme. The application need only to define custom display macro in the HardwareProfile header file. This macro must be unique to the display driver. For example the display macro for the SSD1926 driver is

[GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_SSD1926](#). It is recommended that the application keep the same format when naming the display macro,  
GFX\_USE\_DISPLAY\_CONTROLLER\_<DRIVER NAME>.

## Links

---

[Library Architecture](#)

---

[Library Architecture](#) > [Display Device Driver Layer](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Library API

## Topics

The Microchip Graphics Library is implemented in layers. This section describes the APIs for each layer as well as the Graphics Library Configuration. In addition Advanced Display Device Driver Layer APIs that exists in specific drivers are also described.

## Topics

Name	Description
<a href="#">Graphics Library Configuration</a>	The Graphics Library can be customized by adding or specifying the compile time options located in the application code named <a href="#">GraphicsConfig.h</a> or the HardwareProfile.h files. The following compile time options are supported by the library.
<a href="#">Graphics Object Layer API</a>	Description of Graphics Object Layer API.
<a href="#">Graphics Primitive Layer API</a>	Description of Graphics Primitive Layer API.
<a href="#">Display Device Driver Layer API</a>	Description of Display Device Driver Layer API.

## Links

[Topics](#)

[Library API](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Graphics Library Configuration

## Topics

---

### Topics

Name	Description
<a href="#">Graphics Object Layer Configuration</a>	The following compile time options configures the Graphics Library's Object Layer.
<a href="#">Graphics Primitive Layer Configuration</a>	The following compile time options configures the Graphics Library's Primitive Layer.
<a href="#">Display Device Driver Layer Configuration</a>	The following compile time options configures the Graphics Library's Display Device Driver Layer. The choices are based on the specific hardware used. See <a href="#">Hardware Profile</a> for more options.
<a href="#">Application Configuration</a>	
<a href="#">Hardware Profile</a>	These functions and macros are used to determine hardware profile settings on the chosen PIC microcontroller and hardware such as demo boards used.

### Links

[Library API, Topics](#)

[Library API > Graphics Library Configuration](#)

---

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Graphics Object Layer Configuration

## Topics

The following compile time options configures the Graphics Library's Object Layer.

## Topics

Name	Description
<a href="#">Input Device Selection</a>	<p>The Graphics Library comes with two pre-defined user interface. These are the:</p> <ul style="list-style-type: none"><li>• Keyboard interface</li><li>• Touchscreen interface</li></ul> <p>Enabling one or both requires the declaration of the compile switch macros in the GraphicsConfig.h file.</p> <p>GOL widgets which supports the enabled input device's messages will respond to the user inputs.</p> <p>For Example:</p> <p>When using <a href="#">Button</a> Widget.</p> <p><code>#define <a href="#">USE_TOUCHSCREEN</a></code> - will enable the buttons response to user touch on the button widget. The button will automatically be drawn with a pressed state when pressed and release state when user removes the touch on the screen.</p> <p>The compile option selects the input devices used by... <a href="#">more</a></p>
<a href="#">Focus Support Selection</a>	<p>This compile option allows keyboard input focus. <a href="#">GOLSetFocus()</a>, <a href="#">GOLGetFocus()</a>, <a href="#">GOLCanBeFocused()</a>, <a href="#">GOLGetFocusNext()</a></p>

	<p>functions will be available. Focus is also changed by touch screen.</p> <p>The <a href="#">USE_FOCUS</a> option is located in the GraphicsConfig.h header file.</p>
<a href="#">Graphics Object Selection</a>	<p>These compile options selects objects used. Remove definitions for unused objects to reduce code size.</p> <p>The <a href="#">USE_GOL</a> and USE_OBJECT options are located in the GraphicsConfig.h header file. <a href="#">USE_GOL</a> should be included if any of the objects are to be used.</p>

## Links

---

[Graphics Library Configuration, Topics](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Input Device Selection

## Macros

The Graphics Library comes with two pre-defined user interface. These are the:

- Keyboard interface
- Touchscreen interface

Enabling one or both requires the declaration of the compile switch macros in the GraphicsConfig.h file.

GOL widgets which supports the enabled input device's messages will respond to the user inputs.

For Example:

When using [Button](#) Widget.

`#define USE\_TOUCHSCREEN` - will enable the buttons response to user touch on the button widget. The button will automatically be drawn with a pressed state when pressed and release state when user removes the touch on the screen.

The compile option selects the input devices used by GOL widgets. Remove or comment out the macro declarations for unused input devices to reduce code size.

## Macros

Name	Description
<a href="#">USE_KEYBOARD</a>	Input devices macros that defines the messages that Objects will process. The following definitions indicate the usage of the different input devices:

	<ul style="list-style-type: none"> <li>• <code>USE_TOUCHSCREEN</code> - enables the touch screen support.</li> <li>• <code>USE_KEYBOARD</code> - enables the key board support.</li> </ul> <p>Define in GraphicsConfig.h</p>
<p><u><code>USE_TOUCHSCREEN</code></u></p>	<p>Input devices macros that defines the messages that Objects will process. The following definitions indicate the usage of the different input devices:</p> <ul style="list-style-type: none"> <li>• <code>USE_TOUCHSCREEN</code> - enables the touch screen support.</li> <li>• <code>USE_KEYBOARD</code> - enables the key board support.</li> </ul> <p>Define in GraphicsConfig.h</p>

## Links

### [Graphics Object Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Input Device Selection](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_KEYBOARD Macro

```
C
```

```
#define USE_KEYBOARD
```

## Overview

Input devices macros that defines the messages that Objects will process. The following definitions indicate the usage of the different input devices:

- USE\_TOUCHSCREEN - enables the touch screen support.
- USE\_KEYBOARD - enables the key board support.

Define in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Input Device Selection](#) > [USE\\_KEYBOARD Macro](#)

# USE\_TOUCHSCREEN Macro

```
C
```

```
#define USE_TOUCHSCREEN
```

## Overview

Input devices macros that defines the messages that Objects will process. The following definitions indicate the usage of the different input devices:

- `USE_TOUCHSCREEN` - enables the touch screen support.
- `USE_KEYBOARD` - enables the key board support.

Define in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Input Device Selection](#) > [USE\\_TOUCHSCREEN Macro](#)

## Focus Support Selection

### Macros

This compile option allows keyboard input focus.

[GOLSetFocus\(\)](#), [GOLGetFocus\(\)](#), [GOLCanBeFocused\(\)](#), [GOLGetFocusNext\(\)](#) functions will be available. Focus is also changed by touch screen.

The [USE\\_FOCUS](#) option is located in the GraphicsConfig.h header file.

### Macros

Name	Description
<a href="#">USE_FOCUS</a>	Keyboard control on some objects can be used by enabling the GOL Focus (USE_FOCUS)support. Define this in GraphicsConfig.h

### Links

[Graphics Object Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Focus Support Selection](#)

# USE\_FOCUS Macro

```
C
```

```
#define USE_FOCUS
```

## Overview

Keyboard control on some objects can be used by enabling the GOL Focus (USE\_FOCUS) support. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Focus Support Selection](#) > [USE\\_FOCUS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Graphics Object Selection

## Macros

These compile options selects objects used. Remove definitions for unused objects to reduce code size.

The [USE\\_GOL](#) and USE\_OBJECT options are located in the GraphicsConfig.h header file. [USE\\_GOL](#) should be included if any of the objects are to be used.

## Macros

Name	Description
<a href="#">USE_ANALOGCLOCK</a>	Enable Analog Clock Object.
<a href="#">USE_BUTTON</a>	Enable <a href="#">Button</a> Object.
<a href="#">USE_BUTTON_MULTI_LINE</a>	Enable Multi-Line <a href="#">Button</a> Object
<a href="#">USE_CHECKBOX</a>	Enable <a href="#">Checkbox</a> Object.
<a href="#">USE_DIGITALMETER</a>	Enable DigitalMeter Object.
<a href="#">USE_EDITBOX</a>	Enable Edit Box Object.
<a href="#">USE_GROUPBOX</a>	Enable Group Box Object.
<a href="#">USE_LISTBOX</a>	Enable List Box Object.
<a href="#">USE_METER</a>	Enable <a href="#">Meter</a> Object.
<a href="#">USE_PICTURE</a>	Enable <a href="#">Picture</a> Object.
<a href="#">USE_PROGRESSBAR</a>	Enable Progress <a href="#">Bar</a> Object.
<a href="#">USE_RADIOBUTTON</a>	Enable Radio <a href="#">Button</a> Object.

<a href="#"><u>USE_ROUNDIAL</u></a>	Enable <a href="#"><u>Dial</u></a> Object.
<a href="#"><u>USE_SLIDER</u></a>	Enable <a href="#"><u>Slider</u></a> or <a href="#"><u>Scroll Bar</u></a> Object.
<a href="#"><u>USE_STATICTEXT</u></a>	Enable Static Text Object.
<a href="#"><u>USE_WINDOW</u></a>	Enable <a href="#"><u>Window</u></a> Object.
<a href="#"><u>USE_CUSTOM</u></a>	Enable Custom Control Object (an example to create customized Object).
<a href="#"><u>USE_GOL</u></a>	Enable Graphics Object Layer.
<a href="#"><u>USE_TEXTENTRY</u></a>	Enable TextEntry Object.

## Links

---

### [Graphics Object Layer Configuration, Macros](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_ANALOGCLOCK Macro

---

```
C
```

```
#define USE_ANALOGCLOCK
```

## Description

---

Enable Analog Clock Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_ANALOGCLOCK Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_BUTTON Macro

---

C

```
#define USE_BUTTON
```

## Description

---

Enable [Button](#) Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_BUTTON Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_BUTTON\_MULTI\_LINE Macro

```
C
```

```
#define USE_BUTTON_MULTI_LINE
```

## Description

Enable Multi-[Line Button](#) Object

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_BUTTON\\_MULTI\\_LINE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_CHECKBOX Macro

---

```
C
```

```
#define USE_CHECKBOX
```

## Description

---

Enable [Checkbox](#) Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_CHECKBOX Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_DIGITALMETER Macro

```
C
```

```
#define USE_DIGITALMETER
```

## Description

Enable DigitalMeter Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_DIGITALMETER Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_EDITBOX Macro

---

C

```
#define USE_EDITBOX
```

## Description

---

Enable Edit Box Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_EDITBOX Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_GROUPBOX Macro

```
C
```

```
#define USE_GROUPBOX
```

## Description

Enable Group Box Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_GROUPBOX Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_LISTBOX Macro

---

```
C
```

```
#define USE_LISTBOX
```

## Description

---

Enable List Box Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_LISTBOX Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_METER Macro

---

```
C
```

```
#define USE_METER
```

## Description

---

Enable [Meter](#) Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_METER Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_PICTURE Macro

---

**C**

```
#define USE_PICTURE
```

## Description

---

Enable [Picture](#) Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_PICTURE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_PROGRESSBAR Macro

---

```
C
```

```
#define USE_PROGRESSBAR
```

## Description

---

Enable Progress [Bar](#) Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_PROGRESSBAR Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_RADIOBUTTON Macro

```
C
```

```
#define USE_RADIOBUTTON
```

## Description

Enable Radio [Button](#) Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_RADIOBUTTON Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_ROUNDIAL Macro

```
C
```

```
#define USE_ROUNDIAL
```

## Description

Enable [Dial](#) Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_ROUNDIAL Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_SLIDER Macro

---

```
C
```

```
#define USE_SLIDER
```

## Description

---

Enable [Slider](#) or [Scroll Bar](#) Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_SLIDER Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_STATICTEXT Macro

```
C
```

```
#define USE_STATICTEXT
```

## Description

Enable Static Text Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_STATICTEXT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_WINDOW Macro

---

```
C
```

```
#define USE_WINDOW
```

## Description

---

Enable [Window](#) Object.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_WINDOW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_CUSTOM Macro

---

C

```
#define USE_CUSTOM
```

## Description

---

Enable Custom Control Object (an example to create customized Object).

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_CUSTOM Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_GOL Macro

---

```
C
```

```
#define USE_GOL
```

## Description

---

Enable Graphics Object Layer.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_GOL Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_TEXTENTRY Macro

---

C

```
#define USE_TEXTENTRY
```

## Description

---

Enable TextEntry Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#) > [USE\\_TEXTENTRY Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Graphics Primitive Layer Configuration

## Topics

The following compile time options configures the Graphics Library's Primitive Layer.

## Topics

Name	Description
<a href="#">Image Compression Option</a>	These compile options to set if the images used for OutImage() are RLE compressed or IPU compressed.
<a href="#">Font Type Selection</a>	<p>This compile option selects if the support for unicode fonts, unsigned char or the default signed char type fonts.</p> <p>There are three types of font (characters) that can be used in the Graphics Library. This gives the user the option to implement multi-language application or use the default signed char type.</p>
<a href="#">Advanced Font Features Selection</a>	This compile option enables the advanced font features.
<a href="#">Gradient Bar Rendering</a>	This compile option enables the usage of the Gradient <a href="#">Bar</a> and <a href="#">Bevel</a> function in the Primitive Layer.
<a href="#">Transparent Color Feature in PutImage()</a>	This compile option enables the transparent color feature in <a href="#">PutImage()</a> .
<a href="#">Alpha Blend Option</a>	This compile option enables the Alpha-

[External Memory Buffer](#)

Blend feature in Primitive Layer.

see [EXTERNAL\\_FONT\\_BUFFER\\_SIZE](#)

## Links

---

[Graphics Library Configuration, Topics](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Image Compression Option

## Macros

These compile options to set if the images used for `OutImage()` are RLE compressed or IPU compressed.

## Macros

Name	Description
<a href="#"><u>USE_COMP_IPU</u></a>	To enable support for DEFLATE compressed images for <code>PutImage()</code> . When this macro is enabled, the <code>PutImage()</code> function will be able to process images generated by the Graphics Resource Converter (GRC) that are compressed using the DEFLATE algorithm. <code>PutImage()</code> will need the IPU module of the Microchip Graphics Module to decompress. Enable this feature only when the driver features the IPU module (example: PIC24FJ2456DA210). Define this in <code>GraphicsConfig.h</code>
<a href="#"><u>USE_COMP_RLE</u></a>	To enable support for RLE compressed images for <code>PutImage()</code> . When this macro is enabled, the <code>PutImage()</code> function will be able to process images generated by the Graphics Resource Converter (GRC) that are RLE compressed. Define this in <code>GraphicsConfig.h</code>

## Links

[Graphics Primitive Layer Configuration, Macros](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Image Compression Option](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## USE\_COMP\_IPU Macro

```
C
```

```
#define USE_COMP_IPU
```

### Overview

To enable support for DEFLATE compressed images for [PutImage\(\)](#). When this macro is enabled, the [PutImage\(\)](#) function will be able to process images generated by the Graphics Resource Converter (GRC) that are compressed using the DEFLATE algorithm. [PutImage\(\)](#) will need the IPU module of the Microchip Graphics Module to decompress. Enable this feature only when the driver features the IPU module (example: PIC24FJ2456DA210). Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Image Compression Option](#) > [USE\\_COMP\\_IPU Macro](#)

# USE\_COMP\_RLE Macro

```
C
```

```
#define USE_COMP_RLE
```

## Overview

To enable support for RLE compressed images for [PutImage\(\)](#). When this macro is enabled, the [PutImage\(\)](#) function will be able to process images generated by the Graphics Resource Converter (GRC) that are RLE compressed. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Image Compression Option](#) > [USE\\_COMP\\_RLE Macro](#)

# Font Type Selection

## Macros

This compile option selects if the support for unicode fonts, unsigned char or the default signed char type fonts.

There are three types of font (characters) that can be used in the Graphics Library. This gives the user the option to implement multi-language application or use the default signed char type.

Define in GraphicsConfig.h	<u>XCHAR</u> type	Description
<code>#define <u>USE_MULTIBYTECHAR</u></code>	<code>#define <u>XCHAR</u> unsigned short</code>	Enable support for multi-byte fonts such as Unicode fonts.
<code>#define <u>USE_UNSIGNED_XCHAR</u></code>	<code>#define <u>XCHAR</u> unsigned char</code>	Enable support for character range of 0-255.
none of the two are defined	<code>#define <u>XCHAR</u> char</code>	Character range is set to 0-127.

Note: Only one of the two or none at all are defined in GraphicsConfig.h.

- #define USE\_MULTIBYTECHAR
- #define USE\_UNSIGNED\_XCHAR

- when none are defined, [XCHAR](#) defaults to type char.

See [XCHAR](#) for details.

## Macros

---

Name	Description
<a href="#">USE_MULTIBYTECHAR</a>	To enable support for unicode fonts, <a href="#">USE_MULTIBYTECHAR</a> must be defined. This sets the <a href="#">XCHAR</a> definition (0-2 <sup>16</sup> range). See <a href="#">XCHAR</a> for details. Define this in GraphicsConfig.h
<a href="#">USE_UNSIGNED_XCHAR</a>	To enable support for unsigned characters data type for fonts, <a href="#">USE_UNSIGNED_XCHAR</a> must be defined. This sets the <a href="#">XCHAR</a> definition (0-255 range). See <a href="#">XCHAR</a> for details. Define this in GraphicsConfig.h

## Links

---

[Graphics Primitive Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Font Type Selection](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_MULTIBYTECHAR Macro

```
C
```

```
#define USE_MULTIBYTECHAR
```

## Overview

To enable support for unicode fonts, USE\_MULTIBYTECHAR must be defined. This sets the [XCHAR](#) definition (0-2<sup>16</sup> range). See [XCHAR](#) for details. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Font Type Selection](#) > [USE\\_MULTIBYTECHAR Macro](#)

# USE\_UNSIGNED\_XCHAR Macro

```
C
```

```
#define USE_UNSIGNED_XCHAR
```

## Overview

To enable support for unsigned characters data type for fonts, `USE_UNSIGNED_XCHAR` must be defined. This sets the [XCHAR](#) definition (0-255 range). See [XCHAR](#) for details. Define this in `GraphicsConfig.h`

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Font Type Selection](#) > [USE\\_UNSIGNED\\_XCHAR Macro](#)

# Advanced Font Features Selection

## Macros

This compile option enables the advanced font features.

## Macros

Name	Description
<a href="#"><u>USE_ANTIALIASED_FONTS</u></a>	To enable support for Anti-aliased fonts. Use this feature if the font table generated through the "Graphics Resource Converter" tool has the anti-aliasing enabled. Define this in GraphicsConfig.h

## Links

[Graphics Primitive Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Advanced Font Features Selection](#)

# USE\_ANTIALIASED\_FONTS Macro

```
C
```

```
#define USE_ANTIALIASED_FONTS
```

## Overview

To enable support for Anti-aliased fonts. Use this feature if the font table generated through the "Graphics Resource Converter" tool has the anti-aliasing enabled. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Advanced Font Features Selection](#) > [USE\\_ANTIALIASED\\_FONTS Macro](#)

# Gradient Bar Rendering

## Macros

This compile option enables the usage of the Gradient [Bar](#) and [Bevel](#) function in the Primitive Layer.

## Macros

Name	Description
<a href="#">USE_GRADIENT</a>	To enable support for Gradient bars and bevel primitives. Define this in GraphicsConfig.h.

## Links

[Graphics Primitive Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Gradient Bar Rendering](#)

# USE\_GRADIENT Macro

```
C
```

```
#define USE_GRADIENT
```

## Overview

To enable support for Gradient bars and bevel primitives. Define this in GraphicsConfig.h.

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Gradient Bar Rendering](#) > [USE\\_GRADIENT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Transparent Color Feature in PutImage()

## Macros

This compile option enables the transparent color feature in [PutImage\(\)](#).

## Macros

Name	Description
<a href="#">USE_TRANSPARENT_COLOR</a>	To enable support for transparent color in <a href="#">PutImage()</a> . Enabling this macro enables the use of a transparent color (set by <a href="#">TransparentColorEnable()</a> ) in rendering images by <a href="#">PutImage()</a> . When a pixel in the image matches the transparent color set, the pixel is not rendered to the screen. This is useful in rendering rounded icons or images to the screen with a complex background. Define this in GraphicsConfig.h

## Links

[Graphics Primitive Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Transparent Color Feature in PutImage\(\)](#)

# USE\_TRANSPARENT\_COLOR Macro

```
C
```

```
#define USE_TRANSPARENT_COLOR
```

## Overview

To enable support for transparent color in [PutImage\(\)](#). Enabling this macro enables the use of a transparent color (set by [TransparentColorEnable\(\)](#)) in rendering images by [PutImage\(\)](#). When a pixel in the image matches the transparent color set, the pixel is not rendered to the screen. This is useful in rendering rounded icons or images to the screen with a complex background. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Transparent Color Feature in PutImage\(\)](#) > [USE\\_TRANSPARENT\\_COLOR Macro](#)

# Alpha Blend Option

## Macros

This compile option enables the Alpha-Blend feature in Primitive Layer.

## Macros

Name	Description
<a href="#"><u>USE_ALPHABLEND_LITE</u></a>	To enable support for Alpha Blending on the Primitive Layer. This feature is only limited on Alpha-Blending a <a href="#"><u>Bar()</u></a> of a specified color (set by <a href="#"><u>SetColor()</u></a> ) to the destination defined by the parameters of the <a href="#"><u>Bar()</u></a> function call. The Alpha level used is set by <a href="#"><u>SetAlpha()</u></a> . Define this in GraphicsConfig.h

## Links

[Graphics Primitive Layer Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Alpha Blend Option](#)

# USE\_ALPHABLEND\_LITE Macro

```
C
```

```
#define USE_ALPHABLEND_LITE
```

## Overview

To enable support for Alpha Blending on the Primitive Layer. This feature is only limited on Alpha-Blending a [Bar\(\)](#) of a specified color (set by [SetColor\(\)](#)) to the destination defined by the parameters of the [Bar\(\)](#) function call. The Alpha level used is set by [SetAlpha\(\)](#). Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Alpha Blend Option](#) > [USE\\_ALPHABLEND\\_LITE Macro](#)

# External Memory Buffer

---

see [EXTERNAL\\_FONT\\_BUFFER\\_SIZE](#)

## Links

---

[Graphics Primitive Layer Configuration](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [External Memory Buffer](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Display Device Driver Layer Configuration

## Macros

The following compile time options configures the Graphics Library's Display Device Driver Layer. The choices are based on the specific hardware used.

See [Hardware Profile](#) for more options.

## Macros

Name	Description
<a href="#">USE_ALPHABLEND</a>	To enable support for Alpha Blending. Use this feature only if the display driver used can support alpha blending. Define this in GraphicsConfig.h
<a href="#">USE_DOUBLE_BUFFERING</a>	To enable support for double buffering. Use this feature only if the display driver used can support double buffering. Define this in GraphicsConfig.h
<a href="#">GFX_LCD_TYPE</a>	Sets the type of display glass used. Define this in the Hardware Profile. <ul style="list-style-type: none"><li>• #define GFX_LCD_TYPE <a href="#">GFX_LCD_TFT</a> - sets type TFT display</li><li>• #define GFX_LCD_TYPE <a href="#">GFX_LCD_CSTN</a> - sets type color STN display</li><li>• #define GFX_LCD_TYPE <a href="#">GFX_LCD_MSTN</a> - sets type mon STN display</li></ul>

	<ul style="list-style-type: none"> <li>• #define GFX_LCD_TYPE <a href="#">GFX_LCD_OFF</a> - display is turned off</li> </ul>
<a href="#">STN_DISPLAY_WIDTH</a>	<p>Sets the STN glass data width. Define this in the Hardware Profile.</p> <ul style="list-style-type: none"> <li>• #define STN_DISPLAY_WIDTH <a href="#">STN_DISPLAY_WIDTH_4</a> - use 4-bit wide interface</li> <li>• #define STN_DISPLAY_WIDTH <a href="#">STN_DISPLAY_WIDTH_8</a> - Use 8-bit wide interface</li> <li>• #define STN_DISPLAY_WIDTH <a href="#">STN_DISPLAY_WIDTH_16</a> - Use 16-bit wide interface</li> </ul>

## Links

[Graphics Library Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_ALPHABLEND Macro

```
C
```

```
#define USE_ALPHABLEND
```

## Overview

To enable support for Alpha Blending. Use this feature only if the display driver used can support alpha blending. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [USE\\_ALPHABLEND Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_DOUBLE\_BUFFERING Macro

```
C
```

```
#define USE_DOUBLE_BUFFERING
```

## Overview

To enable support for double buffering. Use this feature only if the display driver used can support double buffering. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [USE\\_DOUBLE\\_BUFFERING Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_LCD\_TYPE Macro

## Macros

C

```
#define GFX_LCD_TYPE
```

## Overview

Sets the type of display glass used. Define this in the Hardware Profile.

- #define GFX\_LCD\_TYPE [GFX\\_LCD\\_TFT](#) - sets type TFT display
- #define GFX\_LCD\_TYPE [GFX\\_LCD\\_CSTN](#) - sets type color STN display
- #define GFX\_LCD\_TYPE [GFX\\_LCD\\_MSTN](#) - sets type mon STN display
- #define GFX\_LCD\_TYPE [GFX\\_LCD\\_OFF](#) - display is turned off

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [GFX\\_LCD\\_TYPE Macro](#)

# GFX\_LCD\_CSTN Macro

```
C
#define GFX_LCD_CSTN 0x03           // Type Color S
```

## Description

Type Color STN Display

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [GFX\\_LCD\\_TYPE Macro](#) > [GFX\\_LCD\\_CSTN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_LCD\_MSTN Macro

```
C
#define GFX_LCD_MSTN 0x02           // Type Mono STN
```

## Description

Type Mono STN Display

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [GFX\\_LCD\\_TYPE Macro](#) > [GFX\\_LCD\\_MSTN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_LCD\_OFF Macro

---

```
C
#define GFX_LCD_OFF 0x00           // display is tu
```

### Description

---

display is turned off

---

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [GFX\\_LCD\\_TYPE Macro](#) > [GFX\\_LCD\\_OFF Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_LCD\_TFT Macro

---

```
C
#define GFX_LCD_TFT 0x01           // Type TFT Disp.
```

## Description

---

Type TFT Display

---

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [GFX\\_LCD\\_TYPE Macro](#) > [GFX\\_LCD\\_TFT Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# STN\_DISPLAY\_WIDTH Macro

## Macros

```
C
```

```
#define STN_DISPLAY_WIDTH
```

## Overview

Sets the STN glass data width. Define this in the Hardware Profile.

- #define STN\_DISPLAY\_WIDTH [STN\\_DISPLAY\\_WIDTH\\_4](#) - use 4-bit wide interface
- #define STN\_DISPLAY\_WIDTH [STN\\_DISPLAY\\_WIDTH\\_8](#) - Use 8-bit wide interface
- #define STN\_DISPLAY\_WIDTH [STN\\_DISPLAY\\_WIDTH\\_16](#) - Use 16-bit wide interface

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [STN\\_DISPLAY\\_WIDTH Macro](#)

## STN\_DISPLAY\_WIDTH\_16 Macro

---

```
C
#define STN_DISPLAY_WIDTH_16 0x02 // display inte
```

### Description

---

display interface is 16 bits wide

---

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [STN\\_DISPLAY\\_WIDTH Macro](#) > [STN\\_DISPLAY\\_WIDTH\\_16 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## STN\_DISPLAY\_WIDTH\_4 Macro

```
C
#define STN_DISPLAY_WIDTH_4 0x00 // display inter
```

### Description

display interface is 4 bits wide

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [STN\\_DISPLAY\\_WIDTH Macro](#) > [STN\\_DISPLAY\\_WIDTH\\_4 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## STN\_DISPLAY\_WIDTH\_8 Macro

```
C
#define STN_DISPLAY_WIDTH_8 0x01 // display inter
```

### Description

display interface is 8 bits wide

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [STN\\_DISPLAY\\_WIDTH Macro](#) > [STN\\_DISPLAY\\_WIDTH\\_8 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Application Configuration

## Topics

### Topics

Name	Description
<a href="#">Configuration Setting</a>	<p>This selects the configuration of the library. When Non-blocking configuration is selected, state machine based rendering is used to perform object rendering. When blocking configuration is used, this line MUST be commented. In this case object rendering will not exit until the object is fully rendered. The <a href="#">USE_NONBLOCKING_CONFIG</a> option is located in the GraphicsConfig.h header file.</p>
<a href="#">Font Source Selection</a>	<p>Font data can be placed in multiple locations. Set these options in the GraphicsConfig.h header file.</p> <ul style="list-style-type: none"><li>- <a href="#">USE_FONT_FLASH</a> - Font in internal flash memory support. When placed in internal flash memory, it can further classified to be placed in program flash by adding <a href="#">USE_GFX_FONT_IN_PROGRAM_SECTION</a>.</li><li>- <a href="#">USE_FONT_EXTERNAL</a> - Font in external memory support. Use this for fonts located in external memory like SPI Flash or external memory mapped to Extended Data Space.</li></ul>
<a href="#">Image Source Selection</a>	<p>Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the</p>

	support for bitmaps located in internal flash and external memory. The <a href="#">USE_BITMAP_FLASH</a> and <a href="#">USE_BITMAP_EXTERNAL</a> options are located in the GraphicsConfig.h header file.
<a href="#">Miscellaneous</a>	This contains miscellaneous macros and functions that can be redefined for various system support such as Operating System defined functions.
<a href="#">GraphicsConfig.h Example</a>	This is an example of the GraphicsConfig.h file implementation:

## Links

---

[Graphics Library Configuration, Topics](#)

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Configuration Setting

## Macros

This selects the configuration of the library. When Non-blocking configuration is selected, state machine based rendering is used to perform object rendering.

When blocking configuration is used, this line **MUST** be commented. In this case object rendering will not exit until the object is fully rendered.

The [USE\\_NONBLOCKING\\_CONFIG](#) option is located in the GraphicsConfig.h header file.

## Macros

Name	Description
<a href="#">USE_NONBLOCKING_CONFIG</a>	Blocking and Non-Blocking configuration selection. To enable non-blocking configuration USE_NONBLOCKING_CONFIG must be defined. If this is not defined, blocking configuration is assumed. Define this in GraphicsConfig.h

## Links

[Application Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Configuration Setting](#)

# USE\_NONBLOCKING\_CONFIG Macro

```
C
```

```
#define USE_NONBLOCKING_CONFIG
```

## Overview

Blocking and Non-Blocking configuration selection. To enable non-blocking configuration USE\_NONBLOCKING\_CONFIG must be defined. If this is not defined, blocking configuration is assumed. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Configuration Setting](#) > [USE\\_NONBLOCKING\\_CONFIG Macro](#)

## Font Source Selection

### Macros

Font data can be placed in multiple locations. Set these options in the GraphicsConfig.h header file.

- [USE\\_FONT\\_FLASH](#) - Font in internal flash memory support. When placed in internal flash memory, it can further be classified to be placed in program flash by adding [USE\\_GFX\\_FONT\\_IN\\_PROGRAM\\_SECTION](#).

- [USE\\_FONT\\_EXTERNAL](#) - Font in external memory support. Use this for fonts located in external memory like SPI Flash or external memory mapped to Extended Data Space.

### Macros

Name	Description
<a href="#"><u>USE_FONT_FLASH</u></a>	Font data can be placed in internal flash memory. One is in FLASH memory from external memory. enables the support for internal flash and external memory. Set this in GraphicsConfig.h. <ul style="list-style-type: none"><li>• <a href="#"><u>USE_FONT_FLASH</u></a> - Font in internal flash memory support</li><li>• <a href="#"><u>USE_FONT_EXTERNAL</u></a> - Font in external memory support</li></ul>
<a href="#"><u>USE_FONT_EXTERNAL</u></a>	Font data can be placed in external memory. One is in FLASH memory from external memory.

	<p>enables the support for internal flash and external flash memory. This is defined in GraphicsConfig.h.</p> <ul style="list-style-type: none"> <li>• <code>USE_FONT_FLASH</code> enables support for internal flash memory.</li> <li>• <code>USE_FONT_EXTERNAL</code> enables support for external memory.</li> </ul>
<p><u><a href="#">USE_GFX_FONT_IN_PROGRAM_SECTION</a></u></p>	<p>For XC16 or C30 build fonts in internal data memory. This macro limits the amount of data space for fonts to 32K. When this macro is defined and the font size exceeds 32K, the font will be relocated to external memory. This macro will remove the 32K restriction on font size at the expense of slower access times. This macro is defined in GraphicsConfig.h to enable fonts to be placed in program space.</p>

## Links

### [Application Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Font Source Selection](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_FONT\_FLASH Macro

```
C
```

```
#define USE_FONT_FLASH
```

## Overview

Font data can be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for fonts located in internal flash and external memory. Define this in GraphicsConfig.h

- `USE_FONT_FLASH` - Font in internal flash memory support.
- `USE_FONT_EXTERNAL` - Font in external memory support (including external memory mapped to EDS).

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Font Source Selection](#) > [USE\\_FONT\\_FLASH Macro](#)

# USE\_FONT\_EXTERNAL Macro

```
C
```

```
#define USE_FONT_EXTERNAL
```

## Overview

Font data can be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for fonts located in internal flash and external memory. Define this in GraphicsConfig.h

- USE\_FONT\_FLASH - Font in internal flash memory support.
- USE\_FONT\_EXTERNAL - Font in external memory support (including external memory mapped to EDS).

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Font Source Selection](#) > [USE\\_FONT\\_EXTERNAL Macro](#)

# USE\_GFX\_FONT\_IN\_PROGRAM\_SECTION Macro

```
C
```

```
#define USE_GFX_FONT_IN_PROGRAM_SECTION
```

## Overview

For XC16 or C30 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 `USE_GFX_FONT_IN_PROGRAM_SECTION` will relocate the font in program space. This will remove the 32K restriction but at the expense of slower access. Define this in `GraphicsConfig.h` to enable the font to be placed in program space.

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Font Source Selection](#) > [USE\\_GFX\\_FONT\\_IN\\_PROGRAM\\_SECTION Macro](#)

# Image Source Selection

## Macros

Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash and external memory.

The [USE\\_BITMAP\\_FLASH](#) and [USE\\_BITMAP\\_EXTERNAL](#) options are located in the GraphicsConfig.h header file.

## Macros

Name	Description
<a href="#"><u>USE_BITMAP_FLASH</u></a>	<p>Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash and external memory. Define this in GraphicsConfig.h</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_BITMAP_FLASH</u></a> - Images located in internal flash memory.</li><li>• <a href="#"><u>USE_BITMAP_EXTERNAL</u></a> - Images located in external memory (including external memory mapped to EDS)..</li></ul>
<a href="#"><u>USE_BITMAP_EXTERNAL</u></a>	<p>Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or</p>

both enables the support for bitmaps located in internal flash and external memory. Define this in GraphicsConfig.h

- `USE_BITMAP_FLASH` - Images located in internal flash memory.
- `USE_BITMAP_EXTERNAL` - Images located in external memory (including external memory mapped to EDS)..

## Links

---

[Application Configuration, Macros](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Image Source Selection](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## USE\_BITMAP\_FLASH Macro

```
C
```

```
#define USE_BITMAP_FLASH
```

### Overview

Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash and external memory. Define this in GraphicsConfig.h

- `USE_BITMAP_FLASH` - Images located in internal flash memory.
- `USE_BITMAP_EXTERNAL` - Images located in external memory (including external memory mapped to EDS)..

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Image Source Selection](#) > [USE\\_BITMAP\\_FLASH Macro](#)

## USE\_BITMAP\_EXTERNAL Macro

```
C
```

```
#define USE_BITMAP_EXTERNAL
```

### Overview

Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash and external memory. Define this in GraphicsConfig.h

- USE\_BITMAP\_FLASH - Images located in internal flash memory.
- USE\_BITMAP\_EXTERNAL - Images located in external memory (including external memory mapped to EDS)..

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Image Source Selection](#) > [USE\\_BITMAP\\_EXTERNAL Macro](#)

# Miscellaneous

## Macros

This contains miscellaneous macros and functions that can be redefined for various system support such as Operating System defined functions.

## Macros

Name	Description
<a href="#"><u>USE_BITMAP_NO_PADDING_LINE</u></a>	When this macro is enabled, bitmap images used has no padding. Define this in GraphicsConfig.h. 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 bitmaps are padded, this means that each horizontal line will start on a byte boundary. Unpadded bitmaps allows the least resource space for a bitmap. Unpadded bitmaps also allows support for display controllers with windowing and auto-increment features.
<a href="#"><u>USE_PALETTE_EXTERNAL</u></a>	Palettes can also be specified to reside in external memory similar to fonts and images. Use this when the palette is located in external memory.

	Define this in GraphicsConfig.h
<a href="#"><u>USE_PALETTE</u></a>	Using Palettes, different colors can be used with the same bit depth. Define this in GraphicsConfig.h
<a href="#"><u>COLOR_DEPTH</u></a>	Specifies the color depth used in the application defined in GraphicsConfig.h.
<a href="#"><u>GFX_free</u></a>	When using Operating Systems (OS), define the OS specific malloc() and free() functions for compatibility with the OS based systems. Define these in GraphicsConfig.h
<a href="#"><u>GFX_malloc</u></a>	When using Operating Systems (OS), define the OS specific malloc() and free() functions for compatibility with the OS based systems. Define these in GraphicsConfig.h

## Links

[Application Configuration, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Miscellaneous](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## USE\_BITMAP\_NO\_PADDING\_LINE Macro

```
C
```

```
#define USE_BITMAP_NO_PADDING_LINE
```

### Overview

When this macro is enabled, bitmap images used has no padding. Define this in GraphicsConfig.h. 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 bitmaps are padded, this means that each horizontal line will start on a byte boundary. Unpadded bitmaps allows the least resource space for a bitmap. Unpadded bitmaps also allows support for display controllers with windowing and auto-increment features.

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Miscellaneous](#) > [USE\\_BITMAP\\_NO\\_PADDING\\_LINE Macro](#)

# USE\_PALETTE\_EXTERNAL Macro

---

```
C
```

```
#define USE_PALETTE_EXTERNAL
```

## Overview

---

Palettes can also be specified to reside in external memory similar to fonts and images. Use this when the palette is located in external memory. Define this in GraphicsConfig.h

---

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Miscellaneous](#) > [USE\\_PALETTE\\_EXTERNAL Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_PALETTE Macro

```
C
```

```
#define USE_PALETTE
```

## Overview

Using Palettes, different colors can be used with the same bit depth. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Miscellaneous](#) > [USE\\_PALETTE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# COLOR\_DEPTH Macro

```
C
```

```
#define COLOR_DEPTH 16
```

## Overview

Specifies the color depth used in the application defined in GraphicsConfig.h.

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Miscellaneous](#) > [COLOR\\_DEPTH Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_free Macro

```
C
#define GFX_free(pObj) free(pObj) // <COPY GF
```

## Overview

When using Operating Systems (OS), define the OS specific malloc() and free() functions for compatibility with the OS based systems. Define these in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Miscellaneous](#) > [GFX\\_free Macro](#)

# GFX\_malloc Macro

```
C
```

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

## Overview

When using Operating Systems (OS), define the OS specific malloc() and free() functions for compatibility with the OS based systems. Define these in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Miscellaneous](#) > [GFX\\_malloc Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GraphicsConfig.h Example

This is an example of the GraphicsConfig.h file implementation:

[Copy Code](#)

```
////////////////////// COMPILE OPTIONS AND DEFAULTS //  
  
#define USE_NONBLOCKING_CONFIG // Comment t  
#define USE_FOCUS // Comment t  
#define USE_TOUCHSCREEN // Enable to  
#define USE_KEYBOARD // Enable ke  
  
#define USE_GOL // Enable Gr  
#define USE_BUTTON // Enable Bu  
#define USE_CHART // Enable Ch  
#define USE_WINDOW // Enable Wi  
#define USE_CHECKBOX // Enable Ch  
#define USE_RADIOBUTTON // Enable Ra  
#define USE_EDITBOX // Enable Ed  
#define USE_LISTBOX // Enable Li  
#define USE_SLIDER // Enable Sl  
#define USE_PROGRESSBAR // Enable Pr  
#define USE_STATICTEXT // Enable St  
#define USE_PICTURE // Enable Pi  
#define USE_GROUPBOX // Enable Gr  
#define USE_ROUNDIDIAL // Enable Di  
#define USE_METER // Enable Me  
#define USE_TEXTENTRY // Enable Te  
#define USE_CUSTOM // Enable Cu  
  
#define USE_MULTIBYTECHAR // Enable un  
#define USE_FONT_FLASH // Support f  
#define USE_BITMAP_FLASH // Support f
```

```
#define GOL_EMOSS_SIZE 2
#define COLOR_DEPTH 16 // The c
```

## Links

---

### [Application Configuration](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [GraphicsConfig.h Example](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Hardware Profile

## Topics

These functions and macros are used to determine hardware profile settings on the chosen PIC microcontroller and hardware such as demo boards used.

## Topics

Name	Description
<a href="#">PMP Interface</a>	Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).
<a href="#">Development Platform Used</a>	Specifies the Development Platform used for the Microchip Graphics Library demos.
<a href="#">Graphics PICtail Used</a>	Specifies the Graphics PICtail Display Panel used.
<a href="#">Display Controller Used</a>	Specifies the controller used in the Graphics Library supplied demo.
<a href="#">Display Panel Used</a>	Specifies the Graphics Display Panel used.
<a href="#">Device Driver Options</a>	The options Graphics Hardware Platform, DISPLAY_CONTROLLER and DISPLAY_PANEL are specific to the hardware used. The Graphics Hardware Platform selects the Graphics PICtail™ Plus Board version, PIC24FJ256DA210 Development Board or any other Microchip demo boards for the Graphics Library. Currently there are two Graphics PICtail™ Plus Board versions supported as shown in

	<p>the <a href="#">Getting Started</a> section.</p> <p>The rest of the settings are used to specify the the display parameters when using an RGB type display controller such as SSD1906 and SSD1926 from Solomon Systech. The table below summarizes the generic parameters found in RGB type display controllers and when each type is used.... <a href="#">more</a></p>
<a href="#">HardwareProfile.h Example</a>	<p>This is an example of the HardwareProfile.h file implementation:</p>

## Links

---

[Graphics Library Configuration, Topics](#)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PMP Interface

## Macros

Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).

## Macros

Name	Description
<a href="#"><u>USE_8BIT_PMP</u></a>	<p>Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).</p> <ul style="list-style-type: none"><li>• <code>USE_8BIT_PMP</code> - Use 8-bit interface to PMP or EPMP</li><li>• <a href="#"><u>USE_16BIT_PMP</u></a> - Use 16-bit interface to PMP or EPMP</li></ul>
<a href="#"><u>USE_16BIT_PMP</u></a>	<p>Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).</p> <ul style="list-style-type: none"><li>• <code>USE_8BIT_PMP</code> - Use 8-bit interface to PMP or EPMP</li><li>• <a href="#"><u>USE_16BIT_PMP</u></a> - Use 16-bit interface to PMP or EPMP</li></ul>

## Links

[Hardware Profile, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [PMP Interface](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_8BIT\_PMP Macro

```
C
```

```
#define USE_8BIT_PMP
```

## Overview

Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).

- `USE_8BIT_PMP` - Use 8-bit interface to PMP or EPMP
- [USE\\_16BIT\\_PMP](#) - Use 16-bit interface to PMP or EPMP

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [PMP Interface](#) > [USE\\_8BIT\\_PMP Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# USE\_16BIT\_PMP Macro

```
C
```

```
#define USE_16BIT_PMP
```

## Overview

Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).

- USE\_8BIT\_PMP - Use 8-bit interface to PMP or EPMP
- USE\_16BIT\_PMP - Use 16-bit interface to PMP or EPMP

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [PMP Interface](#) > [USE\\_16BIT\\_PMP Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Development Platform Used

### Macros

Specifies the Development Platform used for the Microchip Graphics Library demos.

### Macros

Name	Description
<a href="#">EXPLORER_16</a>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p> <ul style="list-style-type: none"><li>• <a href="#">EXPLORER_16</a> - Using the Explorer 16 Development Board (DM240001).</li><li>• <a href="#">PIC24FJ256DA210_DEV_BOARD</a> - Using the PIC24FJ256DA210 Development Board (DM240002).</li><li>• <a href="#">MEB_BOARD</a> - Using the Multimedia Expansion Board (DM320005).</li><li>• <a href="#">PIC_SK</a> - Using PIC32 or dsPIC32 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).</li></ul>
<a href="#">PIC24FJ256DA210_DEV_BOARD</a>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p> <ul style="list-style-type: none"><li>• <a href="#">EXPLORER_16</a> - Using the Explorer 16 Development Board</li></ul>

	<p>(DM240001).</p> <ul style="list-style-type: none"> <li>• <a href="#">PIC24FJ256DA210_DEV_BC</a> - Using the PIC24FJ256DA210 Development Board (DM240001).</li> <li>• <a href="#">MEB_BOARD</a> - Using the Microchip Media Expansion Board (DM320005).</li> <li>• <a href="#">PIC_SK</a> - Using PIC32 or dsPIC33 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).</li> </ul>
<p><a href="#">MEB_BOARD</a></p>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p> <ul style="list-style-type: none"> <li>• EXPLORER_16 - Using the Explorer 16 Development Board (DM240001).</li> <li>• <a href="#">PIC24FJ256DA210_DEV_BC</a> - Using the PIC24FJ256DA210 Development Board (DM240001).</li> <li>• <a href="#">MEB_BOARD</a> - Using the Microchip Media Expansion Board (DM320005).</li> <li>• <a href="#">PIC_SK</a> - Using PIC32 or dsPIC33 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).</li> </ul>
<p><a href="#">PIC_SK</a></p>	<p>Specifies the Development Platform</p>

used for the Microchip Graphics Library demos.

- EXPLORER\_16 - Using the Explorer 16 Development Board (DM240001).
- [PIC24FJ256DA210\\_DEV\\_BOARD](#) - Using the PIC24FJ256DA210 Development Board (DM240001).
- [MEB\\_BOARD](#) - Using the Multimedia Expansion Board (DM320005).
- [PIC\\_SK](#) - Using PIC32 or dsPIC32 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).

## Links

---

[Hardware Profile, Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Development Platform Used](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EXPLORER\_16 Macro

**C****#define EXPLORER\_16**

## Overview

Specifies the Development Platform used for the Microchip Graphics Library demos.

- [EXPLORER\\_16](#) - Using the Explorer 16 Development Board (DM240001).
- [PIC24FJ256DA210\\_DEV\\_BOARD](#) - Using the PIC24FJ256DA210 Development Board (DM240312).
- [MEB\\_BOARD](#) - Using the Multi-Media Expansion Board (DM320005).
- [PIC\\_SK](#) - Using PIC32 or dsPIC Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003-2), PIC32 Ethernet Starter Kit (DM320004)).

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Development Platform Used](#) > [EXPLORER\\_16 Macro](#)

# PIC24FJ256DA210\_DEV\_BOARD Macro

```
C
```

```
#define PIC24FJ256DA210_DEV_BOARD
```

## Overview

Specifies the Development Platform used for the Microchip Graphics Library demos.

- [EXPLORER\\_16](#) - Using the Explorer 16 Development Board (DM240001).
- [PIC24FJ256DA210\\_DEV\\_BOARD](#) - Using the PIC24FJ256DA210 Development Board (DM240312).
- [MEB\\_BOARD](#) - Using the Multi-Media Expansion Board (DM320005).
- [PIC\\_SK](#) - Using PIC32 or dsPIC Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003-2), PIC32 Ethernet Starter Kit (DM320004)).

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Development Platform Used](#) > [PIC24FJ256DA210\\_DEV\\_BOARD Macro](#)

# MEB\_BOARD Macro

```
C
```

```
#define MEB_BOARD
```

## Overview

Specifies the Development Platform used for the Microchip Graphics Library demos.

- [EXPLORER\\_16](#) - Using the Explorer 16 Development Board (DM240001).
- [PIC24FJ256DA210\\_DEV\\_BOARD](#) - Using the PIC24FJ256DA210 Development Board (DM240312).
- [MEB\\_BOARD](#) - Using the Multi-Media Expansion Board (DM320005).
- [PIC\\_SK](#) - Using PIC32 or dsPIC Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003-2), PIC32 Ethernet Starter Kit (DM320004)).

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Development Platform Used](#) > [MEB\\_BOARD Macro](#)

# PIC\_SK Macro

```
C
```

```
#define PIC_SK
```

## Overview

Specifies the Development Platform used for the Microchip Graphics Library demos.

- [EXPLORER\\_16](#) - Using the Explorer 16 Development Board (DM240001).
- [PIC24FJ256DA210\\_DEV\\_BOARD](#) - Using the PIC24FJ256DA210 Development Board (DM240312).
- [MEB\\_BOARD](#) - Using the Multi-Media Expansion Board (DM320005).
- [PIC\\_SK](#) - Using PIC32 or dsPIC Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003-2), PIC32 Ethernet Starter Kit (DM320004)).

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Development Platform Used](#) > [PIC\\_SK Macro](#)

# Graphics PICtail Used

## Macros

Specifies the Graphics PICtail Display Panel used.

## Macros

Name	Description
<a href="#">GFX_PICTAIL_LCC</a>	<p>Specifies the Graphics PICtail Display Panel used.</p> <ul style="list-style-type: none"><li>• <a href="#">GFX_PICTAIL_V3</a> - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)</li><li>• <a href="#">GFX_PICTAIL_V3E</a> - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)</li><li>• <a href="#">GFX_PICTAIL_LCC</a> - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)</li></ul>
<a href="#">GFX_PICTAIL_V3</a>	<p>Specifies the Graphics PICtail Display Panel used.</p> <ul style="list-style-type: none"><li>• <a href="#">GFX_PICTAIL_V3</a> - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)</li><li>• <a href="#">GFX_PICTAIL_V3E</a> - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)</li><li>• <a href="#">GFX_PICTAIL_LCC</a> - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)</li></ul>

## GFX\_PICTAIL\_V3E

Specifies the Graphics PICtail Display Panel used.

- GFX\_PICTAIL\_V3 - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)
- GFX\_PICTAIL\_V3E - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)
- GFX\_PICTAIL\_LCC - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)

## Links

[Hardware Profile](#), [Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Graphics PICtail Used](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_PICTAIL\_LCC Macro

```
C
```

```
#define GFX_PICTAIL_LCC
```

## Overview

Specifies the Graphics PICtail Display Panel used.

- GFX\_PICTAIL\_V3 - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)
- [GFX\\_PICTAIL\\_V3E](#) - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)
- GFX\_PICTAIL\_LCC - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Graphics PICtail Used](#) > [GFX\\_PICTAIL\\_LCC Macro](#)

# GFX\_PICTAIL\_V3 Macro

```
C
```

```
#define GFX_PICTAIL_V3
```

## Overview

Specifies the Graphics PICtail Display Panel used.

- `GFX_PICTAIL_V3` - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)
- [GFX\\_PICTAIL\\_V3E](#) - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)
- [GFX\\_PICTAIL\\_LCC](#) - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Graphics PICtail Used](#) > [GFX\\_PICTAIL\\_V3 Macro](#)

# GFX\_PICTAIL\_V3E Macro

```
C
```

```
#define GFX_PICTAIL_V3E
```

## Overview

Specifies the Graphics PICtail Display Panel used.

- [GFX\\_PICTAIL\\_V3](#) - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)
- [GFX\\_PICTAIL\\_V3E](#) - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)
- [GFX\\_PICTAIL\\_LCC](#) - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Graphics PICtail Used](#) > [GFX\\_PICTAIL\\_V3E Macro](#)

# Display Controller Used

## Macros

Specifies the controller used in the Graphics Library supplied demo.

## Macros

Name	Description
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_DMA</u></a>	<p>Specifies the controller used in the Graphics Library supplied demo.</p> <ul style="list-style-type: none"><li>• <code>GFX_USE_DISPLAY_CONTROLLER_DMA</code> - Use the DMA Controller (PIC32 Family)</li><li>• <a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_DMA</u></a> - Use the DMA Controller (PIC32 Family)</li><li>• <a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_DMA</u></a> - Use the DMA Controller (PIC32 Family)</li><li>• <a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_DMA</u></a> - Use the DMA Controller (PIC32 Family)</li></ul>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</u></a>	<p>Specifies the controller used in the Graphics Library supplied demo.</p> <ul style="list-style-type: none"><li>• <code>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</code> - Use the MCHP DA210 Controller (PIC32 Family)</li><li>• <a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</u></a> - Use the MCHP DA210 Controller (PIC32 Family)</li><li>• <a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</u></a> - Use the MCHP DA210 Controller (PIC32 Family)</li><li>• <a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</u></a> - Use the MCHP DA210 Controller (PIC32 Family)</li></ul>

	<p>Controller</p> <ul style="list-style-type: none"> <li>• <a href="#">GFX_USE</a> Using the</li> </ul>
<p><a href="#">GFX_USE_DISPLAY_CONTROLLER_S1D13517</a></p>	<p>Specifies the c supplied demc</p> <ul style="list-style-type: none"> <li>• GFX_USE the PIC32</li> <li>• <a href="#">GFX_USE</a> - Use the the PIC M Family)</li> <li>• <a href="#">GFX_USE</a> Using the Controller</li> <li>• <a href="#">GFX_USE</a> Using the</li> </ul>
<p><a href="#">GFX_USE_DISPLAY_CONTROLLER_SSD1926</a></p>	<p>Specifies the c supplied demc</p> <ul style="list-style-type: none"> <li>• GFX_USE the PIC32</li> <li>• <a href="#">GFX_USE</a> - Use the the PIC M Family)</li> <li>• <a href="#">GFX_USE</a> Using the Controller</li> <li>• <a href="#">GFX_USE</a> Using the</li> </ul>

## Links

---

[Hardware Profile, Macros](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Controller Used](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_USE\_DISPLAY\_CONTROLLER\_DMA Macro

```
C
```

```
#define GFX_USE_DISPLAY_CONTROLLER_DMA
```

## Overview

Specifies the controller used in the Graphics Library supplied demo.

- `GFX_USE_DISPLAY_CONTROLLER_DMA` - Using the PIC32 Low Cost Controllerless (LCC) solution.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_MCHP\\_DA210](#) - Use the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family)
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_SSD1926](#) - Using the Solomon Systech SSD1926 Display Controller.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_S1D13517](#) - Using the Epson S1D13517 Display Controller.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Controller Used](#) > [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_DMA Macro](#)

# GFX\_USE\_DISPLAY\_CONTROLLER\_MCHP\_DA210 Macro

```
C
```

```
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210
```

## Overview

Specifies the controller used in the Graphics Library supplied demo.

- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_DMA](#) - Using the PIC32 Low Cost Controllerless (LCC) solution.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_MCHP\\_DA210](#) - Use the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family)
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_SSD1926](#) - Using the Solomon Systech SSD1926 Display Controller.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_S1D13517](#) - Using the Epson S1D13517 Display Controller.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) >  
[Display Controller Used](#) >  
[GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_MCHP\\_DA210 Macro](#)

# GFX\_USE\_DISPLAY\_CONTROLLER\_S1D13517 Macro

```
C
```

```
#define GFX_USE_DISPLAY_CONTROLLER_S1D13517
```

## Overview

Specifies the controller used in the Graphics Library supplied demo.

- `GFX_USE_DISPLAY_CONTROLLER_DMA` - Using the PIC32 Low Cost Controllerless (LCC) solution.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_MCHP\\_DA210](#) - Use the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family)
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_SSD1926](#) - Using the Solomon Systech SSD1926 Display Controller.
- `GFX_USE_DISPLAY_CONTROLLER_S1D13517` - Using the Epson S1D13517 Display Controller.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) >  
[Display Controller Used](#) >  
[GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_S1D13517 Macro](#)

# GFX\_USE\_DISPLAY\_CONTROLLER\_SSD1926 Macro

```
C
```

```
#define GFX_USE_DISPLAY_CONTROLLER_SSD1926
```

## Overview

Specifies the controller used in the Graphics Library supplied demo.

- `GFX_USE_DISPLAY_CONTROLLER_DMA` - Using the PIC32 Low Cost Controllerless (LCC) solution.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_MCHP\\_DA210](#) - Use the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family)
- `GFX_USE_DISPLAY_CONTROLLER_SSD1926` - Using the Solomon Systech SSD1926 Display Controller.
- [GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_S1D13517](#) - Using the Epson S1D13517 Display Controller.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) >  
[Display Controller Used](#) >  
[GFX\\_USE\\_DISPLAY\\_CONTROLLER\\_SSD1926 Macro](#)

# Display Panel Used

## Macros

Specifies the Graphics Display Panel used.

## Macros

Name	Description
<a href="#"><u>GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q</u></a>	Specifies <ul style="list-style-type: none"><li>• <a href="#"><u>GL</u></a> - 3</li><li>• <a href="#"><u>GL</u></a> inc</li><li>• <a href="#"><u>GL</u></a> VC</li><li>• <a href="#"><u>GL</u></a> W</li></ul>
<a href="#"><u>GFX_USE_DISPLAY_PANEL_TFT_640480_8_E</u></a>	Specifies <ul style="list-style-type: none"><li>• <a href="#"><u>GL</u></a> - 3</li><li>• <a href="#"><u>GL</u></a> inc</li><li>• <a href="#"><u>GL</u></a> VC</li><li>• <a href="#"><u>GL</u></a> W</li></ul>
<a href="#"><u>GFX_USE_DISPLAY_PANEL_TFT_800480_33_E</u></a>	Specifies <ul style="list-style-type: none"><li>• <a href="#"><u>GL</u></a> - 3</li></ul>

	<ul style="list-style-type: none"> <li>• <a href="#">GI</a> inc</li> <li>• <a href="#">GI</a> VC</li> <li>• <a href="#">GI</a> W</li> </ul>
<p><a href="#">GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_E</a></p>	<p>Specifi</p> <ul style="list-style-type: none"> <li>• GI - 3</li> <li>• <a href="#">GI</a> inc</li> <li>• <a href="#">GI</a> VC</li> <li>• <a href="#">GI</a> W</li> </ul>

## Links

[Hardware Profile](#), [Macros](#)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Panel Used](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_USE\_DISPLAY\_PANEL\_PH480272T\_005\_I11 Macro

```
C
```

```
#define GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q
```

## Overview

Specifies the Graphics Display Panel used.

- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E](#) - 3.2 inch QVGA Truly TFT Display Board (AC164127-4)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_PH480272T\\_005\\_I11Q](#) - 4.3 inch WQVGA Powertip TFT Display Board (AC164127-6)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_640480\\_8\\_E](#) - 5.7 inch VGA Truly TFT Display Board (AC164127-8)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_800480\\_33\\_E](#) - 7 inch WVGA Truly TFT Display Board (AC164127-8)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) >  
[Display Panel Used](#) >  
[GFX\\_USE\\_DISPLAY\\_PANEL\\_PH480272T\\_005\\_I11Q Macro](#)

# GFX\_USE\_DISPLAY\_PANEL\_TFT\_640480\_8\_E Macro

```
C
```

```
#define GFX_USE_DISPLAY_PANEL_TFT_640480_8_E
```

## Overview

Specifies the Graphics Display Panel used.

- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E](#) - 3.2 inch QVGA Truly TFT Display Board (AC164127-4)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_PH480272T\\_005\\_I11Q](#) - 4.3 inch WQVGA Powertip TFT Display Board (AC164127-6)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_640480\\_8\\_E](#) - 5.7 inch VGA Truly TFT Display Board (AC164127-8)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_800480\\_33\\_E](#) - 7 inch WVGA Truly TFT Display Board (AC164127-8)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Panel Used](#) > [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_640480\\_8\\_E Macro](#)

# GFX\_USE\_DISPLAY\_PANEL\_TFT\_800480\_33\_E Macro

```
C
```

```
#define GFX_USE_DISPLAY_PANEL_TFT_800480_33_E
```

## Overview

Specifies the Graphics Display Panel used.

- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E](#) - 3.2 inch QVGA Truly TFT Display Board (AC164127-4)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_PH480272T\\_005\\_I11Q](#) - 4.3 inch WQVGA Powertip TFT Display Board (AC164127-6)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_640480\\_8\\_E](#) - 5.7 inch VGA Truly TFT Display Board (AC164127-8)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_800480\\_33\\_E](#) - 7 inch WVGA Truly TFT Display Board (AC164127-8)

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Panel Used](#) > [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_800480\\_33\\_E Macro](#)

# GFX\_USE\_DISPLAY\_PANEL\_TFT\_G240320LTSW\_Macro

---

**C**

```
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_E
```

---

## Overview

Specifies the Graphics Display Panel used.

- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E](#) - 3.2 inch QVGA Truly TFT Display Board (AC164127-4)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_PH480272T\\_005\\_I11Q](#) - 4.3 inch WQVGA Powertip TFT Display Board (AC164127-6)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_640480\\_8\\_E](#) - 5.7 inch VGA Truly TFT Display Board (AC164127-8)
- [GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_800480\\_33\\_E](#) - 7 inch WVGA Truly TFT Display Board (AC164127-8)

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Panel Used](#) >

[GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E Macro](#)

## Device Driver Options

### Macros

The options Graphics Hardware Platform, DISPLAY\_CONTROLLER and DISPLAY\_PANEL are specific to the hardware used. The Graphics Hardware Platform selects the Graphics PICtail™ Plus Board version, PIC24FJ256DA210 Development Board or any other Microchip demo boards for the Graphics Library. Currently there are two Graphics PICtail™ Plus Board versions supported as shown in the [Getting Started](#) section.

The rest of the settings are used to specify the the display parameters when using an RGB type display controller such as SSD1906 and SSD1926 from Solomon Systech. The table below summarizes the generic parameters found in RGB type display controllers and when each type is used.

Options	Color STN	Mono STN	TFT
<a href="#">DISP_DATA_WIDTH</a>	YES	YES	YES
<a href="#">DISP_ORIENTATION</a>	YES	YES	YES
<a href="#">DISP_HOR_RESOLUTION</a>	YES	YES	YES
<a href="#">DISP_VER_RESOLUTION</a>	YES	YES	YES
<a href="#">DISP_HOR_BACK_PORCH</a>	NO	NO	YES
<a href="#">DISP_HOR_FRONT_PORCH</a>	NO	NO	YES
<a href="#">DISP_VER_FRONT_PORCH</a>	NO	NO	YES
<a href="#">DISP_VER_BACK_PORCH</a>	NO	NO	YES
<a href="#">DISP_HOR_PULSE_WIDTH</a>	YES	YES	YES

<u>DISP_VER_PULSE_WIDTH</u>	YES	YES	YES
<u>DISP_INV_LSHIFT</u>	YES	YES	YES

All the options listed here are set in the HardwareProfile.h header file implemented in the application layer. An example of this file is shown in HardwareProfile.h Example section.

## Macros

Name	Description
<u>DISP_DATA_WIDTH</u>	<p>Defines the display controller's physical interface to the display panel. Valid Values:</p> <ul style="list-style-type: none"> <li>• 1, 4, 8, 16, 18, 24</li> <li>• 1, 4, 8 are usually used in MSTN and CSTN displays</li> <li>• 16, 18 and 24 are usually used in TFT displays.</li> </ul>
<u>DISP_ORIENTATION</u>	<p>Defines the display rotation with respect to its native orientation. For example, if the display has a resolution specifications that says 240x320 (QVGA), the display is natively in portrait mode. If the application uses the display in landscape mode (320x240), then the orientation must be defined as 90 or 180 degree rotation. Graphics Library will calculate the actual pixel location to rotate the contents of the screen. So when users view the display, the image on the screen will come out in the correct orientation.</p>

	<p>Valid values:</p> <ul style="list-style-type: none"> <li>• 0 : display in its native orientation</li> <li>• 90 : rotated 90 degrees clockwise... <a href="#">more</a></li> </ul>
<u>DISP_HOR_RESOLUTION</u>	<p>Defines the native horizontal dimension of the screen. This is the horizontal pixel count given by the display's data sheet. For example a 320x240 display will have DISP_HOR_RESOLUTION of 320.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• dependent on the display glass resolution used.</li> </ul>
<u>DISP_VER_RESOLUTION</u>	<p>Defines the native vertical dimension of the screen. This is the vertical pixel count given by the display's data sheet. For example a 320x240 display will have DISP_VER_RESOLUTION of 240.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• dependent on the display glass resolution used.</li> </ul>
<u>DISP_HOR_FRONT_PORCH</u>	<p>Defines the horizontal front porch. <u>DISP_HOR_BACK_PORCH</u> + <u>DISP_HOR_FRONT_PORCH</u> + <u>DISP_HOR_PULSE_WIDTH</u> makes up the horizontal blanking period. Value used will be based on the display panel used.</p>
<u>DISP_HOR_BACK_PORCH</u>	<p>Defines the horizontal back porch.</p>

	<p>DISP_HOR_BACK_PORCH +  <u>DISP_HOR_FRONT_PORCH</u> +  <u>DISP_HOR_PULSE_WIDTH</u> makes up the horizontal blanking period. Value used will be based on the display panel used.</p>
<u>DISP_VER_FRONT_PORCH</u>	<p>Defines the vertical front porch. <u>DISP_VER_BACK_PORCH</u> + DISP_VER_FRONT_PORCH + <u>DISP_VER_PULSE_WIDTH</u> makes up the vertical blanking period. Value used will be based on the display panel used.</p>
<u>DISP_VER_BACK_PORCH</u>	<p>Defines the vertical back porch. DISP_VER_BACK_PORCH + <u>DISP_VER_FRONT_PORCH</u> + <u>DISP_VER_PULSE_WIDTH</u> makes up the vertical blanking period. Value used will be based on the display panel used.</p>
<u>DISP_HOR_PULSE_WIDTH</u>	<p>Defines the horizontal sync signal pulse width in pixels. Value used will be based on the display panel used.</p>
<u>DISP_VER_PULSE_WIDTH</u>	<p>Defines the vertical sync signal pulse width in lines. Value used will be based on the display panel used.</p>
<u>DISP_INV_LSHIFT</u>	<p>Indicates that the color data is sampled in the falling edge of the pixel clock.</p>

## Links

## Hardware Profile, Macros

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DISP\_DATA\_WIDTH Macro

```
C
```

```
#define DISP_DATA_WIDTH 18
```

### Overview

Defines the display controller's physical interface to the display panel. Valid Values:

- 1, 4, 8, 16, 18, 24
- 1, 4, 8 are usually used in MSTN and CSTN displays
- 16, 18 and 24 are usually used in TFT displays.

### Example

[Copy Code](#)

```
// define in Hardware Profile  
#define DISP_DATA_WIDTH 18
```

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_DATA\\_WIDTH Macro](#)

## DISP\_ORIENTATION Macro

```
C
```

```
#define DISP_ORIENTATION 0
```

### Overview

Defines the display rotation with respect to its native orientation. For example, if the display has a resolution specifications that says 240x320 (QVGA), the display is natively in portrait mode. If the application uses the display in landscape mode (320x240), then the orientation must be defined as 90 or 180 degree rotation. Graphics Library will calculate the actual pixel location to rotate the contents of the screen. So when users view the display, the image on the screen will come out in the correct orientation. Valid values:

- 0 : display in its native orientation
- 90 : rotated 90 degrees clockwise direction
- 180 : rotated 180 degrees clockwise direction
- 270 : rotated 270 degrees clockwise direction

### Example

[Copy Code](#)

```
// define in Hardware Profile  
#define DISP_ORIENTATION 90
```

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_ORIENTATION Macro](#)

## DISP\_HOR\_RESOLUTION Macro

```
C
```

```
#define DISP_HOR_RESOLUTION 320
```

### Overview

Defines the native horizontal dimension of the screen. This is the horizontal pixel count given by the display's data sheet. For example a 320x240 display will have DISP\_HOR\_RESOLUTION of 320. Valid Values:

- dependent on the display glass resolution used.

### Example

[Copy Code](#)

```
// define in Hardware Profile  
#define DISP_HOR_RESOLUTION 320
```

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_HOR\\_RESOLUTION Macro](#)

## DISP\_VER\_RESOLUTION Macro

```
C
```

```
#define DISP_VER_RESOLUTION 240
```

### Overview

Defines the native vertical dimension of the screen. This is the vertical pixel count given by the display's data sheet. For example a 320x240 display will have DISP\_VER\_RESOLUTION of 240. Valid Values:

- dependent on the display glass resolution used.

### Example

[Copy Code](#)

```
// define in Hardware Profile  
#define DISP_VER_RESOLUTION 240
```

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_VER\\_RESOLUTION Macro](#)

## DISP\_HOR\_FRONT\_PORCH Macro

```
C
```

```
#define DISP_HOR_FRONT_PORCH 10
```

### Overview

Defines the horizontal front porch. [DISP\\_HOR\\_BACK\\_PORCH](#) + [DISP\\_HOR\\_FRONT\\_PORCH](#) + [DISP\\_HOR\\_PULSE\\_WIDTH](#) makes up the horizontal blanking period. Value used will be based on the display panel used.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_HOR\\_FRONT\\_PORCH Macro](#)

## DISP\_HOR\_BACK\_PORCH Macro

```
C
```

```
#define DISP_HOR_BACK_PORCH 20
```

### Overview

Defines the horizontal back porch. `DISP_HOR_BACK_PORCH + DISP\_HOR\_FRONT\_PORCH + DISP\_HOR\_PULSE\_WIDTH` makes up the horizontal blanking period. Value used will be based on the display panel used.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_HOR\\_BACK\\_PORCH Macro](#)

## DISP\_VER\_FRONT\_PORCH Macro

```
C
```

```
#define DISP_VER_FRONT_PORCH 5
```

### Overview

Defines the vertical front porch. [DISP\\_VER\\_BACK\\_PORCH](#) + [DISP\\_VER\\_FRONT\\_PORCH](#) + [DISP\\_VER\\_PULSE\\_WIDTH](#) makes up the vertical blanking period. Value used will be based on the display panel used.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_VER\\_FRONT\\_PORCH Macro](#)

## DISP\_VER\_BACK\_PORCH Macro

```
C
```

```
#define DISP_VER_BACK_PORCH 20
```

### Overview

Defines the vertical back porch. `DISP_VER_BACK_PORCH + DISP\_VER\_FRONT\_PORCH + DISP\_VER\_PULSE\_WIDTH` makes up the vertical blanking period. Value used will be based on the display panel used.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_VER\\_BACK\\_PORCH Macro](#)

# DISP\_HOR\_PULSE\_WIDTH Macro

```
C
```

```
#define DISP_HOR_PULSE_WIDTH 10
```

## Overview

Defines the horizontal sync signal pulse width in pixels. Value used will be based on the display panel used.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_HOR\\_PULSE\\_WIDTH Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DISP\_VER\_PULSE\_WIDTH Macro

```
C
```

```
#define DISP_VER_PULSE_WIDTH 5
```

## Overview

Defines the vertical sync signal pulse width in lines. Value used will be based on the display panel used.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_VER\\_PULSE\\_WIDTH Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DISP\_INV\_LSHIFT Macro

```
C
```

```
#define DISP_INV_LSHIFT 18
```

## Overview

Indicates that the color data is sampled in the falling edge of the pixel clock.

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Device Driver Options](#) > [DISP\\_INV\\_LSHIFT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## HardwareProfile.h Example

This is an example of the HardwareProfile.h file implementation:

[Copy Code](#)

```
/*
 * GetSystemClock() returns system clock frequency.
 * GetPeripheralClock() returns peripheral clock frequency.
 * GetInstructionClock() returns instruction clock frequency.
 */

/*
 * Macro: #define GetSystemClock()
 * Overview: This macro returns the system clock frequency.
 *           * value is 8 MHz x 4 PLL for PIC24
 *           * value is 8 MHz/2 x 18 PLL for PIC32
 */
#if defined(__PIC24F__)
    #define GetSystemClock() (32000000uL)
#elif defined(__PIC32MX__)
    #define GetSystemClock() (72000000uL)
#elif defined(__dsPIC33F__) || defined(__PIC24H__)
    #define GetSystemClock() (80000000uL)
#endif

/*
 * Macro: #define GetPeripheralClock()
 * Overview: This macro returns the peripheral clock frequency
 *           used in Hertz.
 *           * value for PIC24 is <PRE>(GetSystemClock())
 *           * value for PIC32 is <PRE>(GetSystemClock())
 */
#if defined(__PIC24F__) || defined(__PIC24H__)
    #define GetPeripheralClock() (GetSystemClock())

```

```

    #elif defined(__PIC32MX__)
        #define GetPeripheralClock()    (GetSystemC
    #endif

/*****
* Macro: #define    GetInstructionClock()
* Overview: This macro returns instruction clock fr
*           used in Hertz.
*           * value for PIC24 is <PRE>(GetSystemCl
*           * value for PIC32 is (GetSystemClock()
*****/
    #if defined(__PIC24F__) || defined(__PIC24H__)
        #define GetInstructionClock()    (GetSystemC
    #elif defined(__PIC32MX__)
        #define GetInstructionClock()    (GetSystemC
    #endif

//Auto Generated Code
#define PIC24FJ256DA210_DEV_BOARD
#define USE_16BIT_PMP
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_
#define GFX_GCLK_DIVIDER 61
#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00020000
#define GFX_DISPLAY_BUFFER_LENGTH 0x00025800u1
#define GFX_EPMP_CS1_BASE_ADDRESS 0x00020000u1
#define GFX_EPMP_CS1_MEMORY_SIZE 0x40000u1
//End Auto Generated Code

/*****
* External Memory Programmer Settings
*****/
#if defined (EXPLORER_16)
    #define USE_COMM_PKT_MEDIA_SERIAL_PORT
    #define BAUDRATE2                115200UL
    #define BRG_DIV2                  4

```

```
    #define BRGH2 1
#else
    #define USE_COMM_PKT_MEDIA_USB

    // #define USE_SELF_POWER_SENSE_IO
    #define tris_self_power TRISAbits.TRISA2
    #define self_power 1

    // #define USE_USB_BUS_SENSE_IO
    #define tris_usb_bus_sense TRISBbits.TRISB5
    #define USB_BUS_SENSE U10TGSTATbits.SESVD

#endif

#define COMM_PKT_RX_MAX_SIZE (1024)

.....
```

## Links

---

### [Hardware Profile](#)

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [HardwareProfile.h Example](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Graphics Object Layer API

## Topics

### Topics

Name	Description
<a href="#">GOL Objects</a>	The Graphics Object Layer (GOL) contains the Advanced Graphics Objects or commonly known as widgets.
<a href="#">Object States</a>	Objects rendered on the display are based on their current Property States and the Drawing States.
<a href="#">Object Management</a>	This section describes the API functions and macros that are used to create, maintain and render individual and list of objects.
<a href="#">GOL Messages</a>	The library provides an interface to accept messages from the input devices.
<a href="#">Style Scheme</a>	All objects uses a style scheme structure that defines the font and colors used.
<a href="#">GOL Global Variables</a>	Graphics Object Layer global variables.

### Links

[Library API, Topics](#)

[Library API](#) > [Graphics Object Layer API](#)

# GOL Objects

[Enumerations](#) | [Modules](#) | [Structures](#) | [Types](#)

All the GOL objects that will be shown in the display have a corresponding data structure that keeps and maintains its parameters.

Each object type has a set of functions that enables the user to change the state of the object. The Microchip graphics library supports the following set of GOL objects.

Object	Type Definition
<a href="#">Button</a>	OBJ_BUTTON
<a href="#">Chart</a>	OBJ_CHART
<a href="#">Checkbox</a>	OBJ_CHECKBOX
<a href="#">Dial</a>	OBJ_ROUNDIDIAL
<a href="#">Digital Meter</a>	OBJ_DIGITALMETER
<a href="#">Edit Box</a>	OBJ_EDITBOX
<a href="#">Grid</a>	OBJ_GRID
<a href="#">Group Box</a>	OBJ_GROUPBOX
<a href="#">List Box</a>	OBJ_LISTBOX
<a href="#">Meter</a>	OBJ_METER
<a href="#">Picture</a>	OBJ_PICTURE
<a href="#">Progress Bar</a>	OBJ_PROGRESSBAR
<a href="#">Radio Button</a>	OBJ_RADIOBUTTON
<a href="#">Slider</a>	OBJ_SLIDER

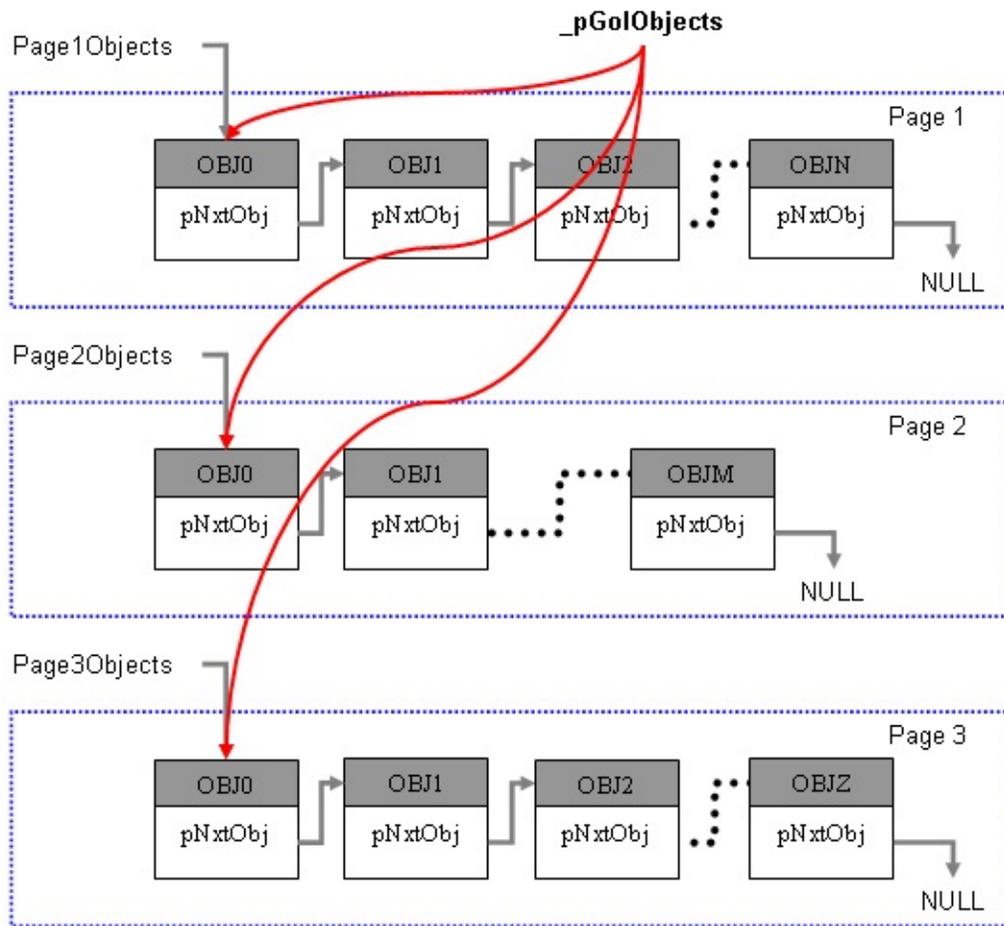
<u>Static Text</u>	OBJ_STATICTEXT
<u>Text Entry</u>	OBJ_TEXTENTRY
<u>Window</u>	OBJ_WINDOW

Each GOL object type uses a style scheme. The style scheme defines the font and color settings. User can create new style schemes and assign it to objects. Multiple style schemes can be used for different objects.

To efficiently manage the different objects the first 9 fields of the structure for each object are defined the same. Collectively they are referred to as the object header structure (OBJ\_HEADER). Defining the OBJ\_HEADER enables the common APIs to manage the objects of different types in the same manner.

The GOL operation is centered on two major processes. First is the rendering process and second is the messaging process. These processes make use of linked list objects. The field `pNxtObj` in OBJ\_HEADER makes this possible. To manage the list a global pointer \_pGolObjects is utilized. It defines the current active linked list. Newly created objects are automatically added to the end of this list. The active linked list is the list that the messaging and rendering operation parses to evaluate if the messages received affect the objects and if objects need to be redrawn.

The use of the \_pGolObjects pointer also provides a way to easily manage objects. In applications where multiple pages are displayed on the screen, objects can be grouped according to pages. The pointer can be manipulated to switch from one page to another depending on the page currently shown on the screen.



## Implementation of Multi-Page Objects

User can remove an object from a list by pointer manipulation. Objects that are removed from the list are not accessible by the rendering and messaging processes.

## Enumerations

Name	Description
<u>GOL_OBJ_TYPE</u>	This structure defines the Object types used in the library.

## Modules

Name	Description
<a href="#">Analog Clock</a>	Analog Clock is an Object that emulates an analog clock with moving hands. It can be used with or without a bitmap image as the background source.
<a href="#">Button</a>	Button is an Object that emulates a press and release effect when operated upon.
<a href="#">Chart</a>	Chart is an Object that draws a bar chart or a pie chart representation of a single data or series of data.
<a href="#">Checkbox</a>	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.
<a href="#">Round Dial</a>	Dial is an Object that can be used to display emulate a turn dial that can both go in clockwise or counterclockwise.
<a href="#">Digital Meter</a>	DigitalMeter 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. A limitation of this Object is that the font used should have equal character widths.
<a href="#">Edit Box</a>	Edit Box is is an Object that emulates a cell or a text area that can be edited dynamically.
<a href="#">Grid</a>	Grid is an Object that draws a grid on the

	screen with each cell capable of displaying an image or a string.
<a href="#"><u>Group Box</u></a>	Group Box is an Object that can be used to group Objects together in the screen.
<a href="#"><u>List Box</u></a>	List Box is an Object that defines a scrollable area where items are listed. User can select a single item or set of items.
<a href="#"><u>Meter</u></a>	Meter is an Object that can be used to graphically display a sampled input.
<a href="#"><u>Picture Control</u></a>	Picture is an Object that can be used to transform a bitmap to be an Object in the screen and have control on the bitmap rendering. This object can be used to create animation using a series of bitmaps.
<a href="#"><u>Progress Bar</u></a>	Progress <a href="#"><u>Bar</u></a> is an Object that can be used to display the progress of a task such as a file download or transfer.
<a href="#"><u>Radio Button</u></a>	Radio <a href="#"><u>Button</u></a> 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.
<a href="#"><u>Slider/Scroll Bar</u></a>	Slider or Scrollbar is an Object that can be used to display a value or scrolling location in a predefined area.
<a href="#"><u>Static Text</u></a>	Static Text is an Object that can be used to display a single or multi-line string of text in a predefined location.
<a href="#"><u>Text Entry</u></a>	Text Entry is an Object that can be used to

	<p>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.</p> <p>Application can utilize the command key to define the behavior of the program based on a command key press.</p>
<a href="#">Window</a>	<p>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 <a href="#">Bar</a>. It also has additional controls for both Title <a href="#">Bar</a> and Client Area.</p>

## Structures

Name	Description
<a href="#">OBJ_HEADER</a>	This structure defines the first nine fields of the Objects structure. This allows generic operations on library Objects.

## Types

Name	Description
<a href="#">DRAW_FUNC</a>	object draw function pointer typedef
<a href="#">FREE_FUNC</a>	object free function pointer typedef

<a href="#"><u>MSG_DEFAULT_FUNC</u></a>	object default message function pointer typedef
<a href="#"><u>MSG_FUNC</u></a>	object message function pointer typedef

## Links

---

[Enumerations](#), [Graphics Object Layer API](#), [Modules](#), [Structures](#), [Types](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GOL\_OBJ\_TYPE Enumeration

C

```
typedef enum {  
    OBJ_BUTTON,  
    OBJ_WINDOW,  
    OBJ_CHECKBOX,  
    OBJ_RADIOBUTTON,  
    OBJ_EDITBOX,  
    OBJ_LISTBOX,  
    OBJ_SLIDER,  
    OBJ_PROGRESSBAR,  
    OBJ_STATICTEXT,  
    OBJ_PICTURE,  
    OBJ_GROUPBOX,  
    OBJ_CUSTOM,  
    OBJ_ROUNDIAL,  
    OBJ_METER,  
    OBJ_GRID,  
    OBJ_CHART,  
    OBJ_TEXTENTRY,  
    OBJ_DIGITALMETER,  
    OBJ_ANALOGCLOCK,  
    OBJ_UNKNOWN  
} GOL_OBJ_TYPE;
```

### Overview

This structure defines the Object types used in the library.

### Members

Members	Description
OBJ_BUTTON	Type defined for <a href="#">Button</a> Object.

OBJ_WINDOW	Type defined for <a href="#">Window</a> Object.
OBJ_CHECKBOX	Type defined for Check Box Object.
OBJ_RADIOBUTTON	Type defined for Radio <a href="#">Button</a> Object.
OBJ_EDITBOX	Type defined for Edit Box Object.
OBJ_LISTBOX	Type defined for List Box Object.
OBJ_SLIDER	Type defined for <a href="#">Slider</a> and/or <a href="#">Scroll Bar</a> Object.
OBJ_PROGRESSBAR	Type defined for Progress Object.
OBJ_STATICTEXT	Type defined for Static Text Object.
OBJ_PICTURE	Type defined for <a href="#">Picture</a> or Bitmap Object.
OBJ_GROUPBOX	Type defined for Group Box Object.
OBJ_CUSTOM	Type defined for Custom Object.
OBJ_ROUNDIAL	Type defined for <a href="#">Dial</a> Object.
OBJ_METER	Type defined for <a href="#">Meter</a> Object.
OBJ_GRID	Type defined for <a href="#">Grid</a> Object.
OBJ_CHART	Type defined for <a href="#">Chart</a> Object.
OBJ_TEXTENTRY	Type defined for Text-Entry Object.
OBJ_DIGITALMETER	Type defined for <a href="#">DIGITALMETER</a> Object.
OBJ_ANALOGCLOCK	Type defined for <a href="#">ANALOGCLOCK</a> Object.
OBJ_UNKNOWN	Type is undefined and not supported by the library.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) >  
[GOL\\_OBJ\\_TYPE Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## OBJ\_HEADER Structure

```
C
typedef struct {
    WORD ID;
    void * pNxtObj;
    GOL_OBJ_TYPE type;
    WORD state;
    SHORT left;
    SHORT top;
    SHORT right;
    SHORT bottom;
    GOL_SCHEME * pGolScheme;
    DRAW_FUNC DrawObj;
    FREE_FUNC FreeObj;
    MSG_FUNC MsgObj;
    MSG_DEFAULT_FUNC MsgDefaultObj;
} OBJ_HEADER;
```

### Overview

This structure defines the first nine fields of the Objects structure. This allows generic operations on library Objects.

### Members

Members	Description
WORD ID;	Unique id assigned for referencing.
void * pNxtObj;	A pointer to the next object.
GOL_OBJ_TYPE type;	Identifies the type of GOL object.
WORD state;	State of object.

SHORT left;	Left position of the Object.
SHORT top;	Top position of the Object.
SHORT right;	Right position of the Object.
SHORT bottom;	Bottom position of the Object.
GOL_SCHEME * pGolScheme;	Pointer to the scheme used.
DRAW_FUNC DrawObj;	function pointer to the object draw function
FREE_FUNC FreeObj;	function pointer to the object free function
MSG_FUNC MsgObj;	function pointer to the object message function
MSG_DEFAULT_FUNC MsgDefaultObj;	function pointer to the object default message function

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) >  
[OBJ\\_HEADER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DRAW\_FUNC Type

---

C

```
typedef WORD (* DRAW_FUNC)(void *);
```

## Description

---

object draw function pointer typedef

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [DRAW\\_FUNC Type](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## FREE\_FUNC Type

---

C

```
typedef void (* FREE_FUNC)(void *);
```

### Description

---

object free function pointer typedef

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [FREE\\_FUNC Type](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MSG\_DEFAULT\_FUNC Type

C

```
typedef void (* MSG_DEFAULT_FUNC)(WORD, void *, GOL_I
```

## Description

object default message function pointer typedef

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) >  
[MSG\\_DEFAULT\\_FUNC Type](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MSG\_FUNC Type

---

C

```
typedef WORD (* MSG_FUNC)(void *, GOL_MSG *);
```

## Description

---

object message function pointer typedef

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [MSG\\_FUNC Type](#)

---

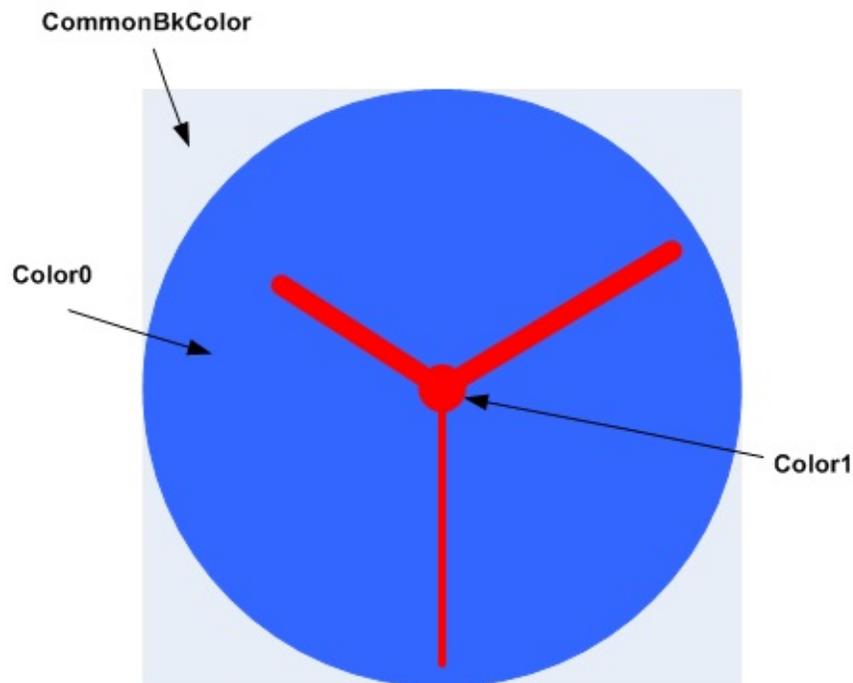
Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Analog Clock

[Functions](#) | [Structures](#) | [Topics](#)

The Analog Clock object is rendered using the assigned style scheme. The following figure illustrates the color assignments for the clock.



## Functions

	Name	Description
◆	<a href="#">AcCreate</a>	This function creates an Analog Clock 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.
◆	<a href="#">AcDraw</a>	This function renders the object on the

		screen using the current parameter settings. 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.
◆	<a href="#"><u>AcSetHour</u></a>	Sets the hour value of the analog clock.
◆	<a href="#"><u>AcSetMinute</u></a>	Sets the minute value of the analog clock.
◆	<a href="#"><u>AcSetSecond</u></a>	Sets the second value of the analog clock.

## Structures

Name	Description
<a href="#"><u>ANALOGCLOCK</u></a>	<p>Defines the parameters required for a clock Object. The following relationships of the parameters determines the general shape of the clock:</p> <ol style="list-style-type: none"> <li>1. centerx and centery determine the middle of the clock.</li> <li>2. radius defines the radius of the clock.</li> <li>4. *pBitmap points to the background image for the analog clock.</li> </ol>

## Topics

Name	Description
<a href="#"><u>Analog Clock States</u></a>	List of Analog Clock bit states.

## Links

[Functions](#), [GOL Objects](#), [Legend](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Analog Clock States

[Macros](#) | [Analog Clock](#)

List of Analog Clock bit states.

## Macros

Name	Description
<a href="#">AC_DRAW</a>	Bit to indicate button must be redrawn.
<a href="#">AC_DISABLED</a>	Bit for disabled state.
<a href="#">AC_HIDE</a>	Bit to indicate button must be removed from screen.
<a href="#">AC_PRESSED</a>	Bit for press state.
<a href="#">AC_TICK</a>	Bit to tick second hand
<a href="#">UPDATE_HOUR</a>	Bit to indicate hour hand must be redrawn
<a href="#">UPDATE_MINUTE</a>	Bit to indicate minute hand must be redrawn
<a href="#">UPDATE_SECOND</a>	Bit to indicate minute hand must be redrawn

## Module

[Analog Clock](#)

## Links

[Macros, Analog Clock](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

## > [Analog Clock States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## AC\_DRAW Macro

---

```
C  
#define AC_DRAW 0x4000 // Bit to indicate button m
```

### Description

---

Bit to indicate button must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [AC\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# AC\_DISABLED Macro

```
C  
#define AC_DISABLED 0x0002 // Bit for disabled sta
```

## Description

Bit for disabled state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [AC\\_DISABLED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## AC\_HIDE Macro

---

```
C
#define AC_HIDE 0x8000 // Bit to indicate button m
```

### Description

---

Bit to indicate button must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [AC\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## AC\_PRESSED Macro

---

```
C
#define AC_PRESSED 0x0004 // Bit for press state.
```

### Description

---

Bit for press state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [AC\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## AC\_TICK Macro

---

C

```
#define AC_TICK 0x1000 // Bit to tick second hand
```

### Description

---

Bit to tick second hand

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [AC\\_TICK Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## UPDATE\_HOUR Macro

---

C

```
#define UPDATE_HOUR 0x2000 // Bit to indicate hour
```

### Description

---

Bit to indicate hour hand must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [UPDATE\\_HOUR Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## UPDATE\_MINUTE Macro

---

C

```
#define UPDATE_MINUTE 0x0100 // Bit to indicate mi
```

### Description

---

Bit to indicate minute hand must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [UPDATE\\_MINUTE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## UPDATE\_SECOND Macro

---

C

```
#define UPDATE_SECOND 0x0200 // Bit to indicate mi
```

### Description

---

Bit to indicate minute hand must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#) > [UPDATE\\_SECOND Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# AcCreate Function

## Analog Clock

C

```
ANALOGCLOCK * AcCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    SHORT hour,  
    SHORT minute,  
    SHORT radius,  
    BOOL sechand,  
    WORD state,  
    void * pBitmap,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates an Analog Clock 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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the object.
SHORT top	Top most position of the object.

SHORT right	Right most position of the object.
SHORT bottom	Bottom most position of the object.
SHORT hour	Initial hour value.
SHORT minute	Initial minute value.
SHORT radius	Sets the radius of the clock.
BOOL sechand	Flag to indicate if the second hand will be drawn or not.
WORD state	Sets the initial state of the object.
void * pBitmap	Pointer to the bitmap used on the face of the analog clock dimension of the image must match the dimension of the widget.
GOL_SCHEME * pScheme	Pointer to the style scheme used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

```
GOL_SCHEME *pScheme;
```

[Copy Code](#)

```
WORD state;  
  
    pScheme = GOLCreateScheme();  
    state = AC_DRAW;  
  
    AnalogClock = AcCreate(ANALOGCLOCK_ID, 20, 64,
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [AcCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# AcDraw Function

## Analog Clock

```
C  
WORD AcDraw(  
    void * pAc  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
void * pAc	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
void MyGOLDDraw(){
    static OBJ_HEADER *pCurrentObj = NULL;
    int done;

    // There are no objects
    if(GOLGetList() == NULL)
        return;

    // If it's last object jump to head
    if(pCurrentObj == NULL)
        pCurrentObj = GOLGetList();

    done = 0;

    // this only process Button and Window
    while(pCurrentObj != NULL){
        // check if object state indicates redrawin
        if(pCurrentObj->state&0xFC00) {
            switch(pCurrentObj->type){
                case OBJ_ANALOGCLOCK:
                    done = AcDraw((ANALOGCLOCK*)pCu
                    break;
                case OBJ_WINDOW:
                    done = WndDraw((WINDOW*)pCurren
                    break;
                default:
                    done = 1;
                    break;
            }
        }
        if(done){
```

```
        // reset only the state if drawing
        pCurrentObj->state = 0;
    }else{
        // done processing the list
        return;
    }
}
// go to next object
pCurrentObj = pCurrentObj->pNxtObj;
}
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [AcDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# AcSetHour Function

## Analog Clock

```
C
void AcSetHour(
    ANALOGCLOCK * pAc,
    SHORT hour
);
```

## Overview

Sets the hour value of the analog clock.

## Input Parameters

Input Parameters	Description
ANALOGCLOCK * pAc	The pointer to the object whose hands will be modified.
SHORT hour	New hour time.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

## > [AcSetHour Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# AcSetMinute Function

## Analog Clock

```
C
void AcSetMinute(
    ANALOGCLOCK * pAc,
    SHORT minute
);
```

## Overview

Sets the minute value of the analog clock.

## Input Parameters

Input Parameters	Description
ANALOGCLOCK * pAc	The pointer to the object whose hands will be modified.
SHORT minute	New minute time.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

## > [AcSetMinute Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# AcSetSecond Function

## Analog Clock

```
C  
void AcSetSecond(  
    ANALOGCLOCK * pAc,  
    SHORT second  
);
```

## Overview

Sets the second value of the analog clock.

## Input Parameters

Input Parameters	Description
ANALOGCLOCK * pAc	The pointer to the object whose hands will be modified.
SHORT second	New second time.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

## > [AcSetSecond Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ANALOGCLOCK Structure

## Analog Clock

```
C
typedef struct {
    OBJ_HEADER hdr;
    SHORT radius;
    SHORT centerx;
    SHORT centery;
    SHORT valueS;
    SHORT prev_valueS;
    SHORT valueM;
    SHORT prev_valueM;
    SHORT valueH;
    SHORT prev_valueH;
    void * pBitmap;
} ANALOGCLOCK;
```

## Overview

Defines the parameters required for a clock Object. The following relationships of the parameters determines the general shape of the clock:

1. centerx and centery determine the middle of the clock.
2. radius defines the radius of the clock.
4. \*pBitmap points to the background image for the analog clock.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).

SHORT radius;	Radius of the clock.
SHORT centerx;	center location in X-axis.
SHORT centery;	center location in Y-axis.
SHORT valueS;	time in Second
SHORT prev_valueS;	previous time in Second
SHORT valueM;	time in Minute
SHORT prev_valueM;	previous time in Minute
SHORT valueH;	time in Hour
SHORT prev_valueH;	previous time in Hour
void * pBitmap;	Pointer to bitmap used.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [ANALOGCLOCK Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

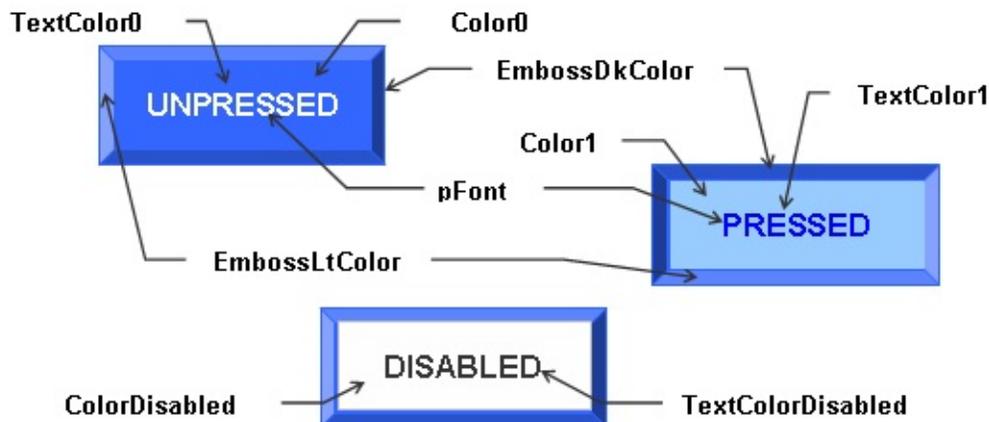
# Button

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Button supports Keyboard and Touchscreen inputs, replying to their events with the following messages:

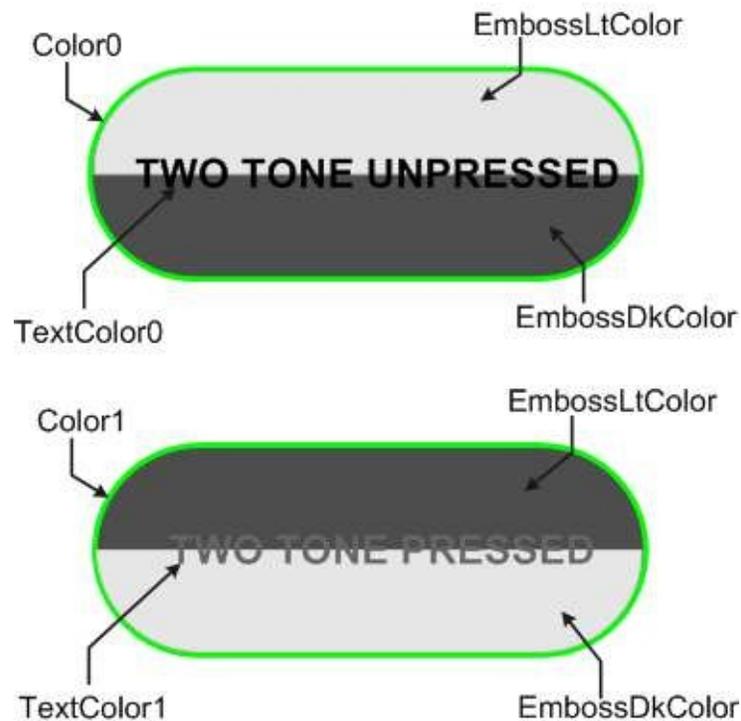
1. `BTN_MSG_PRESSED`
2. `BTN_MSG_RELEASED`

The button object is rendered using the assigned style scheme. The following figure illustrates the color assignments for the button.



`CommonBkColor` – used to hide the button on the screen.

A variant of the button widget, is to implement it as a two tone button. The button object is rendered using the modified color assignments in the style scheme. The two tone button is rendered when the `BTN_TWOTONE` object state bit is set.



### Two Tone Button

When using images on the button widget, the image is drawn on the center of the widget. If the image size is larger than the widget dimension, the image will cover the widget. When this happens, it is recommended to disable the rendering of the button panel so no time is spent rendering the panel which will be covered by the image. To do this enable the [BTN\\_NOPANEL](#) object state bit.

The text or string used with the button widget is always rendered last. Even with images, the string will always be rendered on top of the image. Text rendering on the button widget is not affected by [BTN\\_NOPANEL](#) object state bit. Text will only be affected by the following object state bits [BTN\\_TEXTRIGHT](#), [BTN\\_TEXTLEFT](#), [BTN\\_TEXTBOTTOM](#), and [BTN\\_TEXTTOP](#).

To enable multiple lines of text on the button widget, enable the [USE\\_BUTTON\\_MULTI\\_LINE](#) in the GraphicsConfig.h. Enabling the [USE\\_BUTTON](#) will only enable single lines of text.

## Functions

	Name	Description
◆	<a href="#">BtnCreate</a>	This function creates a <a href="#">BUTTON</a> 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.
◆	<a href="#">BtnDraw</a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>The text on the face of the button is drawn on top of the bitmap. Text is always rendered centered on the face of the button.</p> <p>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... <a href="#">more</a></p>
◆	<a href="#">BtnSetText</a>	This function sets the string used for the object.
◆	<a href="#">BtnMsgDefault</a>	This function performs the actual state change based on the translated message given. The following state changes are

		supported:
	<a href="#"><u>BtnTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

## Macros

Name	Description
<a href="#"><u>BtnGetText</u></a>	This macro returns the address of the current text string used for the object.
<a href="#"><u>BtnGetBitmap</u></a>	This macro returns the location of the currently used bitmap for the object.
<a href="#"><u>BtnSetBitmap</u></a>	This macro sets the bitmap used in the object. The size of the bitmap must match the face of the button.

## Structures

Name	Description
<a href="#"><u>BUTTON</u></a>	<p>Defines the parameters required for a button Object. The following relationships of the parameters determines the general shape of the button:</p> <ol style="list-style-type: none"> <li>1. Width is determined by right - left.</li> <li>2. Height is determined by top - bottom.</li> <li>3. Radius - specifies if the button will have a rounded edge. If zero then the button will have sharp (cornered) edge.</li> <li>4. If <math>2 * \text{radius} = \text{height} = \text{width}</math>, the button</li> </ol>

is a circular button.

## Topics

---

Name	Description
<a href="#">Button States</a>	List of Button bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Button States

Macros | [Button](#)

List of [Button](#) bit states.

## Macros

Name	Description
<a href="#">BTN_DISABLED</a>	Bit for disabled state.
<a href="#">BTN_DRAW</a>	Bit to indicate button must be redrawn.
<a href="#">BTN_DRAW_FOCUS</a>	Bit to indicate focus must be redrawn.
<a href="#">BTN_FOCUSED</a>	Bit for focus state.
<a href="#">BTN_HIDE</a>	Bit to indicate button must be removed from screen.
<a href="#">BTN_PRESSED</a>	Bit for press state.
<a href="#">BTN_TEXTBOTTOM</a>	Bit to indicate text is top aligned.
<a href="#">BTN_TEXTLEFT</a>	Bit to indicate text is left aligned.
<a href="#">BTN_TEXTRIGHT</a>	Bit to indicate text is right aligned.
<a href="#">BTN_TEXTTOP</a>	Bit to indicate text is bottom aligned.
<a href="#">BTN_TOGGLE</a>	Bit to indicate button will have a toggle behavior.
<a href="#">BTN_TWOTONE</a>	Bit to indicate the button is a two tone type.
<a href="#">BTN_NOPANEL</a>	Bit to indicate the button will be drawn without a panel (for faster drawing when the

button image used is larger than the button panel).

## Module

---

[Button](#)

## Links

---

[Macros](#), [Button](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_DISABLED Macro

---

C

```
#define BTN_DISABLED 0x0002 // Bit for disabled sta
```

### Description

---

Bit for disabled state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_DRAW Macro

---

C

```
#define BTN_DRAW 0x4000 // Bit to indicate button m
```

### Description

---

Bit to indicate button must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_DRAW\_FOCUS Macro

---

C

```
#define BTN_DRAW_FOCUS 0x2000 // Bit to indicate fo
```

### Description

---

Bit to indicate focus must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_DRAW\\_FOCUS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_FOCUSED Macro

---

C

```
#define BTN_FOCUSED 0x0001 // Bit for focus state.
```

### Description

---

Bit for focus state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_HIDE Macro

---

C

```
#define BTN_HIDE 0x8000 // Bit to indicate button m
```

### Description

---

Bit to indicate button must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_PRESSED Macro

---

C

```
#define BTN_PRESSED 0x0004 // Bit for press state.
```

### Description

---

Bit for press state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_TEXTBOTTOM Macro

---

C

```
#define BTN_TEXTBOTTOM 0x0040 // Bit to indicate te
```

### Description

---

Bit to indicate text is top aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_TEXTBOTTOM Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_TEXTLEFT Macro

---

C

```
#define BTN_TEXTLEFT 0x0020 // Bit to indicate text
```

### Description

---

Bit to indicate text is left aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_TEXTLEFT Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_TEXTRIGHT Macro

---

C

```
#define BTN_TEXTRIGHT 0x0010 // Bit to indicate text
```

### Description

---

Bit to indicate text is right aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_TEXTRIGHT Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_TEXTTOP Macro

---

C

```
#define BTN_TEXTTOP 0x0080 // Bit to indicate text .
```

### Description

---

Bit to indicate text is bottom aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_TEXTTOP Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_TOGGLE Macro

---

C

```
#define BTN_TOGGLE 0x0008 // Bit to indicate button
```

### Description

---

Bit to indicate button will have a toggle behavior.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_TOGGLE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_TWOTONE Macro

---

C

```
#define BTN_TWOTONE 0x0100 // Bit to indicate the b
```

### Description

---

Bit to indicate the button is a two tone type.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_TWOTONE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## BTN\_NOPANEL Macro

```
C
```

```
#define BTN_NOPANEL 0x0200 // Bit to indicate the b
```

### Description

Bit to indicate the button will be drawn without a panel (for faster drawing when the button image used is larger than the button panel).

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [Button States](#) > [BTN\\_NOPANEL Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnCreate Function

## Button

```
C
BUTTON * BtnCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    SHORT radius,
    WORD state,
    void * pBitmap,
    XCHAR * pText,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a **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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the object.
SHORT top	Top most position of the object.
SHORT right	Right most position of the object.

SHORT bottom	Bottom most position of the object.
SHORT radius	Radius of the rounded edge.
WORD state	Sets the initial state of the object.
void * pBitmap	Pointer to the bitmap used on the face of the button dimension of the bitmap must match the dimension of the button.
XCHAR * pText	Pointer to the text of the button.
GOL_SCHEME * pScheme	Pointer to the style scheme used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

```
GOL_SCHEME *pScheme;  
BUTTON *buttons[3];  
WORD state;  
  
    pScheme = GOLCreateScheme();  
    state = BTN_DRAW;
```

[Copy Code](#)

```
buttons[0] = BtnCreate(1,20,64,50,118,0, state,  
// check if button 0 is created  
if (buttons[0] == NULL)  
    return 0;  
  
buttons[1] = BtnCreate(2,52,64,82,118,0, state,  
// check if button 1 is created  
if (buttons[1] == NULL)  
    return 0;  
  
buttons[2] = BtnCreate(3,84,64,114,118,0, state  
// check if button 2 is created  
if (buttons[2] == NULL)  
    return 0;  
  
return 1;
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) >  
[BtnCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnDraw Function

## Button

```
C  
WORD BtnDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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 button is drawn on top of the bitmap. Text is always rendered centered on the face of the button.

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.

## Input Parameters

Input Parameters	Description
pB	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
void MyGOLDraw(){
    static OBJ_HEADER *pCurrentObj = NULL;
    int done;

    // There are no objects
    if(GOLGetList() == NULL)
        return;

    // If it's last object jump to head
    if(pCurrentObj == NULL)
        pCurrentObj = GOLGetList();

    done = 0;

    // this only process Button and Window
    while(pCurrentObj != NULL){
        // check if object state indicates redrawin
        done = pCurrentObj->draw(pCurrentObj);

        if(done){
            // reset only the state if drawing
            pCurrentObj->state = 0;
        }else{
```

```
        // done processing the list
        return;
    }
}
// go to next object
pCurrentObj = pCurrentObj->pNxtObj;
}
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [BtnDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnGetText Macro

## Button

C

```
#define BtnGetText(pB) ((BUTTON *)pB)->pText
```

## Overview

This macro returns the address of the current text string used for the object.

## Input Parameters

Input Parameters	Description
pB	Pointer to the object.

## Returns

Returns pointer to the text string being used.

## Preconditions

none

## Side Effects

none

## Example

```
XCHAR *pChar;  
BUTTON Button[2];
```

[Copy Code](#)

```
|| pChar = BtnGetText(Button[0]);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) >  
[BtnGetText Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnSetText Function

## Button

```
C
void BtnSetText(
    BUTTON * pB,
    XCHAR * pText
);
```

## Overview

This function sets the string used for the object.

## Input Parameters

Input Parameters	Description
BUTTON * pB	The pointer to the object whose text will be modified.
XCHAR * pText	Pointer to the text that will be used.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
XCHAR Label0[] = "ON";  
XCHAR Label1[] = "OFF";  
BUTTON Button[2];  
  
    BtnSetText(Button[0], Label0);  
    BtnSetText(Button[1], Label1);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [BtnSetText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnGetBitmap Macro

## Button

C

```
#define BtnGetBitmap(pB) ((BUTTON *)pB)->pBitmap
```

## Overview

This macro returns the location of the currently used bitmap for the object.

## Input Parameters

Input Parameters	Description
pB	Pointer to the object.

## Returns

Returns the pointer to the current bitmap used.

## Preconditions

none

## Side Effects

none

## Example

```
BUTTON *pButton;  
BITMAP_FLASH *pUsedBitmap;
```

[Copy Code](#)

```
pUsedbitmap = BtnGetBitmap(pButton);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) >  
[BtnGetBitmap Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnSetBitmap Macro

## Button

C

```
#define BtnSetBitmap(pB, pBtmap) ((BUTTON *)pB)->pBi
```

## Overview

This macro sets the bitmap used in the object. The size of the bitmap must match the face of the button.

## Input Parameters

Input Parameters	Description
pB	Pointer to the object.
pBitmap	Pointer to the bitmap to be used.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

```
extern BITMAP_FLASH myIcon;
```

[Copy Code](#)

```
BUTTON *pButton;
```

```
    BtnSetBitmap(pButton , &myIcon);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) >  
[BtnSetBitmap Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BtnMsgDefault Function

## Button

C

```
void BtnMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
BTN_MSG_PRESSED	Touch Screen, Keyboard	Set <a href="#">BTN_PRESSED</a>	<a href="#">Button</a> be re in presse state.
BTN_MSG_RELEASED	Touch Screen, Keyboard	Clear <a href="#">BTN_PRESSED</a>	<a href="#">Button</a> be re in unprese state.
BTN_MSG_CANCELPRESS	Touch Screen,	Clear <a href="#">BTN_PRESSED</a>	<a href="#">Button</a> be re in unprese state.

## Input Parameters

---

Input Parameters	Description
WORD translatedMsg	The translated message.
void * pObj	The pointer to the object whose state will be modified.
GOL_MSG * pMsg	The pointer to the GOL message.

## Returns

---

none

## Preconditions

---

none

## Side Effects

---

none

## Example

---

See [BtnTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [BtnMsgDefault Function](#)

---

# BtnTranslateMsg Function

## Button

```
C  
WORD BtnTranslateMsg(  
    void * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit
BTN_MSG_PRESSED	Touch Screen	EVENT_PRESS, EVENT_MOVE
	Keyboard	EVENT_KEYSCAN

BTN_MSG_STILLPRESSED	Touch Screen	EVENT_STILLPRESS
BTN_MSG_RELEASED	Touch Screen	EVENT_RELEASE
	Keyboard	EVENT_KEYSCAN
BTN_MSG_CANCELPRESS	Touch Screen	EVENT_MOVE
OBJ_MSG_INVALID	Any	Any

## Input Parameters

Input Parameters	Description
void * pObj	The pointer to the object where the message will be evaluated to check if the message will affect the object.
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.

## Returns

---

Returns the translated message depending on the received GOL message:

- BTN\_MSG\_PRESSED – [Button](#) is pressed
- BTN\_MSG\_RELEASED – [Button](#) is released
- BTN\_MSG\_CANCELPRESS – [Button](#) will be released, user cancels press action on the button
- OBJ\_MSG\_INVALID – [Button](#) is not affected

## Preconditions

---

none

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
void MyGOLMsg(GOL_MSG *pMsg){  
  
    OBJ_HEADER *pCurrentObj;  
    WORD objMsg;  
  
    if(pMsg->uiEvent == EVENT_INVALID)  
        return;  
    pCurrentObj = GOLGetList();  
  
    while(pCurrentObj != NULL){  
        // If the object must be redrawn  
        // It cannot accept message  
        if(!IsObjUpdated(pCurrentObj)){  
            translatedMsg = pCurrentObj->MsgObj(pCu
```

```
        if(translatedMsg != OBJ_MSG_INVALID)
        {
            if(GOLMsgCallback(translatedMsg, p
                if(pCurrentObj->MsgDefaultObj)
                    pCurrentObj->MsgDefaultObj
            }
        }
    }
    pCurrentObj = pCurrentObj->pNxtObj;
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > [BtnTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BUTTON Structure

## Button

C

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT radius;
    SHORT textWidth;
    SHORT textHeight;
    XCHAR * pText;
    void * pBitmap;
    GFX_COLOR previousAlphaColor;
} BUTTON;
```

## Overview

Defines the parameters required for a button Object. The following relationships of the parameters determines the general shape of the button:

1. Width is determined by right - left.
2. Height is determined by top - bottom.
3. Radius - specifies if the button will have a rounded edge. If zero then the button will have sharp (cornered) edge.
4. If  $2 * \text{radius} = \text{height} = \text{width}$ , the button is a circular button.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT radius;	Radius for rounded buttons.
SHORT textWidth;	Computed text width, done at creation.

SHORT textHeight;	Computed text height, done at creation.
XCHAR * pText;	Pointer to the text used.
void * pBitmap;	Pointer to bitmap used.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > **[BUTTON Structure](#)**

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

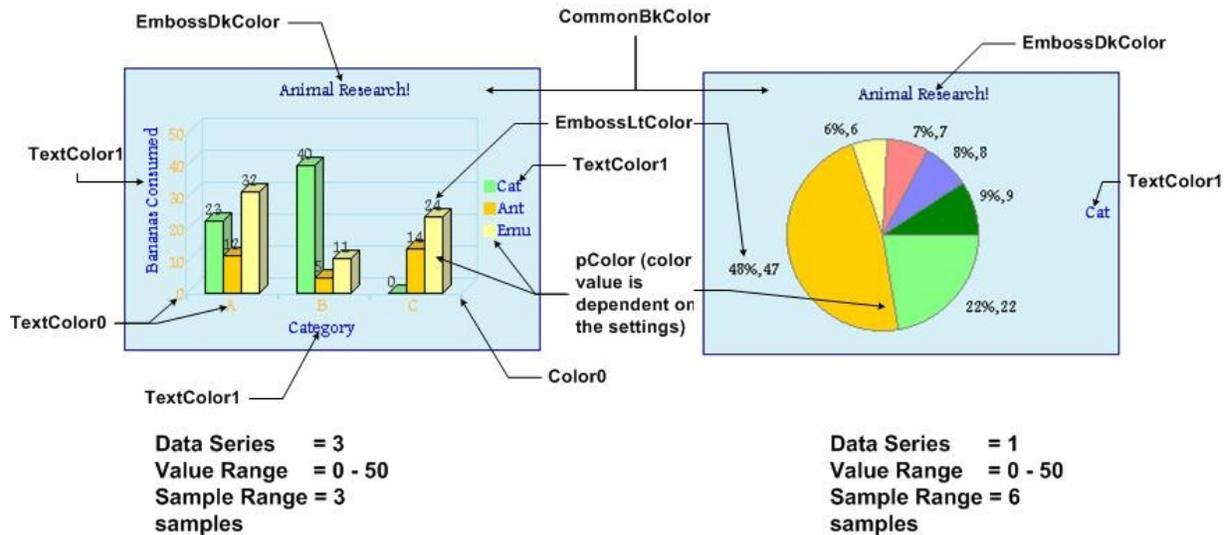
[Contents](#) | [Index](#) | [Home](#)

# Chart

Functions | Macros | Structures | Topics

It supports only Keyboard inputs, replying to any touch screen events with the message: CH\_MSG\_SELECTED.

