SEGGER emWin Graphic Library (emWinGraphics) 1.0

Features

- The component integrates emWin 8051 Graphic Library for PSoC3 and full-featured emWin Graphic Library V5.02 for PSoC 5
- The libraries can be used with the Keil_PK51, GCC, Keil MDK, and Keil RVDS toolchains
- Drivers are available for Graphics LCD Interface and Graphics LCD Controller components

General Description

emWin is an embedded graphic library and graphical user interface (GUI) designed to provide an efficient, processor- and LCD controller-independent GUI for any application that operates with a graphical display. It is compatible with single-task and multitask environments. Developed by SEGGER Microcontroller, emWin is extremely popular in the embedded industry. Cypress has licensed the emWin library from SEGGER and offers a full-featured graphic library free to customers.

When to Use an emWinGraphics

This component provides access to and the functionality of the SEGGER emWin graphic library. The emWinGraphics component is a complete, easy to use, software package for the PSoC 3 and PSoC 5 target hardware. The package consists of the emWin GUI provided in library form, the required header files, and the source files specific to the PSoC implementation.

The emWin Graphics Library component allows you fast and efficient GUI development for any application that operates with a graphical LCD.

Getting Started

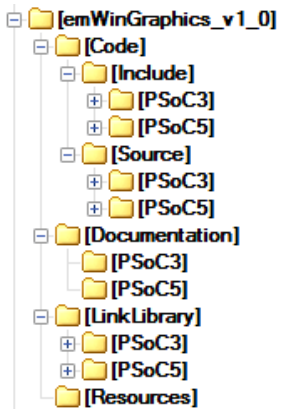
Setup

The emWinGraphics component is supplied as a zip file. You can get the latest component version from the following location:

http://www.cypress.com/go/comp_emWin

Extract it to any folder, preserving the directory structure of the zip file. Make sure the files are not read-only after extracting. The directory structure for the unzipped installation will be similar to that shown in Figure 1.

Figure 1. Directory Structure



Each of the major directories except Resources is divided into PSoC 3 and PSoC 5 subdirectories. This is because the implementations are different for the two part families. PSoC 3 supports the limited library version (emWin 8051), while PSoC 5 supports the full-featured version of the standard emWin library. The detailed functionality provided by each library version is documented in the SEGGER emWin documentation that located in the *Documentation\PSoC3* and *Documentation\PSoC5* directories. For details about the different areas of functionality for PSoC 3 and PSoC 5, see the Functional Description section of this document.

The following table shows the contents of all major emWinGraphics directories.

Directory	Contents
Code\Include	The GUI header files.
Code\Source	The source files that you need to add to your project
Documentation	The emWin User Guides
LinkLibrary	The emWin libraries for each of the supported toolchains
Resources	emWin sample files and windows programs from SEGGER to work with bitmaps

Using the emWinGraphics Library

To use the emWinGraphics library, follow these steps. All steps are explained further in subsequent sections.

1. Integrate the emWinGraphics Library into your PSoC Creator project.

2. Add a component to communicate with a graphics LCD panel to your project. The emWinGraphics library is a software library. The communication to the panel is done using the appropriate hardware component for the panel. The Cypress component library includes two components that interface to an LCD panel: Graphics LCD Interface or Graphics LCD Controller.
3. Add a touchscreen component to your project (optional). A touchscreen component such as the Resistive Touch component can be used with the emWinGraphics library.
4. Compile, link and test using the provided sample code. The emWinGraphics library comes with sample code for both single- and multitask environments. These examples can be used to learn the usage methodology of the library. These examples are found in the Resources directory.
5. For multitasking applications adapt the library to your operation system (OS). If multiple tasks will have access to the display simultaneously, then OS interface routines must be defined. For details and sample applications, please refer to Chapter 13: "Execution Model: Single Task/Multitask" of the *emWin User Manual*.
6. Write your own application using emWin. At this point you should be ready to start programming using emWin. Consult the reference chapters in the *emWin User Manual* for the details of each capability the library provides.