The Chart Object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



## Chart Terminologies

1. Value Axis - This is the vertical range of a chart for normal bar charts and horizontal range of the chart for horizontally drawn bar charts. In most cases this axis will represent values (\$ amounts), temperatures, or other numeric data.
2. Sample Axis - This is the horizontal range of a chart for normal bar

charts and vertical range of the chart for horizontally drawn bar charts. In most cases this axis will represent categories, such as months, sample segments, or other non-numeric data.

3. Title - The text used to define the Title of the chart.
4. Data Points (or the sample points) These are the individual points where a value is graphed, as a point on a line, a bar, or a pie slice.
5. Data Series - A complete series of data, distinguished by the same color and type of sample point.
6. Legend - Labels that indicate how each data series is displayed on the chart. Each color represents a different data series. For pie charts with only one data series shown, each color represents one sector or one sample point.
7. Data Sample Range - The scale for the data sample axis.  
Example: months from January to December. Internally, this range is represented by:
  - Numeric Sequence 1, 2, 3, ... and so on
  - Alphabet Sequence A, B, C, .. and so on.
8. Value Range - The scale for the value axis. Example: range of numbers from the lowest to the highest to be charted.

## Functions

	Name	Description
◆	<u><a href="#">ChCreate</a></u>	This function creates a <u><a href="#">CHART</a></u> 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.
◆	<u><a href="#">ChDraw</a></u>	This function renders the object on the screen using the current parameter settings. 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

		<p>determined by the style scheme set. The colors of the bars of the bar chart or sectors of the pie chart can be the default color table or user defined color table set by <a href="#">ChSetColorTable()</a> function.</p> <p>When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is... <a href="#">more</a></p>
◆	<a href="#">ChAddDataSeries</a>	This function creates a <a href="#">DATASERIES</a> object and populates the structure with the given parameters.
◆	<a href="#">ChRemoveDataSeries</a>	This function removes <a href="#">DATASERIES</a> object from the list of <a href="#">DATASERIES</a> objects and frees the memory used of that removed object. The position of the object to be removed is specified by the number parameter. If the list has only one member, it removes the member regardless of the number given.
◆	<a href="#">ChSetValueRange</a>	This function sets the minimum and maximum range of values that the bar chart will show. The criteria is that $min \leq max$ .
◆	<a href="#">ChSetPercentRange</a>	This function sets the minimum and maximum range of percentage that the bar chart will show. The criteria is that $min \leq max$ . This affects bar charts only and <code>CH_PERCENTAGE</code> bit state is set.

	<a href="#"><u>ChSetSampleRange</u></a>	<p>This function sets the sample start and sample end when drawing the chart. Together with the data series' <a href="#"><u>SHOW_DATA</u></a> flags the different way of displaying the chart data is achieved.</p>
	<a href="#"><u>ChFreeDataSeries</u></a>	<p>This function removes <a href="#"><u>DATASERIES</u></a> object from the list of <a href="#"><u>DATASERIES</u></a> objects and frees the memory used of that removed object.</p>
	<a href="#"><u>ChTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

## Macros

Name	Description
<a href="#"><u>ChShowSeries</u></a>	<p>This macro sets the specified data series number show flag to be set to <a href="#"><u>SHOW_DATA</u></a>.</p>
<a href="#"><u>ChHideSeries</u></a>	<p>This macro sets the specified data series number show flag to be set to <a href="#"><u>HIDE_DATA</u></a>.</p>
<a href="#"><u>ChGetShowSeriesCount</u></a>	<p>This macro shows the number of data series that has its show flag set to <a href="#"><u>SHOW_DATA</u></a></p>

<a href="#"><u>ChGetShowSeriesStatus</u></a>	This macro returns the show ID status of the <a href="#"><u>DATASERIES</u></a> .
<a href="#"><u>ChSetValueLabel</u></a>	This macro sets the address of the current text string used for the value axis label of the bar chart.
<a href="#"><u>ChGetValueLabel</u></a>	This macro returns the address of the current text string used for the value axis label of the bar chart.
<a href="#"><u>ChGetValueMax</u></a>	This macro returns the current maximum value that will be drawn for bar charts.
<a href="#"><u>ChGetValueMin</u></a>	This macro returns the current minimum value that will be drawn for bar charts.
<a href="#"><u>ChGetValueRange</u></a>	This macro gets the current range for bar charts. The value returned is calculated from the current (valMax - valMin) set. To get the minimum use <a href="#"><u>ChGetValueMin()</u></a> and to get the maximum use <a href="#"><u>ChGetValueMax()</u></a> .
<a href="#"><u>ChSetSampleLabel</u></a>	This macro sets the address of the current text string used for the sample axis label of the bar chart.
<a href="#"><u>ChGetSampleLabel</u></a>	This macro returns the address of the current text string used for the sample axis label of the bar chart.
<a href="#"><u>ChGetSampleStart</u></a>	This macro returns the sampling start value.
<a href="#"><u>ChGetSampleEnd</u></a>	This macro returns the sampling end value.

<a href="#"><u>ChGetPercentRange</u></a>	This macro gets the percentage range for bar charts. The value returned is calculated from percentage max - min. To get the minimum use <a href="#"><u>ChGetPercentMin()</u></a> and to get the maximum use <a href="#"><u>ChGetPercentMax()</u></a> .
<a href="#"><u>ChGetSampleRange</u></a>	This macro gets the sample range for pie or bar charts. The value returned is calculated from smplEnd - smplStart.
<a href="#"><u>ChGetPercentMax</u></a>	This macro returns the current maximum value of the percentage range that will be drawn for bar charts when CH_PERCENTAGE bit state is set.
<a href="#"><u>ChGetPercentMin</u></a>	This macro returns the current minimum value of the percentage range that will be drawn for bar charts when CH_PERCENTAGE bit state is set.
<a href="#"><u>ChSetColorTable</u></a>	This macro sets the color table used to draw the data in pie and bar charts.
<a href="#"><u>ChGetColorTable</u></a>	This macro returns the current color table used for the pie and bar charts.
<a href="#"><u>ChSetTitle</u></a>	This macro sets the address of the current text string used for the title of the chart.
<a href="#"><u>ChGetTitle</u></a>	This macro returns the address of the current text string used for the title of the chart.
<a href="#"><u>ChSetTitleFont</u></a>	This macro sets the location of the font used for the title of the chart.

<a href="#"><u>ChGetTitleFont</u></a>	This macro returns the location of the font used for the title of the chart.
<a href="#"><u>ChGetAxisLabelFont</u></a>	This macro returns the location of the font used for the X and Y axis labels of the chart.
<a href="#"><u>ChSetAxisLabelFont</u></a>	This macro sets the location of the font used for the X and Y axis labels of the chart.
<a href="#"><u>ChGetGridLabelFont</u></a>	This macro returns the location of the font used for the X and Y axis grid labels of the chart.
<a href="#"><u>ChSetGridLabelFont</u></a>	This macro sets the location of the font used for the X and Y axis grid labels of the chart.

## Structures

Name	Description
<a href="#"><u>CHART</u></a>	Defines the parameters required for a chart Object.
<a href="#"><u>DATASERIES</u></a>	Defines a variable for the <a href="#"><u>CHART</u></a> object. It specifies the number of samples, pointer to the array of samples for the data series and pointer to the next data series. A member of this structure (show) is used as a flag to determine if the series is to be drawn or not. Together with the smplStart and smplEnd it will determine what kind of chart will be drawn.
<a href="#"><u>CHARTPARAM</u></a>	Defines the parameters for the <a href="#"><u>CHART</u></a>

object.

## Topics

---

Name	Description
<a href="#">Chart States</a>	List of Chart bit states.
<a href="#">Data Series Status Settings</a>	Data Series show status flag settings.
<a href="#">Chart Examples</a>	Examples of generated bar charts based on settings.
<a href="#">Color Table</a>	Default color table used to draw data points in a chart.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Chart States

Macros | [Chart](#)

List of [Chart](#) bit states.

## Macros

Name	Description
<a href="#">CH_DISABLED</a>	Bit for disabled state.
<a href="#">CH_DRAW</a>	Bit to indicate chart must be redrawn.
<a href="#">CH_DRAW_DATA</a>	Bit to indicate data portion of the chart must be redrawn.
<a href="#">CH_3D_ENABLE</a>	Bit to indicate that bar charts are to be drawn with 3-D effect
<a href="#">CH_BAR</a>	Bit to indicate the chart is type bar. If both PIE and BAR types are set BAR type has higher priority.
<a href="#">CH_BAR_HOR</a>	These bits (with <a href="#">CH_BAR</a> bit set), sets the bar chart to be drawn horizontally.
<a href="#">CH_DONUT</a>	These bits (with <a href="#">CH_PIE</a> bit set), sets the pie chart to be drawn in a donut shape.
<a href="#">CH_LEGEND</a>	Bit to indicate that legend is to be shown. Usable only when seriesCount > 1.
<a href="#">CH_NUMERIC</a>	This bit is used only for bar charts. If this bit is set, it indicates that the bar chart labels for variables are numeric. If this bit is not set, it indicates that the bar chart labels for variables are alphabets.

<a href="#"><u>CH_PERCENT</u></a>	Bit to indicate that the pie chart will be drawn with percentage values shown for the sample data. For bar chart, if <a href="#"><u>CH_VALUE</u></a> is set, it toggles the value shown to percentage.
<a href="#"><u>CH_PIE</u></a>	Bit to indicate the chart is type pie. If both PIE and BAR types are set BAR type has higher priority.
<a href="#"><u>CH_VALUE</u></a>	Bit to indicate that the values of the bar chart data or pie chart data are to be shown
<a href="#"><u>CH_HIDE</u></a>	Bit to indicate chart must be removed from screen.

## Module

---

### [Chart](#)

## Links

---

### [Macros, Chart](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_DISABLED Macro

```
C
#define CH_DISABLED 0x0002 // Bit for disabled state
```

### Description

Bit for disabled state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_DISABLED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_DRAW Macro

---

```
C
#define CH_DRAW 0x4000 // Bit to indicate chart
```

### Description

---

Bit to indicate chart must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_DRAW\_DATA Macro

---

```
C
#define CH_DRAW_DATA 0x2000 // Bit to indicate (
```

### Description

---

Bit to indicate data portion of the chart must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_DRAW\\_DATA Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_3D\_ENABLE Macro

```
C
#define CH_3D_ENABLE 0x0008 // Bit to indicate
```

### Description

Bit to indicate that bar charts are to be drawn with 3-D effect

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_3D\\_ENABLE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_BAR Macro

```
C
#define CH_BAR 0x0200 // Bit to indicate the cha
```

### Description

Bit to indicate the chart is type bar. If both PIE and BAR types are set BAR type has higher priority.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_BAR Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_BAR\_HOR Macro

```
C
#define CH_BAR_HOR 0x0240 // These bits (with CH_BAR)
```

### Description

These bits (with [CH\\_BAR](#) bit set), sets the bar chart to be drawn horizontally.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_BAR\\_HOR Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_DONUT Macro

```
C
#define CH_DONUT 0x0140 // These bits (with CH_I
```

### Description

These bits (with [CH\\_PIE](#) bit set), sets the pie chart to be drawn in a donut shape.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_DONUT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_LEGEND Macro

```
C  
#define CH_LEGEND 0x0001 // Bit to indicate tha
```

### Description

Bit to indicate that legend is to be shown. Usable only when seriesCount > 1.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_LEGEND Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_NUMERIC Macro

```
C
#define CH_NUMERIC 0x0080 // This bit is used o
```

### Description

This bit is used only for bar charts. If this bit is set, it indicates that the bar chart labels for variables are numeric. If this bit is not set, it indicates that the bar chart labels for variables are alphabets.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_NUMERIC Macro](#)

## CH\_PERCENT Macro

```
C
#define CH_PERCENT 0x0010 // Bit to indicate th
```

### Description

Bit to indicate that the pie chart will be drawn with percentage values shown for the sample data. For bar chart, if [CH\\_VALUE](#) is set, it toggles the value shown to percentage.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_PERCENT Macro](#)

## CH\_PIE Macro

```
C
#define CH_PIE 0x0100 // Bit to indicate the cha
```

### Description

Bit to indicate the chart is type pie. If both PIE and BAR types are set BAR type has higher priority.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_PIE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_VALUE Macro

```
C
#define CH_VALUE 0x0004 // Bit to indicate that
```

### Description

Bit to indicate that the values of the bar chart data or pie chart data are to be shown

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_VALUE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_HIDE Macro

---

```
C
#define CH_HIDE 0x8000 // Bit to indicate chart
```

### Description

---

Bit to indicate chart must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#) > [CH\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Data Series Status Settings

[Macros](#) | [Chart](#)

Data Series show status flag settings.

## Macros

Name	Description
<a href="#">HIDE_DATA</a>	Macro used to reset the data series show flag or indicate that the data series will be not be shown when the chart is drawn.
<a href="#">SHOW_DATA</a>	Macro used to set the data series show flag or indicate that the data series will be shown when the chart is drawn.

## Module

[Chart](#)

## Links

[Macros](#), [Chart](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Data Series Status Settings](#)

## HIDE\_DATA Macro

```
C
#define HIDE_DATA 0 // Macro used to r
```

### Description

Macro used to reset the data series show flag or indicate that the data series will be not be shown when the chart is drawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Data Series Status Settings](#) > [HIDE\\_DATA Macro](#)

# SHOW\_DATA Macro

```
C
```

```
#define SHOW_DATA 1 // Macro used to s
```

## Description

Macro used to set the data series show flag or indicate that the data series will be shown when the chart is drawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Data Series Status Settings](#) > [SHOW\\_DATA Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Chart Examples

## Chart

The following are some examples of settings and output chart that will be generated.

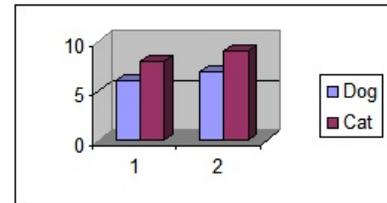
### Example 1

CH\_NUMERIC = 1  
 SERIESDATA Series  
 SERIESDATA Series

	Sample 1	Sample 2	Show
Dog	6	7	x
Cat	8	9	x

*ChSetSampleRange(1,2);*

↑ smplStart    ↑ smplEnd

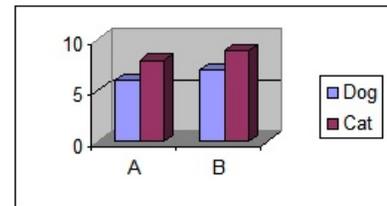


### Example 2

CH\_NUMERIC = 0  
 SERIESDATA Series  
 SERIESDATA Series

	Sample 1	Sample 2	Show
Dog	6	7	x
Cat	8	9	x

↑ smplStart    ↑ smplEnd

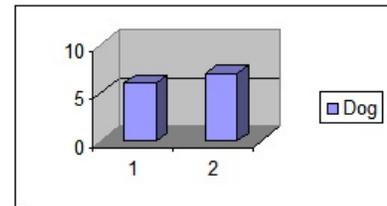


### Example 3

CH\_NUMERIC = 1  
 SERIESDATA Series  
 SERIESDATA Series

	Sample 1	Sample 2	Show
Dog	6	7	x
Cat	8	9	

↑ smplStart    ↑ smplEnd





# ChCreate Function

## Chart

```
C
CHART * ChCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    DATASERIES * pData,
    CHARTPARAM * pParam,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [CHART](#) 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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the object.
SHORT top	Top most position of the object.
SHORT right	Right most position of the object.
SHORT bottom	Bottom most position of the object.

WORD state	Sets the initial state of the object.
DATASERIES * pData	Pointer to the data for the contents of the chart. NULL can be assigned initially when creating the object.
CHARTPARAM * pParam	Pointer to the chart parameters. NULL can be assigned initially when creating the object and the chart parameters can be populated using the API provided.
GOL_SCHEME * pScheme	Pointer to the style scheme used. When set to NULL, the default style scheme will be used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
extern const FONT_FLASH GOLSmallFont;
extern const FONT_FLASH GOLMediumFont;

// Note that strings are declared as such to cover
// where XCHAR type is declared as short (2 bytes).
XCHAR ChTitle[] = {'E', 'x', 'a', 'm', 'p', 'l', 'e',
```

```

XCHAR SampleName[] = {'C', 'a', 't', 'e', 'g', 'o', 'r',
XCHAR ValueName[] = {'#', 'H', 'i', 't', 's', 0};
XCHAR SeriesName[2] = {
                        {'V', '1', 0},
                        {'V', '2', 0},
                        };
V1Data[2] = { 50, 100};
V2Data[2] = { 5, 10};

GOL_SCHEME *pScheme;
CHART *pChart;
CHARTPARAM Contents;
WORD state;

pScheme = GOLCreateScheme();
state = CH_BAR|CH_DRAW|CH_BAR_HOR; // Bar Char

pChart = ChCreate(0x01,
                 0, 0,
                 GetMaxX(),
                 GetMaxY(),
                 state,
                 NULL,
                 NULL,
                 pScheme);

if (pMyChart == NULL)
    return 0;

ChSetTitleFont(pChart, (void*)&GOLMediumFont);
ChSetTitle(pChart, ChTitle);

// set the grid labels and axis labels font
ChSetGridLabelFont(pChart, (void*)&GOLSmallFont
ChSetAxisLabelFont(pChart, (void*)&GOLSmallFont

```

```
// set the labels for the X and Y axis
ChSetSampleLabel(pChart, (XCHAR*)SampleName);
ChSetValueLabel(pChart, (XCHAR*)ValueName);

ChAddDataSeries(pChart, 2, V1Data, (XCHAR*)Seri
ChAddDataSeries(pChart, 2, V2Data, (XCHAR*)Seri

// set the range of the sample values
ChSetValueRange(pChart, 0, 100);

// show all two samples to be displayed (start
ChSetSampleRange(pChart, 1, 2);

GOLDraw();
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChDraw Function

## Chart

```
C  
WORD ChDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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 colors of the bars of the bar chart or sectors of the pie chart can be the default color table or user defined color table set by [ChSetColorTable\(\)](#) function.

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.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and

- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChAddDataSeries Function

## Chart

```
C
DATASERIES * ChAddDataSeries(
    CHART * pCh,
    WORD nSamples,
    WORD * pData,
    XCHAR * pName
);
```

## Overview

This function creates a [DATASERIES](#) object and populates the structure with the given parameters.

## Input Parameters

Input Parameters	Description
CHART * pCh	Pointer to the chart object.
WORD nSamples	The number of samples or data points.
WORD * pData	Pointer to the array of samples or data points.
XCHAR * pName	Pointer to the string used to label the data series.

## Returns

Returns the pointer to the data variable ([DATASERIES](#)) object created. If NULL is returned, the addition of the new object failed due to not enough memory for the object.

## Preconditions

---

none

## Side Effects

---

Appends to the list of [DATASERIES](#) that the chart is operating on. By default, the show flag of the newly added data series is set to [SHOW\\_DATA](#) or enabled.

## Example

---

See [ChCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChAddDataSeries Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChRemoveDataSeries Function

## Chart

```
C
void ChRemoveDataSeries(
    CHART * pCh,
    WORD number
);
```

## Overview

This function removes [DATASERIES](#) object from the list of [DATASERIES](#) objects and frees the memory used of that removed object. The position of the object to be removed is specified by the number parameter. If the list has only one member, it removes the member regardless of the number given.

## Input Parameters

Input Parameters	Description
CHART * pCh	Pointer to the chart object.
WORD number	The position of the object to be removed in the list where the first object in the list is assigned a value of 1. If this parameter is set to zero, the whole list of DATA_SERIES is removed.

## Returns

none.

## Preconditions

none

## Side Effects

---

none.

## Example

---

[Copy Code](#)

```
void ClearChartData(CHART *pCh) {  
    if(pCh->pChartData != NULL)  
        // remove the all data series  
        ChRemoveDataSeries(pCh, 0);  
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChRemoveDataSeries Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChShowSeries Macro

## Chart

```
C
```

```
#define ChShowSeries(pCh, seriesNum) (ChSetDataSerie:
```

## Overview

This macro sets the specified data series number show flag to be set to [SHOW\\_DATA](#).

## Input Parameters

Input Parameters	Description
pCh	Pointer to the chart object.
seriesNum	The data series number that will be modified. If this number is zero, all the entries' flag in the list will be set to <a href="#">SHOW_DATA</a> .

## Returns

Returns the same passed number if successful otherwise -1 if unsuccessful.

## Preconditions

none

## Side Effects

none

## Example

---

[Copy Code](#)

```
// from the example in ChCreate() we change the it  
// GOLDDraw() is called.  
  
// reset all data series to be HIDE_DATA  
ChHideSeries(pMyChart, 0);  
// set data series 1 (V1Data) to be shown  
ChShowSeries(pMyChart, 1);  
// draw the chart  
GOLDDraw();  
.....
```

\*

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChShowSeries Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChHideSeries Macro

## Chart

```
C
```

```
#define ChHideSeries(pCh, seriesNum) (ChSetDataSerie:
```

## Overview

This macro sets the specified data series number show flag to be set to [HIDE\\_DATA](#).

## Input Parameters

Input Parameters	Description
pCh	Pointer to the chart object.
seriesNum	The data series number that will be modified. If this number is zero, all the entries' flag in the list will be set to <a href="#">HIDE_DATA</a> .

## Returns

Returns the same passed number if successful otherwise -1 if unsuccessful.

## Preconditions

none

## Side Effects

none

## Example

---

See [ChShowSeries\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChHideSeries Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetShowSeriesCount Macro

## Chart

```
C
```

```
#define ChGetShowSeriesCount(pCh) (pCh->prm.seriesCo
```

## Overview

This macro shows the number of data series that has its show flag set to [SHOW\\_DATA](#)

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the number of data series with its show flag set to [SHOW\\_DATA](#).

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetShowSeriesCount Macro](#)

# ChGetShowSeriesStatus Macro

## Chart

```
C
#define ChGetShowSeriesStatus(pDSeries) (pDSeries->sl
```

## Overview

This macro returns the show ID status of the [DATASERIES](#).

## Input Parameters

Input Parameters	Description
pDSeries	Pointer to the data series( <a href="#">DATASERIES</a> ) that is being checked.

## Returns

Returns the status of the show flag. 1 - ([SHOW\\_DATA](#)) means that the show status flag is set. 0 - ([HIDE\\_DATA](#)) means that the show status flag is not set.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetShowSeriesStatus Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# ChSetValueLabel Macro

## Chart

C

```
#define ChSetValueLabel(pCh, pNewValueLabel) (((CHAR*
```

## Overview

This macro sets the address of the current text string used for the value axis label of the bar chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewYLabel	pointer to the string to be used as an value axis label of the bar chart.

## Returns

none.

## Preconditions

none

## Side Effects

none

## Example

See [ChCreate\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChSetValueLabel Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetValueLabel Macro

## Chart

```
C
#define ChGetValueLabel(pCh) (((CHART *)pCh)->prm.pVl
```

## Overview

This macro returns the address of the current text string used for the value axis label of the bar chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the pointer to the current value axis label text of the bar chart.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetValueLabel Macro](#)

# ChGetValueMax Macro

## Chart

```
C
```

```
#define ChGetValueMax(pCh) (pCh->prm.valMax)
```

## Overview

This macro returns the current maximum value that will be drawn for bar charts.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the maximum value set when bar charts are drawn.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetValueMax Macro](#)

# ChGetValueMin Macro

## Chart

```
C
```

```
#define ChGetValueMin(pCh) (pCh->prm.valMin)
```

## Overview

This macro returns the current minimum value that will be drawn for bar charts.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the minimum value set when bar charts are drawn.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetValueMin Macro](#)

# ChSetValueRange Function

## Chart

```
C
void ChSetValueRange(
    CHART * pCh,
    WORD min,
    WORD max
);
```

## Overview

This function sets the minimum and maximum range of values that the bar chart will show. The criteria is that  $\text{min} \leq \text{max}$ .

## Input Parameters

Input Parameters	Description
CHART * pCh	Pointer to the chart object.
WORD min	Minimum value that will be displayed in the bar chart.
WORD max	Maximum value that will be displayed in the bar chart.

## Returns

none.

## Preconditions

none

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChSetValueRange Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetValueRange Macro

## Chart

```
C
#define ChGetValueRange(pCh) (pCh->prm.valMax - pCh->prm.valMin)
```

## Overview

This macro gets the current range for bar charts. The value returned is calculated from the current (valMax - valMin) set. To get the minimum use [ChGetValueMin\(\)](#) and to get the maximum use [ChGetValueMax\(\)](#).

## Input Parameters

Input Parameters	Description
pCh	Pointer to the chart object.

## Returns

Value range computed from valMax-valMin.

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetValueRange Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# ChSetSampleLabel Macro

## Chart

C

```
#define ChSetSampleLabel(pCh, pNewXLabel) (((CHART *
```

## Overview

This macro sets the address of the current text string used for the sample axis label of the bar chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewXLabel	pointer to the string to be used as an sample axis label of the bar chart.

## Returns

none.

## Preconditions

none

## Side Effects

none

## Example

See [ChCreate\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChSetSampleLabel Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetSampleLabel Macro

## Chart

```
C
```

```
#define ChGetSampleLabel(pCh) (((CHART *)pCh)->prm.p!
```

## Overview

This macro returns the address of the current text string used for the sample axis label of the bar chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the pointer to the current sample axis label text of the bar chart.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetSampleLabel Macro](#)

# ChGetSampleStart Macro

## Chart

```
C
#define ChGetSampleStart(pCh) (((CHART *)pCh)->prm.si
```

## Overview

This macro returns the sampling start value.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the sample start point.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetSampleStart Macro](#)

# ChGetSampleEnd Macro

## Chart

```
C
#define ChGetSampleEnd(pCh) ((CHART *)pCh)->prm.smp1l
```

## Overview

This macro returns the sampling end value.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the sample end point.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetSampleEnd Macro](#)

# ChSetPercentRange Function

## Chart

```
C
void ChSetPercentRange(
    CHART * pCh,
    WORD min,
    WORD max
);
```

## Overview

This function sets the minimum and maximum range of percentage that the bar chart will show. The criteria is that  $\text{min} \leq \text{max}$ . This affects bar charts only and CH\_PERCENTAGE bit state is set.

## Input Parameters

Input Parameters	Description
CHART * pCh	Pointer to the chart object.
WORD min	Minimum percentage value that will be displayed in the bar chart.
WORD max	Maximum percentage value that will be displayed in the bar chart.

## Returns

none.

## Preconditions

none

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChSetPercentRange Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetPercentRange Macro

## Chart

```
C
```

```
#define ChGetPercentRange(pCh) (pCh->prm.perMax - pCh->prm.perMin)
```

## Overview

This macro gets the percentage range for bar charts. The value returned is calculated from percentage max - min. To get the minimum use [ChGetPercentMin\(\)](#) and to get the maximum use [ChGetPercentMax\(\)](#).

## Input Parameters

Input Parameters	Description
pCh	Pointer to the chart object.

## Returns

Percentage range computed from max-min.

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetPercentRange Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# ChSetSampleRange Function

## Chart

```
C
void ChSetSampleRange(
    CHART * pCh,
    WORD start,
    WORD end
);
```

## Overview

This function sets the sample start and sample end when drawing the chart. Together with the data series' [SHOW\\_DATA](#) flags the different way of displaying the chart data is achieved.

Start & End Value	The # of Data Series Flag Set	<a href="#">Chart</a> Description
Start <= End	1	Show the data indicated by Start and End points of the <a href="#">DATASERIES</a> with the flag set
Start = End	1	Show the data indicated by Start or End points of the <a href="#">DATASERIES</a> with the flag set
Start, End don't care	> 1	Show the data indicated by Start point of the <a href="#">DATASERIES</a> with the flag set. Each samples of all checked data series are grouped together according to sample number.

## Input Parameters

---

Input Parameters	Description
CHART * pCh	Pointer to the chart object.
WORD start	Start point of the data samples to be displayed.
WORD end	End point of the data samples to be displayed.

## Returns

---

none.

## Preconditions

---

none

## Side Effects

---

none.

## Example

---

See [ChCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChSetSampleRange Function](#)

# ChGetSampleRange Macro

## Chart

```
C
```

```
#define ChGetSampleRange(pCh) (ChGetSampleEnd(pCh) -
```

## Overview

This macro gets the sample range for pie or bar charts. The value returned is calculated from `smplEnd` - `smplStart`.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the chart object.

## Returns

Sample range computed from `smplEnd` - `smplStart`.

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetSampleRange Macro](#)

# ChGetPercentMax Macro

## Chart

```
C
```

```
#define ChGetPercentMax(pCh) (pCh->prm.perMax)
```

## Overview

This macro returns the current maximum value of the percentage range that will be drawn for bar charts when CH\_PERCENTAGE bit state is set.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the maximum percentage value set when bar charts are drawn.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetPercentMax Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# ChGetPercentMin Macro

## Chart

```
C
#define ChGetPercentMin(pCh) (pCh->prm.perMin)
```

## Overview

This macro returns the current minimum value of the percentage range that will be drawn for bar charts when CH\_PERCENTAGE bit state is set.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the minimum percentage value when bar charts are drawn.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetPercentMin Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# ChSetColorTable Macro

## Chart

```
C
#define ChSetColorTable(pCh, pNewTable) (((CHART *)|
```

## Overview

This macro sets the color table used to draw the data in pie and bar charts.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewTable	Pointer to the color table that will be used.

## Returns

none.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChSetColorTable Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# ChGetColorTable Macro

## Chart

```
C
#define ChGetColorTable(pCh) (((CHART *)pCh)->prm.pCt
```

## Overview

This macro returns the current color table used for the pie and bar charts.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the address of the color table used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetColorTable Macro](#)

# ChSetTitle Macro

## Chart

```
C
#define ChSetTitle(pCh, pNewTitle) (((CHART *)pCh)->
```

## Overview

This macro sets the address of the current text string used for the title of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewTitle	pointer to the string to be used as a title of the chart.

## Returns

none.

## Preconditions

none

## Side Effects

none

## Example

See [ChCreate\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChSetTitle Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetTitle Macro

## Chart

```
C
```

```
#define ChGetTitle(pCh) (((CHART *)pCh)->prm.pTitle)
```

## Overview

This macro returns the address of the current text string used for the title of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the pointer to the current title text used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetTitle Macro](#)

# ChSetTitleFont Macro

## Chart

```
C
#define ChSetTitleFont(pCh, pNewFont) (((CHART *)pCh
```

## Overview

This macro sets the location of the font used for the title of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewFont	Pointer to the font used.

## Returns

none.

## Preconditions

none

## Side Effects

none

## Example

See [ChCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChSetTitleFont Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetTitleFont Macro

## Chart

```
C
#define ChGetTitleFont(pCh) (((CHART *)pCh)->prm.pTi:
```

## Overview

This macro returns the location of the font used for the title of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the address of the current font used for the title text.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetTitleFont Macro](#)

# ChGetAxisLabelFont Macro

## Chart

```
C
```

```
#define ChGetAxisLabelFont(pCh) (((CHART *)pCh)->prm
```

## Overview

This macro returns the location of the font used for the X and Y axis labels of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the address of the current font used for the title text.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetAxisLabelFont Macro](#)

# ChSetAxisLabelFont Macro

## Chart

```
C
#define ChSetAxisLabelFont(pCh, pNewFont) (((CHART *
```

## Overview

This macro sets the location of the font used for the X and Y axis labels of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewFont	Pointer to the font used.

## Returns

none.

## Preconditions

none

## Side Effects

none

## Example

See [ChCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChSetAxisLabelFont Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChGetGridLabelFont Macro

## Chart

C

```
#define ChGetGridLabelFont(pCh) (((CHART *)pCh)->prm
```

## Overview

This macro returns the location of the font used for the X and Y axis grid labels of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.

## Returns

Returns the address of the current font used for the title text.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChGetGridLabelFont Macro](#)

# ChSetGridLabelFont Macro

## Chart

```
C
#define ChSetGridLabelFont(pCh, pNewFont) (((CHART *
```

## Overview

This macro sets the location of the font used for the X and Y axis grid labels of the chart.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the object.
pNewFont	Pointer to the font used.

## Returns

none.

## Preconditions

none

## Side Effects

none

## Example

See [ChCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChSetGridLabelFont Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChFreeDataSeries Function

## Chart

```
C
void ChFreeDataSeries(
    void * pObj
);
```

## Overview

This function removes [DATASERIES](#) object from the list of [DATASERIES](#) objects and frees the memory used of that removed object.

## Input Parameters

Input Parameters	Description
pCh	Pointer to the chart object.

## Returns

none.

## Preconditions

none

## Side Effects

none.

## Example

[Copy Code](#)

```
void ClearChartData(CHART *pCh) {  
    if(pCh->pChartData != NULL)  
        // remove the all data series  
        ChFreeDataSeries(pCh;  
    }  
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) >  
[ChFreeDataSeries Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ChTranslateMsg Function

## Chart

```
C
WORD ChTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
CH_MSG_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If event occurs and the x, y position falls in the area of the chart.
OBJ_MSG_INVALID	Any	Any	If the message does not affect the object.

## Input Parameters

--	--

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pCh	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- CH\_MSG\_SELECTED – [Chart](#) area is selected
- OBJ\_MSG\_INVALID – [Chart](#) is not affected

none.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [ChTranslateMsg Function](#)

# CHART Structure

## Chart

```
C
typedef struct {
    OBJ_HEADER  hdr;
    CHARTPARAM  prm;
    DATASERIES * pChData;
} CHART;
```

## Overview

Defines the parameters required for a chart Object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
CHARTPARAM prm;	Structure for the parameters of the chart.
DATASERIES * pChData;	Pointer to the first chart data series in the link list of data series.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [CHART Structure](#)

# DATASERIES Structure

## Chart

```
C
typedef struct {
    XCHAR * pSData;
    WORD samples;
    BYTE show;
    WORD * pData;
    void * pNextData;
} DATASERIES;
```

## Overview

Defines a variable for the [CHART](#) object. It specifies the number of samples, pointer to the array of samples for the data series and pointer to the next data series. A member of this structure (show) is used as a flag to determine if the series is to be drawn or not. Together with the `smplStart` and `smplEnd` it will determine what kind of chart will be drawn.

## Members

Members	Description
XCHAR * pSData;	Pointer to the data series name.
WORD samples;	Indicates the number of data samples (or data points) contained in the array specified by <code>pData</code> .
BYTE show;	The flag to indicate if the data series will be shown or not. If this flag is set to <a href="#">SHOW_DATA</a> , the data series will be shown. If <a href="#">HIDE_DATA</a> , the data series will

	not be shown.
WORD * pData;	Pointer to the array of data samples.
void * pNextData;	Pointer to the next data series. NULL if no other data series follows.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [DATASERIES Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CHARTPARAM Structure

## Chart

```
C
typedef struct {
    XCHAR * pTitle;
    XCHAR * pSmplLabel;
    XCHAR * pValLabel;
    SHORT seriesCount;
    WORD smplStart;
    WORD smplEnd;
    WORD valMax;
    WORD valMin;
    WORD perMax;
    WORD perMin;
    GFX_COLOR * pColor;
    void * pTitleFont;
    void * pAxisLabelsFont;
    void * pGridLabelsFont;
} CHARTPARAM;
```

## Overview

Defines the parameters for the [CHART](#) object.

## Members

Members	Description
XCHAR * pTitle;	Pointer to the Title of the chart.
XCHAR * pSmplLabel;	Pointer to the bar chart sample axis label. Depending
XCHAR * pValLabel;	Pointer to the bar chart value axis label.

	Depending
SHORT seriesCount;	Number of data series that will be displayed when chart is drawn.
WORD smplStart;	Start point of data sample range to be displayed (minimum/default value = 1)
WORD smplEnd;	End point of data sample range to be displayed.
WORD valMax;	Maximum value of a sample that can be displayed.
WORD valMin;	Minimum value of a sample that can be displayed.
WORD perMax;	Maximum value of the percentage range that can be displayed.
WORD perMin;	Minimum value of the percentage range that can be displayed.
GFX_COLOR * pColor;	Pointer to the color table used to draw the chart data.
void * pTitleFont;	Pointer to the font used for the title label of the chart.
void * pAxisLabelsFont;	Pointer to the font used for X and Y axis labels.
void * pGridLabelsFont;	Pointer to the font used for X and Y axis grid labels.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [CHARTPARAM Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Color Table

[Macros](#) | [Chart](#)

Default color table used to draw data points in a chart.

## Macros

Name	Description
<a href="#">CH_CLR0</a>	Bright Blue
<a href="#">CH_CLR1</a>	Bright Red
<a href="#">CH_CLR2</a>	Bright Green
<a href="#">CH_CLR3</a>	Bright Yellow
<a href="#">CH_CLR4</a>	Orange
<a href="#">CH_CLR5</a>	Blue
<a href="#">CH_CLR6</a>	Red
<a href="#">CH_CLR7</a>	Green
<a href="#">CH_CLR8</a>	Yellow
<a href="#">CH_CLR9</a>	Dark Orange
<a href="#">CH_CLR10</a>	Light Blue
<a href="#">CH_CLR11</a>	Light Red
<a href="#">CH_CLR12</a>	Light Green
<a href="#">CH_CLR13</a>	Light Yellow
<a href="#">CH_CLR14</a>	Light Orange

[CH\\_CLR15](#)

Gold

## Module

---

[Chart](#)

## Links

---

[Macros, Chart](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_CLR0 Macro

---

C

```
#define CH_CLR0 WHITE
```

### Description

---

Bright Blue

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR0 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR1 Macro

---

```
C
```

```
#define CH_CLR1 BLACK
```

## Description

---

Bright Red

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR1 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_CLR2 Macro

---

```
C
```

```
#define CH_CLR2 WHITE
```

### Description

---

Bright Green

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR2 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR3 Macro

```
C
```

```
#define CH_CLR3 BLACK
```

## Description

Bright Yellow

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR3 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_CLR4 Macro

---

```
C
```

```
#define CH_CLR4 WHITE
```

### Description

---

Orange

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR4 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR5 Macro

---

```
C
```

```
#define CH_CLR5 BLACK
```

## Description

---

Blue

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR5 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_CLR6 Macro

---

C

```
#define CH_CLR6 WHITE
```

### Description

---

Red

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR6 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR7 Macro

---

```
C
```

```
#define CH_CLR7 BLACK
```

## Description

---

Green

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR7 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CH\_CLR8 Macro

---

```
C
```

```
#define CH_CLR8 WHITE
```

### Description

---

Yellow

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR8 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR9 Macro

---

```
C
```

```
#define CH_CLR9 BLACK
```

## Description

---

Dark Orange

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR9 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR10 Macro

---

```
C
```

```
#define CH_CLR10 WHITE
```

## Description

---

Light Blur

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR10 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR11 Macro

---

```
C
```

```
#define CH_CLR11 BLACK
```

## Description

---

Light Red

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR11 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR12 Macro

---

```
C
```

```
#define CH_CLR12 WHITE
```

## Description

---

Light Green

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR12 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR13 Macro

---

```
C
```

```
#define CH_CLR13 BLACK
```

## Description

---

Light Yellow

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR13 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR14 Macro

---

```
C
```

```
#define CH_CLR14 WHITE
```

## Description

---

Light Orange

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR14 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CH\_CLR15 Macro

---

```
C
```

```
#define CH_CLR15 BLACK
```

## Description

---

Gold

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#) > [CH\\_CLR15 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

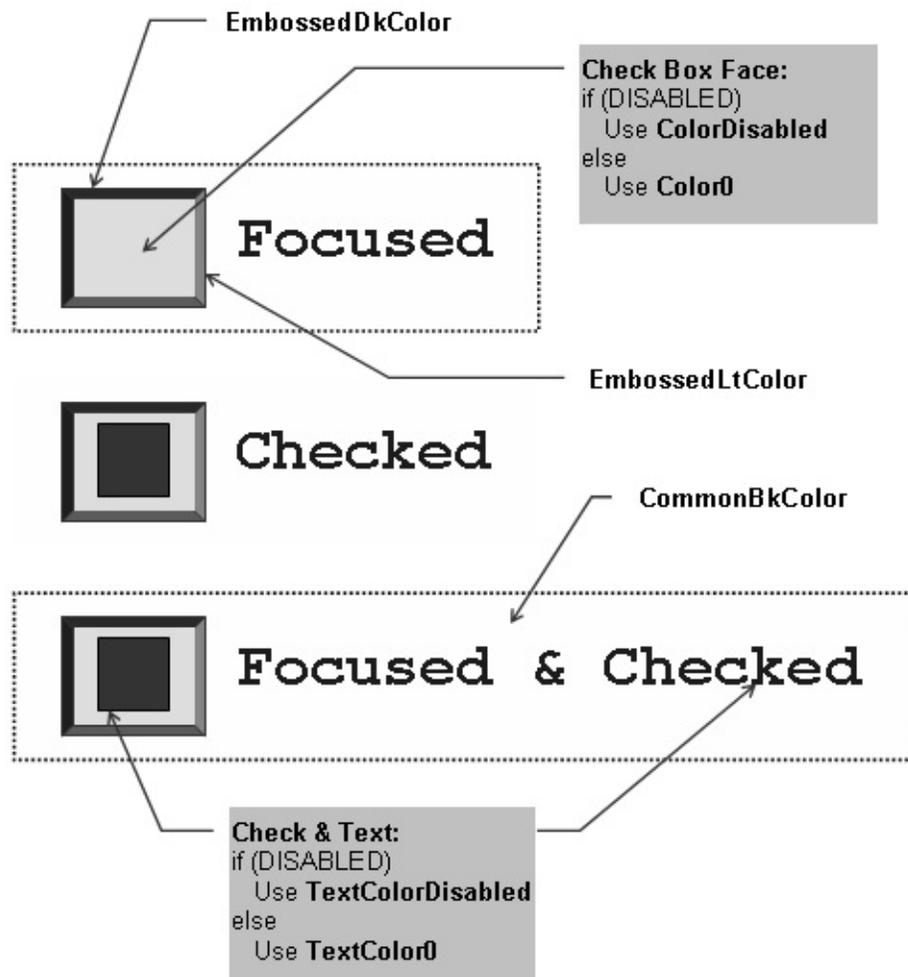
# Checkbox

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Check Box supports Keyboard and Touchscreen inputs, replying to their events with the following messages:

1. `CB_MSG_UNCHECKED` - When the check box is unchecked.
2. `CB_MSG_CHECKED` - When check box is unchecked.

The Check Box Object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



## Functions

	Name	Description
◆	<a href="#">CbCreate</a>	This function creates a <b>CHECKBOX</b> 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.

	<a href="#"><u>CbDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<a href="#"><u>CbSetText</u></a>	<p>This function sets the text that will be used.</p>
	<a href="#"><u>CbMsgDefault</u></a>	<p>This function performs the actual state change based on the translated message given. The following state changes are supported:</p>
	<a href="#"><u>CbTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

## Macros

Name	Description
<a href="#"><u>CbGetText</u></a>	<p>This macro returns the location of the text used for the check box.</p>

## Structures

Name	Description
<a href="#">CHECKBOX</a>	The structure contains check box data

## Topics

---

Name	Description
<a href="#">Check Box States</a>	List of Checkbox bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Check Box States

Macros | [Checkbox](#)

List of [Checkbox](#) bit states.

## Macros

Name	Description
<a href="#">CB_CHECKED</a>	Checked state
<a href="#">CB_DISABLED</a>	Disabled state
<a href="#">CB_DRAW</a>	Whole check box must be redrawn
<a href="#">CB_DRAW_CHECK</a>	Check box mark should be redrawn
<a href="#">CB_DRAW_FOCUS</a>	Focus must be redrawn
<a href="#">CB_FOCUSED</a>	Focus state
<a href="#">CB_HIDE</a>	Check box must be removed from screen

## Module

[Checkbox](#)

## Links

[Macros](#), [Checkbox](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#)

# CB\_CHECKED Macro

---

```
C
#define CB_CHECKED 0x0004 // Checked state
```

## Description

---

Checked state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_CHECKED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CB\_DISABLED Macro

---

C

```
#define CB_DISABLED 0x0002 // Disabled state
```

### Description

---

Disabled state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CB\_DRAW Macro

---

```
C
#define CB_DRAW 0x4000 // whole check box must be r
```

### Description

---

Whole check box must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CB\_DRAW\_CHECK Macro

---

C

```
#define CB_DRAW_CHECK 0x1000 // Check box mark shou.
```

### Description

---

Check box mark should be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_DRAW\\_CHECK Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CB\_DRAW\_FOCUS Macro

---

C

```
#define CB_DRAW_FOCUS 0x2000 // Focus must be redra
```

### Description

---

Focus must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_DRAW\\_FOCUS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CB\_FOCUSED Macro

---

C

```
#define CB_FOCUSED 0x0001 // Focus state
```

### Description

---

Focus state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CB\_HIDE Macro

---

```
C
#define CB_HIDE 0x8000 // Check box must be removed
```

### Description

---

Check box must be removed from screen

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#) > [CB\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CbCreate Function

## Checkbox

```
C
CHECKBOX * CbCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    XCHAR * pText,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object
SHORT bottom	Bottom most position of the object

WORD state	Sets the initial state of the object
XCHAR * pText	Pointer to the text of the check box.
GOL_SCHEME * pScheme	Pointer to the style scheme

## Returns

Returns the pointer to the object created

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
GOL_SCHEME *pScheme;
CHECKBOX *pCb[2];

    pScheme = GOLCreateScheme();
    pCb = CbCreate(ID_CHECKBOX1,           // ID
                  20,135,150,175,        // dims
                  CB_DRAW,              // Draw t
                  "Scale",              // text
                  pScheme);            // use th

    pCb = CbCreate(ID_CHECKBOX2,         // ID
                  170,135,300,175,      // dims
                  CB_DRAW,              // Draw t
                  "Animate",            // text
                  pScheme);            // use th
```

```
while( !CbDraw(pCb[0]) );           // draw t
while( !CbDraw(pCb[1]) );
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [CbCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CbDraw Function

## Checkbox

```
C  
WORD CbDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pCb	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending

drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [CbCreate\(\)](#) Example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [CbDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CbGetText Macro

## Checkbox

```
C
```

```
#define CbGetText(pCb) pCb->pText
```

## Overview

This macro returns the location of the text used for the check box.

## Input Parameters

Input Parameters	Description
pCb	Pointer to the object

## Returns

Returns the location of the text used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [CbGetText Macro](#)

# CbSetText Function

## Checkbox

```
C
void CbSetText(
    CHECKBOX * pCb,
    XCHAR * pText
);
```

## Overview

This function sets the text that will be used.

## Input Parameters

Input Parameters	Description
CHECKBOX * pCb	The pointer to the check box whose text will be modified.
XCHAR * pText	The pointer to the text that will be used.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) >

## CbSetText Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CbMsgDefault Function

## Checkbox

C

```
void CbMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
CB_MSG_CHECKED	Touch Screen, Keyboard	Set <a href="#">CB_CHECKED</a>	Check Bit will be redrawn checked state.
CB_MSG_UNCHECKED	Touch Screen, Keyboard	Clear <a href="#">CB_CHECKED</a>	Check Bit will be redrawn un-checked state.

## Input Parameters

Input Parameters	Description

WORD translatedMsg	The translated message
GOL_MSG * pMsg	The pointer to the GOL message
pCb	The pointer to the object whose state will be modified

## Returns

---

none

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [CbMsgDefault Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CbTranslateMsg Function

## Checkbox

```
C
WORD CbTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
CB_MSG_CHECKED	Touch Screen	EVENT_PRESS	If event occurs in the x,y position of the object area while the object is unchecked.
	Keyboard	EVENT_KEYSCAN	If event occurs and the parameter matches the object parameter and the parameter matches the object parameter <u>SCAN</u> while the object is unchecked.

CB_MSG_UNCHECKED	Touch Screen	EVENT_PRESS	If ever x,y po area c while checke
	Keyboard	EVENT_KEYSCAN	If ev parame matche and pa matche <u>SCAN</u> <u>SCAN</u> while checke
OBJ_MSG_INVALID	Any	Any	If the affect tl

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	pointer to the message struct containing the message the user
pCb	the pointer to the object where the message will be evaluated to check if the message will affect the object

## Returns

Returns the translated message depending on the received GOL message:

- CB\_MSG\_CHECKED – Check Box is checked.
- CB\_MSG\_UNCHECKED – Check Box is unchecked.

- OBJ\_MSG\_INVALID – Check Box is not affected.

## Preconditions

---

none

## Side Effects

---

none

## Example

---

Usage is similar to [BtnTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [CbTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CHECKBOX Structure

## Checkbox

C

```
typedef struct {  
    OBJ_HEADER hdr;  
    SHORT textHeight;  
    XCHAR * pText;  
} CHECKBOX;
```

## Overview

The structure contains check box data

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT textHeight;	Pre-computed text height
XCHAR * pText;	Pointer to text

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [CHECKBOX Structure](#)

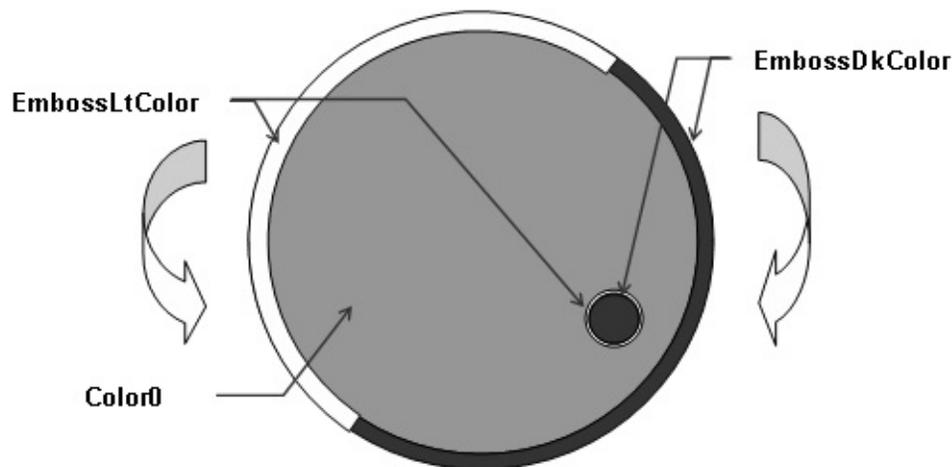
# Round Dial

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Dial supports only Touchscreen inputs, replying to their events with the following messages:

1. `RD_MSG_CLOCKWISE` - When movement of the touch is in the face of the dial and in the clockwise direction.
2. `RD_MSG_CTR_CLOCKWISE` - When movement of the touch is in the face of the dial and in the counter clockwise direction.

The Dial object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



`CommonBkColor` – used to hide the slider from the screen.

## Functions

	Name	Description
◆	<a href="#"><u>RdiaCreate</u></a>	This function creates a <b>ROUND DIAL</b> 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.
◆	<a href="#"><u>RdiaDraw</u></a>	This function renders the object on the screen using the current parameter settings. Location of the object is determined by the center (x,y) position and the radius parameters. The colors used are dependent on the state of 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.
◆	<a href="#"><u>RdiaMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
◆	<a href="#"><u>RdiaTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

## Macros

Name	Description

<a href="#">RdiaIncVal</a>	Used to directly increment the value. The delta change used is the resolution setting (res).
<a href="#">RdiaDecVal</a>	Used to directly decrement the value. The delta change used is the resolution setting (res).
<a href="#">RdiaGetVal</a>	Returns the current dial value. Value is always in the 0-max range inclusive.
<a href="#">RdiaSetVal</a>	Sets the value to the given new value. Value set must be in 0-max range inclusive.

## Structures

Name	Description
<a href="#">ROUNDIAL</a>	Defines the parameters required for a dial Object. The curr_xPos, curr_yPos, new_xPos and new_yPos parameters are internally generated to aid in the redrawing of the dial. User must avoid modifying these values.

## Topics

Name	Description
<a href="#">Dial States</a>	List of Dial bit states.

## Links

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Dial States

[Macros](#) | [Round Dial](#)

List of [Dial](#) bit states.

## Macros

Name	Description
<a href="#">RDIA_DISABLED</a>	Bit for disabled state.
<a href="#">RDIA_DRAW</a>	Bit to indicate object must be redrawn.
<a href="#">RDIA_HIDE</a>	Bit to indicate object must be removed from screen.
<a href="#">RDIA_ROT_CCW</a>	Bit for rotate counter clockwise state.
<a href="#">RDIA_ROT_CW</a>	Bit for rotate clockwise state.

## Module

[Round Dial](#)

## Links

[Macros, Round Dial](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#)

## RDIA\_DISABLED Macro

---

C

```
#define RDIA_DISABLED 0x0002 // Bit for disabled sta
```

### Description

---

Bit for disabled state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#) > [RDIA\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RDIA\_DRAW Macro

---

C

```
#define RDIA_DRAW 0x4000 // Bit to indicate object i
```

### Description

---

Bit to indicate object must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#) > [RDIA\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RDIA\_HIDE Macro

---

```
C
#define RDIA_HIDE 0x8000 // Bit to indicate object i
```

### Description

---

Bit to indicate object must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#) > [RDIA\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RDIA\_ROT\_CCW Macro

---

C

```
#define RDIA_ROT_CCW 0x0008 // Bit for rotate counter
```

### Description

---

Bit for rotate counter clockwise state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#) > [RDIA\\_ROT\\_CCW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RDIA\_ROT\_CW Macro

---

```
C
#define RDIA_ROT_CW 0x0004 // Bit for rotate clockw.
```

### Description

---

Bit for rotate clockwise state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#) > [RDIA\\_ROT\\_CW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaCreate Function

## Round Dial

C

```
ROUND DIAL * RdiaCreate(  
    WORD ID,  
    SHORT x,  
    SHORT y,  
    SHORT radius,  
    WORD state,  
    SHORT res,  
    SHORT value,  
    SHORT max,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a **ROUND DIAL** 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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT x	Location of the center of the dial in the x coordinate.
SHORT y	Location of the center of the dial in the y coordinate.

SHORT radius	Defines the radius of the dial.
WORD state	Sets the initial state of the object.
SHORT res	Sets the resolution of the dial when rotating clockwise or counter clockwise.
SHORT value	Sets the initial value of the dial.
SHORT max	Sets the maximum value of the dial.
GOL_SCHEME * pScheme	Pointer to the style scheme used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
GOL_SCHEME *pScheme;  
ROUNDIAL *pDial;  
WORD state;  
  
pScheme = GOLCreateScheme();  
state = RDIA_DRAW;  
  
// creates a dial at (50,50) x,y location, with
```

```
// of 50, a resolution of 2 and maximum value o  
pDial = RdiaCreate(1,50,50,25,118,0, state, 2,  
// check if dial was created  
if (pDial == NULL)  
    return 0;  
  
return 1;
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) >  
[RdiaCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaDraw Function

## Round Dial

```
C  
WORD RdiaDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. Location of the object is determined by the center (x,y) position and the radius parameters. The colors used are dependent on the state of 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.

## Input Parameters

Input Parameters	Description
pDia	Pointer to the object

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

## Preconditions

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaIncVal Macro

## Round Dial

```
C
```

```
#define RdiaIncVal(pDia) RdiaSetVal(pDia, (pDia->val
```

## Overview

Used to directly increment the value. The delta change used is the resolution setting (res).

## Input Parameters

Input Parameters	Description
pDia	Pointer to the object.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

```
WORD updatedVal, prevVal;  
ROUND DIAL *pDia;
```

[Copy Code](#)

```
// assuming pDia is initialized to an existing  
// assume GetInput() is a function that retrieves  
prevVal = RdiaGetVal(pDia);  
updatedVal = GetInput();  
if (updatedVal > prevVal)  
    RdiaIncVal(pDia);  
if (updatedVal < prevVal)  
    RdiaDecVal(pDia);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaIncVal Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaDecVal Macro

## Round Dial

```
C
#define RdiaDecVal(pDia) RdiaSetVal(pDia, (pDia->pos
```

## Overview

Used to directly decrement the value. The delta change used is the resolution setting (res).

## Input Parameters

Input Parameters	Description
pDia	Pointer to the object.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

Refer to [RdiaIncVal\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaDecVal Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaGetVal Macro

## Round Dial

```
C
```

```
#define RdiaGetVal(pDia) (pDia)->value
```

## Overview

Returns the current dial value. Value is always in the 0-max range inclusive.

## Input Parameters

Input Parameters	Description
pDia	Pointer to the object.

## Returns

Returns the current value of the dial.

## Preconditions

none

## Side Effects

none

## Example

```
WORD currVal;  
ROUND DIAL *pDia;
```

[Copy Code](#)

```
// assuming pDia is initialized to an existing  
currVal = RdiaGetVal(pDia);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaGetVal Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaSetVal Macro

## Round Dial

```
C
```

```
#define RdiaSetVal(pDia, newVal) (pDia)->value = newVal
```

## Overview

Sets the value to the given new value. Value set must be in 0-max range inclusive.

## Input Parameters

Input Parameters	Description
pDia	Pointer to the object.
newVal	New dial value.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

```
WORD updatedVal;
```

[Copy Code](#)

```
ROUND DIAL *pDia;
```

```
// assuming pDia is initialized to an existing  
// assume GetInput() is a function that retrieves  
updatedVal = GetInput();  
RdiaSetVal(pDia, updatedVal);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaSetVal Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaMsgDefault Function

## Round Dial

C

```
void RdiaMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear Bit	State	Desc
RD_MSG_CLOCKWISE	Touch Screen	Set <a href="#">RDIA_ROT_CW</a> , Set <a href="#">RDIA_DRAW</a>		<a href="#">Dial</a> redra clock updat
RD_MSG_CTR_CLOCKWISE	Touch Screen	Set <a href="#">RDIA_ROT_CCW</a> , Set <a href="#">RDIA_DRAW</a>		<a href="#">Dial</a> redra count clock updat

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message

GOL_MSG * pMsg	The pointer to the GOL message
pDia	The pointer to the object whose state will be modified

---

## Returns

none

---

## Preconditions

none

---

## Side Effects

none

---

## Example

See [RdiaTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaMsgDefault Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RdiaTranslateMsg Function

## Round Dial

```
C
WORD RdiaTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

Translated Message	Input Source	Events	Description
RD_MSG_CLOCKWISE	Touch Screen	EVENT_MOVE	If an event occurs the position in the face of the <a href="#">Dial</a> moving the clockwise rotation.
RD_MSG_CTR_CLOCKWISE	Touch Screen	EVENT_MOVE	If an event occurs the position in the face of the <a href="#">Dial</a> moving the counter-clockwise rotation.

			the <a href="#">Dial</a> moving the counter clockwise rotation.
OBJ_MSG_INVALID	Any	Any	If message not affect object.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pDia	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- RD\_MSG\_CLOCKWISE – [Dial](#) is moved in a clockwise direction.
- RD\_MSG\_CTR\_CLOCKWISE – [Dial](#) is moved in a counter clockwise direction.
- OBJ\_MSG\_INVALID – [Dial](#) is not affected

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
void MyGOLMsg(GOL_MSG *pMsg){

    OBJ_HEADER *pCurrentObj;
    WORD objMsg;

    if(pMsg->event == EVENT_INVALID)
        return;
    pCurrentObj = GOLGetList();

    while(pCurrentObj != NULL){
        // Process only ROUND DIAL
        if(!IsObjUpdated(pCurrentObj)){
            switch(pCurrentObj->type){
                case OBJ_ROUNDIAL:
                    objMsg = RdiaTranslateMsg((ROUND
                    if(objMsg == OBJ_MSG_INVALID)
                        break;
                    if(GOLMsgCallback(objMsg, pCurrentObj,
                        RdiaMsgDefault(objMsg, (ROUND
                    break;
                default: break;
            }
        }
    }
    pCurrentObj = pCurrentObj->pNxtObj;
}
```

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [RdiaTranslateMsg Function](#)

[Contents](#) | [Index](#) | [Home](#)

# ROUNDIAL Structure

## Round Dial

C

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT xCenter;
    SHORT yCenter;
    SHORT radius;
    SHORT value;
    WORD max;
    WORD res;
    SHORT curr_xPos;
    SHORT curr_yPos;
    SHORT new_xPos;
    SHORT new_yPos;
    SHORT vAngle;
} ROUNDIAL;
```

## Overview

Defines the parameters required for a dial Object. The curr\_xPos, curr\_yPos, new\_xPos and new\_yPos parameters are internally generated to aid in the redrawing of the dial. User must avoid modifying these values.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT xCenter;	x coordinate center position.

SHORT yCenter;	y coordinate center position.
SHORT radius;	Radius of the dial.
SHORT value;	Initial value of the dial.
WORD max;	Maximum value of variable value (maximum = 65535).
WORD res;	Resolution of movement.
SHORT curr_xPos;	Current x position.
SHORT curr_yPos;	Current y position.
SHORT new_xPos;	New x position.
SHORT new_yPos;	New y position.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [ROUND DIAL Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

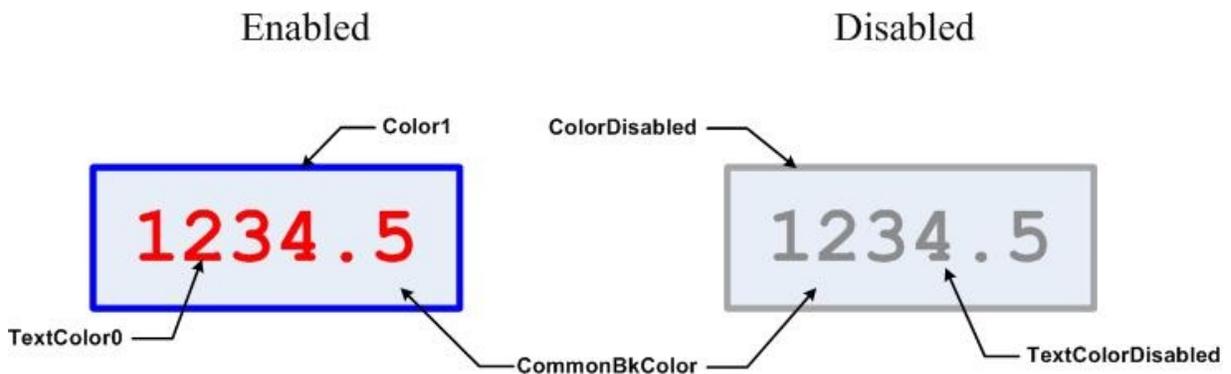
# Digital Meter

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

DigitalMeter supports only Touchscreen inputs, replying to touch screen events with the message:

DM\_MSG\_SELECTED.

The DigitalMeter object is rendered using the assigned style scheme. The following figure illustrates the color assignments for the digital meter.



## Functions

	Name	Description
◆	<a href="#">DmCreate</a>	This function creates a <b>DIGITALMETER</b> 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.
◆	<a href="#">DmDraw</a>	This function renders the object on the

		<p>screen using the current parameter settings. 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.</p> <p>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.</p>
◆	<a href="#"><u>DmSetValue</u></a>	This function sets the value that will be used for the object.
◆	<a href="#"><u>DmTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

## Macros

Name	Description
<a href="#"><u>DmGetValue</u></a>	This macro returns the current value used for the object.
<a href="#"><u>DmDecVal</u></a>	This macro is used to directly decrement the value.
<a href="#"><u>DmIncVal</u></a>	This macro is used to directly increment the value.

## Structures

Name	Description
<a href="#">DIGITALMETER</a>	Defines the parameters required for a Digital Meter Object.

## Topics

---

Name	Description
<a href="#">Digital Meter States</a>	List of Digital <a href="#">Meter</a> bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Digital Meter States

[Macros](#) | [Digital Meter](#)

List of Digital [Meter](#) bit states.

## Macros

Name	Description
<a href="#">DM_DISABLED</a>	Bit for disabled state.
<a href="#">DM_DRAW</a>	Bit to indicate object must be redrawn.
<a href="#">DM_HIDE</a>	Bit to remove object from screen.
<a href="#">DM_CENTER_ALIGN</a>	Bit to indicate value is center aligned.
<a href="#">DM_RIGHT_ALIGN</a>	Bit to indicate value is left aligned.
<a href="#">DM_FRAME</a>	Bit to indicate frame is displayed.
<a href="#">DM_UPDATE</a>	Bit to indicate that only text must be redrawn.

## Module

[Digital Meter](#)

## Links

[Macros](#), [Digital Meter](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#)

[Contents](#) | [Index](#) | [Home](#)

## DM\_DISABLED Macro

```
C  
#define DM_DISABLED 0x0002 // Bit for disabled state
```

### Description

Bit for disabled state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_DISABLED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DM\_DRAW Macro

---

```
C
#define DM_DRAW 0x4000 // Bit to indicate object must be redrawn.
```

### Description

---

Bit to indicate object must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DM\_HIDE Macro

---

C

```
#define DM_HIDE 0x8000 // Bit to remove object from
```

### Description

---

Bit to remove object from screen.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DM\_CENTER\_ALIGN Macro

```
C
#define DM_CENTER_ALIGN 0x0008 // Bit to indicate value is center aligned
```

### Description

Bit to indicate value is center aligned.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_CENTER\\_ALIGN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DM\_RIGHT\_ALIGN Macro

---

C

```
#define DM_RIGHT_ALIGN 0x0004 // Bit to indicate va.
```

### Description

---

Bit to indicate value is left aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_RIGHT\\_ALIGN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DM\_FRAME Macro

---

C

```
#define DM_FRAME 0x0010 // Bit to indicate frame is
```

### Description

---

Bit to indicate frame is displayed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_FRAME Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DM\_UPDATE Macro

---

```
C  
#define DM_UPDATE 0x2000 // Bit to indicate that on.
```

### Description

---

Bit to indicate that only text must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#) > [DM\\_UPDATE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DmCreate Function

## Digital Meter

C

```
DIGITALMETER * DmCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    WORD state,  
    DWORD Value,  
    BYTE NoOfDigits,  
    BYTE DotPos,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the object.
SHORT top	Top most position of the object.
SHORT right	Right most position of the object.

SHORT bottom	Bottom most position of the object.
WORD state	Sets the initial state of the object.
DWORD Value	Sets the initial value to be displayed
BYTE NoOfDigits	Sets the number of digits to be displayed
BYTE DotPos	Sets the position of decimal point in the display
GOL_SCHEME * pScheme	Pointer to the style scheme. Set to NULL if default style scheme is used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

```

GOL_SCHEME *pScheme;
DIGITALMETER *pDm;

pScheme = GOLCreateScheme();
state = DM_DRAW | DM_FRAME | DM_CENTER_ALIGN;
DmCreate(ID_DIGITALMETER1, // ID
        30,80,235,160, // dimension
        state, // has frame and center aligned
        789,4,1, // to display 078.9
        pScheme); // use given scheme

```

```
while(!DmDraw(pDm));           // draw the object
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DmCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DmDraw Function

## Digital Meter

```
C  
WORD DmDraw(  
    void * pObj  
);
```

### Overview

This function renders the object on the screen using the current parameter settings. 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.

### Input Parameters

Input Parameters	Description
pDm	Pointer to the object to be rendered.

### Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [DmCreate\(\)](#) Example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DmDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DmGetValue Macro

## Digital Meter

```
C
```

```
#define DmGetValue(pDm) pDm->Cvalue
```

## Overview

This macro returns the current value used for the object.

## Input Parameters

Input Parameters	Description
pDm	Pointer to the object.

## Returns

Returns the value used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DmGetValue Macro](#)

# DmSetValue Function

## Digital Meter

```
C
void DmSetValue(
    DIGITALMETER * pDm,
    DWORD Value
);
```

### Overview

This function sets the value that will be used for the object.

### Input Parameters

Input Parameters	Description
DIGITALMETER * pDm	The pointer to the object whose value will be modified.
DWORD Value	New value to be set for the <a href="#">Digital Meter</a> .

### Returns

none

### Preconditions

none

### Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) >

## DmSetValue Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DmDecVal Macro

## Digital Meter

C

```
#define DmDecVal(pDm, deltaValue) DmSetValue(pDm, (p
```

## Overview

This macro is used to directly decrement the value.

## Input Parameters

Input Parameters	Description
pDm	Pointer to the object.
deltaValue	Number to be subtracted to the current <a href="#">Digital Meter</a> value.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DmDecVal Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# DmIncVal Macro

## Digital Meter

```
C
```

```
#define DmIncVal(pDm, deltaValue) DmSetValue(pDm, (p
```

## Overview

This macro is used to directly increment the value.

## Input Parameters

Input Parameters	Description
pDm	Pointer to the object.
deltaValue	Number to be added to the current <a href="#">Digital Meter</a> value.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DmIncVal Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# DmTranslateMsg Function

## Digital Meter

```
C
WORD DmTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
DM_MSG_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If event occurs and the x position falls in the area of the <a href="#">Digital Meter</a> .
OBJ_MSG_INVALID	Any	Any	If the message does not affect the object.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pDm	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- DM\_MSG\_SELECTED – [Digital Meter](#) is selected
- OBJ\_MSG\_INVALID – [Digital Meter](#) is not affected

## Preconditions

none

## Side Effects

none

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DmTranslateMsg Function](#)

# DIGITALMETER Structure

## Digital Meter

C

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT textHeight;
    DWORD Cvalue;
    DWORD Pvalue;
    BYTE NoOfDigits;
    BYTE DotPos;
} DIGITALMETER;
```

## Overview

Defines the parameters required for a [Digital Meter](#) Object.

## Description

Structure: DIGITALMETER

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT textHeight;	Pre-computed text height
DWORD Cvalue;	Current value
DWORD Pvalue;	Previous value
BYTE NoOfDigits;	Number of digits to be displayed

|| BYTE DotPos;

|| Position of decimal point

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [DIGITALMETER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

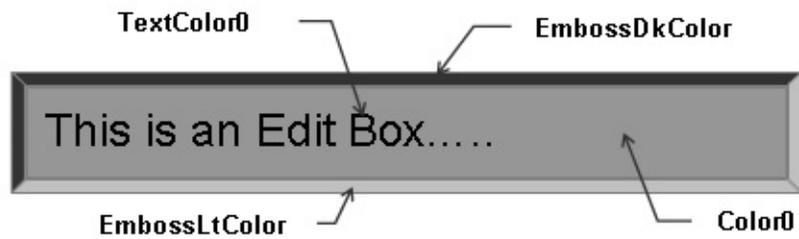
# Edit Box

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

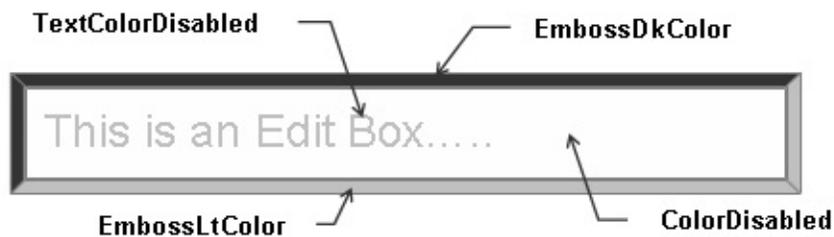
Edit Box supports only Keyboard inputs, replying to their events with the following messages:

1. EB\_MSG\_CHAR - when a character is to be inserted at the end of the current text.
2. EB\_MSG\_DEL - when a character is to be removed from the current text.

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**

## Functions

	Name	Description
◆	<a href="#"><u>EbCreate</u></a>	This function creates a <a href="#"><u>EDITBOX</u></a> object with the parameters given and initializes the default settings. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
◆	<a href="#"><u>EbDraw</u></a>	This function renders the object on the screen using the current parameter settings. Location of the object is determined by the left, top, right and bottom parameters. The colors used are

		<p>dependent on the state of the object. The font used is determined by the style scheme set.</p> <p>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.</p>
⇒	<a href="#"><u>EbSetText</u></a>	This function sets the text to be used for the object.
⇒	<a href="#"><u>EbAddChar</u></a>	This function inserts a character at the end of the text used by the object.
⇒	<a href="#"><u>EbDeleteChar</u></a>	This function removes a character at the end of the text used by the object.
⇒	<a href="#"><u>EbMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
⇒	<a href="#"><u>EbTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

## Macros

Name	Description
<a href="#"><u>EbGetText</u></a>	This macro returns the address of the current text string used for the object.

## Structures

---

Name	Description
<a href="#">EDITBOX</a>	Defines the parameters required for a Edit Box Object.

## Topics

---

Name	Description
<a href="#">Edit Box States</a>	List of Edit Box bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Edit Box States

[Macros](#) | [Edit Box](#)

List of Edit Box bit states.

## Macros

Name	Description
<a href="#">EB_CENTER_ALIGN</a>	Bit to indicate text is center aligned.
<a href="#">EB_DISABLED</a>	Bit for disabled state.
<a href="#">EB_DRAW</a>	Bit to indicate whole edit box must be redrawn.
<a href="#">EB_HIDE</a>	Bit to remove object from screen.
<a href="#">EB_FOCUSED</a>	Bit for focused state. Cursor caret will be drawn when <a href="#">EB_DRAW_CARET</a> is also set.
<a href="#">EB_RIGHT_ALIGN</a>	Bit to indicate text is left aligned.
<a href="#">EB_DRAW_CARET</a>	Bit to indicate the cursor caret will be drawn if <a href="#">EB_FOCUSED</a> state bit is set and erase when <a href="#">EB_FOCUSED</a> state bit is not set.
<a href="#">EB_CARET</a>	Bit to indicate the cursor caret will always be shown.

## Module

[Edit Box](#)

## Links

## Macros, Edit Box

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_CENTER\_ALIGN Macro

```
C  
#define EB_CENTER_ALIGN 0x0008 // Bit to indicate text
```

### Description

Bit to indicate text is center aligned.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_CENTER\\_ALIGN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_DISABLED Macro

---

```
C  
#define EB_DISABLED 0x0002 // Bit for disabled state
```

### Description

---

Bit for disabled state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_DRAW Macro

```
C  
#define EB_DRAW 0x4000 // Bit to indicate whole edit box must be redrawn.
```

### Description

Bit to indicate whole edit box must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_DRAW Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_HIDE Macro

---

```
C
#define EB_HIDE 0x8000 // Bit to remove object from
```

### Description

---

Bit to remove object from screen.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_FOCUSED Macro

---

C

```
#define EB_FOCUSED 0x0001 // Bit for focused state.
```

### Description

---

Bit for focused state. Cursor caret will be drawn when [EB\\_DRAW\\_CARET](#) is also set.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_RIGHT\_ALIGN Macro

---

C

```
#define EB_RIGHT_ALIGN 0x0004 // Bit to indicate te
```

### Description

---

Bit to indicate text is left aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_RIGHT\\_ALIGN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_DRAW\_CARET Macro

---

```
C
```

```
#define EB_DRAW_CARET 0x2000 // Bit to indicate the
```

### Description

---

Bit to indicate the cursor caret will be drawn if [EB\\_FOCUSED](#) state bit is set and erase when [EB\\_FOCUSED](#) state bit is not set.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_DRAW\\_CARET Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## EB\_CARET Macro

---

```
C  
#define EB_CARET 0x0010 // Bit to indicate the cursor
```

### Description

---

Bit to indicate the cursor caret will always be shown.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#) > [EB\\_CARET Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EbCreate Function

[Edit Box](#)

```
C
EDITBOX * EbCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    XCHAR * pText,
    WORD charMax,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [EDITBOX](#) object with the parameters given and initializes the default settings. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.

SHORT bottom	Bottom most position of the object.
WORD state	Sets the initial state of the object.
XCHAR * pText	Pointer to the text to be used.
WORD charMax	Defines the maximum number of characters in the edit box.
GOL_SCHEME * pScheme	Pointer to the style scheme.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

```
Copy Code  
  
#define ID_MYEDITBOX    101  
EDITBOX *pEb;  
  
pEb = EbCreate(ID_MYEDITBOX,    // ID  
               10,              // left  
               10,              // top  
               100,             // right  
               30,              // bottom  
               EB_DRAW,        // redraw after cre  
               NULL,           // no text yet
```

```
        4,                // display only fou
        pScheme);        // pointer to the s

if( pEb == NULL )
{
    // MEMORY ERROR. Object was not created.
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) >  
[EbCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EbDraw Function

[Edit Box](#)

```
C  
WORD EbDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pEb	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending

drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EbDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EbGetText Macro

## Edit Box

```
C
#define EbGetText(pEb) (pEb->pBuffer)
```

## Overview

This macro returns the address of the current text string used for the object.

## Input Parameters

Input Parameters	Description
pEb	Pointer to the object

## Returns

Returns pointer to the text string being used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EbGetText Macro](#)

# EbSetText Function

## Edit Box

```
C
void EbSetText(
    EDITBOX * pEb,
    XCHAR * pText
);
```

## Overview

This function sets the text to be used for the object.

## Input Parameters

Input Parameters	Description
EDITBOX * pEb	The pointer to the object whose text will be modified.
XCHAR * pText	Pointer to the text that will be used.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) >

## [EbSetText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EbAddChar Function

[Edit Box](#)

```
C
void EbAddChar(
    EDITBOX * pEb,
    XCHAR ch
);
```

## Overview

This function inserts a character at the end of the text used by the object.

## Input Parameters

Input Parameters	Description
EDITBOX * pEb	The pointer to the object whose text will be modified.
XCHAR ch	Character to be inserted.

## Returns

none

## Preconditions

none

## Side Effects

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EbAddChar Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EbDeleteChar Function

## Edit Box

```
C
void EbDeleteChar (
    EDITBOX * pEb
);
```

## Overview

This function removes a character at the end of the text used by the object.

## Input Parameters

Input Parameters	Description
EDITBOX * pEb	The pointer to the object to be modified.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EbDeleteChar Function](#)

[Contents](#) | [Index](#) | [Home](#)

# EbMsgDefault Function

[Edit Box](#)

```
C
void EbMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
EB_MSG_CHAR	Keyboard	Set <a href="#">EB_DRAW</a>	New character is added and Edit Box will be redrawn.
EB_MSG_DEL	Keyboard	Set <a href="#">EB_DRAW</a>	Last character is removed and Edit Box will be redrawn.

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message.
GOL_MSG * pMsg	The pointer to the GOL message.
pEb	The pointer to the object whose state will be

|| modified.

## Returns

---

none

## Preconditions

---

none

## Side Effects

---

none

## Example

---

Usage is similar to [BtnMsgDefault\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EbMsgDefault Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EbTranslateMsg Function

[Edit Box](#)

```
C
WORD EbTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
EB_MSG_CHAR	Keyboard	EVENT_CHARCODE	New character should added.
EB_MSG_DEL	Keyboard	EVENT_KEYPRESS	Last character should removed.
OBJ_MSG_INVALID	Any	Any	If message not affect object.

## Input Parameters

---

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pEb	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

---

Returns the translated message depending on the received GOL message:

- EB\_MSG\_CHAR – New character should be added.
- EB\_MSG\_DEL – Last character should be removed.
- OBJ\_MSG\_INVALID – Object is not affected.

## Preconditions

---

none

## Side Effects

---

none

## Example

---

Usage is similar to [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EbTranslateMsg Function](#)

# EDITBOX Structure

## Edit Box

C

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT textHeight;
    XCHAR * pBuffer;
    WORD charMax;
    WORD length;
} EDITBOX;
```

## Overview

Defines the parameters required for a Edit Box Object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT textHeight;	Pre-computed text height.
XCHAR * pBuffer;	Pointer to text buffer.
WORD charMax;	Maximum number of characters in the edit box.
WORD length;	Current text length.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [EDITBOX Structure](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Grid

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Grid supports Keyboard and Touchscreen inputs, replying to their events with the following messages:

1. GRID\_MSG\_TOUCHED
2. GRID\_MSG\_ITEM\_SELECTED
3. GRID\_MSG\_LEFT
4. GRID\_MSG\_RIGHT
5. GRID\_MSG\_UP
6. GRID\_MSG\_DOWN

See [GridTranslateMsg\(\)](#) and [GridMsgDefault\(\)](#) for details.

The Grid lines are drawn using the EmbossLitColor, the string drawn using the TextColor0 and the background is drawn using the CommonBkColor.

## Functions

	Name	Description
	<a href="#">GridCreate</a>	This function creates a <a href="#">GRID</a> 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.
	<a href="#">GridDraw</a>	This function renders the object on the screen using the current parameter

		<p>settings. 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.</p> <p>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.</p>
◆	<a href="#"><u>GridClearCellState</u></a>	This function clears the state of the cell (or Grid Item) specified by the column and row.
◆	<a href="#"><u>GridFreeItems</u></a>	This function removes all grid items for the given Grid and frees the memory used.
◆	<a href="#"><u>GridGetCell</u></a>	This function removes all grid items for the given Grid and frees the memory used.
◆	<a href="#"><u>GridSetCell</u></a>	This function sets the Grid Item state and data.
◆	<a href="#"><u>GridSetCellStyle</u></a>	This function sets the state of the Grid Item or cell.
◆	<a href="#"><u>GridSetFocus</u></a>	This function sets the focus of the specified Grid Item or cell.
◆	<a href="#"><u>GridMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:

	<a href="#"><u>GridTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>
-----------------------------------------------------------------------------------	-----------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Macros

Name	Description
<a href="#"><u>GridGetFocusX</u></a>	<p>This macro returns the x position of the focused cell.</p>
<a href="#"><u>GridGetFocusY</u></a>	<p>This macro returns the y position of the focused cell.</p>
<a href="#"><u>GRID_OUT_OF_BOUNDS</u></a>	<p>Status of an out of bounds cell <a href="#"><u>GridSetCell()</u></a> operation.</p>
<a href="#"><u>GRID_SUCCESS</u></a>	<p>Status of a successful <a href="#"><u>GridSetCell()</u></a> operation.</p>

## Structures

Name	Description
<a href="#"><u>GRID</u></a>	<p>Defines the parameters required for a grid Object. Clipping is not supported in grid object.</p>
<a href="#"><u>GRIDITEM</u></a>	<p>Defines the grid item.</p>

## Topics

<hr/>
-------

Name	Description
<a href="#">Grid States</a>	List of Grid bit states.
<a href="#">Grid Item States</a>	List of Grid Items bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Grid States

Macros | [Grid](#)

List of [Grid](#) bit states.

## Macros

Name	Description
<a href="#">GRID_FOCUSED</a>	Bit for focused state
<a href="#">GRID_DISABLED</a>	Bit for disabled state
<a href="#">GRID_SHOW_LINES</a>	Display grid lines
<a href="#">GRID_SHOW_FOCUS</a>	Highlight the focused cell.
<a href="#">GRID_SHOW_BORDER_ONLY</a>	Draw only the outside border of the grid.
<a href="#">GRID_SHOW_SEPARATORS_ONLY</a>	Draw only the lines between cells (like Tic-tac-toe)
<a href="#">GRID_DRAW_ITEMS</a>	Bit to indicate that at least one item must be redrawn, but not the entire grid.
<a href="#">GRID_DRAW_ALL</a>	Bit to indicate whole edit box must be redrawn
<a href="#">GRID_HIDE</a>	Bit to remove object from screen

## Module

[Grid](#)

## Links

---

[Macros](#), [Grid](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_FOCUSED Macro

---

C

```
#define GRID_FOCUSED 0x0001 // Bit for focused state
```

## Description

---

Bit for focused state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_DISABLED Macro

---

C

```
#define GRID_DISABLED 0x0002 // Bit for disabled sta
```

## Description

---

Bit for disabled state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_SHOW\_LINES Macro

C

```
#define GRID_SHOW_LINES 0x0004 // Display grid lines
```

## Description

Display grid lines

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_SHOW\\_LINES Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_SHOW\_FOCUS Macro

---

```
C
```

```
#define GRID_SHOW_FOCUS 0x0008 // Highlight the focus
```

## Description

---

Highlight the focused cell.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_SHOW\\_FOCUS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_SHOW\_BORDER\_ONLY Macro

C

```
#define GRID_SHOW_BORDER_ONLY 0x0010 // Draw only the outside border of the grid.
```

## Description

Draw only the outside border of the grid.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_SHOW\\_BORDER\\_ONLY Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_SHOW\_SEPARATORS\_ONLY Macro

---

C

```
#define GRID_SHOW_SEPARATORS_ONLY 0x0020 // Draw on.
```

## Description

---

Draw only the lines between cells (like Tic-tac-toe)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_SHOW\\_SEPARATORS\\_ONLY Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GRID\_DRAW\_ITEMS Macro

```
C  
#define GRID_DRAW_ITEMS 0x1000 // Bit to indicate t
```

### Description

Bit to indicate that at least one item must be redrawn, but not the entire grid.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_DRAW\\_ITEMS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GRID\_DRAW\_ALL Macro

---

C

```
#define GRID_DRAW_ALL 0x4000 // Bit to indicate who.
```

### Description

---

Bit to indicate whole edit box must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_DRAW\\_ALL Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_HIDE Macro

```
C  
#define GRID_HIDE 0x8000 // Bit to remove object from screen
```

## Description

Bit to remove object from screen

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#) > [GRID\\_HIDE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Grid Item States

[Macros](#) | [Grid](#)

List of [Grid](#) Items bit states.

## Macros

Name	Description
<a href="#">GRIDITEM_SELECTED</a>	The cell is selected.
<a href="#">GRIDITEM_IS_TEXT</a>	The grid item is a text string.
<a href="#">GRIDITEM_IS_BITMAP</a>	The grid item is a bitmap.
<a href="#">GRIDITEM_TEXTBOTTOM</a>	Bit to indicate text is top aligned.
<a href="#">GRIDITEM_TEXTLEFT</a>	Text in the cell is left aligned.
<a href="#">GRIDITEM_TEXTRIGHT</a>	Text in the cell is right aligned.
<a href="#">GRIDITEM_TEXTTOP</a>	Bit to indicate text is bottom aligned.
<a href="#">GRIDITEM_DRAW</a>	Draw this cell

## Module

[Grid](#)

## Links

[Macros](#), [Grid](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#)

[Contents](#) | [Index](#) | [Home](#)

## GRIDITEM\_SELECTED Macro

---

C

```
#define GRIDITEM_SELECTED 0x0001 // The cell is selected
```

### Description

---

The cell is selected.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_SELECTED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GRIDITEM\_IS\_TEXT Macro

---

C

```
#define GRIDITEM_IS_TEXT 0x0000 // The grid item is
```

### Description

---

The grid item is a text string.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_IS\\_TEXT Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRIDITEM\_IS\_BITMAP Macro

C

```
#define GRIDITEM_IS_BITMAP 0x0008 // The grid item .
```

## Description

The grid item is a bitmap.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_IS\\_BITMAP Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRIDITEM\_TEXTBOTTOM Macro

C

```
#define GRIDITEM_TEXTBOTTOM 0x0040 // Bit to indica
```

## Description

Bit to indicate text is top aligned.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_TEXTBOTTOM Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRIDITEM\_TEXTLEFT Macro

---

C

```
#define GRIDITEM_TEXTLEFT 0x0020 // Text in the cel.
```

## Description

---

Text in the cell is left aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_TEXTLEFT Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRIDITEM\_TEXTRIGHT Macro

```
C
```

```
#define GRIDITEM_TEXTRIGHT 0x0010 // Text in the ce.
```

## Description

Text in the cell is right aligned.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_TEXTRIGHT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GRIDITEM\_TEXTTOP Macro

C

```
#define GRIDITEM_TEXTTOP 0x0080 // Bit to indicate
```

### Description

Bit to indicate text is bottom aligned.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_TEXTTOP Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRIDITEM\_DRAW Macro

C

```
#define GRIDITEM_DRAW 0x0100 // Draw this cell
```

## Description

Draw this cell

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#) > [GRIDITEM\\_DRAW Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridCreate Function

## Grid

C

```
GRID * GridCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    WORD state,  
    SHORT numColumns,  
    SHORT numRows,  
    SHORT cellwidth,  
    SHORT cellHeight,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a **GRID** 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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.

SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the object.
WORD state	Sets the initial state of the object.
SHORT numColumns	Sets the number of columns for the grid.
SHORT numRows	Sets the number of rows for the grid.
SHORT cellWidth	Sets the width of each cell of the grid.
SHORT cellHeight	Sets the height of each cell of the grid.
GOL_SCHEME * pScheme	Pointer to the style scheme used for the object. Set to NULL if default style scheme is used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridCreate Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridDraw Function

## Grid

```
C  
WORD GridDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pGb	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridClearCellState Function

## Grid

```
C
void GridClearCellState(
    GRID * pGrid,
    SHORT column,
    SHORT row,
    WORD state
);
```

## Overview

This function clears the state of the cell (or [Grid](#) Item) specified by the column and row.

## Input Parameters

Input Parameters	Description
GRID * pGrid	Pointer to the object.
SHORT column	column index of the cell
SHORT row	row index of the cell
atate	specifies the state to be cleared. See <a href="#">Grid</a> Item State.

## Returns

none.

## Preconditions

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridClearCellState Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridGetFocusX Macro

## Grid

```
C
#define GridGetFocusX(pGrid) pGrid->focusX
```

## Overview

This macro returns the x position of the focused cell.

## Input Parameters

Input Parameters	Description
pGrid	Pointer to the object.

## Returns

Returns the x position of the focused cell.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridGetFocusX Macro](#)

# GridGetFocusY Macro

## Grid

```
C
#define GridGetFocusY(pGrid) pGrid->focusY
```

## Overview

This macro returns the y position of the focused cell.

## Input Parameters

Input Parameters	Description
pGrid	Pointer to the object.

## Returns

Returns the y position of the focused cell.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridGetFocusY Macro](#)

# GRID\_OUT\_OF\_BOUNDS Macro

## Grid

```
C
#define GRID_OUT_OF_BOUNDS 0x0001 // Status of an out of bounds cell
```

## Description

Status of an out of bounds cell [GridSetCell\(\)](#) operation.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GRID\\_OUT\\_OF\\_BOUNDS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID\_SUCCESS Macro

## Grid

```
C
#define GRID_SUCCESS 0x0000 // Status of a success
```

## Description

Status of a successful [GridSetCell\(\)](#) operation.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GRID\\_SUCCESS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridFreeItems Function

## Grid

```
C
void GridFreeItems(
    void * pObjj
);
```

## Overview

This function removes all grid items for the given [Grid](#) and frees the memory used.

## Input Parameters

Input Parameters	Description
pGrid	The pointer to the <a href="#">Grid</a> object.

## Returns

none.

## Preconditions

Object must be created before this function is called.

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridFreeItems Function](#)

[Contents](#) | [Index](#) | [Home](#)

# GridGetCell Function

## Grid

```
C
void * GridGetCell(
    GRID * pGrid,
    SHORT column,
    SHORT row,
    WORD * cellType
);
```

## Overview

This function removes all grid items for the given [Grid](#) and frees the memory used.

## Input Parameters

Input Parameters	Description
GRID * pGrid	The pointer to the <a href="#">Grid</a> object.
SHORT column	the column index of the cell
SHORT row	the row index of the cell
WORD * cellType	pointer that will receive the type of grid item or cell ( <a href="#">GRIDITEM_IS_TEXT</a> or <a href="#">GRIDITEM_IS_BITMAP</a> ).

## Returns

Returns a pointer to the grid item or cell data.

## Preconditions

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridGetCell Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridSetCell Function

## Grid

```
C
WORD GridSetCell(
    GRID * pGrid,
    SHORT column,
    SHORT row,
    WORD state,
    void * data
);
```

## Overview

This function sets the [Grid](#) Item state and data.

## Input Parameters

Input Parameters	Description
GRID * pGrid	The pointer to the <a href="#">Grid</a> object.
SHORT column	the column index of the cell
SHORT row	the row index of the cell
WORD state	sets the state of the <a href="#">Grid</a> Item specified.
void * data	pointer to the data used for the <a href="#">Grid</a> Item.

## Returns

Returns the status of the operation

- [GRID\\_SUCCESS](#) - if the set succeeded
- [GRID\\_OUT\\_OF\\_BOUNDS](#) - if the row and column given results in

an out of bounds location.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridSetCell Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridSetCellState Function

## Grid

```
C
void GridSetCellState(
    GRID * pGrid,
    SHORT column,
    SHORT row,
    WORD state
);
```

## Overview

This function sets the state of the [Grid](#) Item or cell.

## Input Parameters

Input Parameters	Description
GRID * pGrid	The pointer to the <a href="#">Grid</a> object.
SHORT column	the column index of the cell
SHORT row	the row index of the cell
WORD state	sets the state of the <a href="#">Grid</a> Item specified.

## Returns

none.

## Preconditions

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridSetCellState Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridSetFocus Function

## Grid

```
C
void GridSetFocus(
    GRID * pGrid,
    SHORT column,
    SHORT row
);
```

## Overview

This function sets the focus of the specified [Grid](#) Item or cell.

## Input Parameters

Input Parameters	Description
GRID * pGrid	The pointer to the <a href="#">Grid</a> object.
SHORT column	the column index of the cell
SHORT row	the row index of the cell

## Returns

none.

## Preconditions

Object must be created before this function is called.

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridSetFocus Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridMsgDefault Function

## Grid

C

```
void GridMsgDefault(  
    WORD translatedMsg,  
    void * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit
GRID_MSG_TOUCHED	Touch Screen	none
GRID_MSG_ITEM_SELECTED	Keyboard	Set <a href="#">GRIDITEM_SELECTE</a> <a href="#">GRID_DRAW_ITEMS</a>
GRID_MSG_UP	Keyboard	Set <a href="#">GRIDITEM_DRAW</a>

		<a href="#">GRID_DRAW_ITEMS</a>
GRID_MSG_DOWN	Keyboard	Set <a href="#">GRIDITEM_DRAW</a>
		<a href="#">GRID_DRAW_ITEMS</a>
GRID_MSG_LEFT	Keyboard	Set <a href="#">GRIDITEM_DRAW</a>
		<a href="#">GRID_DRAW_ITEMS</a>
GRID_MSG_RIGHT	Keyboard	Set <a href="#">GRIDITEM_DRAW</a>
		<a href="#">GRID_DRAW_ITEMS</a>

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message.
GOL_MSG * pMsg	The pointer to the GOL message.
pGrid	The pointer to the object whose state will be

|| modified.

## Returns

---

none

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridMsgDefault Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GridTranslateMsg Function

## Grid

```
C  
WORD GridTranslateMsg(  
    void * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Set/Clear State Bit
GRID_MSG_TOUCHED	Touch Screen	none
GRID_MSG_ITEM_SELECTED	Keyboard	EVENT_KEYSCAN
GRID_MSG_UP	Keyboard	EVENT_KEYSCAN

GRID_MSG_DOWN	Keyboard	EVENT_KEYSCAN
GRID_MSG_LEFT	Keyboard	EVENT_KEYSCAN
GRID_MSG_RIGHT	Keyboard	EVENT_KEYSCAN
OBJ_MSG_INVALID	Any	Any

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pGrid	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

---

Returns the translated message depending on the received GOL message:

- GRID\_MSG\_TOUCHED - when the grid object is touched.
- GRID\_MSG\_ITEM\_SELECTED – when key scan [SCAN\\_SPACE\\_PRESSED](#) or [SCAN\\_CR\\_PRESSED](#) are detected.
- GRID\_MSG\_UP – when key scan [SCAN\\_UP\\_PRESSED](#) is detected.
- GRID\_MSG\_DOWN – when key scan [SCAN\\_DOWN\\_PRESSED](#) is detected.
- GRID\_MSG\_LEFT – when key scan [SCAN\\_LEFT\\_PRESSED](#) is detected.
- GRID\_MSG\_RIGHT – when key scan [SCAN\\_RIGHT\\_PRESSED](#) is detected.
- OBJ\_MSG\_INVALID – [Button](#) is not affected

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GridTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRID Structure

## Grid

```
C
typedef struct {
    OBJ_HEADER hdr;
    SHORT numColumns;
    SHORT numRows;
    SHORT cellHeight;
    SHORT cellWidth;
    SHORT focusX;
    SHORT focusY;
    GRIDITEM * gridObjects;
} GRID;
```

## Overview

Defines the parameters required for a grid Object. Clipping is not supported in grid object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT numColumns;	Number of columns drawn for the <a href="#">Grid</a> .
SHORT numRows;	Number of rows drawn for the <a href="#">Grid</a> .
SHORT cellHeight;	The height of each cell in pixels.
SHORT cellWidth;	The width of each cell in pixels.
SHORT focusX;	The x position of the cell to be focused.

SHORT focusY;	The y position of the cell to be focused.
GRIDITEM * gridObjects;	The pointer to grid items

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GRID Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRIDITEM Structure

## Grid

```
C
typedef struct {
    void * data;
    WORD status;
} GRIDITEM;
```

## Overview

Defines the grid item.

## Members

Members	Description
void * data;	Indicates if the <a href="#">Grid</a> Item is type <a href="#">GRIDITEM_IS_TEXT</a> or <a href="#">GRIDITEM_IS_BITMAP</a>
WORD status;	indicates the status of the <a href="#">Grid</a> Item

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [GRIDITEM Structure](#)

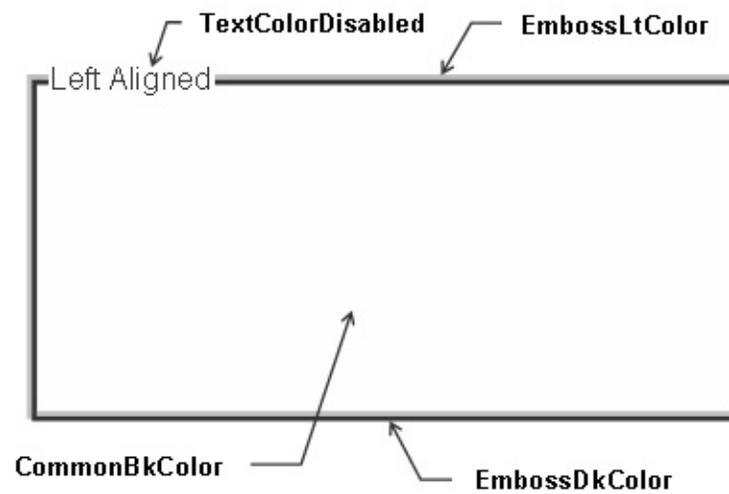
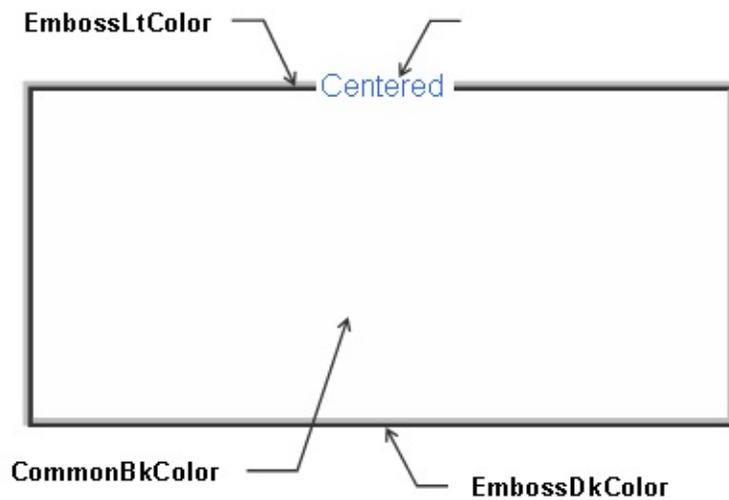
# Group Box

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Group Box supports only Touchscreen inputs, replying to their events with the message:

GB\_MSG\_SELECTED - when the touch is within the dimension of the object.

The Group box object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



## Functions

	Name	Description
◆	<a href="#"><u>GbCreate</u></a>	This function creates a <b>GROUPBOX</b> 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.
◆	<a href="#"><u>GbDraw</u></a>	This function renders the object on the screen using the current parameter

		<p>settings. 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.</p> <p>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.</p>
◆	<a href="#"><u>GbSetText</u></a>	This function sets the text used by passing the pointer to the static string.
◆	<a href="#"><u>GbTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

## Macros

Name	Description
<a href="#"><u>GbGetText</u></a>	This macro returns the location of the text used.

## Structures

Name	Description
<a href="#"><u>GROUPBOX</u></a>	Defines the parameters required for a group box Object. The textwidth and textHeight is not checked with the actual dimension of the object. Clipping is not supported in group

box object. It is possible for the text to exceed the dimension of the Object.

## Topics

---

Name	Description
<a href="#">Group Box States</a>	List of Group Box bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Group Box States

[Macros](#) | [Group Box](#)

List of Group Box bit states.

## Macros

Name	Description
<a href="#">GB_CENTER_ALIGN</a>	Bit to indicate text is center aligned
<a href="#">GB_DISABLED</a>	Bit for disabled state
<a href="#">GB_DRAW</a>	Bit to indicate group box must be redrawn
<a href="#">GB_HIDE</a>	Bit to remove object from screen
<a href="#">GB_RIGHT_ALIGN</a>	Bit to indicate text is right aligned

## Module

[Group Box](#)

## Links

[Macros](#), [Group Box](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#)

## GB\_CENTER\_ALIGN Macro

```
C
#define GB_CENTER_ALIGN 0x0008 // Bit to indicate text is center aligned
```

### Description

Bit to indicate text is center aligned

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#) > [GB\\_CENTER\\_ALIGN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GB\_DISABLED Macro

---

```
C  
#define GB_DISABLED 0x0002 // Bit for disabled state
```

### Description

---

Bit for disabled state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#) > [GB\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GB\_DRAW Macro

---

```
C
#define GB_DRAW 0x4000           // Bit to indicate g
```

### Description

---

Bit to indicate group box must be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#) > [GB\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GB\_HIDE Macro

```
C
#define GB_HIDE 0x8000 // Bit to remove object
```

### Description

Bit to remove object from screen

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#) > [GB\\_HIDE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GB\_RIGHT\_ALIGN Macro

```
C
#define GB_RIGHT_ALIGN 0x0004 // Bit to indicate text
```

### Description

Bit to indicate text is right aligned

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#) > [GB\\_RIGHT\\_ALIGN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GbCreate Function

## Group Box

```
C
GROUPBOX * GbCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    XCHAR * pText,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the object.

WORD state	Sets the initial state of the object.
XCHAR * pText	The pointer to the text used for the group box. Length of string must be checked not to exceed the object's width. Clipping is not supported for the text of this object.
GOL_SCHEME * pScheme	Pointer to the style scheme used for the object. Set to NULL if default style scheme is used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
GOL_SCHEME *pScheme;
GROUPBOX *groupbox[2];
WORD state;

pScheme = GOLCreateScheme();
state = GB_DRAW | GB_RIGHT_ALIGN;
groupbox[0] = GbCreate( 10, 14, 48, 152, 122,
                      state, "Power", scheme);
if (groupbox[0] == NULL)
    return 0;
state = GB_DRAW;
```

```
groupbox[1] = GbCreate( 11, 160, 48, 298, 122,  
                      state, "Pressure", scheme);  
if (groupbox[1] == NULL)  
    return 0;  
  
while(!GbDraw(groupbox[0]));  
while(!GbDraw(groupbox[1]));  
return 1;
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) >  
[GbCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GbDraw Function

## Group Box

```
C  
WORD GbDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pGb	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [GbCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [GbDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GbGetText Macro

## Group Box

C

```
#define GbGetText(pB) pGb->pText
```

## Overview

This macro returns the location of the text used.

## Input Parameters

Input Parameters	Description
pGb	Pointer to the object.

## Returns

Returns the address of the text string used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [GbGetText Macro](#)

# GbSetText Function

## Group Box

```
C
void GbSetText(
    GROUPBOX * pGb,
    XCHAR * pText
);
```

## Overview

This function sets the text used by passing the pointer to the static string.

## Input Parameters

Input Parameters	Description
GROUPBOX * pGb	the pointer to the object whose state will be modified.
XCHAR * pText	pointer to the text that will be used.

## Returns

none

## Preconditions

The style scheme used for the object MUST be initialized with a valid font. If font is not valid, textWidth and textHeight parameter of [GROUPBOX](#) will be undefined.

## Side Effects

Modifies the object width and height depending on the selected string width and font height.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [GbSetText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GbTranslateMsg Function

## Group Box

```
C
WORD GbTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

Translated Message	Input Source	Events	Description
GB_MSG_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If event occurs at the x, y position fall in the area of the group box.
OBJ_MSG_INVALID	Any	Any	If the message does not affect the object.

## Input Parameters

--	--

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pGb	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- GB\_MSG\_SELECTED – Group Box is selected
- OBJ\_MSG\_INVALID – Group Box is not affected

## Preconditions

none

## Side Effects

none

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [GbTranslateMsg Function](#)

# GROUPBOX Structure

## Group Box

C

```
typedef struct {  
    OBJ_HEADER hdr;  
    SHORT textWidth;  
    SHORT textHeight;  
    XCHAR * pText;  
} GROUPBOX;
```

## Overview

Defines the parameters required for a group box Object. The textwidth and textHeight is not checked with the actual dimension of the object. Clipping is not supported in group box object. It is possible for the text to exceed the dimension of the Object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT textWidth;	Pre-computed text width.
SHORT textHeight;	Pre-computed text height.
XCHAR * pText;	Text string used.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [GROUPBOX Structure](#)

[Contents](#) | [Index](#) | [Home](#)

# List Box

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

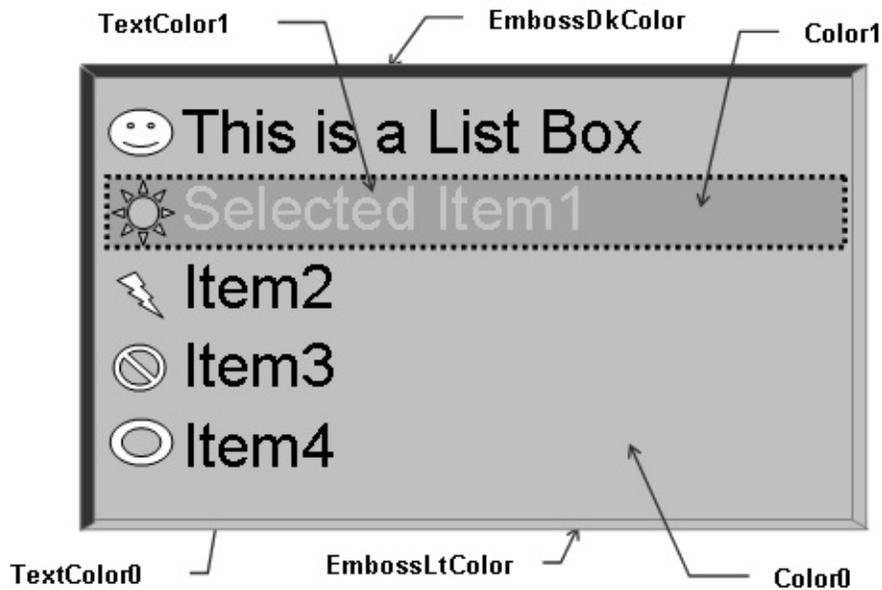
List Box supports both Touchscreen and Keyboard inputs, replying to their events with the following messages:

LB\_MSG\_TOUCHSCREEN – Item is selected using touch screen

LB\_MSG\_MOVE – Focus is moved to the next item depending on the key pressed (UP or DOWN key).

LB\_MSG\_SEL – Selection is set to the currently focused item.

The List Box Object is rendered using the assigned style scheme. The following figure illustrates the color assignments. Icons can be added to each item when adding items to the list using [LbAddItem\(\)](#).



## Functions

	Name	Description
◆	<a href="#"><u>LbCreate</u></a>	This function creates a <b>LISTBOX</b> 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.
◆	<a href="#"><u>LbDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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 or items drawn in the visible window of the list box is dependent on

		<p>the alignment set.</p> <p>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.</p>
◆	<a href="#"><u>LbAddItem</u></a>	<p>This function allocates memory for the <a href="#"><u>LISTITEM</u></a> and adds it to the list box. The newly created <a href="#"><u>LISTITEM</u></a> will store the location of pText, pBitmap and other parameters describing the added item.</p>
◆	<a href="#"><u>LbDelItem</u></a>	<p>This function removes an item from the list box and frees the memory used.</p>
◆	<a href="#"><u>LbChangeSel</u></a>	<p>This function changes the selection status of an item in the list box. If the item is currently selected, it resets the selection. If the item is currently not selected it is set to be selected.</p>
◆	<a href="#"><u>LbGetSel</u></a>	<p>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 list. It returns the pointer to the first selected item found or NULL if there are no items selected.</p>
◆	<a href="#"><u>LbGetFocusedItem</u></a>	<p>This function returns the index of the focused item in the list box.</p>
◆	<a href="#"><u>LbSetFocusedItem</u></a>	<p>This function sets the focus for the item with the given index.</p>
◆	<a href="#"><u>LbDelItemsList</u></a>	<p>This function removes all items from the list box and frees the memory used.</p>

	<a href="#"><u>LbMsgDefault</u></a>	<p>This function performs the actual state change based on the translated message given. The following state changes are supported:</p>
	<a href="#"><u>LbTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

## Macros

Name	Description
<a href="#"><u>LbGetItemList</u></a>	<p>This function returns the pointer to the current item list used in the list box.</p>
<a href="#"><u>LbSetSel</u></a>	<p>This macro sets the selection status of an item to selected.</p>
<a href="#"><u>LbGetCount</u></a>	<p>This macro returns the number of items in the list box.</p>
<a href="#"><u>LbGetVisibleCount</u></a>	<p>This macro returns the number of items visible in the list box window.</p>
<a href="#"><u>LbSetBitmap</u></a>	<p>This macro sets the bitmap used in the item.</p>
<a href="#"><u>LbGetBitmap</u></a>	<p>This macro returns the location of the currently used bitmap for the item.</p>

## Structures

Name	Description
<a href="#">LISTBOX</a>	Defines the parameters required for a list box Object.
<a href="#">LISTITEM</a>	Defines the parameters required for a list item used in list box.

## Topics

---

Name	Description
<a href="#">List Box States</a>	List of List Box bit states.
<a href="#">List Item Status</a>	List of Items status.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# List Box States

[Macros](#) | [List Box](#)

List of List Box bit states.

## Macros

Name	Description
<a href="#">LB_RIGHT_ALIGN</a>	Bit to indicate text is left aligned
<a href="#">LB_SINGLE_SEL</a>	Bit to indicate the only item can be selected
<a href="#">LB_CENTER_ALIGN</a>	Bit to indicate text is center aligned
<a href="#">LB_DISABLED</a>	Bit for disabled state
<a href="#">LB_DRAW</a>	Bit to indicate whole edit box must be redrawn
<a href="#">LB_DRAW_FOCUS</a>	Bit to indicate whole edit box must be redrawn
<a href="#">LB_DRAW_ITEMS</a>	Bit to indicate whole edit box must be redrawn
<a href="#">LB_FOCUSED</a>	Bit for focused state
<a href="#">LB_HIDE</a>	Bit to remove object from screen

## Module

[List Box](#)

## Links

[Macros](#), [List Box](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_RIGHT\_ALIGN Macro

---

C

```
#define LB_RIGHT_ALIGN 0x0004 // Bit to indicate text
```

### Description

---

Bit to indicate text is left aligned

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_RIGHT\\_ALIGN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_SINGLE\_SEL Macro

---

C

```
#define LB_SINGLE_SEL 0x0010 // Bit to indicate the
```

### Description

---

Bit to indicate the only item can be selected

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_SINGLE\\_SEL Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_CENTER\_ALIGN Macro

```
C
#define LB_CENTER_ALIGN 0x0008 // Bit to indicate text is center aligned
```

### Description

Bit to indicate text is center aligned

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_CENTER\\_ALIGN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_DISABLED Macro

---

C

```
#define LB_DISABLED 0x0002 // Bit for disabled state
```

### Description

---

Bit for disabled state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_DRAW Macro

---

```
C
#define LB_DRAW 0x4000 // Bit to indicate whole edit
```

### Description

---

Bit to indicate whole edit box must be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_DRAW\_FOCUS Macro

---

C

```
#define LB_DRAW_FOCUS 0x2000 // Bit to indicate who.
```

### Description

---

Bit to indicate whole edit box must be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_DRAW\\_FOCUS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_DRAW\_ITEMS Macro

---

C

```
#define LB_DRAW_ITEMS 0x1000 // Bit to indicate who.
```

### Description

---

Bit to indicate whole edit box must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_DRAW\\_ITEMS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_FOCUSED Macro

---

```
C
#define LB_FOCUSED 0x0001 // Bit for focused state
```

### Description

---

Bit for focused state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_HIDE Macro

---

```
C
#define LB_HIDE 0x8000 // Bit to remove object from
```

### Description

---

Bit to remove object from screen

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#) > [LB\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# List Item Status

[Macros](#) | [List Box](#)

List of Items status.

## Macros

Name	Description
<a href="#">LB_STS_SELECTED</a>	Item is selected.
<a href="#">LB_STS_REDRAW</a>	Item is to be redrawn.

## Module

[List Box](#)

## Links

[Macros](#), [List Box](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Item Status](#)

## LB\_STS\_SELECTED Macro

---

C

```
#define LB_STS_SELECTED 0x0001 // Item is selected.
```

### Description

---

Item is selected.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Item Status](#) > [LB\\_STS\\_SELECTED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LB\_STS\_REDRAW Macro

---

```
C
#define LB_STS_REDRAW 0x0002 // Item is to be redra
```

### Description

---

Item is to be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Item Status](#) > [LB\\_STS\\_REDRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbCreate Function

## List Box

```
C
LISTBOX * LbCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    XCHAR * pText,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the Object.

WORD state	Sets the initial state of the object.
XCHAR * pText	Pointer to the initialization text for the items.
GOL_SCHEME * pScheme	Pointer to the style scheme.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
#define LISTBOX_ID 10

const XCHAR ItemList[] = "Line1n" "Line2n";

GOL_SCHEME *pScheme;
LISTBOX *pLb;
XCHAR *pTemp;
WORD state, counter;

    pScheme = GOLCreateScheme();
    state = LB_DRAW;

    // create an empty listbox with default style s
    pLb = LbCreate( LISTBOX_ID,          // ID numbe
                  10,10,150,200,      // dimensio
```

```

        state,                // initial
        NULL,                 // set item
        NULL);               // use defa
// check if Listbox was created
if (pLb == NULL)
    return 0;

// create the list of items to be placed in the
// 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 assi
    // LISTITEM pointer.
    if(NULL == LbAddItem(pLb, NULL, pTemp, NULL)
        break;
    while((unsigned XCHAR)*pTemp++ > (unsigned
    if(*pTemp == 0)
        break;
    counter++;
}

```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbDraw Function

## List Box

```
C  
WORD LbDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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 or items drawn in the visible window of the list box is dependent on the alignment 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.

## Input Parameters

Input Parameters	Description
pLb	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbGetItemList Macro

## List Box

```
C
#define LbGetItemList(pLb) ((LISTITEM *)((LISTBOX *)|
```

## Overview

This function returns the pointer to the current item list used in the list box.

## Input Parameters

Input Parameters	Description
pLb	The pointer to the list box object.

## Returns

Returns the pointer to the [LISTITEM](#) used in the list box.

## Preconditions

none

## Side Effects

none

## Example

See [LbAddItem\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbGetItemList Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbAddItem Function

## List Box

```
C
LISTITEM * LbAddItem(
    LISTBOX * pLb,
    LISTITEM * pPrevItem,
    XCHAR * pText,
    void * pBitmap,
    WORD status,
    WORD data
);
```

## Overview

This function allocates memory for the [LISTITEM](#) and adds it to the list box. The newly created [LISTITEM](#) will store the location of pText, pBitmap and other parameters describing the added item.

## Input Parameters

Input Parameters	Description
LISTBOX * pLb	The pointer to the list box object.
LISTITEM * 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.
XCHAR * 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.

void * pBitmap	Pointer to the bitmap for the item. Bitmap must persist in memory for as long as it is referenced by the an item in the list box.
WORD status	This parameter specifies if the item being added will be selected or redrawn ( <a href="#">LB_STS_SELECTED</a> or <a href="#">LB_STS_REDRAW</a> ). Refer to <a href="#">LISTITEM</a> structure for details.
WORD data	User assigned data associated with the item.

## Returns

Return a pointer to the item created, NULL if the operation was not successful.

## Preconditions

none

## Side Effects

none

## Example

```
Copy Code  
const XCHAR ItemList[] = "Line1n" "Line2n" "Line3n"  
  
extern BITMAP_FLASH myIcon;  
LISTBOX *pLb;  
LISTITEM *pItem, *pItemList;  
XCHAR *pTemp;  
  
// Assume that pLb is pointing to an existing list
```

```

// that is empty (no list).

// Create the list of the list box

// Initialize this to NULL to indicate that items w
// at the end of the list if the list exist on the
// start a new list if the list box is empty.
pItem = NULL;
pTemp = ItemList;
pItem = LbAddItem(pLb, pItem, pTemp, NULL, LB_STS_S
if(pItem == NULL)
    return 0;
LbSetBitmap(pItem, &myIcon);

// Adjust pTemp to point to the next line
while((unsigned XCHAR)*pTemp++ > (unsigned XCHAR)31

// add the next item
pItem = LbAddItem(pLb, pItem, pTemp, NULL, 0, 2)
if(pItem == NULL)
    return 0;
LbSetBitmap(pItem, &myIcon);

// Adjust pTemp to point to the next line
while((unsigned XCHAR)*pTemp++ > (unsigned XCHAR)31

// this time insert the next item after the first i
pItem = LbGetItemList(pLb);
pItem = LbAddItem(pLb, pItem, pTemp, NULL, 0, 3)
if(pItem == NULL)
    return 0;
LbSetBitmap(pItem, &myIcon);

```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbAddItem Function](#)

[Contents](#) | [Index](#) | [Home](#)

# LbDelItem Function

## List Box

```
C
void LbDelItem(
    LISTBOX * pLb,
    LISTITEM * pItem
);
```

## Overview

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

## Input Parameters

Input Parameters	Description
LISTBOX * pLb	The pointer to the list box object.
LISTITEM * pItem	The pointer to the item that will be removed.

## Returns

none

## Preconditions

none

## Side Effects

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbDelItem Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbChangeSel Function

## List Box

```
C
void LbChangeSel(
    LISTBOX * pLb,
    LISTITEM * pItem
);
```

## Overview

This function changes the selection status of an item in the list box. If the item is currently selected, it resets the selection. If the item is currently not selected it is set to be selected.

## Input Parameters

Input Parameters	Description
LISTBOX * pLb	The pointer to the list box object.
LISTITEM * pItem	The pointer to the item the selection status will be changed.

## Returns

none

## Preconditions

none

## Side Effects

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbChangeSel Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbSetSel Macro

## List Box

C

```
#define LbSetSel(pLb, pItem) \  
    if(!(pItem->status & LB_STS_SELECTED)) \  
        LbChangeSel((LISTBOX *)pLb, pItem);
```

## Overview

This macro sets the selection status of an item to selected.

## Input Parameters

Input Parameters	Description
pLb	The pointer to the list box object.
pItem	The pointer to the item the selection status will be set.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbSetSel Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbGetSel Function

## List Box

```
C
LISTITEM * LbGetSel(
    LISTBOX * pLb,
    LISTITEM * pFromItem
);
```

## Overview

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 list. It returns the pointer to the first selected item found or NULL if there are no items selected.

## Input Parameters

Input Parameters	Description
LISTBOX * pLb	The pointer to the list box object.
LISTITEM * pFromItem	The pointer to the item the search must start from, if the pointer is NULL the search begins from the start of the items list.

## Returns

pointer to the selected item, NULL if there are no items selected

## Preconditions

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbGetSel Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbGetFocusedItem Function

## List Box

```
C
SHORT LbGetFocusedItem(
    LISTBOX * pLb
);
```

## Overview

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

## Input Parameters

Input Parameters	Description
LISTBOX * pLb	The pointer to the list box object.

## Returns

Returns the index of the focused item in the list box.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbGetFocusedItem Function](#)

# LbSetFocusedItem Function

## List Box

```
C
void LbSetFocusedItem(
    LISTBOX * pLb,
    SHORT index
);
```

## Overview

This function sets the focus for the item with the given index.

## Input Parameters

Input Parameters	Description
LISTBOX * pLb	The pointer to the list box object.
SHORT index	The index number of the item to be focused. First item on the list is always indexed 0.

## Returns

none.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) >

## LbSetFocusedItem Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbGetCount Macro

## List Box

```
C
#define LbGetCount(pLb) ((LISTBOX *)pLb)->itemsNumber
```

## Overview

This macro returns the number of items in the list box.

## Input Parameters

Input Parameters	Description
pLb	The pointer to the list box object.

## Returns

The number of items the list box contains.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbGetCount Macro](#)

# LbGetVisibleCount Macro

## List Box

C

```
#define LbGetVisibleCount(pLb) \  
    (  
        (((LISTBOX *)pLb)->hdr.bottom - ((LISTBOX *)pLb)->textHeight)  
    )
```

## Overview

This macro returns the number of items visible in the list box window.

## Input Parameters

Input Parameters	Description
pLb	The pointer to the list box object.

## Returns

The number of items visible in the list box window.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbGetVisibleCount Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbSetBitmap Macro

## List Box

```
C
#define LbSetBitmap(pItem, pBtmap) ((LISTITEM *)pItem->Data->Bitmap = pBtmap)
```

## Overview

This macro sets the bitmap used in the item.

## Input Parameters

Input Parameters	Description
pItem	Pointer to the item.
pBtmap	Pointer to the bitmap to be used.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

See [LbAddItem\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) >

## LbSetBitmap Macro

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbGetBitmap Macro

## List Box

```
C
```

```
#define LbGetBitmap(pItem) ((LISTITEM *)pItem)->pBitmap
```

## Overview

This macro returns the location of the currently used bitmap for the item.

## Input Parameters

Input Parameters	Description
pItem	Pointer to the list item.

## Returns

Returns the pointer to the current bitmap used.

## Preconditions

none

## Side Effects

none

## Example

```
Copy Code  
// Assume pLb is initialized to an existing list box  
LISTITEM *pItem;  
void *pBitmap;
```

```
pItem = LbGetItemList(pLb);  
pBitmap = LbGetBitmap(pItem);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) >  
[LbGetBitmap Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LbDelItemsList Function

## List Box

```
C
void LbDelItemsList(
    void * pObj
);
```

## Overview

This function removes all items from the list box and frees the memory used.

## Input Parameters

Input Parameters	Description
pLb	The pointer to the list box object.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbDelItemsList Function](#)

[Contents](#) | [Index](#) | [Home](#)

# LbMsgDefault Function

## List Box

C

```
void LbMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Des
LB_MSG_TOUCHSCREEN	Touch Screen	Set <a href="#">LB_FOCUSED</a> ,	If fo the <a href="#">LB_</a> be <a href="#">LB_</a> draw forc
		Set <a href="#">LB_DRAW_FOCUS</a>	the redr focu
		Set <a href="#">LB_DRAW_ITEMS</a>	List redr sele
LB_MSG_MOVE	KeyBoard	Set	List

		<a href="#">LB_DRAW_ITEMS</a>	redr on c
LB_MSG_SEL	KeyBoard	Set <a href="#">LB_DRAW_ITEMS</a>	List redr sele curr focu

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message
GOL_MSG * pMsg	The pointer to the GOL message.
pB	The pointer to the object whose state will be modified.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LbMsgDefault Function](#)

# LbTranslateMsg Function

## List Box

```
C
WORD LbTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
LB_MSG_TOUCHSCREEN	Touch Screen	Any	Item selected on touch screen
LB_MSG_MOVE	Keyboard	EVENT_KEYSCAN	Focus moved to next object (depends on the press of a key)
LB_MSG_SEL	Keyboard	EVENT_KEYSCAN	LB_I - Selected

			set curre focu:
OBJ_MSG_INVALID	Any	Any	If mes: not obje

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pLB	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- LB\_MSG\_TOUCHSCREEN – Item is selected using touch screen.
- LB\_MSG\_MOVE – Focus is moved to the next item depending on the key pressed (UP or DOWN key).
- LB\_MSG\_SEL – Selection is set to the currently focused item.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) >  
[LbTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LISTBOX Structure

## List Box

```
C
typedef struct {
    OBJ_HEADER hdr;
    LISTITEM * pItemList;
    LISTITEM * pFocusItem;
    WORD itemsNumber;
    SHORT scrollY;
    SHORT textHeight;
} LISTBOX;
```

## Overview

Defines the parameters required for a list box Object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
LISTITEM * pItemList;	Pointer to the list of items.
LISTITEM * pFocusItem;	Pointer to the focused item.
WORD itemsNumber;	Number of items in the list box.
SHORT scrollY;	<a href="#">Scroll</a> displacement for the list.
SHORT textHeight;	Pre-computed text height.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) >  
[LISTBOX Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LISTITEM Structure

## List Box

```
C
typedef struct {
    void * pPrevItem;
    void * pNextItem;
    WORD status;
    XCHAR * pText;
    void * pBitmap;
    WORD data;
} LISTITEM;
```

## Overview

Defines the parameters required for a list item used in list box.

## Members

Members	Description
void * pPrevItem;	Pointer to the next item
void * pNextItem;	Pointer to the next item
WORD status;	Specifies the status of the item.
XCHAR * pText;	Pointer to the text for the item
void * pBitmap;	Pointer to the bitmap
WORD data;	Some data associated with the item

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [LISTITEM Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Meter

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

There are three meter types that you can draw:

1. MTR\_WHOLE\_TYPE
2. MTR\_HALF\_TYPE
3. MTR\_QUARTER\_TYPE

It supports only System inputs, replying to the event EVENT\_SET with the message: MTR\_MSG\_SET (see [MtrTranslateMsg\(\)](#) for details). This action ID means that the message contains the new value of the meter on the parameter 2 of the message.

The Meter Object is rendered using the assigned style scheme, value range colors (see [MtrSetScaleColors\(\)](#) for details) and compile time settings. The following figure illustrates the assignments for a MTR\_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.

## Functions

	Name	Description
◆	<a href="#">MtrCreate</a>	This function creates a <b>METER</b> 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.
	<a href="#"><u>MtrDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>Depending on the defined settings, value of the meter will displayed or hidden. Displaying the value will require a little bit more rendering time depending on the size of the meter and font used.</p> <p>When rendering objects of the same type, each object must be rendered completely before the rendering of the... <a href="#"><u>more</u></a></p>
	<a href="#"><u>MtrSetVal</u></a>	<p>This function sets the value of the meter to the passed newVal. newVal is checked to be in the minValue-maxValue range inclusive. If newVal is not in the range, minValue maxValue is assigned depending on the given newVal if less than minValue or above maxValue.</p>
	<a href="#"><u>MtrMsgDefault</u></a>	<p>This function performs the actual state change based on the translated message given. Meter value is set based on parameter 2 of the message given. The following state changes are supported:</p>
	<a href="#"><u>MtrTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

## Macros

Name	Description
<a href="#"><u>MtrGetVal</u></a>	This macro returns the current value of the meter. Value is always in the minValue-maxValue range inclusive.
<a href="#"><u>MtrDecVal</u></a>	This macro is used to directly decrement the value.
<a href="#"><u>MtrIncVal</u></a>	This macro is used to directly increment the value.
<a href="#"><u>MtrSetScaleColors</u></a>	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. Limitation is that color settings are set to the following angles: Color Boundaries Type Whole Type Half Type Quarter <a href="#"><u>Arc</u></a> 6 225 to 180 not used not used <a href="#"><u>Arc</u></a> 5 179 to 135 179 to 135 not used <a href="#"><u>Arc</u></a> 4 134 to 90 134 to 90 not used <a href="#"><u>Arc</u></a> 3 89 to 45 89 to 45 <a href="#"><u>Arc</u></a> 2 44 to 0 44... <a href="#"><u>more</u></a>
<a href="#"><u>MtrSetTitleFont</u></a>	This function sets the font of title.
<a href="#"><u>MtrSetValueFont</u></a>	This function sets the font of value.
<a href="#"><u>METER_TYPE</u></a>	This is a compile time setting to select the type if meter shape. There are three types: <ul style="list-style-type: none"><li>• MTR_WHOLE_TYPE - Meter drawn with 6 octants used.</li><li>• MTR_HALF_TYPE - Meter drawn with</li></ul>

	<p>semi circle shape.</p> <ul style="list-style-type: none"> <li>• MTR_QUARTER_TYPE - Meter drawn with quarter circle shape.</li> </ul> <p>Set only one value at a time. This is done to save code space. User can define the colors of the arcs for each type.</p> <p>MTR_WHOLE_TYPE will use all the arc colors (arcColor1 - arcColor6)</p> <p>MTR_HALF_TYPE will use arc colors (arcColor5, arcColor4, arcColor3, arcColor2)</p> <p>MTR_QUARTER_TYPE will use arc colors (arcColor3, arcColor2) Set the meter type in Meter.h file and... <a href="#">more</a></p>
<a href="#">MTR_ACCURACY</a>	<p>Sets the meter accuracy to one decimal places when displaying the values.</p> <p>Application must multiply the minValue, maxValue and values passed to the widget by RESOLUTION.</p>

## Structures

Name	Description
<a href="#">METER</a>	<p>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.</p>

## Topics

Name	Description
<a href="#">Meter States</a>	<p>List of Meter bit states.</p>

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Meter States

[Macros](#) | [Meter](#)

List of [Meter](#) bit states.

## Macros

Name	Description
<a href="#">MTR_DISABLED</a>	Bit for disabled state.
<a href="#">MTR_DRAW</a>	Bit to indicate object must be redrawn.
<a href="#">MTR_HIDE</a>	Bit to indicate object must be removed from screen.
<a href="#">MTR_RING</a>	Bit for ring type, scales are drawn over the ring default is only scales drawn.
<a href="#">MTR_DRAW_UPDATE</a>	Bit to indicate an update only.

## Module

[Meter](#)

## Links

[Macros](#), [Meter](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#)

## MTR\_DISABLED Macro

---

C

```
#define MTR_DISABLED 0x0002 // Bit for disabled
```

### Description

---

Bit for disabled state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#) > [MTR\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## MTR\_DRAW Macro

```
C
#define MTR_DRAW 0x4000 // Bit to indicate object
```

### Description

Bit to indicate object must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#) > [MTR\\_DRAW Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## MTR\_HIDE Macro

```
C
#define MTR_HIDE 0x8000 // Bit to indicate object
```

### Description

Bit to indicate object must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#) > [MTR\\_HIDE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## MTR\_RING Macro

```
C  
#define MTR_RING 0x0004 // Bit for ring type, scales are drawn over the ring default is only scales drawn.
```

### Description

Bit for ring type, scales are drawn over the ring default is only scales drawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#) > [MTR\\_RING Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## MTR\_DRAW\_UPDATE Macro

C

```
#define MTR_DRAW_UPDATE 0x1000 // Bit to indica
```

### Description

Bit to indicate an update only.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#) > [MTR\\_DRAW\\_UPDATE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MtrCreate Function

## Meter

C

```
METER * MtrCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    WORD state,  
    SHORT value,  
    SHORT minValue,  
    SHORT maxValue,  
    void * pTitleFont,  
    void * pValueFont,  
    XCHAR * pText,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a **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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the object.

SHORT top	Top most position of the object.
SHORT right	Right most position of the object.
SHORT bottom	Bottom most position of the object.
WORD state	Sets the initial state of the object.
SHORT value	Initial value set to the meter.
SHORT minValue	The minimum value the meter will display.
SHORT maxValue	The maximum value the meter will display.
void * pTitleFont	Pointer to the font used for the Title.
XCHAR * pText	Pointer to the text label of the meter.
GOL_SCHEME * pScheme	Pointer to the font used for the Value. Pointer to the style scheme used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

```
#define ID_METER 101
```

[Copy Code](#)

```

extern const FONT_FLASH GOLMediumFont;           // medi
extern const FONT_FLASH GOLSmallFont;           // smal

GOL_SCHEME *pMeterScheme;
METER *pMtr;

    pMeterScheme = GOLCreateScheme();

    pMtr = MtrCreate(
        ID_METER,                                // assign ID
        30, 50, 150, 180,                        // set dimensio
        MTR_DRAW|MTR_RING,                       // draw object
        0,                                        // set initial
        0, 100,                                   // set minimum
        (void*)&GOLMediumFont,                  // set title fo
        (void*)&GOLSmallFont,                   // set value fo
        "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 hav
MtrSetScaleColors(pMtr, WHITE, WHITE, WHITE, GR

// use GOLDraw() to draw the meter and all othe
while(!GOLDraw());
// OR to draw the meter manually use this:
//while(!MtrDraw(pMtr);

```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MtrDraw Function

## Meter

```
C  
WORD MtrDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

Depending on the defined settings, value of the meter will displayed or hidden. Displaying the value will require a little bit more rendering time depending on the size of the meter and font used.

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.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [MtrCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MtrSetVal Function

## Meter

```
C
void MtrSetVal(
    METER * pMtr,
    SHORT newVal
);
```

## Overview

This function sets the value of the meter to the passed newVal. newVal is checked to be in the minValue-maxValue range inclusive. If newVal is not in the range, minValue maxValue is assigned depending on the given newVal if less than minValue or above maxValue.

## Input Parameters

Input Parameters	Description
METER * pMtr	The pointer to the object.
SHORT newVal	New value to be set for the <a href="#">Meter</a> .

## Returns

none

## Preconditions

none

## Side Effects

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrSetVal Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MtrGetVal Macro

## Meter

```
C
```

```
#define MtrGetVal(pMtr) ((pMtr)->value)
```

## Overview

This macro returns the current value of the meter. Value is always in the minValue-maxValue range inclusive.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object.

## Returns

Returns current value of the meter.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrGetVal Macro](#)

# MtrDecVal Macro

## Meter

```
C
```

```
#define MtrDecVal(pMtr, deltaValue) MtrSetVal(pMtr,
```

## Overview

This macro is used to directly decrement the value.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object.
deltaValue	Number to be subtracted to the current <a href="#">Meter</a> value.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrDecVal Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# MtrIncVal Macro

## Meter

```
C
```

```
#define MtrIncVal(pMtr, deltaValue) MtrSetVal(pMtr,
```

## Overview

This macro is used to directly increment the value.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object.
deltaValue	Number to be added to the current <a href="#">Meter</a> value.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrIncVal Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# MtrSetScaleColors Macro

## Meter

```
C
#define MtrSetScaleColors(pMtr, arc1, arc2, arc3, arc4, arc5, arc6)
{
    pMtr->arcColor6 = arc6;
    pMtr->arcColor5 = arc5;
    pMtr->arcColor4 = arc4;
    pMtr->arcColor3 = arc3;
    pMtr->arcColor2 = arc2;
    pMtr->arcColor1 = arc1;
}
```

## Overview

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.

Limitation is that 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.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object.
arc1	color for arc 1.

arc2	color for arc 2.
arc3	color for arc 3.
arc4	color for arc 4.
arc5	color for arc 5.
arc6	color for arc 6.

---

## Returns

none

---

## Preconditions

The object must be created (using [MtrCreate\(\)](#)) before a call to this macro is performed.

---

## Side Effects

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrSetScaleColors Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MtrSetTitleFont Macro

## Meter

```
C
#define MtrSetTitleFont(pMtr, pNewFont) (((METER *)pMtr->titleFont = pNewFont)
```

## Overview

This function sets the font of title.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object.
pNewFont	Pointer to the new font used for the title.

## Returns

N/A

## Preconditions

Font must be created before this function is called.

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrSetTitleFont Macro](#)

# MtrSetValueFont Macro

## Meter

```
C
#define MtrSetValueFont(pMtr, pNewFont) (((METER *)pMtr->value->font = pNewFont)
```

## Overview

This function sets the font of value.

## Input Parameters

Input Parameters	Description
pMtr	Pointer to the object.
pNewFont	Pointer to the new font used for the value.

## Returns

N/A

## Preconditions

Font must be created before this function is called.

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrSetValueFont Macro](#)

# METER\_TYPE Macro

## Meter

```
C
```

```
#define METER_TYPE MTR_WHOLE_TYPE
```

## Overview

This is a compile time setting to select the type of meter shape. There are three types:

- MTR\_WHOLE\_TYPE - [Meter](#) drawn with 6 octants used.
- MTR\_HALF\_TYPE - [Meter](#) drawn with semi circle shape.
- MTR\_QUARTER\_TYPE - [Meter](#) drawn with quarter circle shape.

Set only one value at a time. This is done to save code space. User can define the colors of the arcs for each type.

MTR\_WHOLE\_TYPE will use all the arc colors (arcColor1 - arcColor6) MTR\_HALF\_TYPE will use arc colors (arcColor5, arcColor4, arcColor3, arcColor2) MTR\_QUARTER\_TYPE will use arc colors (arcColor3, arcColor2) Set the meter type in Meter.h file and arc colors using [MtrSetScaleColors](#)(pMtr, arc1, arc2, arc3, arc4, arc5, arc6) macro.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [METER\\_TYPE Macro](#)

# MTR\_ACCURACY Macro

## Meter

```
C
#define MTR_ACCURACY 0x0008 // Sets the meter a
```

## Description

Sets the meter accuracy to one decimal places when displaying the values. Application must multiply the minValue, maxValue and values passed to the widget by RESOLUTION.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MTR\\_ACCURACY Macro](#)

# MtrMsgDefault Function

## Meter

C

```
void MtrMsgDefault(  
    WORD translatedMsg,  
    void * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

This function performs the actual state change based on the translated message given. [Meter](#) value is set based on parameter 2 of the message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
MTR_MSG_SET	System	Set <a href="#">MTR_DRAW_UPDATE</a>	<a href="#">Meter</a> will be redrawn to update the needle position and value displayed.

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message.
GOL_MSG * pMsg	The pointer to the GOL message.

pMtr

The pointer to the object whose state will be modified.

## Returns

---

none

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrMsgDefault Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MtrTranslateMsg Function

## Meter

```
C  
WORD MtrTranslateMsg(  
    void * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
MTR_MSG_SET	System	EVENT_SET	If event set occurs and the meter ID is sent in parameter 1.
OBJ_MSG_INVALID	Any	Any	If the message did not affect the object.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.

pMtr

The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

---

Returns the translated message depending on the received GOL message:

- MTR\_MSG\_SET - [Meter](#) ID is given in parameter 1 for a TYPE\_SYSTEM message.
- OBJ\_MSG\_INVALID - [Meter](#) is not affected.

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [MtrTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# METER Structure

## Meter

```
C
typedef struct {
    OBJ_HEADER hdr;
    XCHAR * pText;
    SHORT value;
    SHORT minValue;
    SHORT maxValue;
    SHORT xCenter;
    SHORT yCenter;
    SHORT radius;
    SHORT xPos;
    SHORT yPos;
    GFX_COLOR arcColor6;
    GFX_COLOR arcColor5;
    GFX_COLOR arcColor4;
    GFX_COLOR arcColor3;
    GFX_COLOR arcColor2;
    GFX_COLOR arcColor1;
    void * pTitleFont;
    void * pValueFont;
} METER;
```

## Overview

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.

## Members

Members	Description

OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
XCHAR * pText;	The text label of the meter.
SHORT value;	Current value of the meter.
SHORT minValue;	minimum value the meter can display
SHORT maxValue;	maximum value the meter can display (range is maxValue - minValue)
SHORT xCenter;	The x coordinate center position. This is computed automatically.
SHORT yCenter;	The y coordinate center position. This is computed automatically.
SHORT radius;	Radius of the meter, also defines the needle length.
SHORT xPos;	The current x position of the needle. This is computed automatically.
SHORT yPos;	The current y position of the needle. This is computed automatically.
GFX_COLOR arcColor6;	<a href="#">Arc</a> 6 color parameter.
GFX_COLOR arcColor5;	<a href="#">Arc</a> 5 color parameter
GFX_COLOR arcColor4;	<a href="#">Arc</a> 4 color parameter
GFX_COLOR arcColor3;	<a href="#">Arc</a> 3 color parameter

GFX_COLOR arcColor2;	<a href="#">Arc</a> 2 color parameter
GFX_COLOR arcColor1;	<a href="#">Arc</a> 1 color parameter
void * pTitleFont;	Pointer to the font used in the title of the meter
void * pValueFont;	Pointer to the font used in the current reading (if displayed) of the meter

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [METER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)



		determined by the left, top, right and bottom parameters. The colors used are dependent on the state of 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.
	<a href="#">PictTranslateMsg</a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event accepted by the <a href="#">PICTURE</a> Object.

## Macros

Name	Description
<a href="#">PictSetBitmap</a>	This macro sets the bitmap used in the object.
<a href="#">PictGetBitmap</a>	This macro returns the pointer to the bitmap used in the object.
<a href="#">PictGetScale</a>	This macro returns the current scale factor used to render the bitmap.
<a href="#">PictSetScale</a>	This macro sets the scale factor used to render the bitmap used in the object.

## Structures

Name	Description
<a href="#">PICTURE</a>	The structure contains data for picture

control

## Topics

---

Name	Description
<a href="#">Picture States</a>	List of Picture Control bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Picture States

Macros | [Picture Control](#)

List of [Picture](#) Control bit states.

## Macros

Name	Description
<a href="#">PICT_DISABLED</a>	Bit to indicate <a href="#">Picture</a> is in a disabled state.
<a href="#">PICT_DRAW</a>	Bit to indicate <a href="#">Picture</a> will be redrawn.
<a href="#">PICT_FRAME</a>	Bit to indicate <a href="#">Picture</a> has a frame.
<a href="#">PICT_HIDE</a>	Bit to indicate <a href="#">Picture</a> must be hidden.

## Module

[Picture Control](#)

## Links

[Macros, Picture Control](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [Picture States](#)

## PICT\_DISABLED Macro

C

```
#define PICT_DISABLED 0x0002 // Bit to indicate Pic
```

### Description

Bit to indicate [Picture](#) is in a disabled state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [Picture States](#) > [PICT\\_DISABLED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PICT\_DRAW Macro

---

C

```
#define PICT_DRAW 0x4000 // Bit to indicate Picture
```

### Description

---

Bit to indicate [Picture](#) will be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [Picture States](#) > [PICT\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PICT\_FRAME Macro

```
C  
#define PICT_FRAME 0x0004 // Bit to indicate Picture
```

### Description

Bit to indicate [Picture](#) has a frame.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [Picture States](#) > [PICT\\_FRAME Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PICT\_HIDE Macro

---

C

```
#define PICT_HIDE 0x8000 // Bit to indicate Picture
```

### Description

---

Bit to indicate [Picture](#) must be hidden.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [Picture States](#) > [PICT\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PictCreate Function

## Picture Control

C

```
PICTURE * PictCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    WORD state,  
    char scale,  
    void * pBitmap,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a **PICTURE** 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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the object.

WORD state	Sets the initial state of the object.
char scale	Sets the scale factor used to render the bitmap.
void * pBitmap	Pointer to the bitmap that will be used.
GOL_SCHEME * pScheme	Pointer to the style scheme
radius	Radius of the rounded edge.

---

## Returns

Returns the pointer to the object created

---

## Preconditions

none

---

## Side Effects

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PictDraw Function

## Picture Control

```
C  
WORD PictDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

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.

## Input Parameters

Input Parameters	Description
pPict	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

Object must be created before this function is called.

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PictSetBitmap Macro

## Picture Control

C

```
#define PictSetBitmap(pPict, pBtmap) ((PICTURE*)pPic
```

## Overview

This macro sets the bitmap used in the object.

## Input Parameters

Input Parameters	Description
pPict	Pointer to the object
pBtMap	Pointer to the bitmap to be used

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictSetBitmap Macro](#)

# PictGetBitmap Macro

## Picture Control

```
C
#define PictGetBitmap(pPict) ((PICTURE*)pPict)->pBitm
```

## Overview

This macro returns the pointer to the bitmap used in the object.

## Input Parameters

Input Parameters	Description
pPict	Pointer to the object

## Returns

Returns the pointer to the bitmap used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictGetBitmap Macro](#)

# PictGetScale Macro

## Picture Control

```
C
```

```
#define PictGetScale(pPict) pPict->scale
```

## Overview

This macro returns the current scale factor used to render the bitmap.

## Input Parameters

Input Parameters	Description
pPict	Pointer to the object

## Returns

Returns the current scale factor used to display the bitmap.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictGetScale Macro](#)

# PictSetScale Macro

## Picture Control

```
C
```

```
#define PictSetScale(pPict, scl) pPict->scale = scl
```

## Overview

This macro sets the scale factor used to render the bitmap used in the object.

## Input Parameters

Input Parameters	Description
pPict	Pointer to the object
scl	The scale factor that will be used to display the bitmap.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictSetScale Macro](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PictTranslateMsg Function

## Picture Control

```
C
WORD PictTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event accepted by the [PICTURE](#) Object.

Translated Message	Input Source	Events	Description
PICT_MSG_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If event occurs at the position of in the area of picture.
OBJ_MSG_INVALID	Any	Any	If message not affect object.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pPict	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- PICT\_MSG\_SELECTED – [Picture](#) is touched.
- OBJ\_MSG\_INVALID – [Picture](#) is not affected

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PictTranslateMsg Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PICTURE Structure

## Picture Control

**C**

```
typedef struct {
    OBJ_HEADER hdr;
    char scale;
    void * pBitmap;
    PUTIMAGE_PARAM partial;
} PICTURE;
```

## Overview

The structure contains data for picture control

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
char scale;	Scale factor for the bitmap
void * pBitmap;	Pointer to the bitmap
PUTIMAGE_PARAM partial;	structure containing parital image data

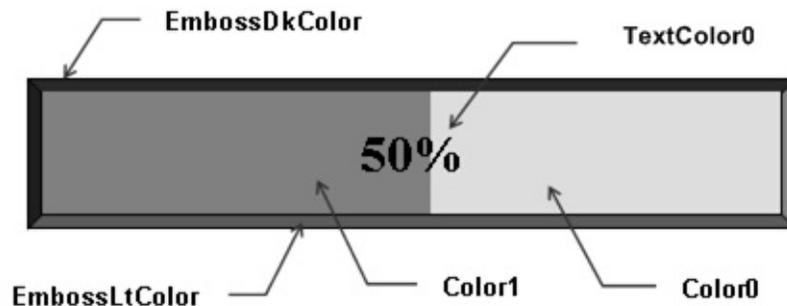
[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [PICTURE Structure](#)

# Progress Bar

Functions | Macros | Structures | Topics

Progress [Bar](#) supports only Touchscreen inputs, replying to their events with the message: PB\_MSG\_SELECTED.

The Progress [Bar](#) Object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



**CommonBkColor** – used to hide/remove the progress bar from the screen.

## Functions

	Name	Description
	<a href="#">PbCreate</a>	This function creates a <a href="#">PROGRESSBAR</a> 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.

	<u><a href="#">PbDraw</a></u>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<u><a href="#">PbSetRange</a></u>	<p>This function sets the range of the progress bar. Calling this function also resets the position equal to the new range value.</p>
	<u><a href="#">PbSetPos</a></u>	<p>This function sets the position of the progress bar. Position should be in the given range inclusive.</p>
	<u><a href="#">PbTranslateMsg</a></u>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.</p>

## Macros

Name	Description
<u><a href="#">PbGetRange</a></u>	<p>This macro returns the current range of the progress bar.</p>
<u><a href="#">PbGetPos</a></u>	<p>This macro returns the current progress bar</p>

position.

## Structures

---

Name	Description
<a href="#">PROGRESSBAR</a>	The structure contains data for the progress bar

## Topics

---

Name	Description
<a href="#">Progress Bar States</a>	List of Progress Bar bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Progress Bar States

Macros | [Progress Bar](#)

List of [Progress Bar](#) bit states.

## Macros

Name	Description
<a href="#">PB_DISABLED</a>	Bit to indicate Progress <a href="#">Bar</a> is in a disabled state.
<a href="#">PB_DRAW</a>	Bit to indicate Progress <a href="#">Bar</a> must be redrawn.
<a href="#">PB_DRAW_BAR</a>	Bit to indicate Progress <a href="#">Bar</a> must be redrawn.
<a href="#">PB_HIDE</a>	Bit to indicate Progress <a href="#">Bar</a> must be hidden.
<a href="#">PB_VERTICAL</a>	Bit for orientation (0 - horizontal, 1 - vertical)

## Module

[Progress Bar](#)

## Links

[Macros](#), [Progress Bar](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [Progress Bar States](#)

[Contents](#) | [Index](#) | [Home](#)

## PB\_DISABLED Macro

C

```
#define PB_DISABLED 0x0002 // Bit to indicate Progress
```

### Description

Bit to indicate Progress [Bar](#) is in a disabled state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [Progress Bar States](#) > [PB\\_DISABLED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PB\_DRAW Macro

```
C
#define PB_DRAW 0x4000 // Bit to indicate Progress I
```

### Description

Bit to indicate Progress [Bar](#) must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [Progress Bar States](#) > [PB\\_DRAW Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PB\_DRAW\_BAR Macro

---

C

```
#define PB_DRAW_BAR 0x2000 // Bit to indicate Progress Bar
```

### Description

---

Bit to indicate Progress **Bar** must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [Progress Bar States](#) > [PB\\_DRAW\\_BAR Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PB\_HIDE Macro

---

```
C
#define PB_HIDE 0x8000 // Bit to indicate Progress I
```

### Description

---

Bit to indicate Progress [Bar](#) must be hidden.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [Progress Bar States](#) > [PB\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PB\_VERTICAL Macro

C

```
#define PB_VERTICAL 0x0004 // Bit for orientation
```

### Description

Bit for orientation (0 - horizontal, 1 - vertical)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [Progress Bar States](#) > [PB\\_VERTICAL Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PbCreate Function

## Progress Bar

```
C
PROGRESSBAR * PbCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    WORD pos,
    WORD range,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a **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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the Object.

WORD state	Sets the initial state of the Object.
WORD pos	Defines the initial position of the progress.
WORD range	This specifies the maximum value of the progress bar when the progress bar is at 100% position.
GOL_SCHEME * pScheme	Pointer to the style scheme used for the object. Set to NULL if default style scheme is used.

## Returns

Returns the pointer to the object created

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```

PROGRESSBAR *pPBar;
void CreateProgressBar(){
    pPBar = PbCreate(ID_PROGRESSBAR1,    // ID
                    50,90,270,140,      // dimensi
                    PB_DRAW,            // Draw th
                    25,                  // positio
                    50,                  // set the
                    NULL);              // use def
    while( !PbDraw(pPBar) );

```



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [PbCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PbDraw Function

## Progress Bar

```
C  
WORD PbDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pPb	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [PbCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [PbDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PbSetRange Function

## Progress Bar

```
C
void PbSetRange(
    PROGRESSBAR * pPb,
    WORD range
);
```

## Overview

This function sets the range of the progress bar. Calling this function also resets the position equal to the new range value.

## Input Parameters

Input Parameters	Description
PROGRESSBAR * pPb	Pointer to the object

## Returns

none.

## Preconditions

none

## Side Effects

Sets the position equal to the new range.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [PbSetRange Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PbGetRange Macro

## Progress Bar

```
C
#define PbGetRange(pPb) (pPb->range)
```

## Overview

This macro returns the current range of the progress bar.

## Input Parameters

Input Parameters	Description
pPb	Pointer to the object

## Returns

Returns the range value.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#) > [PbGetRange Macro](#)

# PbSetPos Function

## Progress Bar

```
C
void PbSetPos(
    PROGRESSBAR * pPb,
    WORD position
);
```

## Overview

This function sets the position of the progress bar. Position should be in the given range inclusive.

## Input Parameters

Input Parameters	Description
PROGRESSBAR * pPb	Pointer to the object
WORD position	New position.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
PROGRESSBAR *pPb;
BYTE  direction = 1;

// this code increments and decrements the prog
// assume progress bar was created and initiali
while (1) {
    if(direction) {
        if(pPb ->pos == pPb ->range)
            direction = 0;
        else
            PbSetPos(pPb,PbGetPos(pPb)+1);
    } else {
        if(pPb ->pos == 0)
            direction = 1;
        else
            PbSetPos(pPb,PbGetPos(pPb)-1);
    }
}
```

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [PbSetPos Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PbGetPos Macro

## Progress Bar

```
C
#define PbGetPos(pPb) pPb->pos
```

## Overview

This macro returns the current progress bar position.

## Input Parameters

Input Parameters	Description
pPb	Pointer to the object

## Returns

Returns the progress bar position.

## Preconditions

none

## Side Effects

none

## Example

See [PbSetPos\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [PbGetPos Macro](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PbTranslateMsg Function

## Progress Bar

```
C
WORD PbTranslateMsg(
    PROGRESSBAR * pPb,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

Translated Message	Input Source	Events	Description
PB_MSG_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE, EVENT_MOVE	If event occurs and the x, y position fall in the area of the progress bar.
OBJ_MSG_INVALID	Any	Any	If the message does not affect the object.

## Input Parameters

Input Parameters	Description
PROGRESSBAR * pPb	The pointer to the object where the message will be evaluated to check if the message will affect the object.
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.

## Returns

Returns the translated message depending on the received GOL message:

- PB\_MSG\_SELECTED – Progress [Bar](#) is selected.
- OBJ\_MSG\_INVALID – Progress [Bar](#) is not affected

## Preconditions

none

## Side Effects

none

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#) > [PbTranslateMsg Function](#)

# PROGRESSBAR Structure

## Progress Bar

C

```
typedef struct {  
    OBJ_HEADER hdr;  
    WORD pos;  
    WORD prevPos;  
    WORD range;  
} PROGRESSBAR;
```

## Overview

The structure contains data for the progress bar

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
WORD pos;	Current progress position.
WORD prevPos;	Previous progress position.
WORD range;	Sets the range of the object.

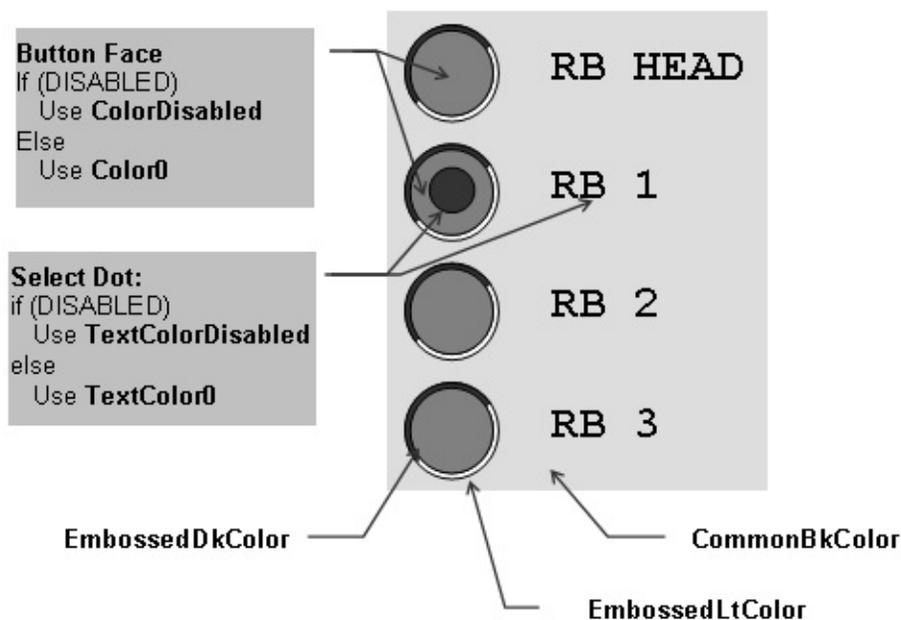
[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)  
> [PROGRESSBAR Structure](#)

# Radio Button

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Radio [Button](#) supports both Keyboard and Touchscreen inputs, replying to their events with the message: `RB_MSG_CHECKED`.

The Radio [Button](#) Object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



## Functions

	Name	Description
	<a href="#">RbCreate</a>	This function creates a <a href="#">RADIOBUTTON</a>

		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.
◆	<a href="#"><u>RbDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
◆	<a href="#"><u>RbGetCheck</u></a>	This function returns the ID of the currently checked Radio <a href="#"><u>Button</u></a> in the group.
◆	<a href="#"><u>RbSetCheck</u></a>	This function sets the Radio <a href="#"><u>Button</u></a> with the given ID to its checked state.
◆	<a href="#"><u>RbSetText</u></a>	This function sets the string used for the object.
◆	<a href="#"><u>RbMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
◆	<a href="#"><u>RbTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

---

## Macros

---

Name	Description
<a href="#">RbGetText</a>	This macro returns the address of the current text string used for the object.

---

## Structures

---

Name	Description
<a href="#">RADIOBUTTON</a>	the structure contains data for the <a href="#">Radio Button</a>

---

## Topics

---

Name	Description
<a href="#">Radio Button States</a>	List of Radio Button bit states.

---

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Radio Button States

[Macros](#) | [Radio Button](#)

List of [Radio Button](#) bit states.

## Macros

Name	Description
<a href="#">RB_CHECKED</a>	Bit to indicate Radio <a href="#">Button</a> is checked.
<a href="#">RB_DISABLED</a>	Bit for disabled state.
<a href="#">RB_DRAW</a>	Bit to indicate whole Radio <a href="#">Button</a> must be redrawn.
<a href="#">RB_DRAW_CHECK</a>	Bit to indicate check mark should be redrawn.
<a href="#">RB_DRAW_FOCUS</a>	Bit to indicate focus must be redrawn.
<a href="#">RB_FOCUSED</a>	Bit for focused state.
<a href="#">RB_GROUP</a>	Bit to indicate the first Radio <a href="#">Button</a> in the group.
<a href="#">RB_HIDE</a>	Bit to indicate that button must be removed from screen.

## Module

[Radio Button](#)

## Links

[Macros](#), [Radio Button](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_CHECKED Macro

---

C

```
#define RB_CHECKED 0x0004 // Bit to indicate Radio I
```

### Description

---

Bit to indicate Radio [Button](#) is checked.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_CHECKED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_DISABLED Macro

---

C

```
#define RB_DISABLED 0x0002 // Bit for disabled state
```

### Description

---

Bit for disabled state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_DRAW Macro

---

C

```
#define RB_DRAW 0x4000 // Bit to indicate whole Rad.
```

### Description

---

Bit to indicate whole Radio [Button](#) must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_DRAW\_CHECK Macro

---

C

```
#define RB_DRAW_CHECK 0x1000 // Bit to indicate che
```

### Description

---

Bit to indicate check mark should be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_DRAW\\_CHECK Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_DRAW\_FOCUS Macro

---

C

```
#define RB_DRAW_FOCUS 0x2000 // Bit to indicate focus
```

### Description

---

Bit to indicate focus must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_DRAW\\_FOCUS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_FOCUSED Macro

---

C

```
#define RB_FOCUSED 0x0001 // Bit for focused state.
```

### Description

---

Bit for focused state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_GROUP Macro

---

C

```
#define RB_GROUP 0x0008 // Bit to indicate the first
```

### Description

---

Bit to indicate the first Radio [Button](#) in the group.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_GROUP Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RB\_HIDE Macro

---

```
C  
#define RB_HIDE 0x8000 // Bit to indicate that button
```

### Description

---

Bit to indicate that button must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#) > [RB\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbCreate Function

## Radio Button

C

```
RADIOBUTTON * RbCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    WORD state,  
    XCHAR * pText,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object
SHORT bottom	Bottom most position of the object

WORD state	Sets the initial state of the object pText – The pointer to the text used for the <a href="#">Radio Button</a> .
GOL_SCHEME * pScheme	Pointer to the style scheme used.

## Returns

Returns the pointer to the object created

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
GOL_SCHEME *pScheme;  
RADIOBUTTON *pRb[3];  
  
pScheme = GOLCreateScheme();  
pRb[0] = RbCreate(ID_RADIOBUTTON1,  
                 255, 40, 310, 80,  
                 RB_DRAW|RB_GROUP|RB_CHECKED,  
  
                 "RB1",  
                 pScheme);  
  
pRb[1] = RbCreate(ID_RADIOBUTTON2,  
                 255, 85, 310, 125,  
                 RB_DRAW,
```

```
        "RB2",  
        pScheme);  
  
pRb[2] = RbCreate(ID_RADIOBUTTON3,  
                255, 130, 310, 170,  
                RB_DRAW,  
  
                "RB3",  
                pScheme);  
  
while( !RbDraw(pRb[0]) );  
while( !RbDraw(pRb[1]) );  
while( !RbDraw(pRb[2]) );
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RbCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbDraw Function

## Radio Button

```
C  
WORD RbDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pB	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [RbCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RbDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbGetCheck Function

## Radio Button

```
C  
WORD RbGetCheck(  
    RADIOBUTTON * pRb  
);
```

## Overview

This function returns the ID of the currently checked [Radio Button](#) in the group.

## Input Parameters

Input Parameters	Description
RADIOBUTTON * pRb	Pointer to the <a href="#">Radio Button</a> in the group.

## Returns

Returns the ID of the selected button in the group. It returns -1 if there is no object checked.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```

GOL_SCHEME *pScheme;
RADIOBUTTON *pRb[3];
SHORT ID;

    pScheme = GOLCreateScheme();
    pRb[0] = RbCreate(ID_RADIOBUTTON1,
                    255, 40, 310, 80,
                    RB_DRAW|RB_GROUP|RB_CHECKED,

                    "RB1",
                    pScheme);

    pRb[1] = RbCreate(ID_RADIOBUTTON2,
                    255, 85, 310, 125,
                    RB_DRAW,

                    "RB2",
                    pScheme);

    pRb[2] = RbCreate(ID_RADIOBUTTON3,
                    255, 130, 310, 170,
                    RB_DRAW,

                    "RB3",
                    pScheme);

    // draw the Radio Buttons here

    ID = RbGetCheck(pRb[2]);

    if (ID == ID_RADIOBUTTON1) {
        // do something here then clear the check
        ClrState(pRb[0], RB_CHECKED);
    }

```

```
// Change the checked object. Pointer used  
// the ID used will find the correct object  
RbSetCheck(pRb[3], ID_RADIOBUTTON2);  
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RbGetCheck Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbSetCheck Function

## Radio Button

```
C
void RbSetCheck(
    RADIOBUTTON * pRb,
    WORD ID
);
```

## Overview

This function sets the Radio [Button](#) with the given ID to its checked state.

## Input Parameters

Input Parameters	Description
RADIOBUTTON * pRb	Pointer to the Radio <a href="#">Button</a> in the group.
WORD ID	ID of the object to be checked.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

See [RbGetCheck\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RbSetCheck Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbGetText Macro

## Radio Button

C

```
#define RbGetText(pRb) pRb->pText
```

## Overview

This macro returns the address of the current text string used for the object.

## Input Parameters

Input Parameters	Description
pRb	Pointer to the object

## Returns

Returns pointer to the text string being used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#) > [RbGetText Macro](#)

# RbSetText Function

## Radio Button

```
C  
void RbSetText(  
    RADIOBUTTON * pRb,  
    XCHAR * pText  
);
```

## Overview

This function sets the string used for the object.

## Input Parameters

Input Parameters	Description
RADIOBUTTON * pRb	The pointer to the object whose text will be modified
XCHAR * pText	Pointer to the text that will be used

## Returns

none

## Preconditions

none

## Side Effects

none

## > [RbSetText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbMsgDefault Function

## Radio Button

C

```
void RbMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
RB_MSG_CHECKED	Touch Screen,	Set <a href="#">RB_DRAW</a> ,	Depending on the current value of <a href="#">RB_CHECKED</a> Check Box will be redrawn.
	Keyboard	Set/Clear <a href="#">RB_CHECKED</a>	

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message
GOL_MSG * pObj	The pointer to the GOL message

pRb

The pointer to the object whose state will be modified

---

## Returns

none

---

## Preconditions

none

---

## Side Effects

none

---

## Example

See [BtnTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RbMsgDefault Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RbTranslateMsg Function

## Radio Button

```
C
WORD RbTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
RB_MSG_CHECKED	Touch Screen	EVENT_PRESS	If event occurs at the position of the Radio Button, the Radio Button is checked.
	Keyboard	EVENT_KEYSCAN	If event parameter matches and parameter matches <a href="#">SCAN_CR</a> and <a href="#">SCAN_SP</a> while the button is not checked.

OBJ_MSG_INVALID	Any	Any	If the message affects the object
-----------------	-----	-----	-----------------------------------

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pRb	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- RB\_MSG\_CHECKED – Radio [Button](#) is checked
- OBJ\_MSG\_INVALID – Radio [Button](#) is not affected

## Preconditions

none

## Side Effects

none

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RbTranslateMsg Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RADIOBUTTON Structure

## Radio Button

```
C
typedef struct {
    OBJ_HEADER hdr;
    OBJ_HEADER * pHead;
    OBJ_HEADER * pNext;
    SHORT textHeight;
    XCHAR * pText;
} RADIOBUTTON;
```

## Overview

the structure contains data for the Radio [Button](#)

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
OBJ_HEADER * pHead;	Pointer to the first Radio <a href="#">Button</a> in the group
OBJ_HEADER * pNext;	Pointer to the next Radio <a href="#">Button</a> in the group
SHORT textHeight;	Pre-computed text height
XCHAR * pText;	Pointer to the text

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [RADIOBUTTON Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

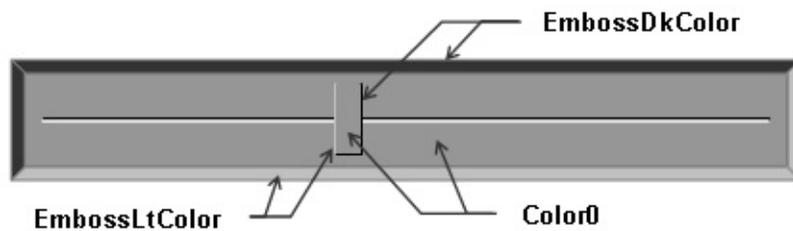
## Slider/Scroll Bar

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

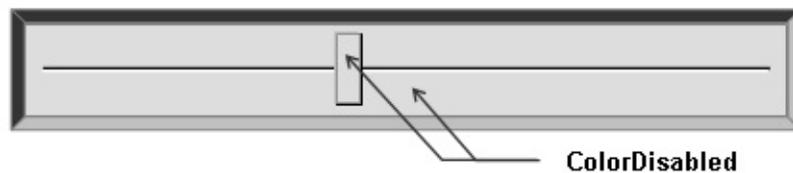
Slider or Scrollbar supports both Keyboard and Touchscreen inputs, replying to their events with the following messages:

1. SLD\_MSG\_INC - when the thumb is to be incremented in position
2. SLD\_MSG\_DEC - when the thumb is to be decremented in position

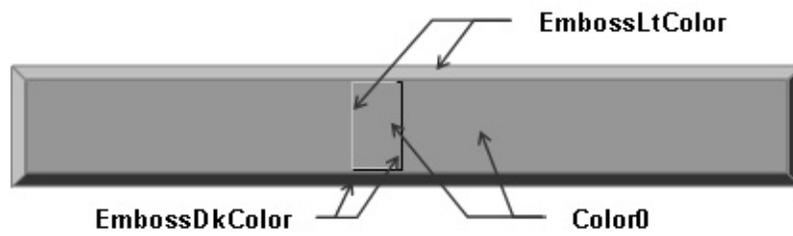
The Slider object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



**Slider Mode enabled state**



**Slider Mode disabled state**



**Scrollbar Mode enabled state**

## Functions

	Name	Description
◆	<a href="#"><u>SldCreate</u></a>	This function creates a <a href="#"><u>SLIDER</u></a> object with the parameters given. Depending on the <a href="#"><u>SLD_SCROLLBAR</u></a> state bit slider or scrollbar mode is set. If <a href="#"><u>SLD_SCROLLBAR</u></a> is set, mode is scrollbar; if not set mode is slider. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
◆	<a href="#"><u>SldDraw</u></a>	This function renders the object on the

		<p>screen using the current parameter settings. 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. 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.</p>
◆	<a href="#"><u>SldSetPage</u></a>	<p>This sets the page size of the object. Page size defines the delta change of the thumb position when incremented via <a href="#"><u>SldIncPos()</u></a> or decremented via <a href="#"><u>SldDecPos()</u></a>. Page size minimum value is 1. Maximum value is range/2.</p>
◆	<a href="#"><u>SldSetPos</u></a>	<p>This function sets the position of the slider thumb. Value should be in the set range inclusive. Object must be redrawn to reflect the change.</p>
◆	<a href="#"><u>SldSetRange</u></a>	<p>This sets the range of the thumb. If this field is changed Object must be completely redrawn to reflect the change.</p>
◆	<a href="#"><u>SldMsgDefault</u></a>	<p>This function performs the actual state change based on the translated message given. The following state changes are supported:</p>
◆	<a href="#"><u>SldTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

## Macros

Name	Description
<a href="#"><u>SldGetPage</u></a>	Returns the current page size of the object. Page size defines the delta change of the thumb position when incremented via <a href="#"><u>SldIncPos()</u></a> or decremented via <a href="#"><u>SldDecPos()</u></a> . Page size minimum value is 1. Maximum value is range/2.
<a href="#"><u>SldGetPos</u></a>	Returns returns the current position of the slider thumb.
<a href="#"><u>SldGetRange</u></a>	Returns the current range of the thumb.
<a href="#"><u>SldIncPos</u></a>	This macro increment the slider position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.
<a href="#"><u>SldDecPos</u></a>	This macro decrement the slider position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.

## Structures

Name	Description
<a href="#"><u>SLIDER</u></a>	Defines the parameters required for a slider/scrollbar Object. Depending on the <a href="#"><u>SLD_SCROLLBAR</u></a> state bit slider or scrollbar mode is set. If <a href="#"><u>SLD_SCROLLBAR</u></a> is set, mode is scrollbar; if not set mode is slider. For scrollbar mode, focus rectangle is

|| | not drawn.

## Topics

---

Name	Description
<a href="#">Slider States</a>	List of Slider bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Slider States

Macros | [Slider/Scroll Bar](#)

List of [Slider](#) bit states.

## Macros

Name	Description
<a href="#">SLD_DISABLED</a>	Bit for disabled state
<a href="#">SLD_DRAW</a>	Bit to indicate whole slider must be redrawn
<a href="#">SLD_DRAW_FOCUS</a>	Bit to indicate that only the focus will be redrawn
<a href="#">SLD_DRAW_THUMB</a>	Bit to indicate that only thumb area must be redrawn
<a href="#">SLD_FOCUSED</a>	Bit for focus state
<a href="#">SLD_HIDE</a>	Bit to remove object from screen
<a href="#">SLD_SCROLLBAR</a>	Bit for type usage (0 - <a href="#">Slider</a> , 1 - ScrollBar)
<a href="#">SLD_VERTICAL</a>	Bit for orientation (0 - horizontal, 1 - vertical)

## Module

[Slider/Scroll Bar](#)

## Links

[Macros](#), [Slider/Scroll Bar](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll](#)

## [Bar](#) > [Slider States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SLD\_DISABLED Macro

---

C

```
#define SLD_DISABLED 0x0002 // Bit for disabled sta
```

## Description

---

Bit for disabled state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SLD\_DRAW Macro

---

C

```
#define SLD_DRAW 0x4000 // Bit to indicate whole sl.
```

### Description

---

Bit to indicate whole slider must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SLD\_DRAW\_FOCUS Macro

---

C

```
#define SLD_DRAW_FOCUS 0x2000 // Bit to indicate the
```

### Description

---

Bit to indicate that only the focus will be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_DRAW\\_FOCUS Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SLD\_DRAW\_THUMB Macro

---

C

```
#define SLD_DRAW_THUMB 0x1000 // Bit to indicate th
```

### Description

---

Bit to indicate that only thumb area must be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_DRAW\\_THUMB Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SLD\_FOCUSED Macro

---

C

```
#define SLD_FOCUSED 0x0001 // Bit for focus state
```

## Description

---

Bit for focus state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SLD\_HIDE Macro

---

```
C
#define SLD_HIDE 0x8000 // Bit to remove object from screen
```

### Description

---

Bit to remove object from screen

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SLD\_SCROLLBAR Macro

---

C

```
#define SLD_SCROLLBAR 0x0010 // Bit for type usage
```

### Description

---

Bit for type usage (0 - [Slider](#), 1 - ScrollBar)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/ScrollBar](#) > [Slider States](#) > [SLD\\_SCROLLBAR Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SLD\_VERTICAL Macro

C

```
#define SLD_VERTICAL 0x0004 // Bit for orientation
```

## Description

Bit for orientation (0 - horizontal, 1 - vertical)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [Slider States](#) > [SLD\\_VERTICAL Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldCreate Function

## Slider/Scroll Bar

```
C
SLIDER * SldCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    WORD range,
    WORD page,
    WORD pos,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [SLIDER](#) object with the parameters given. Depending on the [SLD\\_SCROLLBAR](#) state bit slider or scrollbar mode is set. If [SLD\\_SCROLLBAR](#) is set, mode is scrollbar; if not set mode is slider. It automatically attaches the new object into a global linked list of objects and returns the address of the object.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.

SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the Object.
WORD state	This defines the initial state of the Object.
WORD 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.
WORD 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.
WORD pos	This defines the initial position of the thumb.
GOL_SCHEME * pScheme	The pointer to the style scheme used for the Object. Set to NULL if default style scheme is used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
GOL_SCHEME *pScheme;
SLIDER *slider[3];
WORD state;

    pScheme = GOLCreateScheme();

    // create a slider with
    //     range = [0 - 50000]
    //     delta = 500
    //     initial position = 25000
    state = SLD_DRAW;
    slider[0] = SldCreate( 5,
                          150, 145, 285, 181,
                          state,
                          50000, 500, 25000,
                          pScheme);

    if (slider[0] == NULL)
        return 0;

    // create a scrollbar with
    //     range = [0 - 100]
    //     delta = 20
    //     initial position = 0

    state = SLD_DRAW|SLD_SCROLLBAR;
    slider[1] = SldCreate( 6,
                          150, 190, 285, 220,
                          state,
                          100, 20, 0,
                          pScheme);

    if (slider[1] == NULL)
        return 0;
```

```

// create a vertical slider with
//     range = [0 - 30]
//     delta = 2
//     initial position = 20

state = SLD_DRAW|SLD_VERTICAL;
slider[2] = SldCreate( 7,
                      120, 145, 140, 220,
                      state,
                      30, 2, 20,
                      pScheme);

if (slider[2] == NULL)
    return 0;

// draw the sliders
while(!sldDraw(slider[0])); // draw the hor
while(!sldDraw(slider[1])); // draw the hor
while(!sldDraw(slider[2])); // draw the ver
return 1;

```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldDraw Function

## Slider/Scroll Bar

```
C  
WORD SldDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

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.

## Input Parameters

Input Parameters	Description
pSld	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [SldCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldSetPage Function

## Slider/Scroll Bar

```
C
void SldSetPage(
    SLIDER * pSld,
    WORD newPage
);
```

## Overview

This sets the page size of the object. Page size defines the delta change of the thumb position when incremented via [SldIncPos\(\)](#) or decremented via [SldDecPos\(\)](#). Page size minimum value is 1. Maximum value is range/2.

## Input Parameters

Input Parameters	Description
SLIDER * pSld	Pointer to the object.
WORD newPage	Value of the new page used.

## Returns

None.

## Preconditions

none

## Side Effects

Position of the thumb may change when redrawn.

## Example

---

See [SldIncPos\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldSetPage Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldGetPage Macro

## Slider/Scroll Bar

```
C
#define SldGetPage(pSld) (((SLIDER*)pSld)->page)
```

## Overview

Returns the current page size of the object. Page size defines the delta change of the thumb position when incremented via [SldIncPos\(\)](#) or decremented via [SldDecPos\(\)](#). Page size minimum value is 1. Maximum value is range/2.

## Input Parameters

Input Parameters	Description
pSld	Pointer to the object.

## Returns

Returns the current value of the slider page.

## Preconditions

none

## Side Effects

none

## Example

```
WORD page;
```

[Copy Code](#)

```
SLIDER *pSld;  
  
    page = SldGetPage(pSld);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldGetPage Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldSetPos Function

## Slider/Scroll Bar

```
C  
void SldSetPos(  
    SLIDER * pSld,  
    SHORT newPos  
);
```

## Overview

This function sets the position of the slider thumb. Value should be in the set range inclusive. Object must be redrawn to reflect the change.

## Input Parameters

Input Parameters	Description
SLIDER * pSld	Pointer to the object.
SHORT newPos	The new position of the slider's thumb.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
SLIDER *pSlider;
DWORD ctr = 0;

    // create slider here and initialize parameters
    SetState(pSlider, SLD_DRAW);
    SldDraw(pSlider);

    while("some condition") {
        SldSetPos(pSlider, ctr);
        SetState(pSlider, SLD_DRAW_THUMB);
        SldDraw(pSlider);
        // update ctr here
        ctr = "some source of value";
    }
```

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldSetPos Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldGetPos Macro

## Slider/Scroll Bar

C

```
#define SldGetPos(pSld) (((SLIDER*)pSld)->pos)
```

## Overview

Returns returns the current position of the slider thumb.

## Input Parameters

Input Parameters	Description
pSld	Pointer to the object.

## Returns

Returns the current position of the slider's thumb.

## Preconditions

none

## Side Effects

none

## Example

```
#define MAXVALUE 100;  
  
SLIDER *pSlider;  
DWORD ctr = 0;
```

[Copy Code](#)

```
// create slider here and initialize parameters
SetState(pSlider, SLD_DRAW);
SldDraw(pSlider);

while("some condition") {
    SldSetPos(pSlider, ctr);
    SetState(pSlider, SLD_DRAW_THUMB);
    SldDraw(pSlider);
    // update ctr here
    ctr = "some source of value";
}

if (SldGetPos(pSlider) > (MAXVALUE - 1))
    return 0;
else
    "do something else"
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldGetPos Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldSetRange Function

## Slider/Scroll Bar

C

```
void SldSetRange(  
    SLIDER * pSld,  
    SHORT newRange  
);
```

## Overview

This sets the range of the thumb. If this field is changed Object must be completely redrawn to reflect the change.

## Input Parameters

Input Parameters	Description
SLIDER * pSld	Pointer to the object.
SHORT newRange	Value of the new range used.

## Returns

None.

## Preconditions

none

## Side Effects

Position of the thumb may change when redrawn.

## Example

[Copy Code](#)

```
SLIDER *pSld;  
  
    SldSetRange(pSld, 100);  
    // to completely redraw the object when GOLDraw  
    SetState(pSld, SLD_DRAW);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldSetRange Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldGetRange Macro

## Slider/Scroll Bar

C

```
#define SldGetRange(pSld) (((SLIDER*)pSld)->range)
```

## Overview

Returns the current range of the thumb.

## Input Parameters

Input Parameters	Description
pSld	Pointer to the object.

## Returns

Returns the current range of the slider thumb.

## Preconditions

none

## Side Effects

none

## Example

```
WORD range;  
SLIDER *pSld;  
  
range = SldGetRange(pSld);
```

[Copy Code](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldGetRange Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldIncPos Macro

## Slider/Scroll Bar

C

```
#define SldIncPos(pSld) \  
    SldSetPos  
    (  
        pSld,  
        (((DWORD) pSld->pos + (DWORD) ((SLIDER*)pSld  
            (((SLIDER*)pSld)->pos + ((SLI
```

## Overview

This macro increment the slider position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.

## Input Parameters

Input Parameters	Description
pSld	Pointer to the object.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
void ControlSpeed(SLIDER* pSld, int setSpeed, int c
{
    SldSetPage(pSld, 1); // set
    if (setSpeed < curSpeed) {
        while(SldGetPos(pSld) < SetSpeed)
            SldIncPos(pSld); // incr
    }
    else if (setSpeed > curSpeed) {
        while(SldGetPos(pSld) > SetSpeed)
            SldDecPos(pSld); // decr
    }
}
```

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldIncPos Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldDecPos Macro

## Slider/Scroll Bar

C

```
#define SldDecPos(pSld) \  
    SldSetPos  
    (  
        pSld,  
        (((LONG) ((SLIDER*)pSld)->pos - (LONG) ((SLIDER*)pSld)->pos - ((SLIDER*)pSld)->delta) / ((SLIDER*)pSld)->delta)  
    )
```

## Overview

This macro decrement the slider position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.

## Input Parameters

Input Parameters	Description
pSld	Pointer to the object.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

---

See [SldIncPos\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldDecPos Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SldMsgDefault Function

## Slider/Scroll Bar

C

```
void SldMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
SLD_MSG_INC	Touch Screen, Keyboard	Set <a href="#">SLD_DRAW_THUMB</a>	<a href="#">Slider</a> will be redrawn with thumb in the incremented position.
SLD_MSG_DEC	Touch Screen, Keyboard	Set <a href="#">SLD_DRAW_THUMB</a>	<a href="#">Slider</a> will be redrawn with thumb in the decremented position.

## Input Parameters

Input Parameters	Description

WORD translatedMsg	The translated message.
GOL_MSG * pMsg	The pointer to the GOL message.
pSld	The pointer to the object whose state will be modified.

---

## Returns

none

---

## Preconditions

none

---

## Side Effects

none

---

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldMsgDefault Function](#)

# SldTranslateMsg Function

## Slider/Scroll Bar

```
C
WORD SldTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
SLD_MSG_INC	Touch Screen	EVENT_PRESS, EVENT_MOVE	If events occur at the position of the slider, the x,y horizontal position is the x,y vertical slice.
	Keyboard	EVENT_KEYSCAN	If event parameter matches the parameter matches <a href="#">SCAN_UP</a>

			<a href="#">SCAN_LEI</a>
SLD_MSG_DEC	Touch Screen	EVENT_PRESS, EVENT_MOVE	If events o position fa the slider position is the x,y horizontal the x,y vertical slic
	Keyboard	EVENT_KEYSCAN	If event parameter: matches th parameter matches <a href="#">SCAN_DO</a> or <a href="#">SCAN_RIC</a>
OBJ_MSG_PASSIVE	Touch Screen	EVENT_RELEASE	If events o position fa the slider.
OBJ_MSG_INVALID	Any	Any	If the m affect the c

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pSld	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

---

Returns the translated message depending on the received GOL message:

- SLD\_MSG\_INC – Increment slider position
- SLD\_MSG\_DEC – Decrement slider position
- OBJ\_MSG\_PASSIVE – [Slider](#) is not affected
- OBJ\_MSG\_INVALID – [Slider](#) is not affected

## Preconditions

---

none

## Side Effects

---

none

## Example

---

Usage is similar to [BtnTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SldTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SLIDER Structure

## Slider/Scroll Bar

```
C
typedef struct {
    OBJ_HEADER hdr;
    WORD currPos;
    WORD prevPos;
    WORD range;
    WORD pos;
    WORD page;
    WORD thWidth;
    WORD thHeight;
} SLIDER;
```

## Overview

Defines the parameters required for a slider/scrollbar Object. Depending on the [SLD\\_SCROLLBAR](#) state bit slider or scrollbar mode is set. If [SLD\\_SCROLLBAR](#) is set, mode is scrollbar; if not set mode is slider. For scrollbar mode, focus rectangle is not drawn.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
WORD currPos;	Position of the slider relative to minimum.
WORD prevPos;	Previous position of the slider relative to minimum.

WORD range;	User defined range of the slider. Minimum value at 0 and maximum at 0x7FFF.
WORD pos;	Position of the slider in range domain.
WORD page;	User specified resolution to incrementally change the position
WORD thWidth;	Thumb width. This is computed internally.
WORD thHeight;	Thumb width. This is computed internally. User must not change this value.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#) > [SLIDER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

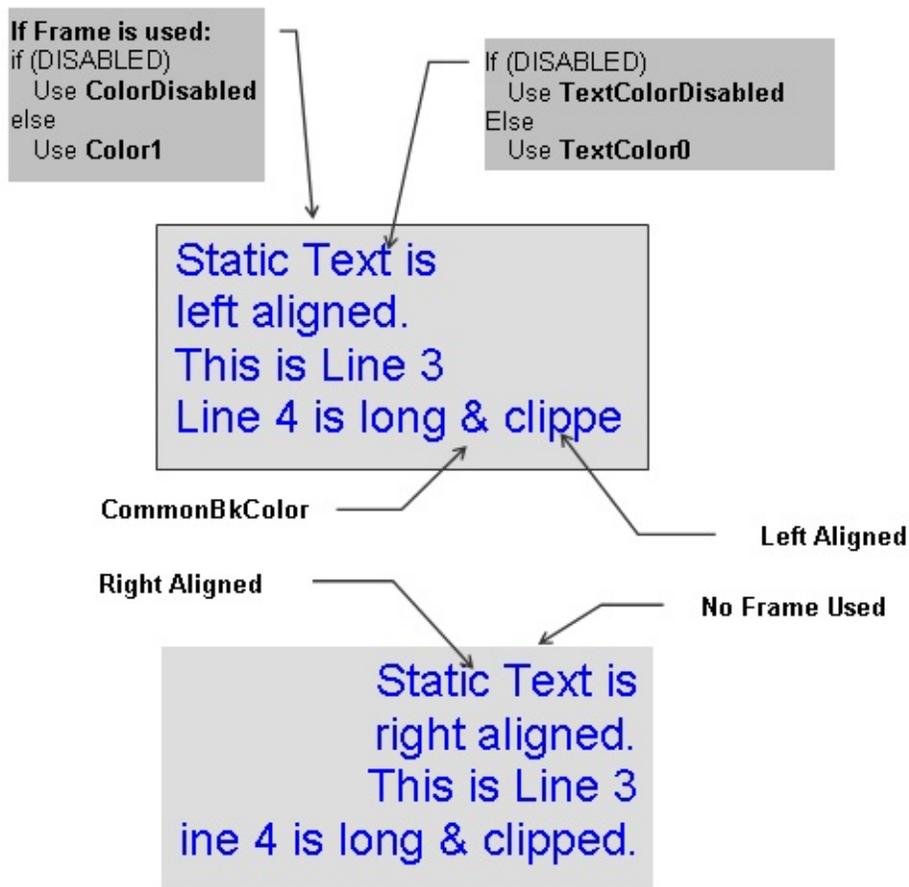
# Static Text

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Static Text supports only Touchscreen inputs, replying to their events with the message:

ST\_MSG\_SELECTED - when the touch is within the dimension of the object.

The Static Text Object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



## Functions

	Name	Description
◆	<u><a href="#">StCreate</a></u>	This function creates a <b>STATICTEXT</b> 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.
◆	<u><a href="#">StDraw</a></u>	This function renders the object on the screen using the current parameter settings. Location of the object is determined by the left, top, right and bottom parameters. The

		<p>colors used are dependent on the state of the object. The font used is determined by the style scheme set.</p> <p>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.</p>
◆	<a href="#"><u>StSetText</u></a>	This function sets the string that will be used for the object.
◆	<a href="#"><u>StTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

## Macros

Name	Description
<a href="#"><u>StGetText</u></a>	This macro returns the address of the current text string used for the object.

## Structures

Name	Description
<a href="#"><u>STATICTEXT</u></a>	Defines the parameters required for a Static Text Object.

## Topics

Name	Description
------	-------------

[Static Text States](#)

List of Static Text bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Static Text States

[Macros](#) | [Static Text](#)

List of Static Text bit states.

## Macros

Name	Description
<a href="#">ST_CENTER_ALIGN</a>	Bit to indicate text is center aligned.
<a href="#">ST_DISABLED</a>	Bit for disabled state.
<a href="#">ST_DRAW</a>	Bit to indicate static text must be redrawn.
<a href="#">ST_FRAME</a>	Bit to indicate frame is displayed.
<a href="#">ST_HIDE</a>	Bit to remove object from screen.
<a href="#">ST_RIGHT_ALIGN</a>	Bit to indicate text is left aligned.
<a href="#">ST_UPDATE</a>	Bit to indicate that text area only is redrawn.

## Module

[Static Text](#)

## Links

[Macros](#), [Static Text](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#)

## ST\_CENTER\_ALIGN Macro

```
C
#define ST_CENTER_ALIGN 0x0008 // Bit to indicate text is center aligned
```

### Description

Bit to indicate text is center aligned.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_CENTER\\_ALIGN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ST\_DISABLED Macro

```
C  
#define ST_DISABLED 0x0002 // Bit for disabled state
```

## Description

Bit for disabled state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_DISABLED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ST\_DRAW Macro

---

C

```
#define ST_DRAW 0x4000 // Bit to indicate static text
```

### Description

---

Bit to indicate static text must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ST\_FRAME Macro

---

C

```
#define ST_FRAME 0x0010 // Bit to indicate frame is
```

### Description

---

Bit to indicate frame is displayed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_FRAME Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ST\_HIDE Macro

---

```
C
#define ST_HIDE 0x8000 // Bit to remove object from
```

### Description

---

Bit to remove object from screen.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ST\_RIGHT\_ALIGN Macro

---

C

```
#define ST_RIGHT_ALIGN 0x0004 // Bit to indicate te
```

### Description

---

Bit to indicate text is left aligned.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_RIGHT\\_ALIGN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ST\_UPDATE Macro

---

```
C
#define ST_UPDATE 0x2000 // Bit to indicate that te
```

### Description

---

Bit to indicate that text area only is redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#) > [ST\\_UPDATE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# StCreate Function

## Static Text

```
C
STATICTEXT * StCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    XCHAR * pText,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the object.
SHORT top	Top most position of the object.
SHORT right	Right most position of the object.
SHORT bottom	Bottom most position of the object.

WORD state	Sets the initial state of the object.
XCHAR * pText	Pointer to the text used in the static text.
GOL_SCHEME * pScheme	Pointer to the style scheme. Set to NULL if default style scheme is used.

## Returns

Returns the pointer to the object created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
GOL_SCHEME *pScheme;
STATICTEXT *pSt;

    pScheme = GOLCreateScheme();
    state = ST_DRAW | ST_FRAME | ST_CENTER_ALIGN;
    StCreate(ID_STATICTEXT1,           // ID
            30,80,235,160,           // dimension
            state,                   // has frame
            "Static Textn Example", // text
            pScheme);                // use given

    while(!StDraw(pSt));           // draw the o
```

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [StCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# StDraw Function

## Static Text

```
C  
WORD StDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pSt	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [StCreate\(\)](#) Example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [StDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# StGetText Macro

## Static Text

```
C
#define StGetText(pSt) pSt->pText
```

## Overview

This macro returns the address of the current text string used for the object.

## Input Parameters

Input Parameters	Description
pSt	Pointer to the object.

## Returns

Returns the pointer to the text string used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [StGetText Macro](#)

# StSetText Function

## Static Text

```
C
void StSetText(
    STATICTEXT * pSt,
    XCHAR * pText
);
```

## Overview

This function sets the string that will be used for the object.

## Input Parameters

Input Parameters	Description
STATICTEXT * pSt	The pointer to the object whose text string will be modified.
XCHAR * pText	The pointer to the string that will be used.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) >

## StSetText Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# StTranslateMsg Function

## Static Text

```
C
WORD StTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

Translated Message	Input Source	Events	Description
ST_MSG_SELECTED	Touch Screen	EVENT_PRESS, EVENT_RELEASE	If event occurs and the x, y position falls in the area of the static text.
OBJ_MSG_INVALID	Any	Any	If the message does not affect the object.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pSt	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- ST\_MSG\_SELECTED – Static Text is selected
- OBJ\_MSG\_INVALID – Static Text is not affected

## Preconditions

none

## Side Effects

none

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [StTranslateMsg Function](#)

# STATICTEXT Structure

## Static Text

C

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT textHeight;
    XCHAR * pText;
} STATICTEXT;
```

## Overview

Defines the parameters required for a Static Text Object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT textHeight;	Pre-computed text height.
XCHAR * pText;	The pointer to text used.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [STATICTEXT Structure](#)

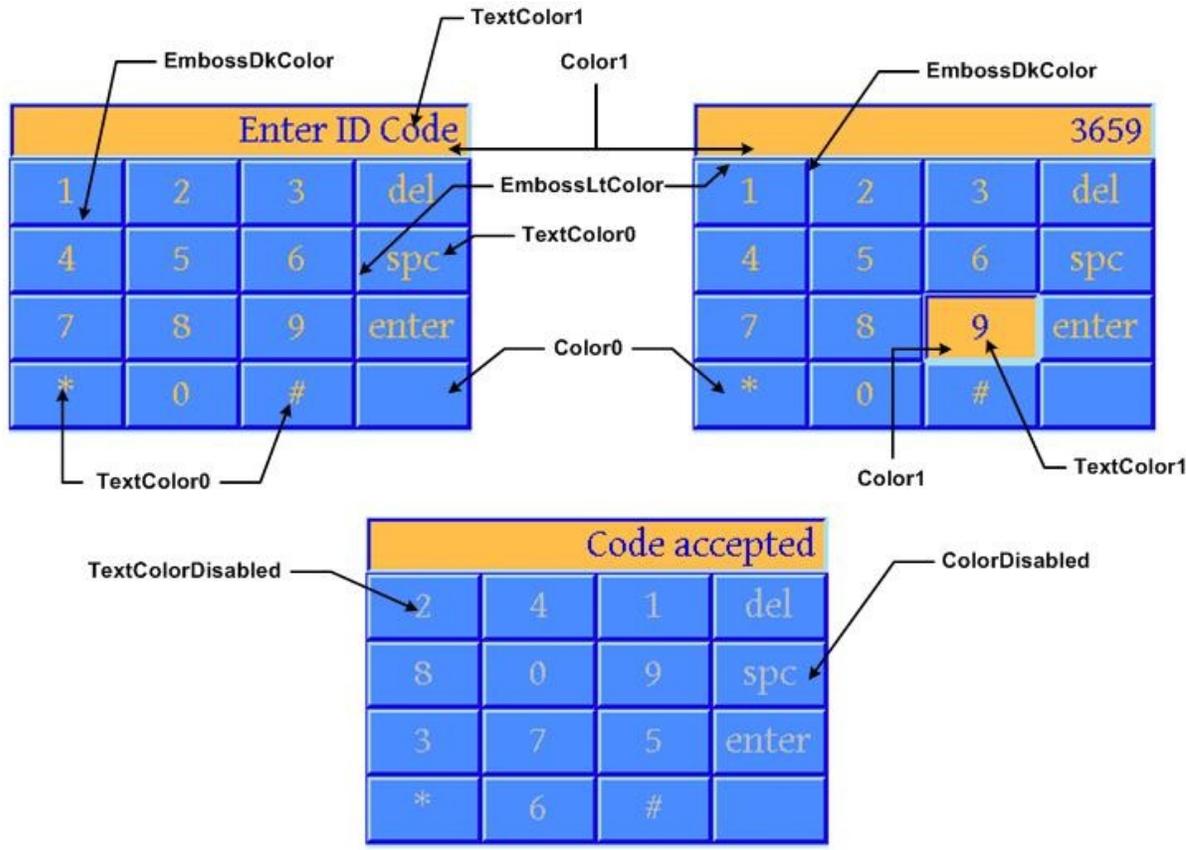
# Text Entry

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Text Entry supports only Touchscreen inputs, replying to their events with the following messages:

1. `TE_MSG_RELEASE` – A key has been released.
2. `TE_MSG_PRESS` – A key is pressed.
3. `TE_MSG_ADD_CHAR` – A key was released with character assigned.
4. `TE_MSG_DELETE` – A key was released with delete command assigned.
5. `TE_MSG_SPACE` - A key was released with space command assigned.
6. `TE_MSG_ENTER` - A key was released with enter command assigned.

The Text Entry Object is rendered using the assigned style scheme. The following figure illustrates the color assignments.



## Functions

	Name	Description
◆	<u><a href="#">TeCreate</a></u>	This function creates a <u><a href="#">TEXTENTRY</a></u> 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.
◆	<u><a href="#">TeDraw</a></u>	This function renders the object on the screen using the current parameter settings. Location of the object is determined by the left, top, right and bottom parameters. The colors used are dependent on the state of the

		<p>object.</p> <p>This widget will draw the keys using the function <a href="#">GOLPanelDraw()</a>. The number of keys will depend on the horizontal and vertical parameters given (<math>\text{horizontalKeys} * \text{verticalKeys}</math>).</p>
◆	<a href="#">TeSetBuffer</a>	<p>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 <code>MaxSize</code>, string will be truncated to <code>MaxSize</code>. <code>pText</code> must point to a valid memory location with size equal to <code>MaxSize+1</code>. The +1 is used for the string terminator.</p>
◆	<a href="#">TeClearBuffer</a>	<p>This function will clear the data in the display. You must set the drawing state bit <a href="#">TE_UPDATE_TEXT</a> to update the <a href="#">TEXTENTRY</a> on the screen.</p>
◆	<a href="#">TeGetKeyCommand</a>	<p>This function will return the currently used command by a key with the given index.</p>
◆	<a href="#">TeSetKeyCommand</a>	<p>This function will assign a command (<a href="#">TE_DELETE_COM</a>, <a href="#">TE_SPACE_COM</a> or <a href="#">TE_ENTER_COM</a>) to a key with the given index.</p>
◆	<a href="#">TeCreateKeyMembers</a>	<p>This function will create the list of <code>KEYMEMBERS</code> that holds the information on each key. The number of keys is determined by the equation (<math>\text{verticalKeys} * \text{horizontalKeys}</math>). The</p>

		<p>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 equal or greater than (verticalKeys*horizontalKeys). The last key is assigned with an index of (verticalKeys*horizontalKeys)-1. No checking is performed on the length of *pText[] entries to match (verticalKeys*horizontalKeys).</p>
◆	<u>TeAddChar</u>	<p>This function will insert a character to the end of the buffer. The character inserted is dependent on the currently pressed key. Drawing states <u>TE_UPDATE_TEXT</u> or <u>TE_DRAW</u> must be set to see the effect of this insertion.</p>
◆	<u>TeIsKeyPressed</u>	<p>This function will test if a key given by its index in the TextEntry object has been pressed.</p>
◆	<u>TeSpaceChar</u>	<p>This function will insert a space character to the end of the buffer. Drawing states <u>TE_UPDATE_TEXT</u> or <u>TE_DRAW</u> must be set to see the effect of this insertion.</p>
◆	<u>TeDelKeyMembers</u>	<p>This function will delete the <u>KEYMEMBER</u> list assigned to the object from memory. Pointer to the <u>KEYMEMBER</u> list is then initialized to NULL.</p>

◆	<a href="#"><u>TeSetKeyText</u></a>	This function will set the text assigned to a key with the given index.
◆	<a href="#"><u>TeMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
◆	<a href="#"><u>TeTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. If the message is valid, the keys in the Text Entry object will be scanned to detect which key was pressed. If True, the corresponding text will be displayed, the 'text' will also be stored in the TeOutput parameter of the object.

## Macros

Name	Description
<a href="#"><u>TeGetBuffer</u></a>	This macro will return the currently used buffer in the TextEntry object.

## Structures

Name	Description
<a href="#"><u>TEXTENTRY</u></a>	Defines the parameters required for a TextEntry Object.
<a href="#"><u>KEYMEMBER</u></a>	Defines the parameters and the strings assigned for each key.

## Topics

---

Name	Description
<a href="#">TextEntry States</a>	List of Text Entry bit states.
<a href="#">Key Command Types</a>	List of available Key command types.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TextEntry States

[Macros](#) | [Text Entry](#)

List of [Text Entry](#) bit states.

## Macros

Name	Description
<a href="#">TE_KEY_PRESSED</a>	Bit for press state of one of the keys.
<a href="#">TE_DISABLED</a>	Bit for disabled state.
<a href="#">TE_ECHO_HIDE</a>	Bit to hide the entered characters and instead echo "*" characters to the display.
<a href="#">TE_DRAW</a>	Bit to indicate object must be redrawn.
<a href="#">TE_HIDE</a>	Bit to indicate object must be removed from screen.
<a href="#">TE_UPDATE_KEY</a>	Bit to indicate redraw of a key is needed.
<a href="#">TE_UPDATE_TEXT</a>	Bit to indicate redraw of the text displayed is needed.

## Module

[Text Entry](#)

## Links

[Macros](#), [Text Entry](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_KEY\_PRESSED Macro

---

C

```
#define TE_KEY_PRESSED 0x0004 // Bit for press state
```

### Description

---

Bit for press state of one of the keys.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_KEY\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_DISABLED Macro

---

```
C
#define TE_DISABLED 0x0002 // Bit for disabled state
```

### Description

---

Bit for disabled state.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_ECHO\_HIDE Macro

```
C
```

```
#define TE_ECHO_HIDE 0x0008 // Bit to hide the ente
```

### Description

Bit to hide the entered characters and instead echo "\*" characters to the display.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_ECHO\\_HIDE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_DRAW Macro

---

```
C
#define TE_DRAW 0x4000 // Bit to indicate object must be redrawn.
```

### Description

---

Bit to indicate object must be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_HIDE Macro

---

```
C
#define TE_HIDE 0x8000 // Bit to indicate object mu
```

### Description

---

Bit to indicate object must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_UPDATE\_KEY Macro

---

```
C
#define TE_UPDATE_KEY 0x2000 // Bit to indicate red
```

### Description

---

Bit to indicate redraw of a key is needed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_UPDATE\\_KEY Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_UPDATE\_TEXT Macro

---

C

```
#define TE_UPDATE_TEXT 0x1000 // Bit to indicate red
```

### Description

---

Bit to indicate redraw of the text displayed is needed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#) > [TE\\_UPDATE\\_TEXT Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Key Command Types

[Macros](#) | [Text Entry](#)

List of available Key command types.

## Macros

Name	Description
<a href="#">TE_DELETE_COM</a>	This macro is used to assign a "delete" command on a key.
<a href="#">TE_ENTER_COM</a>	This macro is used to assign an "enter" (carriage return) command on a key.
<a href="#">TE_SPACE_COM</a>	This macro is used to assign an insert "space" command on a key.

## Module

[Text Entry](#)

## Links

[Macros](#), [Text Entry](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [Key Command Types](#)

## TE\_DELETE\_COM Macro

```
C
#define TE_DELETE_COM 0x01 // This macro is used
```

### Description

This macro is used to assign a "delete" command on a key.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [Key Command Types](#) > [TE\\_DELETE\\_COM Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_ENTER\_COM Macro

---

```
C
#define TE_ENTER_COM 0x03 // This macro is used to
```

### Description

---

This macro is used to assign an "enter" (carriage return) command on a key.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [Key Command Types](#) > [TE\\_ENTER\\_COM Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TE\_SPACE\_COM Macro

```
C
#define TE_SPACE_COM 0x02 // This macro is used to
```

### Description

This macro is used to assign an insert "space" command on a key.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [Key Command Types](#) > [TE\\_SPACE\\_COM Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeCreate Function

## Text Entry

C

```
TEXTENTRY * TeCreate(  
    WORD ID,  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    WORD state,  
    SHORT horizontalKeys,  
    SHORT verticalKeys,  
    XCHAR * pText[],  
    void * pBuffer,  
    WORD bufferLength,  
    void * pDisplayFont,  
    GOL_SCHEME * pScheme  
);
```

## Overview

This function creates a [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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance
SHORT left	Left most position of the object.

SHORT top	Top most position of the object.
SHORT right	Right most position of the object.
SHORT bottom	Bottom most position of the object.
WORD state	state of the widget.
SHORT horizontalKeys	Number of horizontal keys
SHORT verticalKeys	Number of vertical keys
XCHAR * pText[]	array of pointer to the custom "text" assigned by the user.
void * pBuffer	pointer to the buffer that holds the text to be displayed.
WORD bufferLength	length of the buffer assigned by the user.
void * pDisplayFont	pointer to the font image to be used on the editbox
GOL_SCHEME * pScheme	Pointer to the style scheme used. Output Returns the pointer to the object created.

## Preconditions

If the object will use customized keys, the structure CUSTOMEKEYS must be populated before calling this function.

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) >

# TeCreate Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeDraw Function

## Text Entry

```
C  
WORD TeDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

This widget will draw the keys using the function [GOLPanelDraw\(\)](#). The number of keys will depend on the horizontal and vertical parameters given (horizontalKeys\*verticalKeys).

## Input Parameters

Input Parameters	Description
pTe	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeGetBuffer Macro

## Text Entry

```
C
#define TeGetBuffer(pTe) (((TEXTENTRY *)pTe)->pTeOut)
```

## Overview

This macro will return the currently used buffer in the TextEntry object.

## Input Parameters

Input Parameters	Description
pTe	pointer to the object

## Returns

It will return a pointer to the buffer used.

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeGetBuffer Macro](#)

# TeSetBuffer Function

## Text Entry

```
C
void TeSetBuffer(
    TEXTENTRY * pTe,
    XCHAR * pText,
    WORD MaxSize
);
```

## Overview

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+1. The +1 is used for the string terminator.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object
XCHAR * pText	pointer to the new text buffer to be displayed
maxSize	maximum length of the new buffer to be used.

## Returns

none.

## Preconditions

---

none

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeSetBuffer Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeClearBuffer Function

## Text Entry

```
C
void TeClearBuffer(
    TEXTENTRY * pTe
);
```

## Overview

This function will clear the data in the display. You must set the drawing state bit [TE\\_UPDATE\\_TEXT](#) to update the [TEXTENTRY](#) on the screen.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object

## Returns

none

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeClearBuffer Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeGetKeyCommand Function

## Text Entry

```
C
WORD TeGetKeyCommand(
    TEXTENTRY * pTe,
    WORD index
);
```

## Overview

This function will return the currently used command by a key with the given index.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object
WORD index	index to the key in the link list

## Returns

It will return the command ID currently set for the key. If the given index is not in the list the function returns zero. 0x00 - no command is assigned or the index given does not exist. 0x01 - [TE\\_DELETE\\_COM](#) 0x02 - [TE\\_SPACE\\_COM](#) 0x03 - [TE\\_ENTER\\_COM](#)

## Preconditions

none

## Side Effects

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeGetKeyCommand Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeSetKeyCommand Function

## Text Entry

```
C
BOOL TeSetKeyCommand(
    TEXTENTRY * pTe,
    WORD index,
    WORD command
);
```

## Overview

This function will assign a command ([TE\\_DELETE\\_COM](#), [TE\\_SPACE\\_COM](#) or [TE\\_ENTER\\_COM](#)) to a key with the given index.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object
WORD index	index to the key in the link list
WORD command	command assigned for the key

## Returns

Returns TRUE if successful and FALSE if not.

## Preconditions

none

## Side Effects

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeSetKeyCommand Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeCreateKeyMembers Function

## Text Entry

```
C
KEYMEMBER * TeCreateKeyMembers(
    TEXTENTRY * pTe,
    XCHAR * pText[]
);
```

## Overview

This function will create the list of KEYMEMBERS that holds the information on each key. The number of keys is determined by the equation (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 equal or greater than (verticalKeys\*horizontalKeys). The last key is assigned with an index of (verticalKeys\*horizontalKeys)-1. No checking is performed on the length of \*pText[] entries to match (verticalKeys\*horizontalKeys).

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object
XCHAR * pText[]	pointer to the text defined by the user

## Returns

Returns the pointer to the newly created **KEYMEMBER** list. A

NULL is returned if the list is not created successfully.

## Preconditions

---

none

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeCreateKeyMembers Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeAddChar Function

## Text Entry

```
C
void TeAddChar(
    TEXTENTRY * pTe
);
```

## Overview

This function will insert a character to the end of the buffer. The character inserted is dependent on the currently pressed key. Drawing states [TE\\_UPDATE\\_TEXT](#) or [TE\\_DRAW](#) must be set to see the effect of this insertion.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeAddChar Function](#)

# TeIsKeyPressed Function

## Text Entry

```
C
BOOL TeIsKeyPressed(
    TEXTENTRY * pTe,
    WORD index
);
```

## Overview

This function will test if a key given by its index in the TextEntry object has been pressed.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object
WORD index	index to the key in the link list

## Returns

Returns a TRUE if the key is pressed. FALSE if key is not pressed or the given index does not exist in the list.

## Preconditions

none

## Side Effects

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TelsKeyPressed Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeSpaceChar Function

## Text Entry

```
C
void TeSpaceChar(
    TEXTENTRY * pTe
);
```

## Overview

This function will insert a space character to the end of the buffer. Drawing states [TE\\_UPDATE\\_TEXT](#) or [TE\\_DRAW](#) must be set to see the effect of this insertion.

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object

## Returns

none.

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeSpaceChar Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeDelKeyMembers Function

## Text Entry

```
C
void TeDelKeyMembers(
    void * pObj
);
```

## Overview

This function will delete the [KEYMEMBER](#) list assigned to the object from memory. Pointer to the [KEYMEMBER](#) list is then initialized to NULL.

## Input Parameters

Input Parameters	Description
pTe	pointer to the object

## Returns

none.

## Preconditions

none

## Side Effects

none.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeDelKeyMembers Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeSetKeyText Function

## Text Entry

```
C
BOOL TeSetKeyText(
    TEXTENTRY * pTe,
    WORD index,
    XCHAR * pText
);
```

## Overview

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

## Input Parameters

Input Parameters	Description
TEXTENTRY * pTe	pointer to the object
WORD index	index to the key in the link list
XCHAR * pText	pointer to the new string to be used

## Returns

Returns TRUE if successful and FALSE if not.

## Preconditions

none

## Side Effects

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeSetKeyText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TeMsgDefault Function

## Text Entry

C

```
void TeMsgDefault(
    WORD translatedMsg,
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function performs the actual state change based on the translated message given. The following state changes are supported:

Translated Message	Input Source	Set/Clear State Bit	Description
TE_MSG_ADD_CHAR	Touch Screen,	Set <a href="#">TE_UPDATE_TEXT</a> , <a href="#">TE_UPDATE_KEY</a> ,	Add character to the buffer and update the text displayed.
		Clear <a href="#">TE_KEY_PRESSED</a>	
TE_MSG_SPACE	Touch Screen,	Set <a href="#">TE_UPDATE_TEXT</a> , <a href="#">TE_UPDATE_KEY</a> ,	Insert space character to the buffer and update the text displayed.

		Clear <u>TE_KEY_PRESSED</u>	
TE_MSG_DELETE	Touch Screen,	Set <u>TE_UPDATE_TEXT</u> , <u>TE_UPDATE_KEY</u> ,	Delete the most recent character from the buffer but do not update the text displayed.
		Clear <u>TE_KEY_PRESSED</u>	
TE_MSG_ENTER	Touch Screen,	Set <u>TE_UPDATE_TEXT</u> , <u>TE_UPDATE_KEY</u> ,	User can define the use of this event in the message callback. Object will just update the key.
		Clear <u>TE_KEY_PRESSED</u>	
TE_MSG_RELEASED	Touch Screen,	Clear <u>TE_KEY_PRESSED</u>	A Key in the object will redraw the unpressed state.
		Set Te_UPDATE_KEY	
TE_MSG_PRESSED	Touch Screen,	Set <u>TE_KEY_PRESSED</u> <u>TE_UPDATE_KEY</u>	A Key in the object will redraw

the press  
state.

## Input Parameters

Input Parameters	Description
WORD translatedMsg	The translated message.
GOL_MSG * pMsg	The pointer to the GOL message.
pTe	The pointer to the object whose state will be modified.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

See [BtnTranslateMsg\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeMsgDefault Function](#)

# TeTranslateMsg Function

## Text Entry

```
C
WORD TeTranslateMsg(
    void * pObj,
    GOL_MSG * pMsg
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. If the message is valid, the keys in the Text Entry object will be scanned to detect which key was pressed. If True, the corresponding text will be displayed, the 'text' will also be stored in the TeOutput parameter of the object.

Translated Message	Input Source	Events	Description
TE_MSG_PRESS	Touch Screen	EVENT_PRESS, EVENT_MOVE	If the event occurs and the x, y position falls within the face of one of the keys of the object while the key is unpressed.
TE_MSG_RELEASED	Touch Screen	EVENT_MOVE	If the event occurs and the x, y

			position fall outside the face of one of the keys of the object while the key is pressed.
TE_MSG_RELEASED	Touch Screen	EVENT_RELEASE	If the event occurs at the x, y position fall does not fall inside any of the faces of the keys of the object.
TE_MSG_ADD_CHAR	Touch Screen	EVENT_RELEASE, EVENT_MOVE	If the event occurs at the x, y position fall in the face of one of the keys of the object while the key is unpressed and the key is associated with a command.
TE_MSG_DELETE	Touch Screen	EVENT_RELEASE, EVENT_MOVE	If the event occurs at the x, y position fall

			in the face of one of the keys of the object while the key is unpressed and the key is associated with the delete command.
TE_MSG_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 the space command.
TE_MSG_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.
OBJ_MSG_INVALID	Any	Any	If the message does not affect the object.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pTe	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- TE\_MSG\_PRESS – A key is pressed
- TE\_MSG\_RELEASED - A key was released (generic for keys with no commands or characters assigned)
- TE\_MSG\_ADD\_CHAR – A key was released with character assigned
- TE\_MSG\_DELETE – A key was released with delete command assigned
- TE\_MSG\_SPACE - A key was released with space command assigned
- TE\_MSG\_ENTER - A key was released with enter command assigned
- OBJ\_MSG\_INVALID – Text Entry is not affected

## Preconditions

---

none

## Side Effects

---

none.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TeTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TEXTENTRY Structure

## Text Entry

C

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT horizontalKeys;
    SHORT verticalKeys;
    XCHAR * pTeOutput;
    WORD CurrentLength;
    WORD outputLenMax;
    KEYMEMBER * pActiveKey;
    KEYMEMBER * pHeadOfList;
    void * pDisplayFont;
} TEXTENTRY;
```

## Overview

Defines the parameters required for a TextEntry Object.

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all objects (see <a href="#">OBJ_HEADER</a> ).
SHORT horizontalKeys;	Number of horizontal keys
SHORT verticalKeys;	Number of vertical keys
XCHAR * pTeOutput;	Pointer to the buffer assigned by the user which holds the text shown in the editbox.
WORD	Current length of the string in the buffer. The

CurrentLength;	maximum value of this is equal to outputLenMax.
WORD outputLenMax;	Maximum expected length of output buffer pTeOutput
KEYMEMBER * pActiveKey;	Pointer to the active key <a href="#">KEYMEMBER</a> . This is only used by the Widget. User must not change
KEYMEMBER * pHeadOfList;	Pointer to head of the list
void * pDisplayFont;	Pointer to the font used in displaying the text.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TEXTENTRY Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# KEYMEMBER Structure

## Text Entry

```
C
typedef struct {
    SHORT left;
    SHORT top;
    SHORT right;
    SHORT bottom;
    SHORT index;
    WORD state;
    BOOL update;
    WORD command;
    XCHAR * pKeyName;
    SHORT textWidth;
    SHORT textHeight;
    void * pNextKey;
} KEYMEMBER;
```

## Overview

Defines the parameters and the strings assigned for each key.

## Members

Members	Description
SHORT left;	Left position of the key
SHORT top;	Top position of the key
SHORT right;	Right position of the key
SHORT bottom;	Bottom position of the key
SHORT index;	Index of the key in the list

WORD state;	State of the key. Either Pressed ( <a href="#">TE_KEY_PRESSED</a> ) or Released (0)
BOOL update;	flag to indicate key is to be redrawn with the current state
WORD command;	Command of the key. Either <a href="#">TE_DELETE_COM</a> , <a href="#">TE_SPACE_COM</a> or <a href="#">TE_ENTER_COM</a> .
XCHAR * pKeyName;	Pointer to the custom text assigned to the key. This is displayed over the face of the key.
SHORT textWidth;	Computed text width, done at creation. Used to predict size and position of text on the key face.
SHORT 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.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [KEYMEMBER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

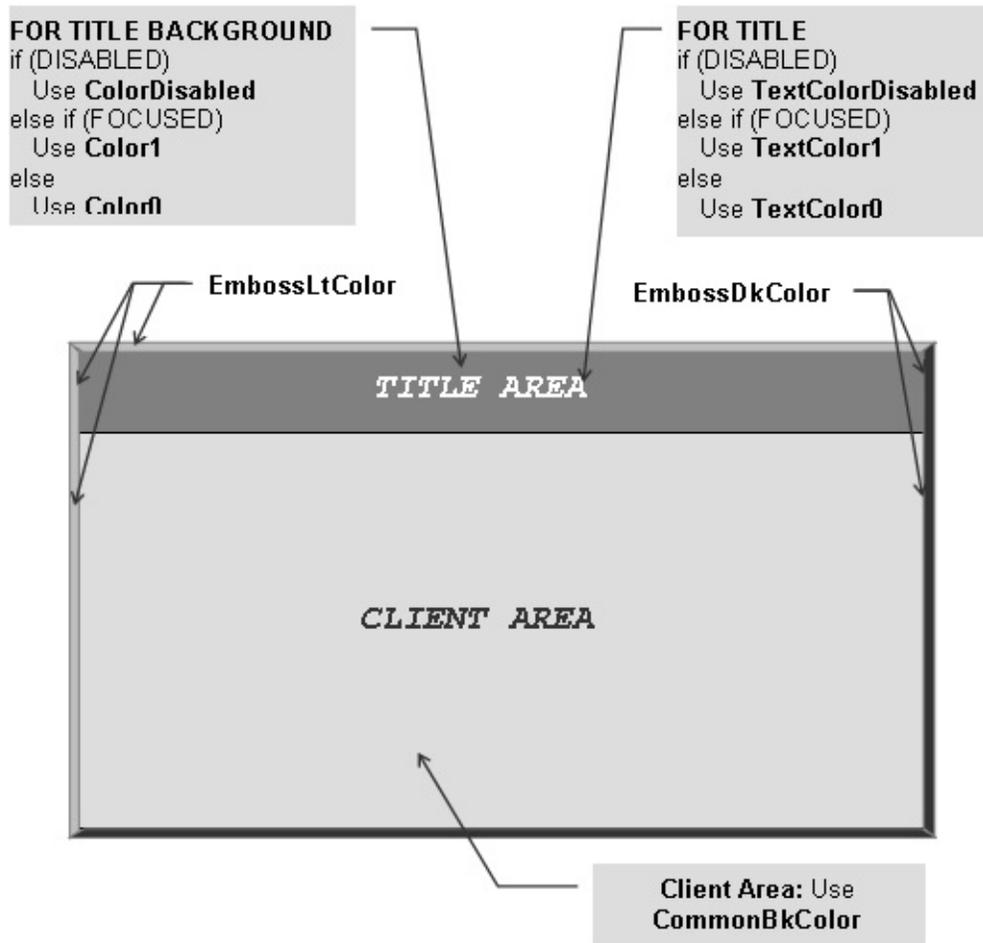
# Window

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

Window supports only Touchscreen inputs, replying to their events with the following messages:

1. WND\_MSG\_TITLE – Title area is selected.
2. WND\_MSG\_CLIENT – Client area is selected.

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.

## Functions

	Name	Description
◆	<u><a href="#">WndCreate</a></u>	This function creates a <b>WINDOW</b> 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.
◆	<u><a href="#">WndDraw</a></u>	This function renders the object on the screen using the current parameter settings. Location of the object is

		<p>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.</p> <p>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.</p>
◆	<a href="#"><u>WndSetText</u></a>	This function sets the string used for the title bar.
◆	<a href="#"><u>WndTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

## Macros

Name	Description
<a href="#"><u>WndGetText</u></a>	This macro returns the address of the current text string used for the title bar.

## Structures

Name	Description
<a href="#"><u>WINDOW</u></a>	The structure contains data for the window

## Topics

--	--

Name	Description
<a href="#">Window States</a>	List of Window bit states.

## Links

---

[Functions](#), [GOL Objects](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Window States

[Macros](#) | [Window](#)

List of [Window](#) bit states.

## Macros

Name	Description
<a href="#">WND_DISABLED</a>	Bit for disabled state
<a href="#">WND_DRAW</a>	Bits to indicate whole window must be redrawn
<a href="#">WND_DRAW_CLIENT</a>	Bit to indicate client area must be redrawn
<a href="#">WND_DRAW_TITLE</a>	Bit to indicate title area must be redrawn
<a href="#">WND_FOCUSED</a>	Bit for focus state
<a href="#">WND_HIDE</a>	Bit to indicate window must be removed from screen
<a href="#">WND_TITLECENTER</a>	Bit to center the text on the Title Area

## Module

[Window](#)

## Links

[Macros](#), [Window](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WND\_DISABLED Macro

---

C

```
#define WND_DISABLED 0x0002 // Bit for disabled sta
```

## Description

---

Bit for disabled state

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## WND\_DRAW Macro

---

```
C  
#define WND_DRAW 0x6000 // Bits to indicate whole w.
```

### Description

---

Bits to indicate whole window must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_DRAW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WND\_DRAW\_CLIENT Macro

C

```
#define WND_DRAW_CLIENT 0x4000 // Bit to indicate c.
```

## Description

Bit to indicate client area must be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_DRAW\\_CLIENT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## WND\_DRAW\_TITLE Macro

---

C

```
#define WND_DRAW_TITLE 0x2000 // Bit to indicate ti
```

### Description

---

Bit to indicate title area must be redrawn

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_DRAW\\_TITLE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WND\_FOCUSED Macro

C

```
#define WND_FOCUSED 0x0001 // Bit for focus state
```

## Description

Bit for focus state

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_FOCUSED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## WND\_HIDE Macro

---

C

```
#define WND_HIDE 0x8000 // Bit to indicate window m
```

### Description

---

Bit to indicate window must be removed from screen

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_HIDE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WND\_TITLECENTER Macro

---

C

```
#define WND_TITLECENTER 0x0004 // Bit to center the
```

## Description

---

Bit to center the text on the Title Area

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#) > [WND\\_TITLECENTER Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WndCreate Function

## Window

```
C
WINDOW * WndCreate(
    WORD ID,
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    WORD state,
    void * pBitmap,
    XCHAR * pText,
    GOL_SCHEME * pScheme
);
```

## Overview

This function creates a **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.

## Input Parameters

Input Parameters	Description
WORD ID	Unique user defined ID for the object instance.
SHORT left	Left most position of the Object.
SHORT top	Top most position of the Object.
SHORT right	Right most position of the Object.
SHORT bottom	Bottom most position of the object.

WORD state	Sets the initial state of the object.
void * pBitmap	Pointer to the bitmap used in the title bar.
XCHAR * pText	Pointer to the text used as a title of the window.
GOL_SCHEME * pScheme	Pointer to the style scheme used.

## Returns

Returns the pointer to the object created

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
WINDOW *pWindow;  
pWindow = WndCreate(ID_WINDOW1,  
                   0, 0, GetMaxX(), GetMaxY(),  
                   WND_DRAW,  
                   (char*)myIcon,  
                   "Place Title Here.",  
                   NULL);  
  
if (pWindow == NULL)  
    return 0;  
WndDraw(pWindow);
```

```
return 1;
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [WndCreate Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WndDraw Function

## Window

```
C  
WORD WndDraw(  
    void * pObj  
);
```

## Overview

This function renders the object on the screen using the current parameter settings. 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.

## Input Parameters

Input Parameters	Description
pW	Pointer to the object to be rendered.

## Returns

Returns the status of the drawing

- 1 - If the rendering was completed and
- 0 - If the rendering is not yet finished.

Next call to the function will resume the rendering on the pending drawing state.

## Preconditions

---

Object must be created before this function is called.

## Side Effects

---

none

## Example

---

See [WndCreate\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [WndDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WndGetText Macro

## Window

```
C
#define WndGetText(pW) pW->pText
```

## Overview

This macro returns the address of the current text string used for the title bar.

## Input Parameters

Input Parameters	Description
pW	Pointer to the object

## Returns

Returns pointer to the text string being used.

## Preconditions

none

## Side Effects

none

## Example

```
WINDOW *pWindow;
XCHAR textUsed = "USE THIS!";
```

[Copy Code](#)

```
if (WndGetText(pWindow) == NULL)
    WndSetText(&textUsed);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [WndGetText Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WndSetText Function

## Window

```
C
void WndSetText(
    WINDOW * pW,
    XCHAR * pText
);
```

## Overview

This function sets the string used for the title bar.

## Input Parameters

Input Parameters	Description
WINDOW * pW	The pointer to the object whose text will be modified
XCHAR * pText	Pointer to the text that will be used

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

See [WndGetText\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [WndSetText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WndTranslateMsg Function

## Window

```
C  
WORD WndTranslateMsg(  
    void * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

Translated Message	Input Source	Events	Description
WND_MSG_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
WND_MSG_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

OBJ_MSG_INVALID	Any	Any	If the message did not affect the object.
-----------------	-----	-----	-------------------------------------------

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the message struct containing the message from the user interface.
pW	The pointer to the object where the message will be evaluated to check if the message will affect the object.

## Returns

Returns the translated message depending on the received GOL message:

- WND\_MSG\_TITLE – Title area is selected
- WND\_MSG\_CLIENT – Client area is selected
- OBJ\_MSG\_INVALID – [Window](#) is not affected

## Preconditions

none

## Side Effects

none

## Example

Usage is similar to [BtnTranslateMsg\(\)](#) example.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [WndTranslateMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WINDOW Structure

## Window

**C**

```
typedef struct {
    OBJ_HEADER hdr;
    SHORT textHeight;
    XCHAR * pText;
    void * pBitmap;
} WINDOW;
```

## Overview

The structure contains data for the window

## Members

Members	Description
OBJ_HEADER hdr;	Generic header for all Objects (see <a href="#">OBJ_HEADER</a> ).
SHORT textHeight;	Pre-computed text height
XCHAR * pText;	Pointer to the title text
void * pBitmap;	Pointer to the bitmap for the title bar

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [WINDOW Structure](#)

# Object States

## Macros | Topics

The GOL objects follow two types of states, the Property States and the Drawing States. Property States defines action and appearance of objects. Drawing States on the other hand indicate if the object needs to be hidden, partially redrawn, or fully redrawn in the display. To store the states, the field state defined in [OBJ\\_HEADER](#) structure is used. Six most significant bits are allocated for Drawing States and the rest is allocated for the Property States. Some common Property States and Drawing States are shown in the following table.

State	Type	Bit Location	Description
OBJ_FOCUSED	P	0x0001	Object is in the focused state. This is usually used to show selection of the object. Not all objects have this feature.
OBJ_DISABLED	P	0x0002	Object is disabled and will ignore all messages.
OBJ_DRAW_FOCUS	D	0x2000	Focus for the object should be redrawn.
OBJ_DRAW	D	0x4000	Object should be redrawn completely.
OBJ_HIDE	D	0x8000	Object will be hidden by filling the area occupied by the object with the common background

			color. This has the highest priority over all Drawing States. When an object is set to be hidden, all other drawing states are overridden.
--	--	--	--------------------------------------------------------------------------------------------------------------------------------------------

Where:

- OBJ – represent the prefix assigned to a GOL object.
- P – Property states, D – Drawing states

Individual Object drawing function (e.g. [BtnDraw\(\)](#), [SldDraw\(\)](#), etc...) does not reset the draw states instead use [GOLDraw\(\)](#) to automatically reset and manage the draw states. If the call to individual drawing function cannot be avoided, draw states must be reset manually after the drawing functions returns a 1.

## Macros

Name	Description
<a href="#">GetState</a>	This macro retrieves the current value of the state bits of an object. It is possible to get several state bits.
<a href="#">ClrState</a>	This macro clear the state bits of an object. Object must be redrawn to display the changes. It is possible to clear several state bits with this macro.
<a href="#">SetState</a>	This macro sets the state bits of an object. Object must be redrawn to display the changes. It is possible to set several state bits with this macro.

## Topics

Name	Description
<a href="#">Common Object States</a>	List of common Object bit states.

---

## Links

[Graphics Object Layer API](#), [Macros](#), [Topics](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Common Object States

## Macros

List of common Object bit states.

## Macros

Name	Description
<a href="#"><u>FOCUSED</u></a>	Focus state bit.
<a href="#"><u>DISABLED</u></a>	Disabled state bit.
<a href="#"><u>HIDE</u></a>	Object hide state bit. Object will be hidden from the screen by drawing over it the common background color.
<a href="#"><u>DRAW</u></a>	Object redraw state bit. The whole Object must be redrawn.
<a href="#"><u>DRAW_FOCUS</u></a>	Focus redraw state bit. The focus rectangle must be redrawn.
<a href="#"><u>DRAW_UPDATE</u></a>	Partial Object redraw state bit. A part or parts of of the Object must be redrawn to show updated state.

## Links

[Object States, Macros](#)

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#)

# FOCUSED Macro

---

```
C
```

```
#define FOCUSED 0x0001
```

## Description

---

Focus state bit.

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#) > [FOCUSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DISABLED Macro

---

```
C
```

```
#define DISABLED 0x0002
```

## Description

---

Disabled state bit.

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#) > [DISABLED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# HIDE Macro

```
C
```

```
#define HIDE 0x8000
```

## Description

Object hide state bit. Object will be hidden from the screen by drawing over it the common background color.

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#) > [HIDE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DRAW Macro

```
C
```

```
#define DRAW 0x7C00
```

## Description

Object redraw state bit. The whole Object must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#) > [DRAW Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DRAW\_FOCUS Macro

```
C
```

```
#define DRAW_FOCUS 0x2000
```

## Description

Focus redraw state bit. The focus rectangle must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#) > [DRAW\\_FOCUS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DRAW\_UPDATE Macro

```
C
```

```
#define DRAW_UPDATE 0x3C00
```

### Description

Partial Object redraw state bit. A part or parts of of the Object must be redrawn to show updated state.

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#) > [DRAW\\_UPDATE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GetState Macro

```
C
```

```
#define GetState(pObj, stateBits) (((OBJ_HEADER *)pObj)
```

### Overview

This macro retrieves the current value of the state bits of an object. It is possible to get several state bits.

### Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.
stateBits	Defines which state bits are requested. Please refer to specific objects for object state bits definition for details

### Returns

Macro returns a non-zero if any bit in the stateBits mask is set.

### Preconditions

none

### Side Effects

none

### Example

[Copy Code](#)

```
#define BTN_HIDE 0x8000
BUTTON *pB;
// pB is created and initialized
// do something here to set state

// Hide the button (remove from screen)
if (GetState(pB,BTN_HIDE)) {
    SetColor(pB->pGolScheme->CommonBkColor);
    Bar(pB->left,pB->top,pB->right,pB->bottom);
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [GetState Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ClrState Macro

```
C
```

```
#define ClrState(pObj, stateBits) ((OBJ_HEADER *)pObj
```

### Overview

This macro clear the state bits of an object. Object must be redrawn to display the changes. It is possible to clear several state bits with this macro.

### Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.
stateBits	Defines which state bits are to be cleared. Please refer to specific objects for object state bits definition for details

### Returns

none

### Preconditions

none

### Side Effects

none

### Example

See example for [SetState\(\)](#).

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [ClrState Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetState Macro

```
C
```

```
#define SetState(pObj, stateBits) ((OBJ_HEADER *)pObj
```

## Overview

This macro sets the state bits of an object. Object must be redrawn to display the changes. It is possible to set several state bits with this macro.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.
stateBits	Defines which state bits are to be set. Please refer to specific objects for object state bits definition for details

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
void BtnMsgDefault(WORD msg, BUTTON* pB){
    switch(msg){
        case BTN_MSG_PRESSED:
            // set pressed and redraw
            SetState(pB, BTN_PRESSED|BTN_DRAW);
            break;
        case BTN_MSG_RELEASED:
            ClearState(pB, BTN_PRESSED);           // res
            SetState(pB, BTN_DRAW);                // red
            break;
    }
}
```

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [SetState Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Object Management

## Functions | Macros

This section describes the API functions and macros that are used to create, maintain and render individual and list of objects.

### Functions

	Name	Description
	<a href="#"><u>GOLAddObject</u></a>	This function adds an object to the tail of the active list pointed to by <a href="#"><u>_pGolObjects</u></a> . The new list tail is set to point to NULL.
	<a href="#"><u>GOLFindObject</u></a>	This function finds an object in the active list pointed to by <a href="#"><u>_pGolObjects</u></a> using the given object ID.
	<a href="#"><u>GOLRedrawRec</u></a>	This function marks all objects in the active list intersected by the given rectangular area to be redrawn.
	<a href="#"><u>GOLDraw</u></a>	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. <a href="#"><u>GOLDrawCallback()</u></a> function is called by GOLDraw() when drawing of objects in the active list is completed.

	<p><u><a href="#">GOLDrawCallback</a></u></p>	<p>GOLDrawCallback() function MUST BE implemented by the user. This is called inside the <u><a href="#">GOLDraw()</a></u> function when the drawing of objects in the active list is completed. User drawing must be done here. Drawing color, line type, clipping region, graphic cursor position and current font will not be changed by GOL if this function returns a zero. To pass drawing control to GOL this function must return a non-zero value. If GOL messaging is not using the active link list, it is safe to modify the list here.</p>
	<p><u><a href="#">GOLFree</a></u></p>	<p>This function frees all the memory used by objects in the active list and initializes <u><a href="#">_pGolObjects</a></u> pointer to NULL to start a new empty list. This function must be called only inside the <u><a href="#">GOLDrawCallback()</a></u>function when using <u><a href="#">GOLDraw()</a></u> and <u><a href="#">GOLMsg()</a></u> functions. This requirement assures that primitive rendering settings are not altered by the rendering state machines of the objects.</p>
	<p><u><a href="#">GOLDeleteObject</a></u></p>	<p>deletes an object to the linked list objects for the current screen.</p>
	<p><u><a href="#">GOLDeleteObjectByID</a></u></p>	<p>Deletes an object in the current active linked list of objects using</p>

		the ID parameter of the object.
◆	<u><a href="#">GOLSetFocus</a></u>	This function sets the keyboard input focus to the object. If the object cannot accept keyboard messages focus will not be changed. This function resets <u><a href="#">FOCUSED</a></u> state for the object was in focus previously, set <u><a href="#">FOCUSED</a></u> state for the required object and marks the objects to be redrawn.
◆	<u><a href="#">GOLInit</a></u>	This function initializes the graphics library and creates a default style scheme with default settings referenced by the global scheme pointer. GOLInit() function must be called before GOL functions can be used. It is not necessary to call GraphInit() function if this function is used.
◆	<u><a href="#">GOLCanBeFocused</a></u>	This function returns non-zero if the object can be focused. Only button, check box, radio button, slider, edit box, list box, scroll bar can accept focus. If the object is disabled it cannot be set to focused state.
◆	<u><a href="#">GOLGetFocusNext</a></u>	This function returns the pointer to the next object in the active linked list which is able to receive keyboard input.
◆	<u><a href="#">GOLGetFocusPrev</a></u>	This function returns the pointer to the previous object in the

		active linked list which is able to receive keyboard input.
	<a href="#"><u>GOLPanelDrawTsk</u></a>	This function draws a panel on the screen with parameters set by <a href="#"><u>GOLPanelDraw()</u></a> macro. This function must be called repeatedly (depending on the return value) for a successful rendering of the panel.
	<a href="#"><u>GOLTwoTonePanelDrawTsk</u></a>	This function draws a two tone panel on the screen with parameters set by <a href="#"><u>GOLPanelDraw()</u></a> macro. This function must be called repeatedly (depending on the return value) for a successful rendering of the panel.

## Macros

Name	Description
<a href="#"><u>GOLRedraw</u></a>	This macro 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 to be the active list.
<a href="#"><u>GOLDrawComplete</u></a>	This macro resets the drawing states of the object (6 MSBits of the object's state).
<a href="#"><u>GetObjType</u></a>	This macro returns the object type.
<a href="#"><u>GetObjID</u></a>	This macro returns the object ID.