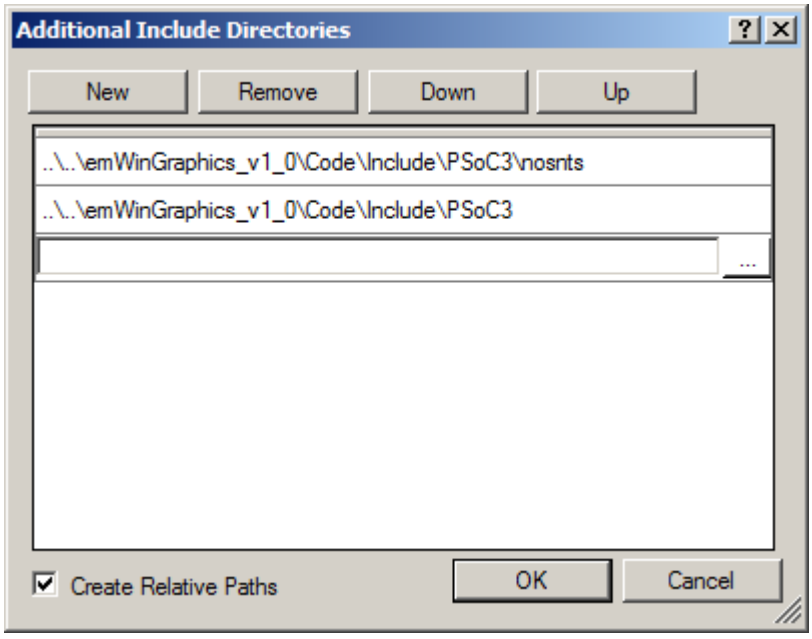
Integrating the emWinGraphics Library into PSoC Creator

To integrate the emWinGraphics Library into your project, follow these steps. This example uses the Keil_PK51 toolchain. Other toolchains are supported in a similar fashion.

1. First, decide which library you need. The decision is based on whether you are using an RTOS (os), a touchscreen (ts), or both. The emWin libraries are named according to the supported options. In this example, the library name is emWinnosnts.lib, which is a library without OS (nos) and touchscreen (nts) support.
2. Add the necessary include files. The emWin include directory contains include files that are general to the entire application and specific include files in subdirectories based on the options chosen. The **Additional Include Directories** field of the Compiler options should contain an entry that addresses the path to the general and library-specific include files.

   From the menu **Project > Build Settings > Compiler > General > Additional Include Directories**, specify the include path, as shown in Figure 2.

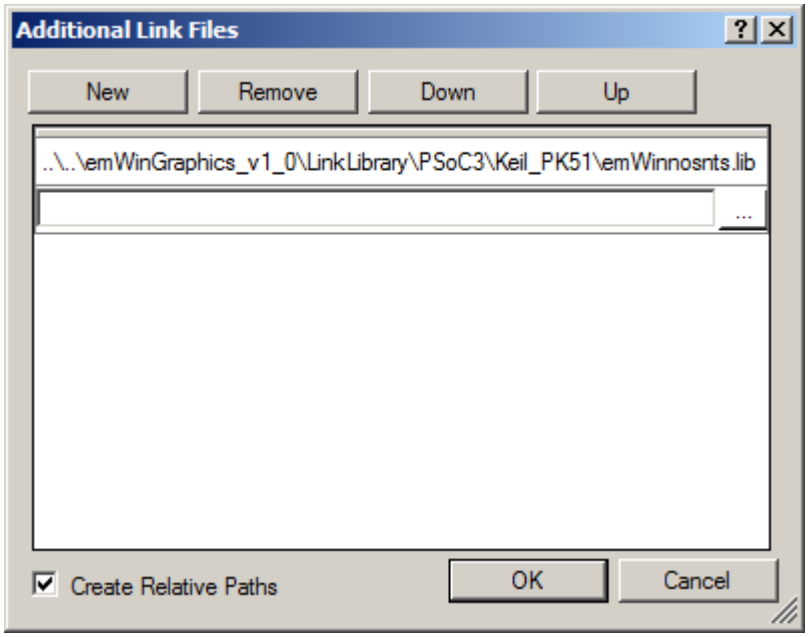
Figure 2. Additional Include Directories Dialog



3. Add the link library file. The emWin GUI can be used with the Keil_PK51, GNU CC, KEIL MDK, or KEIL RVDS toolchain, so select the library according to the toolchain you choose.

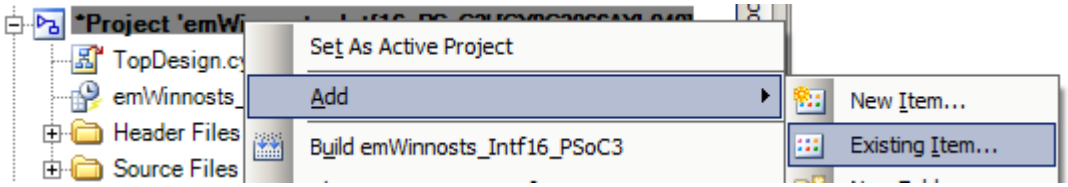
   From the menu **Project > Build Settings > Linker > General > Additional Link Files**, select the library file shown in Figure 3.