<a href="#"><u>GetObjNext</u></a>	This macro returns the next object after the specified object.
<a href="#"><u>GOLNewList</u></a>	This macro starts a new linked list of objects and resets the keyboard focus to none. This macro assigns the current active list <a href="#"><u>_pGoObjects</u></a> and current receiving keyboard input <a href="#"><u>_pObjectFocused</u></a> object pointers to NULL. Any keyboard inputs at this point will be ignored. Previous active list must be saved in another pointer if to be referenced later. If not needed anymore memory used by that list should be freed by <a href="#"><u>GOLFree()</u></a> function.
<a href="#"><u>GOLGetList</u></a>	This macro gets the current active list.
<a href="#"><u>GOLSetList</u></a>	This macro sets the given object list as the active list and resets the keyboard focus to none. This macro assigns the receiving keyboard input object <a href="#"><u>_pObjectFocused</u></a> pointer to NULL. If the new active list has an object's state set to focus, the <a href="#"><u>_pObjectFocused</u></a> pointer must be set to this object or the object's state must be change to unfocused. This is to avoid two objects displaying a focused state when only one object in the active list must be set to a focused state at anytime.
<a href="#"><u>IsObjUpdated</u></a>	This macro tests if the object is pending to be redrawn. This is done by testing the 6 MSBits of object's state to detect if the object must be redrawn.
<a href="#"><u>GOLGetFocus</u></a>	This macro returns the pointer to the current object receiving keyboard input.

[GOLPanelDraw](#)

This is macro GOLPanelDraw.

## Links

---

[Functions](#), [Graphics Object Layer API](#), [Legend](#), [Macros](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLAddObject Function

C

```
void GOLAddObject(  
    OBJ_HEADER * object  
);
```

## Overview

This function adds an object to the tail of the active list pointed to by [\\_pGolObjects](#). The new list tail is set to point to NULL.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object to be added on the current active list.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

```
Copy Code  
void MoveObject(OBJ_HEADER *pSrcList, OBJ_HEADER *p
```

```
OBJ_HEADER *pObjtoMove) {
OBJ_HEADER *pTemp = pSrcList;

if(pTemp != pObjtoMove) {
    while(pTemp->pNxtObj != pObjtoMove)
        pTemp = pTemp->pNxtObj;
}

pTemp->pNxtObj = pObjtoMove ->pNxt; // remove o
GOLSetList(pDstList);             // destinat
GOLAddObject(pObjtoMove);         // add obje
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GOLAddObject Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLFindObject Function

```
C
OBJ_HEADER * GOLFindObject(
    WORD ID
);
```

## Overview

This function finds an object in the active list pointed to by [\\_pGolObjects](#) using the given object ID.

## Input Parameters

Input Parameters	Description
WORD ID	User assigned value set during the creation of the object.

## Returns

Pointer to the object with the given ID.

## Preconditions

none

## Side Effects

none

## Example

```
Copy Code
void CopyObject(OBJ_HEADER *pSrcList, OBJ_HEADER *p
```

```
{
    OBJ_HEADER *pTemp;

    pTemp = GOLFindObject(ID);           // find
    if (pTemp != NULL) {
        GOLSetList(pDstList);          // dest
        GOLAddObject(pObj);            // add
    }
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GOLFindObject Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLRedraw Macro

```
C
```

```
#define GOLRedraw(pObj) ((OBJ_HEADER *)pObj)->state
```

## Overview

This macro 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 to be the active list.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object to be redrawn.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

```
Copy Code  
void GOLRedrawRec(SHORT left, SHORT top, SHORT right  
    // set all objects encompassed by the rectangle
```

```
OBJ_HEADER *pCurrentObj;

pCurrentObj = GOLGetList();
while(pCurrentObj != NULL){
    if (
        ((pCurrentObj->left >= left) && (pCurre
        ((pCurrentObj->right >= left) && (pCurr
        ((pCurrentObj->top >= top) && (pCurrent
        ((pCurrentObj->bottom >= top) && (pCurr
            GOLRedraw(pCurrentObj);
        }
        pCurrentObj = pCurrentObj->pNxtObj;
    }//end of while
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLRedraw Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLRedrawRec Function

C

```
void GOLRedrawRec(  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom  
);
```

## Overview

This function marks all objects in the active list intersected by the given rectangular area to be redrawn.

## Input Parameters

Input Parameters	Description
SHORT left	Defines the left most border of the rectangle area.
SHORT top	Defines the top most border of the rectangle area.
SHORT right	Defines the right most border of the rectangle area.
SHORT bottom	Defines the bottom most border of the rectangle area.

## Returns

none

## Preconditions

---

none

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
OBJ_HEADER *pTemp;
OBJ_HEADER *pAllObjects;

// assume *pAllObjects points to a list of all exist
// created and initialized

// mark all objects inside the rectangle to be redrawn
GOLRedrawRec(10,10,100,100);

pTemp = pAllObjects;
GOLStartNewList(); // reset ac
while(pTemp->pNxtObj != NULL) {
    if (pTemp->state&0x7C00) // add only
        GOLAddObject(pTemp); // redrawn
    pTemp = pTemp->pNxtObj;
}
GOLDraw(); // redraw a
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLRedrawRec Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLDDraw Function

**C**`WORD GOLDDraw();`

## Overview

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. [GOLDDrawCallback\(\)](#) function is called by GOLDDraw() when drawing of objects in the active list is completed.

## Returns

Non-zero if the active link list drawing is completed.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
// Assume objects are created & states are set to d
while(1){
    if(GOLDDraw()){ // parse active lis

        // here GOL drawing is completed
        // it is safe to modify objects states and
```

```
    TouchGetMsg(&msg);           // evaluate message
    GOLMsg(&msg);                // evaluate each ob
}
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GOLDraw Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLDDrawComplete Macro

```
C
```

```
#define GOLDDrawComplete(pObj) ((OBJ_HEADER *)pObj)->
```

## Overview

This macro resets the drawing states of the object (6 MSBits of the object's state).

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
// This function should be called again whenever an  
// rendering (done = 0) of an object occurs.  
// internal states in the BtnDraw() or WndDraw() sh  
// on the state where it left off to continue rende
```

```

void GOLDraw() {
    static OBJ_HEADER *pCurrentObj = NULL;
    SHORT done;

    if(pCurrentObj == NULL) {
        if(GOLDrawCallback()) {
            // If it's last object jump to head
            pCurrentObj = GOLGetList();
        } else {
            return;
        }
    }
    done = 0;

    while(pCurrentObj != NULL) {
        if(IsObjUpdated(pCurrentObj)) {
            done = pCurrentObj->draw(pCurrentObj);

            if(done){
                GOLDrawComplete(pCurrentObj);
            }else{
                return;
            }
        }
        pCurrentObj = pCurrentObj->pNxtObj;
    }
}

```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLDrawComplete Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLDDrawCallback Function

```
C
```

```
WORD GOLDDrawCallback();
```

## Overview

GOLDDrawCallback() function MUST BE implemented by the user. This is called inside the [GOLDDraw\(\)](#) function when the drawing of objects in the active list is completed. User drawing must be done here. Drawing color, line type, clipping region, graphic cursor position and current font will not be changed by GOL if this function returns a zero. To pass drawing control to GOL this function must return a non-zero value. If GOL messaging is not using the active link list, it is safe to modify the list here.

## Returns

Return a one if [GOLDDraw\(\)](#) will have drawing control on the active list. Return a zero if user wants to keep the drawing control.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```

#define SIG_STATE_SET    0
#define SIG_STATE_DRAW  1
WORD GOLDrawCallback(){
    static BYTE state = SIG_STATE_SET;
    if(state == SIG_STATE_SET){
        // Draw the button with disabled colors
        GOLPanelDraw(SIG_PANEL_LEFT,SIG_PANEL_TOP,
                    SIG_PANEL_RIGHT,SIG_PANEL_BOTT
                    WHITE, altScheme->EmbossLtColor,
                    altScheme->EmbossDkColor,
                    NULL, GOL_EMBOSS_SIZE);

        state = SIG_STATE_DRAW;
    }

    if(!GOLPanelDrawTsk()){
        // do not return drawing control to GOL
        // drawing is not complete
        return 0;
    }else{
        state = SIG_STATE_SET;
        // return drawing control to GOL, drawing i
        return 1;
    }
}

```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLDrawCallback Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLFree Function

```
C
```

```
void GOLFree();
```

## Overview

This function frees all the memory used by objects in the active list and initializes `_pGolObjects` pointer to NULL to start a new empty list. This function must be called only inside the `GOLDrawCallback()` function when using `GOLDraw()` and `GOLMsg()` functions. This requirement assures that primitive rendering settings are not altered by the rendering state machines of the objects.

## Returns

none

## Preconditions

none

## Side Effects

All objects in the active list are deleted from memory.

## Example

[Copy Code](#)

```
void DeletePage(OBJ_HEADER *pPage) {
    OBJ_HEADER *pTemp;

    // assuming pPage is different from the current
    pTemp = GOLGetList();           // save the
```

```
GOLSetList(pPage);           // set list
GolFree();                   // pPage ob

GOLSetList(pTemp);          // restore
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GOLFree Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetObjType Macro

```
C
```

```
#define GetObjType(pObj) ((OBJ_HEADER *)pObj)->type
```

## Overview

This macro returns the object type.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.

## Returns

Returns the OBJ\_TYPE of the object.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GetObjType Macro](#)

# GetObjID Macro

C

```
#define GetObjID(pObj) ((OBJ_HEADER *)pObj)->ID
```

## Overview

This macro returns the object ID.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.

## Returns

Returns the ID of the object.

## Preconditions

none

## Side Effects

none

## Example

```
void UseOfGetObjID(OBJ_HEADER *pObj) {  
    WORD id;  
    switch(id = GetObjID(pObj)) {  
        case ID_WINDOW1:  
            // do something
```

[Copy Code](#)

```
    case ID_WINDOW2:  
        // do something else  
    case ID_WINDOW3:  
        // do something else  
    default:  
        // do something else  
    }  
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GetObjID Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GetObjNext Macro

C

```
#define GetObjNext(pObj) ((OBJ_HEADER *)pObj) ->pNextObj
```

### Overview

This macro returns the next object after the specified object.

### Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.

### Returns

Returns the pointer of the next object.

### Preconditions

none

### Side Effects

none

### Example

[Copy Code](#)

```
// This is the same example for the GetObjType() macro  
// We just replaced one line  
void RedrawButtons(void) {  
    OBJ_HEADER *pCurr;
```

```
pCurr = GOLGetList();           // get acti
while(pCurr->pNxtObj != NULL) {
    if (GetObjType(pCurr) == BUTTON)
        pCurr->state = BTN_DRAW; // set butt
    pCurr = GetObjNext(pCurr);   // Use of G
    // replaces the old line
}
GolDraw();                      // redraw a
                                // active l
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GetObjNext Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLDeleteObject Function

C

```
BOOL GOLDeleteObject(  
    OBJ_HEADER * object  
);
```

## Overview

deletes an object to the linked list objects for the current screen.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLDeleteObject Function](#)

# GOLDeleteObjectByID Function

C

```
BOOL GOLDeleteObjectByID(  
    WORD ID  
);
```

## Overview

Deletes an object in the current active linked list of objects using the ID parameter of the object.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLDeleteObjectByID Function](#)

## GOLNewList Macro

C

```
#define GOLNewList \  
    _pGolObjects = NULL; \  
    _pObjectFocused = NULL
```

### Overview

This macro starts a new linked list of objects and resets the keyboard focus to none. This macro assigns the current active list [\\_pGolObjects](#) and current receiving keyboard input [\\_pObjectFocused](#) object pointers to NULL. Any keyboard inputs at this point will be ignored. Previous active list must be saved in another pointer if to be referenced later. If not needed anymore memory used by that list should be freed by [GOLFree\(\)](#) function.

### Returns

none

### Preconditions

none

### Side Effects

This macro sets the focused object pointer ([\\_pObjectFocused](#)) to NULL.

### Example

```
OBJ_HEADER *pSave;
```

[Copy Code](#)

```
pSave = GOLGetList();           // save current li
GOLNewList();                   // start the new l
                                // current list is

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

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GOLNewList Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLGetList Macro

```
C
```

```
#define GOLGetList _pGolObjects
```

## Overview

This macro gets the current active list.

## Returns

Returns the pointer to the current active list.

## Preconditions

none

## Side Effects

none

## Example

See [GOLNewList\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLGetList Macro](#)

## GOLSetList Macro

C

```
#define GOLSetList(objsList) \
    _pGolObjects = objsList; \
    _pObjectFocused = NULL
```

### Overview

This macro sets the given object list as the active list and resets the keyboard focus to none. This macro assigns the receiving keyboard input object [\\_pObjectFocused](#) pointer to NULL. If the new active list has an object's state set to focus, the [\\_pObjectFocused](#) pointer must be set to this object or the object's state must be change to unfocused. This is to avoid two objects displaying a focused state when only one object in the active list must be set to a focused state at anytime.

### Input Parameters

Input Parameters	Description
objsList	The pointer to the new active list.

### Returns

none

### Preconditions

none

### Side Effects

This macro sets the focused object pointer ([\\_pObjectFocused](#)) to

NULL. Previous active list should be saved if needed to be referenced later. If not, use [GOLFree\(\)](#) function to free the memory used by the objects before calling GOLSetList().

## Example

[Copy Code](#)

```
OBJ_HEADER *pSave;
pSave = GOLGetList();           // save current list
GOLNewList();                  // start the new list
                                // current list is

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

// do something here on the new list
// return the old list
GOLSetList(pSave);
```

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLSetList Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLSetFocus Function

```
C
void GOLSetFocus(
    OBJ_HEADER * object
);
```

## Overview

This function sets the keyboard input focus to the object. If the object cannot accept keyboard messages focus will not be changed. This function resets **FOCUSED** state for the object was in focus previously, set **FOCUSED** state for the required object and marks the objects to be redrawn.

## Input Parameters

Input Parameters	Description
OBJ_HEADER * object	Pointer to the object of interest.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >

## GOLSetFocus Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# IsObjUpdated Macro

```
C
```

```
#define IsObjUpdated(pObj) (((OBJ_HEADER *)pObj)->sta
```

## Overview

This macro tests if the object is pending to be redrawn. This is done by testing the 6 MSBits of object's state to detect if the object must be redrawn.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.

## Returns

Returns a nonzero value if the object needs to be redrawn. Zero if not.

## Preconditions

none

## Side Effects

none

## Example

```
int DrawButtonWindowOnly() {  
    static OBJ_HEADER *pCurrentObj = NULL;
```

[Copy Code](#)

```

SHORT done = 0;

if (pCurrentObj == NULL)
    pCurrentObj = GOLGetList();           // get

while(pCurrentObj != NULL){
    if(IsObjUpdated(pCurrentObj)){
        done = pCurrentObj->draw(pCurrentObj);

        // reset state of object if done
        if (done)
            GOLDrawComplete(pCurrentObj)
        // Return if not done. This means that
        // was terminated prematurely by device
        // and must be recalled to finish rende
        // objects in the list that have new st
        else
            return 0;
    }
    // go to the next object in the list
    pCurrentObj = pCurrentObj->pNxtObj;
}
return 1;
}
}

```

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [IsObjUpdated Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLInit Function

```
C
```

```
void GOLInit();
```

## Overview

This function initializes the graphics library and creates a default style scheme with default settings referenced by the global scheme pointer. GOLInit() function must be called before GOL functions can be used. It is not necessary to call GraphInit() function if this function is used.

## Returns

none

## Preconditions

none

## Side Effects

This sets the line type to [SOLID\\_LINE](#), sets the screen to all [BLACK](#), sets the current drawing color to [WHITE](#), sets the graphic cursor position to upper left corner of the screen, sets active and visual pages to page #0, clears the active page and disables clipping. This also creates a default style scheme.

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLInit Function](#)

# GOLGetFocus Macro

```
C
```

```
#define GOLGetFocus _pObjectFocused
```

## Overview

This macro returns the pointer to the current object receiving keyboard input.

## Returns

Returns the pointer to the object receiving keyboard input. If there is no object in focus NULL is returned.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLGetFocus Macro](#)

# GOLCanBeFocused Function

```
C  
WORD GOLCanBeFocused(  
    OBJ_HEADER * object  
);
```

## Overview

This function returns non-zero if the object can be focused. Only button, check box, radio button, slider, edit box, list box, scroll bar can accept focus. If the object is disabled it cannot be set to focused state.

## Input Parameters

Input Parameters	Description
OBJ_HEADER * object	Pointer to the object of interest.

## Returns

This returns a non-zero if the object can be focused and zero if not.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >

## GOLCanBeFocused Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLGetFocusNext Function

```
C  
OBJ_HEADER * GOLGetFocusNext();
```

## Overview

This function returns the pointer to the next object in the active linked list which is able to receive keyboard input.

## Returns

This returns the pointer of the next object in the active list capable of receiving keyboard input. If there is no object capable of receiving keyboard inputs (i.e. none can be focused) NULL is returned.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLGetFocusNext Function](#)

# GOLGetFocusPrev Function

```
C
```

```
OBJ_HEADER * GOLGetFocusPrev();
```

## Overview

This function returns the pointer to the previous object in the active linked list which is able to receive keyboard input.

## Returns

This returns the pointer of the previous object in the active list capable of receiving keyboard input. If there is no object capable of receiving keyboard inputs (i.e. none can be focused) NULL is returned.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLGetFocusPrev Function](#)

# GOLPanelDraw Macro

C

```
#define GOLPanelDraw(left, top, right, bottom, radius,  
    _rpnlX1 = left;  
    _rpnlY1 = top;  
    _rpnlX2 = right;  
    _rpnlY2 = bottom;  
    _rpnlR = radius;  
    _rpnlFaceColor = faceClr;  
    _rpnlEmbossLtColor = embossLtClr;  
    _rpnlEmbossDkColor = embossDkClr;  
    _pRpnlBitmap = pBitmap;  
    _rpnlEmbossSize = embossSize;
```

## Description

This is macro GOLPanelDraw.

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) >  
[GOLPanelDraw Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLPanelDrawTsk Function

```
C
```

```
WORD GOLPanelDrawTsk();
```

## Overview

This function draws a panel on the screen with parameters set by [GOLPanelDraw\(\)](#) macro. This function must be called repeatedly (depending on the return value) for a successful rendering of the panel.

## Returns

Returns the status of the panel rendering

[Copy Code](#)

```
0 ½ Rendering of the panel is not yet fini  
1 ½ Rendering of the panel is finished.
```

## Preconditions

Parameters of the panel must be set by [GOLPanelDraw\(\)](#) macro.

## Side Effects

none

## Example

See [GOLPanelDraw\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLPanelDrawTsk Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLTwoTonePanelDrawTsk Function

```
C
```

```
WORD GOLTwoTonePanelDrawTsk();
```

## Overview

This function draws a two tone panel on the screen with parameters set by [GOLPanelDraw\(\)](#) macro. This function must be called repeatedly (depending on the return value) for a successful rendering of the panel.

## Returns

Returns the status of the panel rendering

[Copy Code](#)

```
0 ½ Rendering of the panel is not yet fini  
1 ½ Rendering of the panel is finished.
```

## Preconditions

Parameters of the panel must be set by [GOLPanelDraw\(\)](#) macro.

## Side Effects

none

## Example

Usage is similar to [GOLPanelDraw\(\)](#) example.

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#) > [GOLTwoTonePanelDrawTsk Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Messages

[Enumerations](#) | [Functions](#) | [Structures](#) | [Topics](#)

To facilitate the processing of user actions on the objects, messaging are used.

User passes messages from the input devices to GOL using the GOL message structure. The structure is described by the following table.

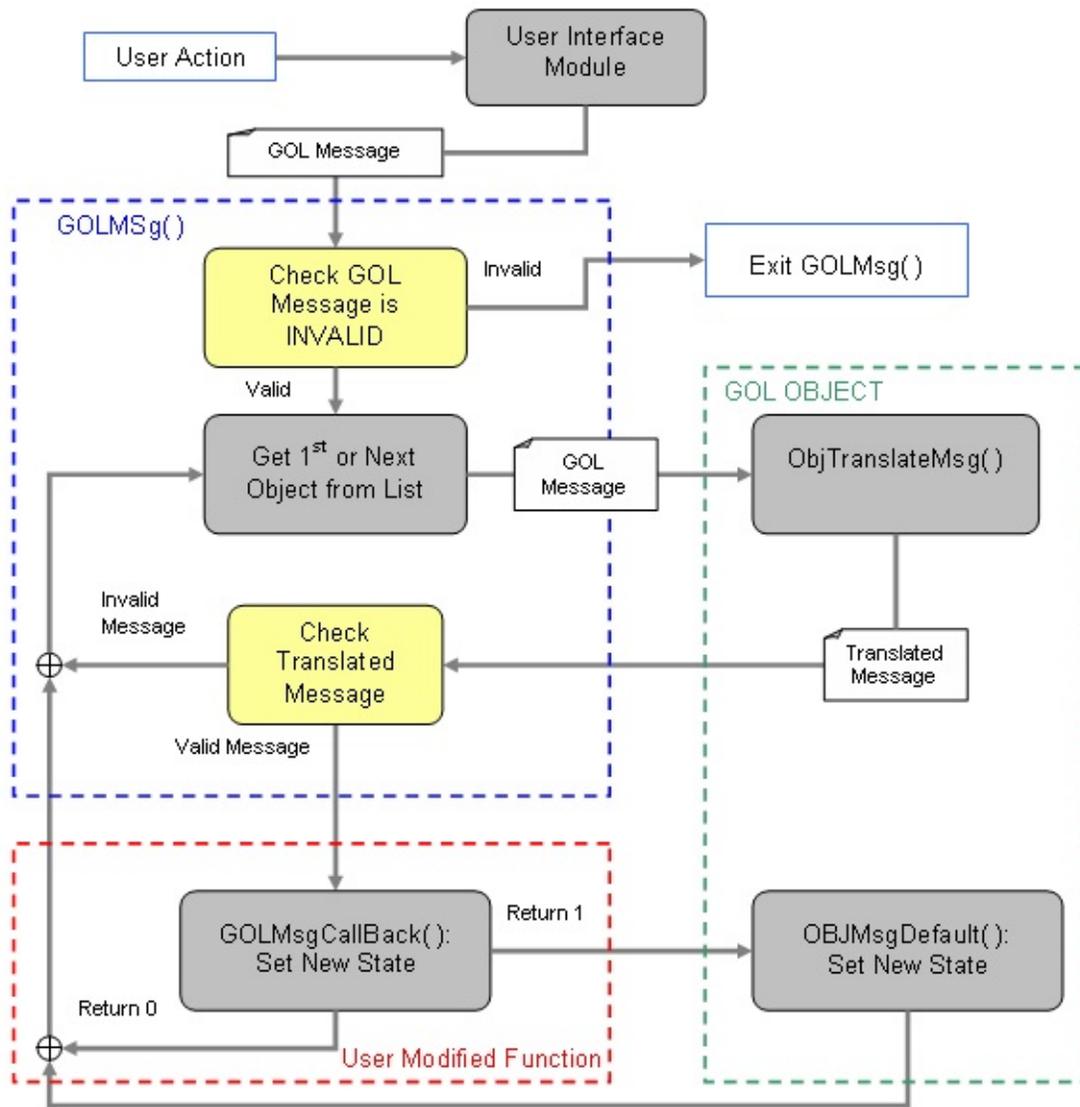
[Copy Code](#)

```
typedef struct {
    BYTE      type;
    BYTE      uiEvent;
    int       param1;
    int       param2;
} GOL_MSG;
```

Field	Description
type	Defines the type of device where the message was created. These are the devices implemented in the User Interface Layer. Possible device types are the following: TYPE_UNKNOWN TYPE_KEYBOARD TYPE_TOUCHSCREEN TYPE_MOUSE
uiEvent	Event ID of the user or device type action on the object. Possible event IDs are the following: EVENT_INVALID EVENT_MOVE EVENT_PRESS EVENT_STILLPRESS EVENT_RELEASE

	EVENT_KEYSCAN EVENT_CHARCODE
param1 param2	Parameters 1 and 2 definition varies from device types. For example, param1 and param2 are defined as x-coordinate position and y-coordinate position respectively for TYPE_TOUCHSCREEN. For TYPE_KEYBOARD, param1 is defined as the ID of the receiving object and param2 is defined as the keyboard scan or character code.

GOLMsg() function accepts this structure and processes the message for all objects in the active list.



## Messaging Flow

The messaging mechanism follows this flow:

1. User Interface Module (touch screen, keypad) sends a GOL message (the **GOL\_MSG** structure).
2. A loop evaluates which object is affected by the message. This is done inside **GOLMSG()** function.
3. Affected object returns the translated message based on the GOL message parameters.
4. User can change default action with the callback function. If the call

back function returns a non-zero value message will be processed by default.

5. Object should be redrawn to reflect new state.

The translated message is a set of actions unique to each object type. Please refer to each object translated message ID for details.

Objects that are disabled will not accept any messages.

[GOLMsg\(\)](#) function must be called when GOL drawing is completed. In this case all objects have been drawn and it is safe to change objects states. [GOLMsg\(\)](#) call can be done if [GOLDraw\(\)](#) function returns non-zero or inside [GOLDrawCallback\(\)](#) function.

## Enumerations

Name	Description
<a href="#">TRANS_MSG</a>	This structure defines the list of translated messages for GOL Objects used in the library.
<a href="#">INPUT_DEVICE_EVENT</a>	This structure defines the types of GOL message events used in the library.
<a href="#">INPUT_DEVICE_TYPE</a>	This structure defines the types of input devices used in the library.

## Functions

	Name	Description
	<a href="#">GOLMsg</a>	This function receives a GOL message from user and loops through the active list of objects to check which object is affected by the message. For affected

		<p>objects the message is translated and <a href="#">GOLMsgCallback()</a> 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.</p> <p>This function should be called when GOL drawing is completed. It can be done... <a href="#">more</a></p>
	<p><a href="#">GOLMsgCallback</a></p>	<p>The user MUST implement this function. <a href="#">GOLMsg()</a> calls this function when a valid message for an object in the active list is received. User action for the message should be implemented here. If this function returns non-zero, the message for the object will be processed by default. If zero is returned, GOL will not perform any action.</p>

## Structures

Name	Description
<p><a href="#">GOL_MSG</a></p>	<p>This structure defines the GOL message used in the library.</p> <ul style="list-style-type: none"> <li>The types must be one of the <a href="#">INPUT_DEVICE_TYPE</a>: <ul style="list-style-type: none"> <li>TYPE_UNKNOWN</li> <li>TYPE_KEYBOARD</li> <li>TYPE_TOUCHSCREEN</li> <li>TYPE_MOUSE</li> </ul> </li> <li>uiEvent must be one of the</li> </ul>

## 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

## Topics

Name	Description
<a href="#">Scan Key Codes</a>	The defined scan codes for AT keyboard.

## Links

[Enumerations](#), [Functions](#), [Graphics Object Layer API](#), [Legend](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#)

# GOLMsg Function

```
C
void GOLMsg(
    GOL_MSG * pMsg
);
```

## Overview

This function receives a GOL 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 [GOLMsgCallback\(\)](#) 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.

This function should be called when GOL drawing is completed. It can be done when [GOLDraw\(\)](#) returns non-zero value or inside [GOLDrawCallback\(\)](#) function.

## Input Parameters

Input Parameters	Description
GOL_MSG * pMsg	Pointer to the GOL message from user.

## Returns

none

## Preconditions

none

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
// Assume objects are created & states are set to d  
while(1){  
    if(GOLDraw()){  
        // GOL drawing is completed here  
        // it is safe to change objects  
        TouchGetMsg(&msg);           // from user int  
        GOLMsg(&msg);  
    }  
}
```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [GOLMsg Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLMsgCallback Function

C

```
WORD GOLMsgCallback(  
    WORD objMsg,  
    OBJ_HEADER * pObj,  
    GOL_MSG * pMsg  
);
```

## Overview

The user MUST implement this function. [GOLMsg\(\)](#) calls this function when a valid message for an object in the active list is received. User action for the message should be implemented here. If this function returns non-zero, the message for the object will be processed by default. If zero is returned, GOL will not perform any action.

## Input Parameters

Input Parameters	Description
WORD objMsg	Translated message for the object or the action ID response from the object.
OBJ_HEADER * pObj	Pointer to the object that processed the message.
GOL_MSG * pMsg	Pointer to the GOL message from user.

## Returns

Return a non-zero if the message will be processed by default. If a zero is returned, the message will not be processed by GOL.

## Preconditions

---

none

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER* pObj,
    static char focusSwitch = 1;

    switch(GetObjID(pObj)){
        case ID_BUTTON1:
            // Change text and focus state
            if(objMsg == BTN_MSG_RELEASED){
                focusSwitch ^= 1;
                if(focusSwitch){
                    BtnSetText((BUTTON*)pObj, "Focu
                    SetState(pObj, BTN_FOCUSED);
                }else{
                    BtnSetText((BUTTON*)pObj, "Unfo
                    ClrState(pObj, BTN_FOCUSED);
                }
            }
            // Process by default
            return 1;
        case ID_BUTTON2:
            // Change text
            if(objMsg == BTN_MSG_PRESSED){
                BtnSetText((BUTTON*)pObj, "Pressed"
            }
            if(objMsg == BTN_MSG_RELEASED){
                BtnSetText((BUTTON*)pObj, "Released"
            }
    }
```

```

        // Process by default
        return 1;
    case ID_BUTTON3:
        // Change face picture
        if(objMsg == BTN_MSG_PRESSED){
            BtnSetBitmap(pObj, arrowLeft);
        }
        if(objMsg == BTN_MSG_RELEASED){
            BtnSetBitmap(pObj, (char*)arrowRight);
        }
        // Process by default
        return 1;
    case ID_BUTTON_NEXT:
        if(objMsg == BTN_MSG_RELEASED){
            screenState = CREATE_CHECKBOXES;
        }
        // Process by default
        return 1;
    case ID_BUTTON_BACK:
        return 1;
    default:
        return 1;
}
}

```

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [GOLMsgCallback Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GOL\_MSG Structure

C

```
typedef struct {  
    BYTE type;  
    BYTE uiEvent;  
    SHORT param1;  
    SHORT param2;  
} GOL_MSG;
```

### Overview

This structure defines the GOL message 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

## Members

---

Members	Description
BYTE type;	Type of input device.
BYTE uiEvent;	The generic events for input device.
SHORT param1;	Parameter 1 meaning is dependent on the type of input device.
SHORT param2;	Parameter 2 meaning is dependent on the type of input device.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [GOL\\_MSG Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TRANS\_MSG Enumeration

C

```
typedef enum {  
    OBJ_MSG_INVALID = 0,  
    CB_MSG_CHECKED,  
    CB_MSG_UNCHECKED,  
    RB_MSG_CHECKED,  
    WND_MSG_CLIENT,  
    WND_MSG_TITLE,  
    BTN_MSG_PRESSED,  
    BTN_MSG_STILLPRESSED,  
    BTN_MSG_RELEASED,  
    BTN_MSG_CANCELPRESS,  
    PICT_MSG_SELECTED,  
    GB_MSG_SELECTED,  
    CC_MSG_SELECTED,  
    SLD_MSG_INC,  
    SLD_MSG_DEC,  
    ST_MSG_SELECTED,  
    DM_MSG_SELECTED,  
    PB_MSG_SELECTED,  
    RD_MSG_CLOCKWISE,  
    RD_MSG_CTR_CLOCKWISE,  
    MTR_MSG_SET,  
    EB_MSG_CHAR,  
    EB_MSG_DEL,  
    EB_MSG_TOUCHSCREEN,  
    LB_MSG_SEL,  
    LB_MSG_MOVE,  
    LB_MSG_TOUCHSCREEN,  
    GRID_MSG_TOUCHED,  
    GRID_MSG_ITEM_SELECTED,  
    GRID_MSG_UP,  
    GRID_MSG_DOWN,  
}
```

```

GRID_MSG_LEFT,
GRID_MSG_RIGHT,
CH_MSG_SELECTED,
TE_MSG_RELEASED,
TE_MSG_PRESSED,
TE_MSG_ADD_CHAR,
TE_MSG_DELETE,
TE_MSG_SPACE,
TE_MSG_ENTER,
AC_MSG_PRESSED,
AC_MSG_RELEASED,
OBJ_MSG_PASSIVE
} TRANS_MSG;

```

## Overview

This structure defines the list of translated messages for GOL Objects used in the library.

## Members

Members	Description
OBJ_MSG_INVALID = 0	Invalid message response.
CB_MSG_CHECKED	Check Box check action ID.
CB_MSG_UNCHECKED	Check Box un-check action ID.
RB_MSG_CHECKED	Radio <a href="#">Button</a> check action ID.
WND_MSG_CLIENT	<a href="#">Window</a> client area selected action ID.
WND_MSG_TITLE	<a href="#">Window</a> title bar selected action ID.
BTN_MSG_PRESSED	<a href="#">Button</a> pressed action ID.

BTN_MSG_STILLPRESSED	<a href="#">Button</a> is continuously pressed ID.
BTN_MSG_RELEASED	<a href="#">Button</a> released action ID.
BTN_MSG_CANCELPRESS	<a href="#">Button</a> released action ID with button press canceled.
PICT_MSG_SELECTED	<a href="#">Picture</a> selected action ID.
GB_MSG_SELECTED	Group Box selected action ID.
CC_MSG_SELECTED	Custom Control selected action ID.
SLD_MSG_INC	<a href="#">Slider</a> or <a href="#">Scroll Bar</a> increment action ID.
SLD_MSG_DEC	<a href="#">Slider</a> or <a href="#">Scroll Bar</a> decrement action ID.
ST_MSG_SELECTED	Static Text selected action ID.
DM_MSG_SELECTED	Digital <a href="#">Meter</a> selected action ID.
PB_MSG_SELECTED	Progress <a href="#">Bar</a> selected action ID.
RD_MSG_CLOCKWISE	<a href="#">Dial</a> move clockwise action ID.
RD_MSG_CTR_CLOCKWISE	<a href="#">Dial</a> move counter clockwise action ID.
MTR_MSG_SET	<a href="#">Meter</a> set value action ID.
EB_MSG_CHAR	Edit Box insert character action ID.
EB_MSG_DEL	Edit Box remove character action ID.

EB_MSG_TOUCHSCREEN	Edit Box touchscreen selected action ID.
LB_MSG_SEL	List Box item select action ID.
LB_MSG_MOVE	List Box item move action ID.
LB_MSG_TOUCHSCREEN	List Box touchscreen selected action ID.
GRID_MSG_TOUCHED	<a href="#">Grid</a> item touched action ID.
GRID_MSG_ITEM_SELECTED	<a href="#">Grid</a> item selected action ID.
GRID_MSG_UP	<a href="#">Grid</a> up action ID.
GRID_MSG_DOWN	<a href="#">Grid</a> down action ID.
GRID_MSG_LEFT	<a href="#">Grid</a> left action ID.
GRID_MSG_RIGHT	<a href="#">Grid</a> right action ID.
CH_MSG_SELECTED	<a href="#">Chart</a> selected action ID
TE_MSG_RELEASED	TextEntry released action ID
TE_MSG_PRESSED	TextEntry pressed action ID
TE_MSG_ADD_CHAR	TextEntry add character action ID
TE_MSG_DELETE	TextEntry delete character action ID
TE_MSG_SPACE	TextEntry add space character action ID
TE_MSG_ENTER	TextEntry enter action ID

AC_MSG_PRESSED	Analog Clock Pressed Action
AC_MSG_RELEASED	Analog Clock Released Action
OBJ_MSG_PASSIVE	Passive message response. No change in object needed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [TRANS\\_MSG Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## INPUT\_DEVICE\_EVENT Enumeration

C

```
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;
```

### Overview

This structure defines the types of GOL message events used in the library.

### Members

Members	Description
EVENT_INVALID = 0	An invalid event.
EVENT_MOVE	A move event.
EVENT_PRESS	A press event.
EVENT_STILLPRESS	A continuous press event.
EVENT_RELEASE	A release event.
EVENT_KEYSCAN	A keyscan event, parameters has the object

	ID keyboard scan code.
EVENT_CHARCODE	Character code is presented at the parameters.
EVENT_SET	A generic set event.
EVENT_SET_STATE	A set state event.
EVENT_CLR_STATE	A clear state event.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [INPUT\\_DEVICE\\_EVENT Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# INPUT\_DEVICE\_TYPE Enumeration

C

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

## Overview

This structure defines the types of input devices used in the library.

## Members

Members	Description
TYPE_UNKNOWN = 0	Unknown device.
TYPE_KEYBOARD	Keyboard.
TYPE_TOUCHSCREEN	Touchscreen.
TYPE_MOUSE	Mouse.
TYPE_TIMER	Timer.
TYPE_SYSTEM	System Messages.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [INPUT\\_DEVICE\\_TYPE Enumeration](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Scan Key Codes

## Macros

The defined scan codes for AT keyboard.

## Macros

Name	Description
<a href="#"><u>SCAN_BS_PRESSED</u></a>	Back space key pressed.
<a href="#"><u>SCAN_BS_RELEASED</u></a>	Back space key released.
<a href="#"><u>SCAN_CR_PRESSED</u></a>	Carriage return pressed.
<a href="#"><u>SCAN_CR_RELEASED</u></a>	Carriage return released.
<a href="#"><u>SCAN_DEL_PRESSED</u></a>	Delete key pressed.
<a href="#"><u>SCAN_DEL_RELEASED</u></a>	Delete key released.
<a href="#"><u>SCAN_DOWN_PRESSED</u></a>	Down key pressed.
<a href="#"><u>SCAN_DOWN_RELEASED</u></a>	Down key released.
<a href="#"><u>SCAN_END_PRESSED</u></a>	End key pressed.
<a href="#"><u>SCAN_END_RELEASED</u></a>	End key released.
<a href="#"><u>SCAN_HOME_PRESSED</u></a>	Home key pressed.
<a href="#"><u>SCAN_HOME_RELEASED</u></a>	Home key released.
<a href="#"><u>SCAN_LEFT_PRESSED</u></a>	Left key pressed.
<a href="#"><u>SCAN_LEFT_RELEASED</u></a>	Left key released.
<a href="#"><u>SCAN_PGDOWN_PRESSED</u></a>	Page down key pressed.

<a href="#"><u>SCAN_PGDOWN_RELEASED</u></a>	Page down key released.
<a href="#"><u>SCAN_PGUP_PRESSED</u></a>	Page up key pressed.
<a href="#"><u>SCAN_PGUP_RELEASED</u></a>	Page up key released.
<a href="#"><u>SCAN_RIGHT_PRESSED</u></a>	Right key pressed.
<a href="#"><u>SCAN_RIGHT_RELEASED</u></a>	Right key released.
<a href="#"><u>SCAN_SPACE_PRESSED</u></a>	Space key pressed.
<a href="#"><u>SCAN_SPACE_RELEASED</u></a>	Space key released.
<a href="#"><u>SCAN_TAB_PRESSED</u></a>	Tab key pressed.
<a href="#"><u>SCAN_TAB_RELEASED</u></a>	Tab key released.
<a href="#"><u>SCAN_UP_PRESSED</u></a>	Up key pressed.
<a href="#"><u>SCAN_UP_RELEASED</u></a>	Up key released.

## Links

[GOL Messages](#), [Macros](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_BS\_PRESSED Macro

---

C

```
#define SCAN_BS_PRESSED 0x0E
```

### Description

---

Back space key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_BS\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_BS\_RELEASED Macro

```
C
```

```
#define SCAN_BS_RELEASED 0x8E
```

## Description

Back space key released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_BS\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_CR\_PRESSED Macro

---

C

```
#define SCAN_CR_PRESSED 0x1C
```

### Description

---

Carriage return pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_CR\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_CR\_RELEASED Macro

```
C
```

```
#define SCAN_CR_RELEASED 0x9C
```

## Description

Carriage return released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_CR\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_DEL\_PRESSED Macro

---

C

```
#define SCAN_DEL_PRESSED 0x53
```

## Description

---

Delete key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_DEL\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_DEL\_RELEASED Macro

```
C
```

```
#define SCAN_DEL_RELEASED 0xD3
```

## Description

Delete key released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_DEL\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_DOWN\_PRESSED Macro

---

C

```
#define SCAN_DOWN_PRESSED 0x50
```

### Description

---

Down key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_DOWN\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_DOWN\_RELEASED Macro

---

C

```
#define SCAN_DOWN_RELEASED 0xD0
```

### Description

---

Down key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_DOWN\\_RELEASED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_END\_PRESSED Macro

C

```
#define SCAN_END_PRESSED 0x4F
```

## Description

End key pressed.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_END\\_PRESSED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_END\_RELEASED Macro

---

C

```
#define SCAN_END_RELEASED 0xCF
```

### Description

---

End key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_END\\_RELEASED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_HOME\_PRESSED Macro

C

```
#define SCAN_HOME_PRESSED 0x47
```

## Description

Home key pressed.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_HOME\\_PRESSED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_HOME\_RELEASED Macro

```
C
```

```
#define SCAN_HOME_RELEASED 0xC7
```

## Description

Home key released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_HOME\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_LEFT\_PRESSED Macro

```
C
```

```
#define SCAN_LEFT_PRESSED 0x4B
```

### Description

Left key pressed.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_LEFT\\_PRESSED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_LEFT\_RELEASED Macro

C

```
#define SCAN_LEFT_RELEASED 0xCB
```

## Description

Left key released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_LEFT\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_PGDOWN\_PRESSED Macro

---

C

```
#define SCAN_PGDOWN_PRESSED 0x51
```

### Description

---

Page down key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_PGDOWN\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_PGDOWN\_RELEASED Macro

---

```
C
```

```
#define SCAN_PGDOWN_RELEASED 0xD1
```

### Description

---

Page down key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_PGDOWN\\_RELEASED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_PGUP\_PRESSED Macro

C

```
#define SCAN_PGUP_PRESSED 0x49
```

## Description

Page up key pressed.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_PGUP\\_PRESSED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_PGUP\_RELEASED Macro

---

```
C
```

```
#define SCAN_PGUP_RELEASED 0xC9
```

## Description

---

Page up key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_PGUP\\_RELEASED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_RIGHT\_PRESSED Macro

---

C

```
#define SCAN_RIGHT_PRESSED 0x4D
```

### Description

---

Right key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_RIGHT\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_RIGHT\_RELEASED Macro

C

```
#define SCAN_RIGHT_RELEASED 0xCD
```

## Description

Right key released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_RIGHT\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_SPACE\_PRESSED Macro

C

```
#define SCAN_SPACE_PRESSED 0x39
```

## Description

Space key pressed.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_SPACE\\_PRESSED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_SPACE\_RELEASED Macro

---

C

```
#define SCAN_SPACE_RELEASED 0xB9
```

## Description

---

Space key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_SPACE\\_RELEASED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_TAB\_PRESSED Macro

---

C

```
#define SCAN_TAB_PRESSED 0x0F
```

## Description

---

Tab key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_TAB\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_TAB\_RELEASED Macro

---

C

```
#define SCAN_TAB_RELEASED 0x8F
```

### Description

---

Tab key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_TAB\\_RELEASED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## SCAN\_UP\_PRESSED Macro

---

C

```
#define SCAN_UP_PRESSED 0x48
```

### Description

---

Up key pressed.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_UP\\_PRESSED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SCAN\_UP\_RELEASED Macro

```
C
```

```
#define SCAN_UP_RELEASED 0xC8
```

## Description

Up key released.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#) > [SCAN\\_UP\\_RELEASED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Style Scheme

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#) | [Variables](#)

All objects use a style scheme structure that defines the font and colors used. Upon the object's creation a user defined style scheme can be assigned to the object. In the absence of the user defined scheme, the default scheme is used.

[Copy Code](#)

```
typedef struct {  
    WORD EmbossDkColor;  
    WORD EmbossLtColor;  
    WORD TextColor0;  
    WORD TextColor1;  
    WORD TextColorDisabled;  
    WORD Color0;  
    WORD Color1;  
    WORD ColorDisabled;  
    WORD CommonBkColor;  
    BYTE *pFont;  
    BYTE AlphaValue;  
    GFX_GRADIENT_STYLE gradientScheme;  
} GOL_SCHEME;
```

Field	Description
EmbossDkColor	Dark emboss color used for the 3-D effect of the object.
EmbossLtColor	Light emboss color used for the 3-D effect of the object.
TextColor0 TextColor1	Generic text colors used by the objects. Usage may vary from one object type to another.

TextColorDisabled	Text color used for objects that are disabled.
Color0 Color1	Generic colors used to render objects. Usage may vary from one object type to another.
ColorDisabled	Color used to render objects that are disabled.
CommonBkColor	A common background color of objects. Typically used to hide objects from the screen.
pFont	Pointer to the font table used by the object.
AlphaValue	Alpha value used for alpha blending, this is only available only when <a href="#">USE_ALPHABLEND</a> is defined in the GraphicsConfig.h.
gradientScheme	Gradient Scheme for supported widgets, this is available only when <a href="#">USE_GRADIENT</a> is defined in the GraphicsConfig.h.

TextColorDisabled and ColorDisabled are used when the object is in the disabled state. Otherwise, TextColor0, TextColor1, Color0 and Color1 are used. When object Draw state is set to [HIDE](#), the CommonBkColor is used to fill area occupied by object.

Style scheme can be created with [GOLCreateScheme\(\)](#) function that returns a pointer to the newly created [GOL\\_SCHEME](#) structure with default values automatically assigned. The default settings of the style scheme for a 16bpp setup are shown below:

Style Parameter	Default Value
EmbossDkColor	EMBOSSDKCOLORDEFAULT
EmbossLtColor	EMBOSSLTCOLORDEFAULT
Textcolor0	TEXTCOLOR0DEFAULT
Textcolor1	TEXTCOLOR1DEFAULT

TextColorDisabled	TEXTCOLORDISABLEDDEFAULT
Color0	COLOR0DEFAULT
Color1	COLOR1DEFAULT
ColorDisabled	COLORDISABLEDDEFAULT
CommonBkColor	COMMONBACKGROUNDCOLORDEFAULT
pFont	<a href="#">FONTDEFAULT</a>
AlphaValue	0
gradientScheme	{ GRAD_NONE, <a href="#">RGBConvert</a> (0xA9, 0xDB, 0xEF), <a href="#">RGBConvert</a> (0x26, 0xC7, 0xF2), 50 }

The default values can be changed in the GOLSchemeDefault.c file.

The application code can define its own default style scheme by defining the macro GFX\_SCHEMEDEFAULT in the GraphicsConfig.h.

Then [GOL\\_SCHEME GOLSchemeDefault](#) must be defined in the application code with each structure member initialized to the desired values. See GOLSchemeDefault.c file for an example on how to initialize the style scheme.

## Functions

	Name	Description
	<a href="#">GOLCreateScheme</a>	This function creates a new style scheme object and initializes the parameters to default values. Default values are based on the <a href="#">GOLSchemeDefault</a> defined in

GOLSchemeDefault.c file. Application code can override this initialization, See [GOLSchemeDefault](#).

## Macros

Name	Description
<a href="#">GOLSetScheme</a>	This macro sets the GOL scheme to be used for the object.
<a href="#">GOLGetScheme</a>	This macro gets the GOL scheme used by the given object.
<a href="#">GOLGetSchemeDefault</a>	This macro returns the default GOL scheme pointer.
<a href="#">GOL_EMBOSS_SIZE</a>	This option defines the 3-D effect emboss size for objects. The default value of this is 3 set in GOL.h. If it is not defined in GraphicsConfig.h file then the default value is used.
<a href="#">RGBConvert</a>	This macro converts the color data to the selected <a href="#">COLOR_DEPTH</a> . This macro is only valid when <a href="#">COLOR_DEPTH</a> is 8, 16, or 24.

## Structures

Name	Description
<a href="#">GOL_SCHEME</a>	GOL scheme defines the style scheme to be used by an object.

## Topics

---

Name	Description
<a href="#">Default Style Scheme Settings</a>	Lists the default settings for the style scheme.

## Variables

---

Name	Description
<a href="#">GOLFontDefault</a>	This is variable GOLFontDefault.
<a href="#">GOLSchemeDefault</a>	This defines a default GOL scheme that gets populated when an application calls the <a href="#">GOLCreateScheme()</a> . The application can override this definition by defining the macro GFX_SCHEMEDEFAULT in the GraphicsConfig.h header file and defining GOLSchemeDefault structure in the application code. It is important to use the same structure name since the library assumes that this object exists and assigns the default style scheme pointer to this object.

## Links

---

[Functions](#), [Graphics Object Layer API](#), [Legend](#), [Macros](#),  
[Structures](#), [Topics](#), [Variables](#)

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

# GOLCreateScheme Function

C

```
GOL_SCHEME * GOLCreateScheme();
```

## Overview

This function creates a new style scheme object and initializes the parameters to default values. Default values are based on the [GOLSchemeDefault](#) defined in GOLSchemeDefault.c file. Application code can override this initialization, See [GOLSchemeDefault](#).

## Returns

Pointer to the new [GOL\\_SCHEME](#) created.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
extern const char Font22[] __attribute__((aligned(2)))  
extern const char Font16[] __attribute__((aligned(2)))  
  
GOL_SCHEME *pScheme1, *pScheme2;  
pScheme1 = GOLCreateScheme();  
pScheme2 = GOLCreateScheme();
```

```
pScheme1->pFont = (BYTE*)Font22;  
pScheme2->pFont = (BYTE*)Font16;
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) >  
[GOLCreateScheme Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLSetScheme Macro

```
C
```

```
#define GOLSetScheme(pObj, pScheme) ((OBJ_HEADER *)p
```

## Overview

This macro sets the GOL scheme to be used for the object.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.
pScheme	Pointer to the style scheme to be used.

## Returns

none

## Preconditions

none

## Side Effects

none

## Example

```
extern FONT_FLASH Gentium12;  
GOL_SCHEME *pScheme1;  
BUTTON *pButton;
```

[Copy Code](#)

```
pScheme1 = GOLCreateScheme();  
pScheme1->pFont = &Gentium12;  
  
// assume button is created and initialized  
  
// reassign the scheme used by pButton to pScheme1  
GOLSetScheme(pButton, pScheme1);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) >  
[GOLSetScheme Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLGetScheme Macro

```
C
```

```
#define GOLGetScheme(pObj) ((OBJ_HEADER *)pObj)->pGo.
```

## Overview

This macro gets the GOL scheme used by the given object.

## Input Parameters

Input Parameters	Description
pObj	Pointer to the object of interest.

## Returns

Returns the style scheme used by the given object.

## Preconditions

none

## Side Effects

none

## Example

```
GOL_SCHEME *pScheme2;  
BUTTON *pButton;  
  
// assume button is created and initialized  
// get the scheme assigned to pButton
```

[Copy Code](#)

```
|| pScheme2 = GOLGetScheme(pButton);
```

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) >  
[GOLGetScheme Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLGetSchemeDefault Macro

```
C
```

```
#define GOLGetSchemeDefault _pDefaultGolScheme
```

## Overview

This macro returns the default GOL scheme pointer.

## Returns

Returns the pointer to the default style scheme.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [GOLGetSchemeDefault Macro](#)

## GOL\_SCHEME Structure

C

```
typedef struct {
    GFX_COLOR EmbossDkColor;
    GFX_COLOR EmbossLtColor;
    GFX_COLOR TextColor0;
    GFX_COLOR TextColor1;
    GFX_COLOR TextColorDisabled;
    GFX_COLOR Color0;
    GFX_COLOR Color1;
    GFX_COLOR ColorDisabled;
    GFX_COLOR CommonBkColor;
    void * pFont;
    WORD AlphaValue;
    GFX_GRADIENT_STYLE gradientScheme;
} GOL_SCHEME;
```

### Overview

GOL scheme defines the style scheme to be used by an object.

### Members

Members	Description
GFX_COLOR EmbossDkColor;	Emboss dark color used for 3d effect.
GFX_COLOR EmbossLtColor;	Emboss light color used for 3d effect.
GFX_COLOR TextColor0;	Character color 0 used for objects that supports text.
GFX_COLOR	Character color 1 used for objects that

TextColor1;	supports text.
GFX_COLOR TextColorDisabled;	Character color used when object is in a disabled state.
GFX_COLOR Color0;	Color 0 usually assigned to an Object state.
GFX_COLOR Color1;	Color 1 usually assigned to an Object state.
GFX_COLOR ColorDisabled;	Color used when an Object is in a disabled state.
GFX_COLOR CommonBkColor;	Background color used to hide Objects.
void * pFont;	Font selected for the scheme.
WORD AlphaValue;	Alpha value used for alpha blending, this is available only when <a href="#">USE_ALPHABLEND</a> is defined in the GraphicsConfig.h.
GFX_GRADIENT_STYLE gradientScheme;	Gradient Scheme for widgets, this is available only when <a href="#">USE_GRADIENT</a> is defined in the GraphicsConfig.h.

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [GOL\\_SCHEME Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Default Style Scheme Settings

## Variables

Lists the default settings for the style scheme.

## Variables

Name	Description
<a href="#">FONTDEFAULT</a>	Default GOL font.

## Links

[Style Scheme](#), [Variables](#)

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [Default Style Scheme Settings](#)

# FONTDEFAULT Variable

---

**C**

```
const FONT_FLASH FONTDEFAULT;
```

## Description

---

Default GOL font.

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [Default Style Scheme Settings](#) > [FONTDEFAULT Variable](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLFontDefault Variable

---

C

```
const FONT_FLASH GOLFontDefault = { (FLASH | COMP_NOI
```

## Description

---

This is variable GOLFontDefault.

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) >  
[GOLFontDefault Variable](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL\_EMBOSS\_SIZE Macro

```
C
```

```
#define GOL_EMBOSS_SIZE 3
```

## Overview

This option defines the 3-D effect emboss size for objects. The default value of this is 3 set in GOL.h. If it is not defined in GraphicsConfig.h file then the default value is used.

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) >  
[GOL\\_EMBOSS\\_SIZE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOLSchemeDefault Variable

```
C
```

```
const GOL_SCHEME GOLSchemeDefault = { BLACK, WHITE, \
```

## Overview

This defines a default GOL scheme that gets populated when an application calls the [GOLCreateScheme\(\)](#). The application can override this definition by defining the macro `GFX_SCHEMEDEFAULT` in the `GraphicsConfig.h` header file and defining `GOLSchemeDefault` structure in the application code. It is important to use the same structure name since the library assumes that this object exists and assigns the default style scheme pointer to this object.

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [GOLSchemeDefault Variable](#)

## RGBConvert Macro

```
C
```

```
#define RGBConvert(red, green, blue) (GFX_COLOR) (((
```

### Overview

This macro converts the color data to the selected [COLOR\\_DEPTH](#). This macro is only valid when [COLOR\\_DEPTH](#) is 8, 16, or 24.

<a href="#">COLOR_DEPTH</a>	Conversion
8	8-8-8 to 3-3-2 conversion
16	8-8-8 to 5-6-5 conversion
24	8-8-8 to 8-8-8 conversion or no conversion

### Input Parameters

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

### Returns

none

### Preconditions

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [RGBConvert Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Global Variables

## Variables

Graphics Object Layer global variables.

## Variables

Name	Description
<a href="#"><u>__pDefaultGolScheme</u></a>	Pointer to the GOL default scheme ( <a href="#"><u>GOL_SCHEME</u></a> ). This scheme is created in <a href="#"><u>GOLInit()</u></a> function. GOL scheme defines the style scheme to be used by an object. Use <a href="#"><u>GOLGetSchemeDefault()</u></a> to get this pointer.
<a href="#"><u>__pGolObjects</u></a>	Pointer to the current linked list of objects displayed and receiving messages. <a href="#"><u>GOLDraw()</u></a> and <a href="#"><u>GOLMsg()</u></a> process objects referenced by this pointer.
<a href="#"><u>__pObjectFocused</u></a>	Pointer to the object receiving keyboard input. This pointer is used or modified by the following APIs: <ul style="list-style-type: none"><li>• <a href="#"><u>GOLSetFocus()</u></a></li><li>• <a href="#"><u>GOLGetFocus()</u></a></li><li>• <a href="#"><u>GOLGetFocusNext()</u></a></li><li>• <a href="#"><u>GOLGetFocusPrev()</u></a></li><li>• <a href="#"><u>GOLCanBeFocused()</u></a></li></ul>

## Links

[Graphics Object Layer API, Variables](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Global Variables](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## pDefaultGolScheme Variable

C

```
GOL_SCHEME * _pDefaultGolScheme;
```

### Overview

Pointer to the GOL default scheme ([GOL\\_SCHEME](#)). This scheme is created in [GOLInit\(\)](#) function. GOL scheme defines the style scheme to be used by an object. Use [GOLGetSchemeDefault\(\)](#) to get this pointer.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Global Variables](#) > [\\_pDefaultGolScheme Variable](#)

# **\_pGoObjects Variable**

```
C
```

```
OBJ_HEADER * _pGo1objects;
```

## Overview

Pointer to the current linked list of objects displayed and receiving messages. [GOLDraw\(\)](#) and [GOLMsg\(\)](#) process objects referenced by this pointer.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Global Variables](#) > [\\_pGoObjects Variable](#)

## pObjectFocused Variable

C

```
OBJ_HEADER * _pObjectFocused;
```

### Overview

Pointer to the object receiving keyboard input. This pointer is used or modified by the following APIs:

- [GOLSetFocus\(\)](#)
- [GOLGetFocus\(\)](#)
- [GOLGetFocusNext\(\)](#)
- [GOLGetFocusPrev\(\)](#)
- [GOLCanBeFocused\(\)](#)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Global Variables](#) > [\\_pObjectFocused Variable](#)

# Graphics Primitive Layer API

[Enumerations](#) | [Structures](#) | [Topics](#)

## Enumerations

Name	Description
<a href="#">GFX_RESOURCE</a>	Memory type enumeration to determine the source of data. Used in interpreting bitmap and font from different memory sources.

## Structures

Name	Description
<a href="#">GFX_IMAGE_HEADER</a>	Structure for images stored in various system memory (Flash, External Memory (SPI, Parallel Flash, or memory in EPMP).
<a href="#">IMAGE_FLASH</a>	Structure for images stored in FLASH memory.
<a href="#">IMAGE_RAM</a>	Structure for images stored in RAM memory.
<a href="#">GFX_EXTDATA</a>	This structure is used to describe external memory.

## Topics

Name	Description
<a href="#">Text Functions</a>	This lists the Primitive level text functions.
<a href="#">Gradient</a>	Gradients can be drawn dynamically with

	the Microchip Graphics Library.
<a href="#">Line Functions</a>	This lists the Primitive line text functions.
<a href="#">Rectangle Functions</a>	This lists the Primitive level rectangle functions.
<a href="#">Circle Functions</a>	This lists the Primitive level circle functions.
<a href="#">Graphic Cursor</a>	This lists the functions to control the graphics cursor.
<a href="#">Alpha Blending Functions</a>	This lists the functions to control Alpha Blending. This feature is enabled only in selected drivers.
<a href="#">Bitmap Functions</a>	This lists the functions to display bitmaps.
<a href="#">External Memory</a>	This lists the external memory access functions and descriptions.
<a href="#">Set Up Functions</a>	This lists the Primitive set up and initialization functions.

## Links

[Enumerations](#), [Library API](#), [Structures](#), [Topics](#)

[Library API](#) > [Graphics Primitive Layer API](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Text Functions

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#) | [Types](#)

This lists the Primitive level text functions.

## Functions

	Name	Description
	<a href="#"><u>SetFont</u></a>	This function sets the current font used in <a href="#"><u>OutTextXY()</u></a> , <a href="#"><u>OutText()</u></a> and <a href="#"><u>OutChar()</u></a> functions.
	<a href="#"><u>OutChar</u></a>	This function outputs a character from the current graphic cursor position. <a href="#"><u>OutChar()</u></a> uses the current active font set with <a href="#"><u>SetFont()</u></a> .
	<a href="#"><u>OutText</u></a>	This function outputs a string of characters starting at the current graphic cursor position. The string must be terminated by a line feed or zero. For Non-Blocking configuration, <a href="#"><u>OutText()</u></a> may return control to the program due to display device busy status. When this happens zero is returned and <a href="#"><u>OutText()</u></a> must be called again to continue the outputting of the string. For Blocking configuration, this function always returns a 1. <a href="#"><u>OutText()</u></a> uses the current active font set with <a href="#"><u>SetFont()</u></a> .
	<a href="#"><u>OutTextXY</u></a>	This function outputs a string of characters starting at the given x, y position. The string must be terminated by a line feed or zero. For Non-Blocking configuration, <a href="#"><u>OutTextXY()</u></a> may return control to the

		program due to display device busy status. When this happens zero is returned and OutTextXY() must be called again to continue the outputting of the string. For Blocking configuration, this function always returns a 1. OutTextXY() uses the current active font set with <a href="#">SetFont()</a> .
◆	<a href="#">GetTextHeight</a>	This macro returns the height of the specified font. All characters in a given font table have a constant height.
◆	<a href="#">GetTextWidth</a>	This function returns the width of the specified string for the specified font. The string must be terminated by a line feed or zero.

## Macros

Name	Description
<a href="#">GetFontOrientation</a>	Returns font orientation.
<a href="#">SetFontOrientation</a>	Sets font orientation vertical or horizontal.
<a href="#">GFX_Font_GetAntiAliasType</a>	Returns the font anti-alias type.
<a href="#">GFX_Font_SetAntiAliasType</a>	Sets font anti-alias type to either Translucent or opaque.
<a href="#">XCHAR</a>	This macro sets the data type for the strings and characters. There are three types used for XCHAR and the type is selected by adding one of the macros in GraphicsConfig.h.

## Structures

---

Name	Description
<a href="#">FONT_HEADER</a>	Structure describing the font header.
<a href="#">FONT_FLASH</a>	Structure for font stored in FLASH memory.

## Topics

---

Name	Description
<a href="#">Anti-Alias Type</a>	Anti-alias type definitions.

## Types

---

Name	Description
<a href="#">FONT_EXTERNAL</a>	Structure for font stored in EXTERNAL memory space. (example: External SPI or parallel Flash, EDS_EPMP)

## Links

---

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#), [Types](#)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## FONT\_HEADER Structure

```
C
typedef struct {
    BYTE fontID;
    BYTE extendedGlyphEntry : 1;
    BYTE res1 : 1;
    BYTE bpp : 2;
    BYTE orientation : 2;
    BYTE res2 : 2;
    WORD firstChar;
    WORD lastChar;
    WORD height;
} FONT_HEADER;
```

### Overview

Structure describing the font header.

### Members

Members	Description
BYTE fontID;	User assigned value
BYTE extendedGlyphEntry : 1;	Extended Glyph entry flag. When set font has extended glyph feature enabled.
BYTE res1 : 1;	Reserved for future use (must be set to 0)
BYTE bpp : 2;	Actual BPP = $2^{\text{bpp}}$ <ul style="list-style-type: none"><li>• 0 - 1 BPP</li><li>• 1 - 2 BPP</li><li>• 2 - 4 BPP</li><li>• 3 - 8 BPP</li></ul>

BYTE orientation : 2;	Orientation of the character glyphs (0,90,180,270 degrees) <ul style="list-style-type: none"> <li>• 00 - Normal</li> <li>• 01 - Characters rotated 270 degrees clockwise</li> <li>• 10 - Characters rotated 180 degrees</li> <li>• 11 - Characters rotated 90 degrees clockwise</li> </ul>
BYTE res2 : 2;	Reserved for future use (must be set to 0).
WORD firstChar;	Character code of first character (e.g. 32).
WORD lastChar;	Character code of last character in font (e.g. 3006).
WORD height;	Font characters height in pixels.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [FONT\\_HEADER Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## FONT\_FLASH Structure

C

```
typedef struct {  
    GFX_RESOURCE type;  
    GFX_FONT_SPACE char * address;  
} FONT_FLASH;
```

### Overview

Structure for font stored in FLASH memory.

### Members

Members	Description
GFX_RESOURCE type;	must be FLASH
GFX_FONT_SPACE char * address;	font image address in FLASH

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) >  
[FONT\\_FLASH Structure](#)

# FONT\_EXTERNAL Type

```
C
```

```
typedef GFX_EXTDATA FONT_EXTERNAL;
```

## Overview

Structure for font stored in EXTERNAL memory space. (example: External SPI or parallel Flash, EDS\_EPMP)

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [FONT\\_EXTERNAL Type](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetFont Function

```
C  
void SetFont(  
    void * pFont  
);
```

## Overview

This function sets the current font used in [OutTextXY\(\)](#), [OutText\(\)](#) and [OutChar\(\)](#) functions.

## Input Parameters

Input Parameters	Description
void * pFont	Pointer to the new font image to be used.

## Returns

none

## Side Effects

none

## Example

See [OutTextXY\(\)](#) example.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [SetFont Function](#)

# GetFontOrientation Macro

C

```
#define GetFontOrientation _fontOrientation
```

## Overview

Returns font orientation.

## Returns

Return the current font orientation.

- 1 when font orientation is vertical
- 0 when font orientation is horizontal

## Preconditions

none

## Example

[Copy Code](#)

```
void PlaceText(SHORT x, SHORT y, WORD space, XCHAR
{
    SHORT width;

    SetColor(BRIGHTRED);           // set colo
    SetFont(pMyFont);             // set to u

    // get string width
    width = GetTextWidth(pString, pMyFont);

    // check if it fits
    if (space < width)
```

```
{
    if (GetFontOrientation() == 0)
        // reset the orientation to vertical
        SetFontOrientation(1);
}
else
{
    if (GetFontOrientation() == 1)
        // reset the orientation to horizontal
        SetFontOrientation(0);
}
// place string in the middle of the screen
OutTextXY(x, y, pString);
}
```

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) >  
[GetFontOrientation Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetFontOrientation Macro

```
C
```

```
#define SetFontOrientation(orient) _fontOrientation :
```

## Overview

Sets font orientation vertical or horizontal.

## Input Parameters

Input Parameters	Description
orient	sets font orientation when rendering characters and strings. <ul style="list-style-type: none"><li>• 1 when font orientation is vertical</li><li>• 0 when font orientation is horizontal</li></ul>

## Returns

none

## Preconditions

none

## Example

See [SetFontOrientation\(\)](#) example.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [SetFontOrientation Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# GFX\_Font\_GetAntiAliasType Macro

```
C
```

```
#define GFX_Font_GetAntiAliasType
```

## Overview

Returns the font anti-alias type.

## Returns

Return the current font anti-alias type.

- [ANTIALIAS\\_TRANSLUCENT](#) - (or 1) when font anti-alias is type translucent
- [ANTIALIAS\\_OPAQUE](#) - (or 0) when font anti-alias is type opaque

## Preconditions

Compiler switch [USE\\_ANTIALIASED\\_FONTS](#) must be enabled

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) >  
[GFX\\_Font\\_GetAntiAliasType Macro](#)

# GFX\_Font\_SetAntiAliasType Macro

```
C
```

```
#define GFX_Font_SetAntiAliasType(transparency)
```

## Overview

Sets font anti-alias type to either Translucent or opaque.

## Input Parameters

Input Parameters	Description
transparency	sets font font anti-alias type <ul style="list-style-type: none"><li>• <a href="#">ANTIALIAS_TRANSLUCENT</a> - (or 1) when font anti-alias type is translucent</li><li>• <a href="#">ANTIALIAS_OPAQUE</a> - (or 0) when font anti-alias type is opaque</li></ul>

## Returns

none

## Preconditions

Compiler switch [USE\\_ANTIALIASED\\_FONTS](#) must be enabled

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [GFX\\_Font\\_SetAntiAliasType Macro](#)

# OutChar Function

```
C  
WORD OutChar(  
    XCHAR ch  
);
```

## Overview

This function outputs a character from the current graphic cursor position. OutChar() uses the current active font set with [SetFont\(\)](#).

## Input Parameters

Input Parameters	Description
XCHAR ch	The character code to be displayed.

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the character is not yet completely drawn.
- Returns 1 when the character is completely drawn.

For Blocking configuration:

- Always return 1.

## Preconditions

none

## Side Effects

After the function is completed, the graphic cursor position is moved in the horizontal direction by the character width. Vertical position of the graphic cursor is not changed.

## Example

[Copy Code](#)

```
static WORD counter = 0;
XCHAR  ch;

// render characters until null character
while((XCHAR)(ch = *(textString + counter)) != 0)
{
    if(OutChar(ch) == 0)
        return (0);
    counter++;
}
```

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [OutChar Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# OutText Function

```
C
WORD OutText(
    XCHAR * textString
);
```

## Overview

This function outputs a string of characters starting at the current graphic cursor position. The string must be terminated by a line feed or zero. For Non-Blocking configuration, OutText() may return control to the program due to display device busy status. When this happens zero is returned and OutText() must be called again to continue the outputting of the string. For Blocking configuration, this function always returns a 1. OutText() uses the current active font set with [SetFont\(\)](#).

## Input Parameters

Input Parameters	Description
XCHAR * textString	Pointer to the string to be displayed.

## Returns

For NON-Blocking configuration:

- Returns 0 when string is not yet outputted completely.
- Returns 1 when string is outputted completely.

For Blocking configuration:

- Always return 1.

## Side Effects

Current horizontal graphic cursor position will be moved to the end of the text. The vertical graphic cursor position will not be changed.

## Example

---

[Copy Code](#)

```
SetFont(pMyFont);  
SetColor(WHITE);  
// place the string at the upper left corner of  
MoveTo(0, 0);  
OutText("Test String!");
```

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [OutText Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# OutTextXY Function

```
C
WORD OutTextXY(
    SHORT x,
    SHORT y,
    XCHAR * textString
);
```

## Overview

This function outputs a string of characters starting at the given x, y position. The string must be terminated by a line feed or zero. For Non-Blocking configuration, OutTextXY() may return control to the program due to display device busy status. When this happens zero is returned and OutTextXY() must be called again to continue the outputting of the string. For Blocking configuration, this function always returns a 1. OutTextXY() uses the current active font set with [SetFont\(\)](#).

## Input Parameters

Input Parameters	Description
SHORT x	Defines the x starting position of the string.
SHORT y	Defines the y starting position of the string.
XCHAR * textString	Pointer to the string to be displayed.

## Returns

For NON-Blocking configuration:

- Returns 0 when string is not yet outputted completely.

- Returns 1 when string is outputted completely.
- For Blocking configuration:
- Always return 1.

## Side Effects

---

Current horizontal graphic cursor position will be moved to the end of the text. The vertical graphic cursor position will not be changed.

## Example

---

[Copy Code](#)

```
void PlaceText(void)
{
    SHORT width, height;
    static const XCHAR text[] = "Touch screen to co

    SetColor(BRIGHTRED);           // set colo
    SetFont(pMyFont);              // set font

    // get string width & height
    width = GetTextWidth(text, pMyFont);
    height = GetTextHeight(pMyFont);

    // place string in the middle of the screen
    OutTextXY( (GetMaxX() - width) >> 1,
               (GetMaxY() - height) >> 1,
               (char*)text);
}
```

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [OutTextXY Function](#)

# GetTextHeight Function

```
C  
SHORT GetTextHeight(  
    void * pFont  
);
```

## Overview

This macro returns the height of the specified font. All characters in a given font table have a constant height.

## Input Parameters

Input Parameters	Description
void * pFont	Pointer to the font image.

## Returns

Returns the font height.

## Side Effects

none

## Example

See [OutTextXY\(\)](#) example.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [GetTextHeight Function](#)

# GetTextWidth Function

```
C
SHORT GetTextWidth(
    XCHAR * textString,
    void * pFont
);
```

## Overview

This function returns the width of the specified string for the specified font. The string must be terminated by a line feed or zero.

## Input Parameters

Input Parameters	Description
XCHAR * textString	Pointer to the string.
void * pFont	Pointer to the font image.

## Returns

Returns the string width in the specified font.

## Side Effects

none

## Example

See [OutTextXY\(\)](#) example.

## GetTextWidth Function

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# XCHAR Macro

**C****#define XCHAR char**

## Overview

This macro sets the data type for the strings and characters. There are three types used for XCHAR and the type is selected by adding one of the macros in GraphicsConfig.h.

In GraphicsConfig.h	XCHAR	Description
<code>#define <a href="#">USE_MULTIBYTECHAR</a></code>	<code>#define XCHAR unsigned short</code>	Use multibyte characters (0-2 <sup>16</sup> range).
<code>#define <a href="#">USE_UNSIGNED_XCHAR</a></code>	<code>#define XCHAR unsigned char</code>	Use unsigned char (0-255 range).
none of the two defined	<code>#define XCHAR char</code>	Use signed char (0-127 range).

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [XCHAR Macro](#)

# Anti-Alias Type

## Macros

Anti-alias type definitions.

## Macros

Name	Description
<a href="#"><u>ANTIALIAS_OPAQUE</u></a>	Mid colors are calculated only once while rendering each character. This is ideal for rendering text over a constant background.
<a href="#"><u>ANTIALIAS_TRANSLUCENT</u></a>	Mid values 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.

## Links

[Text Functions](#), [Macros](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [Anti-Alias Type](#)

# ANTIALIAS\_OPAQUE Macro

```
C
```

```
#define ANTIALIAS_OPAQUE 0
```

## Description

Mid colors are calculated only once while rendering each character. This is ideal for rendering text over a constant background.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [Anti-Alias Type](#) > [ANTIALIAS\\_OPAQUE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ANTIALIAS\_TRANSLUCENT Macro

```
C
```

```
#define ANTIALIAS_TRANSLUCENT 1
```

## Description

Mid values 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.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [Anti-Alias Type](#) > [ANTIALIAS\\_TRANSLUCENT Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Gradient

[Enumerations](#) | [Functions](#) | [Structures](#)

## Enumerations

Name	Description
<a href="#">GFX_GRADIENT_TYPE</a>	Enumeration for gradient type

## Functions

	Name	Description
	<a href="#">BarGradient</a>	This renders a bar onto the screen, but instead of one color, a gradient is drawn depending on the direction ( <a href="#">GFX_GRADIENT_TYPE</a> ), length, and colors chosen. This function is a blocking call.
	<a href="#">BevelGradient</a>	This renders a filled bevel with gradient color on the fill. It works the same as the fillbevel function, except a gradient out of color1 and color2 is drawn depending on the direction ( <a href="#">GFX_GRADIENT_TYPE</a> ). This function is a blocking call.

## Structures

Name	Description
<a href="#">GFX_GRADIENT_STYLE</a>	This structure is used to describe the gradient style.

## Links

---

[Enumerations](#), [Functions](#), [Graphics Primitive Layer API](#), [Legend](#),  
[Structures](#)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BarGradient Function

---

**C**

```
WORD BarGradient(  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    GFX_COLOR color1,  
    GFX_COLOR color2,  
    DWORD length,  
    BYTE direction  
);
```

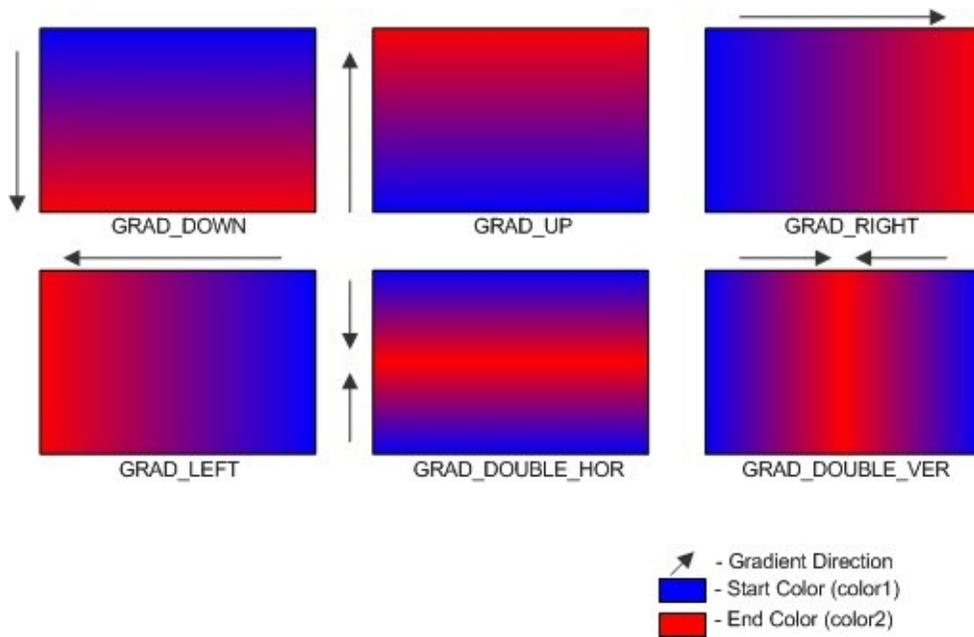
## Overview

---

This renders a bar onto the screen, but instead of one color, a gradient is drawn depending on the direction ([GFX\\_GRADIENT\\_TYPE](#)), length, and colors chosen. This function is a blocking call.

## Description

---



## Input Parameters

Input Parameters	Description
SHORT left	x position of the left top corner.
SHORT top	y position of the left top corner.
SHORT right	x position of the right bottom corner.
SHORT bottom	y position of the right bottom corner.
GFX_COLOR color1	start color for the gradient
GFX_COLOR color2	end color for the gradient
DWORD length	From 0-100%. How much of a gradient is wanted
BYTE direction	Gradient Direction

## Returns

Always returns a 1 since it is a blocking function.

## Preconditions

---

[USE\\_GRADIENT](#) macro must be defined (in GraphicsConfig.h)

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
// draw a full screen gradient background  
// with color transitioning from BRIGHTRED to  
// BLACK in the upward direction.  
  
GFX_GRADIENT_STYLE  gradScheme;  
  
gradScheme.gradientType      = GRAD_UP;  
gradScheme.gradientStartColor = BRIGHTRED;  
gradScheme.gradientEndColor  = BLACK;  
  
  BarGradient(0,  
              0,  
              GetMaxX(),  
              GetMaxY(),  
              gradScheme.gradientStartColor,  
              gradScheme.gradientEndColor,  
              50,  
  
              gradScheme.gradientType);
```

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#) > [BarGradient Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

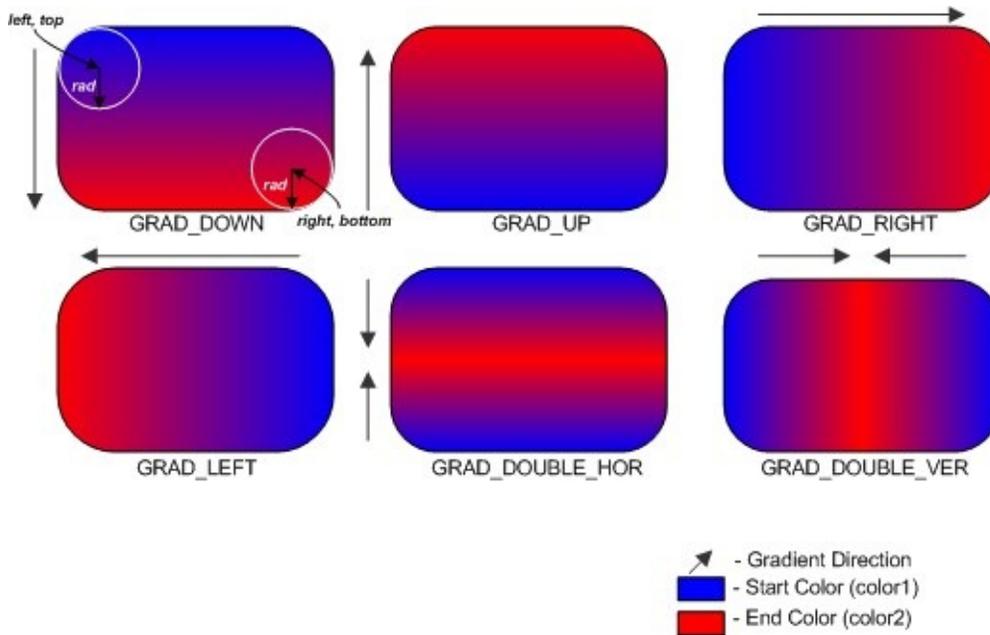
## BevelGradient Function

```
C
WORD BevelGradient(
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    SHORT rad,
    GFX_COLOR color1,
    GFX_COLOR color2,
    DWORD length,
    BYTE direction
);
```

### Overview

This renders a filled bevel with gradient color on the fill. It works the same as the fillbevel function, except a gradient out of color1 and color2 is drawn depending on the direction ([GFX\\_GRADIENT\\_TYPE](#)). This function is a blocking call.

### Description



## Input Parameters

Input Parameters	Description
SHORT left	x coordinate position of the upper left center of the circle that draws the rounded corners.
SHORT top	y coordinate position of the upper left center of the circle that draws the rounded corners.
SHORT right	x coordinate position of the lower right center of the circle that draws the rounded corners.
SHORT bottom	y coordinate position of the lower right center of the circle that draws the rounded corners.
SHORT rad	defines the radius of the circle, that draws the rounded corners. When rad = 0, the object drawn is a rectangular gradient.

GFX_COLOR color1	start color for the gradient
GFX_COLOR color2	end color for the gradient
DWORD length	From 0-100%. How much of a gradient is wanted
BYTE direction	see <a href="#">GFX_GRADIENT_TYPE</a>

---

## Returns

Always returns a 1 since it is a blocking function.

---

## Preconditions

[USE\\_GRADIENT](#) macro must be defined (in GraphicsConfig.h)

---

## Side Effects

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#) > [BevelGradient Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_GRADIENT\_TYPE Enumeration

C

```
typedef enum {  
    GRAD_NONE = 0,  
    GRAD_DOWN,  
    GRAD_RIGHT,  
    GRAD_UP,  
    GRAD_LEFT,  
    GRAD_DOUBLE_VER,  
    GRAD_DOUBLE_HOR  
} GFX_GRADIENT_TYPE;
```

### Overview

Enumeration for gradient type

### Members

Members	Description
GRAD_NONE = 0	No <a href="#">Gradients</a> to be drawn
GRAD_DOWN	gradient changes in the vertical direction
GRAD_RIGHT	gradient change in the horizontal direction
GRAD_UP	gradient changes in the vertical direction
GRAD_LEFT	gradient change in the horizontal direction
GRAD_DOUBLE_VER	two gradient transitions in the vertical direction
GRAD_DOUBLE_HOR	two gradient transitions in the horizontal direction

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#) >  
[GFX\\_GRADIENT\\_TYPE Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_GRADIENT\_STYLE Structure

C

```
typedef struct {
    GFX_GRADIENT_TYPE gradientType;
    DWORD gradientStartColor;
    DWORD gradientEndColor;
    DWORD gradientLength;
} GFX_GRADIENT_STYLE;
```

### Overview

This structure is used to describe the gradient style.

### Members

Members	Description
GFX_GRADIENT_TYPE gradientType;	selected the gradient type
DWORD gradientStartColor;	sets the starting color of gradient transition
DWORD gradientEndColor;	sets the ending color of gradient transition
DWORD gradientLength;	defines the length of the gradient transition in pixels

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#) >  
[GFX\\_GRADIENT\\_STYLE Structure](#)

# Line Functions

[Functions](#) | [Macros](#) | [Topics](#)

This lists the Primitive line text functions.

## Functions

	Name	Description
	<a href="#">Line</a>	This function draws a line with the current line type from the start point to the end point.

## Macros

Name	Description
<a href="#">LineRel</a>	This macro draws a line with the current line type from the current graphic cursor position to the position defined by displacement.
<a href="#">LineTo</a>	This macro draws a line with the current line type from the current graphic cursor position to the given x, y position.
<a href="#">SetLineThickness</a>	This macro sets sets line thickness to 1 pixel or 3 pixels.
<a href="#">SetLineType</a>	This macro sets the line type to draw.

## Topics

Name	Description
<a href="#">Line Types</a>	Line type definitions.

[Line Size](#)

Line sizes definition.

## Links

---

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#), [Topics](#)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Line Function

```
C  
WORD Line(  
    SHORT x1,  
    SHORT y1,  
    SHORT x2,  
    SHORT y2  
);
```

## Overview

This function draws a line with the current line type from the start point to the end point.

## Input Parameters

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

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

The graphic cursor position is moved to the end point of the line.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LineRel Macro

```
C
```

```
#define LineRel(dX, dY) Line(GetX(), GetY(), GetX() + dX, GetY() + dY)
```

## Overview

This macro draws a line with the current line type from the current graphic cursor position to the position defined by displacement.

## Input Parameters

Input Parameters	Description
dX	Displacement from the current x position.
dY	Displacement from the current y position.

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

The graphic cursor position is moved to the end point of the line.

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [LineRel Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## LineTo Macro

```
C
```

```
#define LineTo(x, y) Line(_cursorX, _cursorY, x, y)
```

### Overview

This macro draws a line with the current line type from the current graphic cursor position to the given x, y position.

### Input Parameters

Input Parameters	Description
x	End point x position.
y	End point y position.

### Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

### Side Effects

The graphic cursor position is moved to the end point of the line.

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [LineTo Macro](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetLineThickness Macro

```
C
```

```
#define SetLineThickness(InThickness) _lineThickness
```

## Overview

This macro sets sets line thickness to 1 pixel or 3 pixels.

## Input Parameters

Input Parameters	Description
InThickness	<u>Line</u> thickness code <ul style="list-style-type: none"><li><a href="#">NORMAL_LINE</a> : 1 pixel</li><li><a href="#">THICK_LINE</a> : 3 pixels</li></ul>

## Returns

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [SetLineThickness Macro](#)

# SetLineType Macro

```
C
```

```
#define SetLineType(InType) _lineType = InType;
```

## Overview

This macro sets the line type to draw.

## Input Parameters

Input Parameters	Description
InType	The type of line to be used. Supported line types: <ul style="list-style-type: none"><li>• <a href="#">SOLID_LINE</a></li><li>• <a href="#">DOTTED_LINE</a></li><li>• <a href="#">DASHED_LINE</a></li></ul>

## Returns

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [SetLineType Macro](#)

# Line Types

## Macros

Line type definitions.

## Macros

Name	Description
<a href="#">SOLID_LINE</a>	Solid <a href="#">Line</a> Style
<a href="#">DASHED_LINE</a>	Dashed <a href="#">Line</a> Style
<a href="#">DOTTED_LINE</a>	Dotted <a href="#">Line</a> Style

## Links

[Line Functions](#), [Macros](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Types](#)

# SOLID\_LINE Macro

---

```
C
```

```
#define SOLID_LINE 0
```

## Description

---

Solid [Line](#) Style

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Types](#)  
> [SOLID\\_LINE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DASHED\_LINE Macro

```
C
```

```
#define DASHED_LINE 4
```

## Description

Dashed [Line](#) Style

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Types](#)  
> [DASHED\\_LINE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DOTTED\_LINE Macro

---

```
C
```

```
#define DOTTED_LINE 1
```

## Description

---

Dotted [Line](#) Style

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Types](#)  
> [DOTTED\\_LINE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Line Size

## Macros

Line sizes definition.

## Macros

Name	Description
<a href="#"><u>NORMAL_LINE</u></a>	Normal <a href="#"><u>Line</u></a> (thickness is 1 pixel)
<a href="#"><u>THICK_LINE</u></a>	Thick <a href="#"><u>Line</u></a> (thickness is 3 pixels)

## Links

[Line Functions, Macros](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Size](#)

# NORMAL\_LINE Macro

```
C
```

```
#define NORMAL_LINE 0
```

## Description

Normal Line (thickness is 1 pixel)

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Size](#)  
> [NORMAL\\_LINE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# THICK\_LINE Macro

---

```
C
```

```
#define THICK_LINE 1
```

## Description

---

Thick [Line](#) (thickness is 3 pixels)

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Size](#)  
> [THICK\\_LINE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Rectangle Functions

[Functions](#) | [Macros](#)

This lists the Primitive level rectangle functions.

## Functions

	Name	Description
	<a href="#">Bar</a>	This function draws a bar given the left, top and right, bottom corners with the current set color ( <a href="#">SetColor()</a> ). When alpha blending is enabled the bar is alpha blended with the existing pixels specified by the parameters. The alpha percentage used is the last value set by <a href="#">SetAlpha()</a> .
	<a href="#">DrawPoly</a>	This function draws a polygon with the current line type using the given number of points. The polygon points (polyPoints) are stored in an array arranged in the following order:

## Macros

Name	Description
<a href="#">Rectangle</a>	This macro draws a rectangle with the given left, top and right, bottom corners. Current line type is used.

## Links

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#)

---

## [Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bar Function

```
C
WORD Bar(
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom
);
```

## Overview

This function draws a bar given the left, top and right, bottom corners with the current set color ([SetColor\(\)](#)). When alpha blending is enabled the bar is alpha blended with the existing pixels specified by the parameters. The alpha percentage used is the last value set by [SetAlpha\(\)](#).

## Input Parameters

Input Parameters	Description
SHORT left	x position of the left top corner.
SHORT top	y position of the left top corner.
SHORT right	x position of the right bottom corner.
SHORT bottom	y position of the right bottom corner.

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely

drawn.

- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#) > [Bar Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Rectangle Macro

```
C
```

```
#define Rectangle(left, top, right, bottom) Bevel(le
```

### Overview

This macro draws a rectangle with the given left, top and right, bottom corners. Current line type is used.

### Input Parameters

Input Parameters	Description
left	x position of the left top corner.
top	y position of the left top corner.
right	x position of the right bottom corner.
bottom	y position of the right bottom corner.

### Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

### Side Effects

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#) >  
[Rectangle Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DrawPoly Function

**C**

```
WORD DrawPoly(  
    SHORT numPoints,  
    SHORT * polyPoints  
);
```

## Overview

This function draws a polygon with the current line type using the given number of points. The polygon points (polyPoints) are stored in an array arranged in the following order:

```
SHORT polyPoints[size] = {x0, y0, x1, y1, x2, y2 ... xn, yn};  
Where n = # of polygon sides  
size = numPoints * 2
```

DrawPoly() draws any shape defined by the polyPoints. The function will just draw the lines connecting all the x,y points enumerated by polyPoints[].

## Input Parameters

Input Parameters	Description
SHORT numPoints	Defines the number of x,y points in the polygon.
SHORT * polyPoints	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.

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

none

## Example

---

[Copy Code](#)

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

SetColor(WHITE);           // set
SetLineType(SOLID_LINE);   // set
SetLineThickness(THICK_LINE); // set
DrawPoly(6, OpenShapeXYPoints); // dra
DrawPoly(8, ClosedShapeXYPoints); // dra
```

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#) > [DrawPoly Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Circle Functions

## Functions | Macros

This lists the Primitive level circle functions.

### Functions

	Name	Description
	<a href="#"><u>Arc</u></a>	<p>Draws the octant arc of the beveled figure with the given centers, radii and octant mask. When octant = 0xFF and the following are true:</p> <ol style="list-style-type: none"> <li>1. <math>x_L = x_R</math>, <math>y_T = y_B</math>, <math>r_1 = 0</math> and <math>r_2 = z</math>, a filled circle is drawn with a radius of <math>z</math>.</li> <li>2. radii have values (where <math>r_1 &lt; r_2</math>), a full ring with thickness of <math>(r_2 - r_1)</math> is drawn.</li> <li>3. <math>x_L \neq x_R</math>, <math>y_T \neq y_B</math>, <math>r_1 = 0</math> and <math>r_2 = 0</math> (where <math>x_R &gt; x_L</math> and <math>y_B &gt; y_T</math>) a rectangle is drawn. <math>x_L</math>, <math>y_T</math> specifies the left top corner and <math>x_R, \dots</math> <a href="#">more</a></li> </ol>
	<a href="#"><u>DrawArc</u></a>	<p>This renders an arc with from startAngle to endAngle with the thickness of <math>r_2 - r_1</math>. The function returns 1 when the arc is rendered successfully and returns a 0 when it is not yet finished. The next call to the function will continue the rendering.</p>
	<a href="#"><u>Bevel</u></a>	<p>Draws a beveled figure on the screen. When <math>x_1 = x_2</math> and <math>y_1 = y_2</math>, a circular object is drawn. When <math>x_1 &lt; x_2</math> and <math>y_1 &lt; y_2</math> and rad (radius) = 0, a rectangular object is drawn.</p>
	<a href="#"><u>FillBevel</u></a>	<p>Draws a filled beveled figure on the screen. For a filled circular object <math>x_1 = x_2</math> and <math>y_1 =</math></p>

y2. For a filled rectangular object radius = 0.

## Macros

Name	Description
<a href="#">Circle</a>	This macro draws a circle with the given center and radius.
<a href="#">FillCircle</a>	This macro draws a filled circle. Uses the <a href="#">FillBevel()</a> function.
<a href="#">SetBevelDrawType</a>	This macro sets the fill bevel type to be drawn.

## Links

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Circle Macro

```
C
#define Circle(x, y, radius) Bevel(x, y, x, y, radius)
```

### Overview

This macro draws a circle with the given center and radius.

### Input Parameters

Input Parameters	Description
x	Center x position.
y	Center y position.
radius	the radius of the circle.

### Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

### Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#) > [Circle Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## FillCircle Macro

```
C
```

```
#define FillCircle(x1, y1, rad) FillBevel(x1, y1, x1,
```

### Overview

This macro draws a filled circle. Uses the [FillBevel\(\)](#) function.

### Input Parameters

Input Parameters	Description
x1	x coordinate position of the center of the circle.
y1	y coordinate position of the center of the circle.
rad	defines the radius of the circle.

### Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

### Side Effects

none

# Macro

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Arc Function

**C**

```
WORD Arc(  
    SHORT xL,  
    SHORT yT,  
    SHORT xR,  
    SHORT yB,  
    SHORT r1,  
    SHORT r2,  
    BYTE octant  
);
```

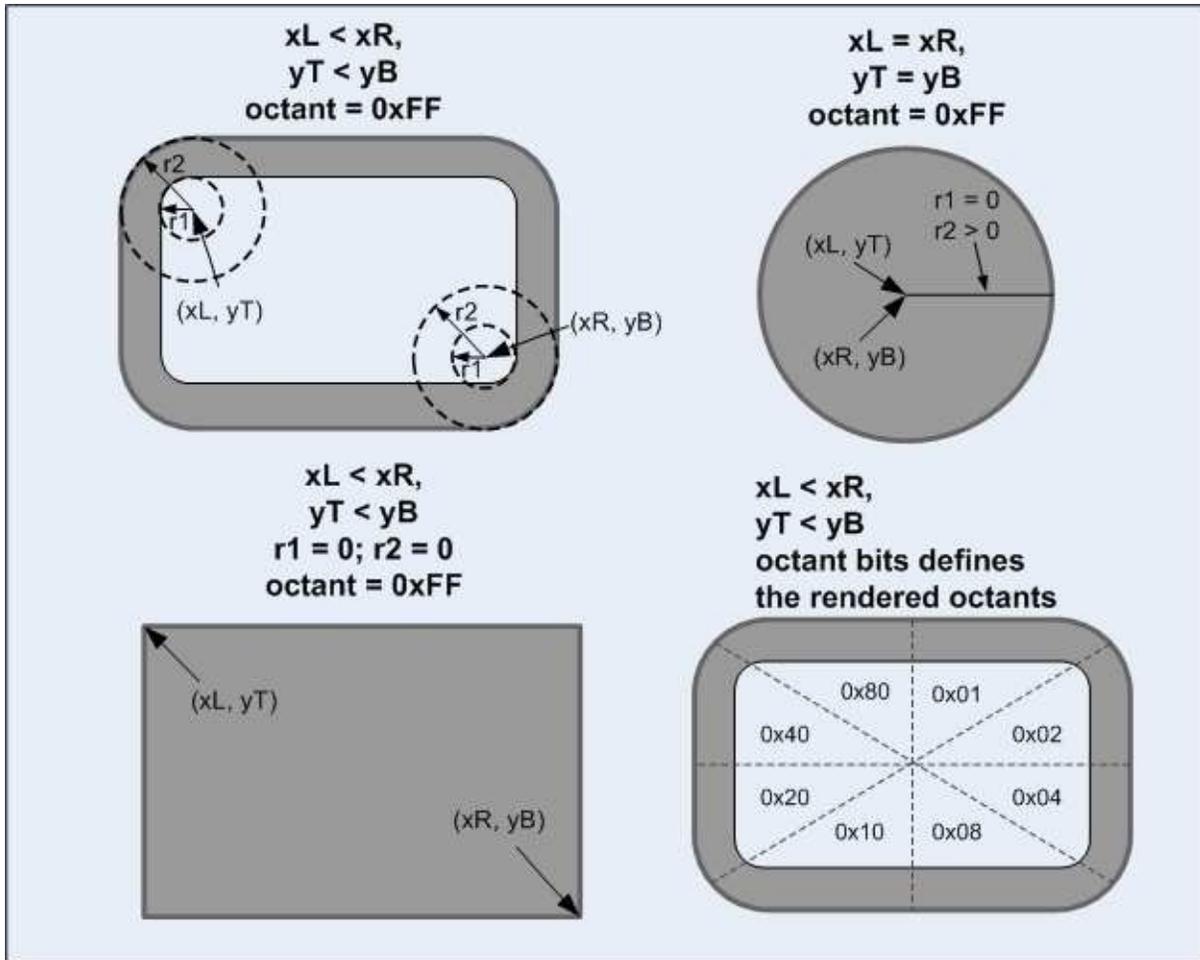
## Overview

Draws the octant arc of the beveled figure with the given centers, radii and octant mask. When octant = 0xFF and the following are true:

1.  $xL = xR$ ,  $yT = yB$ ,  $r1 = 0$  and  $r2 = z$ , a filled circle is drawn with a radius of  $z$ .
2. radii have values (where  $r1 < r2$ ), a full ring with thickness of  $(r2-r1)$  is drawn.
3.  $xL \neq xR$ ,  $yT \neq yB$ ,  $r1 = 0$  and  $r2 = 0$  (where  $xR > xL$  and  $yB > yT$ ) a rectangle is drawn.  $xL$ ,  $yT$  specifies the left top corner and  $xR$ ,  $yB$  specifies the right bottom corner.

When octant  $\neq 0xFF$  the figure drawn is the subsection of the 8 section figure where each non-zero bit of the octant value specifies the octants that will be drawn.

## Description



## Input Parameters

Input Parameters	Description
SHORT xL	x location of the upper left center in the x,y coordinate.
SHORT yT	y location of the upper left center in the x,y coordinate.
SHORT xR	x location of the lower right center in the x,y coordinate.
SHORT yB	y location of the lower right center in the x,y coordinate.

SHORT r1	The smaller radius of the two concentric circles that defines the thickness of the object.
SHORT r2	The larger of radius the two concentric circles that defines the thickness of the object.
BYTE octant	Bitmask of the octant that will be drawn. Moving in a clockwise direction from x = 0, y = +radius <ul style="list-style-type: none"><li>• bit0 : first octant</li><li>• bit1 : second octant</li><li>• bit2 : third octant</li><li>• bit3 : fourth octant</li><li>• bit4 : fifth octant</li><li>• bit5 : sixth octant</li><li>• bit6 : seventh octant</li><li>• bit7 : eight octant</li></ul>

## Returns

Returns the rendering status. 1 - If the rendering was completed and 0 - If the rendering is not yet finished.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#) > [Arc Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## DrawArc Function

```
C  
WORD DrawArc(  
    SHORT cx,  
    SHORT cy,  
    SHORT r1,  
    SHORT r2,  
    SHORT startAngle,  
    SHORT endAngle  
);
```

### Overview

This renders an arc with from startAngle to endAngle with the thickness of r2-r1. The function returns 1 when the arc is rendered successfully and returns a 0 when it is not yet finished. The next call to the function will continue the rendering.

### Input Parameters

Input Parameters	Description
SHORT cx	the location of the center of the arc in the x direction.
SHORT cy	the location of the center of the arc in the y direction.
SHORT r1	the smaller radius of the arc.
SHORT r2	the larger radius of the arc.
SHORT startAngle	start angle of the arc.
SHORT endAngle	end angle of the arc.

---

## Returns

---

Returns 1 if the rendering is done, 0 if not yet done.

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#) > [DrawArc Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

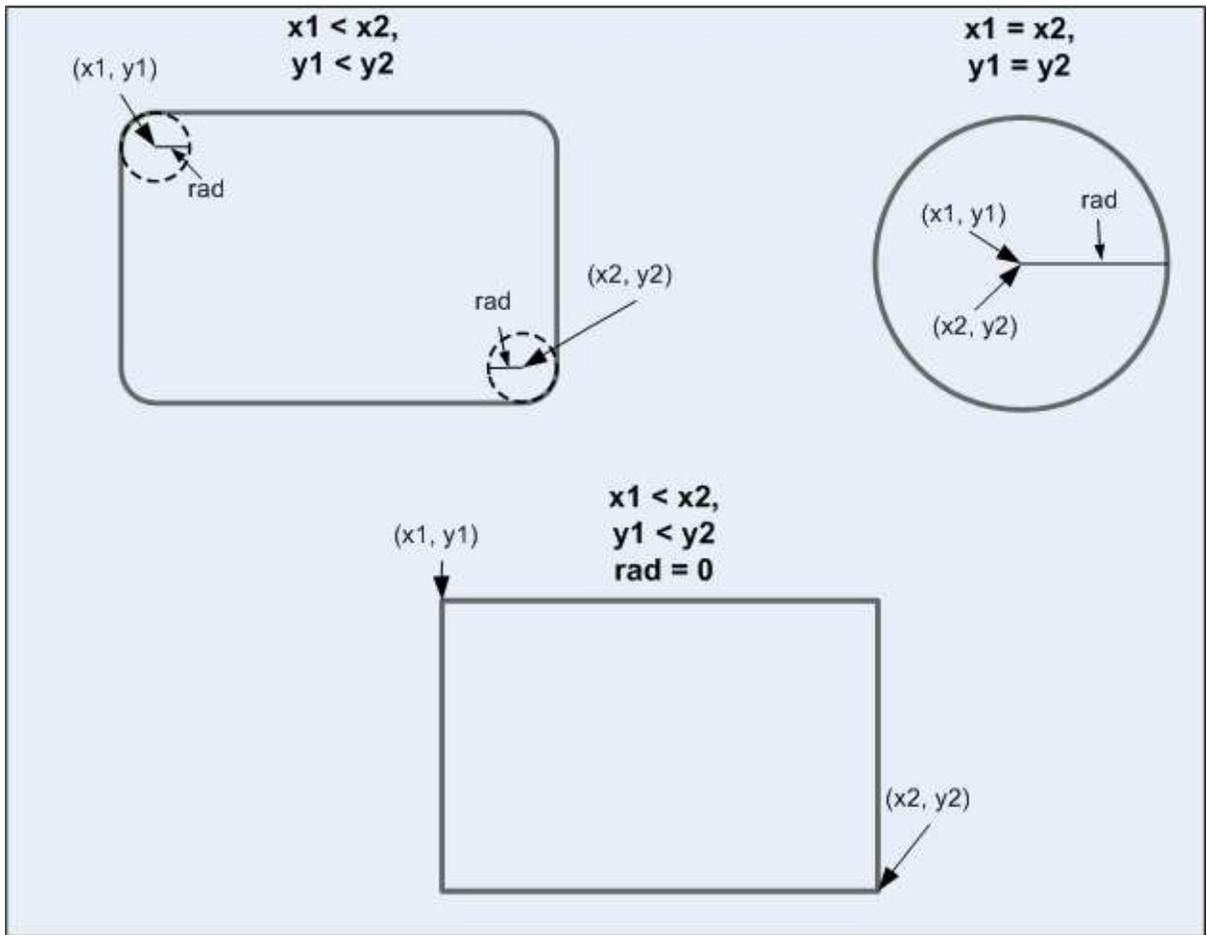
# Bevel Function

```
C
WORD Bevel(
    SHORT x1,
    SHORT y1,
    SHORT x2,
    SHORT y2,
    SHORT rad
);
```

## Overview

Draws a beveled figure on the screen. When  $x1 = x2$  and  $y1 = y2$ , a circular object is drawn. When  $x1 < x2$  and  $y1 < y2$  and  $rad$  (radius) = 0, a rectangular object is drawn.

## Description



## Input Parameters

Input Parameters	Description
SHORT x1	x coordinate position of the upper left center of the circle that draws the rounded corners.
SHORT y1	y coordinate position of the upper left center of the circle that draws the rounded corners.
SHORT x2	x coordinate position of the lower right center of the circle that draws the rounded corners.

SHORT y2	y coordinate position of the lower right center of the circle that draws the rounded corners.
SHORT rad	defines the radius of the circle, that draws the rounded corners.

## Returns

---

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#) > [Bevel Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## FillBevel Function

---

**C**

```
WORD FillBevel(  
    SHORT x1,  
    SHORT y1,  
    SHORT x2,  
    SHORT y2,  
    SHORT rad  
);
```

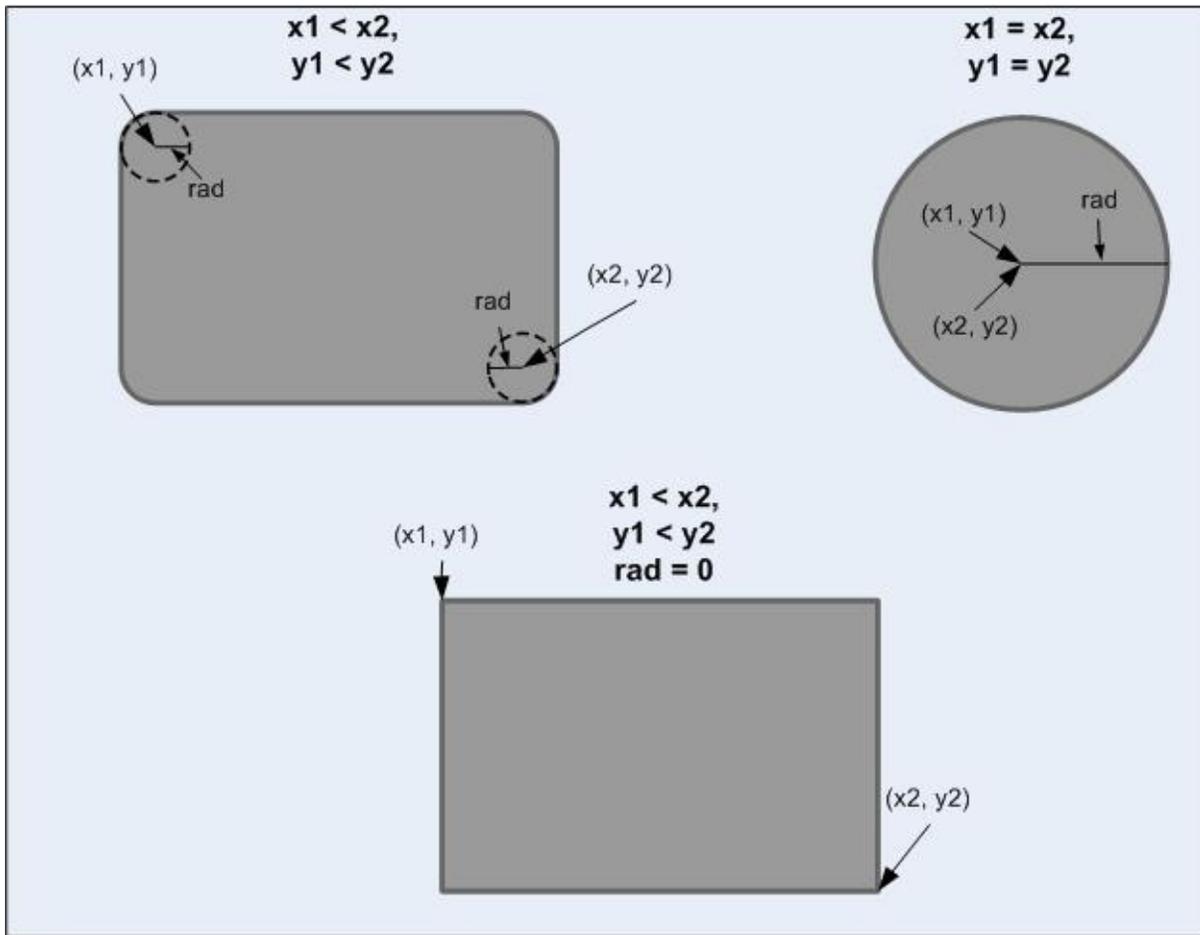
### Overview

---

Draws a filled beveled figure on the screen. For a filled circular object  $x1 = x2$  and  $y1 = y2$ . For a filled rectangular object radius = 0.

### Description

---



## Input Parameters

Input Parameters	Description
SHORT x1	x coordinate position of the upper left center of the circle that draws the rounded corners.
SHORT y1	y coordinate position of the upper left center of the circle that draws the rounded corners.
SHORT x2	x coordinate position of the lower right center of the circle that draws the rounded corners.

SHORT y2	y coordinate position of the lower right center of the circle that draws the rounded corners.
SHORT rad	defines the radius of the circle, that draws the rounded corners.

## Returns

---

For NON-Blocking configuration:

- Returns 0 when device is busy and the shape is not yet completely drawn.
- Returns 1 when the shape is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#) > [FillBevel Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetBevelDrawType Macro

C

```
#define SetBevelDrawType(type) (_bevelDrawType = type)
```

## Overview

This macro sets the fill bevel type to be drawn.

## Input Parameters

Input Parameters	Description
type	is set using the following. <ul style="list-style-type: none"><li>• DRAWFULLBEVEL to draw the full shape</li><li>• DRAWTOPBEVEL to draw the upper half portion</li><li>• DRAWBOTTOMBEVEL to draw the lower half portion</li></ul>

## Returns

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#) > [SetBevelDrawType Macro](#)

# Graphic Cursor

## Macros

This lists the functions to control the graphics cursor.

## Macros

Name	Description
<a href="#">GetX</a>	This macro returns the current graphic cursor x-coordinate.
<a href="#">GetY</a>	This macro returns the current graphic cursor y-coordinate.
<a href="#">MoveRel</a>	This macro moves the graphic cursor relative to the current location. The given dX and dY displacement can be positive or negative numbers.
<a href="#">MoveTo</a>	This macro moves the graphic cursor to new x,y position.

## Links

[Graphics Primitive Layer API, Macros](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Graphic Cursor](#)

# GetX Macro

```
C  
#define GetX _cursorX
```

## Overview

This macro returns the current graphic cursor x-coordinate.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Graphic Cursor](#) > [GetX Macro](#)

# GetY Macro

```
C  
#define GetY _cursorY
```

## Overview

This macro returns the current graphic cursor y-coordinate.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Graphic Cursor](#) > [GetY Macro](#)

## MoveRel Macro

```
C
#define MoveRel(dX, dY) \
    _cursorX += dX;      \
    _cursorY += dY;
```

### Overview

This macro moves the graphic cursor relative to the current location. The given dX and dY displacement can be positive or negative numbers.

### Input Parameters

Input Parameters	Description
dX	Specifies the displacement of the graphic cursor for the horizontal direction.
dY	Specifies the displacement of the graphic cursor for the vertical direction.

### Returns

none

### Preconditions

none

### Side Effects

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Graphic Cursor](#) > [MoveRel Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MoveTo Macro

```
C
#define MoveTo(x, y) \
    _cursorX = x;    \
    _cursorY = y;
```

## Overview

This macro moves the graphic cursor to new x,y position.

## Input Parameters

Input Parameters	Description
x	Specifies the new x position of the graphic cursor.
y	Specifies the new y position of the graphic cursor.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Graphic Cursor](#) > [MoveTo Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Alpha Blending Functions

## Functions | Macros

Alpha-Blend support are two levels. [USE\\_ALPHABLEND\\_LITE](#) and `USE_ALPHA_BLEND`.

### USE\_ALPHABLEND\_LITE

When this macro is enabled in GraphicsConfig.h, Primitive Layer support for an Alpha-Blended [Bar\(\)](#) is enabled.

A rectangular area can be Alpha-Blended with a specific color by doing the following:

1. [SetAlpha\(\)](#) - this will set the Alpha-Blend value
2. [SetColor\(\)](#) - this will define the color that will be Alpha-Blended to the rectangular area
3. [Bar\(left, top, right, bottom\)](#) - this will perform the Alpha-Blending on the rectangular area defined by left, top, right and bottom parameters.

### USE\_ALPHA\_BLEND

This is full Alpha-Blend support but is mainly performed by the display driver used. Windows can be Alpha-Blended using the driver level routines with or without the hardware acceleration. Refer to the specific display driver for support.

## Functions

	Name	Description
	<a href="#">AlphaBlendWindow</a>	This Alpha-Blends a foreground and a background stored in frames to a

destination window. A frame is a memory area that contain array of pixels information. An example would be a display buffer. This operation can be performed on a single frame (where foregroundArea, backgroundArea and destinationArea all points to the same frame), 2 frames (where two of the three areas are pointing to the same frame and one is another frame), or 3 frames (where each area is a separate frame). The Alpha-Blending is performed on the windows inside the specified frames. These windows are defined by the offsets... [more](#)

## Macros

Name	Description
<a href="#">SetAlpha</a>	This macro sets the alpha value. Enabling this feature requires the macros <a href="#">USE_ALPHABLEND_LITE</a> defined in the GraphicsConfig.h. See <a href="#">USE_ALPHABLEND_LITE</a> for information on supported primitive rendering functions.
<a href="#">GetAlpha</a>	This macro returns the current alpha value. Enabling this feature requires the macros <a href="#">USE_ALPHABLEND_LITE</a> defined in the GraphicsConfig.h.

## Links

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#)

---

## [Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetAlpha Macro

```
C
```

```
#define SetAlpha(alpha) (_alpha = alpha)
```

## Overview

This macro sets the alpha value. Enabling this feature requires the macros [USE\\_ALPHABLEND\\_LITE](#) defined in the GraphicsConfig.h. See [USE\\_ALPHABLEND\\_LITE](#) for information on supported primitive rendering functions.

## Input Parameters

Input Parameters	Description
alpha	<p>Defines the alpha blending percentage of the new color set by <a href="#">SetColor()</a> to the existing pixel color. Valid values for alpha for pure primitive layer implementation are:</p> <ul style="list-style-type: none"><li>• 100 : no alpha blending, color set by last <a href="#">SetColor()</a> call will replace the pixels.</li><li>• 75 : alpha blending with new color set by last <a href="#">SetColor()</a> call will be alpha blended with 75% to the existing pixel colors.</li><li>• 50 : alpha blending with new color set by last <a href="#">SetColor()</a> call will be alpha blended with 50% to the existing pixel colors.</li><li>• 25 : alpha blending with new color set by last <a href="#">SetColor()</a> call will be alpha blended with 25% to the existing pixel colors.</li></ul>

---

## Returns

---

None

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
SetAlpha(50);           // set alpha level
SetColor(BLUE);        // set color to use
Bar(5,10,30,50);       // render an alpha blended Bar
```

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#) > [SetAlpha Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetAlpha Macro

```
C
```

```
#define GetAlpha (_alpha)
```

## Overview

This macro returns the current alpha value. Enabling this feature requires the macros [USE\\_ALPHABLEND\\_LITE](#) defined in the GraphicsConfig.h.

## Returns

Returns the current alpha value set by the last call to [SetAlpha\(\)](#).

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#) > [GetAlpha Macro](#)

## AlphaBlendWindow Function

C

```
WORD AlphaBlendWindow(  
    DWORD foregroundArea,  
    SHORT foregroundLeft,  
    SHORT foregroundTop,  
    DWORD backgroundArea,  
    SHORT backgroundLeft,  
    SHORT backgroundTop,  
    DWORD destinationArea,  
    SHORT destinationLeft,  
    SHORT destinationTop,  
    WORD width,  
    WORD height,  
    BYTE alphaPercentage  
);
```

### Overview

This Alpha-Blends a foreground and a background stored in frames to a destination window. A frame is a memory area that contain array of pixels information. An example would be a display buffer. This operation can be performed on a single frame (where foregroundArea, backgroundArea and destinationArea all points to the same frame), 2 frames (where two of the three areas are pointing to the same frame and one is another frame), or 3 frames (where each area is a separate frame). The Alpha-Blending is performed on the windows inside the specified frames. These windows are defined by the offsets for each frame and the given width and height. The Alpha-Blended windows are always equal in sizes. This function is only available when it is supported by the display driver used. Enabling this feature requires the macros [USE\\_ALPHABLEND\\_LITE](#) or

USE\_ALPHABLEND defined in the GraphicsConfig.h.

## Input Parameters

Input Parameters	Description
DWORD foregroundArea	Defines the starting address/page of the foreground window.
SHORT foregroundLeft	Defines the foreground horizontal offset in pixels starting from the starting address/page defined by foregroundArea.
SHORT foregroundTop	Defines the foreground vertical offset in pixels starting from the starting address/page defined by foregroundArea.
DWORD backgroundArea	Defines the starting address/page of the background window.
SHORT backgroundLeft	Defines the background horizontal offset in pixels starting from the starting address/page defined by backgroundArea.
SHORT backgroundTop	Defines the background vertical offset in pixels starting from the starting address/page defined by backgroundArea.
DWORD destinationArea	Defines the starting address/page of the destination window.
SHORT destinationLeft	Defines the destination horizontal offset in pixels starting from the starting address/page defined by destinationArea.
SHORT destinationTop	Defines the destination vertical offset in pixels starting from the starting address/page defined by destinationArea.

WORD width	Defines the width of the window to be alpha blended.
WORD height	Defines the height of the window to be alpha blended.
BYTE alphaPercentage	This defines the amount of transparency to give the foreground <a href="#">Window</a> . Valid range is 0-100. Actual allowed values may be limited by the driver used. Refer to the specific driver for allowed values.

---

## Returns

none

---

## Preconditions

none

---

## Side Effects

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#) > [AlphaBlendWindow Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bitmap Functions

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

This lists the functions to display bitmaps.

## Functions

	Name	Description
◆	<a href="#">PutImagePartial</a>	This function outputs a full or a partial image starting from left,top coordinates. The partial image starts at xoffset and yoffset. Size is specified by the given width and height parameters.
◆	<a href="#">GetImageHeight</a>	This function returns the image height.
◆	<a href="#">GetImageWidth</a>	This function returns the image width.

## Macros

Name	Description
<a href="#">PutImage</a>	This renders the image pointed to by "image" starting from left, top coordinates.

## Structures

Name	Description
<a href="#">BITMAP_HEADER</a>	Structure describing the bitmap header.

## Topics

Name	Description
------	-------------

<a href="#">Bitmap Settings</a>	Bitmap rendering settings.
<a href="#">Bitmap Source</a>	Bitmap data structure is dependent on the location.

## Links

---

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#),  
[Structures](#), [Topics](#)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PutImage Macro

C

```
#define PutImage(left, top, image, stretch) PutImageI
```

## Overview

This renders the image pointed to by "image" starting from left, top coordinates.

## Input Parameters

Input Parameters	Description
left	horizontal starting position of the full image on the screen
top	vertical starting position of the full image on the screen
image	pointer to the image location.
stretch	The image stretch factor. <ul style="list-style-type: none"><li>• <b>IMAGE_NORMAL</b> : no stretch</li><li>• <b>IMAGE_X2</b> : image is stretched to twice its width and height</li></ul>

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the image is not yet completely drawn.
- Returns 1 when the image is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [PutImage Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PutImagePartial Function

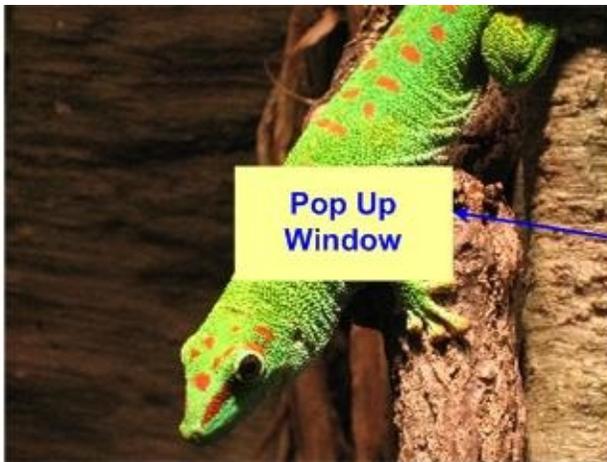
C

```
WORD PutImagePartial(  
    SHORT left,  
    SHORT top,  
    void * image,  
    BYTE stretch,  
    SHORT xoffset,  
    SHORT yoffset,  
    WORD width,  
    WORD height  
);
```

## Overview

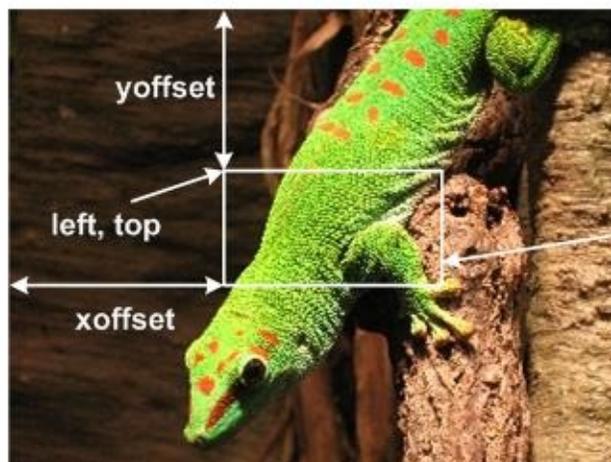
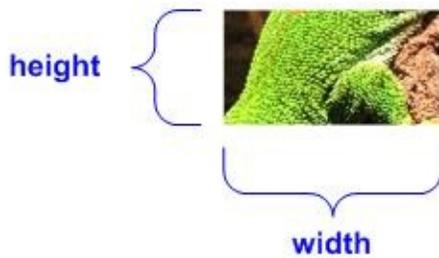
This function outputs a full or a partial image starting from left,top coordinates. The partial image starts at xoffset and yoffset. Size is specified by the given width and height parameters.

## Description



A Pop-Up Window covers a portion of the background image.

When the Pop-Up Window is removed, the area covered needs to be refreshed.



By rendering the partial area that was destroyed by the Pop-Up window refreshing the screen is faster.

## Input Parameters

Input Parameters	Description
------------------	-------------

SHORT left	horizontal starting position of full or partial image on the screen
SHORT top	vertical starting position of full or partial image on the screen,
void * image	pointer to the image location.
BYTE stretch	The image stretch factor. <ul style="list-style-type: none"> <li>• <u>IMAGE_NORMAL</u> : no stretch</li> <li>• <u>IMAGE_X2</u> : image is stretched to twice its width and height</li> </ul>
SHORT xoffset	Specifies the horizontal offset in pixels of the selected partial image from the left most pixel of the full image.
SHORT yoffset	Specifies the vertical offset in pixels of the selected partial image from the top most pixel of the full image.
WORD width	width of the partial image to be rendered. xoffset + width must not exceed the full image width.
WORD height	height of the partial image to be rendered. yoffset + height must not exceed the full image height.

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and the image is not yet completely drawn.
- Returns 1 when the image is completely drawn.

For Blocking configuration:

- Always return 1.

## Side Effects

---

none

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [PutImagePartial Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetImageHeight Function

```
C  
SHORT GetImageHeight(  
    void * bitmap  
);
```

## Overview

This function returns the image height.

## Input Parameters

Input Parameters	Description
void * bitmap	Pointer to the bitmap.

## Returns

Returns the image height in pixels.

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [GetImageHeight Function](#)

# GetWidth Function

```
C  
SHORT GetImageWidth(  
    void * bitmap  
);
```

## Overview

This function returns the image width.

## Input Parameters

Input Parameters	Description
void * bitmap	Pointer to the bitmap.

## Returns

Returns the image width in pixels.

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [GetWidth Function](#)

## BITMAP\_HEADER Structure

C

```
typedef struct {  
    BYTE compression;  
    BYTE colorDepth;  
    SHORT height;  
    SHORT width;  
} BITMAP_HEADER;
```

### Overview

Structure describing the bitmap header.

### Members

Members	Description
BYTE compression;	Compression setting
BYTE colorDepth;	Color depth used
SHORT height;	Image height
SHORT width;	Image width

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [BITMAP\\_HEADER Structure](#)

# Bitmap Settings

## Macros

Bitmap rendering settings.

## Macros

Name	Description
<a href="#"><u>IMAGE_NORMAL</u></a>	Normal image stretch code
<a href="#"><u>IMAGE_X2</u></a>	Stretched image stretch code

## Links

[Bitmap Functions, Macros](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [Bitmap Settings](#)

# IMAGE\_NORMAL Macro

---

```
C
```

```
#define IMAGE_NORMAL 1
```

## Description

---

Normal image stretch code

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [Bitmap Settings](#) > [IMAGE\\_NORMAL Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# IMAGE\_X2 Macro

---

C

```
#define IMAGE_X2 2
```

## Description

---

Stretched image stretch code

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [Bitmap Settings](#) > [IMAGE\\_X2 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bitmap Source

---

Bitmap data structure is dependent on the location.

## Links

---

### [Bitmap Functions](#)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [Bitmap Source](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# External Memory

[Functions](#) | [Macros](#) | [Topics](#)

Bitmaps and Fonts can be located in external memory. To refer the data, EXTDATA structure is used:

```
Copy Code  
typedef struct _EXTDATA_  
{  
    TYPE_MEMORY    type;           // must be set to EX  
    WORD           ID;             // memory ID  
    DWORD          address;        // bitmap or font im  
} EXTDATA;
```

where:

- type – shows type of memory used.
  - 0 – internal
  - 1 - external.
- ID – unique number must be assigned by application to have a way distinguishing memory chips if the application has several of them.
- address – start address of the bitmap or font image in the external memory.

To use the bitmap or font the pointer to EXTDATA structure must be passed into corresponding function ([PutImage\(\)](#) or [SetFont\(\)](#)). Each time the library will need data it will call special call back function [ExternalMemoryCallback\(\)](#).

This function must be implemented in the application. Inside, the application must copy requested bytes quantity into the buffer provided. Data start address can be calculated as a sum of the start image address specified in EXTDATA structure and offset provided.

## Functions

	Name	Description
	<a href="#"><u>ExternalMemoryCallback</u></a>	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 bytes quantity into the buffer provided. Data start address in external memory is a sum of the address in <a href="#"><u>GFX_EXTDATA</u></a> structure and offset.

## Macros

Name	Description
<a href="#"><u>EXTERNAL_FONT_BUFFER_SIZE</u></a>	This defines the size of the buffer used by font functions to retrieve font data from the external memory. The buffer size can be increased to accommodate large font sizes. The user must be aware of the expected glyph sizes of the characters stored in the font table. To modify the size used, declare this macro in the GraphicsConfig.h file with the desired size.

## Topics

Name	Description
<a href="#"><u>Memory Type</u></a>	Memory type enumeration to determine the

source of data. Used in interpreting bitmap and font from different memory sources.

## Links

---

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#), [Macros](#), [Topics](#)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ExternalMemoryCallback Function

```
C
WORD ExternalMemoryCallback(
    GFX_EXTDATA * memory,
    LONG offset,
    WORD nCount,
    void * buffer
);
```

### Overview

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 bytes quantity into the buffer provided. Data start address in external memory is a sum of the address in [GFX\\_EXTDATA](#) structure and offset.

### Input Parameters

Input Parameters	Description
GFX_EXTDATA * memory	Pointer to the external memory bitmap or font structures ( <a href="#">FONT_EXTERNAL</a> or <a href="#">BITMAP_EXTERNAL</a> ).
LONG offset	Data offset.
WORD nCount	Number of bytes to be transferred into the buffer.
void * buffer	Pointer to the buffer.

### Returns

Returns the number of bytes were transferred.

## Side Effects

---

none

## Example

---

[Copy Code](#)

```
// If there are several memories in the system they  
// In this example, ID for memory device used is as  
#define X_MEMORY 0  
  
WORD ExternalMemoryCallback(GFX_EXTDATA* memory, LO  
    int i;  
    long address;  
  
    // Address of the requested data is a start add  
    address = memory->address+offset;  
  
    if(memory->ID == X_MEMORY){  
        // MemoryXReadByte() is some function imple  
        // Implementation will be specific to the m  
        // it reads byte each time it is called.  
        i = 0;  
        while (i < nCount) {  
            (BYTE*)buffer = MemoryXReadByte(address  
            i++;  
        }  
    }  
    // return the actual number of bytes retrieved  
    return (i);  
}
```

---

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#) > [ExternalMemoryCallback Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# EXTERNAL\_FONT\_BUFFER\_SIZE Macro

```
C
```

```
#define EXTERNAL_FONT_BUFFER_SIZE 600
```

## Overview

This defines the size of the buffer used by font functions to retrieve font data from the external memory. The buffer size can be increased to accommodate large font sizes. The user must be aware of the expected glyph sizes of the characters stored in the font table. To modify the size used, declare this macro in the GraphicsConfig.h file with the desired size.

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#) > [EXTERNAL\\_FONT\\_BUFFER\\_SIZE Macro](#)

# Memory Type

---

Memory type enumeration to determine the source of data. Used in interpreting bitmap and font from different memory sources.

## Links

---

[External Memory](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#) > [Memory Type](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Set Up Functions

## Functions

This lists the Primitive set up and initialization functions.

## Functions

	Name	Description
	<a href="#"><u>ClearDevice</u></a>	This function clears the screen with the current color and sets the graphic cursor position to (0, 0). Clipping is NOT supported by ClearDevice().
	<a href="#"><u>InitGraph</u></a>	This function initializes the display controller, sets the line type to <a href="#"><u>SOLID_LINE</u></a> , sets the screen to all <a href="#"><u>BLACK</u></a> , sets the current drawing color to <a href="#"><u>WHITE</u></a> , sets the graphic cursor position to upper left corner of the screen, sets active and visual pages to page #0, clears the active page and disables clipping. This function should be called before using the Graphics Primitive Layer.

## Links

[Functions](#), [Graphics Primitive Layer API](#), [Legend](#)

[Library API](#) > [Graphics Primitive Layer API](#) > [Set Up Functions](#)

# ClearDevice Function

**C**

```
void ClearDevice();
```

## Overview

This function clears the screen with the current color and sets the graphic cursor position to (0, 0). Clipping is NOT supported by ClearDevice().

## Returns

none

## Side Effects

none

## Example

[Copy Code](#)

```
void ClearScreen(void)
{
    SetColor(WHITE);           // set color to WHITE
    ClearDevice();             // set screen to all WH
}
```

[Library API](#) > [Graphics Primitive Layer API](#) > [Set Up Functions](#) > [ClearDevice Function](#)

# InitGraph Function

```
C
```

```
void InitGraph();
```

## Overview

This function initializes the display controller, sets the line type to [SOLID\\_LINE](#), sets the screen to all [BLACK](#), sets the current drawing color to [WHITE](#), sets the graphic cursor position to upper left corner of the screen, sets active and visual pages to page #0, clears the active page and disables clipping. This function should be called before using the Graphics Primitive Layer.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Graphics Primitive Layer API](#) > [Set Up Functions](#) > [InitGraph Function](#)

## GFX\_RESOURCE Enumeration

C

```
typedef enum {  
    FLASH = 0x0000,  
    EXTERNAL = 0x0001,  
    FLASH_JPEG = 0x0002,  
    EXTERNAL_JPEG = 0x0003,  
    RAM = 0x0004,  
    EDS_EPMP = 0x0005,  
    IMAGE_MBITMAP = 0x0000,  
    IMAGE_JPEG = 0x0100,  
    COMP_NONE = 0x0000,  
    COMP_RLE = 0x1000,  
    COMP_IPU = 0x2000  
} GFX_RESOURCE;
```

### Overview

Memory type enumeration to determine the source of data. Used in interpreting bitmap and font from different memory sources.

### Members

Members	Description
FLASH = 0x0000	internal flash
EXTERNAL = 0x0001	external memory
FLASH_JPEG = 0x0002	internal flash
EXTERNAL_JPEG = 0x0003	external memory

RAM = 0x0004	RAM
EDS_EPMP = 0x0005	memory in EPMP, base addresses are are set in the hardware profile
IMAGE_MBITMAP = 0x0000	data resource is type Microchip bitmap
IMAGE_JPEG = 0x0100	data resource is type JPEG
COMP_NONE = 0x0000	no compression
COMP_RLE = 0x1000	compressed with RLE
COMP_IPU = 0x2000	compressed with DEFLATE (for IPU)

---

## [Library API](#) > [Graphics Primitive Layer API](#) > [GFX\\_RESOURCE Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_IMAGE\_HEADER Structure

C

```
typedef struct {
    GFX_RESOURCE type;
    WORD ID;
    union {
        DWORD extAddress;
        FLASH_BYTE * progByteAddress;
        FLASH_WORD * progWordAddress;
        const char * constAddress;
        char * ramAddress;
        __eds__ char * edsAddress;
    } LOCATION;
    WORD width;
    WORD height;
    DWORD param1;
    DWORD param2;
    WORD colorDepth;
} GFX_IMAGE_HEADER;
```

### Overview

Structure for images stored in various system memory (Flash, External Memory (SPI, Parallel Flash, or memory in EPMP).

### Members

Members	Description
GFX_RESOURCE type;	Graphics resource type, determines the type and location of data
WORD ID;	memory ID, user defined value to differentiate between graphics resources of

	<p>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</p>
DWORD extAddress;	generic address
FLASH_BYTE * progByteAddress;	for addresses in program section
FLASH_WORD * progWordAddress;	for addresses in program section
const char * constAddress;	for addresses in FLASH
char * ramAddress;	for addresses in RAM
__eds__ char * edsAddress;	for addresses in EDS
WORD width;	width of the image
WORD height;	height of the image
DWORD param1;	Parameters used for the <a href="#">GFX_RESOURCE</a> . Depending on the <a href="#">GFX_RESOURCE</a> type definition of param1 can change. For IPU and RLE compressed images, param1 indicates the compressed size of the image.
DWORD param2;	Parameters used for the <a href="#">GFX_RESOURCE</a> . Depending on the <a href="#">GFX_RESOURCE</a> type definition of param2 can change. For IPU and RLE compressed images, param2

	indicates the uncompressed size of the image.
WORD colorDepth;	color depth of the image

---

[Library API](#) > [Graphics Primitive Layer API](#) > [GFX\\_IMAGE\\_HEADER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## IMAGE\_FLASH Structure

**C**

```
typedef struct {  
    GFX_RESOURCE type;  
    FLASH_BYTE * address;  
} IMAGE_FLASH;
```

### Overview

Structure for images stored in FLASH memory.

### Members

Members	Description
GFX_RESOURCE type;	must be FLASH
FLASH_BYTE * address;	image address in FLASH

[Library API](#) > [Graphics Primitive Layer API](#) > [IMAGE\\_FLASH Structure](#)

## IMAGE\_RAM Structure

C

```
typedef struct {  
    GFX_RESOURCE type;  
    DWORD * address;  
} IMAGE_RAM;
```

### Overview

Structure for images stored in RAM memory.

### Members

Members	Description
GFX_RESOURCE type;	must be RAM
DWORD * address;	image address in RAM

[Library API](#) > [Graphics Primitive Layer API](#) > [IMAGE\\_RAM Structure](#)

## GFX\_EXTDATA Structure

C

```
typedef struct {  
    GFX_RESOURCE type;  
    WORD ID;  
    DWORD address;  
} GFX_EXTDATA;
```

### Overview

This structure is used to describe external memory.

### Members

Members	Description
GFX_RESOURCE type;	Resource type. Valid types are: <ul style="list-style-type: none"><li>EXTERNAL</li><li>EDS_EPMP</li></ul>
WORD 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 <ul style="list-style-type: none"><li>0 - reserved (do not use)</li><li>1 - reserved for base address of EPMP CS1</li><li>2 - reserved for base address of EPMP CS2</li></ul>
DWORD address;	Data image address (user data, bitmap or font)

---

Library API > Graphics Primitive Layer API > GFX\_EXTDATA Structure

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Display Device Driver Layer API

[Modules](#) | [Topics](#)

## Modules

Name	Description
<a href="#">Advanced Display Driver Features</a>	This section lists advanced Display Device Driver Features implemented in select Display Device Driver.

## Topics

Name	Description
<a href="#">Display Device Driver Level Primitives</a>	This lists the Device Level Primitive rendering functions and macros.
<a href="#">Display Device Driver Control</a>	This lists the device control functions and macros.

## Links

[Library API](#), [Modules](#), [Topics](#)

[Library API](#) > [Display Device Driver Layer API](#)

# Display Device Driver Level Primitives

[Functions](#) | [Macros](#) | [Topics](#)

## Functions

	Name	Description
◆	<a href="#">GetPixel</a>	Returns pixel color at the given x,y coordinate position.
◆	<a href="#">PutPixel</a>	Puts pixel with the given x,y coordinate position.
◆	<a href="#">SetClip</a>	Enables/disables clipping.
◆	<a href="#">SetClipRgn</a>	Sets clipping region.
◆	<a href="#">TransparentColorEnable</a>	Sets current transparent color. <a href="#">PutImage()</a> will not render pixels that matches the set transparent color. To enable Transparent Color feature, define the macro <a href="#">USE_TRANSPARENT_COLOR</a> in the GraphicsConfig.h file.
◆	<a href="#">DisplayBrightness</a>	Sets the brightness of the display.
◆	<a href="#">CopyBlock</a>	Copies a block of data from source specified by srcAddr and srcOffset to the destination specified by dstAddr and dstOffset. This is similar to the <a href="#">CopyWindow()</a> and but instead of using left, top position of the source and destination, it uses offsets instead. This is a blocking function.

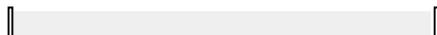
◆	<a href="#"><u>CopyPageWindow</u></a>	This is a blocking call. A windows is a rectangular area with the given width and height of a page. The source and destination window can be located in different pages and each page is assumed to have the same dimensions (equal width and height).
◆	<a href="#"><u>CopyWindow</u></a>	A windows is a rectangular area with the given width and height located in the given base source address. The source and destination window can be located in the same base address. If this is the case, then srcAddr = dstAddr. The operation is similar to <a href="#"><u>CopyPageWindow</u></a> () but instead of using page numbers, addresses are used for versatility. This is a blocking function.
◆	<a href="#"><u>SetActivePage</u></a>	Sets active graphic page.
◆	<a href="#"><u>SetVisualPage</u></a>	Sets graphic page to display.

## Macros

Name	Description
<a href="#"><u>GetColor</u></a>	Returns current drawing color.
<a href="#"><u>SetColor</u></a>	Sets current drawing color.
<a href="#"><u>GetMaxX</u></a>	Returns maximum horizontal coordinate.

<a href="#"><u>GetMaxY</u></a>	Returns maximum vertical coordinate.
<a href="#"><u>GetClipBottom</u></a>	Returns bottom clipping border.
<a href="#"><u>GetClipLeft</u></a>	Returns left clipping border.
<a href="#"><u>GetClipRight</u></a>	Returns right clipping border.
<a href="#"><u>GetClipTop</u></a>	Returns top clipping border.
<a href="#"><u>CLIP_DISABLE</u></a>	Disables clipping.
<a href="#"><u>CLIP_ENABLE</u></a>	Enables clipping.
<a href="#"><u>TransparentColorDisable</u></a>	Disables the transparent color function.
<a href="#"><u>GetTransparentColorStatus</u></a>	Returns the current transparent color function enable status.
<a href="#"><u>GetTransparentColor</u></a>	Returns the current transparent color value.
<a href="#"><u>TRANSPARENT_COLOR_DISABLE</u></a>	Check of transparent color is not performed
<a href="#"><u>TRANSPARENT_COLOR_ENABLE</u></a>	Check pixel if color is equal to transparent color, if equal do not render pixel
<a href="#"><u>GetPageAddress</u></a>	Returns the address of the given page.

## Topics



Name	Description
<a href="#">Color Definition</a>	The device driver must also implement the definition of standard color set based on the color format used.

## Links

---

[Functions](#), [Display Device Driver Layer API](#), [Legend](#), [Macros](#), [Topics](#)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetPixel Function

```
C
GFX_COLOR GetPixel(
    SHORT x,
    SHORT y
);
```

## Overview

Returns pixel color at the given x,y coordinate position.

## Input Parameters

Input Parameters	Description
SHORT x	x position of the pixel.
SHORT y	y position of the pixel.

## Returns

pixel color

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetPixel Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PutPixel Function

```
C
void PutPixel(
    SHORT x,
    SHORT y
);
```

## Overview

Puts pixel with the given x,y coordinate position.

## Input Parameters

Input Parameters	Description
SHORT x	x position of the pixel.
SHORT y	y position of the pixel.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [PutPixel Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetColor Macro

```
C
#define GetColor _color
```

## Overview

Returns current drawing color.

## Returns

Color where coding is based on GFX\_COLOR definition. GFX\_COLOR definition is based on the color depth ([COLOR\\_DEPTH](#)) used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetColor Macro](#)

# SetColor Macro

```
C
#define SetColor(color) _color = (color)
```

## Overview

Sets current drawing color.

## Input Parameters

Input Parameters	Description
color	Color coding is based on GFX_COLOR definition. GFX_COLOR definition is based on the color depth ( <a href="#">COLOR_DEPTH</a> ) used.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [SetColor Macro](#)

# GetMaxX Macro

C

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

## Overview

Returns maximum horizontal coordinate.

## Returns

Maximum horizontal coordinate.

## Preconditions

none

## Side Effects

none

## Example

[Copy Code](#)

```
// Create a window that will occupy the whole screen  
WndCreate(0xFF, // ID  
          0, 0,  
          GetMaxX(), GetMaxY(), // dimension  
          WND_DRAW, // will be displayed  
          (void*)&mchpIcon, // use icon use  
          pText, // set to text  
          NULL); // use default
```

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetMaxX Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetMaxY Macro

```
C
```

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

## Overview

Returns maximum vertical coordinate.

## Returns

Maximum vertical coordinate.

## Preconditions

none

## Side Effects

none

## Example

(see [GetMaxX\(\)](#) example.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetMaxY Macro](#)

# SetClip Function

```
C  
void SetClip(  
    BYTE control  
);
```

## Overview

Enables/disables clipping.

## Input Parameters

Input Parameters	Description
BYTE control	Enables or disables the clipping. <ul style="list-style-type: none"><li>• <a href="#">CLIP_DISABLE</a>: Disable clipping</li><li>• <a href="#">CLIP_ENABLE</a>: Enable clipping</li></ul>

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [SetClip Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetClipRgn Function

C

```
void SetClipRgn(  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom  
);
```

## Overview

Sets clipping region.

## Input Parameters

Input Parameters	Description
SHORT left	Defines the left clipping region border.
SHORT top	Defines the top clipping region border.
SHORT right	Defines the right clipping region border.
SHORT bottom	Defines the bottom clipping region border.

## Returns

none

## Preconditions

none

## Side Effects

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [SetClipRgn Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetClipBottom Macro

```
C
```

```
#define GetClipBottom _clipBottom
```

## Overview

Returns bottom clipping border.

## Returns

Bottom clipping border.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetClipBottom Macro](#)

# GetClipLeft Macro

---

```
C
```

```
#define GetClipLeft _clipLeft
```

## Overview

---

Returns left clipping border.

## Returns

---

Left clipping border.

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetClipLeft Macro](#)

---

# GetClipRight Macro

```
C
```

```
#define GetClipRight _clipRight
```

## Overview

Returns right clipping border.

## Returns

Right clipping border.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetClipRight Macro](#)

# GetClipTop Macro

---

```
C
#define GetClipTop _clipTop
```

## Overview

---

Returns top clipping border.

## Returns

---

Top clipping border.

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetClipTop Macro](#)

## CLIP\_DISABLE Macro

---

```
C
#define CLIP_DISABLE 0 // Disables clipping.
```

### Description

---

Disables clipping.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [CLIP\\_DISABLE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CLIP\_ENABLE Macro

---

C

```
#define CLIP_ENABLE 1 // Enables clipping.
```

## Description

---

Enables clipping.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [CLIP\\_ENABLE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TransparentColorEnable Function

---

C

```
void TransparentColorEnable(  
    GFX_COLOR color  
);
```

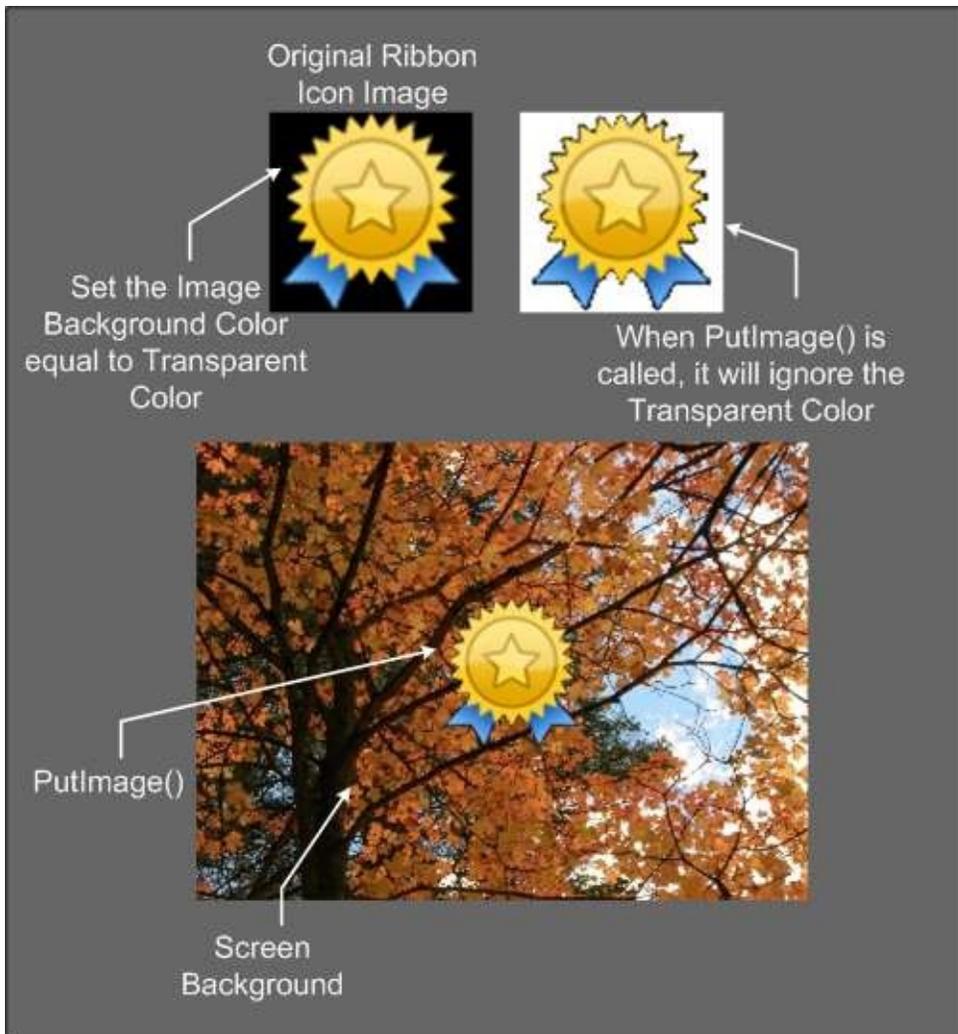
## Overview

---

Sets current transparent color. [PutImage\(\)](#) will not render pixels that matches the set transparent color. To enable Transparent Color feature, define the macro [USE\\_TRANSPARENT\\_COLOR](#) in the GraphicsConfig.h file.

## Description

---



## Input Parameters

Input Parameters	Description
GFX_COLOR color	Chosen transparent color.

## Returns

none

## Preconditions

none

## Side Effects

---

None

## Example

---

[Copy Code](#)

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

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [TransparentColorEnable Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# TransparentColorDisable Macro

```
C
```

```
#define TransparentColorDisable _colorTransparentEnal
```

## Overview

Disables the transparent color function.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [TransparentColorDisable Macro](#)

# GetTransparentColorStatus Macro

```
C
```

```
#define GetTransparentColorStatus _colorTransparentE
```

## Overview

Returns the current transparent color function enable status.

## Returns

Returns the current transparent color function enable status

[Copy Code](#)

```
0 - Transparent color function is disabled.  
1 - Transparent color function is enabled.
```

## Preconditions

none

## Side Effects

None

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetTransparentColorStatus Macro](#)

# GetTransparentColor Macro

```
C
```

```
#define GetTransparentColor _colorTransparent
```

## Overview

Returns the current transparent color value.

## Returns

Returns the current transparent color used.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetTransparentColor Macro](#)

# TRANSPARENT\_COLOR\_DISABLE Macro

---

C

```
#define TRANSPARENT_COLOR_DISABLE 0 // Check of tra
```

## Description

---

Check of transparent color is not performed

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [TRANSPARENT\\_COLOR\\_DISABLE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TRANSPARENT\_COLOR\_ENABLE Macro

C

```
#define TRANSPARENT_COLOR_ENABLE 1 // Check pixel .
```

### Description

Check pixel if color is equal to transparent color, if equal do not render pixel

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [TRANSPARENT\\_COLOR\\_ENABLE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DisplayBrightness Function

```
C  
void DisplayBrightness(  
    WORD level  
);
```

## Overview

Sets the brightness of the display.

## Input Parameters

Input Parameters	Description
WORD level	Brightness level. Valid values are 0 to 100. <ul style="list-style-type: none"><li>• 0: brightness level is zero or display is turned off</li><li>• 1: brightness level is maximum</li></ul>

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [DisplayBrightness Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GetPageAddress Macro

```
C
```

```
#define GetPageAddress(page) (_PageTable[page])
```

## Overview

Returns the address of the given page.

## Returns

Address in memory of the given page number. The number of pages is dictated by GFX\_DRV\_PAGE\_COUNT value defined in the hardware profile. GFX\_DRV\_PAGE\_COUNT is not mandatory. Drivers that do not have enough memory for paging may not define this constant. If GFX\_DRV\_PAGE\_COUNT is not defined, all API's related to paging operation is will not be available.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [GetPageAddress Macro](#)

# CopyBlock Function

C

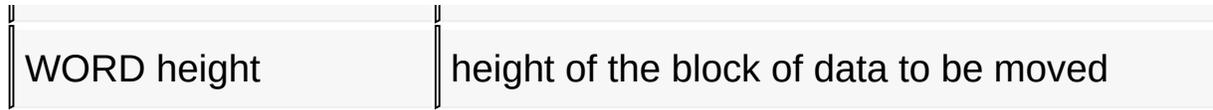
```
WORD CopyBlock(  
    DWORD srcAddr,  
    DWORD dstAddr,  
    DWORD srcOffset,  
    DWORD dstOffset,  
    WORD width,  
    WORD height  
);
```

## Overview

Copies a block of data from source specified by `srcAddr` and `srcOffset` to the destination specified by `dstAddr` and `dstOffset`. This is similar to the [CopyWindow\(\)](#) and but instead of using left, top position of the source and destination, it uses offsets instead. This is a blocking function.

## Input Parameters

Input Parameters	Description
DWORD <code>srcAddr</code>	the base address of the data to be moved
DWORD <code>dstAddr</code>	the base address of the new location of the moved data
DWORD <code>srcOffset</code>	offset of the data to be moved with respect to the source base address.
DWORD <code>dstOffset</code>	offset of the new location of the moved data respect to the source base address.
WORD <code>width</code>	width of the block of data to be moved



---

## Returns

none

---

## Preconditions

none

---

## Side Effects

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [CopyBlock Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## CopyPageWindow Function

C

```
void CopyPageWindow(  
    BYTE srcPage,  
    BYTE dstPage,  
    \ WORD srcX,  
    WORD srcY,  
    \ WORD dstX,  
    WORD dstY,  
    \ WORD width,  
    WORD height  
);
```

### Overview

This is a blocking call. A windows is a rectangular area with the given width and height of a page. The source and destination window can be located in different pages and each page is assumed to have the same dimensions (equal width and height).

### Input Parameters

Input Parameters	Description
BYTE srcPage	page number of the source window,
BYTE dstPage	page number of the destination window,
\ WORD srcX	x location of the left top corner of the source window respect to the srcPage.
WORD srcY	y location of the left top corner of the source window respect to the srcPage.
\ WORD dstX	x location of the left top corner of the

	destination window respect to the dstPage.
WORD dstY	y location of the left top corner of the destination window respect to the dstPage.
\ WORD width	the width in pixels of the window to copy
WORD height	the height in pixels of the window to copy

## Returns

---

None

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [CopyPageWindow Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CopyWindow Function

**C**

```
WORD CopyWindow(  
    DWORD srcAddr,  
    DWORD dstAddr,  
    \ WORD srcX,  
    WORD srcY,  
    \ WORD dstX,  
    WORD dstY,  
    \ WORD width,  
    WORD height  
);
```

## Overview

A window is a rectangular area with the given width and height located in the given base source address. The source and destination window can be located in the same base address. If this is the case, then `srcAddr = dstAddr`. The operation is similar to [CopyPageWindow\(\)](#) but instead of using page numbers, addresses are used for versatility. This is a blocking function.

## Input Parameters

Input Parameters	Description
DWORD srcAddr	Base Address of the source window,
\ WORD srcX	x location of the left top corner of the source window respect to the srcPage.
WORD srcY	y location of the left top corner of the source window respect to the srcPage.

\ WORD dstX	x location of the left top corner of the destination window respect to the dstPage.
WORD dstY	y location of the left top corner of the destination window respect to the dstPage.
\ WORD width	the width in pixels of the window to copy
WORD height	the height in pixels of the window to copy
dstPage	Base Address of the destination window,

---

## Returns

None

---

## Preconditions

none

---

## Side Effects

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [CopyWindow Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetActivePage Function

```
C  
void SetActivePage(  
    WORD page  
);
```

## Overview

Sets active graphic page.

## Description

Functions: SetActivePage(page)

## Input Parameters

Input Parameters	Description
WORD page	Graphic page number.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [SetActivePage Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetVisualPage Function

```
C  
void SetVisualPage(  
    WORD page  
);
```

## Overview

Sets graphic page to display.

## Description

Functions: SetVisualPage(page)

## Input Parameters

Input Parameters	Description
WORD page	Graphic page number

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [SetVisualPage Function](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Color Definition

## Macros

The following lists sample color definitions for a 16-bpp color encoding.

## Macros

Name	Description
<a href="#"><u>BLACK</u></a>	not <a href="#"><u>USE_PALETTE</u></a>
<a href="#"><u>BLUE</u></a>	This is macro BLUE.
<a href="#"><u>BRIGHTBLUE</u></a>	This is macro BRIGHTBLUE.
<a href="#"><u>BRIGHTCYAN</u></a>	This is macro BRIGHTCYAN.
<a href="#"><u>BRIGHTGREEN</u></a>	This is macro BRIGHTGREEN.
<a href="#"><u>BRIGHTMAGENTA</u></a>	This is macro BRIGHTMAGENTA.
<a href="#"><u>BRIGHTRED</u></a>	This is macro BRIGHTRED.
<a href="#"><u>BRIGHTYELLOW</u></a>	This is macro BRIGHTYELLOW.
<a href="#"><u>BROWN</u></a>	This is macro BROWN.
<a href="#"><u>CYAN</u></a>	This is macro CYAN.
<a href="#"><u>DARKGRAY</u></a>	This is macro DARKGRAY.
<a href="#"><u>GRAY0</u></a>	This is macro GRAY0.
<a href="#"><u>GRAY1</u></a>	This is macro GRAY1.
<a href="#"><u>GRAY2</u></a>	This is macro GRAY2.

<a href="#"><u>GRAY3</u></a>	This is macro GRAY3.
<a href="#"><u>GRAY4</u></a>	This is macro GRAY4.
<a href="#"><u>GRAY5</u></a>	This is macro GRAY5.
<a href="#"><u>GRAY6</u></a>	This is macro GRAY6.
<a href="#"><u>GREEN</u></a>	This is macro GREEN.
<a href="#"><u>LIGHTBLUE</u></a>	This is macro LIGHTBLUE.
<a href="#"><u>LIGHTCYAN</u></a>	This is macro LIGHTCYAN.
<a href="#"><u>LIGHTGRAY</u></a>	This is macro LIGHTGRAY.
<a href="#"><u>LIGHTGREEN</u></a>	This is macro LIGHTGREEN.
<a href="#"><u>LIGHTMAGENTA</u></a>	This is macro LIGHTMAGENTA.
<a href="#"><u>LIGHTRED</u></a>	This is macro LIGHTRED.
<a href="#"><u>MAGENTA</u></a>	This is macro MAGENTA.
<a href="#"><u>RED</u></a>	This is macro RED.
<a href="#"><u>WHITE</u></a>	This is macro WHITE.
<a href="#"><u>YELLOW</u></a>	This is macro YELLOW.

## Links

### [Display Device Driver Level Primitives, Macros](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#)



# BLACK Macro

C

```
#define BLACK RGBConvert(0, 0, 0)
```

## Description

not USE\_PALETTE

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BLACK Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BLUE Macro

---

C

```
#define BLUE RGBConvert(0, 0, 128)
```

## Description

---

This is macro BLUE.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BLUE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BRIGHTBLUE Macro

---

C

```
#define BRIGHTBLUE RGBConvert(0, 0, 255)
```

## Description

---

This is macro BRIGHTBLUE.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BRIGHTBLUE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BRIGHTCYAN Macro

C

```
#define BRIGHTCYAN RGBConvert(0, 255, 255)
```

## Description

This is macro BRIGHTCYAN.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BRIGHTCYAN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BRIGHTGREEN Macro

---

C

```
#define BRIGHTGREEN RGBConvert(0, 255, 0)
```

## Description

---

This is macro BRIGHTGREEN.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BRIGHTGREEN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BRIGHTMAGENTA Macro

---

C

```
#define BRIGHTMAGENTA RGBConvert(255, 0, 255)
```

## Description

---

This is macro BRIGHTMAGENTA.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BRIGHTMAGENTA Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BRIGHTRED Macro

C

```
#define BRIGHTRED RGBConvert(255, 0, 0)
```

## Description

This is macro BRIGHTRED.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BRIGHTRED Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BRIGHTYELLOW Macro

---

C

```
#define BRIGHTYELLOW RGBConvert(255, 255, 0)
```

## Description

---

This is macro BRIGHTYELLOW.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BRIGHTYELLOW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# BROWN Macro

---

C

```
#define BROWN RGBConvert(255, 128, 0)
```

## Description

---

This is macro BROWN.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [BROWN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# CYAN Macro

---

C

```
#define CYAN RGBConvert(0, 128, 128)
```

## Description

---

This is macro CYAN.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [CYAN Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# DARKGRAY Macro

---

C

```
#define DARKGRAY RGBConvert(64, 64, 64)
```

## Description

---

This is macro DARKGRAY.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [DARKGRAY Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY0 Macro

---

C

```
#define GRAY0 RGBConvert(224, 224, 224)
```

## Description

---

This is macro GRAY0.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY0 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY1 Macro

C

```
#define GRAY1 RGBConvert(192, 192, 192)
```

## Description

This is macro GRAY1.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY1 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY2 Macro

C

```
#define GRAY2 RGBConvert(160, 160, 160)
```

## Description

This is macro GRAY2.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY2 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY3 Macro

C

```
#define GRAY3 RGBConvert(128, 128, 128)
```

## Description

This is macro GRAY3.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY3 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY4 Macro

---

C

```
#define GRAY4 RGBConvert(96, 96, 96)
```

## Description

---

This is macro GRAY4.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY4 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY5 Macro

C

```
#define GRAY5 RGBConvert(64, 64, 64)
```

## Description

This is macro GRAY5.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY5 Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GRAY6 Macro

---

C

```
#define GRAY6 RGBConvert(32, 32, 32)
```

## Description

---

This is macro GRAY6.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GRAY6 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GREEN Macro

C

```
#define GREEN RGBConvert(0, 128, 0)
```

## Description

This is macro GREEN.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [GREEN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LIGHTBLUE Macro

C

```
#define LIGHTBLUE RGBConvert(128, 128, 255)
```

## Description

This is macro LIGHTBLUE.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [LIGHTBLUE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LIGHTCYAN Macro

**C**

```
#define LIGHTCYAN RGBConvert(128, 255, 255)
```

## Description

This is macro LIGHTCYAN.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [LIGHTCYAN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LIGHTGRAY Macro

C

```
#define LIGHTGRAY RGBConvert(128, 128, 128)
```

## Description

This is macro LIGHTGRAY.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [LIGHTGRAY Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LIGHTGREEN Macro

C

```
#define LIGHTGREEN RGBConvert(128, 255, 128)
```

## Description

This is macro LIGHTGREEN.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [LIGHTGREEN Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LIGHTMAGENTA Macro

---

**C**

```
#define LIGHTMAGENTA RGBConvert(255, 128, 255)
```

## Description

---

This is macro LIGHTMAGENTA.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [LIGHTMAGENTA Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# LIGHTRED Macro

---

C

```
#define LIGHTRED RGBConvert(255, 128, 128)
```

## Description

---

This is macro LIGHTRED.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [LIGHTRED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# MAGENTA Macro

C

```
#define MAGENTA RGBConvert(128, 0, 128)
```

## Description

This is macro MAGENTA.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [MAGENTA Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RED Macro

---

C

```
#define RED RGBConvert(128, 0, 0)
```

## Description

---

This is macro RED.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [RED Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# WHITE Macro

---

C

```
#define WHITE RGBConvert(255, 255, 255)
```

## Description

---

This is macro WHITE.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [WHITE Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# YELLOW Macro

---

C

```
#define YELLOW RGBConvert(255, 255, 128)
```

## Description

---

This is macro YELLOW.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#) > [YELLOW Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Display Device Driver Control

## Functions

### Functions

	Name	Description
	<a href="#"><u>IsDeviceBusy</u></a>	Returns non-zero if LCD controller is busy (previous drawing operation is not completed).
	<a href="#"><u>ResetDevice</u></a>	Initializes LCD module.

### Links

[Functions](#), [Display Device Driver Layer API](#), [Legend](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Control](#)

# IsDeviceBusy Function

```
C
```

```
WORD IsDeviceBusy();
```

## Overview

Returns non-zero if LCD controller is busy (previous drawing operation is not completed).

## Returns

Busy status.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Control](#) > [IsDeviceBusy Function](#)

# ResetDevice Function

```
C
```

```
void ResetDevice();
```

## Overview

Initializes LCD module.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Control](#) > [ResetDevice Function](#)

# Advanced Display Driver Features

## Topics

## Topics

Name	Description
<a href="#">Alpha Blending</a>	The following APIs are used to implement alpha blending features in the Epson S1D13517 Controller.
<a href="#">Transitions</a>	This section describes screen transition effect when changing screens.
<a href="#">Double Buffering</a>	<p>In the Microchip Graphics Library, if double-buffering is enabled, the frame buffer and draw buffer are exchanged after the changes of all the widgets on a screen are done (i.e., the new screen appears after the whole screen is updated and not after updating an individual widget). This feature is enabled only on the following drivers:</p> <ul style="list-style-type: none"><li>• Microchip Graphics Display Driver - customizable driver for RGB Glass. Currently used in PIC24FJ256DA210 device family.</li><li>• Microchip Low-Cost Controllerless (LCC) Graphics Display Driver - customizable driver for RGB Glass. Currently used for selected PIC32MX device families.</li></ul>
<a href="#">Microchip Graphics Controller</a>	This is the generic display driver is intended for the Microchip Graphics Controller Module implemented in PIC24F device

family. This driver will drive TFT, CSTN and STN displays.

## Links

---

[Display Device Driver Layer API](#), [Topics](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#)

---

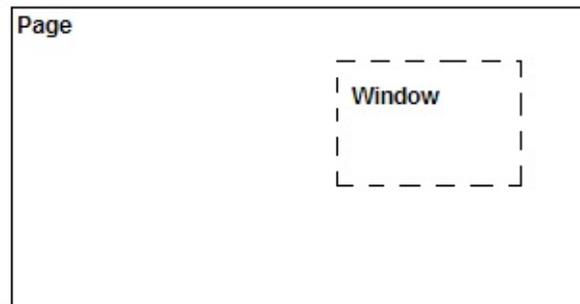
Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Alpha Blending

## Functions | [Advanced Display Driver Features](#)

Alpha Blending in Epson Controller uses blocks of pixels in the memory called Windows. A window can be any rectangular area in a display buffer. The display buffer is also considered as a page. For a QVGA display buffer a page would be 320x240 pixels. A window is a certain width and height contained inside a page.

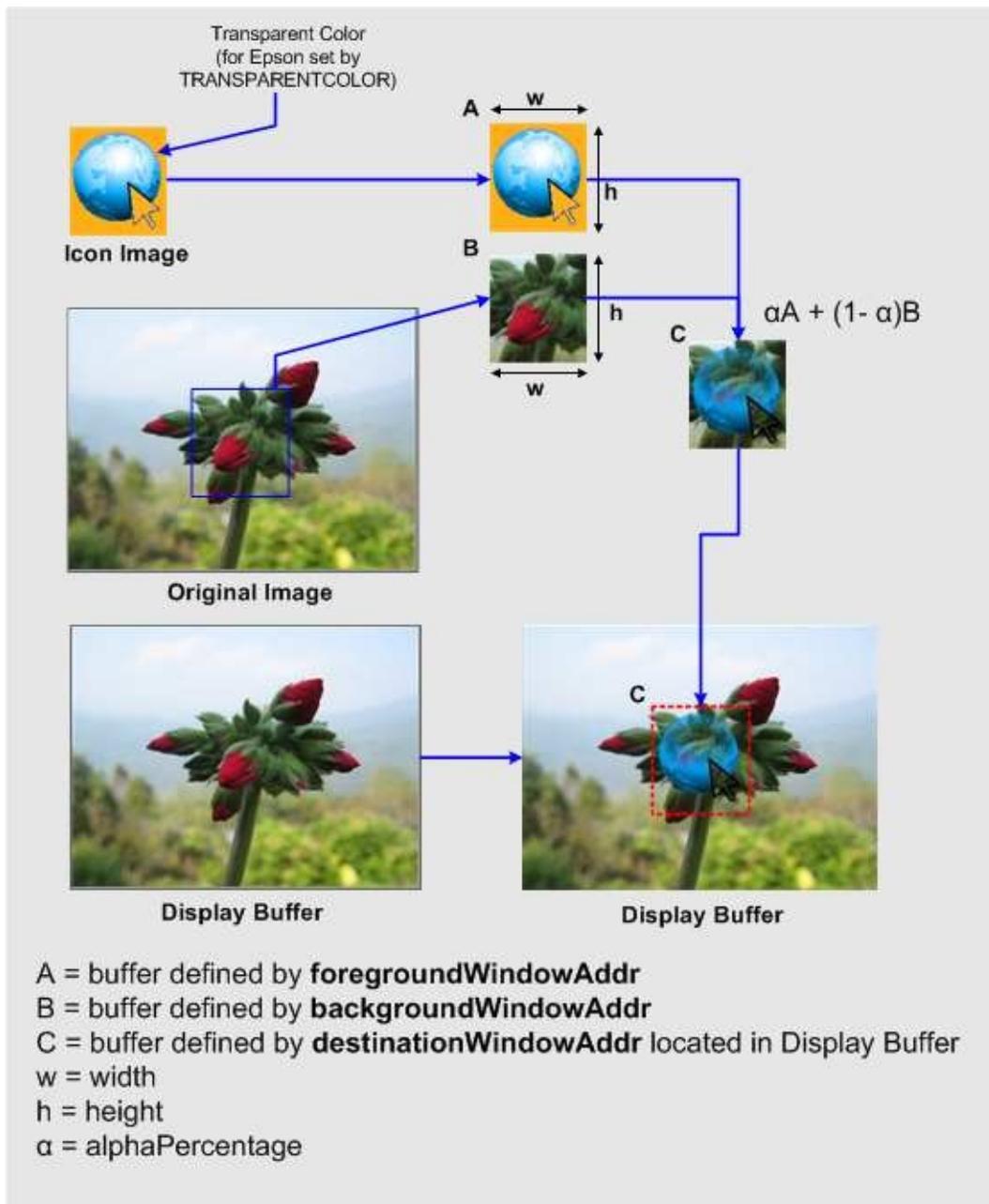


Alpha blending equation:

OutPut Window = Foreground Window \* (Alpha) + Background Window \* (1-Alpha)

when done in software requires a lot of CPU bandwidth. Epson Controller implements alpha blending in hardware which offloads the CPU.

The Display Driver Layer of the Graphics Library utilizes the Epson Controller alpha blending through the [AlphaBlendWindow\(\)](#) function. Three windows are specified with equal widths and heights. An alpha parameter is passed to define the level of alpha blending. The logical flow of the operation is shown below:



An icon image with a given width (w) and height (h) is needed to be alpha blended into the display buffer.

- Buffer A is allocated in memory and its location set by foregroundWindowAddr
- Buffer B is allocated in memory and its location set by

backgroundWindowAddr and

- Buffer C is allocated in memory and its location set by destinationWindowAddr. Note that the destination window can be located within the display buffer. Doing this removes an intermediate step after [AlphaBlendWindow\(\)](#) call to put the result of alpha blend in the display buffer.

The final location of the icon on the Display Buffer is used to locate the affected pixels in the Display Buffer. These affected pixels are copied into Buffer B while Buffer A is filled up with the Icon Image. Once Buffers A and B are ready they are alpha blended to Buffer C. Buffer C is then copied to the Display Buffer.

Note that the Icon Image has a background color of ORANGE. To have the effect of the final output Display Buffer, set the Epson Controllers transparent color to ORANGE. This is set in the TRANSPARENTCOLOR macro defined in the Epson S1D13517 Driver. This TRANSPARENTCOLOR macro is not to be confused with the [TransparentColorEnable\(\)](#) function of the Display Device Driver Level Primitives. TRANSPARENTCOLOR is set at build time. In the future release of the Epson driver, this will be converted to use the [TransparentColorEnable\(\)](#) function.

## Functions

	Name	Description
	<a href="#">GFXGetPageOriginAddress</a>	This function calculates the address of a certain 0,0 location of a page in memory
	<a href="#">GFXGetPageXYAddress</a>	This function calculates the address of a certain x,y location in memory

## Module

---

[Advanced Display Driver Features](#)

## Links

---

[Functions](#), [Legend](#), [Advanced Display Driver Features](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Alpha Blending](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFXGetPageOriginAddress Function

C

```
DWORD GFXGetPageOriginAddress(  
    SHORT pageNumber  
);
```

## Overview

This function calculates the address of a certain 0,0 location of a page in memory

## Input Parameters

Input Parameters	Description
SHORT pageNumber	the page number

## Returns

The address of the start of a certain page in memory

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Alpha Blending](#) > [GFXGetPageOriginAddress Function](#)

# GFXGetPageXYAddress Function

```
C
DWORD GFXGetPageXYAddress(
    SHORT pageNumber,
    WORD x,
    WORD y
);
```

## Overview

This function calculates the address of a certain x,y location in memory

## Input Parameters

Input Parameters	Description
SHORT pageNumber	the page number
WORD x	the x (horizontal) offset from 0,0 of the pagenumber
WORD y	the y (vertical) offset from the 0,0 of the pagenumber

## Returns

The address of an XY position of a certain page in memory

## Preconditions

none

## Side Effects

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Alpha Blending](#) > [GFXGetPageXYAddress Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Transitions

[Enumerations](#) | [Functions](#) | [Advanced Display Driver Features](#)

Applications often require some transition effects while changing screens in order to enhance the look and feel which can be achieved by using Transition APIs provided with the Microchip Graphics Library. Refer to Appendix C of the application note [AN1368: Developing Embedded Graphics Applications using PIC® Microcontrollers with Integrated Graphics Controller](#) for a graphical illustration of how transitions can be achieved.

In order to translate a new screen, the new screen has to be developed in a separate buffer and has to be copied to the frame buffer (visible display buffer) part by part. The way the new screen data is copied into the frame buffer determines the transition effect. A number of transition effects can be achieved by using the API: [GFXTransition](#)(left, top, right, bottom, type, srcpageaddr, destpageaddr, delay\_ms, param1, param2). This API copies the rectangular area defined by left, top, right and bottom from address defined by srcpageaddr to the destpageaddr as per the transition defined by type, param1 and param2. The srcpageaddr should contain the new screen data and the destpageaddr must be the frame buffer address. The speed of the transition can be configured by the parameter delay\_ms which determines the delay between each copy operations. Note that this API must be called only at the end of [GOLDDrawCallback\(\)](#) function in order to ensure that the new screen is fully created in the RAM.

If double buffering is enabled, then using transitions is made easier by another API: [GFXSetupTransition](#)(left, top, right,

bottom, type, delay\_ms, param1, param2). This API can be called immediately after creating the new screen and the graphics library will store the request and initiate the transition after the new screen is fully created in the RAM. Note that this API doesn't have address parameters as the address of draw-buffer is already known to the double buffering subsystem of the graphics engine.

## Enumerations

Name	Description
<a href="#"><u>GFX_TRANSITION_DIRECTION</u></a>	Direction enumeration to determine the direction of the selected <a href="#"><u>GFX_TRANSITION_TYPE</u></a> .
<a href="#"><u>GFX_TRANSITION_TYPE</u></a>	Transition type enumeration to determine the type of the transition. Each type will require specific parameters when setting up the transition operation ( <a href="#"><u>GFXSetupTransition()</u></a> or <a href="#"><u>GFXTransition()</u></a> ).

## Functions

	Name	Description
◆	<a href="#"><u>GFXTransition</u></a>	This immediately executes the transition effect using the <a href="#"><u>GFX_TRANSITION_TYPE</u></a> and the given parameters.
◆	<a href="#"><u>GFXSetupTransition</u></a>	This sets up the transition effect

		<p>using the <a href="#">GFX_TRANSITION_TYPE</a> and the given parameters. The actual transition execution will occur when <a href="#">GFXExecutePendingTransition()</a> is called. When <code>DOUBLE_BUFFERING</code> is enabled, <a href="#">GFXExecutePendingTransition()</a> is executed after the current screen is fully rendered.</p>
	<p><a href="#">GFXExecutePendingTransition</a></p>	<p>This function executes the transition that was set up by <a href="#">GFXSetupTransition()</a>. Status of the transition is returned to indicate if the transition was executed or not. This function is used by the double buffering feature (<a href="#">USE_DOUBLE_BUFFERING</a>) to perform transition operation after the current screen is fully rendered. This function assumes that the source page and destination page are already set up.</p>
	<p><a href="#">GFXIsTransitionPending</a></p>	<p>This function returns the status of a pending transition, set up by <a href="#">GFXSetupTransition()</a>, waiting to be executed.</p>

## Module

### Advanced Display Driver Features

## Links

---

[Enumerations](#), [Functions](#), [Legend](#), [Advanced Display Driver Features](#)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFXTransition Function

```
C
BYTE GFXTransition(
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    GFX_TRANSITION_TYPE type,
    DWORD srcpageaddr,
    DWORD destpageaddr,
    WORD delay_ms,
    WORD param1,
    WORD param2
);
```

## Overview

This immediately executes the transition effect using the [GFX\\_TRANSITION\\_TYPE](#) and the given parameters.

## Input Parameters

Input Parameters	Description
SHORT left	left x coordinate
SHORT top	top y coordinate
SHORT right	right x coordinate
SHORT bottom	bottom y coordinate
GFX_TRANSITION_TYPE type	Transition type

DWORD srcpageaddr	Source page address for the transition
DWORD destpageaddr	Destination page address for the transition
WORD delay_ms	Delay in milli seconds between redraws in the screen while executing the transition
WORD param1	Transition-type specific parameter
WORD param2	Transition-type specific parameter

## Returns

---

Returns status of transition

- 0 : Transition executed successfully
- -1 : Transition not executed

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#) > [GFXTransition Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFXSetupTransition Function

```
C
BYTE GFXSetupTransition(
    SHORT left,
    SHORT top,
    SHORT right,
    SHORT bottom,
    GFX_TRANSITION_TYPE type,
    WORD delay_ms,
    WORD param1,
    WORD param2
);
```

## Overview

This sets up the transition effect using the [GFX\\_TRANSITION\\_TYPE](#) and the given parameters. The actual transition execution will occur when [GFXExecutePendingTransition\(\)](#) is called. When `DOUBLE_BUFFERING` is enabled, [GFXExecutePendingTransition\(\)](#) is executed after the current screen is fully rendered.

## Input Parameters

Input Parameters	Description
SHORT left	left x coordinate
SHORT top	top y coordinate
SHORT right	right x coordinate
SHORT bottom	bottom y coordinate

GFX_TRANSITION_TYPE type	Transition type
WORD delay_ms	Delay in milli seconds between redraws in the screen while executing the transition
WORD param1	Transition-type specific parameter
WORD param2	Transition-type specific parameter

## Returns

---

Returns success of the setup

- 0 : Parameters successfully saved for the new transition
- -1 : Parameters not saved, there is a pending transition

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#) > [GFXSetupTransition Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFXExecutePendingTransition Function

```
C
BYTE GFXExecutePendingTransition(
    DWORD srcpageaddr,
    DWORD destpageaddr
);
```

### Overview

This function executes the transition that was set up by [GFXSetupTransition\(\)](#). Status of the transition is returned to indicate if the transition was executed or not. This function is used by the double buffering feature ([USE\\_DOUBLE\\_BUFFERING](#)) to perform transition operation after the current screen is fully rendered. This function assumes that the source page and destination page are already set up.

### Input Parameters

Input Parameters	Description
DWORD srcpageaddr	Source page address for the transition
DWORD destpageaddr	Destination page address for the transition

### Returns

Returns status of transition

- 0 : Transition executed successfully
- -1 : Transition not executed

## [Features](#) > [Transitions](#) > [GFXExecutePendingTransition Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFXIsTransitionPending Function

```
C
```

```
BYTE GFXIsTransitionPending();
```

## Overview

This function returns the status of a pending transition, set up by [GFXSetupTransition\(\)](#), waiting to be executed.

## Returns

Returns transition status

- 0 : No pending transition
- 1 : There is a pending transition

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#) > [GFXIsTransitionPending Function](#)

## GFX\_TRANSITION\_DIRECTION Enumeration

C

```
typedef enum {  
    LEFT_TO_RIGHT = 0,  
    RIGHT_TO_LEFT,  
    TOP_TO_BOTTOM,  
    BOTTOM_TO_TOP,  
    HORIZONTAL,  
    VERTICAL  
} GFX_TRANSITION_DIRECTION;
```

### Overview

Direction enumeration to determine the direction of the selected [GFX\\_TRANSITION\\_TYPE](#).

### Members

Members	Description
LEFT_TO_RIGHT = 0	option used in SLIDE, PUSH transition type ( <a href="#">GFX_TRANSITION_TYPE</a> )
RIGHT_TO_LEFT	option used in SLIDE, PUSH transition type ( <a href="#">GFX_TRANSITION_TYPE</a> )
TOP_TO_BOTTOM	option used in SLIDE, PUSH transition type ( <a href="#">GFX_TRANSITION_TYPE</a> )
BOTTOM_TO_TOP	option used in SLIDE, PUSH transition type ( <a href="#">GFX_TRANSITION_TYPE</a> )
HORIZONTAL	option used in EXPANDING_LINE and CONTRACTING_LINE transition type ( <a href="#">GFX_TRANSITION_TYPE</a> )

VERTICAL

option used in EXPANDING\_LINE and CONTRACTING\_LINE transition type (GFX\_TRANSITION\_TYPE)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#) > [GFX\\_TRANSITION\\_DIRECTION Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_TRANSITION\_TYPE Enumeration

C

```
typedef enum {  
    PLAIN = 0,  
    SLIDE,  
    PUSH,  
    EXPANDING_RECTANGLE,  
    CONTRACTING_RECTANGLE,  
    EXPANDING_LINE,  
    CONTRACTING_LINE  
} GFX_TRANSITION_TYPE;
```

### Overview

Transition type enumeration to determine the type of the transition. Each type will require specific parameters when setting up the transition operation ([GFXSetupTransition\(\)](#) or [GFXTransition\(\)](#)).

GFX_TRANSITION_TYPE	param1	param2 (sets the direction of transition)
PLAIN	pixel block size	not used
EXPANDING_RECTANGLE	pixel block size	not used
CONTRACTING_RECTANGLE	pixel block	not used

	size	
SLIDE	pixel block size	LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP
PUSH	pixel block size	LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP
EXPANDING_LINE	pixel block size	HORIZONTAL, VERTICAL
CONTRACTING_LINE	pixel block size	HORIZONTAL, VERTICAL

## Members

Members	Description
PLAIN = 0	no transition effect
SLIDE	param1 -> Pixel-block size, param2 - LEFT_TO_RIGHT/RIGHT_TO_LEFT
PUSH	param1 -> Pixel-block size, param2 - LEFT_TO_RIGHT/RIGHT_TO_LEFT
EXPANDING_RECTANGLE	param1 -> Pixel-block size
CONTRACTING_RECTANGLE	param1 -> Pixel-block size
EXPANDING_LINE	param1 -> Pixel-block size, param2 -
CONTRACTING_LINE	param1 -> Pixel-block size, param2 -

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#) > [GFX\\_TRANSITION\\_TYPE Enumeration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Double Buffering

## Functions | [Advanced Display Driver Features](#)

Manipulating pixels on the screen requires direct writes to the frame buffer. While these changes are being executed, the screen is also refreshed. This means that the changes are displayed immediately as the frame buffer is being updated. This is not a suitable scheme when the changes are slow, for example, decoding an image or having a large number of widgets on a screen. The display may appear slow in that case. One solution to this problem is to use a double-buffering scheme supported by the Microchip Graphics Library. In this scheme, the changes are not directly written to the frame buffer, but instead, they are written to a separate buffer, called the 'Draw Buffer'. After all the changes are made, the draw buffer and frame buffer are exchanged. Now the draw buffer becomes the frame buffer, and because of all the changes drawn there, the changes appear spontaneous to the user. Of course, there will be a delay, as all the changes have to be written to the draw buffer before displaying it. This delay is generally more tolerable than displaying the changes slowly. After exchanging of the buffers, the latest buffer (which is now the frame buffer) is copied to the new draw buffer in the background and then the new draw buffer is in sync with what is being displayed. New changes are then written to the draw buffer and the cycle continues. As the double-buffering scheme uses two buffers, the RAM requirement will double.

In the Microchip Graphics Library, if double-buffering is enabled, the frame buffer and draw buffer are exchanged after the changes of all the widgets on a screen are done (i.e., the new screen appears after the whole screen is updated and not after

updating an individual widget).

The work flow of double-buffering is graphically explained along with tips on deciding when to use double buffering in the APPENDIX B of the Application note AN1368: Developing Embedded Graphics Applications using PIC® Microcontrollers with Integrated Graphics Controller.

To use double buffering in an application, follow the steps described below:

1. Enable the option [USE\\_DOUBLE\\_BUFFERING](#) in GraphicsConfig.h
2. Update [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) to include both the buffers in HardwareProfile.h. Note that when using external memory the [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) must fit into the external memory size. For example in PIC24FJ256DA210 external memory can be added using EPMP. External memory mapped to the EPMP must be big enough to accommodate the display buffers required by double buffering. See [Set Up Display Interface](#) for more information on memory requirements.
3. Check the jumper settings to enable the required RAM address space on the development board

If Graphics Objects Layer (GOL) is used in the application, the switching of buffers is handled automatically in order to keep the switching task transparent to the users. If double buffering is enabled in applications using only the Primitive layer, then the

switching of buffers has to be handled by the application itself as explained in the following steps.

Steps required for manually handling the switching of buffers:

1. After [InitGraph\(\)](#) is called, call the APIs [InvalidateAll\(\)](#) followed by [UpdateDisplayNow\(\)](#). The two buffers are properly setup after these calls and from this point onwards, the drawing happens on the draw-buffer.
2. When a shape is drawn on the draw buffer, that rectangular area has to be marked as invalid by using the API [InvalidateRectangle](#)(left, top, right, bottom). Only the invalidated rectangular areas are copied to the frame buffer in order to reduce the copy operations thereby reducing the overall time and energy required to sync the two buffers. Hence, it is important to invalidate the changed rectangular areas failing which the change doesn't show up on the display.
3. Call either [RequestDisplayUpdate\(\)](#) or [UpdateDisplayNow\(\)](#) to sync the two buffers and as a result the changes appear on the display. The former API exchanges the buffers during the next display blanking period on TFT LCDs causing the change to appear smooth and immediate whereas the latter API exchanges the two buffers at the time of the API call probably causing a slight flicker on the display. On displays which doesn't support blanking periods (e.g. CSTN LCDs), the effect of [RequestDisplayUpdate\(\)](#) is same as that of [UpdateDisplayNow\(\)](#).

Even if double buffering is enabled at compile time, it can be

switched off and on at run time using APIs [SwitchOffDoubleBuffering\(\)](#) and [SwitchOnDoubleBuffering\(\)](#). Switching double buffering on/off at runtime is useful in applications which need some screens having fast updates like waveform or animation which requires double buffering to be switched off and some other screens where double buffering is beneficial.

Note: In applications using Graphics Objects Layer and double buffering, the double buffering is not immediately enabled after the API [GOLInit\(\)](#) is called in order to support hassles splash screens but is automatically enabled from the second screen update onwards. If double buffering is needed from the first screen itself, then follow step 1 immediately after calling [GOLInit\(\)](#).

## Functions

	Name	Description
◆	<a href="#">SwitchOffDoubleBuffering</a>	Switches off the double buffering. All rendering will be performed on the frame buffer. Calls to <a href="#">UpdateDisplayNow()</a> or <a href="#">RequestDisplayUpdate()</a> will have no effect.
◆	<a href="#">SwitchOnDoubleBuffering</a>	Switches on the double buffering. Double buffering utilizes two buffers. The frame buffer and the draw buffer. The frame buffer is the buffer that is being displayed while the draw buffer is used for all rendering. When this function is called, it copies the contents of the frame

		buffer to the draw buffer once and all succeeding rendering will be performed on the draw buffer. To update the frame buffer with newly drawn items on the draw buffer call <a href="#">UpdateDisplayNow()</a> or <a href="#">RequestDisplayUpdate()</a> .
	<a href="#">InvalidateRectangle</a>	Invalidates the specified rectangular area. This increments the number of invalidated areas and if the number of invalidated areas exceed the GFX_MAX_INVALIDATE_AREAS, the whole frame buffer is invalidated.
	<a href="#">RequestDisplayUpdate</a>	Synchronizes the draw and frame buffers at next VBlank
	<a href="#">UpdateDisplayNow</a>	Synchronizes the draw and frame buffers immediately.

## Module

### [Advanced Display Driver Features](#)

## Links

### [Functions](#), [Legend](#), [Advanced Display Driver Features](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#)

# SwitchOffDoubleBuffering Function

**C**

```
void SwitchOffDoubleBuffering();
```

## Overview

Switches off the double buffering. All rendering will be performed on the frame buffer. Calls to [UpdateDisplayNow\(\)](#) or [RequestDisplayUpdate\(\)](#) will have no effect.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#) > [SwitchOffDoubleBuffering Function](#)

# SwitchOnDoubleBuffering Function

**C**

```
void SwitchOnDoubleBuffering();
```

## Overview

Switches on the double buffering. Double buffering utilizes two buffers. The frame buffer and the draw buffer. The frame buffer is the buffer that is being displayed while the draw buffer is used for all rendering. When this function is called, it copies the contents of the frame buffer to the draw buffer once and all succeeding rendering will be performed on the draw buffer. To update the frame buffer with newly drawn items on the draw buffer call [UpdateDisplayNow\(\)](#) or [RequestDisplayUpdate\(\)](#).

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#) > [SwitchOnDoubleBuffering Function](#)

# InvalidateRectangle Function

C

```
void InvalidateRectangle(  
    WORD left,  
    WORD top,  
    WORD right,  
    WORD bottom  
);
```

## Overview

Invalidates the specified rectangular area. This increments the number of invalidated areas and if the number of invalidated areas exceed the `GFX_MAX_INVALIDATE_AREAS`, the whole frame buffer is invalidated.

## Input Parameters

Input Parameters	Description
WORD left	left position
WORD top	top position
WORD right	right position
WORD bottom	bottom position

## Returns

None

## Preconditions

None

## Side Effects

---

Copies back the invalidated areas only to the draw buffer after the exchange of draw and frame buffers.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#) > [InvalidateRectangle Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RequestDisplayUpdate Function

```
C
```

```
void RequestDisplayUpdate();
```

## Overview

Synchronizes the draw and frame buffers at next VBlank

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#) > [RequestDisplayUpdate Function](#)

# UpdateDisplayNow Function

```
C
```

```
void UpdateDisplayNow();
```

## Overview

Synchronizes the draw and frame buffers immediately.

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#) > [UpdateDisplayNow Function](#)

# Microchip Graphics Controller

## Advanced Display Driver Features | Topics

The Microchip Graphics Controller has several advanced features that can be enabled by declaring macros in the Graphics Config (GraphicsConfig.h) and Hardware Profile (HardwareProfile.h) of the project. Below is a summary of all macros that pertains to the Microchip Graphics Controller driver.

Macro Name	Description
<a href="#">GFX_DISPLAY_BUFFER_START_ADDRESS</a>	Defines the start address of the display buffer.
<a href="#">GFX_DISPLAY_BUFFER_LENGTH</a>	Defines the size of the display buffer used to map the display. It is used to map the display into the internal memory. <ol style="list-style-type: none"><li>1. Double the value of the macro <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a>.</li><li>2. IPU module is enabled by the macro <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a>.</li></ol> If the application is running in external memory, and the IPU module is not enabled, the display is mapped into the internal memory. <b>NOTE:</b> If the application is running in external memory (i.e. in <a href="#">GFX_DISPLAY_BUFFER_START_ADDRESS</a> ), the value of <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> must be greater than or equal to: <ul style="list-style-type: none"><li>• <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> * <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> * chip select</li><li>• <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> * <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> * <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a></li></ul>

	chip select
<u>USE_PALETTE</u>	Macro that enables the display library and the display driver.
<u>USE_DOUBLE_BUFFERING</u>	<p>Optional feature. Memory is double in size and these buffers are used for library draw operations and the swapping.</p> <p><b>Buffer Location:</b>  Display Buffer: <u>GFX_DISPLAY_BUFFER</u>  Display Buffer: <u>GFX_DISPLAY_BUFFER</u>  Display Buffer: <u>GFX_DISPLAY_BUFFER</u></p> <p><b>NOTE:</b> If the chip select memory (i.e. in the display buffer) is equal to the chip select:</p> <ul style="list-style-type: none"> <li>• (<u>GFX_DISPLAY_BUFFER</u> chip select)</li> <li>• (<u>GFX_DISPLAY_BUFFER</u> chip select)</li> </ul>
<u>USE_COMP_IPU</u>	<p>Optional feature module in the IPU. Memory requirement of the compression module.</p> <p><i>When compression program memory is used...</i></p>

GFX\_COMPF  
memory cache  
GFX\_DECOM  
the memory c  
data  
GFX\_IPU\_TE  
- defines the r  
transferring th  
memory (not r  
to RAM locati  
initialization a  
size cannot n

**Buffer Locati  
enabled):**

Display Buffer  
GFX\_DISPLA  
Decomprese  
GFX\_DISPLA  
(GFX\_DISPLA  
Compressed I  
Buffer + GFX\_

**Buffer Locati  
disabled):**

Display Buffer  
GFX\_DISPLA  
Decomprese  
GFX\_DISPLA  
GFX\_DISPLA  
Compressed I  
Buffer + GFX\_

*When compre  
(RAM or exter  
GFX\_DECOM  
the memory c*

data

**Buffer Location (enabled):**

Display Buffer  
GFX\_DISPLAY\_BUFFER  
Decompress  
GFX\_DISPLAY\_BUFFER  
(GFX\_DISPLAY\_BUFFER)

**Buffer Location (disabled):**

Display Buffer  
GFX\_DISPLAY\_BUFFER  
Decompress  
GFX\_DISPLAY\_BUFFER  
GFX\_DISPLAY\_BUFFER

**NOTE:** If the compressed data is not in memory (i.e. in a display, compressed data must be loaded into a region of the correct size)

**External CS I**

- **Double End**

- (GFX\_DISPLAY\_BUFFER)  
GFX\_DISPLAY\_BUFFER  
GFX\_DISPLAY\_BUFFER  
<= GFX\_DISPLAY\_BUFFER  
location
- (GFX\_DISPLAY\_BUFFER)  
GFX\_DISPLAY\_BUFFER  
GFX\_DISPLAY\_BUFFER  
<= GFX\_DISPLAY\_BUFFER  
location

- **Double End**

- ((GFX\_DISPLAY\_BUFFER)

	<p>GFX_</p> <p>GFX_</p> <p>&lt;= G</p> <p>locat</p> <ul style="list-style-type: none"> <li>o ((GFE</li> </ul> <p>GFX_</p> <p>GFX_</p> <p>&lt;= G</p> <p>locat</p>
<u>GFX_EPMP_CS1_BASE_ADDRESS</u>	<p>For PIC Device Master Port (E Family of Dev Defines the lo address. Whe automatically <u>GFX_EPMP_</u> determines ho will be enable Use this macr <u>GFX_EPMP_</u> memory to EF Microchip Gra</p>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<p>Defines the si Chip Select 1. <u>GFX_EPMP_</u></p> <p><b>NOTE:</b> Refer <u>GFX_DISPLAY_USE_DOUBLE</u> description for <u>GFX_EPMP_</u> memory requi used for displ</p>
<u>GFX_EPMP_CS2_BASE_ADDRESS</u>	<p>For PIC Device</p>

	Master Port (E Family of Dev Defines the lo address.
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<p>Defines the si Chip Select 2. <u>GFX_EPMP_</u></p> <p><b>NOTE:</b> Refer <u>GFX_DISPLA USE_DOUBL</u> description for <u>GFX_EPMP_</u> memory requi used for displ</p>
GFX_COMPRESSED_BUFFER_SIZE	Macro used w memory cach The value cho compressed c images/data a
GFX_DECOMPRESSED_BUFFER_SIZE	Macro used w memory cach The value cho decompresser images/data a
GFX_IPU_TEMP_DATA_TRANSFER_ARRAY_SIZE	<p>Macro used w memory cach compressed c mapped to EC location.</p> <p>The size cho Allowed size i variables and</p>

## Enabling Internal/External Memory

The Microchip Graphics Controller can use internal memory or external memory for the display buffers. The driver decides on the location of the buffers based on the [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) macro. For the PIC24FJ256DA210 device family, there are two variants for internal memory sizes, 96 Kbytes and 24 Kbytes. For the 96 Kbytes variants, the external memory can be immediately mapped after the internal RAM address. For the 24KBytes variants, the external memory cannot be mapped after the internal RAM address. Refer to the PIC24FJ256DA210 Family Data Sheet for details on the mapping of external memory (Section 4.2 Data Memory Space).

External Memory Example:

[Copy Code](#)

```
// Settings for Hardware Profile to use External Me  
// on PIC24FJ256DA210 Development Board  
  
// Microchip Development Board Specific macros  
#define PIC24FJ256DA210_DEV_BOARD  
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118  
  
// Microchip Graphics Controller specific macros  
// PMP port data bus width (for Microchip Graphic  
#define USE_16BIT_PMP  
// Use Microchip Display Controller  
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210  
// Microchip Graphics Controller pixel clock divi  
// The value used is derived from display's refre  
#define GFX_GCLK_DIVIDER 61  
  
// Display Buffer start address  
// Note that the value is mapped to external memo
```

```

// see PIC24FJ256DA210 Family Data Sheet for det
#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x000200
// Display Buffer size
#define GFX_DISPLAY_BUFFER_LENGTH 0x00025800u1

// Since Display Buffer is mapped to external mem
// EPMP needs to be enabled
#define GFX_EPMP_CS1_BASE_ADDRESS 0x00020000u1
#define GFX_EPMP_CS1_MEMORY_SIZE 0x80000u1

// EPMP CS2 can also be used as an external memor
#define GFX_EPMP_CS2_BASE_ADDRESS (0x00020000u1 +
#define GFX_EPMP_CS2_MEMORY_SIZE 0x80000u1

```

Internal Memory Example:

[Copy Code](#)

```

// Settings for Hardware Profile to use Internal Me
// on PIC24FJ256DA210 Development Board

// Hardware Profile
// Microchip Development Board Specific macros
#define PIC24FJ256DA210_DEV_BOARD
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118

// Microchip Graphics Controller specific macros
// Use Microchip Display Controller
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210
// Microchip Graphics Controller pixel clock divi
// The value used is derived from display's refre
#define GFX_GCLK_DIVIDER 61

// Display Buffer start address
// Note that the value is mapped to external memo
// see PIC24FJ256DA210 Family Data Sheet for det
#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00004B
// Display Buffer size

```

```
#define GFX_DISPLAY_BUFFER_LENGTH 0x0004B000u1
```

## Enabling Double Buffering

See [Double Buffering](#).

## Enabling Color Look Up Table

The Graphics Controller Module can be set to operate on 1, 2, 4, 8 and 16 bpp. Color depths 1, 2, 4, and 8 can be used with the 256 entries Color Look Up Table (CLUT). When using external memory, only the 8bpp mode can be used with the CLUT enabled. When using internal memory, 1,2,4 and 8 bpp can be used.

Enabling Color Look Up Table Example:

[Copy Code](#)

```
// Settings for GraphicsConfig.h to use the CLUT at  
  
// enable the palette mode  
#define USE_PALETTE  
  
// set the color depth  
#define COLOR_DEPTH 8
```

## Enabling the IPU Module

Currently, the IPU module is an option to reduce memory external or internal flash memory when storing images for the Graphics Library. The Graphics Resource Converter tool provides an option to convert images to be saved in compressed format for IPU decompression.

## Enabling IPU Decompression Example:

[Copy Code](#)

```
// Settings for GraphicsConfig.h to use the IPU Mod  
  
// enable the use of IPU module  
#define USE_COMP_IPU
```

[Copy Code](#)

```
// Settings for Hardware Profile to use IPU Module  
// on PIC24FJ256DA210 Development Board  
// Use this when compressed data is from memory not  
  
// define the size of the compressed buffer  
#define GFX_COMPRESSED_BUFFER_SIZE      (  
  
// define the size of the decompressed buffer  
#define GFX_DECOMPRESSED_BUFFER_SIZE    (  
  
// define the size of the temporary buffer to tra  
// data from memory not mapped in EDS  
// (example: External SPI memory, Program Flash)  
#define GFX_IPU_TEMP_DATA_TRANSFER_ARRAY_SIZE  (  
  

```

[Copy Code](#)

```
// Settings for Hardware Profile to use IPU Module  
// on PIC24FJ256DA210 Development Board  
// Use this when compressed data is from memory map  
  
// define the size of the decompressed buffer  
#define GFX_DECOMPRESSED_BUFFER_SIZE      (  
  

```

## Module

## Advanced Display Driver Features

### Topics

Name	Description
<a href="#"><u>Rectangle Copy Operations</u></a>	The following APIs are used move blocks of data from one memory location to another.
<a href="#"><u>Decompressing DEFLATED data</u></a>	The Microchip Graphics Controller features a decompression module for data compressed using the DEFLATE algorithm. Compressed data are limited to fixed huffman codes. Compressed data with dynamic huffman codes are not supported.
<a href="#"><u>Palette Mode</u></a>	The Microchip Graphics Controller features a palette mode for a smaller frame buffer requirement. This option uses the built-in 256 entry Color Look-up Table (CLUT) to represent pixels from the display buffer in memory. If the CLUT is enabled, each pixel in the display buffer is assumed to contain the color index. This color index is used as the address of the CLUT entry that contains the color value that will be used for the given pixel.
<a href="#"><u>Set Up Display Interface</u></a>	
<a href="#"><u>External or Internal Memory and Palettes</u></a>	This section shows examples on how to set up applications using external memory, internal memory or use palettes for Microchip Graphics Module.

### Links

## Advanced Display Driver Features, Topics

---

Library API > Display Device Driver Layer API > Advanced Display Driver Features > Microchip Graphics Controller

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Rectangle Copy Operations

Functions | Macros

## Functions

	Name	Description
	<a href="#"><u>ROPBlock</u></a>	Performs a Raster Operation (ROP) on source and destination. The type of ROP is decided by the rop and the copyOp parameter.
	<a href="#"><u>Scroll</u></a>	Scrolls the rectangular area defined by left, top, right, bottom by delta pixels.

## Macros

Name	Description
<a href="#"><u>RCC_SRC_ADDR_CONTINUOUS</u></a>	<p>Source (S) and Destination (D) address macros for the Rectangle Copy Process (RCCGPU). When performing executing the <a href="#"><u>Rectangle</u></a> Copy Process (RCCGPU). The source and destination data can be treated as a copy of data in memory or a display buffer in memory. This gives flexibility in operation where a copy operation is performed to data already present in a display buffer or anywhere in memory. Both source and destination address can be set to continuous or copy of data. These macros are only for RCCGPU operations.</p> <ul style="list-style-type: none"> <li>• <a href="#"><u>RCC_SRC_ADDR_COPY</u></a> source data is continuous</li> </ul>

	<ul style="list-style-type: none"> <li>• <a href="#"><u>RCC_SRC_ADDR_DIS</u></a> - source data is disconti</li> </ul>
<p><a href="#"><u>RCC_SRC_ADDR_DISCONTINUOUS</u></a></p>	<p>Source (S) and Destination (D) (When performing executing the <a href="#"><u>Rectangle</u></a> Copy Process (RCCGPU). The source and data can be treated as a copy of data in memory or a discontinuous in memory. This gives flexibility in operation where an copy operation performed to data already present in display buffer or anywhere else in memory. Both source and destination can be set to continuous or discontinuous data. These macros are only for RCCGPU operations.</p> <ul style="list-style-type: none"> <li>• <a href="#"><u>RCC_SRC_ADDR_CONT</u></a> source data is continuous</li> <li>• <a href="#"><u>RCC_SRC_ADDR_DIS</u></a> - source data is disconti</li> </ul>
<p><a href="#"><u>RCC_DEST_ADDR_CONTINUOUS</u></a></p>	<p>Source (S) and Destination (D) (When performing executing the <a href="#"><u>Rectangle</u></a> Copy Process (RCCGPU). The source and data can be treated as a copy of data in memory or a discontinuous in memory. This gives flexibility in operation where an copy operation performed to data already present in display buffer or anywhere else in memory. Both source and destination can be set to continuous or discontinuous data. These macros are only for RCCGPU operations.</p> <ul style="list-style-type: none"> <li>• <a href="#"><u>RCC_SRC_ADDR_CONT</u></a></li> </ul>

	<ul style="list-style-type: none"> <li>source data is continuous</li> <li>• <u>RCC_SRC_ADDR_DIS</u> - source data is discontinuous</li> </ul>
<u>RCC_DEST_ADDR_DISCONTINUOUS</u>	<p>Source (S) and Destination (D) data can be set to continuous or discontinuous when performing executing the <u>Rectangle</u> Copy Process (RCCGPU). The source and destination data can be treated as a continuous or discontinuous block of data in memory or a discontinuous block in memory. This gives flexibility in operation where an copy operation is performed to data already present in display buffer or anywhere else in memory. Both source and destination data can be set to continuous or discontinuous. These macros are only for RCCGPU operations.</p> <ul style="list-style-type: none"> <li>• <u>RCC_SRC_ADDR_CONT</u> - source data is continuous</li> <li>• <u>RCC_SRC_ADDR_DIS</u> - source data is discontinuous</li> </ul>
<u>RCC_ROP_0</u>	<p>Raster Operation (ROP) options are defined for the following 16 raster operations whenever <u>Rectangle</u> Copy Function (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).</p> <ul style="list-style-type: none"> <li>• <u>RCC_ROP_0</u> - 0 (<u>BLACK</u>)</li> <li>• <u>RCC_ROP_1</u> - not (S) or (D)</li> <li>• <u>RCC_ROP_2</u> - (not S) and (not D)</li> <li>• <u>RCC_ROP_3</u> - not (S)</li> <li>• <u>RCC_ROP_4</u> - S and not (D)</li> <li>• <u>RCC_ROP_5</u> - not (D)</li> <li>• <u>RCC_ROP_6</u> - S xor D</li> </ul>

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_7</a> - not (S and D)</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xor D)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_ROP_1</a></p>	<p>Raster Operation (ROP) options are defined for each of the following 16 raster operations whenever <a href="#">Rectangle Copy Function</a> (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLACK</a>)</li> <li>• <a href="#">RCC_ROP_1</a> - not (S or D)</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) and D</li> <li>• <a href="#">RCC_ROP_3</a> - not (S) and D</li> <li>• <a href="#">RCC_ROP_4</a> - S and not D</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S and D)</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xor D)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_ROP_2</a></p>	<p>Raster Operation (ROP) options are defined for each of the following 16 raster operations whenever <a href="#">Rectangle Copy Function</a> (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLACK</a>)</li> <li>• <a href="#">RCC_ROP_1</a> - not (S or D)</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) and D</li> <li>• <a href="#">RCC_ROP_3</a> - not (S) and D</li> </ul>

--

- [RCC\\_ROP\\_4](#) - S and n
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

<a href="#">RCC_ROP_3</a>
---------------------------

Raster Operation (ROP) opt of the following 16 raster opt whenever [Rectangle](#) Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#)
- [RCC\\_ROP\\_1](#) - not (S o
- [RCC\\_ROP\\_2](#) - (not S) a
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and n
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

<a href="#">RCC_ROP_4</a>
---------------------------

Raster Operation (ROP) opt of the following 16 raster opt whenever [Rectangle](#) Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#)

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_5</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster opt whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_6</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster opt whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras</p>
----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------

performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#))
- [RCC\\_ROP\\_1](#) - not (S or D)
- [RCC\\_ROP\\_2](#) - (not S) and (not D)
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and not D
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S and D)
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xor D)
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B...](#) [more](#)

## [RCC\\_ROP\\_7](#)

Raster Operation (ROP) options are available for the following 16 raster operations whenever [Rectangle Copy Function \(RCCGPU\)](#) is used. The raster operation is performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#))
- [RCC\\_ROP\\_1](#) - not (S or D)
- [RCC\\_ROP\\_2](#) - (not S) and (not D)
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and not D
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S and D)
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xor D)
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B...](#) [more](#)

## RCC\_ROP\_8

Raster Operation (ROP) options are defined for the following 16 raster operations whenever **Rectangle Copy Function** (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 (**BLACK**)
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S) and D
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D
- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D
- RCC\_ROP\_9 - not (S xor D)
- RCC\_ROP\_A - D
- RCC\_ROP\_B... [more](#)

## RCC\_ROP\_9

Raster Operation (ROP) options are defined for the following 16 raster operations whenever **Rectangle Copy Function** (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 (**BLACK**)
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S) and D
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D
- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_9</a> - not (S x</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
<a href="#">RCC_ROP_A</a>	<p>Raster Operation (ROP) opt of the following 16 raster op whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S x)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
<a href="#">RCC_ROP_B</a>	<p>Raster Operation (ROP) opt of the following 16 raster op whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> </ul>

--

- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

<a href="#">RCC_ROP_C</a>
---------------------------

Raster Operation (ROP) opt of the following 16 raster opt whenever [Rectangle](#) Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#)
- [RCC\\_ROP\\_1](#) - not (S o
- [RCC\\_ROP\\_2](#) - (not S) a
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and n
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

<a href="#">RCC_ROP_D</a>
---------------------------

Raster Operation (ROP) opt of the following 16 raster opt whenever [Rectangle](#) Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#)

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_E</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster opt whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_F</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster opt whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras</p>
----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------

	<p>performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a>)</li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_COPY</a></p>	<p>Type of <a href="#">Rectangle</a> Copy Op one of the following rectangl operations and together with source, destination, current ( transparency are evaluated ( and the result written to the (</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_COPY</a> - Copies th to the destination addre selected ROP.</li> <li>• <a href="#">RCC_SOLID_FILL</a> - Fill rectangle with the curre</li> <li>• <a href="#">RCC_TRANSPARENT_</a> Operation is the same a operation except that th is compared against the set. If the values match, data is not written to the The source... <a href="#">more</a></li> </ul>
<p><a href="#">RCC_SOLID_FILL</a></p>	<p>Type of <a href="#">Rectangle</a> Copy Op</p>

one of the following rectangle operations and together with source, destination, current color and transparency are evaluated and the result written to the destination.

- `RCC_COPY` - Copies the data from the source to the destination address selected ROP.
- `RCC_SOLID_FILL` - Fill the rectangle with the current color.
- `RCC_TRANSPARENT_COPY` - Operation is the same as `RCC_COPY` except that the source color is compared against the destination color set. If the values match, data is not written to the destination. The source... [more](#)

### `RCC_TRANSPARENT_COPY`

Type of `Rectangle Copy` Operation  
one of the following rectangle operations and together with source, destination, current color and transparency are evaluated and the result written to the destination.

- `RCC_COPY` - Copies the data from the source to the destination address selected ROP.
- `RCC_SOLID_FILL` - Fill the rectangle with the current color.
- `RCC_TRANSPARENT_COPY` - Operation is the same as `RCC_COPY` except that the source color is compared against the destination color set. If the values match, data is not written to the destination. The source... [more](#)

---

## Links

---

[Functions](#), [Microchip Graphics Controller](#), [Legend](#), [Macros](#)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ROPBlock Function

C

```
WORD ROPBlock(  
    DWORD srcAddr,  
    DWORD dstAddr,  
    DWORD srcOffset,  
    DWORD dstOffset,  
    WORD srcType,  
    WORD dstType,  
    WORD copyOp,  
    WORD rop,  
    WORD color,  
    WORD width,  
    WORD height  
);
```

## Overview

Performs a Raster Operation (ROP) on source and destination. The type of ROP is decided by the rop and the copyOp parameter.

## Input Parameters

Input Parameters	Description
DWORD srcAddr	the base address of the data to be moved
DWORD dstAddr	the base address of the new location of the moved data
DWORD srcOffset	offset of the data to be moved with respect to the source base address.
DWORD dstOffset	offset of the new location of the moved data

	respect to the source base address.
WORD srcType	sets the source type (GFX_DATA_CONTINUOUS or GFX_DATA_DISCONTINUOUS)
WORD dstType	sets the destination type (GFX_DATA_CONTINUOUS or GFX_DATA_DISCONTINUOUS)
WORD copyOp	sets the type of copy operation <ul style="list-style-type: none"> <li>• <u>RCC_SOLID_FILL</u>: Solid fill of the set color</li> <li>• <u>RCC_COPY</u>: direct copy of source to destination</li> <li>• <u>RCC_TRANSPARENT_COPY</u>: copy with transparency. Transparency color is set by color</li> </ul>
WORD rop	sets the raster operation equation <ul style="list-style-type: none"> <li>• <u>RCC_ROP_0</u>: Solid black color fill</li> <li>• <u>RCC_ROP_1</u>: not (Source or Destination)</li> <li>• <u>RCC_ROP_2</u>: (not Source) and Destination</li> <li>• <u>RCC_ROP_3</u>: not Source</li> <li>• <u>RCC_ROP_4</u>: Source and (not Destination)</li> <li>• <u>RCC_ROP_5</u>: not Destination</li> <li>• <u>RCC_ROP_6</u>: Source xor Destination</li> <li>• <u>RCC_ROP_7</u>: not (Source and Destination)</li> <li>• <u>RCC_ROP_8</u>: Source and Destination</li> <li>• <u>RCC_ROP_9</u>: not (Source xor Destination)</li> </ul>

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_A</a>: Destination</li> <li>• <a href="#">RCC_ROP_B</a>: (not Source) or Destination</li> <li>• <a href="#">RCC_ROP_C</a>: Source</li> <li>• <a href="#">RCC_ROP_D</a>: Source or (not Destination)</li> <li>• <a href="#">RCC_ROP_E</a>: Source or Destination</li> <li>• <a href="#">RCC_ROP_F</a>: Solid white color fill</li> </ul>
WORD color	color value used when transparency operation is set or using solid color fill
WORD width	width of the block of data to be moved
WORD height	height of the block of data to be moved

## Returns

For NON-Blocking configuration:

- Returns 0 when device is busy and operation is not completely performed.
- Returns 1 when the operation is completely performed.

For Blocking configuration:

- Always return 1.

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver](#)

[Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#)  
> [ROPBlock Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Scroll Function

```
C  
WORD Scroll(  
    SHORT left,  
    SHORT top,  
    SHORT right,  
    SHORT bottom,  
    SHORT delta,  
    WORD dir  
);
```

## Overview

Scrolls the rectangular area defined by left, top, right, bottom by delta pixels.

## Input Parameters

Input Parameters	Description
SHORT left	left position of the scrolled rectangle
SHORT top	top position of the scrolled rectangle
SHORT right	right position of the scrolled rectangle
SHORT bottom	bottom position of the scrolled rectangle
SHORT delta	defines the scroll delta
WORD dir	defines the direction of the scroll. <ul style="list-style-type: none"><li>• 0 : scroll to the left</li><li>• 1 : scroll to the right</li></ul>

## Returns

---

none

## Preconditions

---

none

## Side Effects

---

none

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#) > [Scroll Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RCC\_SRC\_ADDR\_CONTINUOUS Macro

C

```
#define RCC_SRC_ADDR_CONTINUOUS 0x00000002u1
#define RCC_SRC_ADDR_DISCONTINUOUS 0x00000000u1
#define RCC_DEST_ADDR_CONTINUOUS 0x00000004u1
#define RCC_DEST_ADDR_DISCONTINUOUS 0x00000000u1
```

### Overview

Source (S) and Destination (D) data type. When performing executing commands on the [Rectangle](#) Copy Processing Unit (RCCGPU). The source and destination data can be treated as a continuous block of data in memory or a discontinuous data in memory. This gives flexibility to the operation where an copy operation can be performed to data already present in the display buffer or anywhere else in data memory. Both source and destination data can be set to continuous or discontinuous data. These macros are only used in RCCGPU operations.

- RCC\_SRC\_ADDR\_CONTINUOUS - source data is continuous
- RCC\_SRC\_ADDR\_DISCONTINUOUS - source data is discontinuous
- RCC\_DEST\_ADDR\_CONTINUOUS - destination data is continuous
- RCC\_DEST\_ADDR\_DISCONTINUOUS - destination data is discontinuous

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#) > [RCC\\_SRC\\_ADDR\\_CONTINUOUS Macro](#)

## RCC\_ROP\_0 Macro

C

```
#define RCC_ROP_0 0x00000000u1
#define RCC_ROP_1 0x00000008u1 // not (S or D)
#define RCC_ROP_2 0x00000010u1 // (not S) and D
#define RCC_ROP_3 0x00000018u1 // not (S)
#define RCC_ROP_4 0x00000020u1 // S and not (D)
#define RCC_ROP_5 0x00000028u1 // not (D)
#define RCC_ROP_6 0x00000030u1 // S xor D
#define RCC_ROP_7 0x00000038u1 // not (S and D)
#define RCC_ROP_8 0x00000040u1 // S and D
#define RCC_ROP_9 0x00000048u1 // not (S xor D)
#define RCC_ROP_A 0x00000050u1 // D
#define RCC_ROP_B 0x00000058u1 // not (S) or D
#define RCC_ROP_C 0x00000060u1 // S
#define RCC_ROP_D 0x00000068u1 // S or not (D)
#define RCC_ROP_E 0x00000070u1 // S or D
#define RCC_ROP_F 0x00000078u1 // 1 (WHITE)
```

### Overview

Raster Operation (ROP) option. Select one of the following 16 raster operation options whenever [Rectangle](#) Copy Processing Unit (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 ([BLACK](#))
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S)
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D

- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D
- RCC\_ROP\_9 - not (S xor D)
- RCC\_ROP\_A - D
- RCC\_ROP\_B - not (S) or D
- RCC\_ROP\_C - S
- RCC\_ROP\_D - S or not (D)
- RCC\_ROP\_E - S or D
- RCC\_ROP\_F - 1 (WHITE)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#) > [RCC\\_ROP\\_0 Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## RCC\_COPY Macro

C

```
#define RCC_COPY 0x00000080u1
#define RCC_SOLID_FILL 0x00000000u1
#define RCC_TRANSPARENT_COPY 0x00000300u1
```

### Overview

Type of [Rectangle](#) Copy Operations. Select one of the following rectangle copy operations and together with the ROP; the source, destination, current color set and transparency are evaluated on each pixel and the result written to the destination.

- RCC\_COPY - Copies the source data to the destination address with the selected ROP.
- RCC\_SOLID\_FILL - Fills the specified rectangle with the current color set.
- RCC\_TRANSPARENT\_COPY - Operation is the same as the COPY operation except that the source data is compared against the current color set. If the values match, the source data is not written to the destination. The source image is, therefore, transparent at such a location, allowing

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#) > [RCC\\_COPY Macro](#)

# Decompressing DEFLATEd data

## Functions

### Functions

	Name	Description
	<a href="#">Decompress</a>	Decompresses the nbytes number of data at SrcAddress and places starting from DestAddress. (Blocking)

### Links

[Functions](#), [Microchip Graphics Controller](#), [Legend](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Decompressing DEFLATEd data](#)

# Decompress Function

```
C  
BYTE Decompress(  
    DWORD SrcAddress,  
    DWORD DestAddress,  
    DWORD nbytes  
);
```

## Overview

Decompresses the nbytes number of data at SrcAddress and places starting from DestAddress. (Blocking)

## Input Parameters

Input Parameters	Description
DWORD SrcAddress	Source address
DWORD DestAddress	Destination address
DWORD nbytes	Number of bytes to be decompressed

## Returns

error flag

## Preconditions

SrcAddress must point to the start of a compressed block.

## Side Effects

Modifies workarea\_1 & workarea\_2 registers.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Decompressing DEFLATEd data](#) > [Decompress Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Palette Mode

[Files](#) | [Functions](#) | [Macros](#) | [Structures](#) | [Types](#)

## Files

Name	Description
<a href="#">Palette.h</a>	This is file Palette.h.

## Functions

	Name	Description
◆	<a href="#">ClearPaletteChangeError</a>	Clears the Palette change error status
◆	<a href="#">DisablePalette</a>	Disables the Palette mode.
◆	<a href="#">EnablePalette</a>	Enables the Palette mode.
◆	<a href="#">GetPaletteChangeError</a>	Returns the Palette change error status
◆	<a href="#">IsPaletteEnabled</a>	Returns if the Palette mode is enabled or not.
◆	<a href="#">PaletteInit</a>	Initializes the color look up table (CLUT).
◆	<a href="#">RequestPaletteChange</a>	Loads the palettes from the flash during vertical blanking period if possible, otherwise loads immediately.
◆	<a href="#">SetPalette</a>	Programs a block of palette entries starting from startEntry and until

		startEntry + length from the flash immediately.
◆	<a href="#"><u>SetPaletteBpp</u></a>	Sets the color look up table (CLUT) number of valid entries.
◆	<a href="#"><u>SetPaletteFlash</u></a>	Loads the palettes from the flash immediately.

## Macros

Name	Description
<a href="#"><u>RequestEntirePaletteChange</u></a>	Loads all the palette entries from the flash during vertical blanking period if possible, otherwise loads immediately.
<a href="#"><u>SetEntirePalette</u></a>	Programs the whole 256 entry palette with new color values from flash.

## Structures

Name	Description
<a href="#"><u>PALETTE_FLASH</u></a>	Structure for the palette stored in FLASH memory.
<a href="#"><u>PALETTE_HEADER</u></a>	Structure for the palette header.

## Types

Name	Description
<a href="#"><u>PALETTE_EXTERNAL</u></a>	Structure for palette stored in EXTERNAL memory space. (example: External SPI or

parallel Flash, EDS\_EPMP)

## Links

---

[Files](#), [Functions](#), [Microchip Graphics Controller](#), [Legend](#), [Macros](#),  
[Structures](#), [Types](#)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ClearPaletteChangeError Function

Palette.h

```
C  
void ClearPaletteChangeError();
```

## Overview

Clears the Palette change error status

## Returns

none

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [ClearPaletteChangeError Function](#)

# DisablePalette Function

Palette.h

```
C
```

```
void DisablePalette();
```

## Overview

Disables the Palette mode.

## Returns

none

## Preconditions

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [DisablePalette Function](#)

# EnablePalette Function

Palette.h

```
C
```

```
void EnablePalette();
```

## Overview

Enables the Palette mode.

## Returns

none

## Preconditions

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [EnablePalette Function](#)

# GetPaletteChangeError Function

Palette.h

```
C
```

```
BYTE GetPaletteChangeError();
```

## Overview

Returns the Palette change error status

## Returns

Returns the palette change status. 1 - If the palette change error occurred 0 - If no palette change error occurred

## Preconditions

none

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [GetPaletteChangeError Function](#)

# IsPaletteEnabled Function

Palette.h

```
C
```

```
BYTE IsPaletteEnabled();
```

## Overview

Returns if the Palette mode is enabled or not.

## Returns

Returns the palette mode status. 1 - If the palette mode is enabled 0 - If the palette mode is disabled

## Preconditions

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [IsPaletteEnabled Function](#)

# PaletteInit Function

Palette.h

```
C  
void PaletteInit();
```

## Overview

Initializes the color look up table (CLUT).

## Returns

none

## Preconditions

none

## Side Effects

All rendering will use the new palette entries.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [PaletteInit Function](#)

# RequestPaletteChange Function

Palette.h

```
C
void RequestPaletteChange(
    void * pPalette,
    WORD startEntry,
    WORD length
);
```

## Overview

Loads the palettes from the flash during vertical blanking period if possible, otherwise loads immediately.

## Input Parameters

Input Parameters	Description
void * pPalette	Pointer to the palette structure
WORD startEntry	Start entry to load (inclusive)
WORD length	Number of entries

## Returns

none

## Preconditions

Palette must be initialized by [PaletteInit\(\)](#).

## Side Effects

There may be a slight flicker when the Palette entries are getting loaded one by one.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [RequestPaletteChange Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetPalette Function

Palette.h

```
C
BYTE SetPalette(
    void * pPalette,
    WORD startEntry,
    WORD length
);
```

## Overview

Programs a block of palette entries starting from startEntry and until startEntry + length from the flash immediately.

## Input Parameters

Input Parameters	Description
void * pPalette	Pointer to the palette structure
WORD startEntry	Start entry to load (inclusive)
WORD length	Number of entries

## Returns

Returns the status of the palette set. 0 - Set was successful 1 - Set was not successful

## Preconditions

Palette must be initialized by [PaletteInit\(\)](#).

## Side Effects

There may be a slight flicker when the Palette entries are getting loaded one by one.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [SetPalette Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetPaletteBpp Function

Palette.h

```
C
BYTE SetPaletteBpp(
    BYTE bpp
);
```

## Overview

Sets the color look up table (CLUT) number of valid entries.

## Input Parameters

Input Parameters	Description
BYTE bpp	Bits per pixel

## Returns

Returns the status of the change. 0 - Change was successful 1 - Change was not successful

## Preconditions

Palette must be initialized by [PaletteInit\(\)](#).

## Side Effects

none

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [SetPaletteBpp Function](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# SetPaletteFlash Function

Palette.h

```
C
BYTE SetPaletteFlash(
    PALETTE_ENTRY * pPaletteEntry,
    WORD startEntry,
    WORD length
);
```

## Overview

Loads the palettes from the flash immediately.

## Input Parameters

Input Parameters	Description
PALETTE_ENTRY * pPaletteEntry	Pointer to the palette table in ROM
WORD startEntry	Start entry to load (inclusive)
WORD length	Number of entries

## Returns

Returns the status of the palette set. 0 - Set was successful 1 - Set was not successful

## Preconditions

Palette must be initialized by [PaletteInit\(\)](#).

## Side Effects

There may be a slight flicker when the Palette entries are getting loaded one by one.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [SetPaletteFlash Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# PALETTE\_FLASH Structure

## Palette.h

C

```
typedef struct {  
    SHORT type;  
    PALETTE_HEADER header;  
    PALETTE_ENTRY * pPaletteEntry;  
} PALETTE_FLASH;
```

## Overview

Structure for the palette stored in FLASH memory.

## Members

Members	Description
SHORT type;	Type must be FLASH
PALETTE_HEADER header;	Contains information on the palette
PALETTE_ENTRY * pPaletteEntry;	Pointer to the palette. Number of entries is determined by the header.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [PALETTE\\_FLASH Structure](#)

# PALETTE\_HEADER Structure

Palette.h

```
C
typedef struct {
    WORD id;
    WORD length;
} PALETTE_HEADER;
```

## Overview

Structure for the palette header.

## Members

Members	Description
WORD id;	User defined ID
WORD length;	number of palette entries (number of colors in the palette)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > **PALETTE\_HEADER Structure**

# PALETTE\_EXTERNAL Type

Palette.h

```
C
```

```
typedef GFX_EXTDATA PALETTE_EXTERNAL;
```

## Overview

Structure for palette stored in EXTERNAL memory space.  
(example: External SPI or parallel Flash, EDS\_EPMP)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > **PALETTE\_EXTERNAL Type**

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# RequestEntirePaletteChange Macro

Palette.h

```
C
```

```
#define RequestEntirePaletteChange(pPalette) RequestI
```

## Overview

Loads all the palette entries from the flash during vertical blanking period if possible, otherwise loads immediately.

## Input Parameters

Input Parameters	Description
pPalette	Pointer to the palette structure

## Returns

none

## Preconditions

PPalette must be initialized by [PaletteInit\(\)](#).

## Side Effects

There may be a slight flicker when the Palette entries are getting loaded one by one.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [RequestEntirePaletteChange Macro](#)

[Contents](#) | [Index](#) | [Home](#)

# SetEntirePalette Macro

Palette.h

```
C
```

```
#define SetEntirePalette(pPalette) SetPalette(pPalet
```

## Overview

Programs the whole 256 entry palette with new color values from flash.

## Input Parameters

Input Parameters	Description
pPalette	Pointer to the palette structure

## Returns

Returns the status of the palette set. 0 - Set was successful 1 - Set was not successful

## Preconditions

Palette must be initialized by [PaletteInit\(\)](#).

## Side Effects

There may be a slight flicker when the Palette entries are getting loaded one by one.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [SetEntirePalette Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Palette.h

[Functions](#) | [Macros](#) | [Structures](#) | [Types](#)

This is file Palette.h.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [Palette.h](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Set Up Display Interface

## Macros

### Macros

Name	Description
<a href="#">GFX_GCLK_DIVIDER</a>	<p>The following are additional macros used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet (<a href="#">PIC24FJ256DA210_I</a>).</p> <ul style="list-style-type: none"><li>• When using internal memory:<ul style="list-style-type: none"><li>◦ <a href="#">GFX_GCLK_DIVIDER</a> - Set the pixel clock divider.</li><li>◦ <a href="#">GFX_DISPLAY_BUFFER_SIZE</a> - Set the Display Buffer Size (Display Buffer Size calculated by <math>\text{depth}/2</math>).</li></ul></li><li>• When using external memory: External memory may be required and/or chip select may be required. See the Family Reference Manual for more information.</li></ul>
<a href="#">GFX_EPMP_CS1_BASE_ADDRESS</a>	<p>The following are additional macros used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet (<a href="#">PIC24FJ256DA210_I</a>).</p> <ul style="list-style-type: none"><li>• When using internal memory:<ul style="list-style-type: none"><li>◦ <a href="#">GFX_GCLK_DIVIDER</a></li></ul></li></ul>

--

- for the pixel c
- [GFX\\_DISPL/](#)
- Set the Disp
- [GFX\\_DISPL/](#)
- Display Buffe
- calculated by
- depth/2).
- When using exter
- memory may be p
- and/or chip select
- Family Reference

<a href="#">GFX_EPMP_CS1_MEMORY_SIZE</a>
------------------------------------------

- The following are addi
- used when using the c
- Module that comes wi
- (PIC24FJ256DA210 D
- ([PIC24FJ256DA210\\_I](#)
- When using interr
  - GFX\_GCLK\_
  - for the pixel c
  - [GFX\\_DISPL/](#)
  - Set the Disp
  - [GFX\\_DISPL/](#)
  - Display Buffe
  - calculated by
  - depth/2).
  - When using exter
  - memory may be p
  - and/or chip select
  - Family Reference

<a href="#">GFX_EPMP_CS2_BASE_ADDRESS</a>
-------------------------------------------

The following are addi

used when using the c

Module that comes wi

(PIC24FJ256DA210 D

([PIC24FJ256DA210\\_I](#)

- When using internal memory:
  - GFX\_GCLK\_ for the pixel clock
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) - Set the Display Buffer Size (calculated by depth/2).
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) Display Buffer Size calculated by depth/2).
- When using external memory may be present and/or chip select. See Family Reference Manual.

### GFX\_EPMP\_CS2\_MEMORY\_SIZE

The following are additional registers used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet ([PIC24FJ256DA210\\_I](#)

- When using internal memory:
  - GFX\_GCLK\_ for the pixel clock
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) - Set the Display Buffer Size (calculated by depth/2).
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) Display Buffer Size calculated by depth/2).
- When using external memory may be present and/or chip select. See Family Reference Manual.

## GFX\_DISPLAY\_BUFFER\_LENGTH

The following are additional registers used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet ([PIC24FJ256DA210\\_I](#)

- When using internal memory
  - `GFX_GCLK_PIXEL_DEPTH` for the pixel clock
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer Length (Display Buffer calculated by  $\text{depth}/2$ ).
- When using external memory may be present and/or chip select. See the PIC24FJ256DA210 Family Reference Manual.

## GFX\_DISPLAY\_BUFFER\_START\_ADDRESS

The following are additional registers used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet ([PIC24FJ256DA210\\_I](#)

- When using internal memory
  - `GFX_GCLK_PIXEL_DEPTH` for the pixel clock
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer Start Address (Display Buffer calculated by  $\text{depth}/2$ ).
- When using external memory may be present and/or chip select. See the PIC24FJ256DA210 Family Reference Manual.

memory may be p  
and/or chip select  
Family Reference

## Links

---

### [Microchip Graphics Controller, Macros](#)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_GCLK\_DIVIDER Macro

```
C
```

```
#define GFX_GCLK_DIVIDER 61
```

### Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - `GFX_GCLK_DIVIDER` - Set the clock divider for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer location.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS1.
  - [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS2.
  - [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS2. This value sets how many EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> <span style="float: right;">&gt;=</span> <u>GFX_DISPLAY_BUFFER_LENGTH</u>

EQ2	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ3	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> <u>GFX_DISPLAY_BUFFER_LENGTH</u>	>=
EQ5	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ6	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_GCLK\\_DIVIDER Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_EPMP\_CS1\_BASE\_ADDRESS Macro

```
C
```

```
#define GFX_EPMP_CS1_BASE_ADDRESS 0x00020000uL
```

### Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - GFX\_GCLK\_DIVIDER - Set the clock divider for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer location.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - GFX\_EPMP\_CS1\_BASE\_ADDRESS - Set the location of the external memory mapped to the EPMP CS1.
  - [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS2.
  - [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS2. This value sets how many EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> <span style="float: right;">&gt;=</span> <u>GFX_DISPLAY_BUFFER_LENGTH</u>

EQ2	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ3	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> <u>GFX_DISPLAY_BUFFER_LENGTH</u>	>=
EQ5	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ6	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_EPMP\_CS1\_MEMORY\_SIZE Macro

```
C
```

```
#define GFX_EPMP_CS1_MEMORY_SIZE 0x40000u1
```

### Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - `GFX_GCLK_DIVIDER` - Set the clock divider for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer location.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS1.
  - [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS2.
  - `GFX_EPMP_CS1_MEMORY_SIZE` - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS2. This value sets how many EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
GFX_EPMP_CS1_MEMORY_SIZE	none
GFX_EPMP_CS1_MEMORY_SIZE	<u>USE_DOUBLE_BUFFERING</u>
GFX_EPMP_CS1_MEMORY_SIZE	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	GFX_EPMP_CS1_MEMORY_SIZE $\geq$ <u>GFX_DISPLAY_BUFFER_LENGTH</u>

EQ2	GFX_EPMP_CS1_MEMORY_SIZE ( <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> *2)	>=
EQ3	GFX_EPMP_CS1_MEMORY_SIZE ( <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	<a href="#">GFX_EPMP_CS2_MEMORY_SIZE</a> <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a>	>=
EQ5	<a href="#">GFX_EPMP_CS2_MEMORY_SIZE</a> ( <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> *2)	>=
EQ6	<a href="#">GFX_EPMP_CS2_MEMORY_SIZE</a> ( <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_EPMP\_CS2\_BASE\_ADDRESS Macro

```
C
```

```
#define GFX_EPMP_CS2_BASE_ADDRESS 0x00020000uL
```

### Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - GFX\_GCLK\_DIVIDER - Set the clock divider for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer location.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS1.
  - GFX\_EPMP\_CS2\_BASE\_ADDRESS - Set the location of the external memory mapped to the EPMP CS2.
  - [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS2. This value sets how many EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> <span style="float: right;">&gt;=</span> <u>GFX_DISPLAY_BUFFER_LENGTH</u>

EQ2	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ3	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> <u>GFX_DISPLAY_BUFFER_LENGTH</u>	>=
EQ5	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ6	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_EPMP\_CS2\_MEMORY\_SIZE Macro

```
C
```

```
#define GFX_EPMP_CS2_MEMORY_SIZE 0x40000u1
```

### Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - `GFX_GCLK_DIVIDER` - Set the clock divider for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer location.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS1.
  - [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS2.
  - [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - `GFX_EPMP_CS2_MEMORY_SIZE` - Set the size of the memory mapped to the EPMP CS2. This value sets how many EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
GFX_EPMP_CS2_MEMORY_SIZE	none
GFX_EPMP_CS2_MEMORY_SIZE	<u>USE_DOUBLE_BUFFERING</u>
GFX_EPMP_CS2_MEMORY_SIZE	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> <span style="float: right;">&gt;=</span> <u>GFX_DISPLAY_BUFFER_LENGTH</u>

EQ2	<a href="#">GFX_EPMP_CS1_MEMORY_SIZE</a> <a href="#">(GFX_DISPLAY_BUFFER_LENGTH*2)</a>	>=
EQ3	<a href="#">GFX_EPMP_CS1_MEMORY_SIZE</a> <a href="#">(GFX_DISPLAY_BUFFER_LENGTH*2)</a> GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	GFX_EPMP_CS2_MEMORY_SIZE <a href="#">GFX_DISPLAY_BUFFER_LENGTH</a>	>=
EQ5	GFX_EPMP_CS2_MEMORY_SIZE <a href="#">(GFX_DISPLAY_BUFFER_LENGTH*2)</a>	>=
EQ6	GFX_EPMP_CS2_MEMORY_SIZE <a href="#">(GFX_DISPLAY_BUFFER_LENGTH*2)</a> GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_DISPLAY\_BUFFER\_LENGTH Macro

```
C
```

```
#define GFX_DISPLAY_BUFFER_LENGTH 0x00025800uL
```

### Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - `GFX_GCLK_DIVIDER` - Set the clock divider for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer location.
  - `GFX_DISPLAY_BUFFER_LENGTH` - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS1.
  - [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS2.
  - [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS2. This value sets how many EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> >= <u>GFX_DISPLAY_BUFFER_LENGTH</u>

EQ2	<a href="#"><u>GFX_EPMP_CS1_MEMORY_SIZE</u></a> (GFX_DISPLAY_BUFFER_LENGTH*2)	>=
EQ3	<a href="#"><u>GFX_EPMP_CS1_MEMORY_SIZE</u></a> (GFX_DISPLAY_BUFFER_LENGTH*2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	<a href="#"><u>GFX_EPMP_CS2_MEMORY_SIZE</u></a> GFX_DISPLAY_BUFFER_LENGTH	>=
EQ5	<a href="#"><u>GFX_EPMP_CS2_MEMORY_SIZE</u></a> (GFX_DISPLAY_BUFFER_LENGTH*2)	>=
EQ6	<a href="#"><u>GFX_EPMP_CS2_MEMORY_SIZE</u></a> (GFX_DISPLAY_BUFFER_LENGTH*2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_DISPLAY\\_BUFFER\\_LENGTH Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GFX\_DISPLAY\_BUFFER\_START\_ADDRESS

## Macro

```
C
```

```
#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00020000u
```

## Overview

The following are additional Hardware Profile macros used when using the driver for the Microchip Graphics Module that comes with the PIC Microcontroller (PIC24FJ256DA210 Device Family) ([PIC24FJ256DA210\\_DEV\\_BOARD](#)).

- When using internal or external memory
  - `GFX_GCLK_DIVIDER` - Set the clock divider for the pixel clock.
  - `GFX_DISPLAY_BUFFER_START_ADDRESS` - Set the Display Buffer location.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer length (size) in bytes. This is calculated by the display's width\*height\*(color depth/2).
- When using external memory only. External memory may be placed in chip select 1 (CS1) and/or chip select 2 (CS2) regions. Refer to EPMP Family Reference Manual for details on how to use EPMP for graphics applications.
  - [GFX\\_EPMP\\_CS1\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS1.
  - [GFX\\_EPMP\\_CS2\\_BASE\\_ADDRESS](#) - Set the location of the external memory mapped to the EPMP CS2.
  - [GFX\\_EPMP\\_CS1\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS1. This value sets how many EPMP address lines will be used.
  - [GFX\\_EPMP\\_CS2\\_MEMORY\\_SIZE](#) - Set the size of the memory mapped to the EPMP CS2. This value sets how many

EPMP address lines will be used.

If the display buffer is located in external memory (i.e in PIC24FJ256DA210) then the memory requirement for the graphics use must fit into the chip select region size. The table below summarizes the requirements.

Buffer Name	Enabled Features
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS1_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	none
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u>
<u>GFX_EPMP_CS2_MEMORY_SIZE</u>	<u>USE_DOUBLE_BUFFERING</u> + <u>USE_COMP_IPU</u>

Equation	Description
EQ1	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> >=

	<u>GFX_DISPLAY_BUFFER_LENGTH</u>	
EQ2	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ3	<u>GFX_EPMP_CS1_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +
EQ4	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> <u>GFX_DISPLAY_BUFFER_LENGTH</u>	>=
EQ5	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2)	>=
EQ6	<u>GFX_EPMP_CS2_MEMORY_SIZE</u> ( <u>GFX_DISPLAY_BUFFER_LENGTH</u> *2) GFX_COMPRESSED_BUFFER_SIZE GFX_DECOMPRESSED_BUFFER_SIZE	>= + +

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#) > [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## External or Internal Memory and Palettes

Applications can run in in PIC24FJ256DA210 Device Family using internal memory or external memory with palette enabled or disabled. The table below summarizes the allowed color depth when combining palette with the use of internal or external memory.

Memory Location	Palette Mode	Color Depth (BPP)	Notes
Internal	enabled	1, 2, 4, 8	When using internal memory and running TFT display, palette mode must be enabled.
External	enabled	8, 16	Palette mode can only be used for 8BPP color depth when using external memory.
External	disabled	16	When using external memory and palette is disabled and running TFT display, 16 BPP color depth must be used.

### Applications Using Palettes

Settings in Graphics Configuration (*GraphicsConfig.h*)

[Copy Code](#)

```
#define USE_PALETTE  
#define USE_PALETTE_EXTERNAL
```

```
// (note: PIC24FJ256DA210 Device Family color palet
```

```
// entries, therefore the maximum color depth that
// using palette is 8 BPP)
#define COLOR_DEPTH 8
```

### Settings in Hardware Profile (HardwareProfile.h)

[Copy Code](#)

```
////////////////////////////////////
// Microchip Specific Development Tools
////////////////////////////////////
#define PIC24FJ256DA210_DEV_BOARD
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_

////////////////////////////////////
// PIC24FJ256DA210 Graphics Module required setting
////////////////////////////////////
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210

#define GFX_GCLK_DIVIDER 38

#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00004B00

#define GFX_DISPLAY_BUFFER_LENGTH 0x0004B000u1

// If using external memory add macros to enable EP
// location of GFX_DISPLAY_BUFFER_START_ADDRESS (s
```

*In application code initialize and enable palette before initializing the Graphics Library*

[Copy Code](#)

```
// set the palette color depth
SetPaletteBpp(8);
// initialize the palette
SetPalette((void*)&MainPalette, 0, 256);

// enable the use of the palette
EnablePalette();
```

## Applications Using Internal Memory

Settings in Graphics Configuration (GraphicsConfig.h)

[Copy Code](#)

```
// define the color depth to use (1, 2, 4, or 8BPP)
#define COLOR_DEPTH 8
```

Settings in Graphics Configuration (GraphicsConfig.h) Settings in Hardware Profile (HardwareProfile.h)

[Copy Code](#)

```
////////////////////////////////////
// Microchip Specific Development Tools
////////////////////////////////////
#define PIC24FJ256DA210_DEV_BOARD
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_

////////////////////////////////////
// PIC24FJ256DA210 Graphics Module required setting
////////////////////////////////////
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210
```

```
#define GFX_GCLK_DIVIDER 38

#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00004B00

#define GFX_DISPLAY_BUFFER_LENGTH 0x0004B000u1
```

## Applications Using External Memory

Settings in Graphics Configuration (GraphicsConfig.h)

```
Copy Code
// define the color depth to use (8 or 16BPP)
#define COLOR_DEPTH 16
```

Settings in Hardware Profile (HardwareProfile.h)

```
Copy Code
////////////////////////////////////
// Microchip Specific Development Tools
////////////////////////////////////
#define PIC24FJ256DA210_DEV_BOARD
#define GFX_USE_DISPLAY_PANEL_TFT_G240320LTSW_118W_

////////////////////////////////////
// PIC24FJ256DA210 Graphics Module required setting
////////////////////////////////////
#define GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210

#define GFX_GCLK_DIVIDER 61

#define USE_16BIT_PMP
```

```
#define GFX_DISPLAY_BUFFER_START_ADDRESS 0x00020000

#define GFX_DISPLAY_BUFFER_LENGTH 0x00025800u1

#define GFX_EPMP_CS1_BASE_ADDRESS 0x00020000u1

#define GFX_EPMP_CS1_MEMORY_SIZE 0x80000u1

#define GFX_EPMP_CS2_BASE_ADDRESS 0x000A0000u1

#define GFX_EPMP_CS2_MEMORY_SIZE 0x80000u1
```

## Links

---

### [Microchip Graphics Controller](#)

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [External or Internal Memory and Palettes](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Image Decoders

## Demo Project

The Image Decoder Library supports the decoding of images in JPEG, BMP, and GIF format in PIC24, dsPIC, and PIC32 devices. This is a supplement to the Graphics Library but could be used without the Graphics Library. It not only supports input data through the Microchip's MDD file system but it can also be configured to support user specific inputs from ROM, external EEPROM, etc. The output can be sent to the Graphics Display through the driver provided with the Graphics Library or to a callback function where user can further render the decoded image (even if Graphics Library is not used). The individual decoders provided uses the stack for the work memory and hence a heap is not required.

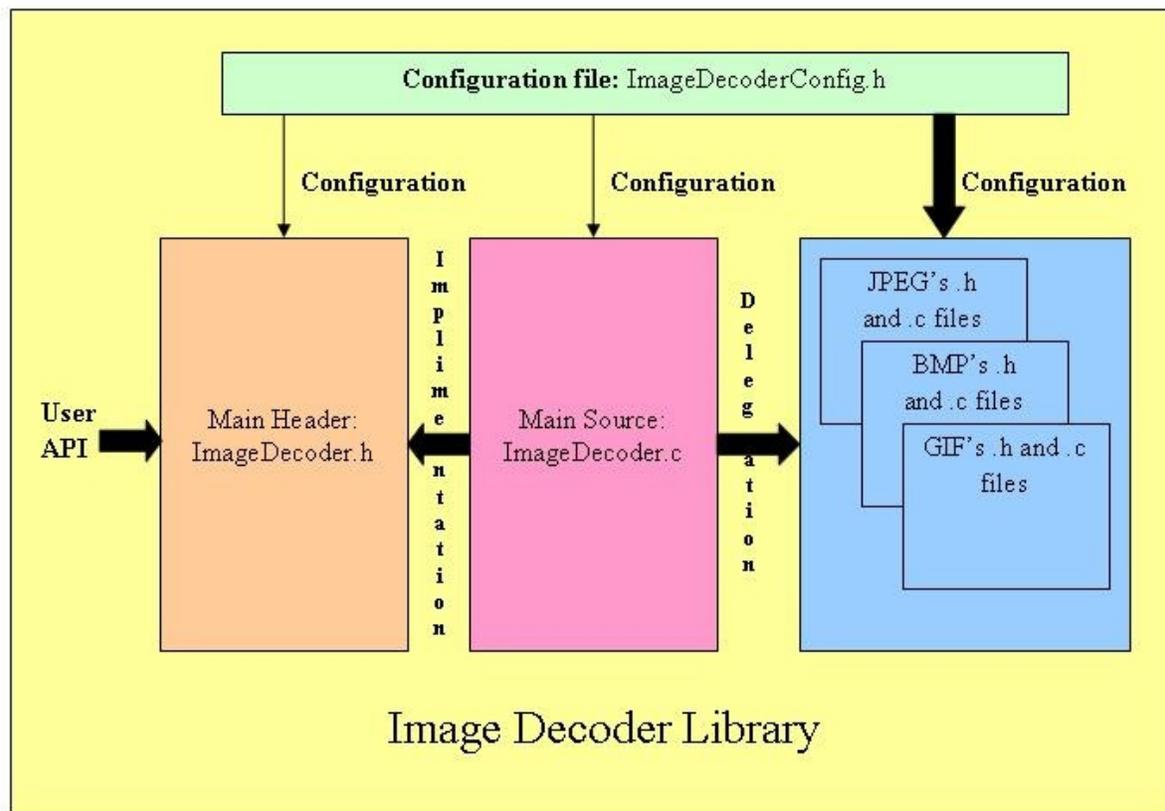
## Design

The Image Decoder library includes the following files:

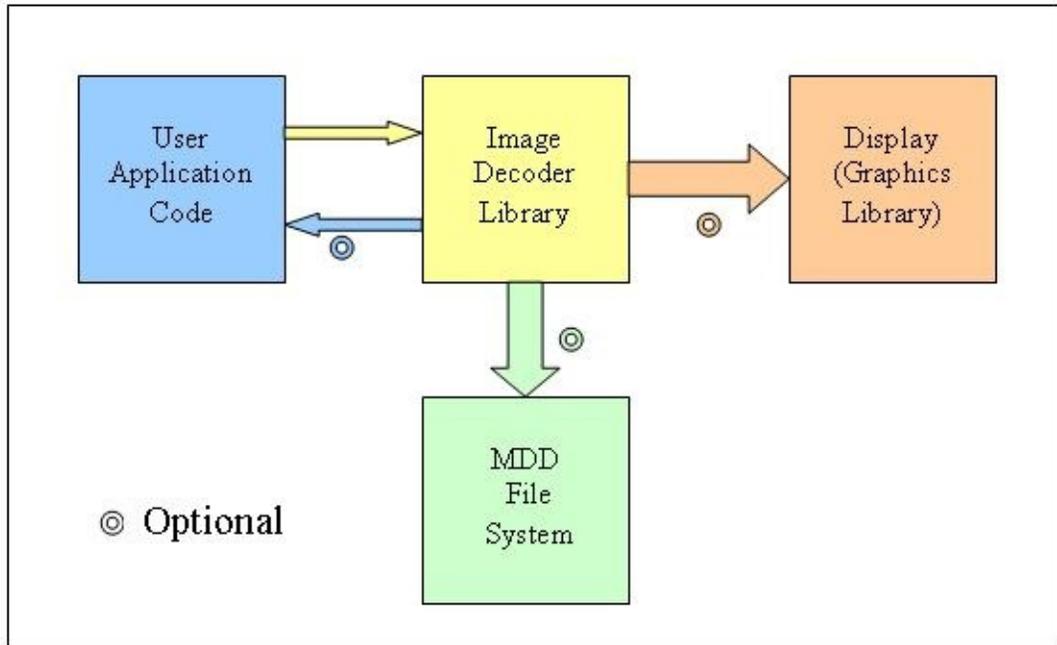
1. Configuration file
  - ImageDecoderConfig.h
2. Header files
  - Main Header
    - ImageDecoder.h
  - Individual decoder headers
    - JpegDecoder.h
    - BmpDecoder.h
    - GifDecoder.h
3. Source files
  - Main Source file

- ImageDecoder.c
  - Individual Source files
    - JpegDecoder.c
    - BmpDecoder.c
    - GifDecoder.c
4. Support files
- jidctint.c

This can be diagrammatically explained as below:



The user application interacts with the Image Decoder Library as per the diagram below:



User can use the Graphics Library for output or can provide his/her own output pixel handler callback function. Similarly, user can use MDD file system or provide his/her own input source which implements some specific APIs explained later.

## Configuration

The compile time configurations can be done using the ImageDecoderConfig.h file. This file must be copied into the application folder like the other config files. The options provided are as explained below:

### 1. Image format support

The image formats which are not required can be compiled out by commenting out the respective defines. The below section says that GIF support to be excluded.

[Copy Code](#)

```
/* Comment out the image formats which are not requ
#define IMG_SUPPORT_BMP
#define IMG_SUPPORT_JPEG
//#define IMG_SUPPORT_GIF
```

## 2. Optimize for Graphics Library

If user application requires to output the image directly to the display without any double buffering, then the code can be optimized by defining as below:

```
/* Comment out if output has to be given through a
/* If defined, the code is optimized to use only th
#define IMG_USE_ONLY_565_GRAPHICS_DRIVER_FOR_OUTPUT
```

If either double buffering has to be done or if user specific rendering has to be done, comment out the above mentioned line and provide a callback function to render the pixel values. The callback function will be explained in the API section. If commented out, then the width and height of the display screen has to be provided using the following defines

```
/* If the above define is commented out (Graphics d
#ifndef IMG_USE_ONLY_565_GRAPHICS_DRIVER_FOR_OUTPUT
    #define DISP_HOR_RESOLUTION        320
    #define DISP_VER_RESOLUTION       240
#endif
```

## 3. Optimize for MDD file system

If user application uses only the MDD file system provided by Microchip Technology Inc., the code can be optimized by defining as below:

[Copy Code](#)

```
/* If defined, the code is optimized to use only th  
#define IMG_USE_ONLY_MDD_FILE_SYSTEM_FOR_INPUT
```

If MDD file system is not used or if additional input formats have to be supported, the above define has to be commented out and a structure pointing to the required APIs (IMG\_FILE\_SYSTEM\_API defined in ImageDecoder.h) must be provided by the user.

#### 4. Loop callback support

Since decoder takes up significant amount of time decoding an image and user may want to update some information on the display or to send/receive data through communication channel, etc... it is possible to release processing power in the middle of decoding process by calling a callback function provided by the user. The user can do all the housekeeping activities inside the function. This option can be enabled by defining as below:

```
Copy Code  
/* If defined, the a loop callback function is call  
decoding loop so that application can do mainten  
getting data, updating display, etc... */  
#define IMG_SUPPORT_IMAGE_DECODER_LOOP_CALLBACK
```

The JPEG decoder calls the callback function after every 8x8 pixel block decode while the BMP and GIF decoders calls after every row decode.

If this support is not required, then this define can be commented out.

#### Footprint

The following are typical values for PIC24 with default compiler configuration. Actual values may vary with various factors such as compiler optimization levels and device used (PIC24/PIC32).

	RAM (stack)	ROM (no compiler optimizations)	
Image Decoder Core Code	30 Bytes	1 KBytes	
<b>Image Types</b>			<b>Decode Time (in seconds) Using 16 MIPS &amp; Compiler Optimization Level 3</b>
BMP	1 KBytes	10 KBytes	2 <sup>1</sup>
JPEG	3 KBytes	13 KBytes	2 <sup>1</sup>
GIF	11 KBytes(13 KBytes <sup>2</sup> )	6 KBytes	2 <sup>1</sup>

(1) Approximate value for average quality image file with a pixel resolution of 320x240. Additional variance in time can be observed due to image quality/image size and file access time.

(2) GIF code "Crush" optimization (GIF\_CRUSH\_PREV\_SYMBOL\_PTR\_TABLE) is turned off. This compile switch can be found in GifDecoder.h file. Enabling this will have an additional effect on the GIF decoding time.

## Demo Project

Name	Description
<a href="#">Image Decoder Demo</a>	This demo demonstrates the decoding of images with JPEG and BMP file formats.

## Links

---

[Demo Project](#)

---

[Image Decoders](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Image Decoders API

[Functions](#) | [Macros](#) | [Structures](#) | [Topics](#)

The Image Decoder Library distributed with the Graphics Library supports the decoding of images in JPEG, BMP, and GIF format in PIC24, dsPIC, and PIC32 devices. This is a supplement to the Graphics Library but could be used without the Graphics Library. This section describes the APIs for Image Decoder Library.

## Functions

	Name	Description
	<a href="#"><u>ImageDecode</u></a>	This function decodes and displays the image on the screen
	<a href="#"><u>ImageDecoderInit</u></a>	This function initializes the global variables to 0 and then initializes the driver. This must be called once before any other function of the library is called
	<a href="#"><u>ImageLoopCallbackRegister</u></a>	This function registers the loop callback function so that the decoder calls this function in every decoding loop. This can be used by the application program to do maintenance activities such as fetching data, updating the display, etc...
	<a href="#"><u>ImageDecodeTask</u></a>	This function completes one small part of the image decode function

## Macros

Name	Description
<a href="#">ImageFullScreenDecode</a>	This function decodes and displays the image on the screen in fullscreen mode with center aligned and downscaled if required
<a href="#">ImageAbort</a>	This function sets the Image Decoder's Abort flag so that decoding aborts in the next decoding loop.

## Structures

	Name	Description
	<a href="#">_BMPDECODER</a>	DATA STRUCTURES
	<a href="#">_GIFDECODER</a>	DATA STRUCTURES
	<a href="#">_JPEGDECODER</a>	DATA STRUCTURES

## Topics

Name	Description
<a href="#">Image Decoder Configuration</a>	The Image Decoder Library can be customized by adding or specifying the compile time options located in the application file named ImageDecoderConfig.h.

## Links

[Functions](#), [Legend](#), [Macros](#), [Structures](#), [Topics](#)

# Image Decoders API

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

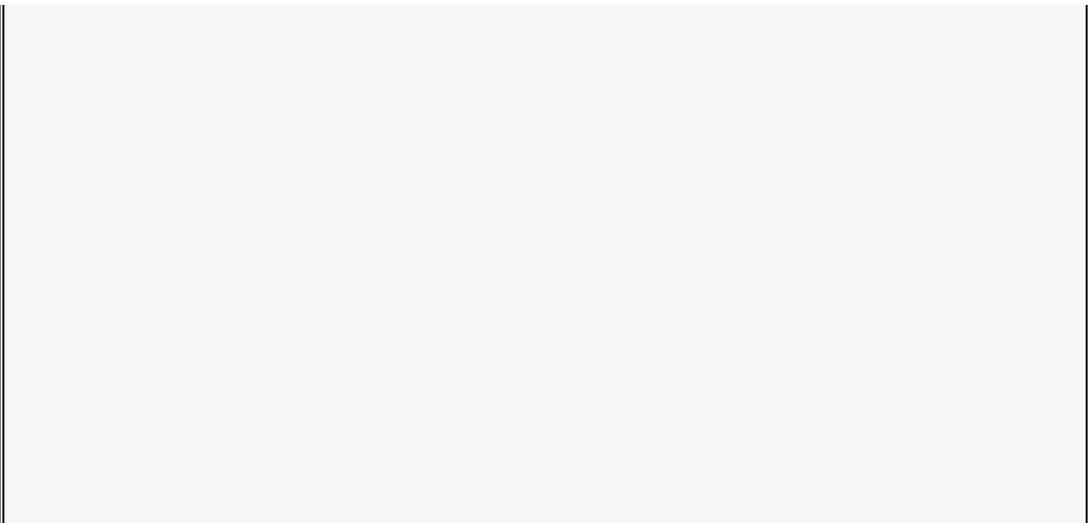
[Contents](#) | [Index](#) | [Home](#)

# Image Decoder Configuration

## Macros

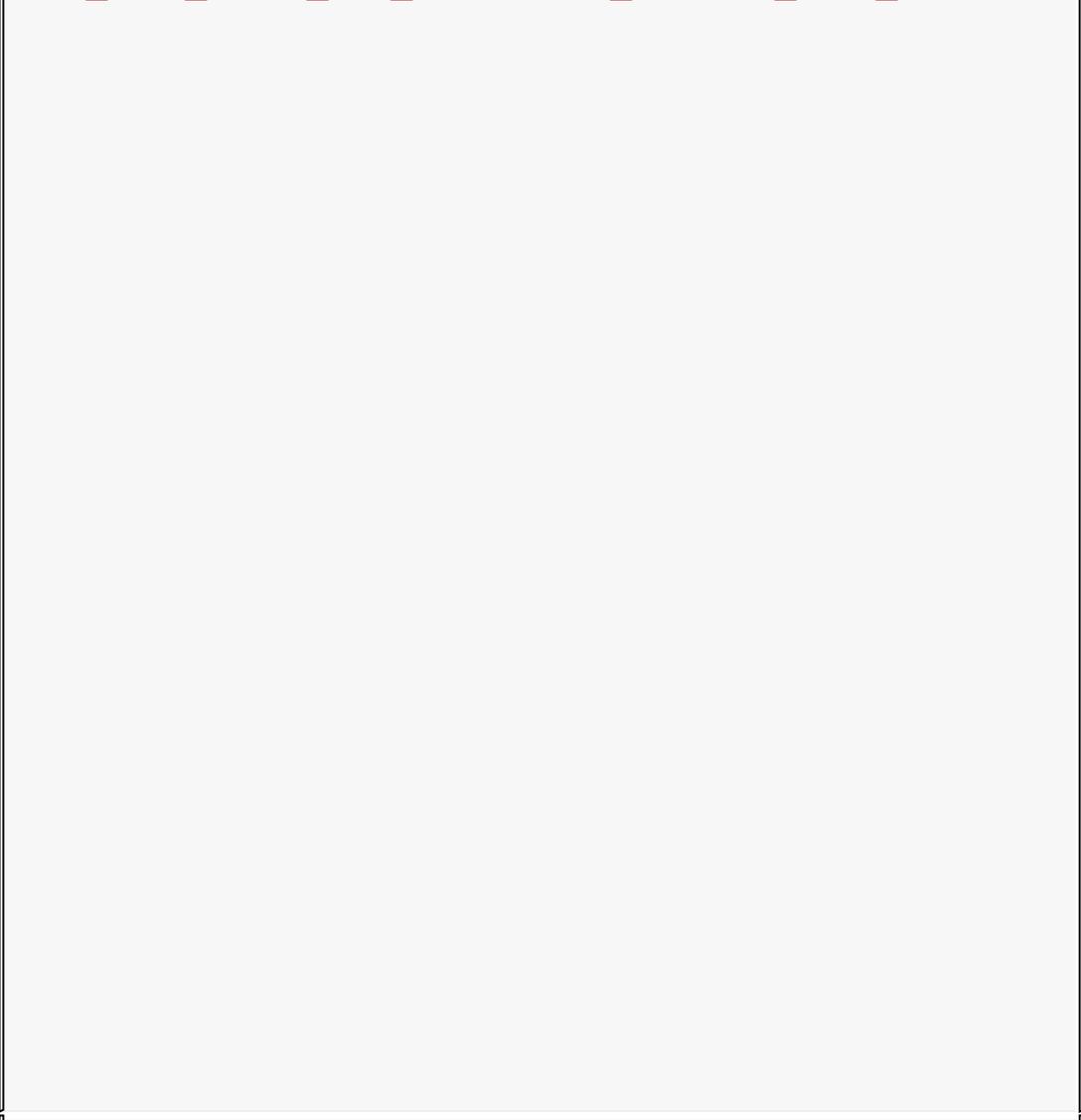
### Macros

Name	Descr
<a href="#"><u>IMG_SUPPORT_BMP</u></a>	Add the Image to enable bitmap decoding.
<a href="#"><u>IMG_SUPPORT_GIF</u></a>	Add the Image to enable image.
<a href="#"><u>IMG_SUPPORT_JPEG</u></a>	Add the Image to enable jpeg in decoding.
<a href="#"><u>IMG_SUPPORT_IMAGE_DECODER_LOOP_CALLBACK</u></a>	The default signal time delay and use update information display send/receive through communication channel to release.



power  
decodi  
calling  
functio  
user. T  
do all t  
activiti  
functio  
be ena  
this ma  
Image

IMG\_USE\_ONLY\_565\_GRAPHICS\_DRIVER\_FOR\_OUTPUT



Add th  
Image  
to opti  
graphi  
suppo  
color f  
render  
directly  
buffer.  
bufferi  
or if us  
render  
done,  
above  
and pr  
functio  
pixel v  
callbac  
explai  
section  
out, th  
height  
screer  
provid  
followi

IMG\_USE\_ONLY\_MDD\_FILE\_SYSTEM\_FOR\_INPUT

Add th  
Image  
to opti  
Memo  
Syster  
Microc  
Syster  
.

## Links

---

[Image Decoders API](#), [Macros](#)

[Image Decoders API](#) > [Image Decoder Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# IMG\_SUPPORT\_BMP Macro

```
C
```

```
#define IMG_SUPPORT_BMP
```

## Overview

Add this macro in ImageDecoderConfig.h to enable support for bitmap image format decoding.

[Image Decoders API](#) > [Image Decoder Configuration](#) > [IMG\\_SUPPORT\\_BMP Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# IMG\_SUPPORT\_GIF Macro

```
C
```

```
#define IMG_SUPPORT_GIF
```

## Overview

Add this macro in ImageDecoderConfig.h to enable support for gif image format decoding.

[Image Decoders API](#) > [Image Decoder Configuration](#) > [IMG\\_SUPPORT\\_GIF Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# IMG\_SUPPORT\_JPEG Macro

```
C
```

```
#define IMG_SUPPORT_JPEG
```

## Overview

Add this macro in ImageDecoderConfig.h to enable support for jpeg image format decoding.

[Image Decoders API](#) > [Image Decoder Configuration](#) > [IMG\\_SUPPORT\\_JPEG Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# IMG\_SUPPORT\_IMAGE\_DECODER\_LOOP\_CALL Macro

```
C
```

```
#define IMG_SUPPORT_IMAGE_DECODER_LOOP_CALLBACK
```

## Overview

The decoder may takes up significant amount of time decoding an image and user may want to update some information on the display or to send/receive data through some communication channels. It is possible to release processing power in the middle of decoding process by calling a callback function provided by the user. The user should do all the housekeeping activities inside the function. This option can be enabled by adding this macro in ImageDecoderConfig.h.

[Image Decoders API](#) > [Image Decoder Configuration](#) > [IMG\\_SUPPORT\\_IMAGE\\_DECODER\\_LOOP\\_CALLBACK Macro](#)

# IMG\_USE\_ONLY\_565\_GRAPHICS\_DRIVER\_FOR\_Macro

```
C
```

```
#define IMG_USE_ONLY_565_GRAPHICS_DRIVER_FOR_OUTPUT
```

## Overview

Add this macro in ImageDecoderConfig.h to optimize code for graphics driver that supports 16-bit 5-6-5 color format and rendering is done directly to the display buffer. If either double buffering has to be done or if user specific rendering has to be done, comment out the above mentioned line and provide a callback function to render the pixel values. The callback function will be explained in the API section. If commented out, then the width and height of the display screen has to be provided using the following defines

[Image Decoders API](#) > [Image Decoder Configuration](#) > [IMG\\_USE\\_ONLY\\_565\\_GRAPHICS\\_DRIVER\\_FOR\\_OUTPUT Macro](#)

# IMG\_USE\_ONLY\_MDD\_FILE\_SYSTEM\_FOR\_INPU Macro

```
C
```

```
#define IMG_USE_ONLY_MDD_FILE_SYSTEM_FOR_INPUT
```

## Overview

Add this macro in ImageDecoderConfig.h to optimize code to use Memory Disk Drive File System(MDDFS) of Microchip MDD File System Interface Library .

[Image Decoders API](#) > [Image Decoder Configuration](#) > [IMG\\_USE\\_ONLY\\_MDD\\_FILE\\_SYSTEM\\_FOR\\_INPUT Macro](#)

# ImageDecode Function

**C**

```
BYTE ImageDecode(  
    IMG_FILE * pImageFile,  
    IMG_FILE_FORMAT eImgFormat,  
    WORD wStartx,  
    WORD wStarty,  
    WORD wWidth,  
    WORD wHeight,  
    WORD wFlags,  
    IMG_FILE_SYSTEM_API * pFileAPIs,  
    IMG_PIXEL_OUTPUT pPixelOutput  
);
```

## Overview

This function decodes and displays the image on the screen

## Returns

Error code -> 0 means no error

## Side Effects

None

## Example

```
void main(void)  
{  
    IMG_FILE pImageFile;  
    IMG_vInitialize();  
    pImageFile = IMG_FOPEN("Image.jpg", "r");  
    if(pImageFile == NULL)  
    {
```

```
        <- Error handling ->
    }
    IMG_bDecode(pImageFile, IMG_JPEG, 0, 0, 320, 240, 0, NULL, NU
    IMG_FCLOSE(pImageFile);
    while(1);
}
```

---

## [Image Decoders API](#) > [ImageDecode Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ImageDecoderInit Function

```
C
```

```
void ImageDecoderInit();
```

## Overview

This function initializes the global variables to 0 and then initializes the driver. This must be called once before any other function of the library is called

## Returns

None

## Side Effects

The graphics driver will be reset

## Example

```
void main(void)
{
    ImageInit();
    ...
}
```

## [Image Decoders API](#) > [ImageDecoderInit Function](#)

# ImageLoopCallbackRegister Function

C

```
void ImageLoopCallbackRegister(  
    IMG_LOOP_CALLBACK pFn  
);
```

## Overview

This function registers the loop callback function so that the decoder calls this function in every decoding loop. This can be used by the application program to do maintenance activities such as fetching data, updating the display, etc...

## Returns

None

## Side Effects

The graphics driver will be reset

## Example

```
void Mainantance(void)  
{  
    ...  
}  
  
void main(void)  
{  
    ImageInit();  
    ImageLoopCallbackRegister(Mainantance);  
    ...  
}
```

---

## [Image Decoders API](#) > [ImageLoopCallbackRegister Function](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# ImageDecodeTask Function

```
C
```

```
BYTE ImageDecodeTask();
```

## Overview

This function completes one small part of the image decode function

## Returns

Status code - '1' means decoding is completed

- '0' means decoding is not yet completed, call this function again

## Side Effects

None

## Example

```
IMG_bFullScreenDecode(pImageFile, IMG_JPEG, NULL, NULL);  
while(!ImageDecodeTask());
```

## [Image Decoders API](#) > [ImageDecodeTask Function](#)

## ImageFullScreenDecode Macro

C

```
#define ImageFullScreenDecode(pImageFile, eImgFormat,  
    ImageDecode(pImageFile, eImgFormat, 0, 0, IM
```

### Overview

This function decodes and displays the image on the screen in fullscreen mode with center aligned and downscaled if required

### Returns

Error code -> 0 means no error

### Side Effects

None

### Example

```
void main(void)  
{  
    IMG_FILE pImageFile;  
    IMG_vInitialize();  
    pImageFile = IMG_FOPEN("Image.jpg", "r");  
    if(pImageFile == NULL)  
    {  
        <- Error handling ->  
    }  
    IMG_bFullScreenDecode(pImageFile, IMG_JPEG, NULL, NULL);  
    IMG_FCLOSE(pImageFile);  
    while(1);  
}
```

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## ImageAbort Macro

```
C
```

```
#define ImageAbort IMG_bAbortImageDecoding = 1;
```

### Overview

This function sets the Image Decoder's Abort flag so that decoding aborts in the next decoding loop.

### Returns

None

### Side Effects

None

### Example

```
void callback(void);
void main(void)
{
    IMG_FILE pImageFile;
    IMG_vInitialize();
    ImageLoopCallbackRegister(callback);
    pImageFile = IMG_FOPEN("Image.jpg", "r");
    if(pImageFile == NULL)
    {
        <- Error handling ->
    }
    IMG_bFullScreenDecode(pImageFile, IMG_JPEG, NULL, NULL);
    IMG_FCLOSE(pImageFile);
    while(1);
}
```

```
void callback(void)
{
    if(<- check for abort condition ->)
    {
        ImageAbort();
    }
}
```

---

## [Image Decoders API](#) > [ImageAbort Macro](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## \_BMPDECODER Structure

C

```
struct _BMPDECODER {
    IMG_FILE * pImageFile;
    LONG lWidth;
    LONG lHeight;
    LONG lImageOffset;
    WORD wPaletteEntries;
    BYTE bBitsPerPixel;
    BYTE bHeaderType;
    BYTE b1BmMarkerFlag : 1;
    BYTE b1CompressionType : 3;
    BYTE bNumOfPlanes : 3;
    BYTE b16bit565flag : 1;
    BYTE aPalette[256][3];
};
```

### Description

DATA STRUCTURES

### Members

Members	Description
IMG_FILE * pImageFile;	Image file pointer
BYTE aPalette[256] [3];	Each palette entry has RGB

[Image Decoders API](#) > [\\_BMPDECODER Structure](#)

[Contents](#) | [Index](#) | [Home](#)

## \_GIFDECODER Structure

**C**

```
struct _GIFDECODER {  
    IMG_FILE * pImageFile;  
    WORD wImageWidth;  
    WORD wImageHeight;  
    WORD wImageX;  
    WORD wImageY;  
    WORD wScreenWidth;  
    WORD wScreenHeight;  
    WORD wGlobalPaletteEntries;  
    WORD wLocalPaletteEntries;  
    BYTE bBgColorIndex;  
    BYTE bPixelAspectRatio;  
    BYTE blGifMarkerFlag : 1;  
    BYTE blGloabalColorTableFlag : 1;  
    BYTE blLocalColorTableFlag : 1;  
    BYTE blInterlacedFlag : 1;  
    BYTE blFirstcodeFlag : 1;  
    BYTE bInterlacePass : 3;  
    BYTE aPalette[256][3];  
    WORD awPalette[256];  
    BYTE abSymbol[4096];  
    WORD awPrevSymbolPtr[(4096 * 3)/4];  
    WORD wInitialSymbols;  
    WORD wMaxSymbol;  
    BYTE bInitialSymbolBits;  
    BYTE bMaxSymbolBits;  
    LONG lGlobalColorTablePos;  
    BYTE bWorkBits;  
    BYTE bRemainingDataInBlock;  
    BYTE bRemainingBits;  
    WORD wCurrentX;  
    WORD wCurrentY;
```

```
};
```

## Description

DATA STRUCTURES

## Members

Members	Description
IMG_FILE * pImageFile;	Image file pointer
BYTE aPalette[256] [3];	Each palette entry has RGB
WORD awPalette[256];	Each palette entry has RGB
BYTE abSymbol[4096];	For decoding
BYTE bWorkBits;	Work memory

## [Image Decoders API](#) > [\\_GIFDECODER Structure](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## JPEGDECODER Structure

C

```
struct _JPEGDECODER {
    IMG_FILE * pImageFile;
    BYTE bJFIF;
    BYTE bMajorRev;
    BYTE bMinorRev;
    BYTE bDataBits;
    WORD wWidth;
    WORD wHeight;
    BYTE bChannels;
    BYTE abChannelType[MAX_CHANNELS];
    BYTE abChannelHSampFactor[MAX_CHANNELS];
    BYTE abChannelVSampFactor[MAX_CHANNELS];
    BYTE abChannelQuantTableMap[MAX_CHANNELS];
    BYTE bQuantUses16bits;
    WORD awQuantTable[MAX_CHANNELS][64];
    WORD wRestartInterval;
    BYTE bHuffTables;
    BYTE abHuffAcSymLen[MAX_HUFF_TABLES][16];
    BYTE abHuffAcSymbol[MAX_HUFF_TABLES][256];
    BYTE abHuffDcSymLen[MAX_HUFF_TABLES][16];
    BYTE abHuffDcSymbol[MAX_HUFF_TABLES][16];
    WORD awHuffAcSymStart[MAX_HUFF_TABLES][16];
    WORD awHuffDcSymStart[MAX_HUFF_TABLES][16];
    BYTE abChannelHuffAcTableMap[MAX_CHANNELS];
    BYTE abChannelHuffDcTableMap[MAX_CHANNELS];
    BYTE bError;
    WORD wWorkBits;
    BYTE bBitsAvailable;
    BYTE bBlocksInOnePass;
    SHORT asOneBlock[MAX_BLOCKS][64];
    WORD wBlockNumber;
    BYTE abChannelMap[MAX_BLOCKS];
}
```

```

BYTE bSubSampleType;
SHORT asPrevDcValue[MAX_CHANNELS];
BYTE * pbCurrentHuffSymLenTable;
BYTE * pbCurrentHuffSymbolTable;
WORD * pwCurrentHuffSymStartTable;
WORD * pwCurrentQuantTable;
BYTE abDataBuffer[MAX_DATA_BUF_LEN];
WORD wBufferLen;
WORD wBufferIndex;
BYTE bFirstBit;
WORD wPrevX;
WORD wPrevY;
};

```

## Description

### DATA STRUCTURES

## Members

Members	Description
IMG_FILE * pImageFile;	Image file pointer
BYTE bJFIF;	JFIF marker found flag
BYTE bMajorRev;	Should be 1
BYTE bMinorRev;	Should be 0-2 but is not a show stopper ----- The x/y densities and thumbnail data are ignored
BYTE bDataBits;	Data precision, can

	be 8(, 12 or 16)
WORD wWidth;	Width in pixels
WORD wHeight;	Height in pixels
BYTE bChannels;	Number of channels e.g. YCbCr = 3
BYTE blQuantUses16bits;	If flag is set, it is an error as 16 bit is not supported
WORD awQuantTable[MAX_CHANNELS][64];	Supports only 8 & 16 bit resolutions
WORD wRestartInterval;	The restart interval in blocks
BYTE bHuffTables;	From DHT
BYTE abHuffAcSymLen[MAX_HUFF_TABLES][16];	Supports only 8 bit resolution
BYTE abHuffAcSymbol[MAX_HUFF_TABLES][256];	Maximum possible symbols are 256
BYTE abHuffDcSymLen[MAX_HUFF_TABLES][16];	Supports only 8 bit resolution
BYTE abHuffDcSymbol[MAX_HUFF_TABLES][16];	Maximum possible symbols are 16 for DC :-)
WORD awHuffAcSymStart[MAX_HUFF_TABLES][16];	Starting symbol for each length

WORD awHuffDcSymStart[MAX_HUFF_TABLES][16];	Starting symbol for each length
BYTE abChannelHuffAcTableMap[MAX_CHANNELS];	From SOS
WORD wWorkBits;	Work memory
SHORT asOneBlock[MAX_BLOCKS][64];	Temporary storage for a 8x8 block

---

## [Image Decoders API](#) > [\\_JPEGDECODER Structure](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Miscellaneous Topics

## Topics

This section contains common procedures to start using the library or to modify the library.

## Topics

Name	Description
<a href="#">Starting a New Project</a>	This outlines the procedure to create a new project that uses the Microchip Graphics Library from scratch.
<a href="#">Changing the default Font</a>	The library comes with the default font (Gentium 18). This font can be changed in two ways.
<a href="#">Advanced Font Features</a>	Fonts used in the library can be configured to use anti-aliasing and extended glyph support.
<a href="#">Using Primitive Rendering Functions in Blocking and Non-Blocking Modes</a>	Basic rendering functions such as <code>Line()</code> , <code>Rectangle()</code> , <code>Circle()</code> etc are referred to as functions in the Graphics 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 directly calls these functions can take advantage of the hardware accelerated primitives. How these functions are used will depend on the " <a href="#">Configuration Setting</a> ".
<a href="#">Using Microchip Graphics Module Color</a>	Utilizing the Color Look Up Table (CLUT) of the Microchip Graphics Module saves

<a href="#"><u>Look Up Table in Applications</u></a>	memory for both storage and display buffer. This short instructional manual outlines the procedure to create source code files to use the CLUT of the Microchip Graphics Module and enable the Microchip Graphics Library to use the hardware feature.
<a href="#"><u>Converting Images to Use a Common Palette in GIMP</u></a>	This manual describes how to convert an image or a set of images to use a common palette in GIMP.
<a href="#"><u>How to Define Colors in your Applications</u></a>	In most cases, the application will define its own set of colors and not use the default colors that comes with the Graphics Library. This section shows an example on how to do it.
<a href="#"><u>Connecting RGB data bus</u></a>	Display glasses will require 24 bit or 18 bit RGB color data. How do you do the connection when your display controller only puts out 16 bit RGB data bus?
<a href="#"><u>Adding New Device Driver</u></a>	This is a summary of the requirements to add a new device driver.

## Links

---

### [Topics](#)

---

### [Miscellaneous Topics](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Starting a New Project

The following is a step by step instruction to create a New Project from scratch with the Microchip Graphics Library

**Step 1:** Install the Microchip Application Libraries to get the Microchip Graphics Library. The Microchip Application Library installer can be downloaded from [www.microchip.com/MLA](http://www.microchip.com/MLA). Once installed the Microchip Graphics Library files will be located in the directory structure shown below.

[Copy Code](#)

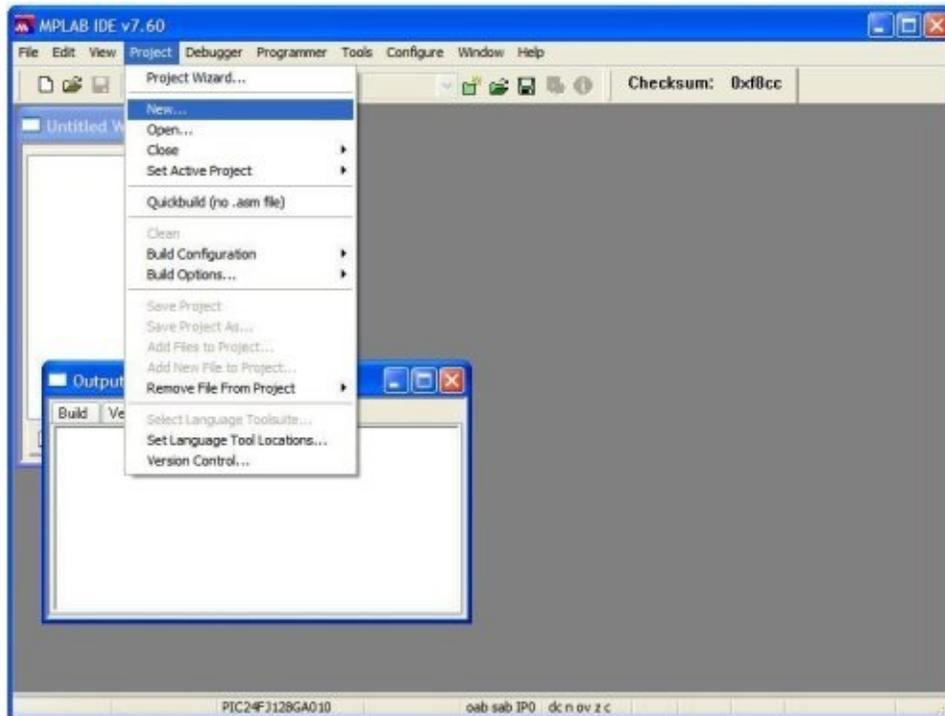
```
<MyProjects>
|
Board Support Package
Graphics
GraphicsProjects1
Microchip
|
Help
Graphics
|
Drivers
Include
|
Graphics
```

- MyProjects can be any name and is the directory name that you chose when you installed the Microchip Application Libraries.
- Graphics directory will contain demos distributed with the Graphics Library.
- Board Support Package directory will contain drivers for components on Microchip Development Boards that are used by the demos.
- GraphicsProject1 directory can be any directory name that you specify later for your own project.

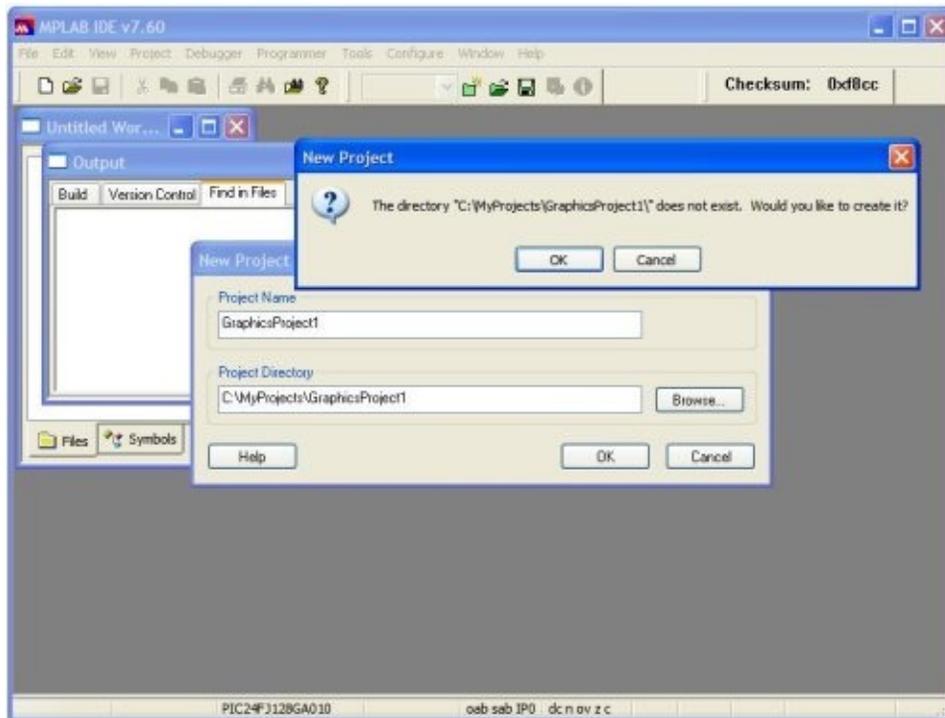
- Microchip directory will contain all the library files.
- Under Microchip directory, the Help directory will contain the "Graphics.chm" file.
- Under Microchip directory, the Graphics directory under the Microchip directory will contain all the source (C) files for the Graphics Library except the actual display driver files. Under the Drivers directory, all the supported driver files released with the library is located.
- Under Microchip directory, the Include directory will have another Graphics directory that will contain all the header files for the Graphics Library. All other header files will be located under Microchip/Include directory.

## Step 2: Creating the MPLAB project

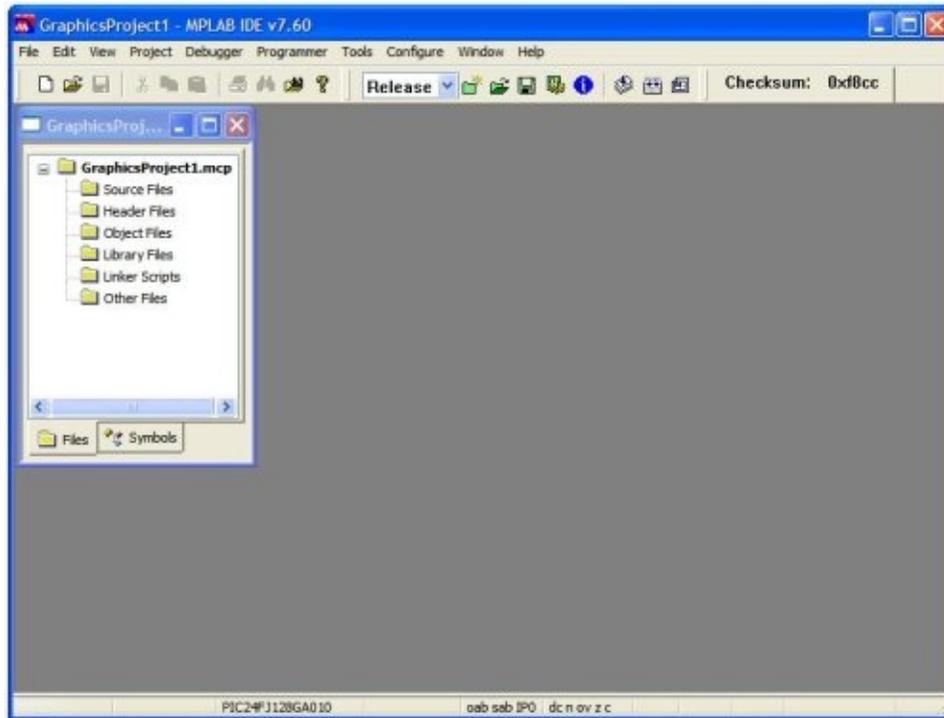
1. Open MPLAB Project.
2. Go to **Project** tab and select **New**



3. **New Project** dialog box will appear. In this dialog box enter your **Project Name** and **Project Directory**. Click **OK**. Project will be created or prompted to create the directory if your project directory does not exist (refer to figure below).



4. The MPLAB **Project** view will appear as soon as the project is created. If it does not appear go to **View** tab and click **Project**.



- Right click on the **Source Files** and click **Add Files**. Browse to *MyApplication->Microchip->Graphics* directory. Select appropriate c files and click **Open**. The C files that you select will depend on the application that you will be creating. The following tables shows the minimum files required for each layer of the Graphics Library.

<b>Display Driver Layer</b>	
DisplayDriver.h	Header that contains required APIs for display driver to be compatible with Microchip Graphics Library.
gfxepmp.c gfxepmp.h	Required if using display drivers for external display controller that will use the Enhanced Parallel Master Port (EPMP) for display controller communications.

gfxpmp.h	Required if using display drivers for external display controller that will use the Parallel Master Port (PMP) for display controller communications.
gfmtcon.h	Required if using Microchip supplied display drivers that requires timing controller initialization. If creating your own display controller driver, you may integrate the timing controller initialization into your display driver file.
Display Driver c and h files	These are the display driver files needed. These can be your own created display driver or Microchip supplied display drivers distributed with the Graphics Library.

<b>Primitive Layer</b>	
Primitive.c Primitive.h	These two files implements the Primitive Layer of the Microchip Graphics Library.
Transitions.c Transitions.h Transitions_weak.c	Implements the extended primitive functions for screen transitions. The Screen Transition features are only supported by the Epson driver (S1D13517.c and S1D13517.h) and the PIC24FJ256DA210 driver (mchpGfxDrv.c and mchpGfxDrv.h)

<b>Graphics Object Layer</b>	
GOL.c GOL.h	These two files along with the individual widget files implements the Graphics Object Layer of the Microchip Graphics Library.

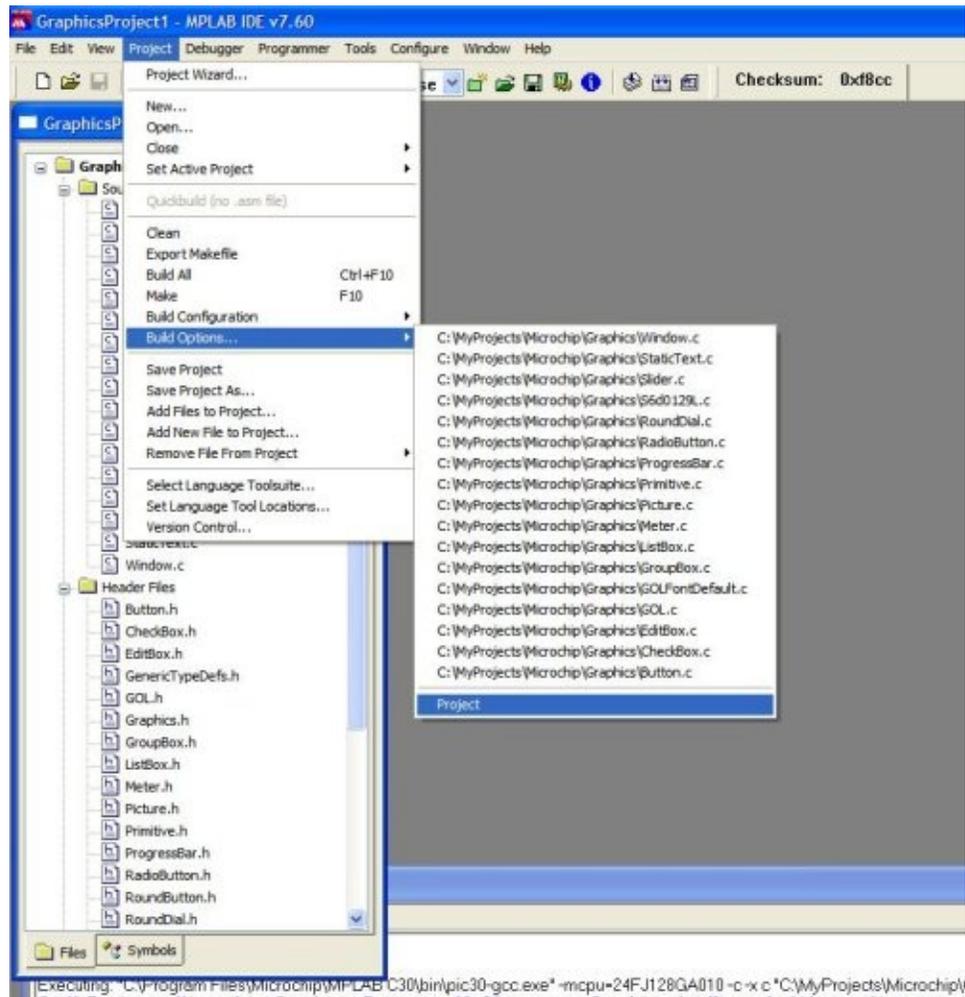
GOLFontDefault.c	This is the default font used for the Graphics Object Layer.
GOLSchemeDefault.c	This is the default style scheme used for the Graphics Object Layer.
Widget files (example: Button.c Button.h)	Required files when using the widgets in the Graphics Object Layer.

Configuration	
Graphics.h	Required file that loads the different components of the Graphics Library.
GraphicsConfig.h	Required file to determine the features and modes that are enabled in the Graphics Library.
HardwareProfile.h	Required file when running demos distributed with the Microchip Graphics Library or using drivers distributed in "Board Support Package".

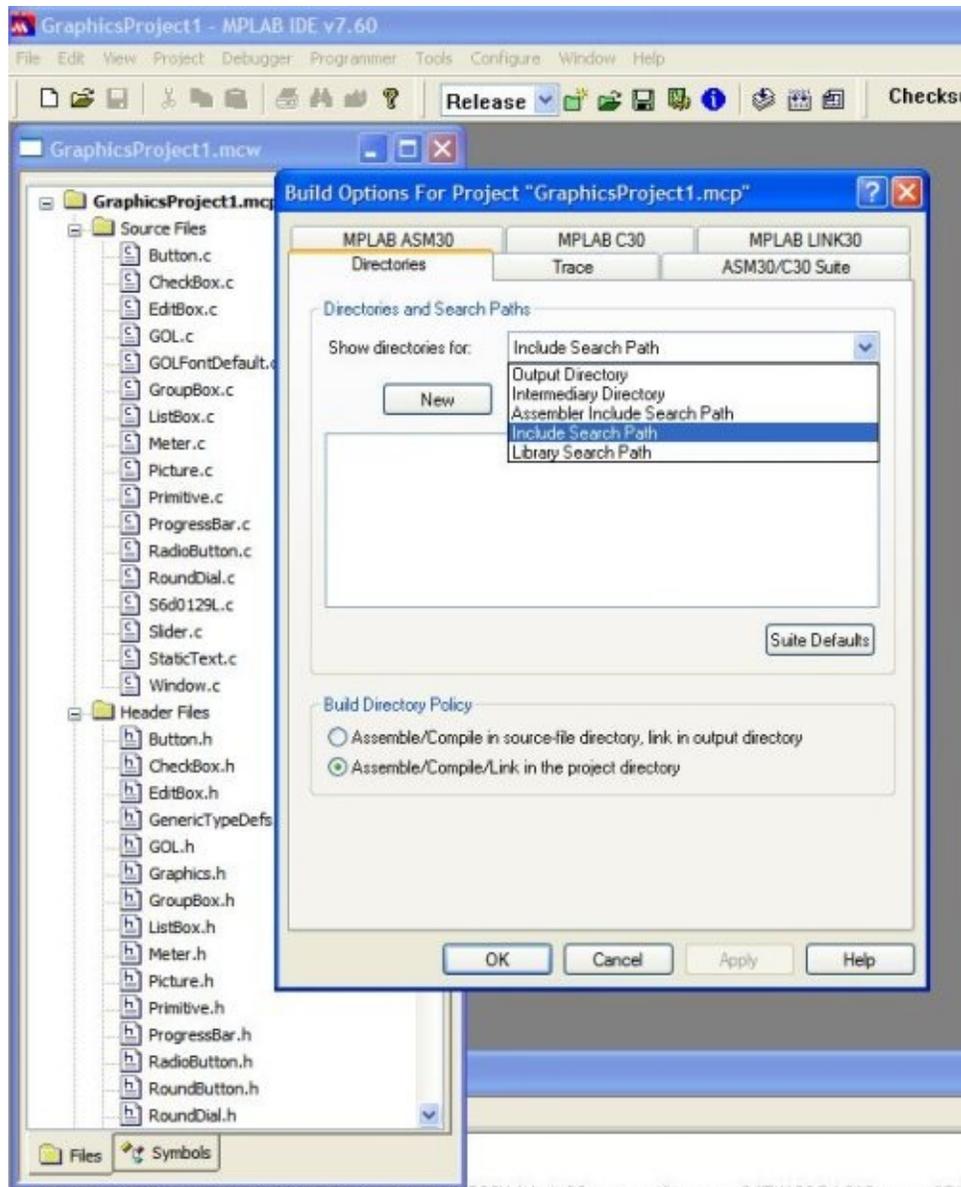
- Right click on the **Header Files** and click **Add Files**. Browse to *MyApplication->Microchip->Include* directory. Select **GenericTypeDefs.h** file and click **Open**.
- Right click on the **Header Files** and click **Add Files**. Browse to *MyApplication->Microchip->Include->Graphics* directory. Select all the h files and click **Open**. The header files to include will depend on the application as described in 5.

### Step 3: Setting the project directories.

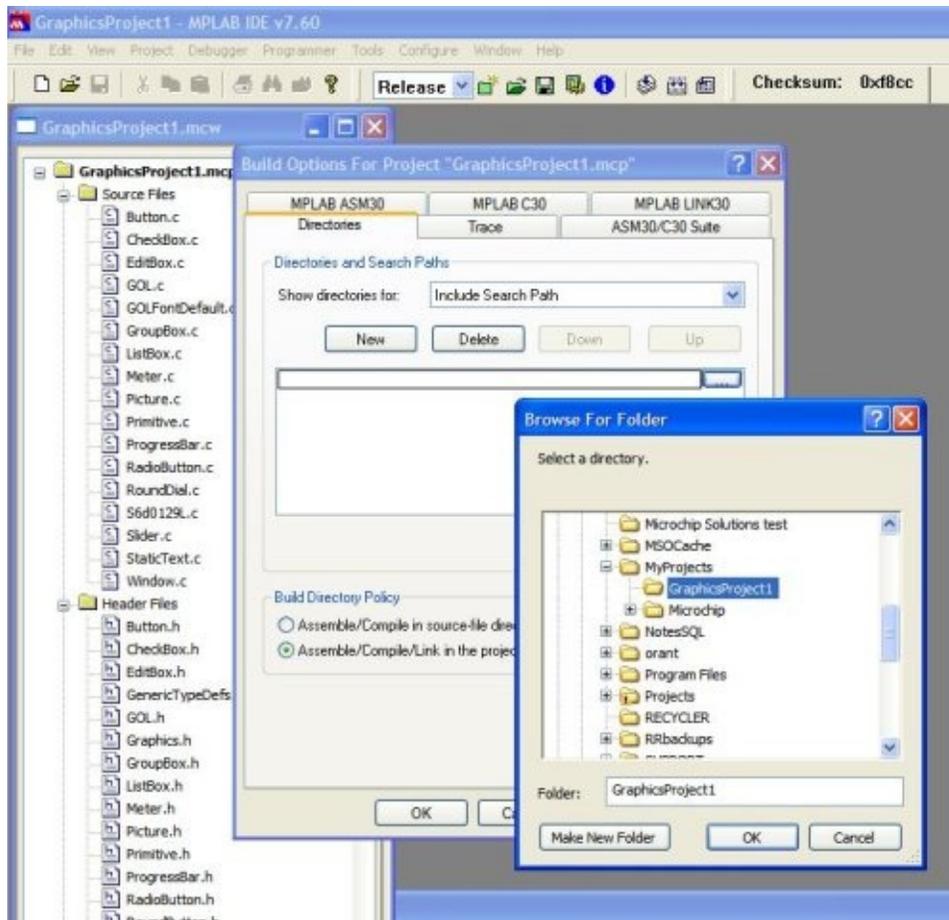
- In MPLAB Project, go to **Project** tab and point to **Build Options**.



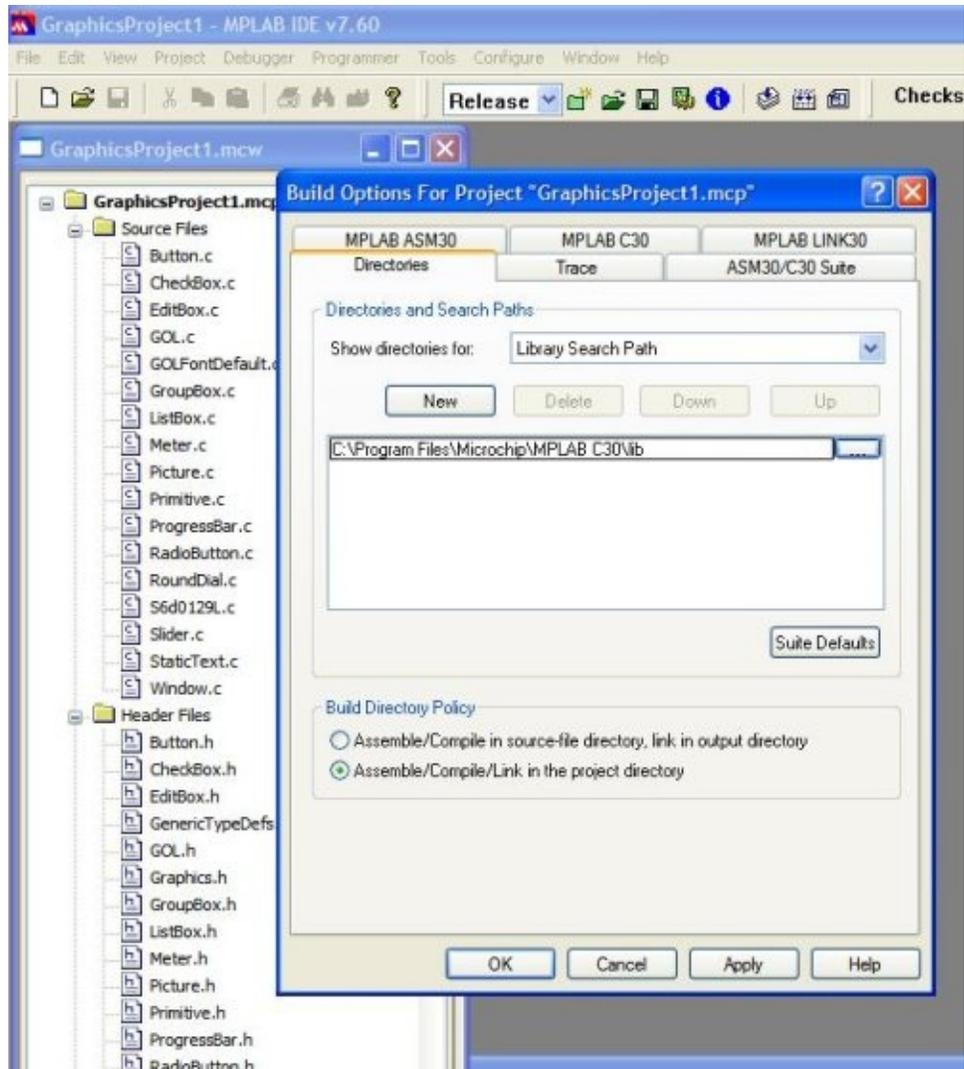
2. A menu will appear showing all your files and the **Project** option. Select the **Project** option. The **Build Options** window will open.
3. Select the **Directories** tab.



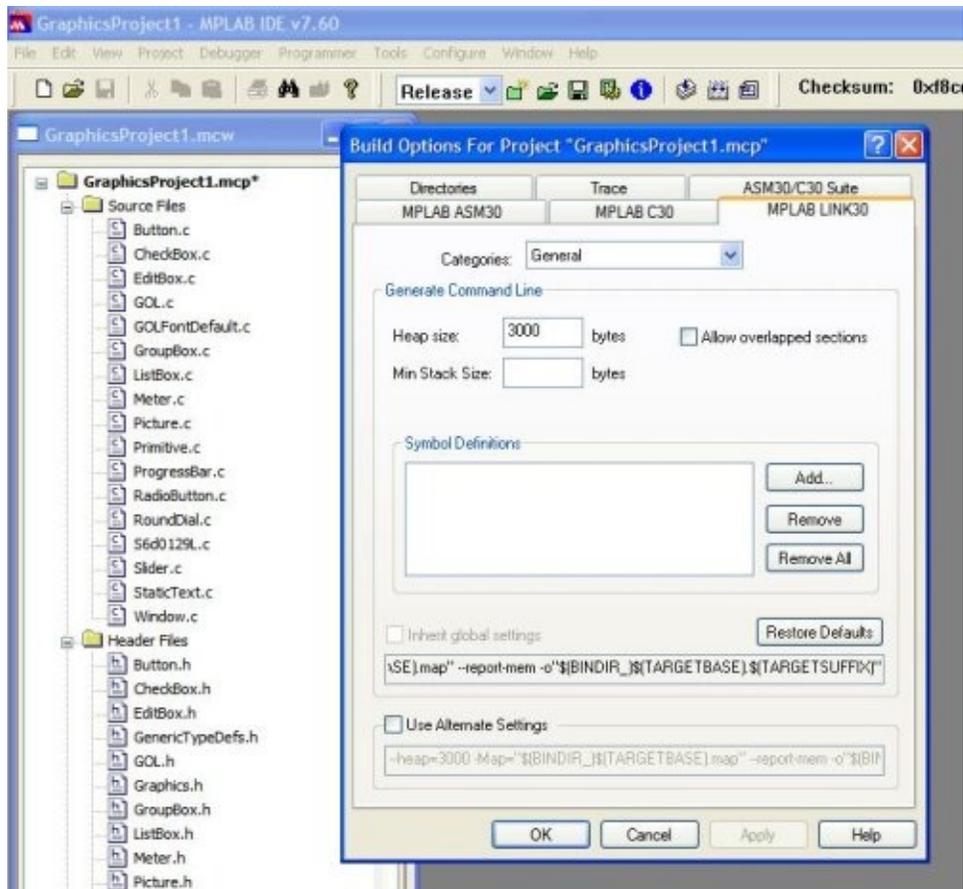
4. In the **Directories** tab, **Directories and Search Paths** group box and **Show directories for:** select **Include Search Path**. Click **New**.
5. Click the button with (...) and browse and select your project directory. Click **OK**. This will add your project directory.



6. Click New again for each of the following paths:
  1. .
  2. ..\..\Microchip\Include
  3. ..\..\Board Support Package (do this only if you are using the drivers supplied in the "Board Support Package" director\_)
7. **Directories and Search Paths** group box and **Show directories for:** select **Library Search Path**. Click **New**.
8. Enter the path to the MPLAB C30 lib. For example: C:\Program Files\Microchip\MPLAB C30\lib. Click **OK**.



9. Change from the **Directories** tab to **MPLAB LINK 30** tab. In the **Generate Command Line** group box enter 3000 in the **Heap Size** setting.



10. Click **Apply** and then **OK**.

**Step 4:** Create your application files.

1. In MPLAB Project, select **File** tab and click **New**. The file editor will open.
2. Create your **GraphicsConfig.h** file. For now add the following lines:

[Copy Code](#)

```
#define USE_NONBLOCKING_CONFIG // Comment this line
#define USE_BUTTON // Enable Button Obj
```

```

#define USE_SLIDER           // Enable Slider or
#define USE_FONT_FLASH      // Support for fonts
#define USE_BITMAP_FLASH    // Support for bitma

```

The file name is important. If the file name is changed, library will not compile properly.

3. Save the file into your **GraphicsProject1** directory.
4. Similarly, create your application header file code by following steps 1 and 2. For simplicity just use this code for now:

[Copy Code](#)

```

#define SYSCLK 32000000      // 8MHz x 4PLL 0scil
// includes
#include <p24Fxxx.h>
#include "GenericTypeDefs.h"
#include "Graphics.h"

```

5. Save the file in your **GraphicsProject1** directory.
6. After saving the two header files, add these files to your project.
7. Create your application code. For simplicity just use this code for now:

[Copy Code](#)

```

#include "GraphicsProject.h" // header file
// Configuration bits
_CONFIG2(FNOSC_PRIPLL & POSCMOD_XT) // Primary XT 0
_CONFIG1(JTAGEN_OFF & FWDTEN_OFF) // JTAG off, wa

int main(void){

    GOL_MSG msg; // GOL message structure to intera

```

[Copy Code](#)

```

GOLInit(); // initialize graphics library &

```

[Copy Code](#)

[Copy Code](#)

```
BtnCreate( 1, // object's ID
           20, 160, 150, 210, // object's dim
           0, // radius of th
           BTN_DRAW, // draw the obj
           NULL, // no bitmap us
           "LEFT", // use this tex
           NULL); // use alternat
```

[Copy Code](#)

[Copy Code](#)

```
BtnCreate( 2,
           170, 160, 300, 210,
           0,
           BTN_DRAW,
           NULL,
           "RIGHT",
           NULL);
```

[Copy Code](#)

[Copy Code](#)

```
SldCreate(3, // object's ID
           20, 105, 300, 150, // object's dim
           SLD_DRAW, // draw the obj
           100, // range
           5, // page
           50, // initial posi
           NULL); // use default
```

[Copy Code](#)

```
while(1){
    if (GOLDdraw()) { // Draw GOL obj
    }
}
```

```
    }  
}  
WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER* pObj,  
    return 1;  
}  
  
WORD GOLDDrawCallback(){  
    return 1;  
}
```

8. Save the file in your **GraphicsProject1** directory.
9. After saving the application file, add the file to your project.

**Step 5** Now build your project. To build go to MPLAB **Project** tab and click **Build All**. Notice there will be some warnings that may come out in the build log. This is due to the fact that we have not used most of the Objects. We only used the [Slider](#) and [Button](#).

**Step 6:** Downloading your application - To download your generated code to the Explorer 16 board follow the steps below.

1. In the MPLAB **Programmer** tab click **Select Programmer**.
2. Select **MPLAB ICD2** or **REAL ICE** depending on your setup.
3. After connection and testing is done go to **Programmer** tab and click **Program**.

Note that you can also do the steps in the Downloading the [Demos](#) sub-section in the [Getting Started](#) section to download your application to the Explorer 16 board using your generated hex file.

## Links

---

## Miscellaneous Topics

---

### Miscellaneous Topics > Starting a New Project

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Changing the default Font

This instructions assumes that you have performed the conversion of your raster font or True Type font into C file. For the conversion of your raster font please see the help file of the Graphics Resource Converter utility that comes with the Graphics Library.

There are two ways to replace the default font in the Graphics Library.

### Renaming User Font to **GOLFontDefault**:

1. Open the generated font file (generated by the "Graphics Resource Converter" utility) and change the font structure name to **GOLFontDefault** (see figure below).

```
extern const char L1191258975[] __attribute__((aligned(2)));  
//FONT STRUCTURE. FONT NAME CAN BE CHANGED HERE.  
const struct{short mem; const char* ptr;} GOLFontDefault =  
{0,L1191258975};  
const char L1191258975[] __attribute__((aligned(2))) = {
```

2. Remove the file **GOLFONTDefault.c** in the project and replace it with your own font file that contains the **GOLFontDefault**.
3. Build and test your new font.

### Replacing the **GOLFontDefault** with any user defined fonts

1. Open the **GraphicsConfig.h** file and add the following lines:

```
#define FONTDEFAULT yourFontName
```

[Copy Code](#)

2. In the project, add the generated font file (generated by the "Graphics Resource Converter" utility).
3. Build and test your new font.

In this method, GOL.h and GOLFontDefault.c files checks if [FONTDEFAULT](#) is defined. If it is, it skips the declaration of the [GOLFontDefault](#) and uses the user defined font.

## Links

---

### [Miscellaneous Topics](#)

---

### [Miscellaneous Topics](#) > [Changing the default Font](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Advanced Font Features

## Font Anti-Aliasing

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 1.

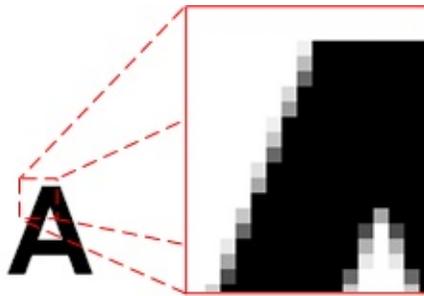


Figure 1: Font with Anti-Aliasing

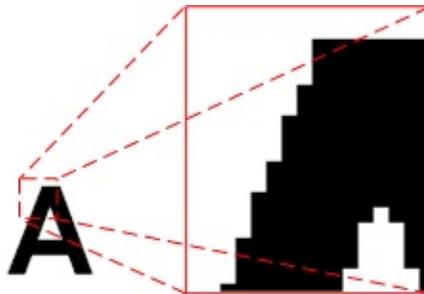


Figure 2: Font with No Anti-Aliasing

When anti-aliasing is turned off, the pixels abruptly changes from background color to foreground color shown in Figure 2. 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. To optimize the rendering speed, a macro named GFX\_Font\_SetAntiAliasType() is available where anti-alias type can be set to ANTIALIAS\_OPAQUE or ANTIALIAS\_TRANSLUCENT.

- ANTIALIAS\_OPAQUE (default after initialization of graphics) - mid colors are calculated once while rendering each character which is ideal for rendering text over a constant background.
- ANTIALIAS\_TRANSLUCENT - the mid values are calculated for every necessary pixel and this feature is useful while rendering text over an image or on a non-constant color background.

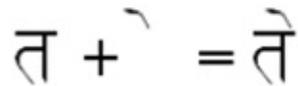
As a result, rendering anti-aliased text takes longer with ANTIALIAS\_TRANSLUCENT type than compared to ANTIALIAS\_OPAQUE type.

To use anti-aliasing, enable the compiler switch `#define USE_ANTIALIASED_FONTS` in the GraphicsConfig.h file and enable the anti-alias checkbox in the Graphics Resource Converter (GRC) tool while selecting the font.

**Note:** Even when anti-aliasing is enabled, normal fonts can be used without the antialias effect.

## Extended Glyphs

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 Figure 3. To use this feature, enable the Extended Glyph checkbox in the Graphics Resource Converter (GRC) tool while selecting the font.



त + े = ते

Figure 3: Example of a Character that is Formed by Two Overlapping Glyphs

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

## Links

---

[Miscellaneous Topics](#)

---

[Miscellaneous Topics](#) > [Advanced Font Features](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Using Primitive Rendering Functions in Blocking and Non-Blocking Modes

All primitive rendering functions returns a status.

- 0 – when the primitive was not successfully rendered
- 1 – when the primitive was successfully rendered

When using Graphics Library you can enable the non-blocking mode when calling drawing/rendering functions. This is done by adding this line in your GraphicsConfig.h file:

```
Copy Code  
#define USE_NONBLOCKING_CONFIG // Comment this line
```

When using a display controller with hardware accelerated primitives (like SSD1926 which is on the Graphics PICtail™ 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 directly calling 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: `USE_NONBLOCKING_CONFIG` disabled

```
Copy Code  
// all primitives are blocking calls  
Line(a,b);  
Rectangle(c,d,e,f);  
Bar(c+2, d+2, e-2, f-2)
```

Case 2: `USE_NONBLOCKING_CONFIG` enabled

[Copy Code](#)

```
// all primitives are non-blocking calls
while(!Line(a,b));
while(!Rectangle(c,d,e,f));
while(!Bar(c+2, d+2, e-2, f-2));
```

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:

[Copy Code](#)

```
WORD DrawMyFigure(a, b, c, d, e, f)
{
    typedef enum {
        DRAW_LINE,
        DRAW_RECT,
        DRAW_BAR,
    } DRAW_MYFIGURE_STATES;
    static DRAW_MYFIGURE_STATES state = DRAW_LINE;

    if(IsDeviceBusy()) // checks if the hardware is
        return 0;

    switch(state){
        case DRAW_LINE:
            if (!Line(a, b))
                return 0;
            state = DRAW_RECT;
        case DRAW_RECT:
            if(!Rectangle(c,d,e,f))
                return 0;
            state = DRAW_BAR;
```

```
    case DRAW_BAR:
        if(!Bar(c+2, d+2, e-2, f-2));
            return 0;
        state = DRAW_LINE;
        return 1;
    }
}
```

This non-blocking code can be used in the application and the application can do other tasks whenever DrawMyFigure() returns 0. Application should call DrawMyFigure() again until it return a 1 signifying that the Line, [Rectangle](#) and [Bar](#) were drawn successfully.

## Links

---

### [Miscellaneous Topics](#)

---

### [Miscellaneous Topics > Using Primitive Rendering Functions in Blocking and Non-Blocking Modes](#)

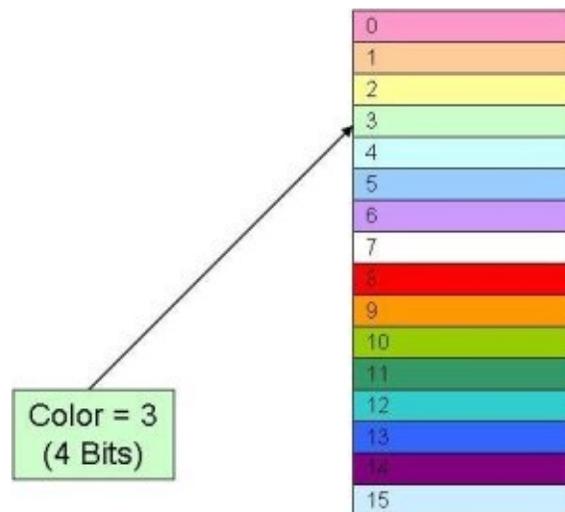
---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Using Microchip Graphics Module Color Look Up Table in Applications

Apart from regular RGB scheme of representing colors, colors may also be represented using a Color Look Up Table (CLUT) also called Palette table, where there is a table of colors and the color is specified by the index of the table. Depending on the size of the table, the bits used to represent the index will vary. For example 256 entries of RGB (8 bit index), 16 entries of RGB (4 bit index), 4 entries of RGB (2 bit index) and 2 entries of RGB (1 bit index). This scheme is mainly used to save memory. See Figure-1 for an example of 16-entry (4-bit) CLUT where a shade of Green is represented by an index value of 3 consuming 4-bits. To figure the memory requirement for a give screen size at a color depth, bits per pixel, the follow equations is used: total number of pixels x bits per pixel / 8 bits per byte. For example the memory required for a 320x240 with 16 BPP screen;  $(320 \times 240) \times (16 / 8) = 153,600$  Bytes. If only 16 different colors are used in the screen, then instead of using raw RGB, a 16-entry (4-bit) CLUT may be used requiring a memory of  $(320 \times 240) \times (4/8) = 38,400$  bytes; thereby saving 75% of memory.



## Figure 1. A 16-Entry (4-bit) Color Look Up Table

If the display driver hardware supports CLUT, the index values are translated to the RGB values by the hardware automatically when the signals are sent out to the display.

If the CLUT is enabled, since a CLUT affects the whole screen, all the color patterns like basic shapes and images which are displayed on the screen must use the same CLUT. It means that color defines like **RED**, **GREEN**, etc... must use the index values instead of the absolute RGB values. The bitmap images used must also use the same CLUT in order to appear properly on the screen. Note that the chosen CLUT length (1, 4, 16, or 256) must accommodate all the different colors needed by a screen. The following section explains how to create such a CLUT for a screen.

### Creating CLUT

Creating CLUT has 2 steps:

1. Creating a part of CLUT manually.
2. Filling the remaining part with the colors of the images used.
3. Using the Graphics Resource Converter to generate a CLUT based on the images to convert.

***Creating a part of CLUT manually*** - Since users need to select specific colors for the shapes and the widgets, they must enter these specific colors in the CLUT. This can be done by manually creating a new CLUT table using a text editor. Create a text file and save it with a .gpl extension with the form:

```
GIMP Palette
```

[Copy Code](#)

```

Name: 16_colors
Columns: 4
#
0 0 0 BLACK
0 0 128 BLUE
128 0 0 RED
0 0 0 Unused
0 0 0 Unused

```

The second line specifies the name of the palette which must be unique. The palette table data starts with line 5 which represent index 0 with the RGB values and a caption. The number of such data lines must be equal to the length of the CLUT. In the example, only 3 colors are used.

### ***Filling the remaining part with the colors of the images used***

- Suppose 256 entries CLUT is being used, since 3 colors are manually set, only  $256 - 3 = 253$  entries are available for the images to be displayed on the same screen. If more than one image is used on the screen, it further implies that

- All images on the screen must use the same CLUT table.
- Different colors used for all the images along with the fixed colors must not exceed the size of the CLUT table being used.

If the source images are of RGB type, they must be converted into CLUT based images. This can be done using free PC tools like GIMP ([www.gimp.org](http://www.gimp.org)).

The following steps shows how to convert the images using GIMP:

#### **Step 1:**

Create a new image with sufficient size to paste all the images required on the screen using *File->New* in the GIMP (see Figure-2).

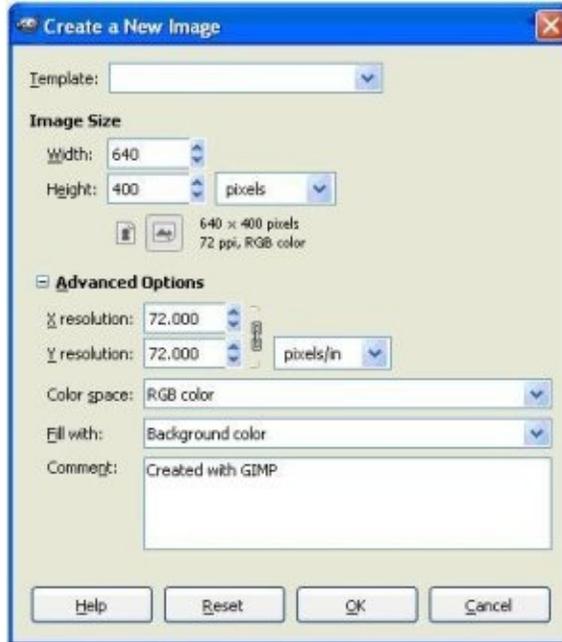


Figure 2. Creating New File in GIMP

## Step 2:

Copy and paste all the images to be displayed on the screen into this image like in Figure 3. If 256 colors is enough for the entire application, a single CLUT can be used. If not multiple CLUT can be used. Each CLUT can be configured to use one or more screens. In this case, switching from one screen to another may require re-initializing the hardware CLUT entries before the screen is displayed.



Figure 3. Images Used in One Screen

### Step 3:

Set the mode to CLUT by selecting *Image->Mode->Indexed* in the GIMP. Select to generate optimum palette with  $256 - 3 = 253$  entries (because we already have 3 fixed entries) as shown in Figure 4 and save it as a BMP (e.g. Collage.bmp) image by selecting *File->SaveAs* menu.

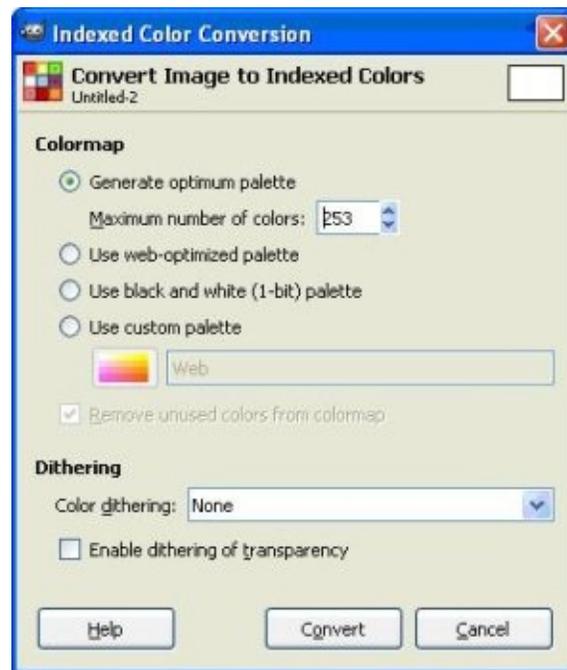


Figure 4. Generate a Palette Table (CLUT)

### Step 4:

The next step is to extract the CLUT from the generated image. To do that, go to the palette selection mode by selecting *Image->Mode->RGB* and then again *Image->Mode->Indexed*. Select Use Custom Palette and open the palette selection dialog as shown in Figure 5 and import the previously saved bitmap image (Collage.bmp) as shown in Figure 6 and Figure 7. The palette file will be created in the [Home Folder]\.gimp-x.y\palettes folder as a

\*.gpl file.

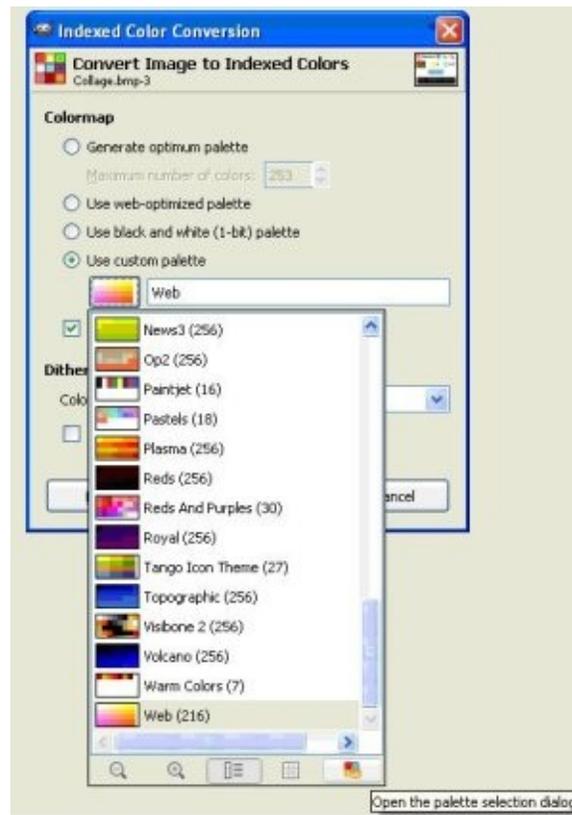


Figure 5. Palette Selection Dialog



Figure 6. Import Palette Dialog

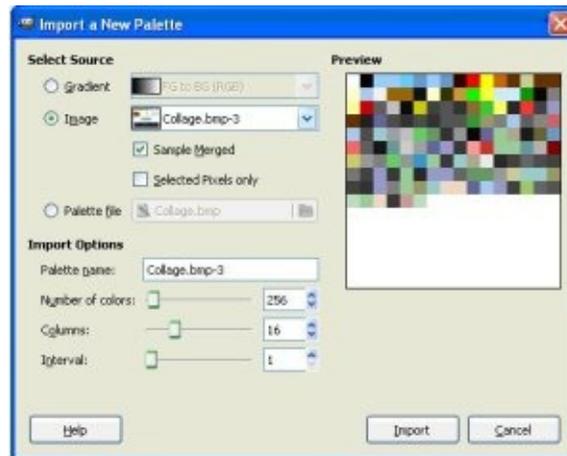


Figure 7. Import Collage Bitmap

### Step 5:

Manually edit this and add the fixed color entries from the

previously stored fixed palette file. (Manual step). Now a master palette file is created which has to be used in the application.

### Step 6:

Open individual images in GIMP and apply this master palette to all of them through *Image->Mode->Indexed* and selecting Use Custom Palette. Select the master palette which was generated and save the images. This will make all the images palette ready.

### Step 7:

The next step is to convert this Palette.gpl and the images into the format recognizable by the Microchip Graphics Library. Start the Microchip's Graphics Resource Converter tool and enable the C30 Build (palette support is currently on selected PIC24F devices only) mode as shown in Figure 8. Open the master palette file previously generated by pressing Add Palette button and save it as a .c file (for storing in internal flash) or as a .hex file (for storing in external memory) similar to bitmap or font conversion as shown in Figure 10.

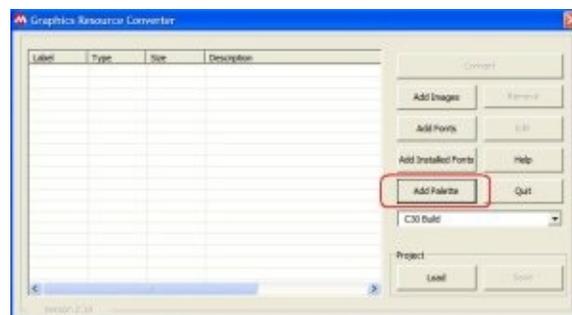


Figure 8. Load Palette

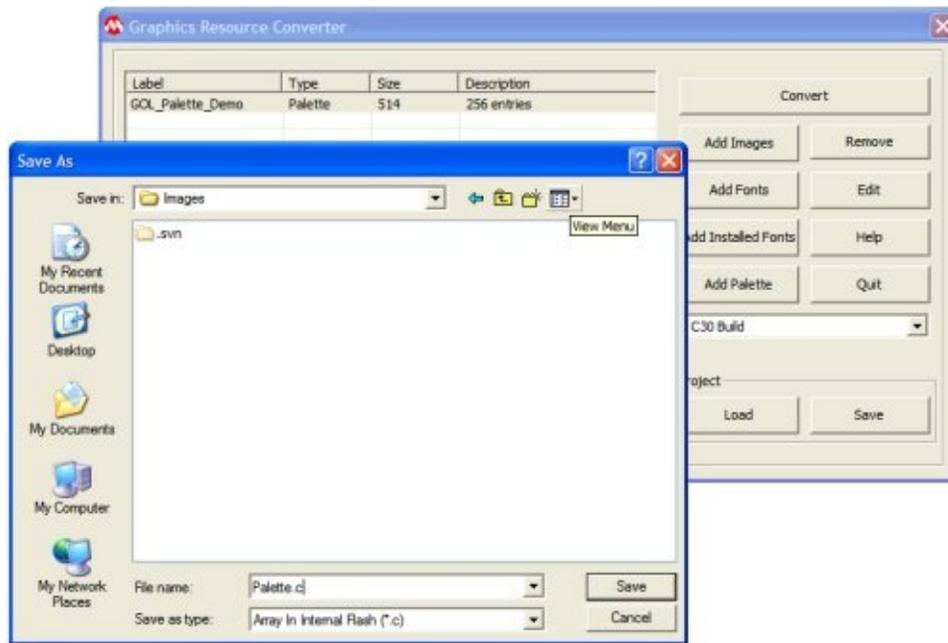


Figure 9. Convert Palette

### Step 8:

Convert all the palette ready images to C file (\*.c) or Hex file (\*.hex) by pressing the Add Images button.

### Step 9:

Import the palette with the extern statement like in “extern const **PALETTE\_FLASH** \_GOL\_Palette\_Demo;” in the application. See MainDemo.c in “Graphics Object Layer Palette Demo”. Set the palette using the APIs **SetPaletteBpp()** and **SetPalette()** and then enable the palette using the API **EnablePalette()** as shown in the below code example.

[Copy Code](#)

```
GOLInit();  
SetPaletteBpp(8);  
SetPalette((void*)&_GOL_Palette_Demo, 0, 256);  
EnablePalette();
```

## Step 10:

Import the images as usual and use them after setting and enabling the palette.

See “Graphics Object Layer Palette Demo” as a practical example.

***Using the Graphics Resource Converter to generate a CLUT based on the images to convert*** - The Graphics Resource Converter will generate a CLUT based on the images to be converted. Please refer to the Graphics Resource Converter help file for more information.

## Links

### [Miscellaneous Topics](#)

[Miscellaneous Topics](#) > [Using Microchip Graphics Module Color Look Up Table in Applications](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Converting Images to Use a Common Palette in GIMP

## INTRODUCTION

Some controllers have predefined palettes associated with them. The palette's colors can not be altered. When converting images, it is desired to have the images use the controller's predefined color palette. For example, a controller may have a grayscale palette of 16 colors. An image may use a palette of 16 grayscale colors, but the palette may define grayscale colors that are not the same as the controller's palette. By using the GNU Image Manipulation Program, GIMP, images can be converted using a palette matching the grayscale colors of the controller. After converting the image to use this palette, it can be converted by the Graphics Resource Converter, GRC, to be used by Microchip's Graphics Library.

## DOWNLOADS

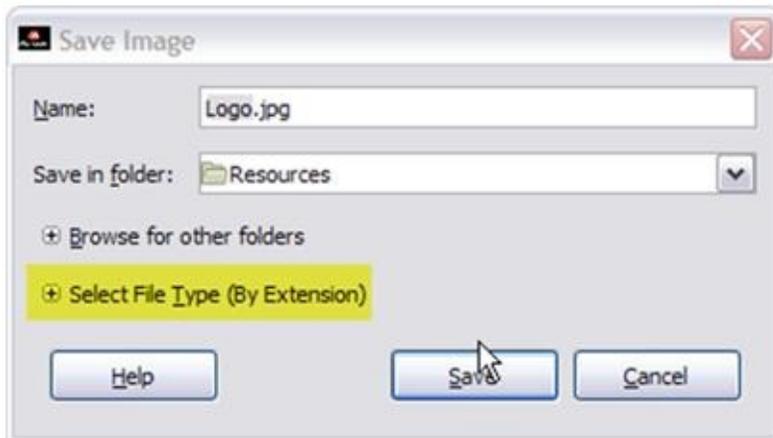
The following are helpful websites for downloading the tools and firmware needed:

- GIMP 2.6 – GNU Image Manipulation Program ([↗ www.gimp.org](http://www.gimp.org))
- Graphics Library – This library is part of the Microchip Application Libraries ([↗ www.microchip.com/mla](http://www.microchip.com/mla))

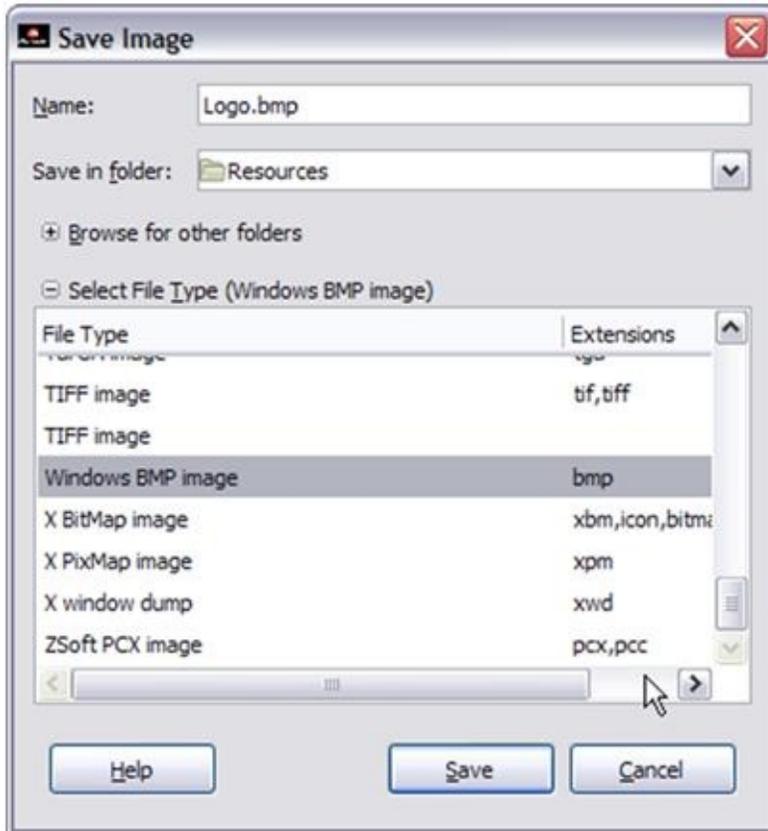
## CONVERTING AN IMAGE

Follow these steps to convert an image to use a predefined color palette (for this example, the palette used has 16 grayscale colors):

1. Open the GIMP application.
2. Load the image. FILE->Open
  - If the image is not a Bitmap, you will need to save it as one.
    1. FILE-Save as...
    2. Choose the Select File Type (By Extension)



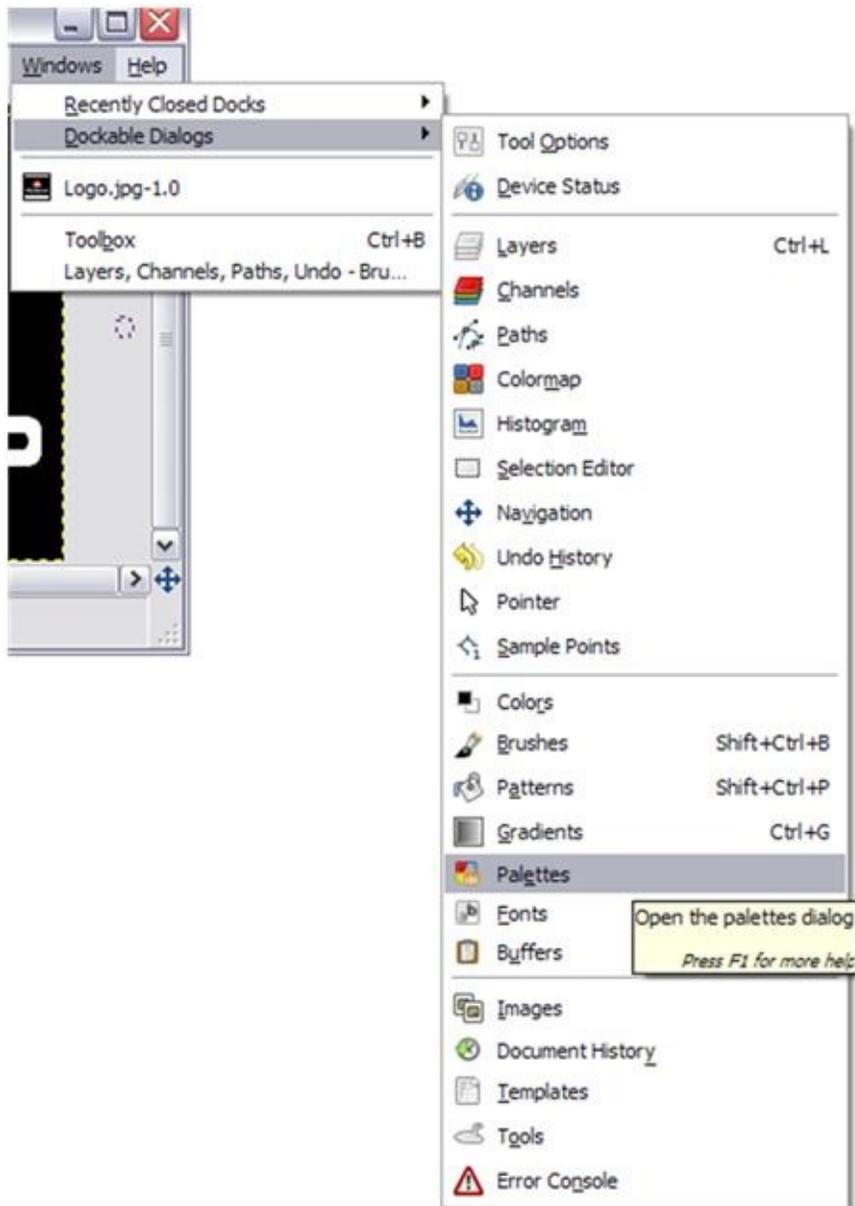
3. Select Windows BMP Image



4. Select Save
5. Select Save under the Save as BMP dialog



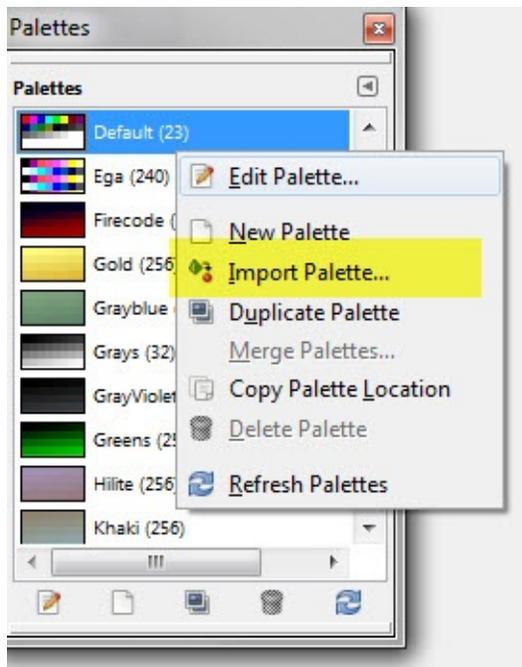
3. Select WINDOWS->Dockable Dialogs->Palettes



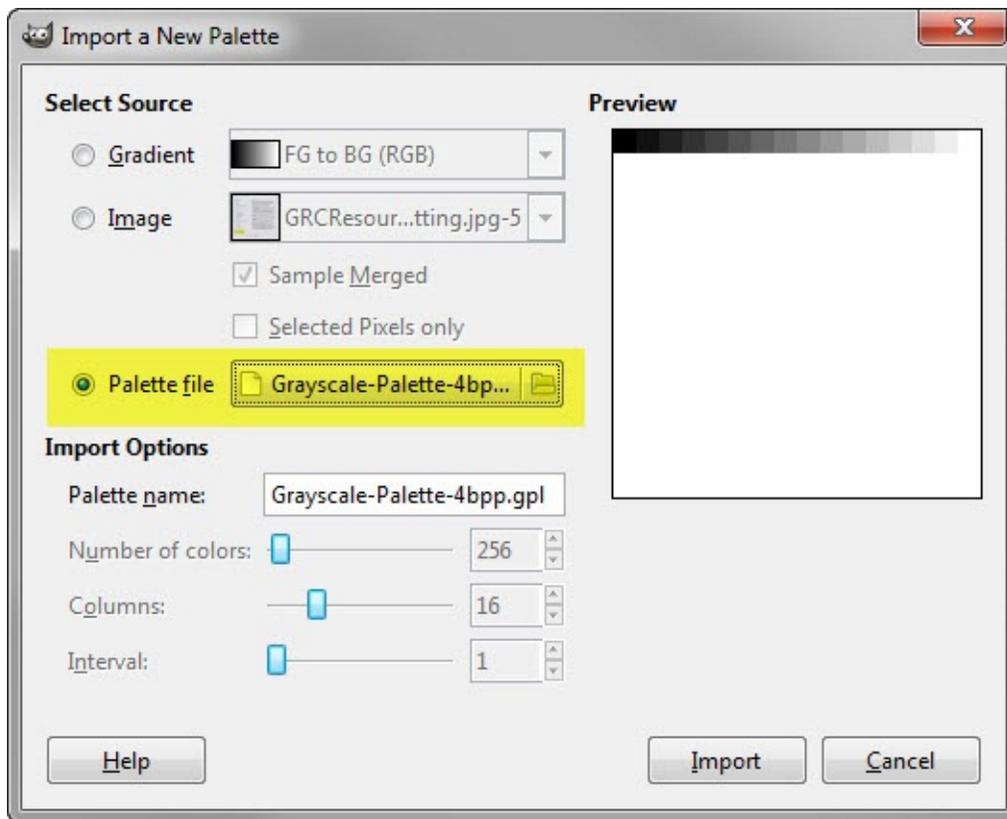
- The Palette Dialog will appear.



4. In the Palette Dialog Select Import Palette. This can be done by a right click on any of the Palettes on the Palette Dialog.



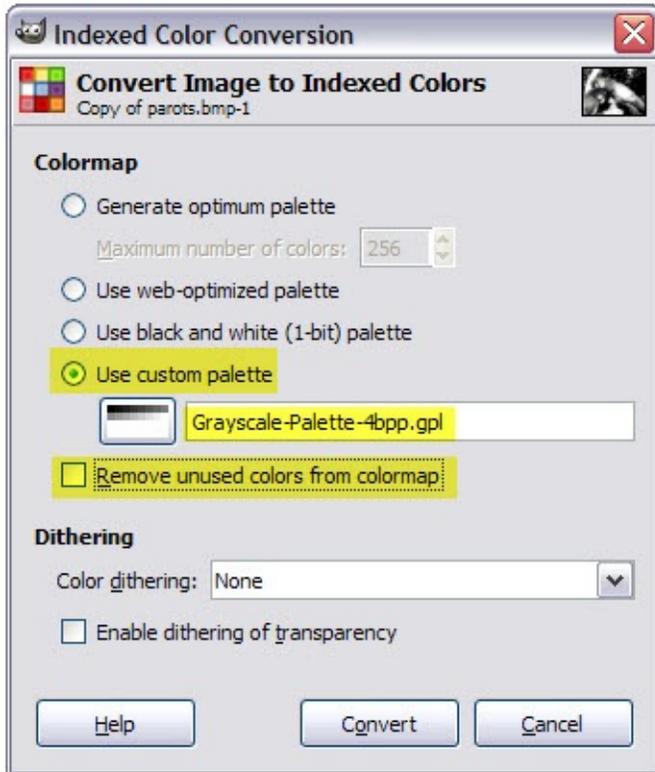
- The Import Palette Dialog will appear.



5. Select the Palette file radio button.
  - Select the palette file .gpl (For example: Grayscale-Palette-4bpp.gpl) (see image in Step 4).
6. Select the Import button.
  - The palette, Grayscale-Palette-4bpp, will show up in the Palette Dialog.
7. Select IMAGE->Mode->Indexed...



8. In the Indexed Color Conversion dialog, select the Use custom palette radio button
  - o Type the name of the palette.
  - o **IMPORTANT:** Make sure that the "Remove unused color from colormap" is unchecked.
  - o Select Convert



9. The image will now be converted using a 16 grayscale color palette.
10. Save the image as a Bitmap.

This image can now be converted by the GRC for use with the Microchip Graphics Library.

Here is an example of a palette file \*.gpl (Grayscale-Palette-4bpp.gpl)

[Copy Code](#)

```
GIMP Palette
Name: Greyscale Palette 4bpp
Columns: 0
#
0 0 0 BLACK
```

17	17	17	Untitled
34	34	34	Untitled
51	51	51	Untitled
68	68	68	Untitled
85	85	85	Untitled
102	102	102	Untitled
119	119	119	Untitled
136	136	136	Untitled
153	153	153	Untitled
170	170	170	Untitled
187	187	187	Untitled
204	204	204	Untitled
221	221	221	Untitled
238	238	238	Untitled
255	255	255	WHITE

## Links

---

### [Miscellaneous Topics](#)

[Miscellaneous Topics](#) > [Converting Images to Use a Common Palette in GIMP](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## How to Define Colors in your Applications

---

To override or define a new set of colors follow this steps:

1. Create a new color header file. The color values and data types will depend on the GFX\_COLOR type used. This data type is defined by the [COLOR\\_DEPTH](#) macro. See [COLOR\\_DEPTH](#) for details.
2. In the application code, include the created color header file ahead of the `#include "Graphics/Graphics.h"`. When Graphics.h includes the gfxcolors.h it will ignore color macros that has been defined already in the new color header file.

In the GOL Palette Demo, there is an example on how it is done. In that project there is a file in the application (or project directory) named PaletteColorDefines.h. This file contains all the color definition used in the demo. The main header file of the demo (Main.h) includes the PaletteColorDefines.h file ahead of the Graphics Library header files. Since the PaletteColorDefines.h declared the color values, the macros redefined in gfxcolors.h will be ignored. How is this done? If you look at the gfxcolors.h file you will notice that all colors defined have a check (for example [BLACK](#)):

```
#ifndef BLACK  
  
#define BLACK 0  
  
#endif
```

So if [BLACK](#) is defined previously, the definition in gfxcolors.h will not take effect.

### Links

---

[Miscellaneous Topics](#)

---

[Miscellaneous Topics](#) > [How to Define Colors in your Applications](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Connecting RGB data bus

To connect the 16-bit RGB data bus to a 24-bit or 18-bit RGB data bus of a display glass follow the recommended connection to evenly spread the color coverage.

16-bit (5-6-5) RGB Display Controller Data Bus	18-bit (6-6-6) RGB Display Data Bus	24-bit (8-8-8) RGB Display Data Bus
<b><i>Red[4:0]</i></b>		
Controller Red[4:0]	Display Red[5:1]	Display Red[7:3]
Controller Red[4]	Display Red[0]	Display Red[2:0]
<b><i>Green[4:0]</i></b>		
Controller Green[5:0]	Display Green[5:0]	Display Green[7:2]
Controller Green[5]	Display Green[5]	Display Green[1:0]
<b><i>Blue[4:0]</i></b>		
Controller Blue[4:0]	Display Blue[5:1]	Display Blue[7:3]
Controller Blue[4]	Display Blue[0]	Display Blue[2:0]

To illustrate the connection take the Red for example:

***To connect the Display Controller Red Data Bus to the Red data bus of a RGB Glass with 18 bit color data bus.***

Connect the 5 Red signals from the display controller to the most significant bits of the glass red signals.

Controller Red[4:0] -> Display Red[5:1]

The remaining Display Red[0] signal will be connected to the most significant bit of the display controller red.

Controller Red[4] -> Display Red[0]

***To connect the Display Controller Red Data Bus to the Red data bus of a RGB Glass with 24 bit color data bus.***

Connect the 5 Red signals from the display controller to the most significant bits of the glass red signals.

Controller Red[4:0] -> Display Red[7:3]

The remaining Display Red[2:0] signals will be connected to the most significant bit of the display controller red.

Controller Red[4] -> Display Red[2]

Controller Red[4] -> Display Red[1]

Controller Red[4] -> Display Red[0]

Doing this spreads out the color coverage while at the same time pure black and pure white colors are achieved.

## Links

---

### [Miscellaneous Topics](#)

---

### [Miscellaneous Topics](#) > [Connecting RGB data bus](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Adding New Device Driver

Adding a new display device driver requires the following functions and macros to be implemented. Please refer to the API section of the Device Driver Layer for details.

Function/Macro	Description
<a href="#"><u>ResetDevice()</u></a>	Initializes the display device.
<a href="#"><u>GetMaxX()</u></a>	Returns the maximum x-coordinate for the display.
<a href="#"><u>GetMaxY()</u></a>	Returns the maximum y-coordinate for the display.
<a href="#"><u>SetColor()</u></a>	Sets the current drawing color.
<a href="#"><u>GetColor()</u></a>	Returns the current drawing color.
<a href="#"><u>SetActivePage()</u></a>	Sets the current active graphic page (optional).
<a href="#"><u>SetVisualPage()</u></a>	Sets the current visual graphic page (optional).
<a href="#"><u>PutPixel()</u></a>	Modifies pixel on the screen.
<a href="#"><u>GetPixel()</u></a>	Returns the pixel color.
<a href="#"><u>PutImage()</u></a>	Renders an image on the screen. This function is dependent on the color format used.
<a href="#"><u>SetClipRgn()</u></a>	Set the current clipping region borders.
<a href="#"><u>GetClipLeft()</u></a> , <a href="#"><u>GetClipTop()</u></a> , <a href="#"><u>GetClipRight()</u></a> , <a href="#"><u>GetClipBottom()</u></a>	Returns the left, top, right and bottom clipping borders.
<a href="#"><u>SetClip()</u></a>	Enables or disables the clipping region.
<a href="#"><u>IsDeviceBusy()</u></a>	Checks if the display controller is busy executing

	the previous rendering operation.
<a href="#">SetPalette()</a>	Sets the palette register of the device.

## Adding new Display Device Drivers

The DisplayDriver.h file should be used as a guide to make your new driver compatible with the Microchip Graphics Library. All the API's defined in this header file are required functions to be implemented in the driver. There are portions of that file that states optional functions. These functions are not needed to interface to the Graphics Library. These are only implemented if the display controller used has hardware features that can implement these functions.

The best way to implement this is to try to find the nearest existing driver and modify the C and H files. Most graphics controllers has a lot of control registers to be initialized. Values programmed into the registers depends on the specification of the LCD glass used. If LCD module has a built-in graphics controller, initialization code for the glass can be found in the LCD specifications or get this information from the manufacturer.

## Links

---

### [Miscellaneous Topics](#)

---

### [Miscellaneous Topics](#) > [Adding New Device Driver](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## References

---

1. "[↗ MPLAB C32 C COMPILER USER'S GUIDE](#)" (DS51686), Microchip Technology Incorporated.
2. "[↗ MPLAB C COMPILER FOR PIC24 MCUs AND dsPIC DSCs USER'S GUIDE](#)" (DS51284), Microchip Technology Incorporated.
3. Microchip Application Note [↗ AN1136](#), "How to Use Widgets in Microchip Graphics Library" (DS01136), Microchip Technology Incorporated.
4. Microchip Application Note [↗ AN1182](#), "Fonts in the Microchip Graphics Library" (DS01182), Microchip Technology Incorporated.
5. Microchip Application Note [↗ AN1227](#), "Using a Keyboard with the Microchip Graphics Library" (DS01227), Microchip Technology Incorporated.
6. Microchip Application Note [↗ AN1246](#), "How to Create Widgets in Microchip Graphics Library" (DS01246), Microchip Technology Incorporated.
7. HIF 2131 – Designing with Microchip Graphics Library, Microchip Regional Training Center web site ([↗ www.microchip.com/rtc](#)).

---

## References

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Contents

This is the table of contents of this documentation.

Introduction

Release Notes

Getting Started

### **Demo Projects**

Demo Summary

Microchip Application Library Abbreviations

Demo Compatibility Matrix

### **Library Architecture**

#### **Graphics Object Layer**

Object Rendering

Graphics Primitive Layer

Display Device Driver Layer

### **Library API**

#### **Graphics Library Configuration**

##### **Graphics Object Layer Configuration**

##### **Input Device Selection**

USE\_KEYBOARD Macro

USE\_TOUCHSCREEN Macro

##### **Focus Support Selection**

USE\_FOCUS Macro

##### **Graphics Object Selection**

USE\_ANALOGCLOCK Macro

USE\_BUTTON Macro

USE\_BUTTON\_MULTI\_LINE Macro

USE\_CHECKBOX Macro

USE\_DIGITALMETER Macro

USE\_EDITBOX Macro

USE\_GROUPBOX Macro

USE\_LISTBOX Macro

USE\_METER Macro  
USE\_PICTURE Macro  
USE\_PROGRESSBAR Macro  
USE\_RADIOBUTTON Macro  
USE\_ROUNDIAL Macro  
USE\_SLIDER Macro  
USE\_STATICTEXT Macro  
USE\_WINDOW Macro  
USE\_CUSTOM Macro  
USE\_GOL Macro  
USE\_TEXTENTRY Macro

## **Graphics Primitive Layer Configuration**

### **Image Compression Option**

USE\_COMP\_IPU Macro  
USE\_COMP\_RLE Macro

### **Font Type Selection**

USE\_MULTIBYTECHAR Macro  
USE\_UNSIGNED\_XCHAR Macro

### **Advanced Font Features Selection**

USE\_ANTIALIASED\_FONTS Macro

### **Gradient Bar Rendering**

USE\_GRADIENT Macro

### **Transparent Color Feature in PutImage()**

USE\_TRANSPARENT\_COLOR Macro

### **Alpha Blend Option**

USE\_ALPHABLEND\_LITE Macro

External Memory Buffer

## **Display Device Driver Layer Configuration**

USE\_ALPHABLEND Macro  
USE\_DOUBLE\_BUFFERING Macro

### **GFX\_LCD\_TYPE Macro**

GFX\_LCD\_CSTN Macro

GFX\_LCD\_MSTN Macro

GFX\_LCD\_OFF Macro

GFX\_LCD\_TFT Macro

### **STN\_DISPLAY\_WIDTH Macro**

STN\_DISPLAY\_WIDTH\_16 Macro

STN\_DISPLAY\_WIDTH\_4 Macro

STN\_DISPLAY\_WIDTH\_8 Macro

## **Application Configuration**

### **Configuration Setting**

USE\_NONBLOCKING\_CONFIG Macro

### **Font Source Selection**

USE\_FONT\_FLASH Macro

USE\_FONT\_EXTERNAL Macro

USE\_GFX\_FONT\_IN\_PROGRAM\_SECTION Macro

### **Image Source Selection**

USE\_BITMAP\_FLASH Macro

USE\_BITMAP\_EXTERNAL Macro

### **Miscellaneous**

USE\_BITMAP\_NO\_PADDING\_LINE Macro

USE\_PALETTE\_EXTERNAL Macro

USE\_PALETTE Macro

COLOR\_DEPTH Macro

GFX\_free Macro

GFX\_malloc Macro

GraphicsConfig.h Example

## **Hardware Profile**

### **PMP Interface**

USE\_8BIT\_PMP Macro

USE\_16BIT\_PMP Macro

### **Development Platform Used**

EXPLORER\_16 Macro

PIC24FJ256DA210\_DEV\_BOARD Macro

MEB\_BOARD Macro

PIC\_SK Macro

### **Graphics PICtail Used**

GFX\_PICTAIL\_LCC Macro

GFX\_PICTAIL\_V3 Macro

GFX\_PICTAIL\_V3E Macro

### **Display Controller Used**

GFX\_USE\_DISPLAY\_CONTROLLER\_DMA Macro

GFX\_USE\_DISPLAY\_CONTROLLER\_MCHP\_DA210 Macro

GFX\_USE\_DISPLAY\_CONTROLLER\_S1D13517 Macro

GFX\_USE\_DISPLAY\_CONTROLLER\_SSD1926 Macro

### **Display Panel Used**

GFX\_USE\_DISPLAY\_PANEL\_PH480272T\_005\_I11Q Macro

GFX\_USE\_DISPLAY\_PANEL\_TFT\_640480\_8\_E Macro

GFX\_USE\_DISPLAY\_PANEL\_TFT\_800480\_33\_E Macro

GFX\_USE\_DISPLAY\_PANEL\_TFT\_G240320LTSW\_118W\_E M

### **Device Driver Options**

DISP\_DATA\_WIDTH Macro

DISP\_ORIENTATION Macro

DISP\_HOR\_RESOLUTION Macro

DISP\_VER\_RESOLUTION Macro

DISP\_HOR\_FRONT\_PORCH Macro

DISP\_HOR\_BACK\_PORCH Macro

DISP\_VER\_FRONT\_PORCH Macro

DISP\_VER\_BACK\_PORCH Macro

DISP\_HOR\_PULSE\_WIDTH Macro

DISP\_VER\_PULSE\_WIDTH Macro

DISP\_INV\_LSHIFT Macro

HardwareProfile.h Example

### **Graphics Object Layer API**

#### **GOL Objects**

GOL\_OBJ\_TYPE Enumeration

OBJ\_HEADER Structure  
DRAW\_FUNC Type  
FREE\_FUNC Type  
MSG\_DEFAULT\_FUNC Type  
MSG\_FUNC Type

## **Analog Clock**

### **Analog Clock States**

AC\_DRAW Macro  
AC\_DISABLED Macro  
AC\_HIDE Macro  
AC\_PRESSED Macro  
AC\_TICK Macro  
UPDATE\_HOUR Macro  
UPDATE\_MINUTE Macro  
UPDATE\_SECOND Macro  
AcCreate Function  
AcDraw Function  
AcSetHour Function  
AcSetMinute Function  
AcSetSecond Function  
ANALOGCLOCK Structure

## **Button**

### **Button States**

BTN\_DISABLED Macro  
BTN\_DRAW Macro  
BTN\_DRAW\_FOCUS Macro  
BTN\_FOCUSED Macro  
BTN\_HIDE Macro  
BTN\_PRESSED Macro  
BTN\_TEXTBOTTOM Macro  
BTN\_TEXTLEFT Macro  
BTN\_TEXTRIGHT Macro  
BTN\_TEXTTOP Macro

BTN\_TOGGLE Macro  
BTN\_TWOTONE Macro  
BTN\_NOPANEL Macro  
BtnCreate Function  
BtnDraw Function  
BtnGetText Macro  
BtnSetText Function  
BtnGetBitmap Macro  
BtnSetBitmap Macro  
BtnMsgDefault Function  
BtnTranslateMsg Function  
BUTTON Structure

## **Chart**

### **Chart States**

CH\_DISABLED Macro  
CH\_DRAW Macro  
CH\_DRAW\_DATA Macro  
CH\_3D\_ENABLE Macro  
CH\_BAR Macro  
CH\_BAR\_HOR Macro  
CH\_DONUT Macro  
CH\_LEGEND Macro  
CH\_NUMERIC Macro  
CH\_PERCENT Macro  
CH\_PIE Macro  
CH\_VALUE Macro  
CH\_HIDE Macro

### **Data Series Status Settings**

HIDE\_DATA Macro  
SHOW\_DATA Macro  
Chart Examples  
ChCreate Function  
ChDraw Function

ChAddDataSeries Function  
ChRemoveDataSeries Function  
ChShowSeries Macro  
ChHideSeries Macro  
ChGetShowSeriesCount Macro  
ChGetShowSeriesStatus Macro  
ChSetValueLabel Macro  
ChGetValueLabel Macro  
ChGetValueMax Macro  
ChGetValueMin Macro  
ChSetValueRange Function  
ChGetValueRange Macro  
ChSetSampleLabel Macro  
ChGetSampleLabel Macro  
ChGetSampleStart Macro  
ChGetSampleEnd Macro  
ChSetPercentRange Function  
ChGetPercentRange Macro  
ChSetSampleRange Function  
ChGetSampleRange Macro  
ChGetPercentMax Macro  
ChGetPercentMin Macro  
ChSetColorTable Macro  
ChGetColorTable Macro  
ChSetTitle Macro  
ChGetTitle Macro  
ChSetTitleFont Macro  
ChGetTitleFont Macro  
ChGetAxisLabelFont Macro  
ChSetAxisLabelFont Macro  
ChGetGridLabelFont Macro  
ChSetGridLabelFont Macro  
ChFreeDataSeries Function

ChTranslateMsg Function

CHART Structure

DATASERIES Structure

CHARTPARAM Structure

### **Color Table**

CH\_CLR0 Macro

CH\_CLR1 Macro

CH\_CLR2 Macro

CH\_CLR3 Macro

CH\_CLR4 Macro

CH\_CLR5 Macro

CH\_CLR6 Macro

CH\_CLR7 Macro

CH\_CLR8 Macro

CH\_CLR9 Macro

CH\_CLR10 Macro

CH\_CLR11 Macro

CH\_CLR12 Macro

CH\_CLR13 Macro

CH\_CLR14 Macro

CH\_CLR15 Macro

### **Checkbox**

#### **Check Box States**

CB\_CHECKED Macro

CB\_DISABLED Macro

CB\_DRAW Macro

CB\_DRAW\_CHECK Macro

CB\_DRAW\_FOCUS Macro

CB\_FOCUSED Macro

CB\_HIDE Macro

CbCreate Function

CbDraw Function

CbGetText Macro

CbSetText Function  
CbMsgDefault Function  
CbTranslateMsg Function  
CHECKBOX Structure

## **Round Dial**

### **Dial States**

RDIA\_DISABLED Macro  
RDIA\_DRAW Macro  
RDIA\_HIDE Macro  
RDIA\_ROT\_CCW Macro  
RDIA\_ROT\_CW Macro  
RdiaCreate Function  
RdiaDraw Function  
RdiaIncVal Macro  
RdiaDecVal Macro  
RdiaGetVal Macro  
RdiaSetVal Macro  
RdiaMsgDefault Function  
RdiaTranslateMsg Function  
ROUNDIAL Structure

## **Digital Meter**

### **Digital Meter States**

DM\_DISABLED Macro  
DM\_DRAW Macro  
DM\_HIDE Macro  
DM\_CENTER\_ALIGN Macro  
DM\_RIGHT\_ALIGN Macro  
DM\_FRAME Macro  
DM\_UPDATE Macro  
DmCreate Function  
DmDraw Function  
DmGetValue Macro  
DmSetValue Function

DmDecVal Macro  
DmIncVal Macro  
DmTranslateMsg Function  
DIGITALMETER Structure

## **Edit Box**

### **Edit Box States**

EB\_CENTER\_ALIGN Macro  
EB\_DISABLED Macro  
EB\_DRAW Macro  
EB\_HIDE Macro  
EB\_FOCUSED Macro  
EB\_RIGHT\_ALIGN Macro  
EB\_DRAW\_CARET Macro  
EB\_CARET Macro  
EbCreate Function  
EbDraw Function  
EbGetText Macro  
EbSetText Function  
EbAddChar Function  
EbDeleteChar Function  
EbMsgDefault Function  
EbTranslateMsg Function  
EDITBOX Structure

## **Grid**

### **Grid States**

GRID\_FOCUSED Macro  
GRID\_DISABLED Macro  
GRID\_SHOW\_LINES Macro  
GRID\_SHOW\_FOCUS Macro  
GRID\_SHOW\_BORDER\_ONLY Macro  
GRID\_SHOW\_SEPARATORS\_ONLY Macro  
GRID\_DRAW\_ITEMS Macro  
GRID\_DRAW\_ALL Macro

GRID\_HIDE Macro

### **Grid Item States**

GRIDITEM\_SELECTED Macro

GRIDITEM\_IS\_TEXT Macro

GRIDITEM\_IS\_BITMAP Macro

GRIDITEM\_TEXTBOTTOM Macro

GRIDITEM\_TEXTLEFT Macro

GRIDITEM\_TEXTRIGHT Macro

GRIDITEM\_TEXTTOP Macro

GRIDITEM\_DRAW Macro

GridCreate Function

GridDraw Function

GridClearCellState Function

GridGetFocusX Macro

GridGetFocusY Macro

GRID\_OUT\_OF\_BOUNDS Macro

GRID\_SUCCESS Macro

GridFreeItems Function

GridGetCell Function

GridSetCell Function

GridSetCellState Function

GridSetFocus Function

GridMsgDefault Function

GridTranslateMsg Function

GRID Structure

GRIDITEM Structure

### **Group Box**

#### **Group Box States**

GB\_CENTER\_ALIGN Macro

GB\_DISABLED Macro

GB\_DRAW Macro

GB\_HIDE Macro

GB\_RIGHT\_ALIGN Macro

GbCreate Function  
GbDraw Function  
GbGetText Macro  
GbSetText Function  
GbTranslateMsg Function  
GROUPBOX Structure

## **List Box**

### **List Box States**

LB\_RIGHT\_ALIGN Macro  
LB\_SINGLE\_SEL Macro  
LB\_CENTER\_ALIGN Macro  
LB\_DISABLED Macro  
LB\_DRAW Macro  
LB\_DRAW\_FOCUS Macro  
LB\_DRAW\_ITEMS Macro  
LB\_FOCUSED Macro  
LB\_HIDE Macro

### **List Item Status**

LB\_STS\_SELECTED Macro  
LB\_STS\_REDRAW Macro

LbCreate Function  
LbDraw Function  
LbGetItemList Macro  
LbAddItem Function  
LbDelItem Function  
LbChangeSel Function  
LbSetSel Macro  
LbGetSel Function  
LbGetFocusedItem Function  
LbSetFocusedItem Function  
LbGetCount Macro  
LbGetVisibleCount Macro  
LbSetBitmap Macro

LbGetBitmap Macro  
LbDelItemsList Function  
LbMsgDefault Function  
LbTranslateMsg Function  
LISTBOX Structure  
LISTITEM Structure

## **Meter**

### **Meter States**

MTR\_DISABLED Macro  
MTR\_DRAW Macro  
MTR\_HIDE Macro  
MTR\_RING Macro  
MTR\_DRAW\_UPDATE Macro  
MtrCreate Function  
MtrDraw Function  
MtrSetVal Function  
MtrGetVal Macro  
MtrDecVal Macro  
MtrIncVal Macro  
MtrSetScaleColors Macro  
MtrSetTitleFont Macro  
MtrSetValueFont Macro  
METER\_TYPE Macro  
MTR\_ACCURACY Macro  
MtrMsgDefault Function  
MtrTranslateMsg Function  
METER Structure

## **Picture Control**

### **Picture States**

PICT\_DISABLED Macro  
PICT\_DRAW Macro  
PICT\_FRAME Macro  
PICT\_HIDE Macro

PictCreate Function  
PictDraw Function  
PictSetBitmap Macro  
PictGetBitmap Macro  
PictGetScale Macro  
PictSetScale Macro  
PictTranslateMsg Function  
PICTURE Structure

## **Progress Bar**

### **Progress Bar States**

PB\_DISABLED Macro  
PB\_DRAW Macro  
PB\_DRAW\_BAR Macro  
PB\_HIDE Macro  
PB\_VERTICAL Macro  
PbCreate Function  
PbDraw Function  
PbSetRange Function  
PbGetRange Macro  
PbSetPos Function  
PbGetPos Macro  
PbTranslateMsg Function  
PROGRESSBAR Structure

## **Radio Button**

### **Radio Button States**

RB\_CHECKED Macro  
RB\_DISABLED Macro  
RB\_DRAW Macro  
RB\_DRAW\_CHECK Macro  
RB\_DRAW\_FOCUS Macro  
RB\_FOCUSED Macro  
RB\_GROUP Macro  
RB\_HIDE Macro

RbCreate Function  
RbDraw Function  
RbGetCheck Function  
RbSetCheck Function  
RbGetText Macro  
RbSetText Function  
RbMsgDefault Function  
RbTranslateMsg Function  
RADIOBUTTON Structure

## **Slider/Scroll Bar**

### **Slider States**

SLD\_DISABLED Macro  
SLD\_DRAW Macro  
SLD\_DRAW\_FOCUS Macro  
SLD\_DRAW\_THUMB Macro  
SLD\_FOCUSED Macro  
SLD\_HIDE Macro  
SLD\_SCROLLBAR Macro  
SLD\_VERTICAL Macro

SldCreate Function  
SldDraw Function  
SldSetPage Function  
SldGetPage Macro  
SldSetPos Function  
SldGetPos Macro  
SldSetRange Function  
SldGetRange Macro  
SldIncPos Macro  
SldDecPos Macro  
SldMsgDefault Function  
SldTranslateMsg Function  
SLIDER Structure

## **Static Text**

## **Static Text States**

ST\_CENTER\_ALIGN Macro

ST\_DISABLED Macro

ST\_DRAW Macro

ST\_FRAME Macro

ST\_HIDE Macro

ST\_RIGHT\_ALIGN Macro

ST\_UPDATE Macro

StCreate Function

StDraw Function

StGetText Macro

StSetText Function

StTranslateMsg Function

STATICTEXT Structure

## **Text Entry**

### **TextEntry States**

TE\_KEY\_PRESSED Macro

TE\_DISABLED Macro

TE\_ECHO\_HIDE Macro

TE\_DRAW Macro

TE\_HIDE Macro

TE\_UPDATE\_KEY Macro

TE\_UPDATE\_TEXT Macro

### **Key Command Types**

TE\_DELETE\_COM Macro

TE\_ENTER\_COM Macro

TE\_SPACE\_COM Macro

TeCreate Function

TeDraw Function

TeGetBuffer Macro

TeSetBuffer Function

TeClearBuffer Function

TeGetKeyCommand Function

TeSetKeyCommand Function  
TeCreateKeyMembers Function  
TeAddChar Function  
TeIsKeyPressed Function  
TeSpaceChar Function  
TeDelKeyMembers Function  
TeSetKeyText Function  
TeMsgDefault Function  
TeTranslateMsg Function  
TEXTENTRY Structure  
KEYMEMBER Structure

## **Window**

### **Window States**

WND\_DISABLED Macro  
WND\_DRAW Macro  
WND\_DRAW\_CLIENT Macro  
WND\_DRAW\_TITLE Macro  
WND\_FOCUSED Macro  
WND\_HIDE Macro  
WND\_TITLECENTER Macro  
WndCreate Function  
WndDraw Function  
WndGetText Macro  
WndSetText Function  
WndTranslateMsg Function  
WINDOW Structure

## **Object States**

### **Common Object States**

FOCUSED Macro  
DISABLED Macro  
HIDE Macro  
DRAW Macro  
DRAW\_FOCUS Macro

DRAW\_UPDATE Macro

GetState Macro

ClrState Macro

SetState Macro

## **Object Management**

GOLAddObject Function

GOLFindObject Function

GOLRedraw Macro

GOLRedrawRec Function

GOLDraw Function

GOLDrawComplete Macro

GOLDrawCallback Function

GOLFree Function

GetObjType Macro

GetObjID Macro

GetObjNext Macro

GOLDeleteObject Function

GOLDeleteObjectByID Function

GOLNewList Macro

GOLGetList Macro

GOLSetList Macro

GOLSetFocus Function

IsObjUpdated Macro

GOLInit Function

GOLGetFocus Macro

GOLCanBeFocused Function

GOLGetFocusNext Function

GOLGetFocusPrev Function

GOLPanelDraw Macro

GOLPanelDrawTsk Function

GOLTwoTonePanelDrawTsk Function

## **GOL Messages**

GOLMsg Function

GOLMsgCallback Function

GOL\_MSG Structure

TRANS\_MSG Enumeration

INPUT\_DEVICE\_EVENT Enumeration

INPUT\_DEVICE\_TYPE Enumeration

### **Scan Key Codes**

SCAN\_BS\_PRESSED Macro

SCAN\_BS\_RELEASED Macro

SCAN\_CR\_PRESSED Macro

SCAN\_CR\_RELEASED Macro

SCAN\_DEL\_PRESSED Macro

SCAN\_DEL\_RELEASED Macro

SCAN\_DOWN\_PRESSED Macro

SCAN\_DOWN\_RELEASED Macro

SCAN\_END\_PRESSED Macro

SCAN\_END\_RELEASED Macro

SCAN\_HOME\_PRESSED Macro

SCAN\_HOME\_RELEASED Macro

SCAN\_LEFT\_PRESSED Macro

SCAN\_LEFT\_RELEASED Macro

SCAN\_PGDOWN\_PRESSED Macro

SCAN\_PGDOWN\_RELEASED Macro

SCAN\_PGUP\_PRESSED Macro

SCAN\_PGUP\_RELEASED Macro

SCAN\_RIGHT\_PRESSED Macro

SCAN\_RIGHT\_RELEASED Macro

SCAN\_SPACE\_PRESSED Macro

SCAN\_SPACE\_RELEASED Macro

SCAN\_TAB\_PRESSED Macro

SCAN\_TAB\_RELEASED Macro

SCAN\_UP\_PRESSED Macro

SCAN\_UP\_RELEASED Macro

### **Style Scheme**

GOLCreateScheme Function  
GOLSetScheme Macro  
GOLGetScheme Macro  
GOLGetSchemeDefault Macro  
GOL\_SCHEME Structure

### **Default Style Scheme Settings**

FONTDEFAULT Variable  
GOLFontDefault Variable  
GOL\_EMBOSS\_SIZE Macro  
GOLSchemeDefault Variable  
RGBConvert Macro

### **GOL Global Variables**

\_pDefaultGolScheme Variable  
\_pGolObjects Variable  
\_pObjectFocused Variable

### **Graphics Primitive Layer API**

#### **Text Functions**

FONT\_HEADER Structure  
FONT\_FLASH Structure  
FONT\_EXTERNAL Type  
SetFont Function  
GetFontOrientation Macro  
SetFontOrientation Macro  
GFX\_Font\_GetAntiAliasType Macro  
GFX\_Font\_SetAntiAliasType Macro  
OutChar Function  
OutText Function  
OutTextXY Function  
GetTextHeight Function  
GetTextWidth Function  
XCHAR Macro

#### **Anti-Alias Type**

ANTI\_ALIAS\_OPAQUE Macro

ANTIALIAS\_TRANSLUCENT Macro

## **Gradient**

BarGradient Function

BevelGradient Function

GFX\_GRADIENT\_TYPE Enumeration

GFX\_GRADIENT\_STYLE Structure

## **Line Functions**

Line Function

LineRel Macro

LineTo Macro

SetLineThickness Macro

SetLineType Macro

## **Line Types**

SOLID\_LINE Macro

DASHED\_LINE Macro

DOTTED\_LINE Macro

## **Line Size**

NORMAL\_LINE Macro

THICK\_LINE Macro

## **Rectangle Functions**

Bar Function

Rectangle Macro

DrawPoly Function

## **Circle Functions**

Circle Macro

FillCircle Macro

Arc Function

DrawArc Function

Bevel Function

FillBevel Function

SetBevelDrawType Macro

## **Graphic Cursor**

GetX Macro

GetY Macro

MoveRel Macro

MoveTo Macro

### **Alpha Blending Functions**

SetAlpha Macro

GetAlpha Macro

AlphaBlendWindow Function

### **Bitmap Functions**

PutImage Macro

PutImagePartial Function

GetImageHeight Function

GetImageWidth Function

BITMAP\_HEADER Structure

### **Bitmap Settings**

IMAGE\_NORMAL Macro

IMAGE\_X2 Macro

Bitmap Source

### **External Memory**

ExternalMemoryCallback Function

EXTERNAL\_FONT\_BUFFER\_SIZE Macro

Memory Type

### **Set Up Functions**

ClearDevice Function

InitGraph Function

GFX\_RESOURCE Enumeration

GFX\_IMAGE\_HEADER Structure

IMAGE\_FLASH Structure

IMAGE\_RAM Structure

GFX\_EXTDATA Structure

## **Display Device Driver Layer API**

### **Display Device Driver Level Primitives**

GetPixel Function  
PutPixel Function  
GetColor Macro  
SetColor Macro  
GetMaxX Macro  
GetMaxY Macro  
SetClip Function  
SetClipRgn Function  
GetClipBottom Macro  
GetClipLeft Macro  
GetClipRight Macro  
GetClipTop Macro  
CLIP\_DISABLE Macro  
CLIP\_ENABLE Macro  
TransparentColorEnable Function  
TransparentColorDisable Macro  
GetTransparentColorStatus Macro  
GetTransparentColor Macro  
TRANSPARENT\_COLOR\_DISABLE Macro  
TRANSPARENT\_COLOR\_ENABLE Macro  
DisplayBrightness Function  
GetPageAddress Macro  
CopyBlock Function  
CopyPageWindow Function  
CopyWindow Function  
SetActivePage Function  
SetVisualPage Function

**Color Definition**

BLACK Macro  
BLUE Macro  
BRIGHTBLUE Macro  
BRIGHTCYAN Macro  
BRIGHTGREEN Macro

BRIGHTMAGENTA Macro  
BRIGHTRED Macro  
BRIGHTYELLOW Macro  
BROWN Macro  
CYAN Macro  
DARKGRAY Macro  
GRAY0 Macro  
GRAY1 Macro  
GRAY2 Macro  
GRAY3 Macro  
GRAY4 Macro  
GRAY5 Macro  
GRAY6 Macro  
GREEN Macro  
LIGHTBLUE Macro  
LIGHTCYAN Macro  
LIGHTGRAY Macro  
LIGHTGREEN Macro  
LIGHTMAGENTA Macro  
LIGHTRED Macro  
MAGENTA Macro  
RED Macro  
WHITE Macro  
YELLOW Macro

### **Display Device Driver Control**

IsDeviceBusy Function  
ResetDevice Function

### **Advanced Display Driver Features**

#### **Alpha Blending**

GFXGetPageOriginAddress Function  
GFXGetPageXYAddress Function

#### **Transitions**

GFXTransition Function

GFXSetupTransition Function  
GFXExecutePendingTransition Function  
GFXIsTransitionPending Function  
GFX\_TRANSITION\_DIRECTION Enumeration  
GFX\_TRANSITION\_TYPE Enumeration

### **Double Buffering**

SwitchOffDoubleBuffering Function  
SwitchOnDoubleBuffering Function  
InvalidateRectangle Function  
RequestDisplayUpdate Function  
UpdateDisplayNow Function

### **Microchip Graphics Controller**

#### **Rectangle Copy Operations**

ROPBlock Function  
Scroll Function  
RCC\_SRC\_ADDR\_CONTINUOUS Macro  
RCC\_ROP\_0 Macro  
RCC\_COPY Macro

#### **Decompressing DEFLATEd data**

Decompress Function

#### **Palette Mode**

ClearPaletteChangeError Function  
DisablePalette Function  
EnablePalette Function  
GetPaletteChangeError Function  
IsPaletteEnabled Function  
PaletteInit Function  
RequestPaletteChange Function  
SetPalette Function  
SetPaletteBpp Function  
SetPaletteFlash Function  
PALETTE\_FLASH Structure  
PALETTE\_HEADER Structure

PALETTE\_EXTERNAL Type  
RequestEntirePaletteChange Macro  
SetEntirePalette Macro  
Palette.h

### **Set Up Display Interface**

GFX\_GCLK\_DIVIDER Macro  
GFX\_EPMP\_CS1\_BASE\_ADDRESS Macro  
GFX\_EPMP\_CS1\_MEMORY\_SIZE Macro  
GFX\_EPMP\_CS2\_BASE\_ADDRESS Macro  
GFX\_EPMP\_CS2\_MEMORY\_SIZE Macro  
GFX\_DISPLAY\_BUFFER\_LENGTH Macro  
GFX\_DISPLAY\_BUFFER\_START\_ADDRESS Macro

External or Internal Memory and Palettes

Image Decoders

### **Image Decoders API**

#### **Image Decoder Configuration**

IMG\_SUPPORT\_BMP Macro  
IMG\_SUPPORT\_GIF Macro  
IMG\_SUPPORT\_JPEG Macro  
IMG\_SUPPORT\_IMAGE\_DECODER\_LOOP\_CALLBACK Macro  
IMG\_USE\_ONLY\_565\_GRAPHICS\_DRIVER\_FOR\_OUTPUT Macro  
IMG\_USE\_ONLY\_MDD\_FILE\_SYSTEM\_FOR\_INPUT Macro

ImageDecode Function

ImageDecoderInit Function

ImageLoopCallbackRegister Function

ImageDecodeTask Function

ImageFullScreenDecode Macro

ImageAbort Macro

\_BMPDECODER Structure

\_GIFDECODER Structure

\_JPEGDECODER Structure

### **Miscellaneous Topics**

Starting a New Project

Changing the default Font  
Advanced Font Features  
Using Primitive Rendering Functions in Blocking and Non-Blocking M  
Using Microchip Graphics Module Color Look Up Table in Application  
Converting Images to Use a Common Palette in GIMP  
How to Define Colors in your Applications  
Connecting RGB data bus  
Adding New Device Driver  
References

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Index

These are all topics and symbols available in this documentation.

[\\_](#) | [A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [R](#) | [S](#) | [T](#) | [U](#) | [W](#) | [X](#) | [Y](#)

<a href="#">_</a>	<a href="#">GridSet</a>
<a href="#">_BMPDECODER</a> structure	<a href="#">GridSet</a>
<a href="#">_GIFDECODER</a> structure	<a href="#">GridTra</a>
<a href="#">_JPEGDECODER</a> structure	<a href="#">Group E</a>
<a href="#">_pDefaultGolScheme</a> variable	<a href="#">Group E</a>
<a href="#">_pGolObjects</a> variable	<a href="#">GROUF</a>
<a href="#">_pObjectFocused</a> variable	<b>H</b>
<b>A</b>	<a href="#">Hardwa</a>
<a href="#">AC_DISABLED</a> macro	<a href="#">Hardwa</a>
<a href="#">AC_DRAW</a> macro	<a href="#">HIDE m</a>
<a href="#">AC_HIDE</a> macro	<a href="#">HIDE_D</a>
<a href="#">AC_PRESSED</a> macro	<a href="#">How to</a>
<a href="#">AC_TICK</a> macro	<b>I</b>
<a href="#">AcCreate</a> function	<a href="#">Image C</a>
<a href="#">AcDraw</a> function	<a href="#">Image L</a>
<a href="#">AcSetHour</a> function	<a href="#">Image L</a>
<a href="#">AcSetMinute</a> function	<a href="#">Image L</a>
<a href="#">AcSetSecond</a> function	<a href="#">Image L</a>
<a href="#">Adding New Device Driver</a>	<a href="#">Image S</a>
<a href="#">Advanced Display Driver Features</a>	<a href="#">IMAGE_</a>
<a href="#">Advanced Font Features</a>	<a href="#">IMAGE_</a>
<a href="#">Advanced Font Features Selection</a>	<a href="#">IMAGE_</a>
<a href="#">Alpha Blend Option</a>	<a href="#">IMAGE_</a>
<a href="#">Alpha Blending</a>	<a href="#">ImageA</a>
<a href="#">Alpha Blending Functions</a>	<a href="#">ImageD</a>
<a href="#">AlphaBlendWindow</a> function	<a href="#">ImageD</a>

[Analog Clock](#)  
[Analog Clock States](#)  
[ANALOGCLOCK structure](#)  
[Anti-Alias Type](#)  
[ANTIALIAS\\_OPAQUE macro](#)  
[ANTIALIAS\\_TRANSLUCENT macro](#)  
[Application Configuration](#)  
[Arc function](#)

## B

[Bar function](#)  
[BarGradient function](#)  
[Bevel function](#)  
[BevelGradient function](#)  
[Bitmap Functions](#)  
[Bitmap Settings](#)  
[Bitmap Source](#)  
[BITMAP\\_HEADER structure](#)  
[BLACK macro](#)  
[BLUE macro](#)  
[BRIGHTBLUE macro](#)  
[BRIGHTCYAN macro](#)  
[BRIGHTGREEN macro](#)  
[BRIGHTMAGENTA macro](#)  
[BRIGHTRED macro](#)  
[BRIGHTYELLOW macro](#)  
[BROWN macro](#)  
[BTN\\_DISABLED macro](#)  
[BTN\\_DRAW macro](#)  
[BTN\\_DRAW\\_FOCUS macro](#)  
[BTN\\_FOCUSED macro](#)  
[BTN\\_HIDE macro](#)  
[BTN\\_NOPANEL macro](#)

[ImageD](#)  
[ImageF](#)  
[ImageL](#)  
[IMG\\_SU](#)  
[IMG\\_SU](#)  
[IMG\\_SU](#)  
[macro](#)  
[IMG\\_SU](#)  
[IMG\\_US](#)  
[macro](#)  
[IMG\\_US](#)  
[InitGrap](#)  
[Input De](#)  
[INPUT\\_](#)  
[INPUT\\_](#)  
[Introduc](#)  
[Invalida](#)  
[IsDevice](#)  
[IsObjUp](#)  
[IsPalett](#)

## K

[Key Cor](#)  
[KEYME](#)

## L

[LB\\_CEN](#)  
[LB\\_DIS](#)  
[LB\\_DR/](#)  
[LB\\_DR/](#)  
[LB\\_DR/](#)  
[LB\\_DR/](#)  
[LB\\_FOC](#)  
[LB\\_HID](#)

<a href="#"><u>BTN_PRESSED macro</u></a>	<a href="#"><u>LB_RIG</u></a>
<a href="#"><u>BTN_TEXTBOTTOM macro</u></a>	<a href="#"><u>LB_SIN</u></a>
<a href="#"><u>BTN_TEXTLEFT macro</u></a>	<a href="#"><u>LB_STS</u></a>
<a href="#"><u>BTN_TEXTRIGHT macro</u></a>	<a href="#"><u>LB_STS</u></a>
<a href="#"><u>BTN_TEXTTOP macro</u></a>	<a href="#"><u>LbAddIt</u></a>
<a href="#"><u>BTN_TOGGLE macro</u></a>	<a href="#"><u>LbChan</u></a>
<a href="#"><u>BTN_TWOTONE macro</u></a>	<a href="#"><u>LbCreat</u></a>
<a href="#"><u>BtnCreate function</u></a>	<a href="#"><u>LbDellte</u></a>
<a href="#"><u>BtnDraw function</u></a>	<a href="#"><u>LbDellte</u></a>
<a href="#"><u>BtnGetBitmap macro</u></a>	<a href="#"><u>LbDraw</u></a>
<a href="#"><u>BtnGetText macro</u></a>	<a href="#"><u>LbGetB</u></a>
<a href="#"><u>BtnMsgDefault function</u></a>	<a href="#"><u>LbGetC</u></a>
<a href="#"><u>BtnSetBitmap macro</u></a>	<a href="#"><u>LbGetF</u></a>
<a href="#"><u>BtnSetText function</u></a>	<a href="#"><u>LbGetIt</u></a>
<a href="#"><u>BtnTranslateMsg function</u></a>	<a href="#"><u>LbGetS</u></a>
<a href="#"><u>Button</u></a>	<a href="#"><u>LbGetV</u></a>
<a href="#"><u>Button States</u></a>	<a href="#"><u>LbMsgD</u></a>
<a href="#"><u>BUTTON structure</u></a>	<a href="#"><u>LbSetBi</u></a>

## C

<a href="#"><u>CB_CHECKED macro</u></a>	<a href="#"><u>LbSetFc</u></a>
<a href="#"><u>CB_DISABLED macro</u></a>	<a href="#"><u>LbSetSe</u></a>
<a href="#"><u>CB_DRAW macro</u></a>	<a href="#"><u>LbSetSe</u></a>
<a href="#"><u>CB_DRAW_CHECK macro</u></a>	<a href="#"><u>LbTrans</u></a>
<a href="#"><u>CB_DRAW_FOCUS macro</u></a>	<a href="#"><u>Library,</u></a>
<a href="#"><u>CB_FOCUSED macro</u></a>	<a href="#"><u>Library,</u></a>
<a href="#"><u>CB_HIDE macro</u></a>	<a href="#"><u>LIGHTB</u></a>
<a href="#"><u>CbCreate function</u></a>	<a href="#"><u>LIGHTC</u></a>
<a href="#"><u>CbDraw function</u></a>	<a href="#"><u>LIGHTC</u></a>
<a href="#"><u>CbGetText macro</u></a>	<a href="#"><u>LIGHTC</u></a>
<a href="#"><u>CbMsgDefault function</u></a>	<a href="#"><u>LIGHTM</u></a>
<a href="#"><u>CbSetText function</u></a>	<a href="#"><u>LIGHTR</u></a>
<a href="#"><u>CbTranslateMsg function</u></a>	<a href="#"><u>Line fun</u></a>
	<a href="#"><u>Line Fur</u></a>
	<a href="#"><u>Line Siz</u></a>
	<a href="#"><u>Line Typ</u></a>

[CH\\_3D\\_ENABLE macro](#)  
[CH\\_BAR macro](#)  
[CH\\_BAR\\_HOR macro](#)  
[CH\\_CLR0 macro](#)  
[CH\\_CLR1 macro](#)  
[CH\\_CLR10 macro](#)  
[CH\\_CLR11 macro](#)  
[CH\\_CLR12 macro](#)  
[CH\\_CLR13 macro](#)  
[CH\\_CLR14 macro](#)  
[CH\\_CLR15 macro](#)  
[CH\\_CLR2 macro](#)  
[CH\\_CLR3 macro](#)  
[CH\\_CLR4 macro](#)  
[CH\\_CLR5 macro](#)  
[CH\\_CLR6 macro](#)  
[CH\\_CLR7 macro](#)  
[CH\\_CLR8 macro](#)  
[CH\\_CLR9 macro](#)  
[CH\\_DISABLED macro](#)  
[CH\\_DONUT macro](#)  
[CH\\_DRAW macro](#)  
[CH\\_DRAW\\_DATA macro](#)  
[CH\\_HIDE macro](#)  
[CH\\_LEGEND macro](#)  
[CH\\_NUMERIC macro](#)  
[CH\\_PERCENT macro](#)  
[CH\\_PIE macro](#)  
[CH\\_VALUE macro](#)  
[ChAddDataSeries function](#)  
[Changing the default Font](#)  
[Chart](#)  
[Chart Examples](#)

[LineRel](#)  
[LineTo r](#)  
[List Box](#)  
[List Box](#)  
[List Item](#)  
[LISTBO](#)  
[LISTITE](#)

## M

[MAGEN](#)  
[MEB\\_B](#)  
[Memory](#)  
[Meter](#)  
[Meter S](#)  
[METER](#)  
[METER](#)  
[Microch](#)  
[Microch](#)  
[Miscella](#)  
[Miscella](#)  
[MoveRe](#)  
[MoveTo](#)  
[MSG\\_D](#)  
[MSG\\_F](#)  
[MTR\\_A](#)  
[MTR\\_D](#)  
[MTR\\_D](#)  
[MTR\\_D](#)  
[MTR\\_H](#)  
[MTR\\_R](#)  
[MtrCrea](#)  
[MtrDec\](#)  
[MtrDraw](#)

[Chart States](#)  
[CHART structure](#)  
[CHARTPARAM structure](#)  
[ChCreate function](#)  
[ChDraw function](#)  
[Check Box States](#)  
[Checkbox](#)  
[CHECKBOX structure](#)  
[ChFreeDataSeries function](#)  
[ChGetAxisLabelFont macro](#)  
[ChGetColorTable macro](#)  
[ChGetGridLabelFont macro](#)  
[ChGetPercentMax macro](#)  
[ChGetPercentMin macro](#)  
[ChGetPercentRange macro](#)  
[ChGetSampleEnd macro](#)  
[ChGetSampleLabel macro](#)  
[ChGetSampleRange macro](#)  
[ChGetSampleStart macro](#)  
[ChGetShowSeriesCount macro](#)  
[ChGetShowSeriesStatus macro](#)  
[ChGetTitle macro](#)  
[ChGetTitleFont macro](#)  
[ChGetValueLabel macro](#)  
[ChGetValueMax macro](#)  
[ChGetValueMin macro](#)  
[ChGetValueRange macro](#)  
[ChHideSeries macro](#)  
[ChRemoveDataSeries function](#)  
[ChSetAxisLabelFont macro](#)  
[ChSetColorTable macro](#)  
[ChSetGridLabelFont macro](#)  
[ChSetPercentRange function](#)

[MtrGetV](#)  
[MtrIncV](#)  
[MtrMsgl](#)  
[MtrSetS](#)  
[MtrSetT](#)  
[MtrSetV](#)  
[MtrSetV](#)  
[MtrTran](#)

## N

[NORMA](#)

## O

[OBJ\\_HF](#)  
[Object M](#)  
[Object F](#)  
[Object S](#)  
[OutCha](#)  
[OutText](#)  
[OutText](#)

## P

[Palette](#)  
[Palette.](#)  
[PALETT](#)  
[PALETT](#)  
[PALETT](#)  
[PaletteI](#)  
[PB\\_DIS](#)  
[PB\\_DR.](#)  
[PB\\_DR.](#)  
[PB\\_HID](#)  
[PB\\_VEF](#)

[ChSetSampleLabel macro](#)

[ChSetSampleRange function](#)

[ChSetTitle macro](#)

[ChSetTitleFont macro](#)

[ChSetValueLabel macro](#)

[ChSetValueRange function](#)

[ChShowSeries macro](#)

[ChTranslateMsg function](#)

[Circle Functions](#)

[Circle macro](#)

[ClearDevice function](#)

[ClearPaletteChangeError function](#)

[CLIP\\_DISABLE macro](#)

[CLIP\\_ENABLE macro](#)

[ClrState macro](#)

[Color Definition](#)

[Color Table](#)

[COLOR\\_DEPTH macro](#)

[Common Object States](#)

[Configuration Setting](#)

[Connecting RGB data bus](#)

[Converting Images to Use a Common Palette in GIMP](#)

[CopyBlock function](#)

[CopyPageWindow function](#)

[CopyWindow function](#)

[CYAN macro](#)

## D

[DARKGRAY macro](#)

[DASHED\\_LINE macro](#)

[Data Series Status Settings](#)

[DATASERIES structure](#)

[Decompress function](#)

[PbCrea](#)

[PbDraw](#)

[PbGetP](#)

[PbGetR](#)

[PbSetP](#)

[PbSetR](#)

[PbTrans](#)

[PIC\\_SK](#)

[PIC24F](#)

[PICT\\_D](#)

[PICT\\_D](#)

[PICT\\_F](#)

[PICT\\_H](#)

[PictCrea](#)

[PictDraw](#)

[PictGetI](#)

[PictGetS](#)

[PictSetF](#)

[PictSetS](#)

[PictTran](#)

[Picture .](#)

[Picture .](#)

[PICTUF](#)

[PMP Int](#)

[Progres](#)

[Progres](#)

[PROGR](#)

[PutImag](#)

[PutImag](#)

[PutPixe](#)

## R

[Radio B](#)

<a href="#">Decompressing DEFLATEd data</a>	<a href="#">Radio B</a>
<a href="#">Default Style Scheme Settings</a>	<a href="#">RADIOE</a>
<a href="#">Demo Compatibility Matrix</a>	<a href="#">RB_CH</a>
<a href="#">Demo Projects</a>	<a href="#">RB_DIS</a>
<a href="#">Demo Summary</a>	<a href="#">RB_DR.</a>
<a href="#">Development Platform Used</a>	<a href="#">RB_DR.</a>
<a href="#">Device Driver Options</a>	<a href="#">RB_DR.</a>
<a href="#">Dial States</a>	<a href="#">RB_FO</a>
<a href="#">Digital Meter</a>	<a href="#">RB_GR</a>
<a href="#">Digital Meter States</a>	<a href="#">RB_HIC</a>
<a href="#">DIGITALMETER structure</a>	<a href="#">RbCrea</a>
<a href="#">DISABLED macro</a>	<a href="#">RbDraw</a>
<a href="#">DisablePalette function</a>	<a href="#">RbGetC</a>
<a href="#">DISP_DATA_WIDTH macro</a>	<a href="#">RbGetT</a>
<a href="#">DISP_HOR_BACK_PORCH macro</a>	<a href="#">RbMsgI</a>
<a href="#">DISP_HOR_FRONT_PORCH macro</a>	<a href="#">RbSetC</a>
<a href="#">DISP_HOR_PULSE_WIDTH macro</a>	<a href="#">RbSetTe</a>
<a href="#">DISP_HOR_RESOLUTION macro</a>	<a href="#">RbTrans</a>
<a href="#">DISP_INV_LSHIFT macro</a>	<a href="#">RCC_C</a>
<a href="#">DISP_ORIENTATION macro</a>	<a href="#">RCC_D</a>
<a href="#">DISP_VER_BACK_PORCH macro</a>	<a href="#">RCC_D</a>
<a href="#">DISP_VER_FRONT_PORCH macro</a>	<a href="#">RCC_R</a>
<a href="#">DISP_VER_PULSE_WIDTH macro</a>	<a href="#">RCC_R</a>
<a href="#">DISP_VER_RESOLUTION macro</a>	<a href="#">RCC_R</a>
<a href="#">Display Controller Used</a>	<a href="#">RCC_R</a>
<a href="#">Display Device Driver Control</a>	<a href="#">RCC_R</a>
<a href="#">Display Device Driver Layer</a>	<a href="#">RCC_R</a>
<a href="#">Display Device Driver Layer API</a>	<a href="#">RCC_R</a>
<a href="#">Display Device Driver Layer Configuration</a>	<a href="#">RCC_R</a>
<a href="#">Display Device Driver Level Primitives</a>	<a href="#">RCC_R</a>
<a href="#">Display Panel Used</a>	<a href="#">RCC_R</a>
<a href="#">DisplayBrightness function</a>	<a href="#">RCC_R</a>
<a href="#">DM_CENTER_ALIGN macro</a>	<a href="#">RCC_R</a>

<a href="#"><u>DM_DISABLED</u></a> macro	<a href="#"><u>RCC_R</u></a>
<a href="#"><u>DM_DRAW</u></a> macro	<a href="#"><u>RCC_R</u></a>
<a href="#"><u>DM_FRAME</u></a> macro	<a href="#"><u>RCC_R</u></a>
<a href="#"><u>DM_HIDE</u></a> macro	<a href="#"><u>RCC_R</u></a>
<a href="#"><u>DM_RIGHT_ALIGN</u></a> macro	<a href="#"><u>RCC_S</u></a>
<a href="#"><u>DM_UPDATE</u></a> macro	<a href="#"><u>RCC_S</u></a>
<a href="#"><u>DmCreate</u></a> function	<a href="#"><u>RCC_S</u></a>
<a href="#"><u>DmDecVal</u></a> macro	<a href="#"><u>RCC_TI</u></a>
<a href="#"><u>DmDraw</u></a> function	<a href="#"><u>RDIA_D</u></a>
<a href="#"><u>DmGetValue</u></a> macro	<a href="#"><u>RDIA_D</u></a>
<a href="#"><u>DmIncVal</u></a> macro	<a href="#"><u>RDIA_H</u></a>
<a href="#"><u>DmSetValue</u></a> function	<a href="#"><u>RDIA_F</u></a>
<a href="#"><u>DmTranslateMsg</u></a> function	<a href="#"><u>RDIA_F</u></a>
<a href="#"><u>DOTTED_LINE</u></a> macro	<a href="#"><u>RdiaCre</u></a>
<a href="#"><u>Double Buffering</u></a>	<a href="#"><u>RdiaDe</u></a>
<a href="#"><u>DRAW</u></a> macro	<a href="#"><u>RdiaDra</u></a>
<a href="#"><u>DRAW_FOCUS</u></a> macro	<a href="#"><u>RdiaGe</u></a>
<a href="#"><u>DRAW_FUNC</u></a> type	<a href="#"><u>RdiaInc'</u></a>
<a href="#"><u>DRAW_UPDATE</u></a> macro	<a href="#"><u>RdiaMs</u></a>
<a href="#"><u>DrawArc</u></a> function	<a href="#"><u>RdiaSet</u></a>
<a href="#"><u>DrawPoly</u></a> function	<a href="#"><u>RdiaTra</u></a>
<b>E</b>	<a href="#"><u>Rectang</u></a>
<a href="#"><u>EB_CARET</u></a> macro	<a href="#"><u>Rectang</u></a>
<a href="#"><u>EB_CENTER_ALIGN</u></a> macro	<a href="#"><u>Rectang</u></a>
<a href="#"><u>EB_DISABLED</u></a> macro	<a href="#"><u>RED ma</u></a>
<a href="#"><u>EB_DRAW</u></a> macro	<a href="#"><u>Referen</u></a>
<a href="#"><u>EB_DRAW_CARET</u></a> macro	<a href="#"><u>Release</u></a>
<a href="#"><u>EB_FOCUSED</u></a> macro	<a href="#"><u>Reques</u></a>
<a href="#"><u>EB_HIDE</u></a> macro	<a href="#"><u>Reques</u></a>
<a href="#"><u>EB_RIGHT_ALIGN</u></a> macro	<a href="#"><u>ResetDi</u></a>
<a href="#"><u>EbAddChar</u></a> function	<a href="#"><u>RGBCo</u></a>
<a href="#"><u>EbCreate</u></a> function	<a href="#"><u>ROPBlc</u></a>

<a href="#">EbDeleteChar function</a>	<a href="#">Round I</a>
<a href="#">EbDraw function</a>	<a href="#">ROUND</a>
<a href="#">EbGetText macro</a>	
<a href="#">EbMsgDefault function</a>	<b>S</b>
<a href="#">EbSetText function</a>	<a href="#">Scan Ke</a>
<a href="#">EbTranslateMsg function</a>	<a href="#">SCAN_</a>
<a href="#">Edit Box</a>	<a href="#">SCAN_</a>
<a href="#">Edit Box States</a>	<a href="#">SCAN_</a>
<a href="#">EDITBOX structure</a>	<a href="#">SCAN_</a>
<a href="#">EnablePalette function</a>	<a href="#">SCAN_</a>
<a href="#">EXPLORER_16 macro</a>	<a href="#">SCAN_</a>
<a href="#">External Memory</a>	<a href="#">SCAN_</a>
<a href="#">External Memory Buffer</a>	<a href="#">SCAN_</a>
<a href="#">External or Internal Memory and Palettes</a>	<a href="#">SCAN_</a>
<a href="#">EXTERNAL_FONT_BUFFER_SIZE macro</a>	<a href="#">SCAN_</a>
<a href="#">ExternalMemoryCallback function</a>	<a href="#">SCAN_</a>
<b>F</b>	<a href="#">SCAN_</a>
<a href="#">FillBevel function</a>	<a href="#">SCAN_</a>
<a href="#">FillCircle macro</a>	<a href="#">SCAN_</a>
<a href="#">Focus Support Selection</a>	<a href="#">SCAN_</a>
<a href="#">FOCUSED macro</a>	<a href="#">SCAN_</a>
<a href="#">Font Source Selection</a>	<a href="#">SCAN_</a>
<a href="#">Font Type Selection</a>	<a href="#">SCAN_</a>
<a href="#">FONT_EXTERNAL type</a>	<a href="#">SCAN_</a>
<a href="#">FONT_FLASH structure</a>	<a href="#">SCAN_</a>
<a href="#">FONT_HEADER structure</a>	<a href="#">SCAN_</a>
<a href="#">FONTDEFAULT variable</a>	<a href="#">SCAN_</a>
<a href="#">FREE_FUNC type</a>	<a href="#">SCAN_</a>
<b>G</b>	<a href="#">SCAN_</a>
<a href="#">GB_CENTER_ALIGN macro</a>	<a href="#">SCAN_</a>
<a href="#">GB_DISABLED macro</a>	<a href="#">Scroll fu</a>
	<a href="#">Set Up</a>

<a href="#"><u>GB_DRAW macro</u></a>	<a href="#"><u>Set Up I</u></a>
<a href="#"><u>GB_HIDE macro</u></a>	<a href="#"><u>SetActive</u></a>
<a href="#"><u>GB_RIGHT_ALIGN macro</u></a>	<a href="#"><u>SetAlph</u></a>
<a href="#"><u>GbCreate function</u></a>	<a href="#"><u>SetBeve</u></a>
<a href="#"><u>GbDraw function</u></a>	<a href="#"><u>SetClip</u></a>
<a href="#"><u>GbGetText macro</u></a>	<a href="#"><u>SetClipF</u></a>
<a href="#"><u>GbSetText function</u></a>	<a href="#"><u>SetColo</u></a>
<a href="#"><u>GbTranslateMsg function</u></a>	<a href="#"><u>SetEntir</u></a>
<a href="#"><u>GetAlpha macro</u></a>	<a href="#"><u>SetFont</u></a>
<a href="#"><u>GetClipBottom macro</u></a>	<a href="#"><u>SetFont</u></a>
<a href="#"><u>GetClipLeft macro</u></a>	<a href="#"><u>SetLine</u></a>
<a href="#"><u>GetClipRight macro</u></a>	<a href="#"><u>SetLine</u></a>
<a href="#"><u>GetClipTop macro</u></a>	<a href="#"><u>SetPale</u></a>
<a href="#"><u>GetColor macro</u></a>	<a href="#"><u>SetPale</u></a>
<a href="#"><u>GetFontOrientation macro</u></a>	<a href="#"><u>SetPale</u></a>
<a href="#"><u>GetImageHeight function</u></a>	<a href="#"><u>SetState</u></a>
<a href="#"><u>GetImageWidth function</u></a>	<a href="#"><u>SetVisu</u></a>
<a href="#"><u>GetMaxX macro</u></a>	<a href="#"><u>SHOW_</u></a>
<a href="#"><u>GetMaxY macro</u></a>	<a href="#"><u>SLD_DI</u></a>
<a href="#"><u>GetObjID macro</u></a>	<a href="#"><u>SLD_DF</u></a>
<a href="#"><u>GetObjNext macro</u></a>	<a href="#"><u>SLD_DF</u></a>
<a href="#"><u>GetObjType macro</u></a>	<a href="#"><u>SLD_DF</u></a>
<a href="#"><u>GetPageAddress macro</u></a>	<a href="#"><u>SLD_FC</u></a>
<a href="#"><u>GetPaletteChangeError function</u></a>	<a href="#"><u>SLD_HI</u></a>
<a href="#"><u>GetPixel function</u></a>	<a href="#"><u>SLD_SC</u></a>
<a href="#"><u>GetState macro</u></a>	<a href="#"><u>SLD_VE</u></a>
<a href="#"><u>GetTextHeight function</u></a>	<a href="#"><u>SldCrea</u></a>
<a href="#"><u>GetTextWidth function</u></a>	<a href="#"><u>SldDecl</u></a>
<a href="#"><u>Getting Started</u></a>	<a href="#"><u>SldDraw</u></a>
<a href="#"><u>GetTransparentColor macro</u></a>	<a href="#"><u>SldGetF</u></a>
<a href="#"><u>GetTransparentColorStatus macro</u></a>	<a href="#"><u>SldGetF</u></a>
<a href="#"><u>GetX macro</u></a>	<a href="#"><u>SldGetF</u></a>
<a href="#"><u>GetY macro</u></a>	<a href="#"><u>SldIncP</u></a>

<a href="#"><u>GFX_DISPLAY_BUFFER_LENGTH</u></a> macro	<a href="#"><u>SldMsgl</u></a>
<a href="#"><u>GFX_DISPLAY_BUFFER_START_ADDRESS</u></a> macro	<a href="#"><u>SldSetP</u></a>
<a href="#"><u>GFX_EPMP_CS1_BASE_ADDRESS</u></a> macro	<a href="#"><u>SldSetP</u></a>
<a href="#"><u>GFX_EPMP_CS1_MEMORY_SIZE</u></a> macro	<a href="#"><u>SldSetF</u></a>
<a href="#"><u>GFX_EPMP_CS2_BASE_ADDRESS</u></a> macro	<a href="#"><u>SldTran</u></a>
<a href="#"><u>GFX_EPMP_CS2_MEMORY_SIZE</u></a> macro	<a href="#"><u>Slider S</u></a>
<a href="#"><u>GFX_EXTDATA</u></a> structure	<a href="#"><u>SLIDER</u></a>
<a href="#"><u>GFX_Font_GetAntiAliasType</u></a> macro	<a href="#"><u>Slider/S</u></a>
<a href="#"><u>GFX_Font_SetAntiAliasType</u></a> macro	<a href="#"><u>SOLID_</u></a>
<a href="#"><u>GFX_free</u></a> macro	<a href="#"><u>ST_CEI</u></a>
<a href="#"><u>GFX_GCLK_DIVIDER</u></a> macro	<a href="#"><u>ST_DIS</u></a>
<a href="#"><u>GFX_GRADIENT_STYLE</u></a> structure	<a href="#"><u>ST_DR/</u></a>
<a href="#"><u>GFX_GRADIENT_TYPE</u></a> enumeration	<a href="#"><u>ST_FRA/</u></a>
<a href="#"><u>GFX_IMAGE_HEADER</u></a> structure	<a href="#"><u>ST_HID</u></a>
<a href="#"><u>GFX_LCD_CSTN</u></a> macro	<a href="#"><u>ST_RIG</u></a>
<a href="#"><u>GFX_LCD_MSTN</u></a> macro	<a href="#"><u>ST_UPI</u></a>
<a href="#"><u>GFX_LCD_OFF</u></a> macro	<a href="#"><u>Starting</u></a>
<a href="#"><u>GFX_LCD_TFT</u></a> macro	<a href="#"><u>Static Te</u></a>
<a href="#"><u>GFX_LCD_TYPE</u></a> macro	<a href="#"><u>Static Te</u></a>
<a href="#"><u>GFX_malloc</u></a> macro	<a href="#"><u>STATIC</u></a>
<a href="#"><u>GFX_PICTAIL_LCC</u></a> macro	<a href="#"><u>StCreat</u></a>
<a href="#"><u>GFX_PICTAIL_V3</u></a> macro	<a href="#"><u>StDraw</u></a>
<a href="#"><u>GFX_PICTAIL_V3E</u></a> macro	<a href="#"><u>StGetTe</u></a>
<a href="#"><u>GFX_RESOURCE</u></a> enumeration	<a href="#"><u>STN_DI</u></a>
<a href="#"><u>GFX_TRANSITION_DIRECTION</u></a> enumeration	<a href="#"><u>STN_DI</u></a>
<a href="#"><u>GFX_TRANSITION_TYPE</u></a> enumeration	<a href="#"><u>STN_DI</u></a>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_DMA</u></a> macro	<a href="#"><u>STN_DI</u></a>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</u></a> macro	<a href="#"><u>StSetTe</u></a>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_S1D13517</u></a> macro	<a href="#"><u>StTrans</u></a>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_SSD1926</u></a> macro	<a href="#"><u>Style Sc</u></a>
<a href="#"><u>GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q</u></a> macro	<a href="#"><u>SwitchC</u></a>
<a href="#"><u>GFX_USE_DISPLAY_PANEL_TFT_640480_8_E</u></a> macro	<a href="#"><u>SwitchC</u></a>
<a href="#"><u>GFX_USE_DISPLAY_PANEL_TFT_800480_33_E</u></a> macro	

[GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E](#) macro

[GFXExecutePendingTransition](#) function

[GFXGetPageOriginAddress](#) function

[GFXGetPageXYAddress](#) function

[GFXIsTransitionPending](#) function

[GFXSetupTransition](#) function

[GFXTransition](#) function

[GOL Global Variables](#)

[GOL Messages](#)

[GOL Objects](#)

[GOL\\_EMBOSS\\_SIZE](#) macro

[GOL\\_MSG](#) structure

[GOL\\_OBJ\\_TYPE](#) enumeration

[GOL\\_SCHEME](#) structure

[GOLAddObject](#) function

[GOLCanBeFocused](#) function

[GOLCreateScheme](#) function

[GOLDeleteObject](#) function

[GOLDeleteObjectByID](#) function

[GOLDraw](#) function

[GOLDrawCallback](#) function

[GOLDrawComplete](#) macro

[GOLFindObject](#) function

[GOLFontDefault](#) variable

[GOLFree](#) function

[GOLGetFocus](#) macro

[GOLGetFocusNext](#) function

[GOLGetFocusPrev](#) function

[GOLGetList](#) macro

[GOLGetScheme](#) macro

[GOLGetSchemeDefault](#) macro

[GOLInit](#) function

T

[TE\\_DEI](#)

[TE\\_DIS](#)

[TE\\_DR/](#)

[TE\\_ECH](#)

[TE\\_EN](#)

[TE\\_HID](#)

[TE\\_KEY](#)

[TE\\_SPA](#)

[TE\\_UPI](#)

[TE\\_UPI](#)

[TeAddC](#)

[TeClear](#)

[TeCreat](#)

[TeCreat](#)

[TeDelKe](#)

[TeDraw](#)

[TeGetBi](#)

[TeGetKe](#)

[TeIsKey](#)

[TeMsgD](#)

[TeSetBu](#)

[TeSetKe](#)

[TeSetKe](#)

[TeSpace](#)

[TeTrans](#)

[Text Ent](#)

[Text Fur](#)

[TextEnt](#)

[TEXTEN](#)

[THICK\\_](#)

[TRANS](#)

[GOLMsg function](#)  
[GOLMsgCallback function](#)  
[GOLNewList macro](#)  
[GOLPanelDraw macro](#)  
[GOLPanelDrawTsk function](#)  
[GOLRedraw macro](#)  
[GOLRedrawRec function](#)  
[GOLSchemeDefault variable](#)  
[GOLSetFocus function](#)  
[GOLSetList macro](#)  
[GOLSetScheme macro](#)  
[GOLTwoTonePanelDrawTsk function](#)  
[Gradient](#)  
[Gradient Bar Rendering](#)  
[Graphic Cursor](#)  
[Graphics Library Configuration](#)  
[Graphics Object Layer](#)  
[Graphics Object Layer API](#)  
[Graphics Object Layer Configuration](#)  
[Graphics Object Selection](#)  
[Graphics PICTail Used](#)  
[Graphics Primitive Layer](#)  
[Graphics Primitive Layer API](#)  
[Graphics Primitive Layer Configuration](#)  
[GraphicsConfig.h Example](#)  
[GRAY0 macro](#)  
[GRAY1 macro](#)  
[GRAY2 macro](#)  
[GRAY3 macro](#)  
[GRAY4 macro](#)  
[GRAY5 macro](#)  
[GRAY6 macro](#)  
[GREEN macro](#)

[Transiti](#)  
[Transpa](#)  
[TRANS](#)  
[TRANS](#)  
[Transpa](#)  
[Transpa](#)

## U

[UPDATI](#)  
[UPDATI](#)  
[UPDATI](#)  
[UpdateI](#)  
[USE\\_16](#)  
[USE\\_8E](#)  
[USE\\_AI](#)  
[USE\\_AI](#)  
[USE\\_AI](#)  
[USE\\_AI](#)  
[USE\\_BI](#)  
[USE\\_BI](#)  
[USE\\_BI](#)  
[USE\\_BI](#)  
[USE\\_BI](#)  
[USE\\_CI](#)  
[USE\\_CI](#)  
[USE\\_CI](#)  
[USE\\_DI](#)  
[USE\\_DI](#)  
[USE\\_EI](#)  
[USE\\_FC](#)  
[USE\\_FC](#)  
[USE\\_FC](#)

<a href="#"><u>Grid</u></a>	<a href="#"><u>USE_G</u></a>
<a href="#"><u>Grid Item States</u></a>	<a href="#"><u>USE_G</u></a>
<a href="#"><u>Grid States</u></a>	<a href="#"><u>USE_G</u></a>
<a href="#"><u>GRID structure</u></a>	<a href="#"><u>USE_G</u></a>
<a href="#"><u>GRID_DISABLED macro</u></a>	<a href="#"><u>USE_KI</u></a>
<a href="#"><u>GRID_DRAW_ALL macro</u></a>	<a href="#"><u>USE_LI</u></a>
<a href="#"><u>GRID_DRAW_ITEMS macro</u></a>	<a href="#"><u>USE_M</u></a>
<a href="#"><u>GRID_FOCUSED macro</u></a>	<a href="#"><u>USE_M</u></a>
<a href="#"><u>GRID_HIDE macro</u></a>	<a href="#"><u>USE_N</u></a>
<a href="#"><u>GRID_OUT_OF_BOUNDS macro</u></a>	<a href="#"><u>USE_P/</u></a>
<a href="#"><u>GRID_SHOW_BORDER_ONLY macro</u></a>	<a href="#"><u>USE_P/</u></a>
<a href="#"><u>GRID_SHOW_FOCUS macro</u></a>	<a href="#"><u>USE_PI</u></a>
<a href="#"><u>GRID_SHOW_LINES macro</u></a>	<a href="#"><u>USE_PI</u></a>
<a href="#"><u>GRID_SHOW_SEPARATORS_ONLY macro</u></a>	<a href="#"><u>USE_R/</u></a>
<a href="#"><u>GRID_SUCCESS macro</u></a>	<a href="#"><u>USE_R/</u></a>
<a href="#"><u>GridClearCellState function</u></a>	<a href="#"><u>USE_SI</u></a>
<a href="#"><u>GridCreate function</u></a>	<a href="#"><u>USE_S</u></a>
<a href="#"><u>GridDraw function</u></a>	<a href="#"><u>USE_T/</u></a>
<a href="#"><u>GridFreeItems function</u></a>	<a href="#"><u>USE_T/</u></a>
<a href="#"><u>GridGetCell function</u></a>	<a href="#"><u>USE_T/</u></a>
<a href="#"><u>GridGetFocusX macro</u></a>	<a href="#"><u>USE_UI</u></a>
<a href="#"><u>GridGetFocusY macro</u></a>	<a href="#"><u>USE_W</u></a>
<a href="#"><u>GRIDITEM structure</u></a>	<a href="#"><u>Using M</u></a>
<a href="#"><u>GRIDITEM_DRAW macro</u></a>	<a href="#"><u>Applicat</u></a>
<a href="#"><u>GRIDITEM_IS_BITMAP macro</u></a>	<a href="#"><u>Using P</u></a>
<a href="#"><u>GRIDITEM_IS_TEXT macro</u></a>	<a href="#"><u>Blocking</u></a>
<a href="#"><u>GRIDITEM_SELECTED macro</u></a>	
<a href="#"><u>GRIDITEM_TEXTBOTTOM macro</u></a>	<b>W</b>
<a href="#"><u>GRIDITEM_TEXTLEFT macro</u></a>	<a href="#"><u>WHITE</u></a>
<a href="#"><u>GRIDITEM_TEXTRIGHT macro</u></a>	<a href="#"><u>Window</u></a>
<a href="#"><u>GRIDITEM_TEXTTOP macro</u></a>	<a href="#"><u>Window</u></a>
<a href="#"><u>GridMsgDefault function</u></a>	<a href="#"><u>WINDO</u></a>
<a href="#"><u>GridSetCell function</u></a>	<a href="#"><u>WND_C</u></a>

[WND\\_D](#)

[WND\\_D](#)

[WND\\_D](#)

[WND\\_F](#)

[WND\\_H](#)

[WND\\_T](#)

[WndCre](#)

[WndDra](#)

[WndGe](#)

[WndSet](#)

[WndTra](#)

**X**

[XCHAR](#)

**Y**

[YELLOW](#)

[Contents](#) | [Index](#)



# **Microchip Graphics Library**

## Topics

[Demo Projects](#)

## Topics

Name	Description
<a href="#">Demo Summary</a>	This is the current list of demo projects released with the Graphics Library.
<a href="#">Microchip Application Library Abbreviations</a>	Summary of Microchip Applications Library Abbreviations used.
<a href="#">Demo Compatibility Matrix</a>	Refer to the Demo Compatibility matrix located in the <install directory>/Microchip/Graphics/Documents/Getting Started/Getting Started - Demo Compatibility Matrix.htm for details.

## [Demo Projects](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

[Library Architecture](#)

## Topics

Name	Description
<a href="#">Graphics Object Layer</a>	Describes the Graphics Object Layer (GOL) structure and its components.
<a href="#">Graphics Primitive Layer</a>	Describes the Graphics Primitive Layer structure and its components.
<a href="#">Display Device Driver Layer</a>	Describes the Display Driver Layer structure and its components.

## [Library Architecture](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Graphics Object Layer

---

## Topics

---

Name	Description
<a href="#">Object Rendering</a>	Describes the difference between the Blocking or Non-Blocking configuration when rendering Objects.

---

[Library Architecture](#) > [Graphics Object Layer](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

[Library API](#)

## Topics

Name	Description
<a href="#">Graphics Library Configuration</a>	The Graphics Library can be customized by adding or specifying the compile time options located in the application code named <a href="#">GraphicsConfig.h</a> or the HardwareProfile.h files. The following compile time options are supported by the library.
<a href="#">Graphics Object Layer API</a>	Description of Graphics Object Layer API.
<a href="#">Graphics Primitive Layer API</a>	Description of Graphics Primitive Layer API.
<a href="#">Display Device Driver Layer API</a>	Description of Display Device Driver Layer API.

## [Library API](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

### Graphics Library Configuration

## Topics

Name	Description
<a href="#">Graphics Object Layer Configuration</a>	The following compile time options configures the Graphics Library's Object Layer.
<a href="#">Graphics Primitive Layer Configuration</a>	The following compile time options configures the Graphics Library's Primitive Layer.
<a href="#">Display Device Driver Layer Configuration</a>	The following compile time options configures the Graphics Library's Display Device Driver Layer. The choices are based on the specific hardware used. See <a href="#">Hardware Profile</a> for more options.
<a href="#">Application Configuration</a>	
<a href="#">Hardware Profile</a>	These functions and macros are used to determine hardware profile settings on the chosen PIC microcontroller and hardware such as demo boards used.

## [Library API](#) > [Graphics Library Configuration](#)

## Topics

Graphics Object Layer Configuration

## Topics

Name	Description
<a href="#">Input Device Selection</a>	<p>The Graphics Library comes with two pre-defined user interface. These are the:</p> <ul style="list-style-type: none"><li>• Keyboard interface</li><li>• Touchscreen interface</li></ul> <p>Enabling one or both requires the declaration of the compile switch macros in the GraphicsConfig.h file.</p> <p>GOL widgets which supports the enabled input device's messages will respond to the user inputs.</p> <p>For Example: When using <a href="#">Button</a> Widget. <code>#define <a href="#">USE_TOUCHSCREEN</a></code> - will enable the buttons response to user touch on the button widget. The button will automatically be drawn with a pressed state when pressed and release state when user removes the touch on the screen.</p> <p>The compile option selects the input devices used by... <a href="#">more</a></p>
<a href="#">Focus Support Selection</a>	<p>This compile option allows keyboard input focus. <a href="#">GOLSetFocus()</a>, <a href="#">GOLGetFocus()</a>, <a href="#">GOLCanBeFocused()</a>, <a href="#">GOLGetFocusNext()</a> functions will be available. Focus is also changed by touch screen.</p> <p>The <a href="#">USE_FOCUS</a> option is located in the</p>

	GraphicsConfig.h header file.
<a href="#">Graphics Object Selection</a>	<p>These compile options selects objects used. Remove definitions for unused objects to reduce code size.</p> <p>The <a href="#">USE_GOL</a> and USE_OBJECT options are located in the GraphicsConfig.h header file. <a href="#">USE_GOL</a> should be included if any of the objects are to be used.</p>

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Input Device Selection Macros

## Input Device Selection

### Macros

Name	Description
<a href="#"><u>USE_KEYBOARD</u></a>	<p>Input devices macros that defines the messages that Objects will process. The following definitions indicate the usage of the different input devices:</p> <ul style="list-style-type: none"><li>• <code>USE_TOUCHSCREEN</code> - enables the touch screen support.</li><li>• <a href="#"><u>USE_KEYBOARD</u></a> - enables the key board support.</li></ul> <p>Define in GraphicsConfig.h</p>
<a href="#"><u>USE_TOUCHSCREEN</u></a>	<p>Input devices macros that defines the messages that Objects will process. The following definitions indicate the usage of the different input devices:</p> <ul style="list-style-type: none"><li>• <code>USE_TOUCHSCREEN</code> - enables the touch screen support.</li><li>• <a href="#"><u>USE_KEYBOARD</u></a> - enables the key board support.</li></ul> <p>Define in GraphicsConfig.h</p>

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Input Device Selection](#)

# Focus Support Selection Macros

## Focus Support Selection

---

### Macros

---

Name	Description
<a href="#"><u>USE_FOCUS</u></a>	Keyboard control on some objects can be used by enabling the GOL Focus (USE_FOCUS)support. Define this in GraphicsConfig.h

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Focus Support Selection](#)

# Graphics Object Selection Macros

Graphics Object Selection

## Macros

Name	Description
<a href="#">USE_ANALOGCLOCK</a>	Enable Analog Clock Object.
<a href="#">USE_BUTTON</a>	Enable <a href="#">Button</a> Object.
<a href="#">USE_BUTTON_MULTI_LINE</a>	Enable Multi- <a href="#">Line Button</a> Object
<a href="#">USE_CHECKBOX</a>	Enable <a href="#">Checkbox</a> Object.
<a href="#">USE_DIGITALMETER</a>	Enable DigitalMeter Object.
<a href="#">USE_EDITBOX</a>	Enable Edit Box Object.
<a href="#">USE_GROUPBOX</a>	Enable Group Box Object.
<a href="#">USE_LISTBOX</a>	Enable List Box Object.
<a href="#">USE_METER</a>	Enable <a href="#">Meter</a> Object.
<a href="#">USE_PICTURE</a>	Enable <a href="#">Picture</a> Object.
<a href="#">USE_PROGRESSBAR</a>	Enable Progress <a href="#">Bar</a> Object.
<a href="#">USE_RADIOBUTTON</a>	Enable Radio <a href="#">Button</a> Object.
<a href="#">USE_ROUNDIAL</a>	Enable <a href="#">Dial</a> Object.
<a href="#">USE_SLIDER</a>	Enable <a href="#">Slider</a> or <a href="#">Scroll Bar</a> Object.
<a href="#">USE_STATICTEXT</a>	Enable Static Text Object.

<a href="#"><u>USE_WINDOW</u></a>	Enable <a href="#"><u>Window</u></a> Object.
<a href="#"><u>USE_CUSTOM</u></a>	Enable Custom Control Object (an example to create customized Object).
<a href="#"><u>USE_GOL</u></a>	Enable Graphics Object Layer.
<a href="#"><u>USE_TEXTENTRY</u></a>	Enable TextEntry Object.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Object Layer Configuration](#) > [Graphics Object Selection](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

### Graphics Primitive Layer Configuration

## Topics

Name	Description
<a href="#">Image Compression Option</a>	These compile options to set if the images used for OutImage() are RLE compressed or IPU compressed.
<a href="#">Font Type Selection</a>	<p>This compile option selects if the support for unicode fonts, unsigned char or the default signed char type fonts.</p> <p>There are three types of font (characters) that can be used in the Graphics Library. This gives the user the option to implement multi-language application or use the default signed char type.</p>
<a href="#">Advanced Font Features Selection</a>	This compile option enables the advanced font features.
<a href="#">Gradient Bar Rendering</a>	This compile option enables the usage of the Gradient <a href="#">Bar</a> and <a href="#">Bevel</a> function in the Primitive Layer.
<a href="#">Transparent Color Feature in PutImage()</a>	This compile option enables the transparent color feature in <a href="#">PutImage()</a> .
<a href="#">Alpha Blend Option</a>	This compile option enables the Alpha-Blend feature in Primitive Layer.
<a href="#">External Memory</a>	see <a href="#">EXTERNAL_FONT_BUFFER_SIZE</a>

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Image Compression Option Macros

### Image Compression Option

#### Macros

Name	Description
<a href="#"><u>USE_COMP_IPU</u></a>	To enable support for DEFLATE compressed images for <a href="#"><u>PutImage()</u></a> . When this macro is enabled, the <a href="#"><u>PutImage()</u></a> function will be able to process images generated by the Graphics Resource Converter (GRC) that are compressed using the DEFLATE algorithm. <a href="#"><u>PutImage()</u></a> will need the IPU module of the Microchip Graphics Module to decompress. Enable this feature only when the driver features the IPU module (example: PIC24FJ2456DA210). Define this in GraphicsConfig.h
<a href="#"><u>USE_COMP_RLE</u></a>	To enable support for RLE compressed images for <a href="#"><u>PutImage()</u></a> . When this macro is enabled, the <a href="#"><u>PutImage()</u></a> function will be able to process images generated by the Graphics Resource Converter (GRC) that are RLE compressed. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Image Compression Option](#)

## Font Type Selection Macros

### Font Type Selection

#### Macros

Name	Description
<a href="#"><u>USE_MULTIBYTECHAR</u></a>	To enable support for unicode fonts, <a href="#"><u>USE_MULTIBYTECHAR</u></a> must be defined. This sets the <a href="#"><u>XCHAR</u></a> definition (0-2 <sup>16</sup> range). See <a href="#"><u>XCHAR</u></a> for details. Define this in GraphicsConfig.h
<a href="#"><u>USE_UNSIGNED_XCHAR</u></a>	To enable support for unsigned characters data type for fonts, <a href="#"><u>USE_UNSIGNED_XCHAR</u></a> must be defined. This sets the <a href="#"><u>XCHAR</u></a> definition (0-255 range). See <a href="#"><u>XCHAR</u></a> for details. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Font Type Selection](#)

## Advanced Font Features Selection Macros

### Advanced Font Features Selection

---

#### Macros

---

Name	Description
<a href="#"><u>USE_ANTIALIASED_FONTS</u></a>	To enable support for Anti-aliased fonts. Use this feature if the font table generated through the "Graphics Resource Converter" tool has the anti-aliasing enabled. Define this in GraphicsConfig.h

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Advanced Font Features Selection](#)

# Gradient Bar Rendering Macros

## Gradient Bar Rendering

---

### Macros

---

Name	Description
<a href="#"><u>USE_GRADIENT</u></a>	To enable support for Gradient bars and bevel primitives. Define this in GraphicsConfig.h.

---

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Gradient Bar Rendering](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Transparent Color Feature in PutImage() Macros

Transparent Color Feature in PutImage()

### Macros

Name	Description
<a href="#"><u>USE_TRANSPARENT_COLOR</u></a>	To enable support for transparent color in <a href="#"><u>PutImage()</u></a> . Enabling this macro enables the use of a transparent color (set by <a href="#"><u>TransparentColorEnable()</u></a> ) in rendering images by <a href="#"><u>PutImage()</u></a> . When a pixel in the image matches the transparent color set, the pixel is not rendered to the screen. This is useful in rendering rounded icons or images to the screen with a complex background. Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Transparent Color Feature in PutImage\(\)](#)

# Alpha Blend Option Macros

## Alpha Blend Option

### Macros

Name	Description
<a href="#"><u>USE_ALPHABLEND_LITE</u></a>	To enable support for Alpha Blending on the Primitive Layer. This feature is only limited on Alpha-Blending a <a href="#"><u>Bar()</u></a> of a specified color (set by <a href="#"><u>SetColor()</u></a> to the destination defined by the parameters of the <a href="#"><u>Bar()</u></a> function call. The Alpha level used is set by <a href="#"><u>SetAlpha()</u></a> . Define this in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Graphics Primitive Layer Configuration](#) > [Alpha Blend Option](#)

## Display Device Driver Layer Configuration Macros

### Display Device Driver Layer Configuration

#### Macros

Name	Description
<a href="#"><u>USE_ALPHABLEND</u></a>	To enable support for Alpha Blending. Use this feature only if the display driver used can support alpha blending. Define this in GraphicsConfig.h
<a href="#"><u>USE_DOUBLE_BUFFERING</u></a>	To enable support for double buffering. Use this feature only if the display driver used can support double buffering. Define this in GraphicsConfig.h
<a href="#"><u>GFX_LCD_TYPE</u></a>	Sets the type of display glass used. Define this in the Hardware Profile. <ul style="list-style-type: none"><li>• #define GFX_LCD_TYPE <a href="#"><u>GFX_LCD_TFT</u></a> - sets type TFT display</li><li>• #define GFX_LCD_TYPE <a href="#"><u>GFX_LCD_CSTN</u></a> - sets type color STN display</li><li>• #define GFX_LCD_TYPE <a href="#"><u>GFX_LCD_MSTN</u></a> - sets type mon STN display</li><li>• #define GFX_LCD_TYPE <a href="#"><u>GFX_LCD_OFF</u></a> - display is turned off</li></ul>

## STN\_DISPLAY\_WIDTH

Sets the STN glass data width.  
Define this in the Hardware Profile.

- #define STN\_DISPLAY\_WIDTH STN\_DISPLAY\_WIDTH\_4 - use 4-bit wide interface
- #define STN\_DISPLAY\_WIDTH STN\_DISPLAY\_WIDTH\_8 - Use 8-bit wide interface
- #define STN\_DISPLAY\_WIDTH STN\_DISPLAY\_WIDTH\_16 - Use 16-bit wide interface

---

## Library API > Graphics Library Configuration > Display Device Driver Layer Configuration

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## GFX\_LCD\_TYPE Macro Macros

GFX\_LCD\_TYPE Macro

### Macros

Name	Description
<a href="#">GFX_LCD_CSTN</a>	Type Color STN Display
<a href="#">GFX_LCD_MSTN</a>	Type Mono STN Display
<a href="#">GFX_LCD_OFF</a>	display is turned off
<a href="#">GFX_LCD_TFT</a>	Type TFT Display

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [GFX\\_LCD\\_TYPE Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## STN\_DISPLAY\_WIDTH Macro Macros

STN\_DISPLAY\_WIDTH Macro

### Macros

Name	Description
<a href="#">STN_DISPLAY_WIDTH_16</a>	display interface is 16 bits wide
<a href="#">STN_DISPLAY_WIDTH_4</a>	display interface is 4 bits wide
<a href="#">STN_DISPLAY_WIDTH_8</a>	display interface is 8 bits wide

[Library API](#) > [Graphics Library Configuration](#) > [Display Device Driver Layer Configuration](#) > [STN\\_DISPLAY\\_WIDTH Macro](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

### Application Configuration

## Topics

Name	Description
<a href="#">Configuration Setting</a>	<p>This selects the configuration of the library. When Non-blocking configuration is selected, state machine based rendering is used to perform object rendering. When blocking configuration is used, this line MUST be commented. In this case object rendering will not exit until the object is fully rendered. The <a href="#">USE_NONBLOCKING_CONFIG</a> option is located in the GraphicsConfig.h header file.</p>
<a href="#">Font Source Selection</a>	<p>Font data can be placed in multiple locations. Set these options in the GraphicsConfig.h header file.</p> <ul style="list-style-type: none"><li>- <a href="#">USE_FONT_FLASH</a> - Font in internal flash memory support. When placed in internal flash memory, it can further classified to be placed in program flash by adding <a href="#">USE_GFX_FONT_IN_PROGRAM_SECTION</a>.</li><li>- <a href="#">USE_FONT_EXTERNAL</a> - Font in external memory support. Use this for fonts located in external memory like SPI Flash or external memory mapped to Extended Data Space.</li></ul>
<a href="#">Image Source Selection</a>	<p>Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash</p>

	and external memory. The <a href="#">USE_BITMAP_FLASH</a> and <a href="#">USE_BITMAP_EXTERNAL</a> options are located in the GraphicsConfig.h header file.
<a href="#">Miscellaneous</a>	This contains miscellaneous macros and functions that can be redefined for various system support such as Operating System defined functions.
<a href="#">GraphicsConfig.h Example</a>	This is an example of the GraphicsConfig.h file implementation:

---

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Configuration Setting Macros

Configuration Setting

### Macros

Name	Description
<a href="#"><u>USE_NONBLOCKING_CONFIG</u></a>	Blocking and Non-Blocking configuration selection. To enable non-blocking configuration <code>USE_NONBLOCKING_CONFIG</code> must be defined. If this is not defined, blocking configuration is assumed. Define this in <code>GraphicsConfig.h</code>

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#) > [Configuration Setting](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Font Source Selection Macros

### Font Source Selection

#### Macros

Name	Description
<a href="#"><u>USE_FONT_FLASH</u></a>	<p>Font data can be placed in FLASH memory or from external memory. One is in FLASH memory and the other is in external memory. This macro enables the support for internal flash and external flash. This is configured in GraphicsConfig.</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_FONT_FLASH</u></a> enables flash memory support for internal flash.</li><li>• <a href="#"><u>USE_FONT_EXTERNAL</u></a> enables external memory support for external memory.</li></ul>
<a href="#"><u>USE_FONT_EXTERNAL</u></a>	<p>Font data can be placed in FLASH memory or from external memory. One is in FLASH memory and the other is in external memory. This macro enables the support for internal flash and external flash. This is configured in GraphicsConfig.</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_FONT_FLASH</u></a> enables flash memory support for internal flash.</li><li>• <a href="#"><u>USE_FONT_EXTERNAL</u></a> enables external memory support for external memory.</li></ul>
<a href="#"><u>USE_GFX_FONT_IN_PROGRAM_SECTION</u></a>	For XC16 or C30 build



## Image Source Selection Macros

### Image Source Selection

#### Macros

Name	Description
<a href="#"><u>USE_BITMAP_FLASH</u></a>	<p>Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash and external memory. Define this in GraphicsConfig.h</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_BITMAP_FLASH</u></a> - Images located in internal flash memory.</li><li>• <a href="#"><u>USE_BITMAP_EXTERNAL</u></a> - Images located in external memory (including external memory mapped to EDS)..</li></ul>
<a href="#"><u>USE_BITMAP_EXTERNAL</u></a>	<p>Similar to Font data bitmaps can also be placed in two locations. One is in FLASH memory and the other is from external memory. Defining one or both enables the support for bitmaps located in internal flash and external memory. Define this in GraphicsConfig.h</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_BITMAP_FLASH</u></a> - Images located in internal flash memory.</li><li>• <a href="#"><u>USE_BITMAP_EXTERNAL</u></a> -</li></ul>

Images located in external memory  
(including external memory  
mapped to EDS)..

---

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Image Source Selection](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Miscellaneous Macros

### Miscellaneous

#### Macros

Name	Description
<a href="#"><u>USE_BITMAP_NO_PADDING_LINE</u></a>	When this macro is enabled, bitmap images used has no padding. Define this in GraphicsConfig.h. 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 bitmaps are padded, this means that each horizontal line will start on a byte boundary. Unpadded bitmaps allows the least resource space for a bitmap. Unpadded bitmaps also allows support for display controllers with windowing and auto-increment features.
<a href="#"><u>USE_PALETTE_EXTERNAL</u></a>	Palettes can also be specified to reside in external memory similar to fonts and images. Use this when the palette is located in external memory. Define this in GraphicsConfig.h
<a href="#"><u>USE_PALETTE</u></a>	Using Palettes, different colors

	can be used with the same bit depth. Define this in GraphicsConfig.h
<u>COLOR_DEPTH</u>	Specifies the color depth used in the application defined in GraphicsConfig.h.
<u>GFX_free</u>	When using Operating Systems (OS), define the OS specific malloc() and free() functions for compatibility with the OS based systems. Define these in GraphicsConfig.h
<u>GFX_malloc</u>	When using Operating Systems (OS), define the OS specific malloc() and free() functions for compatibility with the OS based systems. Define these in GraphicsConfig.h

[Library API](#) > [Graphics Library Configuration](#) > [Application Configuration](#)  
> [Miscellaneous](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Hardware Profile

## Topics

Name	Description
<a href="#">PMP Interface</a>	Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).
<a href="#">Development Platform Used</a>	Specifies the Development Platform used for the Microchip Graphics Library demos.
<a href="#">Graphics PICtail Used</a>	Specifies the Graphics PICtail Display Panel used.
<a href="#">Display Controller Used</a>	Specifies the controller used in the Graphics Library supplied demo.
<a href="#">Display Panel Used</a>	Specifies the Graphics Display Panel used.
<a href="#">Device Driver Options</a>	<p>The options Graphics Hardware Platform, DISPLAY_CONTROLLER and DISPLAY_PANEL are specific to the hardware used. The Graphics Hardware Platform selects the Graphics PICtail™ Plus Board version, PIC24FJ256DA210 Development Board or any other Microchip demo boards for the Graphics Library. Currently there are two Graphics PICtail™ Plus Board versions supported as shown in the <a href="#">Getting Started</a> section.</p> <p>The rest of the settings are used to specify the the display parameters when using an RGB type display controller such as</p>

	SSD1906 and SSD1926 from Solomon Systech. The table below summarizes the generic parameters found in RGB type display controllers and when each type is used.... <a href="#">more</a>
<a href="#">HardwareProfile.h Example</a>	This is an example of the HardwareProfile.h file implementation:

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## PMP Interface Macros

### PMP Interface

#### Macros

Name	Description
<a href="#"><u>USE_8BIT_PMP</u></a>	<p>Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_8BIT_PMP</u></a> - Use 8-bit interface to PMP or EPMP</li><li>• <a href="#"><u>USE_16BIT_PMP</u></a> - Use 16-bit interface to PMP or EPMP</li></ul>
<a href="#"><u>USE_16BIT_PMP</u></a>	<p>Specifies the interface type to the Parallel Master Port (PMP) or Enhanced Parallel Master Port (EPMP).</p> <ul style="list-style-type: none"><li>• <a href="#"><u>USE_8BIT_PMP</u></a> - Use 8-bit interface to PMP or EPMP</li><li>• <a href="#"><u>USE_16BIT_PMP</u></a> - Use 16-bit interface to PMP or EPMP</li></ul>

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [PMP Interface](#)

## Development Platform Used Macros

### Development Platform Used

#### Macros

Name	Description
<a href="#">EXPLORER_16</a>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p> <ul style="list-style-type: none"><li>• <a href="#">EXPLORER_16</a> - Using the Explorer 16 Development Board (DM240001).</li><li>• <a href="#">PIC24FJ256DA210_DEV_BOARD</a> - Using the PIC24FJ256DA210 Development Board (DM240001).</li><li>• <a href="#">MEB_BOARD</a> - Using the Multimedia Expansion Board (DM320005).</li><li>• <a href="#">PIC_SK</a> - Using PIC32 or dsPIC32 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).</li></ul>
<a href="#">PIC24FJ256DA210_DEV_BOARD</a>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p> <ul style="list-style-type: none"><li>• <a href="#">EXPLORER_16</a> - Using the Explorer 16 Development Board (DM240001).</li><li>• <a href="#">PIC24FJ256DA210_DEV_BOARD</a></li></ul>

	<ul style="list-style-type: none"> <li>- Using the PIC24FJ256DA200 Development Board (DM240001).</li> <li>• <a href="#">MEB_BOARD</a> - Using the Multimedia Expansion Board (DM320005).</li> <li>• <a href="#">PIC_SK</a> - Using PIC32 or dsPIC33 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).</li> </ul>
<p><a href="#">MEB_BOARD</a></p>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p> <ul style="list-style-type: none"> <li>• EXPLORER_16 - Using the Explorer 16 Development Board (DM240001).</li> <li>• <a href="#">PIC24FJ256DA210_DEV_BOARD</a> - Using the PIC24FJ256DA200 Development Board (DM240001).</li> <li>• <a href="#">MEB_BOARD</a> - Using the Multimedia Expansion Board (DM320005).</li> <li>• <a href="#">PIC_SK</a> - Using PIC32 or dsPIC33 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).</li> </ul>
<p><a href="#">PIC_SK</a></p>	<p>Specifies the Development Platform used for the Microchip Graphics Library demos.</p>

- EXPLORER\_16 - Using the Explorer 16 Development Board (DM240001).
- [PIC24FJ256DA210\\_DEV\\_BOARD](#) - Using the PIC24FJ256DA210 Development Board (DM240001).
- [MEB\\_BOARD](#) - Using the Multimedia Expansion Board (DM320005).
- [PIC\\_SK](#) - Using PIC32 or dsPIC32 Starter Kit (examples: PIC32 Starter Kit (DM320001), PIC32 USB Starter Kit II (DM320003), PIC32 Ethernet Starter Kit (DM320004)).

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Development Platform Used](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Graphics PICtail Used Macros

Graphics PICtail Used

### Macros

Name	Description
<a href="#"><u>GFX_PICTAIL_LCC</u></a>	<p>Specifies the Graphics PICtail Display Panel used.</p> <ul style="list-style-type: none"><li>• <a href="#"><u>GFX_PICTAIL_V3</u></a> - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)</li><li>• <a href="#"><u>GFX_PICTAIL_V3E</u></a> - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)</li><li>• <a href="#"><u>GFX_PICTAIL_LCC</u></a> - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)</li></ul>
<a href="#"><u>GFX_PICTAIL_V3</u></a>	<p>Specifies the Graphics PICtail Display Panel used.</p> <ul style="list-style-type: none"><li>• <a href="#"><u>GFX_PICTAIL_V3</u></a> - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)</li><li>• <a href="#"><u>GFX_PICTAIL_V3E</u></a> - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)</li><li>• <a href="#"><u>GFX_PICTAIL_LCC</u></a> - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)</li></ul>
<a href="#"><u>GFX_PICTAIL_V3E</u></a>	<p>Specifies the Graphics PICtail Display Panel</p>

used.

- `GFX_PICTAIL_V3` - Graphics LCD Controller PICtail Plus SSD1926 Board (AC164127-5)
- [`GFX\_PICTAIL\_V3E`](#) - Graphics LCD Controller PICtail Plus S1D13517 Board (AC164127-7)
- [`GFX\_PICTAIL\_LCC`](#) - Low Cost Controllerless (LCC) Graphics PICtail Plus Board (AC164144)

---

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Graphics PICtail Used](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Display Controller Used Macros

### Display Controller Used

#### Macros

Name	Description
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_DMA</u></a>	<p>Specifies the c supplied demc</p> <ul style="list-style-type: none"><li>• <a href="#"><u>GFX_USE</u></a> the PIC32</li><li>• <a href="#"><u>GFX_USE</u></a> - Use the the PIC M Family)</li><li>• <a href="#"><u>GFX_USE</u></a> Using the Controller</li><li>• <a href="#"><u>GFX_USE</u></a> Using the</li></ul>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_MCHP_DA210</u></a>	<p>Specifies the c supplied demc</p> <ul style="list-style-type: none"><li>• <a href="#"><u>GFX_USE</u></a> the PIC32</li><li>• <a href="#"><u>GFX_USE</u></a> - Use the the PIC M Family)</li><li>• <a href="#"><u>GFX_USE</u></a> Using the Controller</li><li>• <a href="#"><u>GFX_USE</u></a></li></ul>

	Using the
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_S1D13517</u></a>	<p>Specifies the c supplied demc</p> <ul style="list-style-type: none"> <li>• GFX_USE the PIC32</li> <li>• <a href="#"><u>GFX_USE</u></a> - Use the the PIC M Family)</li> <li>• <a href="#"><u>GFX_USE</u></a> Using the Controller</li> <li>• <a href="#"><u>GFX_USE</u></a> Using the</li> </ul>
<a href="#"><u>GFX_USE_DISPLAY_CONTROLLER_SSD1926</u></a>	<p>Specifies the c supplied demc</p> <ul style="list-style-type: none"> <li>• GFX_USE the PIC32</li> <li>• <a href="#"><u>GFX_USE</u></a> - Use the the PIC M Family)</li> <li>• <a href="#"><u>GFX_USE</u></a> Using the Controller</li> <li>• <a href="#"><u>GFX_USE</u></a> Using the</li> </ul>

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Display Panel Used Macros

Display Panel Used

## Macros

Name	Description
<a href="#">GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q</a>	Specifies the display panel used for the PH480272T_005_I11Q display panel. <ul style="list-style-type: none"><li>• <a href="#">GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q</a></li><li>• <a href="#">GFX_USE_DISPLAY_PANEL_TFT_640480_8_E</a></li><li>• <a href="#">GFX_USE_DISPLAY_PANEL_TFT_800480_33_E</a></li></ul>
<a href="#">GFX_USE_DISPLAY_PANEL_TFT_640480_8_E</a>	Specifies the display panel used for the TFT_640480_8_E display panel. <ul style="list-style-type: none"><li>• <a href="#">GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q</a></li><li>• <a href="#">GFX_USE_DISPLAY_PANEL_TFT_640480_8_E</a></li><li>• <a href="#">GFX_USE_DISPLAY_PANEL_TFT_800480_33_E</a></li></ul>
<a href="#">GFX_USE_DISPLAY_PANEL_TFT_800480_33_E</a>	Specifies the display panel used for the TFT_800480_33_E display panel. <ul style="list-style-type: none"><li>• <a href="#">GFX_USE_DISPLAY_PANEL_PH480272T_005_I11Q</a></li><li>• <a href="#">GFX_USE_DISPLAY_PANEL_TFT_640480_8_E</a></li><li>• <a href="#">GFX_USE_DISPLAY_PANEL_TFT_800480_33_E</a></li></ul>

[GFX\\_USE\\_DISPLAY\\_PANEL\\_TFT\\_G240320LTSW\\_118W\\_E](#)

Specifi

- [GI](#)  
VC
- [GI](#)  
W
- [GI](#)  
- 3
- [GI](#)  
inc
- [GI](#)  
VC
- [GI](#)  
W

[Library API](#) > [Graphics Library Configuration](#) > [Hardware Profile](#) > [Display Panel Used](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Device Driver Options Macros

### Device Driver Options

#### Macros

Name	Description
<u><a href="#">DISP_DATA_WIDTH</a></u>	<p>Defines the display controller's physical interface to the display panel. Valid Values:</p> <ul style="list-style-type: none"><li>• 1, 4, 8, 16, 18, 24</li><li>• 1, 4, 8 are usually used in MSTN and CSTN displays</li><li>• 16, 18 and 24 are usually used in TFT displays.</li></ul>
<u><a href="#">DISP_ORIENTATION</a></u>	<p>Defines the display rotation with respect to its native orientation. For example, if the display has a resolution specifications that says 240x320 (QVGA), the display is natively in portrait mode. If the application uses the display in landscape mode (320x240), then the orientation must be defined as 90 or 180 degree rotation. Graphics Library will calculate the actual pixel location to rotate the contents of the screen. So when users view the display, the image on the screen will come out in the correct orientation. Valid values:</p> <ul style="list-style-type: none"><li>• 0 : display in its native</li></ul>

	<p>orientation</p> <ul style="list-style-type: none"> <li>• 90 : rotated 90 degrees clockwise... <a href="#">more</a></li> </ul>
<u>DISP_HOR_RESOLUTION</u>	<p>Defines the native horizontal dimension of the screen. This is the horizontal pixel count given by the display's data sheet. For example a 320x240 display will have DISP_HOR_RESOLUTION of 320.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• dependent on the display glass resolution used.</li> </ul>
<u>DISP_VER_RESOLUTION</u>	<p>Defines the native vertical dimension of the screen. This is the vertical pixel count given by the display's data sheet. For example a 320x240 display will have DISP_VER_RESOLUTION of 240.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• dependent on the display glass resolution used.</li> </ul>
<u>DISP_HOR_FRONT_PORCH</u>	<p>Defines the horizontal front porch. <u>DISP_HOR_BACK_PORCH</u> + DISP_HOR_FRONT_PORCH + <u>DISP_HOR_PULSE_WIDTH</u> makes up the horizontal blanking period. Value used will be based on the display panel used.</p>
<u>DISP_HOR_BACK_PORCH</u>	<p>Defines the horizontal back porch. DISP_HOR_BACK_PORCH +</p>

	<p><a href="#"><u>DISP_HOR_FRONT_PORCH</u></a> + <a href="#"><u>DISP_HOR_PULSE_WIDTH</u></a> makes up the horizontal blanking period. Value used will be based on the display panel used.</p>
<p><a href="#"><u>DISP_VER_FRONT_PORCH</u></a></p>	<p>Defines the vertical front porch. <a href="#"><u>DISP_VER_BACK_PORCH</u></a> + <a href="#"><u>DISP_VER_FRONT_PORCH</u></a> + <a href="#"><u>DISP_VER_PULSE_WIDTH</u></a> makes up the vertical blanking period. Value used will be based on the display panel used.</p>
<p><a href="#"><u>DISP_VER_BACK_PORCH</u></a></p>	<p>Defines the vertical back porch. <a href="#"><u>DISP_VER_BACK_PORCH</u></a> + <a href="#"><u>DISP_VER_FRONT_PORCH</u></a> + <a href="#"><u>DISP_VER_PULSE_WIDTH</u></a> makes up the vertical blanking period. Value used will be based on the display panel used.</p>
<p><a href="#"><u>DISP_HOR_PULSE_WIDTH</u></a></p>	<p>Defines the horizontal sync signal pulse width in pixels. Value used will be based on the display panel used.</p>
<p><a href="#"><u>DISP_VER_PULSE_WIDTH</u></a></p>	<p>Defines the vertical sync signal pulse width in lines. Value used will be based on the display panel used.</p>
<p><a href="#"><u>DISP_INV_LSHIFT</u></a></p>	<p>Indicates that the color data is sampled in the falling edge of the pixel clock.</p>

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

### Graphics Object Layer API

## Topics

Name	Description
<a href="#">GOL Objects</a>	The Graphics Object Layer (GOL) contains the Advanced Graphics Objects or commonly known as widgets.
<a href="#">Object States</a>	Objects rendered on the display are based on their current Property States and the Drawing States.
<a href="#">Object Management</a>	This section describes the API functions and macros that are used to create, maintain and render individual and list of objects.
<a href="#">GOL Messages</a>	The library provides an interface to accept messages from the input devices.
<a href="#">Style Scheme</a>	All objects uses a style scheme structure that defines the font and colors used.
<a href="#">GOL Global Variables</a>	Graphics Object Layer global variables.

### [Library API](#) > [Graphics Object Layer API](#)

# GOL Objects Enumerations

## GOL Objects

---

### Enumerations

---

Name	Description
<a href="#"><u>GOL_OBJ_TYPE</u></a>	This structure defines the Object types used in the library.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Modules

GOL Objects

### Modules

Name	Description
<a href="#">Analog Clock</a>	Analog Clock is an Object that emulates an analog clock with moving hands. It can be used with or without a bitmap image as the background source.
<a href="#">Button</a>	Button is an Object that emulates a press and release effect when operated upon.
<a href="#">Chart</a>	Chart is an Object that draws a bar chart or a pie chart representation of a single data or series of data.
<a href="#">Checkbox</a>	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.
<a href="#">Round Dial</a>	Dial is an Object that can be used to display emulate a turn dial that can both go in clockwise or counterclockwise.
<a href="#">Digital Meter</a>	DigitalMeter 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. A limitation of this Object is that the font used should have

	equal character widths.
<u>Edit Box</u>	Edit Box is is an Object that emulates a cell or a text area that can be edited dynamically.
<u>Grid</u>	Grid is an Object that draws a grid on the screen with each cell capable of displaying an image or a string.
<u>Group Box</u>	Group Box is an Object that can be used to group Objects together in the screen.
<u>List Box</u>	List Box is an Object that defines a scrollable area where items are listed. User can select a single item or set of items.
<u>Meter</u>	Meter is an Object that can be used to graphically display a sampled input.
<u>Picture Control</u>	Picture is an Object that can be used to transform a bitmap to be an Object in the screen and have control on the bitmap rendering. This object can be used to create animation using a series of bitmaps.
<u>Progress Bar</u>	Progress <u>Bar</u> is an Object that can be used to display the progress of a task such as a file download or transfer.
<u>Radio Button</u>	Radio <u>Button</u> 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.
<u>Slider/Scroll Bar</u>	Slider or Scrollbar is an Object that can be used to display a value or scrolling location

	in a predefined area.
<a href="#">Static Text</a>	Static Text is an Object that can be used to display a single or multi-line string of text in a predefined location.
<a href="#">Text Entry</a>	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.
<a href="#">Window</a>	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 <a href="#">Bar</a> . It also has additional controls for both Title <a href="#">Bar</a> and Client Area.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Objects Structures

## GOL Objects

---

### Structures

---

Name	Description
<a href="#"><u>OBJ_HEADER</u></a>	This structure defines the first nine fields of the Objects structure. This allows generic operations on library Objects.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Objects Types

## GOL Objects

### Types

Name	Description
<a href="#"><u>DRAW_FUNC</u></a>	object draw function pointer typedef
<a href="#"><u>FREE_FUNC</u></a>	object free function pointer typedef
<a href="#"><u>MSG_DEFAULT_FUNC</u></a>	object default message function pointer typedef
<a href="#"><u>MSG_FUNC</u></a>	object message function pointer typedef

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Analog Clock Functions

## Analog Clock

### Functions

	Name	Description
	<a href="#"><u>AcCreate</u></a>	This function creates an Analog Clock 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.
	<a href="#"><u>AcDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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.
	<a href="#"><u>AcSetHour</u></a>	Sets the hour value of the analog clock.
	<a href="#"><u>AcSetMinute</u></a>	Sets the minute value of the analog clock.
	<a href="#"><u>AcSetSecond</u></a>	Sets the second value of the analog clock.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

## Analog Clock Structures

### Analog Clock

#### Structures

Name	Description
<a href="#">ANALOGCLOCK</a>	<p>Defines the parameters required for a clock Object. The following relationships of the parameters determines the general shape of the clock:</p> <ol style="list-style-type: none"><li>1. centerx and centery determine the middle of the clock.</li><li>2. radius defines the radius of the clock.</li><li>3. *pImage points to the background image for the analog clock.</li><li>4. *pBitmap points to the background image for the analog clock.</li></ol>

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

## Topics

Analog Clock

---

## Topics

Name	Description
<a href="#">Analog Clock States</a>	List of Analog Clock bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Analog Clock

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Analog Clock States Macros

### Analog Clock States

#### Macros

Name	Description
<a href="#">AC_DRAW</a>	Bit to indicate button must be redrawn.
<a href="#">AC_DISABLED</a>	Bit for disabled state.
<a href="#">AC_HIDE</a>	Bit to indicate button must be removed from screen.
<a href="#">AC_PRESSED</a>	Bit for press state.
<a href="#">AC_TICK</a>	Bit to tick second hand
<a href="#">UPDATE_HOUR</a>	Bit to indicate hour hand must be redrawn
<a href="#">UPDATE_MINUTE</a>	Bit to indicate minute hand must be redrawn
<a href="#">UPDATE_SECOND</a>	Bit to indicate minute hand must be redrawn

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Analog Clock](#)  
> [Analog Clock States](#)

# Button Functions

## Button

### Functions

	Name	Description
	<a href="#"><u>BtnCreate</u></a>	This function creates a <a href="#"><u>BUTTON</u></a> 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.
	<a href="#"><u>BtnDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>The text on the face of the button is drawn on top of the bitmap. Text is always rendered centered on the face of the button.</p> <p>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... <a href="#"><u>more</u></a></p>
	<a href="#"><u>BtnSetText</u></a>	This function sets the string used for the object.
	<a href="#"><u>BtnMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are

		supported:
	<a href="#"><u>BtnTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Button Macros

### Button

#### Macros

Name	Description
<a href="#">BtnGetText</a>	This macro returns the address of the current text string used for the object.
<a href="#">BtnGetBitmap</a>	This macro returns the location of the currently used bitmap for the object.
<a href="#">BtnSetBitmap</a>	This macro sets the bitmap used in the object. The size of the bitmap must match the face of the button.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#)

## Button Structures

### Button

### Structures

Name	Description
<a href="#">BUTTON</a>	<p>Defines the parameters required for a button Object. The following relationships of the parameters determines the general shape of the button:</p> <ol style="list-style-type: none"><li>1. Width is determined by right - left.</li><li>2. Height is determined by top - bottom.</li><li>3. Radius - specifies if the button will have a rounded edge. If zero then the button will have sharp (cornered) edge.</li><li>4. If <math>2 * \text{radius} = \text{height} = \text{width}</math>, the button is a circular button.</li></ol>

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#)

## Topics

Button

---

## Topics

Name	Description
<a href="#">Button States</a>	List of Button bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Button

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Button States Macros

### Button States

#### Macros

Name	Description
<a href="#"><u>BTN_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>BTN_DRAW</u></a>	Bit to indicate button must be redrawn.
<a href="#"><u>BTN_DRAW_FOCUS</u></a>	Bit to indicate focus must be redrawn.
<a href="#"><u>BTN_FOCUSED</u></a>	Bit for focus state.
<a href="#"><u>BTN_HIDE</u></a>	Bit to indicate button must be removed from screen.
<a href="#"><u>BTN_PRESSED</u></a>	Bit for press state.
<a href="#"><u>BTN_TEXTBOTTOM</u></a>	Bit to indicate text is top aligned.
<a href="#"><u>BTN_TEXTLEFT</u></a>	Bit to indicate text is left aligned.
<a href="#"><u>BTN_TEXTRIGHT</u></a>	Bit to indicate text is right aligned.
<a href="#"><u>BTN_TEXTTOP</u></a>	Bit to indicate text is bottom aligned.
<a href="#"><u>BTN_TOGGLE</u></a>	Bit to indicate button will have a toggle behavior.
<a href="#"><u>BTN_TWOTONE</u></a>	Bit to indicate the button is a two tone type.
<a href="#"><u>BTN_NOPANEL</u></a>	Bit to indicate the button will be drawn without a panel (for faster drawing when the button image used is larger than the button panel).

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Button](#) > **[Button States](#)**

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Chart Functions

## Chart

### Functions

	Name	Description
	<a href="#"><u>ChCreate</u></a>	This function creates a <a href="#"><u>CHART</u></a> 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.
	<a href="#"><u>ChDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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 colors of the bars of the bar chart or sectors of the pie chart can be the default color table or user defined color table set by <a href="#"><u>ChSetColorTable()</u></a> function.</p> <p>When rendering objects of the same type, each object must be rendered completely before the rendering of the next object is... <a href="#"><u>more</u></a></p>
	<a href="#"><u>ChAddDataSeries</u></a>	This function creates a <a href="#"><u>DATASERIES</u></a> object and populates the structure with the given parameters.
	<a href="#"><u>ChRemoveDataSeries</u></a>	This function removes <a href="#"><u>DATASERIES</u></a>

		object from the list of <b>DATASERIES</b> objects and frees the memory used of that removed object. The position of the object to be removed is specified by the number parameter. If the list has only one member, it removes the member regardless of the number given.
◆	<b><u>ChSetValueRange</u></b>	This function sets the minimum and maximum range of values that the bar chart will show. The criteria is that min <= max.
◆	<b><u>ChSetPercentRange</u></b>	This function sets the minimum and maximum range of percentage that the bar chart will show. The criteria is that min <= max. This affects bar charts only and CH_PERCENTAGE bit state is set.
◆	<b><u>ChSetSampleRange</u></b>	This function sets the sample start and sample end when drawing the chart. Together with the data series' <b>SHOW_DATA</b> flags the different way of displaying the chart data is achieved.
◆	<b><u>ChFreeDataSeries</u></b>	This function removes <b>DATASERIES</b> object from the list of <b>DATASERIES</b> objects and frees the memory used of that removed object.
◆	<b><u>ChTranslateMsg</u></b>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages

for each event of the touch screen and keyboard inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Chart Macros

### Chart

#### Macros

Name	Description
<a href="#"><u>ChShowSeries</u></a>	This macro sets the specified data series number show flag to be set to <a href="#"><u>SHOW_DATA</u></a> .
<a href="#"><u>ChHideSeries</u></a>	This macro sets the specified data series number show flag to be set to <a href="#"><u>HIDE_DATA</u></a> .
<a href="#"><u>ChGetShowSeriesCount</u></a>	This macro shows the number of data series that has its show flag set to <a href="#"><u>SHOW_DATA</u></a> .
<a href="#"><u>ChGetShowSeriesStatus</u></a>	This macro returns the show ID status of the <a href="#"><u>DATASERIES</u></a> .
<a href="#"><u>ChSetValueLabel</u></a>	This macro sets the address of the current text string used for the value axis label of the bar chart.
<a href="#"><u>ChGetValueLabel</u></a>	This macro returns the address of the current text string used for the value axis label of the bar chart.
<a href="#"><u>ChGetValueMax</u></a>	This macro returns the current maximum value that will be drawn for bar charts.
<a href="#"><u>ChGetValueMin</u></a>	This macro returns the current minimum value that will be drawn for bar charts.

<a href="#"><u>ChGetValueRange</u></a>	This macro gets the current range for bar charts. The value returned is calculated from the current (valMax - valMin) set. To get the minimum use <a href="#"><u>ChGetValueMin()</u></a> and to get the maximum use <a href="#"><u>ChGetValueMax()</u></a> .
<a href="#"><u>ChSetSampleLabel</u></a>	This macro sets the address of the current text string used for the sample axis label of the bar chart.
<a href="#"><u>ChGetSampleLabel</u></a>	This macro returns the address of the current text string used for the sample axis label of the bar chart.
<a href="#"><u>ChGetSampleStart</u></a>	This macro returns the sampling start value.
<a href="#"><u>ChGetSampleEnd</u></a>	This macro returns the sampling end value.
<a href="#"><u>ChGetPercentRange</u></a>	This macro gets the percentage range for bar charts. The value returned is calculated from percentage max - min. To get the minimum use <a href="#"><u>ChGetPercentMin()</u></a> and to get the maximum use <a href="#"><u>ChGetPercentMax()</u></a> .
<a href="#"><u>ChGetSampleRange</u></a>	This macro gets the sample range for pie or bar charts. The value returned is calculated from smplEnd - smplStart.
<a href="#"><u>ChGetPercentMax</u></a>	This macro returns the current maximum value of the percentage range that will be drawn for bar charts when CH_PERCENTAGE bit state is set.
<a href="#"><u>ChGetPercentMin</u></a>	This macro returns the current minimum

	value of the percentage range that will be drawn for bar charts when CH_PERCENTAGE bit state is set.
<a href="#"><u>ChSetColorTable</u></a>	This macro sets the color table used to draw the data in pie and bar charts.
<a href="#"><u>ChGetColorTable</u></a>	This macro returns the current color table used for the pie and bar charts.
<a href="#"><u>ChSetTitle</u></a>	This macro sets the address of the current text string used for the title of the chart.
<a href="#"><u>ChGetTitle</u></a>	This macro returns the address of the current text string used for the title of the chart.
<a href="#"><u>ChSetTitleFont</u></a>	This macro sets the location of the font used for the title of the chart.
<a href="#"><u>ChGetTitleFont</u></a>	This macro returns the location of the font used for the title of the chart.
<a href="#"><u>ChGetAxisLabelFont</u></a>	This macro returns the location of the font used for the X and Y axis labels of the chart.
<a href="#"><u>ChSetAxisLabelFont</u></a>	This macro sets the location of the font used for the X and Y axis labels of the chart.
<a href="#"><u>ChGetGridLabelFont</u></a>	This macro returns the location of the font used for the X and Y axis grid labels of the chart.
<a href="#"><u>ChSetGridLabelFont</u></a>	This macro sets the location of the font used for the X and Y axis grid labels of

the chart.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Chart Structures

### Chart

### Structures

Name	Description
<a href="#"><u>CHART</u></a>	Defines the parameters required for a chart Object.
<a href="#"><u>DATASERIES</u></a>	Defines a variable for the <a href="#"><u>CHART</u></a> object. It specifies the number of samples, pointer to the array of samples for the data series and pointer to the next data series. A member of this structure (show) is used as a flag to determine if the series is to be drawn or not. Together with the <code>smplStart</code> and <code>smplEnd</code> it will determine what kind of chart will be drawn.
<a href="#"><u>CHARTPARAM</u></a>	Defines the parameters for the <a href="#"><u>CHART</u></a> object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#)

## Topics

Chart

## Topics

Name	Description
<a href="#">Chart States</a>	List of Chart bit states.
<a href="#">Data Series Status Settings</a>	Data Series show status flag settings.
<a href="#">Chart Examples</a>	Examples of generated bar charts based on settings.
<a href="#">Color Table</a>	Default color table used to draw data points in a chart.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

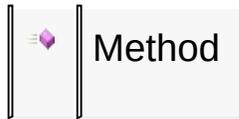
[Contents](#) | [Index](#) | [Home](#)

## Legend

Chart

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Chart States Macros

### Chart States

#### Macros

Name	Description
<a href="#"><u>CH_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>CH_DRAW</u></a>	Bit to indicate chart must be redrawn.
<a href="#"><u>CH_DRAW_DATA</u></a>	Bit to indicate data portion of the chart must be redrawn.
<a href="#"><u>CH_3D_ENABLE</u></a>	Bit to indicate that bar charts are to be drawn with 3-D effect
<a href="#"><u>CH_BAR</u></a>	Bit to indicate the chart is type bar. If both PIE and BAR types are set BAR type has higher priority.
<a href="#"><u>CH_BAR_HOR</u></a>	These bits (with <a href="#"><u>CH_BAR</u></a> bit set), sets the bar chart to be drawn horizontally.
<a href="#"><u>CH_DONUT</u></a>	These bits (with <a href="#"><u>CH_PIE</u></a> bit set), sets the pie chart to be drawn in a donut shape.
<a href="#"><u>CH_LEGEND</u></a>	Bit to indicate that legend is to be shown. Usable only when seriesCount > 1.
<a href="#"><u>CH_NUMERIC</u></a>	This bit is used only for bar charts. If this bit is set, it indicates that the bar chart labels for variables are numeric. If this bit is not set, it indicates that the bar chart labels for variables are alphabets.
<a href="#"><u>CH_PERCENT</u></a>	Bit to indicate that the pie chart will be

	drawn with percentage values shown for the sample data. For bar chart, if <a href="#">CH_VALUE</a> is set, it toggles the value shown to percentage.
<a href="#">CH_PIE</a>	Bit to indicate the chart is type pie. If both PIE and BAR types are set BAR type has higher priority.
<a href="#">CH_VALUE</a>	Bit to indicate that the values of the bar chart data or pie chart data are to be shown
<a href="#">CH_HIDE</a>	Bit to indicate chart must be removed from screen.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Chart States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Data Series Status Settings Macros

### Data Series Status Settings

#### Macros

Name	Description
<a href="#"><u>HIDE_DATA</u></a>	Macro used to reset the data series show flag or indicate that the data series will be not be shown when the chart is drawn.
<a href="#"><u>SHOW_DATA</u></a>	Macro used to set the data series show flag or indicate that the data series will be shown when the chart is drawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Data Series Status Settings](#)

## Color Table Macros

Color Table

### Macros

Name	Description
<a href="#"><u>CH_CLR0</u></a>	Bright Blue
<a href="#"><u>CH_CLR1</u></a>	Bright Red
<a href="#"><u>CH_CLR2</u></a>	Bright Green
<a href="#"><u>CH_CLR3</u></a>	Bright Yellow
<a href="#"><u>CH_CLR4</u></a>	Orange
<a href="#"><u>CH_CLR5</u></a>	Blue
<a href="#"><u>CH_CLR6</u></a>	Red
<a href="#"><u>CH_CLR7</u></a>	Green
<a href="#"><u>CH_CLR8</u></a>	Yellow
<a href="#"><u>CH_CLR9</u></a>	Dark Orange
<a href="#"><u>CH_CLR10</u></a>	Light Blue
<a href="#"><u>CH_CLR11</u></a>	Light Red
<a href="#"><u>CH_CLR12</u></a>	Light Green
<a href="#"><u>CH_CLR13</u></a>	Light Yellow
<a href="#"><u>CH_CLR14</u></a>	Light Orange

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Chart](#) > [Color Table](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Checkbox Functions

## Checkbox

### Functions

	Name	Description
	<a href="#"><u>CbCreate</u></a>	This function creates a <b>CHECKBOX</b> 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.
	<a href="#"><u>CbDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<a href="#"><u>CbSetText</u></a>	This function sets the text that will be used.
	<a href="#"><u>CbMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
	<a href="#"><u>CbTranslateMsg</u></a>	This function evaluates the message from

a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Checkbox Macros

## Checkbox

---

### Macros

---

Name	Description
<a href="#">CbGetText</a>	This macro returns the location of the text used for the check box.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Checkbox Structures

## Checkbox

---

### Structures

---

Name	Description
<a href="#">CHECKBOX</a>	The structure contains check box data

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Checkbox

---

## Topics

Name	Description
<a href="#">Check Box States</a>	List of Checkbox bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

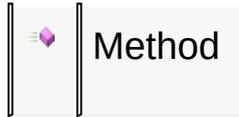
[Contents](#) | [Index](#) | [Home](#)

## Legend

Checkbox

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Check Box States Macros

### Check Box States

#### Macros

Name	Description
<a href="#"><u>CB_CHECKED</u></a>	Checked state
<a href="#"><u>CB_DISABLED</u></a>	Disabled state
<a href="#"><u>CB_DRAW</u></a>	Whole check box must be redrawn
<a href="#"><u>CB_DRAW_CHECK</u></a>	Check box mark should be redrawn
<a href="#"><u>CB_DRAW_FOCUS</u></a>	Focus must be redrawn
<a href="#"><u>CB_FOCUSED</u></a>	Focus state
<a href="#"><u>CB_HIDE</u></a>	Check box must be removed from screen

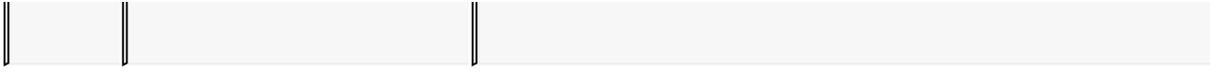
[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Checkbox](#) > [Check Box States](#)

## Round Dial Functions

### Round Dial

#### Functions

	Name	Description
	<a href="#"><u>RdiaCreate</u></a>	This function creates a <a href="#"><u>ROUND DIAL</u></a> 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.
	<a href="#"><u>RdiaDraw</u></a>	This function renders the object on the screen using the current parameter settings. Location of the object is determined by the center (x,y) position and the radius parameters. The colors used are dependent on the state of 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.
	<a href="#"><u>RdiaMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
	<a href="#"><u>RdiaTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Round Dial Macros

### Round Dial

#### Macros

Name	Description
<a href="#">RdiaIncVal</a>	Used to directly increment the value. The delta change used is the resolution setting (res).
<a href="#">RdiaDecVal</a>	Used to directly decrement the value. The delta change used is the resolution setting (res).
<a href="#">RdiaGetVal</a>	Returns the current dial value. Value is always in the 0-max range inclusive.
<a href="#">RdiaSetVal</a>	Sets the value to the given new value. Value set must be in 0-max range inclusive.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#)

# Round Dial Structures

## Round Dial

### Structures

Name	Description
<a href="#">ROUND DIAL</a>	Defines the parameters required for a dial Object. The curr_xPos, curr_yPos, new_xPos and new_yPos parameters are internally generated to aid in the redrawing of the dial. User must avoid modifying these values.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#)

## Topics

Round Dial

---

## Topics

Name	Description
<a href="#">Dial States</a>	List of Dial bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

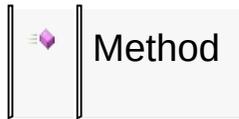
[Contents](#) | [Index](#) | [Home](#)

## Legend

Round Dial

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Dial States Macros

### Dial States

#### Macros

Name	Description
<a href="#"><u>RDIA_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>RDIA_DRAW</u></a>	Bit to indicate object must be redrawn.
<a href="#"><u>RDIA_HIDE</u></a>	Bit to indicate object must be removed from screen.
<a href="#"><u>RDIA_ROT_CCW</u></a>	Bit for rotate counter clockwise state.
<a href="#"><u>RDIA_ROT_CW</u></a>	Bit for rotate clockwise state.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Round Dial](#) > [Dial States](#)

# Digital Meter Functions

## Digital Meter

### Functions

	Name	Description
	<a href="#"><u>DmCreate</u></a>	This function creates a <a href="#"><u>DIGITALMETER</u></a> 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.
	<a href="#"><u>DmDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<a href="#"><u>DmSetValue</u></a>	This function sets the value that will be used for the object.
	<a href="#"><u>DmTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Digital Meter Macros

Digital Meter

### Macros

Name	Description
<a href="#">DmGetValue</a>	This macro returns the current value used for the object.
<a href="#">DmDecVal</a>	This macro is used to directly decrement the value.
<a href="#">DmIncVal</a>	This macro is used to directly increment the value.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#)

# Digital Meter Structures

## Digital Meter

### Structures

Name	Description
<a href="#"><u>DIGITALMETER</u></a>	Defines the parameters required for a Digital Meter Object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Digital Meter

---

## Topics

Name	Description
<a href="#">Digital Meter States</a>	List of Digital <a href="#">Meter</a> bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Digital Meter

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Digital Meter States Macros

### Digital Meter States

#### Macros

Name	Description
<a href="#"><u>DM_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>DM_DRAW</u></a>	Bit to indicate object must be redrawn.
<a href="#"><u>DM_HIDE</u></a>	Bit to remove object from screen.
<a href="#"><u>DM_CENTER_ALIGN</u></a>	Bit to indicate value is center aligned.
<a href="#"><u>DM_RIGHT_ALIGN</u></a>	Bit to indicate value is left aligned.
<a href="#"><u>DM_FRAME</u></a>	Bit to indicate frame is displayed.
<a href="#"><u>DM_UPDATE</u></a>	Bit to indicate that only text must be redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Digital Meter](#) > [Digital Meter States](#)

## Edit Box Functions

### Edit Box

### Functions

	Name	Description
	<a href="#"><u>EbCreate</u></a>	This function creates a <a href="#"><u>EDITBOX</u></a> object with the parameters given and initializes the default settings. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	<a href="#"><u>EbDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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.
	<a href="#"><u>EbSetText</u></a>	This function sets the text to be used for the object.
	<a href="#"><u>EbAddChar</u></a>	This function inserts a character at the end of the text used by the object.
	<a href="#"><u>EbDeleteChar</u></a>	This function removes a character at the end of the text used by the object.

	<a href="#"><u>EbMsgDefault</u></a>	<p>This function performs the actual state change based on the translated message given. The following state changes are supported:</p>
	<a href="#"><u>EbTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Edit Box Macros

## Edit Box

---

### Macros

---

Name	Description
<a href="#">EbGetText</a>	This macro returns the address of the current text string used for the object.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Edit Box Structures

Edit Box

## Structures

Name	Description
<a href="#">EDITBOX</a>	Defines the parameters required for a Edit Box Object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

[Edit Box](#)

---

## Topics

Name	Description
<a href="#">Edit Box States</a>	List of Edit Box bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

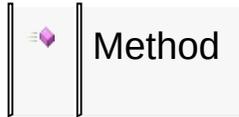
[Contents](#) | [Index](#) | [Home](#)

## Legend

Edit Box

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Edit Box States Macros

### Edit Box States

#### Macros

Name	Description
<a href="#">EB_CENTER_ALIGN</a>	Bit to indicate text is center aligned.
<a href="#">EB_DISABLED</a>	Bit for disabled state.
<a href="#">EB_DRAW</a>	Bit to indicate whole edit box must be redrawn.
<a href="#">EB_HIDE</a>	Bit to remove object from screen.
<a href="#">EB_FOCUSED</a>	Bit for focused state. Cursor caret will be drawn when <a href="#">EB_DRAW_CARET</a> is also set.
<a href="#">EB_RIGHT_ALIGN</a>	Bit to indicate text is left aligned.
<a href="#">EB_DRAW_CARET</a>	Bit to indicate the cursor caret will be drawn if <a href="#">EB_FOCUSED</a> state bit is set and erase when <a href="#">EB_FOCUSED</a> state bit is not set.
<a href="#">EB_CARET</a>	Bit to indicate the cursor caret will always be shown.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Edit Box](#) > [Edit Box States](#)

# Grid Functions

## Grid

### Functions

	Name	Description
	<a href="#"><u>GridCreate</u></a>	This function creates a <a href="#"><u>GRID</u></a> 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.
	<a href="#"><u>GridDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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.
	<a href="#"><u>GridClearCellState</u></a>	This function clears the state of the cell (or Grid Item) specified by the column and row.
	<a href="#"><u>GridFreeItems</u></a>	This function removes all grid items for the given Grid and frees the memory used.
	<a href="#"><u>GridGetCell</u></a>	This function removes all grid items for

		the given Grid and frees the memory used.
⇒	<a href="#"><u>GridSetCell</u></a>	This function sets the Grid Item state and data.
⇒	<a href="#"><u>GridSetCellState</u></a>	This function sets the state of the Grid Item or cell.
⇒	<a href="#"><u>GridSetFocus</u></a>	This function sets the focus of the specified Grid Item or cell.
⇒	<a href="#"><u>GridMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
⇒	<a href="#"><u>GridTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Grid Macros

### Grid

## Macros

Name	Description
<a href="#">GridGetFocusX</a>	This macro returns the x position of the focused cell.
<a href="#">GridGetFocusY</a>	This macro returns the y position of the focused cell.
<a href="#">GRID_OUT_OF_BOUNDS</a>	Status of an out of bounds cell <a href="#">GridSetCell()</a> operation.
<a href="#">GRID_SUCCESS</a>	Status of a successful <a href="#">GridSetCell()</a> operation.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#)

# Grid Structures

## Grid

### Structures

Name	Description
<a href="#"><u>GRID</u></a>	Defines the parameters required for a grid Object. Clipping is not supported in grid object.
<a href="#"><u>GRIDITEM</u></a>	Defines the grid item.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Grid

---

## Topics

---

Name	Description
<a href="#">Grid States</a>	List of Grid bit states.
<a href="#">Grid Item States</a>	List of Grid Items bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

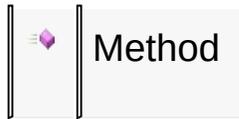
[Contents](#) | [Index](#) | [Home](#)

## Legend

Grid

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Grid States Macros

### Grid States

#### Macros

Name	Description
<a href="#"><u>GRID_FOCUSED</u></a>	Bit for focused state
<a href="#"><u>GRID_DISABLED</u></a>	Bit for disabled state
<a href="#"><u>GRID_SHOW_LINES</u></a>	Display grid lines
<a href="#"><u>GRID_SHOW_FOCUS</u></a>	Highlight the focused cell.
<a href="#"><u>GRID_SHOW_BORDER_ONLY</u></a>	Draw only the outside border of the grid.
<a href="#"><u>GRID_SHOW_SEPARATORS_ONLY</u></a>	Draw only the lines between cells (like Tic-tac-toe)
<a href="#"><u>GRID_DRAW_ITEMS</u></a>	Bit to indicate that at least one item must be redrawn, but not the entire grid.
<a href="#"><u>GRID_DRAW_ALL</u></a>	Bit to indicate whole edit box must be redrawn
<a href="#"><u>GRID_HIDE</u></a>	Bit to remove object from screen

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid States](#)

## Grid Item States Macros

### Grid Item States

#### Macros

Name	Description
<a href="#"><u>GRIDITEM_SELECTED</u></a>	The cell is selected.
<a href="#"><u>GRIDITEM_IS_TEXT</u></a>	The grid item is a text string.
<a href="#"><u>GRIDITEM_IS_BITMAP</u></a>	The grid item is a bitmap.
<a href="#"><u>GRIDITEM_TEXTBOTTOM</u></a>	Bit to indicate text is top aligned.
<a href="#"><u>GRIDITEM_TEXTLEFT</u></a>	Text in the cell is left aligned.
<a href="#"><u>GRIDITEM_TEXTRIGHT</u></a>	Text in the cell is right aligned.
<a href="#"><u>GRIDITEM_TEXTTOP</u></a>	Bit to indicate text is bottom aligned.
<a href="#"><u>GRIDITEM_DRAW</u></a>	Draw this cell

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Grid](#) > [Grid Item States](#)

## Group Box Functions

### Group Box

### Functions

	Name	Description
	<a href="#"><u>GbCreate</u></a>	This function creates a <a href="#"><u>GROUPBOX</u></a> 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.
	<a href="#"><u>GbDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<a href="#"><u>GbSetText</u></a>	This function sets the text used by passing the pointer to the static string.
	<a href="#"><u>GbTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Group Box Macros

### Group Box

---

#### Macros

---

Name	Description
<a href="#">GbGetText</a>	This macro returns the location of the text used.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Group Box Structures

## Group Box

### Structures

Name	Description
<a href="#"><u>GROUPBOX</u></a>	Defines the parameters required for a group box Object. The textwidth and textHeight is not checked with the actual dimension of the object. Clipping is not supported in group box object. It is possible for the text to exceed the dimension of the Object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#)

## Topics

Group Box

---

## Topics

Name	Description
<a href="#">Group Box States</a>	List of Group Box bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

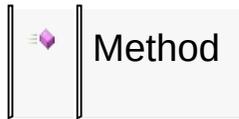
[Contents](#) | [Index](#) | [Home](#)

## Legend

Group Box

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Group Box States Macros

### Group Box States

---

#### Macros

---

Name	Description
<a href="#"><u>GB_CENTER_ALIGN</u></a>	Bit to indicate text is center aligned
<a href="#"><u>GB_DISABLED</u></a>	Bit for disabled state
<a href="#"><u>GB_DRAW</u></a>	Bit to indicate group box must be redrawn
<a href="#"><u>GB_HIDE</u></a>	Bit to remove object from screen
<a href="#"><u>GB_RIGHT_ALIGN</u></a>	Bit to indicate text is right aligned

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Group Box](#) > [Group Box States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

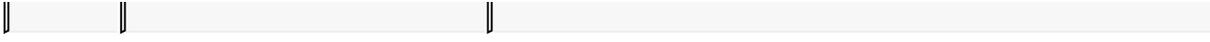
# List Box Functions

## List Box

### Functions

	Name	Description
	<a href="#"><u>LbCreate</u></a>	This function creates a <a href="#"><u>LISTBOX</u></a> 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.
	<a href="#"><u>LbDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>The text or items drawn in the visible window of the list box is dependent on the alignment set.</p> <p>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.</p>
	<a href="#"><u>LbAddItem</u></a>	This function allocates memory for the <a href="#"><u>LISTITEM</u></a> and adds it to the list box. The newly created <a href="#"><u>LISTITEM</u></a> will store the location of pText, pBitmap and other parameters describing the added item.

◆	<a href="#"><u>LbDelItem</u></a>	This function removes an item from the list box and frees the memory used.
◆	<a href="#"><u>LbChangeSel</u></a>	This function changes the selection status of an item in the list box. If the item is currently selected, it resets the selection. If the item is currently not selected it is set to be selected.
◆	<a href="#"><u>LbGetSel</u></a>	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 list. It returns the pointer to the first selected item found or NULL if there are no items selected.
◆	<a href="#"><u>LbGetFocusedItem</u></a>	This function returns the index of the focused item in the list box.
◆	<a href="#"><u>LbSetFocusedItem</u></a>	This function sets the focus for the item with the given index.
◆	<a href="#"><u>LbDelItemsList</u></a>	This function removes all items from the list box and frees the memory used.
◆	<a href="#"><u>LbMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
◆	<a href="#"><u>LbTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## List Box Macros

### List Box

#### Macros

Name	Description
<a href="#"><u>LbGetItemList</u></a>	This function returns the pointer to the current item list used in the list box.
<a href="#"><u>LbSetSel</u></a>	This macro sets the selection status of an item to selected.
<a href="#"><u>LbGetCount</u></a>	This macro returns the number of items in the list box.
<a href="#"><u>LbGetVisibleCount</u></a>	This macro returns the number of items visible in the list box window.
<a href="#"><u>LbSetBitmap</u></a>	This macro sets the bitmap used in the item.
<a href="#"><u>LbGetBitmap</u></a>	This macro returns the location of the currently used bitmap for the item.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#)

# List Box Structures

## List Box

### Structures

Name	Description
<a href="#"><u>LISTBOX</u></a>	Defines the parameters required for a list box Object.
<a href="#"><u>LISTITEM</u></a>	Defines the parameters required for a list item used in list box.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

[List Box](#)

---

## Topics

Name	Description
<a href="#">List Box States</a>	List of List Box bit states.
<a href="#">List Item Status</a>	List of Items status.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

List Box

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## List Box States Macros

### List Box States

#### Macros

Name	Description
<a href="#"><u>LB_RIGHT_ALIGN</u></a>	Bit to indicate text is left aligned
<a href="#"><u>LB_SINGLE_SEL</u></a>	Bit to indicate the only item can be selected
<a href="#"><u>LB_CENTER_ALIGN</u></a>	Bit to indicate text is center aligned
<a href="#"><u>LB_DISABLED</u></a>	Bit for disabled state
<a href="#"><u>LB_DRAW</u></a>	Bit to indicate whole edit box must be redrawn
<a href="#"><u>LB_DRAW_FOCUS</u></a>	Bit to indicate whole edit box must be redrawn
<a href="#"><u>LB_DRAW_ITEMS</u></a>	Bit to indicate whole edit box must be redrawn
<a href="#"><u>LB_FOCUSED</u></a>	Bit for focused state
<a href="#"><u>LB_HIDE</u></a>	Bit to remove object from screen

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Box States](#)

# List Item Status Macros

## List Item Status

---

### Macros

---

Name	Description
<a href="#"><u>LB_STS_SELECTED</u></a>	Item is selected.
<a href="#"><u>LB_STS_REDRAW</u></a>	Item is to be redrawn.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [List Box](#) > [List Item Status](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Meter Functions

Meter

### Functions

	Name	Description
	<a href="#"><u>MtrCreate</u></a>	This function creates a <a href="#"><u>METER</u></a> 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.
	<a href="#"><u>MtrDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>Depending on the defined settings, value of the meter will displayed or hidden. Displaying the value will require a little bit more rendering time depending on the size of the meter and font used.</p> <p>When rendering objects of the same type, each object must be rendered completely before the rendering of the... <a href="#"><u>more</u></a></p>
	<a href="#"><u>MtrSetVal</u></a>	This function sets the value of the meter to the passed newVal. newVal is checked to be in the minValue-maxValue range inclusive. If newVal is not in the range, minValue maxValue is assigned depending on the given newVal if less than minValue

		or above maxValue.
	<a href="#"><u>MtrMsgDefault</u></a>	This function performs the actual state change based on the translated message given. Meter value is set based on parameter 2 of the message given. The following state changes are supported:
	<a href="#"><u>MtrTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Meter Macros

### Meter

#### Macros

Name	Description
<a href="#">MtrGetVal</a>	This macro returns the current value of the meter. Value is always in the minValue-maxValue range inclusive.
<a href="#">MtrDecVal</a>	This macro is used to directly decrement the value.
<a href="#">MtrIncVal</a>	This macro is used to directly increment the value.
<a href="#">MtrSetScaleColors</a>	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. Limitation is that color settings are set to the following angles: Color Boundaries Type Whole Type Half Type Quarter <a href="#">Arc</a> 6 225 to 180 not used not used <a href="#">Arc</a> 5 179 to 135 179 to 135 not used <a href="#">Arc</a> 4 134 to 90 134 to 90 not used <a href="#">Arc</a> 3 89 to 45 89 to 45 89 to 45 <a href="#">Arc</a> 2 44 to 0 44... <a href="#">more</a>
<a href="#">MtrSetTitleFont</a>	This function sets the font of title.
<a href="#">MtrSetValueFont</a>	This function sets the font of value.
<a href="#">METER_TYPE</a>	This is a compile time setting to select the type if meter shape. There are three types: <ul style="list-style-type: none"><li>• MTR_WHOLE_TYPE - Meter drawn</li></ul>

	<p>with 6 octants used.</p> <ul style="list-style-type: none"><li>• MTR_HALF_TYPE - Meter drawn with semi circle shape.</li><li>• MTR_QUARTER_TYPE - Meter drawn with quarter circle shape.</li></ul> <p>Set only one value at a time. This is done to save code space. User can define the colors of the arcs for each type.</p> <p>MTR_WHOLE_TYPE will use all the arc colors (arcColor1 - arcColor6)</p> <p>MTR_HALF_TYPE will use arc colors (arcColor5, arcColor4, arcColor3, arcColor2)</p> <p>MTR_QUARTER_TYPE will use arc colors (arcColor3, arcColor2) Set the meter type in Meter.h file and... <a href="#">more</a></p>
<p><a href="#">MTR_ACCURACY</a></p>	<p>Sets the meter accuracy to one decimal places when displaying the values.</p> <p>Application must multiply the minValue, maxValue and values passed to the widget by RESOLUTION.</p>

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Meter Structures

Meter

## Structures

Name	Description
<a href="#">METER</a>	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.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Meter

---

## Topics

Name	Description
<a href="#">Meter States</a>	List of Meter bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

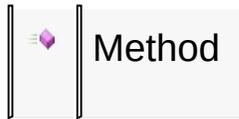
[Contents](#) | [Index](#) | [Home](#)

## Legend

Meter

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Meter States Macros

Meter States

### Macros

Name	Description
<a href="#"><u>MTR_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>MTR_DRAW</u></a>	Bit to indicate object must be redrawn.
<a href="#"><u>MTR_HIDE</u></a>	Bit to indicate object must be removed from screen.
<a href="#"><u>MTR_RING</u></a>	Bit for ring type, scales are drawn over the ring default is only scales drawn.
<a href="#"><u>MTR_DRAW_UPDATE</u></a>	Bit to indicate an update only.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Meter](#) > [Meter States](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Picture Control Functions

### Picture Control

### Functions

	Name	Description
	<a href="#"><u>PictCreate</u></a>	This function creates a <a href="#"><u>PICTURE</u></a> 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.
	<a href="#"><u>PictDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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. 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.
	<a href="#"><u>PictTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event accepted by the <a href="#"><u>PICTURE</u></a> Object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Picture Control Macros

Picture Control

### Macros

Name	Description
<a href="#">PictSetBitmap</a>	This macro sets the bitmap used in the object.
<a href="#">PictGetBitmap</a>	This macro returns the pointer to the bitmap used in the object.
<a href="#">PictGetScale</a>	This macro returns the current scale factor used to render the bitmap.
<a href="#">PictSetScale</a>	This macro sets the scale factor used to render the bitmap used in the object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Picture Control Structures

Picture Control

---

## Structures

---

Name	Description
<a href="#">PICTURE</a>	The structure contains data for picture control

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Picture Control

---

## Topics

Name	Description
<a href="#">Picture States</a>	List of Picture Control bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

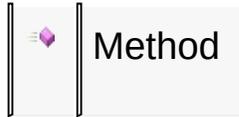
[Contents](#) | [Index](#) | [Home](#)

## Legend

Picture Control

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Picture States Macros

## Picture States

### Macros

Name	Description
<a href="#">PICT_DISABLED</a>	Bit to indicate <a href="#">Picture</a> is in a disabled state.
<a href="#">PICT_DRAW</a>	Bit to indicate <a href="#">Picture</a> will be redrawn.
<a href="#">PICT_FRAME</a>	Bit to indicate <a href="#">Picture</a> has a frame.
<a href="#">PICT_HIDE</a>	Bit to indicate <a href="#">Picture</a> must be hidden.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Picture Control](#) > [Picture States](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Progress Bar Functions

## Progress Bar

### Functions

	Name	Description
	<a href="#"><u>PbCreate</u></a>	This function creates a <a href="#"><u>PROGRESSBAR</u></a> 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.
	<a href="#"><u>PbDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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.
	<a href="#"><u>PbSetRange</u></a>	This function sets the range of the progress bar. Calling this function also resets the position equal to the new range value.
	<a href="#"><u>PbSetPos</u></a>	This function sets the position of the progress bar. Position should be in the given range inclusive.
	<a href="#"><u>PbTranslateMsg</u></a>	This function evaluates the message from a

user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Progress Bar Macros

## Progress Bar

### Macros

Name	Description
<a href="#">PbGetRange</a>	This macro returns the current range of the progress bar.
<a href="#">PbGetPos</a>	This macro returns the current progress bar position.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Progress Bar Structures

## Progress Bar

---

### Structures

---

Name	Description
<a href="#"><u>PROGRESSBAR</u></a>	The structure contains data for the progress bar

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Progress Bar

---

## Topics

Name	Description
<a href="#">Progress Bar States</a>	List of Progress Bar bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

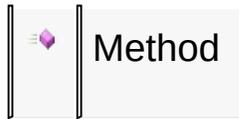
[Contents](#) | [Index](#) | [Home](#)

## Legend

Progress Bar

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Progress Bar States Macros

### Progress Bar States

#### Macros

Name	Description
<a href="#"><u>PB_DISABLED</u></a>	Bit to indicate Progress <a href="#"><u>Bar</u></a> is in a disabled state.
<a href="#"><u>PB_DRAW</u></a>	Bit to indicate Progress <a href="#"><u>Bar</u></a> must be redrawn.
<a href="#"><u>PB_DRAW_BAR</u></a>	Bit to indicate Progress <a href="#"><u>Bar</u></a> must be redrawn.
<a href="#"><u>PB_HIDE</u></a>	Bit to indicate Progress <a href="#"><u>Bar</u></a> must be hidden.
<a href="#"><u>PB_VERTICAL</u></a>	Bit for orientation (0 - horizontal, 1 - vertical)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Progress Bar](#) > [Progress Bar States](#)

# Radio Button Functions

## Radio Button

### Functions

	Name	Description
	<a href="#"><u>RbCreate</u></a>	This function creates a <a href="#"><u>RADIOBUTTON</u></a> 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.
	<a href="#"><u>RbDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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.
	<a href="#"><u>RbGetCheck</u></a>	This function returns the ID of the currently checked Radio <a href="#"><u>Button</u></a> in the group.
	<a href="#"><u>RbSetCheck</u></a>	This function sets the Radio <a href="#"><u>Button</u></a> with the given ID to its checked state.
	<a href="#"><u>RbSetText</u></a>	This function sets the string used for the object.

	<a href="#"><u>RbMsgDefault</u></a>	<p>This function performs the actual state change based on the translated message given. The following state changes are supported:</p>
	<a href="#"><u>RbTranslateMsg</u></a>	<p>This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.</p>

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Radio Button Macros

## Radio Button

### Macros

Name	Description
<a href="#">RbGetText</a>	This macro returns the address of the current text string used for the object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Radio Button Structures

## Radio Button

### Structures

Name	Description
<a href="#">RADIOBUTTON</a>	the structure contains data for the <a href="#">Radio Button</a>

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Radio Button

---

## Topics

Name	Description
<a href="#">Radio Button States</a>	List of Radio Button bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

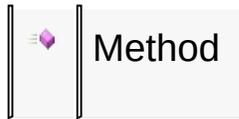
[Contents](#) | [Index](#) | [Home](#)

## Legend

Radio Button

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Radio Button States Macros

### Radio Button States

#### Macros

Name	Description
<a href="#"><u>RB_CHECKED</u></a>	Bit to indicate Radio <a href="#"><u>Button</u></a> is checked.
<a href="#"><u>RB_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>RB_DRAW</u></a>	Bit to indicate whole Radio <a href="#"><u>Button</u></a> must be redrawn.
<a href="#"><u>RB_DRAW_CHECK</u></a>	Bit to indicate check mark should be redrawn.
<a href="#"><u>RB_DRAW_FOCUS</u></a>	Bit to indicate focus must be redrawn.
<a href="#"><u>RB_FOCUSED</u></a>	Bit for focused state.
<a href="#"><u>RB_GROUP</u></a>	Bit to indicate the first Radio <a href="#"><u>Button</u></a> in the group.
<a href="#"><u>RB_HIDE</u></a>	Bit to indicate that button must be removed from screen.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Radio Button](#)  
> [Radio Button States](#)

## Slider/Scroll Bar Functions

### Slider/Scroll Bar

#### Functions

	Name	Description
	<a href="#"><u>SldCreate</u></a>	This function creates a <a href="#"><u>SLIDER</u></a> object with the parameters given. Depending on the <a href="#"><u>SLD_SCROLLBAR</u></a> state bit slider or scrollbar mode is set. If <a href="#"><u>SLD_SCROLLBAR</u></a> is set, mode is scrollbar; if not set mode is slider. It automatically attaches the new object into a global linked list of objects and returns the address of the object.
	<a href="#"><u>SldDraw</u></a>	This function renders the object on the screen using the current parameter settings. 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. 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.
	<a href="#"><u>SldSetPage</u></a>	This sets the page size of the object. Page size defines the delta change of the thumb position when incremented via <a href="#"><u>SldIncPos()</u></a> or decremented via <a href="#"><u>SldDecPos()</u></a> . Page size minimum value is 1. Maximum value is range/2.
	<a href="#"><u>SldSetPos</u></a>	This function sets the position of the slider

		thumb. Value should be in the set range inclusive. Object must be redrawn to reflect the change.
	<a href="#"><u>SldSetRange</u></a>	This sets the range of the thumb. If this field is changed Object must be completely redrawn to reflect the change.
	<a href="#"><u>SldMsgDefault</u></a>	This function performs the actual state change based on the translated message given. The following state changes are supported:
	<a href="#"><u>SldTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Slider/Scroll Bar Macros

### Slider/Scroll Bar

#### Macros

Name	Description
<a href="#">SldGetPage</a>	Returns the current page size of the object. Page size defines the delta change of the thumb position when incremented via <a href="#">SldIncPos()</a> or decremented via <a href="#">SldDecPos()</a> . Page size minimum value is 1. Maximum value is range/2.
<a href="#">SldGetPos</a>	Returns returns the current position of the slider thumb.
<a href="#">SldGetRange</a>	Returns the current range of the thumb.
<a href="#">SldIncPos</a>	This macro increment the slider position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.
<a href="#">SldDecPos</a>	This macro decrement the slider position by the delta change (page) value set. Object must be redrawn after this function is called to reflect the changes to the object.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#)

# Slider/Scroll Bar Structures

## Slider/Scroll Bar

### Structures

Name	Description
<a href="#"><u>SLIDER</u></a>	Defines the parameters required for a slider/scrollbar Object. Depending on the <a href="#"><u>SLD_SCROLLBAR</u></a> state bit slider or scrollbar mode is set. If <a href="#"><u>SLD_SCROLLBAR</u></a> is set, mode is scrollbar; if not set mode is slider. For scrollbar mode, focus rectangle is not drawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#)

## Topics

Slider/Scroll Bar

---

## Topics

Name	Description
<a href="#">Slider States</a>	List of Slider bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

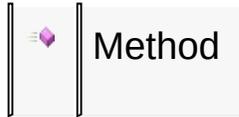
[Contents](#) | [Index](#) | [Home](#)

## Legend

Slider/Scroll Bar

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/Scroll Bar](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Slider States Macros

## Slider States

### Macros

Name	Description
<a href="#"><u>SLD_DISABLED</u></a>	Bit for disabled state
<a href="#"><u>SLD_DRAW</u></a>	Bit to indicate whole slider must be redrawn
<a href="#"><u>SLD_DRAW_FOCUS</u></a>	Bit to indicate that only the focus will be redrawn
<a href="#"><u>SLD_DRAW_THUMB</u></a>	Bit to indicate that only thumb area must be redrawn
<a href="#"><u>SLD_FOCUSED</u></a>	Bit for focus state
<a href="#"><u>SLD_HIDE</u></a>	Bit to remove object from screen
<a href="#"><u>SLD_SCROLLBAR</u></a>	Bit for type usage (0 - <a href="#">Slider</a> , 1 - <a href="#">ScrollBar</a> )
<a href="#"><u>SLD_VERTICAL</u></a>	Bit for orientation (0 - horizontal, 1 - vertical)

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Slider/ScrollBar](#) > [Slider States](#)

## Static Text Functions

### Static Text

#### Functions

	Name	Description
	<a href="#"><u>StCreate</u></a>	This function creates a <a href="#"><u>STATICTEXT</u></a> 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.
	<a href="#"><u>StDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<a href="#"><u>StSetText</u></a>	This function sets the string that will be used for the object.
	<a href="#"><u>StTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen and keyboard inputs.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Static Text Macros

## Static Text

---

### Macros

---

Name	Description
<a href="#">StGetText</a>	This macro returns the address of the current text string used for the object.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Static Text Structures

## Static Text

---

### Structures

---

Name	Description
<a href="#"><u>STATICTEXT</u></a>	Defines the parameters required for a Static Text Object.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Static Text

---

## Topics

Name	Description
<a href="#">Static Text States</a>	List of Static Text bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

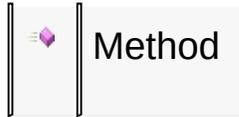
[Contents](#) | [Index](#) | [Home](#)

## Legend

Static Text

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Static Text States Macros

### Static Text States

#### Macros

Name	Description
<a href="#"><u>ST_CENTER_ALIGN</u></a>	Bit to indicate text is center aligned.
<a href="#"><u>ST_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>ST_DRAW</u></a>	Bit to indicate static text must be redrawn.
<a href="#"><u>ST_FRAME</u></a>	Bit to indicate frame is displayed.
<a href="#"><u>ST_HIDE</u></a>	Bit to remove object from screen.
<a href="#"><u>ST_RIGHT_ALIGN</u></a>	Bit to indicate text is left aligned.
<a href="#"><u>ST_UPDATE</u></a>	Bit to indicate that text area only is redrawn.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Static Text](#) > [Static Text States](#)

## Text Entry Functions

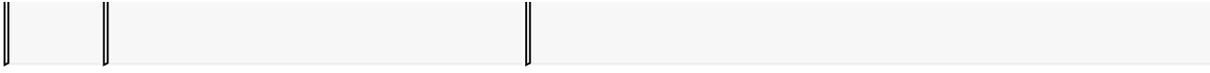
### Text Entry

### Functions

	Name	Description
	<a href="#"><u>TeCreate</u></a>	This function creates a <a href="#"><u>TEXTENTRY</u></a> 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.
	<a href="#"><u>TeDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>This widget will draw the keys using the function <a href="#"><u>GOLPanelDraw()</u></a>. The number of keys will depend on the horizontal and vertical parameters given (horizontalKeys*verticalKeys).</p>
	<a href="#"><u>TeSetBuffer</u></a>	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+1. The +1 is used for the string terminator.

	<u><a href="#">TeClearBuffer</a></u>	<p>This function will clear the data in the display. You must set the drawing state bit <u><a href="#">TE_UPDATE_TEXT</a></u> to update the <u><a href="#">TEXTENTRY</a></u> on the screen.</p>
	<u><a href="#">TeGetKeyCommand</a></u>	<p>This function will return the currently used command by a key with the given index.</p>
	<u><a href="#">TeSetKeyCommand</a></u>	<p>This function will assign a command (<u><a href="#">TE_DELETE_COM</a></u>, <u><a href="#">TE_SPACE_COM</a></u> or <u><a href="#">TE_ENTER_COM</a></u>) to a key with the given index.</p>
	<u><a href="#">TeCreateKeyMembers</a></u>	<p>This function will create the list of KEYMEMBERS that holds the information on each key. The number of keys is determined by the equation (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 equal or greater than (verticalKeys*horizontalKeys). The last key is assigned with an index of (verticalKeys*horizontalKeys)-1. No checking is performed on the length of *pText[] entries to match (verticalKeys*horizontalKeys).</p>
	<u><a href="#">TeAddChar</a></u>	<p>This function will insert a character to the end of the buffer. The character inserted is dependent on the currently pressed key. Drawing states</p>

		<u>TE_UPDATE_TEXT</u> or <u>TE_DRAW</u> must be set to see the effect of this insertion.
◆	<u>TeIsKeyPressed</u>	This function will test if a key given by its index in the TextEntry object has been pressed.
◆	<u>TeSpaceChar</u>	This function will insert a space character to the end of the buffer. Drawing states <u>TE_UPDATE_TEXT</u> or <u>TE_DRAW</u> must be set to see the effect of this insertion.
◆	<u>TeDelKeyMembers</u>	This function will delete the <u>KEYMEMBER</u> list assigned to the object from memory. Pointer to the <u>KEYMEMBER</u> list is then initialized to NULL.
◆	<u>TeSetKeyText</u>	This function will set the text assigned to a key with the given index.
◆	<u>TeMsgDefault</u>	This function performs the actual state change based on the translated message given. The following state changes are supported:
◆	<u>TeTranslateMsg</u>	This function evaluates the message from a user if the message will affect the object or not. If the message is valid, the keys in the Text Entry object will be scanned to detect which key was pressed. If True, the corresponding text will be displayed, the 'text' will also be stored in the TeOutput parameter of the object.



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Text Entry Macros

Text Entry

---

## Macros

---

Name	Description
<a href="#"><u>TeGetBuffer</u></a>	This macro will return the currently used buffer in the TextEntry object.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Text Entry Structures

Text Entry

## Structures

Name	Description
<a href="#"><u>TEXTENTRY</u></a>	Defines the parameters required for a TextEntry Object.
<a href="#"><u>KEYMEMBER</u></a>	Defines the parameters and the strings assigned for each key.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

[Text Entry](#)

---

## Topics

Name	Description
<a href="#">TextEntry States</a>	List of Text Entry bit states.
<a href="#">Key Command Types</a>	List of available Key command types.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Text Entry

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## TextEntry States Macros

### TextEntry States

#### Macros

Name	Description
<a href="#"><u>TE_KEY_PRESSED</u></a>	Bit for press state of one of the keys.
<a href="#"><u>TE_DISABLED</u></a>	Bit for disabled state.
<a href="#"><u>TE_ECHO_HIDE</u></a>	Bit to hide the entered characters and instead echo "*" characters to the display.
<a href="#"><u>TE_DRAW</u></a>	Bit to indicate object must be redrawn.
<a href="#"><u>TE_HIDE</u></a>	Bit to indicate object must be removed from screen.
<a href="#"><u>TE_UPDATE_KEY</u></a>	Bit to indicate redraw of a key is needed.
<a href="#"><u>TE_UPDATE_TEXT</u></a>	Bit to indicate redraw of the text displayed is needed.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [TextEntry States](#)

## Key Command Types Macros

### Key Command Types

#### Macros

Name	Description
<a href="#"><u>TE_DELETE_COM</u></a>	This macro is used to assign a "delete" command on a key.
<a href="#"><u>TE_ENTER_COM</u></a>	This macro is used to assign an "enter" (carriage return) command on a key.
<a href="#"><u>TE_SPACE_COM</u></a>	This macro is used to assign an insert "space" command on a key.

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Text Entry](#) > [Key Command Types](#)

# Window Functions

## Window

### Functions

	Name	Description
	<a href="#"><u>WndCreate</u></a>	This function creates a <a href="#"><u>WINDOW</u></a> 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.
	<a href="#"><u>WndDraw</u></a>	<p>This function renders the object on the screen using the current parameter settings. 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.</p> <p>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.</p>
	<a href="#"><u>WndSetText</u></a>	This function sets the string used for the title bar.
	<a href="#"><u>WndTranslateMsg</u></a>	This function evaluates the message from a user if the message will affect the object or not. The table below enumerates the translated messages for each event of the touch screen inputs.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Window Macros

Window

---

## Macros

---

Name	Description
<a href="#"><u>WndGetText</u></a>	This macro returns the address of the current text string used for the title bar.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Window Structures

Window

---

## Structures

---

Name	Description
<a href="#">WINDOW</a>	The structure contains data for the window

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Window

---

## Topics

Name	Description
<a href="#">Window States</a>	List of Window bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

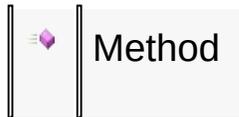
[Contents](#) | [Index](#) | [Home](#)

## Legend

Window

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Window States Macros

### Window States

#### Macros

Name	Description
<a href="#"><u>WND_DISABLED</u></a>	Bit for disabled state
<a href="#"><u>WND_DRAW</u></a>	Bits to indicate whole window must be redrawn
<a href="#"><u>WND_DRAW_CLIENT</u></a>	Bit to indicate client area must be redrawn
<a href="#"><u>WND_DRAW_TITLE</u></a>	Bit to indicate title area must be redrawn
<a href="#"><u>WND_FOCUSED</u></a>	Bit for focus state
<a href="#"><u>WND_HIDE</u></a>	Bit to indicate window must be removed from screen
<a href="#"><u>WND_TITLECENTER</u></a>	Bit to center the text on the Title Area

[Library API](#) > [Graphics Object Layer API](#) > [GOL Objects](#) > [Window](#) > [Window States](#)

## Object States Macros

### Object States

---

#### Macros

---

Name	Description
<a href="#">GetState</a>	This macro retrieves the current value of the state bits of an object. It is possible to get several state bits.
<a href="#">ClrState</a>	This macro clear the state bits of an object. Object must be redrawn to display the changes. It is possible to clear several state bits with this macro.
<a href="#">SetState</a>	This macro sets the state bits of an object. Object must be redrawn to display the changes. It is possible to set several state bits with this macro.

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#)

## Topics

Object States

---

## Topics

---

Name	Description
<a href="#">Common Object States</a>	List of common Object bit states.

---

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Common Object States Macros

### Common Object States

#### Macros

Name	Description
<a href="#"><u>FOCUSED</u></a>	Focus state bit.
<a href="#"><u>DISABLED</u></a>	Disabled state bit.
<a href="#"><u>HIDE</u></a>	Object hide state bit. Object will be hidden from the screen by drawing over it the common background color.
<a href="#"><u>DRAW</u></a>	Object redraw state bit. The whole Object must be redrawn.
<a href="#"><u>DRAW_FOCUS</u></a>	Focus redraw state bit. The focus rectangle must be redrawn.
<a href="#"><u>DRAW_UPDATE</u></a>	Partial Object redraw state bit. A part or parts of of the Object must be redrawn to show updated state.

[Library API](#) > [Graphics Object Layer API](#) > [Object States](#) > [Common Object States](#)

# Object Management Functions

## Object Management

### Functions

	Name	Description
◆	<a href="#"><u>GOLAddObject</u></a>	This function adds an object to the tail of the active list pointed to by <a href="#"><u>_pGolObjects</u></a> . The new list tail is set to point to NULL.
◆	<a href="#"><u>GOLFindObject</u></a>	This function finds an object in the active list pointed to by <a href="#"><u>_pGolObjects</u></a> using the given object ID.
◆	<a href="#"><u>GOLRedrawRec</u></a>	This function marks all objects in the active list intersected by the given rectangular area to be redrawn.
◆	<a href="#"><u>GOLDraw</u></a>	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. <a href="#"><u>GOLDrawCallback()</u></a> function is called by <a href="#"><u>GOLDraw()</u></a> when drawing of objects in the active list is completed.
◆	<a href="#"><u>GOLDrawCallback</u></a>	<a href="#"><u>GOLDrawCallback()</u></a> function MUST BE implemented by the user. This is called inside the

		<p><a href="#">GOLDraw()</a> function when the drawing of objects in the active list is completed. User drawing must be done here. Drawing color, line type, clipping region, graphic cursor position and current font will not be changed by GOL if this function returns a zero. To pass drawing control to GOL this function must return a non-zero value. If GOL messaging is not using the active link list, it is safe to modify the list here.</p>
◆	<a href="#">GOLFree</a>	<p>This function frees all the memory used by objects in the active list and initializes <a href="#">_pGolObjects</a> pointer to NULL to start a new empty list. This function must be called only inside the <a href="#">GOLDrawCallback()</a> function when using <a href="#">GOLDraw()</a> and <a href="#">GOLMsg()</a> functions. This requirement assures that primitive rendering settings are not altered by the rendering state machines of the objects.</p>
◆	<a href="#">GOLDeleteObject</a>	<p>deletes an object to the linked list objects for the current screen.</p>
◆	<a href="#">GOLDeleteObjectByID</a>	<p>Deletes an object in the current active linked list of objects using the ID parameter of the object.</p>
◆	<a href="#">GOLSetFocus</a>	<p>This function sets the keyboard input focus to the object. If the</p>

		<p>object cannot accept keyboard messages focus will not be changed. This function resets <b><u>FOCUSED</u></b> state for the object was in focus previously, set <b><u>FOCUSED</u></b> state for the required object and marks the objects to be redrawn.</p>
◆	<b><u>GOLInit</u></b>	<p>This function initializes the graphics library and creates a default style scheme with default settings referenced by the global scheme pointer. GOLInit() function must be called before GOL functions can be used. It is not necessary to call GraphInit() function if this function is used.</p>
◆	<b><u>GOLCanBeFocused</u></b>	<p>This function returns non-zero if the object can be focused. Only button, check box, radio button, slider, edit box, list box, scroll bar can accept focus. If the object is disabled it cannot be set to focused state.</p>
◆	<b><u>GOLGetFocusNext</u></b>	<p>This function returns the pointer to the next object in the active linked list which is able to receive keyboard input.</p>
◆	<b><u>GOLGetFocusPrev</u></b>	<p>This function returns the pointer to the previous object in the active linked list which is able to receive keyboard input.</p>

◆	<a href="#"><u>GOLPanelDrawTsk</u></a>	This function draws a panel on the screen with parameters set by <a href="#"><u>GOLPanelDraw()</u></a> macro. This function must be called repeatedly (depending on the return value) for a successful rendering of the panel.
◆	<a href="#"><u>GOLTwoTonePanelDrawTsk</u></a>	This function draws a two tone panel on the screen with parameters set by <a href="#"><u>GOLPanelDraw()</u></a> macro. This function must be called repeatedly (depending on the return value) for a successful rendering of the panel.

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Object Management Macros

### Object Management

#### Macros

Name	Description
<a href="#"><u>GOLRedraw</u></a>	This macro 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 to be the active list.
<a href="#"><u>GOLDrawComplete</u></a>	This macro resets the drawing states of the object (6 MSBits of the object's state).
<a href="#"><u>GetObjType</u></a>	This macro returns the object type.
<a href="#"><u>GetObjID</u></a>	This macro returns the object ID.
<a href="#"><u>GetObjNext</u></a>	This macro returns the next object after the specified object.
<a href="#"><u>GOLNewList</u></a>	This macro starts a new linked list of objects and resets the keyboard focus to none. This macro assigns the current active list <a href="#"><u>_pGolObjects</u></a> and current receiving keyboard input <a href="#"><u>_pObjectFocused</u></a> object pointers to NULL. Any keyboard inputs at this point will be ignored. Previous active list must be saved in another pointer if to be referenced later. If not needed anymore memory used by that list should be freed by <a href="#"><u>GOLFree()</u></a> function.

<a href="#"><u>GOLGetList</u></a>	This macro gets the current active list.
<a href="#"><u>GOLSetList</u></a>	This macro sets the given object list as the active list and resets the keyboard focus to none. This macro assigns the receiving keyboard input object <a href="#"><u>_pObjectFocused</u></a> pointer to NULL. If the new active list has an object's state set to focus, the <a href="#"><u>_pObjectFocused</u></a> pointer must be set to this object or the object's state must be change to unfocused. This is to avoid two objects displaying a focused state when only one object in the active list must be set to a focused state at anytime.
<a href="#"><u>IsObjUpdated</u></a>	This macro tests if the object is pending to be redrawn. This is done by testing the 6 MSBits of object's state to detect if the object must be redrawn.
<a href="#"><u>GOLGetFocus</u></a>	This macro returns the pointer to the current object receiving keyboard input.
<a href="#"><u>GOLPanelDraw</u></a>	This is macro GOLPanelDraw.

---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

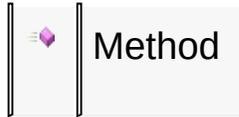
## Legend

Object Management

---

## Legend

---



---

[Library API](#) > [Graphics Object Layer API](#) > [Object Management](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Messages Enumerations

## GOL Messages

---

### Enumerations

---

Name	Description
<a href="#"><u>TRANS_MSG</u></a>	This structure defines the list of translated messages for GOL Objects used in the library.
<a href="#"><u>INPUT_DEVICE_EVENT</u></a>	This structure defines the types of GOL message events used in the library.
<a href="#"><u>INPUT_DEVICE_TYPE</u></a>	This structure defines the types of input devices used in the library.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Messages Functions

## GOL Messages

### Functions

	Name	Description
	<a href="#"><u>GOLMsg</u></a>	<p>This function receives a GOL 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 <a href="#"><u>GOLMsgCallback()</u></a> 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.</p> <p>This function should be called when GOL drawing is completed. It can be done... <a href="#"><u>more</u></a></p>
	<a href="#"><u>GOLMsgCallback</u></a>	<p>The user MUST implement this function. <a href="#"><u>GOLMsg()</u></a> calls this function when a valid message for an object in the active list is received. User action for the message should be implemented here. If this function returns non-zero, the message for the object will be processed by default. If zero is returned, GOL will not perform any action.</p>

---

## [Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Messages Structures

## GOL Messages

### Structures

Name	Description
<a href="#">GOL_MSG</a>	<p>This structure defines the GOL message used in the library.</p> <ul style="list-style-type: none"><li>• The types must be one of the <a href="#">INPUT_DEVICE_TYPE</a>:<ul style="list-style-type: none"><li>◦ TYPE_UNKNOWN</li><li>◦ TYPE_KEYBOARD</li><li>◦ TYPE_TOUCHSCREEN</li><li>◦ TYPE_MOUSE</li></ul></li><li>• uiEvent must be one of the <a href="#">INPUT_DEVICE_EVENT</a>.<ul style="list-style-type: none"><li>◦ for touch screen:<ul style="list-style-type: none"><li>▪ EVENT_INVALID</li><li>▪ EVENT_MOVE</li><li>▪ EVENT_PRESS</li><li>▪ EVENT_STILLPRESS</li><li>▪ EVENT_RELEASE</li></ul></li><li>◦ for keyboard:<ul style="list-style-type: none"><li>▪ EVENT_KEYSCAN (param2 contains scan code)</li><li>▪ EVENT_KEYCODE (param2 contains character code)</li></ul></li></ul></li><li>• param1:<ul style="list-style-type: none"><li>◦ for touch screen is the x position</li><li>◦ for keyboard ID of object receiving the message</li></ul></li></ul>

- param2
  - for touch screen y position
  - for keyboard scan or key code

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

GOL Messages

---

## Topics

---

Name	Description
<a href="#">Scan Key Codes</a>	The defined scan codes for AT keyboard.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

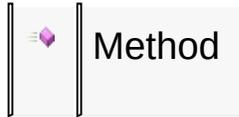
[Contents](#) | [Index](#) | [Home](#)

## Legend

GOL Messages

---

## Legend



---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Scan Key Codes Macros

### Scan Key Codes

#### Macros

Name	Description
<a href="#"><u>SCAN_BS_PRESSED</u></a>	Back space key pressed.
<a href="#"><u>SCAN_BS_RELEASED</u></a>	Back space key released.
<a href="#"><u>SCAN_CR_PRESSED</u></a>	Carriage return pressed.
<a href="#"><u>SCAN_CR_RELEASED</u></a>	Carriage return released.
<a href="#"><u>SCAN_DEL_PRESSED</u></a>	Delete key pressed.
<a href="#"><u>SCAN_DEL_RELEASED</u></a>	Delete key released.
<a href="#"><u>SCAN_DOWN_PRESSED</u></a>	Down key pressed.
<a href="#"><u>SCAN_DOWN_RELEASED</u></a>	Down key released.
<a href="#"><u>SCAN_END_PRESSED</u></a>	End key pressed.
<a href="#"><u>SCAN_END_RELEASED</u></a>	End key released.
<a href="#"><u>SCAN_HOME_PRESSED</u></a>	Home key pressed.
<a href="#"><u>SCAN_HOME_RELEASED</u></a>	Home key released.
<a href="#"><u>SCAN_LEFT_PRESSED</u></a>	Left key pressed.
<a href="#"><u>SCAN_LEFT_RELEASED</u></a>	Left key released.
<a href="#"><u>SCAN_PGDOWN_PRESSED</u></a>	Page down key pressed.

<a href="#"><u>SCAN_PGDOWN_RELEASED</u></a>	Page down key released.
<a href="#"><u>SCAN_PGUP_PRESSED</u></a>	Page up key pressed.
<a href="#"><u>SCAN_PGUP_RELEASED</u></a>	Page up key released.
<a href="#"><u>SCAN_RIGHT_PRESSED</u></a>	Right key pressed.
<a href="#"><u>SCAN_RIGHT_RELEASED</u></a>	Right key released.
<a href="#"><u>SCAN_SPACE_PRESSED</u></a>	Space key pressed.
<a href="#"><u>SCAN_SPACE_RELEASED</u></a>	Space key released.
<a href="#"><u>SCAN_TAB_PRESSED</u></a>	Tab key pressed.
<a href="#"><u>SCAN_TAB_RELEASED</u></a>	Tab key released.
<a href="#"><u>SCAN_UP_PRESSED</u></a>	Up key pressed.
<a href="#"><u>SCAN_UP_RELEASED</u></a>	Up key released.

---

[Library API](#) > [Graphics Object Layer API](#) > [GOL Messages](#) > [Scan Key Codes](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Style Scheme Functions

## Style Scheme

### Functions

	Name	Description
	<a href="#"><u>GOLCreateScheme</u></a>	This function creates a new style scheme object and initializes the parameters to default values. Default values are based on the <a href="#"><u>GOLSchemeDefault</u></a> defined in GOLSchemeDefault.c file. Application code can override this initialization, See <a href="#"><u>GOLSchemeDefault</u></a> .

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

## Style Scheme Macros

### Style Scheme

#### Macros

Name	Description
<a href="#"><u>GOLSetScheme</u></a>	This macro sets the GOL scheme to be used for the object.
<a href="#"><u>GOLGetScheme</u></a>	This macro gets the GOL scheme used by the given object.
<a href="#"><u>GOLGetSchemeDefault</u></a>	This macro returns the default GOL scheme pointer.
<a href="#"><u>GOL_EMBOSS_SIZE</u></a>	This option defines the 3-D effect emboss size for objects. The default value of this is 3 set in GOL.h. If it is not defined in GraphicsConfig.h file then the default value is used.
<a href="#"><u>RGBConvert</u></a>	This macro converts the color data to the selected <a href="#"><u>COLOR_DEPTH</u></a> . This macro is only valid when <a href="#"><u>COLOR_DEPTH</u></a> is 8, 16, or 24.

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

# Style Scheme Structures

## Style Scheme

---

### Structures

---

Name	Description
<a href="#"><u>GOL_SCHEME</u></a>	GOL scheme defines the style scheme to be used by an object.

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Style Scheme

---

## Topics

---

Name	Description
<a href="#">Default Style Scheme Settings</a>	Lists the default settings for the style scheme.

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Style Scheme Variables

### Style Scheme

#### Variables

Name	Description
<a href="#"><u>GOLFontDefault</u></a>	This is variable GOLFontDefault.
<a href="#"><u>GOLSchemeDefault</u></a>	This defines a default GOL scheme that gets populated when an application calls the <a href="#"><u>GOLCreateScheme()</u></a> . The application can override this definition by defining the macro GFX_SCHEMEDEFAULT in the GraphicsConfig.h header file and defining GOLSchemeDefault structure in the application code. It is important to use the same structure name since the library assumes that this object exists and assigns the default style scheme pointer to this object.

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

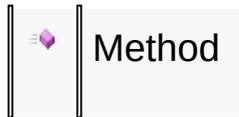
## Legend

Style Scheme

---

## Legend

---



---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Default Style Scheme Settings Variables

Default Style Scheme Settings

---

### Variables

---

Name	Description
<a href="#"><u>FONTDEFAULT</u></a>	Default GOL font.

---

[Library API](#) > [Graphics Object Layer API](#) > [Style Scheme](#) > [Default Style Scheme Settings](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# GOL Global Variables Variables

## GOL Global Variables

### Variables

Name	Description
<a href="#">_pDefaultGolScheme</a>	Pointer to the GOL default scheme ( <a href="#">GOL_SCHEME</a> ). This scheme is created in <a href="#">GOLInit()</a> function. GOL scheme defines the style scheme to be used by an object. Use <a href="#">GOLGetSchemeDefault()</a> to get this pointer.
<a href="#">_pGolObjects</a>	Pointer to the current linked list of objects displayed and receiving messages. <a href="#">GOLDraw()</a> and <a href="#">GOLMsg()</a> process objects referenced by this pointer.
<a href="#">_pObjectFocused</a>	Pointer to the object receiving keyboard input. This pointer is used or modified by the following APIs: <ul style="list-style-type: none"><li>• <a href="#">GOLSetFocus()</a></li><li>• <a href="#">GOLGetFocus()</a></li><li>• <a href="#">GOLGetFocusNext()</a></li><li>• <a href="#">GOLGetFocusPrev()</a></li><li>• <a href="#">GOLCanBeFocused()</a></li></ul>

[Library API](#) > [Graphics Object Layer API](#) > [GOL Global Variables](#)

# Graphics Primitive Layer API Enumerations

## Graphics Primitive Layer API

---

### Enumerations

---

Name	Description
<a href="#"><u>GFX_RESOURCE</u></a>	Memory type enumeration to determine the source of data. Used in interpreting bitmap and font from different memory sources.

---

### [Library API](#) > [Graphics Primitive Layer API](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Graphics Primitive Layer API Structures

## Graphics Primitive Layer API

### Structures

Name	Description
<a href="#"><u>GFX_IMAGE_HEADER</u></a>	Structure for images stored in various system memory (Flash, External Memory (SPI, Parallel Flash, or memory in EPMP).
<a href="#"><u>IMAGE_FLASH</u></a>	Structure for images stored in FLASH memory.
<a href="#"><u>IMAGE_RAM</u></a>	Structure for images stored in RAM memory.
<a href="#"><u>GFX_EXTDATA</u></a>	This structure is used to describe external memory.

### [Library API](#) > [Graphics Primitive Layer API](#)

## Topics

[Graphics Primitive Layer API](#)

## Topics

Name	Description
<a href="#">Text Functions</a>	This lists the Primitive level text functions.
<a href="#">Gradient</a>	Gradients can be drawn dynamically with the Microchip Graphics Library.
<a href="#">Line Functions</a>	This lists the Primitive line text functions.
<a href="#">Rectangle Functions</a>	This lists the Primitive level rectangle functions.
<a href="#">Circle Functions</a>	This lists the Primitive level circle functions.
<a href="#">Graphic Cursor</a>	This lists the functions to control the graphics cursor.
<a href="#">Alpha Blending Functions</a>	This lists the functions to control Alpha Blending. This feature is enabled only in selected drivers.
<a href="#">Bitmap Functions</a>	This lists the functions to display bitmaps.
<a href="#">External Memory</a>	This lists the external memory access functions and descriptions.
<a href="#">Set Up Functions</a>	This lists the Primitive set up and initialization functions.

[Library API](#) > [Graphics Primitive Layer API](#)

Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Text Functions Functions

### Text Functions

#### Functions

	Name	Description
	<a href="#"><u>SetFont</u></a>	This function sets the current font used in <a href="#"><u>OutTextXY()</u></a> , <a href="#"><u>OutText()</u></a> and <a href="#"><u>OutChar()</u></a> functions.
	<a href="#"><u>OutChar</u></a>	This function outputs a character from the current graphic cursor position. <a href="#"><u>OutChar()</u></a> uses the current active font set with <a href="#"><u>SetFont()</u></a> .
	<a href="#"><u>OutText</u></a>	This function outputs a string of characters starting at the current graphic cursor position. The string must be terminated by a line feed or zero. For Non-Blocking configuration, <a href="#"><u>OutText()</u></a> may return control to the program due to display device busy status. When this happens zero is returned and <a href="#"><u>OutText()</u></a> must be called again to continue the outputting of the string. For Blocking configuration, this function always returns a 1. <a href="#"><u>OutText()</u></a> uses the current active font set with <a href="#"><u>SetFont()</u></a> .
	<a href="#"><u>OutTextXY</u></a>	This function outputs a string of characters starting at the given x, y position. The string must be terminated by a line feed or zero. For Non-Blocking configuration, <a href="#"><u>OutTextXY()</u></a> may return control to the program due to display device busy status. When this happens zero is returned and

		OutTextXY() must be called again to continue the outputting of the string. For Blocking configuration, this function always returns a 1. OutTextXY() uses the current active font set with <a href="#">SetFont()</a> .
≡	<a href="#">GetTextHeight</a>	This macro returns the height of the specified font. All characters in a given font table have a constant height.
≡	<a href="#">GetTextWidth</a>	This function returns the width of the specified string for the specified font. The string must be terminated by a line feed or zero.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Text Functions Macros

### Text Functions

#### Macros

Name	Description
<a href="#"><u>GetFontOrientation</u></a>	Returns font orientation.
<a href="#"><u>SetFontOrientation</u></a>	Sets font orientation vertical or horizontal.
<a href="#"><u>GFX_Font_GetAntiAliasType</u></a>	Returns the font anti-alias type.
<a href="#"><u>GFX_Font_SetAntiAliasType</u></a>	Sets font anti-alias type to either Translucent or opaque.
<a href="#"><u>XCHAR</u></a>	This macro sets the data type for the strings and characters. There are three types used for XCHAR and the type is selected by adding one of the macros in GraphicsConfig.h.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

# Text Functions Structures

## Text Functions

---

## Structures

---

Name	Description
<a href="#"><u>FONT_HEADER</u></a>	Structure describing the font header.
<a href="#"><u>FONT_FLASH</u></a>	Structure for font stored in FLASH memory.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Text Functions

---

## Topics

Name	Description
<a href="#">Anti-Alias Type</a>	Anti-alias type definitions.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Text Functions Types

## Text Functions

---

### Types

---

Name	Description
<a href="#"><u>FONT_EXTERNAL</u></a>	Structure for font stored in EXTERNAL memory space. (example: External SPI or parallel Flash, EDS_EPMP)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Text Functions

---

## Legend

---



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Anti-Alias Type Macros

Anti-Alias Type

## Macros

Name	Description
<a href="#"><u>ANTIALIAS_OPAQUE</u></a>	Mid colors are calculated only once while rendering each character. This is ideal for rendering text over a constant background.
<a href="#"><u>ANTIALIAS_TRANSLUCENT</u></a>	Mid values 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.

[Library API](#) > [Graphics Primitive Layer API](#) > [Text Functions](#) > [Anti-Alias Type](#)

# Gradient Enumerations

Gradient

---

## Enumerations

---

Name	Description
<a href="#">GFX_GRADIENT_TYPE</a>	Enumeration for gradient type

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Gradient Functions

## Gradient

### Functions

	Name	Description
	<a href="#"><u>BarGradient</u></a>	This renders a bar onto the screen, but instead of one color, a gradient is drawn depending on the direction ( <a href="#"><u>GFX_GRADIENT_TYPE</u></a> ), length, and colors chosen. This function is a blocking call.
	<a href="#"><u>BevelGradient</u></a>	This renders a filled bevel with gradient color on the fill. It works the same as the fillbevel function, except a gradient out of color1 and color2 is drawn depending on the direction ( <a href="#"><u>GFX_GRADIENT_TYPE</u></a> ). This function is a blocking call.

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#)

# Gradient Structures

## Gradient

### Structures

Name	Description
<a href="#"><u>GFX_GRADIENT_STYLE</u></a>	This structure is used to describe the gradient style.

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

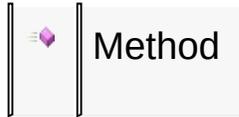
[Contents](#) | [Index](#) | [Home](#)

# Legend

Gradient

---

# Legend



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Gradient](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Line Functions

## Line Functions

---

### Functions

---

	Name	Description
	<a href="#">Line</a>	This function draws a line with the current line type from the start point to the end point.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Line Functions Macros

## Line Functions

### Macros

Name	Description
<a href="#">LineRel</a>	This macro draws a line with the current line type from the current graphic cursor position to the position defined by displacement.
<a href="#">LineTo</a>	This macro draws a line with the current line type from the current graphic cursor position to the given x, y position.
<a href="#">SetLineThickness</a>	This macro sets sets line thickness to 1 pixel or 3 pixels.
<a href="#">SetLineType</a>	This macro sets the line type to draw.

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#)

## Topics

Line Functions

---

## Topics

Name	Description
<a href="#">Line Types</a>	Line type definitions.
<a href="#">Line Size</a>	Line sizes definition.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

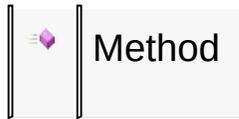
# Legend

Line Functions

---

## Legend

---



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Line Types Macros

## Line Types

---

### Macros

---

Name	Description
<a href="#"><u>SOLID_LINE</u></a>	Solid <a href="#"><u>Line</u></a> Style
<a href="#"><u>DASHED_LINE</u></a>	Dashed <a href="#"><u>Line</u></a> Style
<a href="#"><u>DOTTED_LINE</u></a>	Dotted <a href="#"><u>Line</u></a> Style

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Types](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Line Size Macros

## Line Size

---

### Macros

---

Name	Description
<a href="#"><u>NORMAL_LINE</u></a>	Normal <a href="#"><u>Line</u></a> (thickness is 1 pixel)
<a href="#"><u>THICK_LINE</u></a>	Thick <a href="#"><u>Line</u></a> (thickness is 3 pixels)

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Line Functions](#) > [Line Size](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Rectangle Functions Functions

### Rectangle Functions

#### Functions

	Name	Description
	<a href="#"><u>Bar</u></a>	This function draws a bar given the left, top and right, bottom corners with the current set color ( <a href="#"><u>SetColor()</u></a> ). When alpha blending is enabled the bar is alpha blended with the existing pixels specified by the parameters. The alpha percentage used is the last value set by <a href="#"><u>SetAlpha()</u></a> .
	<a href="#"><u>DrawPoly</u></a>	This function draws a polygon with the current line type using the given number of points. The polygon points (polyPoints) are stored in an array arranged in the following order:

[Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#)

# Rectangle Functions Macros

## Rectangle Functions

---

### Macros

---

Name	Description
<a href="#">Rectangle</a>	This macro draws a rectangle with the given left, top and right, bottom corners. Current line type is used.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

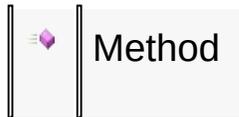
# Legend

Rectangle Functions

---

## Legend

---



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Rectangle Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Circle Functions Functions

## Circle Functions

### Functions

	Name	Description
	<a href="#"><u>Arc</u></a>	<p>Draws the octant arc of the beveled figure with the given centers, radii and octant mask. When octant = 0xFF and the following are true:</p> <ol style="list-style-type: none"> <li>1. <math>x_L = x_R</math>, <math>y_T = y_B</math>, <math>r_1 = 0</math> and <math>r_2 = z</math>, a filled circle is drawn with a radius of <math>z</math>.</li> <li>2. radii have values (where <math>r_1 &lt; r_2</math>), a full ring with thickness of <math>(r_2 - r_1)</math> is drawn.</li> <li>3. <math>x_L \neq x_R</math>, <math>y_T \neq y_B</math>, <math>r_1 = 0</math> and <math>r_2 = 0</math> (where <math>x_R &gt; x_L</math> and <math>y_B &gt; y_T</math>) a rectangle is drawn. <math>x_L</math>, <math>y_T</math> specifies the left top corner and <math>x_R</math>,... <a href="#"><u>more</u></a></li> </ol>
	<a href="#"><u>DrawArc</u></a>	<p>This renders an arc with from startAngle to endAngle with the thickness of <math>r_2 - r_1</math>. The function returns 1 when the arc is rendered successfully and returns a 0 when it is not yet finished. The next call to the function will continue the rendering.</p>
	<a href="#"><u>Bevel</u></a>	<p>Draws a beveled figure on the screen. When <math>x_1 = x_2</math> and <math>y_1 = y_2</math>, a circular object is drawn. When <math>x_1 &lt; x_2</math> and <math>y_1 &lt; y_2</math> and rad (radius) = 0, a rectangular object is drawn.</p>
	<a href="#"><u>FillBevel</u></a>	<p>Draws a filled beveled figure on the screen. For a filled circular object <math>x_1 = x_2</math> and <math>y_1 = y_2</math>. For a filled rectangular object radius =</p>

---

## [Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Circle Functions Macros

## Circle Functions

### Macros

Name	Description
<a href="#">Circle</a>	This macro draws a circle with the given center and radius.
<a href="#">FillCircle</a>	This macro draws a filled circle. Uses the <a href="#">FillBevel()</a> function.
<a href="#">SetBevelDrawType</a>	This macro sets the fill bevel type to be drawn.

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#)

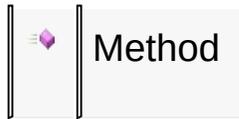
# Legend

Circle Functions

---

## Legend

---



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Circle Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Graphic Cursor Macros

### Graphic Cursor

#### Macros

Name	Description
<a href="#">GetX</a>	This macro returns the current graphic cursor x-coordinate.
<a href="#">GetY</a>	This macro returns the current graphic cursor y-coordinate.
<a href="#">MoveRel</a>	This macro moves the graphic cursor relative to the current location. The given dX and dY displacement can be positive or negative numbers.
<a href="#">MoveTo</a>	This macro moves the graphic cursor to new x,y position.

[Library API](#) > [Graphics Primitive Layer API](#) > [Graphic Cursor](#)

# Alpha Blending Functions Functions

## Alpha Blending Functions

### Functions

	Name	Description
	<a href="#">AlphaBlendWindow</a>	This Alpha-Blends a foreground and a background stored in frames to a destination window. A frame is a memory area that contain array of pixels information. An example would be a display buffer. This operation can be performed on a single frame (where foregroundArea, backgroundArea and destinationArea all points to the same frame), 2 frames (where two of the three areas are pointing to the same frame and one is another frame), or 3 frames (where each area is a separate frame). The Alpha-Blending is performed on the windows inside the specified frames. These windows are defined by the offsets... <a href="#">more</a>

[Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#)

# Alpha Blending Functions Macros

## Alpha Blending Functions

### Macros

Name	Description
<a href="#">SetAlpha</a>	This macro sets the alpha value. Enabling this feature requires the macros <a href="#">USE_ALPHABLEND_LITE</a> defined in the GraphicsConfig.h. See <a href="#">USE_ALPHABLEND_LITE</a> for information on supported primitive rendering functions.
<a href="#">GetAlpha</a>	This macro returns the current alpha value. Enabling this feature requires the macros <a href="#">USE_ALPHABLEND_LITE</a> defined in the GraphicsConfig.h.

[Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#)

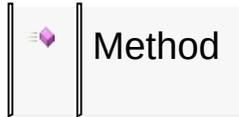
## Legend

Alpha Blending Functions

---

## Legend

---



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Alpha Blending Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bitmap Functions Functions

## Bitmap Functions

### Functions

	Name	Description
	<a href="#"><u>PutImagePartial</u></a>	This function outputs a full or a partial image starting from left,top coordinates. The partial image starts at xoffset and yoffset. Size is specified by the given width and height parameters.
	<a href="#"><u>GetImageHeight</u></a>	This function returns the image height.
	<a href="#"><u>GetImageWidth</u></a>	This function returns the image width.

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bitmap Functions Macros

## Bitmap Functions

---

### Macros

---

Name	Description
<a href="#">PutImage</a>	This renders the image pointed to by "image" starting from left, top coordinates.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bitmap Functions Structures

## Bitmap Functions

---

### Structures

---

Name	Description
<a href="#">BITMAP_HEADER</a>	Structure describing the bitmap header.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Bitmap Functions

---

## Topics

Name	Description
<a href="#">Bitmap Settings</a>	Bitmap rendering settings.
<a href="#">Bitmap Source</a>	Bitmap data structure is dependent on the location.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

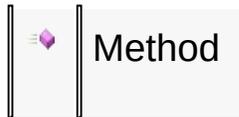
## Legend

Bitmap Functions

---

## Legend

---



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Bitmap Settings Macros

## Bitmap Settings

### Macros

Name	Description
<a href="#"><u>IMAGE_NORMAL</u></a>	Normal image stretch code
<a href="#"><u>IMAGE_X2</u></a>	Stretched image stretch code

[Library API](#) > [Graphics Primitive Layer API](#) > [Bitmap Functions](#) > [Bitmap Settings](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# External Memory Functions

## External Memory

### Functions

	Name	Description
	<a href="#">ExternalMemoryCallback</a>	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 bytes quantity into the buffer provided. Data start address in external memory is a sum of the address in <a href="#">GFX_EXTDATA</a> structure and offset.

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#)

## External Memory Macros

External Memory

### Macros

Name	Description
<a href="#"><u>EXTERNAL_FONT_BUFFER_SIZE</u></a>	This defines the size of the buffer used by font functions to retrieve font data from the external memory. The buffer size can be increased to accommodate large font sizes. The user must be aware of the expected glyph sizes of the characters stored in the font table. To modify the size used, declare this macro in the GraphicsConfig.h file with the desired size.

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#)

## Topics

External Memory

---

## Topics

Name	Description
<a href="#">Memory Type</a>	Memory type enumeration to determine the source of data. Used in interpreting bitmap and font from different memory sources.

---

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

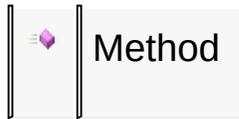
[Contents](#) | [Index](#) | [Home](#)

## Legend

External Memory

---

## Legend



---

[Library API](#) > [Graphics Primitive Layer API](#) > [External Memory](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Set Up Functions Functions

## Set Up Functions

### Functions

	Name	Description
	<a href="#">ClearDevice</a>	This function clears the screen with the current color and sets the graphic cursor position to (0, 0). Clipping is NOT supported by ClearDevice().
	<a href="#">InitGraph</a>	This function initializes the display controller, sets the line type to <a href="#">SOLID_LINE</a> , sets the screen to all <a href="#">BLACK</a> , sets the current drawing color to <a href="#">WHITE</a> , sets the graphic cursor position to upper left corner of the screen, sets active and visual pages to page #0, clears the active page and disables clipping. This function should be called before using the Graphics Primitive Layer.

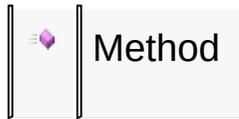
### [Library API](#) > [Graphics Primitive Layer API](#) > [Set Up Functions](#)

## Legend

Set Up Functions

---

## Legend



---

[Library API](#) > [Graphics Primitive Layer API](#) > [Set Up Functions](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Modules

Display Device Driver Layer API

---

## Modules

---

Name	Description
<a href="#">Advanced Display Driver Features</a>	This section lists advanced Display Device Driver Features implemented in select Display Device Driver.

---

[Library API](#) > [Display Device Driver Layer API](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Display Device Driver Layer API

---

## Topics

Name	Description
<a href="#">Display Device Driver Level Primitives</a>	This lists the Device Level Primitive rendering functions and macros.
<a href="#">Display Device Driver Control</a>	This lists the device control functions and macros.

---

[Library API](#) > [Display Device Driver Layer API](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Display Device Driver Level Primitives Functions

### Display Device Driver Level Primitives

#### Functions

	Name	Description
◆	<a href="#"><u>GetPixel</u></a>	Returns pixel color at the given x,y coordinate position.
◆	<a href="#"><u>PutPixel</u></a>	Puts pixel with the given x,y coordinate position.
◆	<a href="#"><u>SetClip</u></a>	Enables/disables clipping.
◆	<a href="#"><u>SetClipRgn</u></a>	Sets clipping region.
◆	<a href="#"><u>TransparentColorEnable</u></a>	Sets current transparent color. <a href="#"><u>PutImage()</u></a> will not render pixels that matches the set transparent color. To enable Transparent Color feature, define the macro <a href="#"><u>USE_TRANSPARENT_COLOR</u></a> in the GraphicsConfig.h file.
◆	<a href="#"><u>DisplayBrightness</u></a>	Sets the brightness of the display.
◆	<a href="#"><u>CopyBlock</u></a>	Copies a block of data from source specified by srcAddr and srcOffset to the destination specified by dstAddr and dstOffset. This is similar to the <a href="#"><u>CopyWindow()</u></a> and but instead of using left, top position of the source and destination, it uses offsets instead. This is a blocking function.

	<a href="#"><u>CopyPageWindow</u></a>	<p>This is a blocking call. A windows is a rectangular area with the given width and height of a page. The source and destination window can be located in different pages and each page is assumed to have the same dimensions (equal width and height).</p>
	<a href="#"><u>CopyWindow</u></a>	<p>A windows is a rectangular area with the given width and height located in the given base source address. The source and destination window can be located in the same base address. If this is the case, then srcAddr = dstAddr. The operation is similar to <a href="#"><u>CopyPageWindow()</u></a> but instead of using page numbers, addresses are used for versatility. This is a blocking function.</p>
	<a href="#"><u>SetActivePage</u></a>	<p>Sets active graphic page.</p>
	<a href="#"><u>SetVisualPage</u></a>	<p>Sets graphic page to display.</p>

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#)

## Display Device Driver Level Primitives Macros

### Display Device Driver Level Primitives

#### Macros

Name	Description
<a href="#"><u>GetColor</u></a>	Returns current drawing color.
<a href="#"><u>SetColor</u></a>	Sets current drawing color.
<a href="#"><u>GetMaxX</u></a>	Returns maximum horizontal coordinate.
<a href="#"><u>GetMaxY</u></a>	Returns maximum vertical coordinate.
<a href="#"><u>GetClipBottom</u></a>	Returns bottom clipping border.
<a href="#"><u>GetClipLeft</u></a>	Returns left clipping border.
<a href="#"><u>GetClipRight</u></a>	Returns right clipping border.
<a href="#"><u>GetClipTop</u></a>	Returns top clipping border.
<a href="#"><u>CLIP_DISABLE</u></a>	Disables clipping.
<a href="#"><u>CLIP_ENABLE</u></a>	Enables clipping.
<a href="#"><u>TransparentColorDisable</u></a>	Disables the transparent color function.
<a href="#"><u>GetTransparentColorStatus</u></a>	Returns the current transparent color function enable status.

<a href="#"><u>GetTransparentColor</u></a>	Returns the current transparent color value.
<a href="#"><u>TRANSPARENT_COLOR_DISABLE</u></a>	Check of transparent color is not performed
<a href="#"><u>TRANSPARENT_COLOR_ENABLE</u></a>	Check pixel if color is equal to transparent color, if equal do not render pixel
<a href="#"><u>GetPageAddress</u></a>	Returns the address of the given page.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
 Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Display Device Driver Level Primitives

---

## Topics

Name	Description
<a href="#">Color Definition</a>	The device driver must also implement the definition of standard color set based on the color format used.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

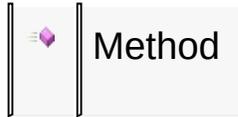
## Legend

Display Device Driver Level Primitives

---

## Legend

---



---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Color Definition Macros

Color Definition

### Macros

Name	Description
<a href="#"><u>BLACK</u></a>	not <a href="#"><u>USE_PALETTE</u></a>
<a href="#"><u>BLUE</u></a>	This is macro BLUE.
<a href="#"><u>BRIGHTBLUE</u></a>	This is macro BRIGHTBLUE.
<a href="#"><u>BRIGHTCYAN</u></a>	This is macro BRIGHTCYAN.
<a href="#"><u>BRIGHTGREEN</u></a>	This is macro BRIGHTGREEN.
<a href="#"><u>BRIGHTMAGENTA</u></a>	This is macro BRIGHTMAGENTA.
<a href="#"><u>BRIGHTRED</u></a>	This is macro BRIGHTRED.
<a href="#"><u>BRIGHTYELLOW</u></a>	This is macro BRIGHTYELLOW.
<a href="#"><u>BROWN</u></a>	This is macro BROWN.
<a href="#"><u>CYAN</u></a>	This is macro CYAN.
<a href="#"><u>DARKGRAY</u></a>	This is macro DARKGRAY.
<a href="#"><u>GRAY0</u></a>	This is macro GRAY0.
<a href="#"><u>GRAY1</u></a>	This is macro GRAY1.
<a href="#"><u>GRAY2</u></a>	This is macro GRAY2.
<a href="#"><u>GRAY3</u></a>	This is macro GRAY3.

<a href="#"><u>GRAY4</u></a>	This is macro GRAY4.
<a href="#"><u>GRAY5</u></a>	This is macro GRAY5.
<a href="#"><u>GRAY6</u></a>	This is macro GRAY6.
<a href="#"><u>GREEN</u></a>	This is macro GREEN.
<a href="#"><u>LIGHTBLUE</u></a>	This is macro LIGHTBLUE.
<a href="#"><u>LIGHTCYAN</u></a>	This is macro LIGHTCYAN.
<a href="#"><u>LIGHTGRAY</u></a>	This is macro LIGHTGRAY.
<a href="#"><u>LIGHTGREEN</u></a>	This is macro LIGHTGREEN.
<a href="#"><u>LIGHTMAGENTA</u></a>	This is macro LIGHTMAGENTA.
<a href="#"><u>LIGHTRED</u></a>	This is macro LIGHTRED.
<a href="#"><u>MAGENTA</u></a>	This is macro MAGENTA.
<a href="#"><u>RED</u></a>	This is macro RED.
<a href="#"><u>WHITE</u></a>	This is macro WHITE.
<a href="#"><u>YELLOW</u></a>	This is macro YELLOW.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Level Primitives](#) > [Color Definition](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Display Device Driver Control Functions

## Display Device Driver Control

### Functions

	Name	Description
	<a href="#"><u>IsDeviceBusy</u></a>	Returns non-zero if LCD controller is busy (previous drawing operation is not completed).
	<a href="#"><u>ResetDevice</u></a>	Initializes LCD module.

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Control](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Display Device Driver Control

---

## Legend



---

[Library API](#) > [Display Device Driver Layer API](#) > [Display Device Driver Control](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

### Advanced Display Driver Features

## Topics

Name	Description
<a href="#">Alpha Blending</a>	The following APIs are used to implement alpha blending features in the Epson S1D13517 Controller.
<a href="#">Transitions</a>	This section describes screen transition effect when changing screens.
<a href="#">Double Buffering</a>	<p>In the Microchip Graphics Library, if double-buffering is enabled, the frame buffer and draw buffer are exchanged after the changes of all the widgets on a screen are done (i.e., the new screen appears after the whole screen is updated and not after updating an individual widget). This feature is enabled only on the following drivers:</p> <ul style="list-style-type: none"><li>• Microchip Graphics Display Driver - customizable driver for RGB Glass. Currently used in PIC24FJ256DA210 device family.</li><li>• Microchip Low-Cost Controllerless (LCC) Graphics Display Driver - customizable driver for RGB Glass. Currently used for selected PIC32MX device families.</li></ul>
<a href="#">Microchip Graphics Controller</a>	This is the generic display driver is intended for the Microchip Graphics Controller

Module implemented in PIC24F device family. This driver will drive TFT, CSTN and STN displays.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Alpha Blending Functions

## Alpha Blending

### Functions

	Name	Description
	<a href="#">GFXGetPageOriginAddress</a>	This function calculates the address of a certain 0,0 location of a page in memory
	<a href="#">GFXGetPageXYAddress</a>	This function calculates the address of a certain x,y location in memory

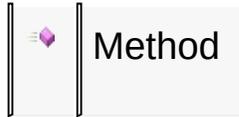
[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Alpha Blending](#)

## Legend

Alpha Blending

---

## Legend



---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Alpha Blending](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Transitions Enumerations

### Transitions

### Enumerations

Name	Description
<a href="#"><u>GFX_TRANSITION_DIRECTION</u></a>	Direction enumeration to determine the direction of the selected <a href="#"><u>GFX_TRANSITION_TYPE</u></a> .
<a href="#"><u>GFX_TRANSITION_TYPE</u></a>	Transition type enumeration to determine the type of the transition. Each type will require specific parameters when setting up the transition operation ( <a href="#"><u>GFXSetupTransition()</u></a> or <a href="#"><u>GFXTransition()</u></a> ).

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#)

## Transitions Functions

### Transitions

### Functions

	Name	Description
	<a href="#"><u>GFXTransition</u></a>	This immediately executes the transition effect using the <a href="#"><u>GFX_TRANSITION_TYPE</u></a> and the given parameters.
	<a href="#"><u>GFXSetupTransition</u></a>	This sets up the transition effect using the <a href="#"><u>GFX_TRANSITION_TYPE</u></a> and the given parameters. The actual transition execution will occur when <a href="#"><u>GFXExecutePendingTransition()</u></a> is called. When <code>DOUBLE_BUFFERING</code> is enabled, <a href="#"><u>GFXExecutePendingTransition()</u></a> is executed after the current screen is fully rendered.
	<a href="#"><u>GFXExecutePendingTransition</u></a>	This function executes the transition that was set up by <a href="#"><u>GFXSetupTransition()</u></a> . Status of the transition is returned to indicate if the transition was executed or not. This function is used by the double buffering feature ( <a href="#"><u>USE_DOUBLE_BUFFERING</u></a> ) to perform transition operation

		after the current screen is fully rendered. This function assumes that the source page and destination page are already set up.
	<a href="#"><u>GFXIsTransitionPending</u></a>	This function returns the status of a pending transition, set up by <a href="#"><u>GFXSetupTransition()</u></a> , waiting to be executed.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Transitions

---

## Legend



---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Transitions](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Double Buffering Functions

### Double Buffering

#### Functions

	Name	Description
	<a href="#"><u>SwitchOffDoubleBuffering</u></a>	Switches off the double buffering. All rendering will be performed on the frame buffer. Calls to <a href="#"><u>UpdateDisplayNow()</u></a> or <a href="#"><u>RequestDisplayUpdate()</u></a> will have no effect.
	<a href="#"><u>SwitchOnDoubleBuffering</u></a>	Switches on the double buffering. Double buffering utilizes two buffers. The frame buffer and the draw buffer. The frame buffer is the buffer that is being displayed while the draw buffer is used for all rendering. When this function is called, it copies the contents of the frame buffer to the draw buffer once and all succeeding rendering will be performed on the draw buffer. To update the frame buffer with newly drawn items on the draw buffer call <a href="#"><u>UpdateDisplayNow()</u></a> or <a href="#"><u>RequestDisplayUpdate()</u></a> .
	<a href="#"><u>InvalidateRectangle</u></a>	Invalidates the specified rectangular area. This increments the number of invalidated areas and if the number of invalidated areas exceed the <code>GFX_MAX_INVALIDATE_AREAS</code> , the whole frame buffer is

		invalidated.
	<a href="#"><u>RequestDisplayUpdate</u></a>	Synchronizes the draw and frame buffers at next VBlank
	<a href="#"><u>UpdateDisplayNow</u></a>	Synchronizes the draw and frame buffers immediately.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Double Buffering

---

## Legend



---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Double Buffering](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Microchip Graphics Controller

## Topics

Name	Description
<a href="#">Rectangle Copy Operations</a>	The following APIs are used move blocks of data from one memory location to another.
<a href="#">Decompressing DEFLATED data</a>	The Microchip Graphics Controller features a decompression module for data compressed using the DEFLATE algorithm. Compressed data are limited to fixed huffman codes. Compressed data with dynamic huffman codes are not supported.
<a href="#">Palette Mode</a>	The Microchip Graphics Controller features a palette mode for a smaller frame buffer requirement. This option uses the built-in 256 entry Color Look-up Table (CLUT) to represent pixels from the display buffer in memory. If the CLUT is enabled, each pixel in the display buffer is assumed to contain the color index. This color index is used as the address of the CLUT entry that contains the color value that will be used for the given pixel.
<a href="#">Set Up Display Interface</a>	
<a href="#">External or Internal Memory and Palettes</a>	This section shows examples on how to set up applications using external memory, internal memory or use palettes for Microchip Graphics Module.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Rectangle Copy Operations Functions

### Rectangle Copy Operations

#### Functions

	Name	Description
	<a href="#"><u>ROPBlock</u></a>	Performs a Raster Operation (ROP) on source and destination. The type of ROP is decided by the rop and the copyOp parameter.
	<a href="#"><u>Scroll</u></a>	Scrolls the rectangular area defined by left, top, right, bottom by delta pixels.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#)

## Rectangle Copy Operations Macros

### Rectangle Copy Operations

#### Macros

Name	Description
<a href="#"><u>RCC_SRC_ADDR_CONTINUOUS</u></a>	<p>Source (S) and Destination (D) (When performing executing the <b>Rectangle</b> Copy Process (RCCGPU). The source and data can be treated as a copy of data in memory or a display in memory. This gives flexibility in operation where an copy operation performed to data already present in display buffer or anywhere in memory. Both source and destination can be set to continuous or discontinuous data. These macros are only for RCCGPU operations.</p> <ul style="list-style-type: none"><li>• <a href="#"><u>RCC_SRC_ADDR_CONTINUOUS</u></a> - source data is continuous</li><li>• <a href="#"><u>RCC_SRC_ADDR_DISCONTINUOUS</u></a> - source data is discontinuous</li></ul>
<a href="#"><u>RCC_SRC_ADDR_DISCONTINUOUS</u></a>	<p>Source (S) and Destination (D) (When performing executing the <b>Rectangle</b> Copy Process (RCCGPU). The source and data can be treated as a copy of data in memory or a display in memory. This gives flexibility in operation where an copy operation performed to data already present in display buffer or anywhere in memory. Both source and destination can be set to continuous or discontinuous data. These macros are only for RCCGPU operations.</p>

	<p>display buffer or anywhere e memory. Both source and de can be set to continuous or ( data. These macros are only RCCGPU operations.</p> <ul style="list-style-type: none"> <li>• <code>RCC_SRC_ADDR_CONTINUOUS</code> - source data is continuous</li> <li>• <code>RCC_SRC_ADDR_DISCONTINUOUS</code> - source data is discontinuous</li> </ul>
<p><u><code>RCC_DEST_ADDR_CONTINUOUS</code></u></p>	<p>Source (S) and Destination (D) (When performing executing the <u>Rectangle</u> Copy Process (RCCGPU). The source and data can be treated as a copy of data in memory or a display in memory. This gives flexibility in operation where an copy operation performed to data already present in display buffer or anywhere e memory. Both source and destination can be set to continuous or discontinuous data. These macros are only RCCGPU operations.</p> <ul style="list-style-type: none"> <li>• <code>RCC_SRC_ADDR_CONTINUOUS</code> - source data is continuous</li> <li>• <code>RCC_SRC_ADDR_DISCONTINUOUS</code> - source data is discontinuous</li> </ul>
<p><u><code>RCC_DEST_ADDR_DISCONTINUOUS</code></u></p>	<p>Source (S) and Destination (D) (When performing executing the <u>Rectangle</u> Copy Process (RCCGPU). The source and data can be treated as a copy of data in memory or a display in memory. This gives flexibility in operation where an copy operation performed to data already present in display buffer or anywhere e memory. Both source and destination can be set to continuous or discontinuous data. These macros are only RCCGPU operations.</p>

operation where an copy op performed to data already pr display buffer or anywhere e memory. Both source and de can be set to continuous or c data. These macros are only RCCGPU operations.

- `RCC_SRC_ADDR_COI` - source data is continuo
- `RCC_SRC_ADDR_DIS` - source data is disconti

### RCC\_ROP\_0

Raster Operation (ROP) opt of the following 16 raster op whenever **Rectangle** Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).

- `RCC_ROP_0` - 0 (**BLAC**)
- `RCC_ROP_1` - not (S o
- `RCC_ROP_2` - (not S) a
- `RCC_ROP_3` - not (S)
- `RCC_ROP_4` - S and n
- `RCC_ROP_5` - not (D)
- `RCC_ROP_6` - S xor D
- `RCC_ROP_7` - not (S a
- `RCC_ROP_8` - S and D
- `RCC_ROP_9` - not (S xi
- `RCC_ROP_A` - D
- `RCC_ROP_B...` [more](#)

### RCC\_ROP\_1

Raster Operation (ROP) opt of the following 16 raster op whenever **Rectangle** Copy F (RCCGPU) is used. The ras

performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#))
- [RCC\\_ROP\\_1](#) - not (S or D)
- [RCC\\_ROP\\_2](#) - (not S) and (not D)
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and not D
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S and D)
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xor D)
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B...](#) [more](#)

## [RCC\\_ROP\\_2](#)

Raster Operation (ROP) options of the following 16 raster operations whenever [Rectangle Copy Function \(RCCGPU\)](#) is used. The raster operation is performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#))
- [RCC\\_ROP\\_1](#) - not (S or D)
- [RCC\\_ROP\\_2](#) - (not S) and (not D)
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and not D
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S and D)
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xor D)
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B...](#) [more](#)

### RCC\_ROP\_3

Raster Operation (ROP) options are defined for the following 16 raster operations whenever **Rectangle Copy Function** (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 (**BLACK**)
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S)
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D
- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D
- RCC\_ROP\_9 - not (S xor D)
- RCC\_ROP\_A - D
- RCC\_ROP\_B... [more](#)

### RCC\_ROP\_4

Raster Operation (ROP) options are defined for the following 16 raster operations whenever **Rectangle Copy Function** (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 (**BLACK**)
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S)
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D
- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_9</a> - not (S x</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_ROP_5</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster op whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S x)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_ROP_6</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster op whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> </ul>

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S and D)</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xor D)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_7</a></p>	<p>Raster Operation (ROP) options are defined for the following 16 raster operations whenever <a href="#">Rectangle Copy Function (RCCGPU)</a> is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLACK</a>)</li> <li>• <a href="#">RCC_ROP_1</a> - not (S or D)</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) and D</li> <li>• <a href="#">RCC_ROP_3</a> - not (S and D)</li> <li>• <a href="#">RCC_ROP_4</a> - S and not D</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S and D)</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xor D)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
----------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_8</a></p>	<p>Raster Operation (ROP) options are defined for the following 16 raster operations whenever <a href="#">Rectangle Copy Function (RCCGPU)</a> is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLACK</a>)</li> </ul>
----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_9</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster opt whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S xi</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B... more</a></li> </ul>
----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><a href="#">RCC_ROP_A</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster opt whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras</p>
----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------

performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#))
- [RCC\\_ROP\\_1](#) - not (S o
- [RCC\\_ROP\\_2](#) - (not S) a
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and n
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

## [RCC\\_ROP\\_B](#)

Raster Operation (ROP) opt of the following 16 raster op whenever [Rectangle](#) Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).

- [RCC\\_ROP\\_0](#) - 0 ([BLAC](#))
- [RCC\\_ROP\\_1](#) - not (S o
- [RCC\\_ROP\\_2](#) - (not S) a
- [RCC\\_ROP\\_3](#) - not (S)
- [RCC\\_ROP\\_4](#) - S and n
- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

## RCC\_ROP\_C

Raster Operation (ROP) options of the following 16 raster operations whenever **Rectangle Copy Function** (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 (**BLACK**)
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S) and D
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D
- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D
- RCC\_ROP\_9 - not (S xor D)
- RCC\_ROP\_A - D
- RCC\_ROP\_B... [more](#)

## RCC\_ROP\_D

Raster Operation (ROP) options of the following 16 raster operations whenever **Rectangle Copy Function** (RCCGPU) is used. The raster operation is performed on the source (S) and destination (D) data, and the result is written to the destination (D).

- RCC\_ROP\_0 - 0 (**BLACK**)
- RCC\_ROP\_1 - not (S or D)
- RCC\_ROP\_2 - (not S) and D
- RCC\_ROP\_3 - not (S) and D
- RCC\_ROP\_4 - S and not (D)
- RCC\_ROP\_5 - not (D)
- RCC\_ROP\_6 - S xor D
- RCC\_ROP\_7 - not (S and D)
- RCC\_ROP\_8 - S and D

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_9</a> - not (S x</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_ROP_E</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster op whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> <li>• <a href="#">RCC_ROP_5</a> - not (D)</li> <li>• <a href="#">RCC_ROP_6</a> - S xor D</li> <li>• <a href="#">RCC_ROP_7</a> - not (S a</li> <li>• <a href="#">RCC_ROP_8</a> - S and D</li> <li>• <a href="#">RCC_ROP_9</a> - not (S x)</li> <li>• <a href="#">RCC_ROP_A</a> - D</li> <li>• <a href="#">RCC_ROP_B...</a> <a href="#">more</a></li> </ul>
<p><a href="#">RCC_ROP_F</a></p>	<p>Raster Operation (ROP) opt of the following 16 raster op whenever <a href="#">Rectangle</a> Copy F (RCCGPU) is used. The ras performed on the source (S) destination (D) data. and the to the destination (D).</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_ROP_0</a> - 0 (<a href="#">BLAC</a></li> <li>• <a href="#">RCC_ROP_1</a> - not (S o</li> <li>• <a href="#">RCC_ROP_2</a> - (not S) a</li> <li>• <a href="#">RCC_ROP_3</a> - not (S)</li> <li>• <a href="#">RCC_ROP_4</a> - S and n</li> </ul>

--

- [RCC\\_ROP\\_5](#) - not (D)
- [RCC\\_ROP\\_6](#) - S xor D
- [RCC\\_ROP\\_7](#) - not (S a
- [RCC\\_ROP\\_8](#) - S and D
- [RCC\\_ROP\\_9](#) - not (S xi
- [RCC\\_ROP\\_A](#) - D
- [RCC\\_ROP\\_B... more](#)

<a href="#">RCC_COPY</a>
--------------------------

Type of [Rectangle](#) Copy Op one of the following rectangl operations and together with source, destination, current ( transparency are evaluated ( and the result written to the (

- RCC\_COPY - Copies th to the destination addre selected ROP.
- [RCC\\_SOLID\\_FILL](#) - Fill rectangle with the curre
- [RCC\\_TRANSPARENT](#) Operation is the same a operation except that th is compared against the set. If the values match, data is not written to the The source... [more](#)

<a href="#">RCC_SOLID_FILL</a>
--------------------------------

Type of [Rectangle](#) Copy Op one of the following rectangl operations and together with source, destination, current ( transparency are evaluated ( and the result written to the (

- RCC\_COPY - Copies th to the destination addre selected ROP.

	<ul style="list-style-type: none"> <li>• <a href="#">RCC_SOLID_FILL</a> - Fill rectangle with the current color.</li> <li>• <a href="#">RCC_TRANSPARENT_COPY</a> - Copy operation is the same as <a href="#">RCC_COPY</a> except that the source and destination are compared against the transparency mask. If the values match, data is not written to the destination. The source... <a href="#">more</a></li> </ul>
<p><a href="#">RCC_TRANSPARENT_COPY</a></p>	<p>Type of <a href="#">Rectangle Copy Operation</a>. One of the following rectangle copy operations and together with source, destination, current color, and transparency are evaluated and the result written to the destination.</p> <ul style="list-style-type: none"> <li>• <a href="#">RCC_COPY</a> - Copies the source to the destination address selected ROP.</li> <li>• <a href="#">RCC_SOLID_FILL</a> - Fill rectangle with the current color.</li> <li>• <a href="#">RCC_TRANSPARENT_COPY</a> - Copy operation is the same as <a href="#">RCC_COPY</a> except that the source and destination are compared against the transparency mask. If the values match, data is not written to the destination. The source... <a href="#">more</a></li> </ul>

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#)

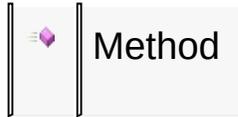
## Legend

Rectangle Copy Operations

---

## Legend

---



---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Rectangle Copy Operations](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Decompressing DEFLATEd data Functions

Decompressing DEFLATEd data

### Functions

	Name	Description
	<a href="#">Decompress</a>	Decompresses the nbytes number of data at SrcAddress and places starting from DestAddress. (Blocking)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Decompressing DEFLATEd data](#)

## Legend

Decompressing DEFLATEd data

---

## Legend



---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Decompressing DEFLATEd data](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette Mode Files

Palette Mode

---

### Files

---

Name	Description
<a href="#">Palette.h</a>	This is file Palette.h.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette Mode Functions

### Palette Mode

#### Functions

	Name	Description
◆	<a href="#"><u>ClearPaletteChangeError</u></a>	Clears the Palette change error status
◆	<a href="#"><u>DisablePalette</u></a>	Disables the Palette mode.
◆	<a href="#"><u>EnablePalette</u></a>	Enables the Palette mode.
◆	<a href="#"><u>GetPaletteChangeError</u></a>	Returns the Palette change error status
◆	<a href="#"><u>IsPaletteEnabled</u></a>	Returns if the Palette mode is enabled or not.
◆	<a href="#"><u>PaletteInit</u></a>	Initializes the color look up table (CLUT).
◆	<a href="#"><u>RequestPaletteChange</u></a>	Loads the palettes from the flash during vertical blanking period if possible, otherwise loads immediately.
◆	<a href="#"><u>SetPalette</u></a>	Programs a block of palette entries starting from startEntry and until startEntry + length from the flash immediately.
◆	<a href="#"><u>SetPaletteBpp</u></a>	Sets the color look up table (CLUT) number of valid entries.
◆	<a href="#"><u>SetPaletteFlash</u></a>	Loads the palettes from the flash

|| immediately.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette Mode Macros

### Palette Mode

#### Macros

Name	Description
<a href="#"><u>RequestEntirePaletteChange</u></a>	Loads all the palette entries from the flash during vertical blanking period if possible, otherwise loads immediately.
<a href="#"><u>SetEntirePalette</u></a>	Programs the whole 256 entry palette with new color values from flash.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

## Palette Mode Structures

Palette Mode

### Structures

Name	Description
<a href="#"><u>PALETTE_FLASH</u></a>	Structure for the palette stored in FLASH memory.
<a href="#"><u>PALETTE_HEADER</u></a>	Structure for the palette header.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette Mode Types

### Palette Mode

#### Types

Name	Description
<a href="#"><u>PALETTE_EXTERNAL</u></a>	Structure for palette stored in EXTERNAL memory space. (example: External SPI or parallel Flash, EDS_EPMP)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Legend

Palette Mode

---

## Legend



---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette.h Functions

Palette.h

### Functions

	Name	Description
◆	<a href="#"><u>ClearPaletteChangeError</u></a>	Clears the Palette change error status
◆	<a href="#"><u>DisablePalette</u></a>	Disables the Palette mode.
◆	<a href="#"><u>EnablePalette</u></a>	Enables the Palette mode.
◆	<a href="#"><u>GetPaletteChangeError</u></a>	Returns the Palette change error status
◆	<a href="#"><u>IsPaletteEnabled</u></a>	Returns if the Palette mode is enabled or not.
◆	<a href="#"><u>PaletteInit</u></a>	Initializes the color look up table (CLUT).
◆	<a href="#"><u>RequestPaletteChange</u></a>	Loads the palettes from the flash during vertical blanking period if possible, otherwise loads immediately.
◆	<a href="#"><u>SetPalette</u></a>	Programs a block of palette entries starting from startEntry and until startEntry + length from the flash immediately.
◆	<a href="#"><u>SetPaletteBpp</u></a>	Sets the color look up table (CLUT) number of valid entries.
◆	<a href="#"><u>SetPaletteFlash</u></a>	Loads the palettes from the flash

|| immediately.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [Palette.h](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette.h Macros

Palette.h

### Macros

Name	Description
<a href="#">RequestEntirePaletteChange</a>	Loads all the palette entries from the flash during vertical blanking period if possible, otherwise loads immediately.
<a href="#">SetEntirePalette</a>	Programs the whole 256 entry palette with new color values from flash.

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [Palette.h](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette.h Structures

Palette.h

---

### Structures

---

Name	Description
<a href="#"><u>PALETTE_FLASH</u></a>	Structure for the palette stored in FLASH memory.
<a href="#"><u>PALETTE_HEADER</u></a>	Structure for the palette header.

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [Palette.h](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Palette.h Types

Palette.h

### Types

Name	Description
<a href="#"><u>PALETTE_EXTERNAL</u></a>	Structure for palette stored in EXTERNAL memory space. (example: External SPI or parallel Flash, EDS_EPMP)

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Palette Mode](#) > [Palette.h](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Set Up Display Interface Macros

## Set Up Display Interface

### Macros

Name	Description
<a href="#">GFX_GCLK_DIVIDER</a>	<p>The following are added and used when using the <code>gfx</code> Module that comes with the PIC24FJ256DA210 Datasheet (<a href="#">PIC24FJ256DA210_DS</a>) (<a href="#">PIC24FJ256DA210_I</a>).</p> <ul style="list-style-type: none"><li>• When using internal memory:<ul style="list-style-type: none"><li>◦ <code>GFX_GCLK_DIVIDER</code> - Set the pixel clock divider.</li><li>◦ <a href="#">GFX_DISPLAY_BUFFER_SIZE</a> - Set the Display Buffer Size (Display Buffer Size calculated by <math>\text{depth}/2</math>).</li></ul></li><li>• When using external memory may be present and/or chip select is used. See Family Reference Manual.</li></ul>
<a href="#">GFX_EPMP_CS1_BASE_ADDRESS</a>	<p>The following are added and used when using the <code>gfx</code> Module that comes with the PIC24FJ256DA210 Datasheet (<a href="#">PIC24FJ256DA210_DS</a>) (<a href="#">PIC24FJ256DA210_I</a>).</p> <ul style="list-style-type: none"><li>• When using internal memory:<ul style="list-style-type: none"><li>◦ <code>GFX_GCLK_DIVIDER</code></li></ul></li></ul>

--

- for the pixel c
- [GFX\\_DISPL/](#)
- Set the Disp
- [GFX\\_DISPL/](#)
- Display Buffe
- calculated by
- depth/2).
- When using exter
- memory may be p
- and/or chip select
- Family Reference

<a href="#">GFX_EPMP_CS1_MEMORY_SIZE</a>
------------------------------------------

- The following are addi
- used when using the c
- Module that comes wi
- (PIC24FJ256DA210 D
- ([PIC24FJ256DA210\\_I](#)
- When using interr
  - GFX\_GCLK\_
  - for the pixel c
  - [GFX\\_DISPL/](#)
  - Set the Disp
  - [GFX\\_DISPL/](#)
  - Display Buffe
  - calculated by
  - depth/2).
  - When using exter
  - memory may be p
  - and/or chip select
  - Family Reference

<a href="#">GFX_EPMP_CS2_BASE_ADDRESS</a>
-------------------------------------------

The following are addi

used when using the c

Module that comes wi

(PIC24FJ256DA210 D

([PIC24FJ256DA210\\_I](#)

- When using internal memory:
  - GFX\_GCLK\_ for the pixel clock
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) - Set the Display Buffer Size (calculated by depth/2).
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) Display Buffer Size calculated by depth/2).
- When using external memory may be present and/or chip select. See Family Reference.

### GFX\_EPMP\_CS2\_MEMORY\_SIZE

The following are additional registers used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet ([PIC24FJ256DA210\\_I](#)

- When using internal memory:
  - GFX\_GCLK\_ for the pixel clock
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) - Set the Display Buffer Size (calculated by depth/2).
  - [GFX\\_DISPLAY\\_BUFFER\\_SIZE](#) Display Buffer Size calculated by depth/2).
- When using external memory may be present and/or chip select. See Family Reference.

## GFX\_DISPLAY\_BUFFER\_LENGTH

The following are additional registers used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet ([PIC24FJ256DA210\\_I](#)).

- When using internal memory:
  - `GFX_GCLK_PIXEL_DEPTH` for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_LENGTH](#) - Set the Display Buffer Length (Display Buffer calculated by  $\text{depth}/2$ ).
- When using external memory may be present and/or chip select. See the PIC24FJ256DA210 Family Reference Manual.

## GFX\_DISPLAY\_BUFFER\_START\_ADDRESS

The following are additional registers used when using the GFX Module that comes with the PIC24FJ256DA210 Datasheet ([PIC24FJ256DA210\\_I](#)).

- When using internal memory:
  - `GFX_GCLK_PIXEL_DEPTH` for the pixel clock.
  - [GFX\\_DISPLAY\\_BUFFER\\_START\\_ADDRESS](#) - Set the Display Buffer Start Address (Display Buffer calculated by  $\text{depth}/2$ ).
- When using external memory may be present and/or chip select. See the PIC24FJ256DA210 Family Reference Manual.

memory may be p  
and/or chip select  
Family Reference

---

[Library API](#) > [Display Device Driver Layer API](#) > [Advanced Display Driver Features](#) > [Microchip Graphics Controller](#) > [Set Up Display Interface](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Image Decoder Demo

## Image Decoders

---

Please refer to the getting started htm document located at <Install Directory>/Microchip/Graphics/Documents/Getting Started/Getting Started - Running the Image Decoders Demo.htm, where <Install Directory> is the root directory of the Microchip Applications Library.

## Module

---

[Image Decoders](#)

## Links

---

[Image Decoders](#)

[Image Decoders](#) > Demo Project > [Image Decoder Demo](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Demo Project

[Image Decoders](#)

---

### Demo Project

---

Name	Description
<a href="#">Image Decoder Demo</a>	This demo demonstrates the decoding of images with JPEG and BMP file formats.

---

### [Image Decoders](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Image Decoders API Functions

### Image Decoders API

#### Functions

	Name	Description
	<a href="#"><u>ImageDecode</u></a>	This function decodes and displays the image on the screen
	<a href="#"><u>ImageDecoderInit</u></a>	This function initializes the global variables to 0 and then initializes the driver. This must be called once before any other function of the library is called
	<a href="#"><u>ImageLoopCallbackRegister</u></a>	This function registers the loop callback function so that the decoder calls this function in every decoding loop. This can be used by the application program to do maintenance activities such as fetching data, updating the display, etc...
	<a href="#"><u>ImageDecodeTask</u></a>	This function completes one small part of the image decode function

### [Image Decoders API](#)

## Image Decoders API Macros

### Image Decoders API

---

#### Macros

---

Name	Description
<a href="#"><u>ImageFullScreenDecode</u></a>	This function decodes and displays the image on the screen in fullscreen mode with center aligned and downscaled if required
<a href="#"><u>ImageAbort</u></a>	This function sets the Image Decoder's Abort flag so that decoding aborts in the next decoding loop.

### [Image Decoders API](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

# Image Decoders API Structures

## Image Decoders API

### Structures

	Name	Description
	<u><a href="#">_BMPDECODER</a></u>	DATA STRUCTURES
	<u><a href="#">_GIFDECODER</a></u>	DATA STRUCTURES
	<u><a href="#">_JPEGDECODER</a></u>	DATA STRUCTURES

## [Image Decoders API](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

[Image Decoders API](#)

---

## Topics

Name	Description
<a href="#">Image Decoder Configuration</a>	The Image Decoder Library can be customized by adding or specifying the compile time options located in the application file named ImageDecoderConfig.h.

## [Image Decoders API](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

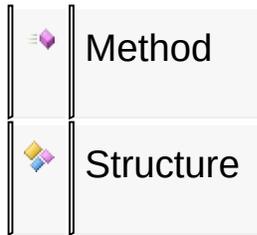
# Legend

Image Decoders API

---

## Legend

---



## [Image Decoders API](#)

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

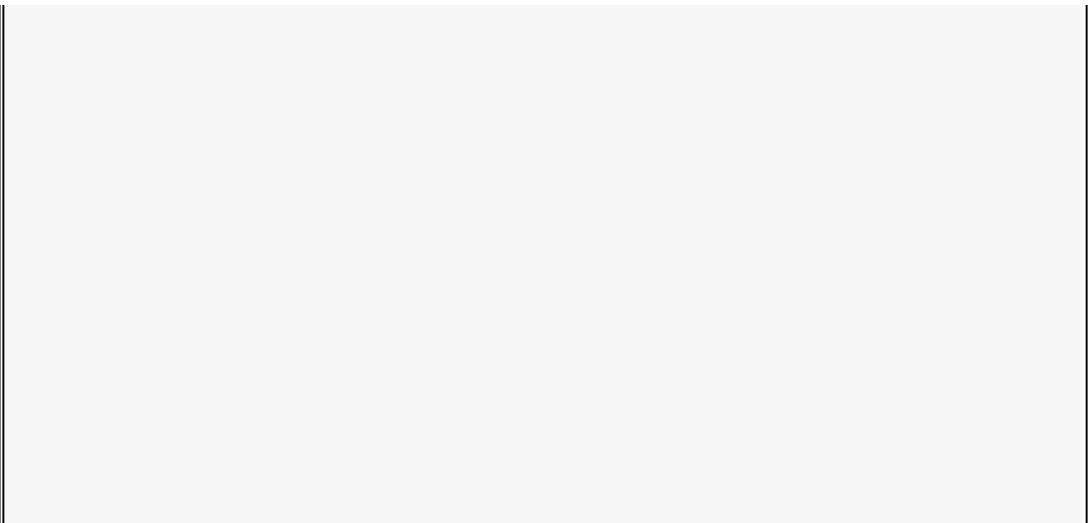
[Contents](#) | [Index](#) | [Home](#)

# Image Decoder Configuration Macros

## Image Decoder Configuration

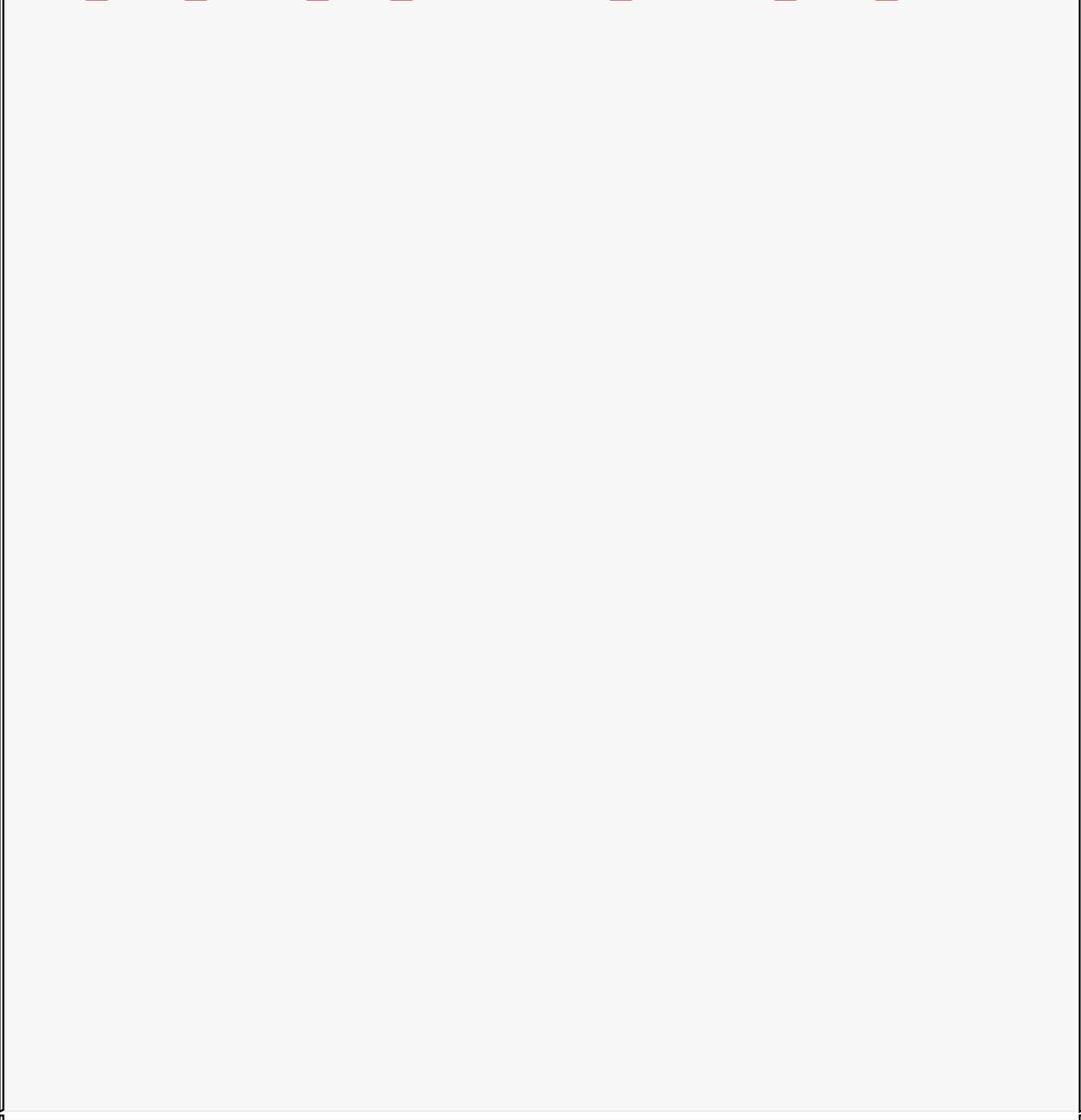
### Macros

Name	Descr
<a href="#"><u>IMG_SUPPORT_BMP</u></a>	Add th Image to ena bitmap decod
<a href="#"><u>IMG_SUPPORT_GIF</u></a>	Add th Image to ena image
<a href="#"><u>IMG_SUPPORT_JPEG</u></a>	Add th Image to ena jpeg in decod
<a href="#"><u>IMG_SUPPORT_IMAGE_DECODER_LOOP_CALLBACK</u></a>	The de up sig time d and us update inform display send/r throug comm chann to rele



power  
decodi  
calling  
functio  
user. T  
do all t  
activiti  
functio  
be ena  
this ma  
Image

IMG\_USE\_ONLY\_565\_GRAPHICS\_DRIVER\_FOR\_OUTPUT



Add th  
Image  
to opti  
graphi  
suppo  
color f  
render  
directly  
buffer.  
bufferi  
or if us  
render  
done,  
above  
and pr  
functio  
pixel v  
callbac  
explai  
section  
out, th  
height  
screer  
provid  
followi

## IMG\_USE\_ONLY\_MDD\_FILE\_SYSTEM\_FOR\_INPUT

Add th  
Image  
to opti  
Memo  
Syster  
Microc  
Syster  
.

---

### Image Decoders API > Image Decoder Configuration

---

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)

## Topics

Miscellaneous Topics

## Topics

Name	Description
<a href="#">Starting a New Project</a>	This outlines the procedure to create a new project that uses the Microchip Graphics Library from scratch.
<a href="#">Changing the default Font</a>	The library comes with the default font (Gentium 18). This font can be changed in two ways.
<a href="#">Advanced Font Features</a>	Fonts used in the library can be configured to use anti-aliasing and extended glyph support.
<a href="#">Using Primitive Rendering Functions in Blocking and Non-Blocking Modes</a>	Basic rendering functions such as <a href="#">Line()</a> , <a href="#">Rectangle()</a> , <a href="#">Circle()</a> etc are referred to as functions in the Graphics 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 directly calls these functions can take advantage of the hardware accelerated primitives. How these functions are used will depend on the " <a href="#">Configuration Setting</a> ".
<a href="#">Using Microchip Graphics Module Color Look Up Table in Applications</a>	Utilizing the Color Look Up Table (CLUT) of the Microchip Graphics Module saves memory for both storage and display buffer. This short instructional manual outlines the procedure to create source code files to use

	the CLUT of the Microchip Graphics Module and enable the Microchip Graphics Library to use the hardware feature.
<a href="#"><u>Converting Images to Use a Common Palette in GIMP</u></a>	This manual describes how to convert an image or a set of images to use a common palette in GIMP.
<a href="#"><u>How to Define Colors in your Applications</u></a>	In most cases, the application will define its own set of colors and not use the default colors that comes with the Graphics Library. This section shows an example on how to do it.
<a href="#"><u>Connecting RGB data bus</u></a>	Display glasses will require 24 bit or 18 bit RGB color data. How do you do the connection when your display controller only puts out 16 bit RGB data bus?
<a href="#"><u>Adding New Device Driver</u></a>	This is a summary of the requirements to add a new device driver.

---

## [Miscellaneous Topics](#)

Microchip Graphics Library Version 3.06.02 - October 15, 2012  
Copyright © 2012 Microchip Technology, Inc. All rights reserved

[Contents](#) | [Index](#) | [Home](#)