Figure 3. Additional Link Files Dialog

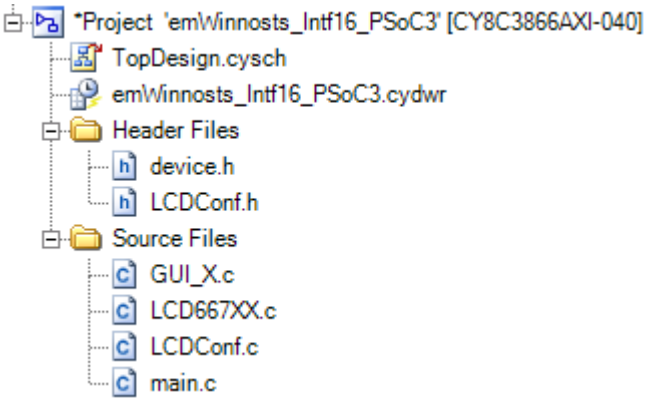


Note The process of adding the library to the linker is different for the GCC toolchain. In this case, the directory containing the library is added to the linker with the “Additional Library Directories” setting and the library is specified with the “Additional Libraries” setting. The library name used for this setting must have the “lib” prefix and “.a” suffix removed from the name.

4. Add the required source files to the project for the LCD component chosen. Select either the Interface component or the Control component. Based on that selection, the files are available at either the *Source\PSoC3\Graphics LCD Interface Parallel* directory or the *Source\PSoC3\Graphics LCD Controller* directory. These files are typically edited for the specific project, so it is recommended that the files needed be copied into the root folder of your project and then added to your project. The files are added to the project using the menu selection **Project > Add > Existing Item**. If the include files are not placed in the root of the project directory, then the path to the include files must also be added to the build settings.



The project shown here has added all the files from the *Source\PSoC3\Graphics LCD Interface Parallel* directory except for *GUI_X_Touch.c*. This example uses neither OS nor touchscreen, so the touchscreen-related file is excluded.

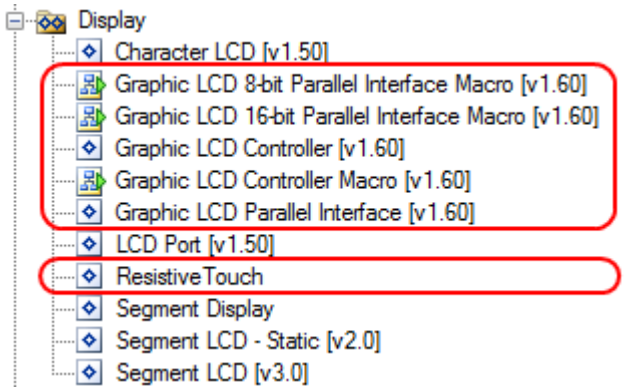


PSoC Creator Components to Use with emWin

The emWinGraphics library is a software library. It does not include any hardware to drive an LCD panel. A graphics LCD component (Graphics LCD Interface or Graphics LCD Controller) is required for you to build a design using the emWinGraphics library. Both components provide support for multiple graphic panels, but they drive different types of panels. Additionally, an optional touchscreen component can be used with emWin to implement a touch user interface. The communication between these components is implemented entirely in software.

A single LCD graphics design contains exactly one instance of a Graphics LCD component, and optionally, one instance of a touchscreen component. All these components are located in the Display folder of the Component Catalog as highlighted in Figure 4.

Figure 4. Display Folder



Graphics LCD Interface (GraphicLCDIntf) Component

The GraphicLCDIntf component is used to interface to the panels that have a graphic LCD controller and driver device integrated into the LCD panel. This type of panel also includes a frame buffer that is managed by the LCD controller integrated into the panel. The component performs read and write transactions to this controller.

The following steps are required in order to create a GraphicLCDIntf application

1. Place the Graphic LCD Parallel Interface Macro onto the schematic. Select an 8-bit or 16-bit interface, based on the LCD controller used.
2. Configure the GraphicLCDIntf component to satisfy the requirements for write and read transactions to the LCD controller. For details about the configuration refer to the GraphicLCDIntf datasheet.
3. Make sure that all emWin files specific for the GraphicLCDIntf component are included in your project. The files are located in the subdirectory Graphics LCD Interface Parallel which is located in the *Code\Source\PSoC3 or Code\Source\PSoC5* directory depending on the PSoC device.
4. Modify the required source files according to your LCD display hardware. Most of the source files can be used unchanged from the emWin distribution, but there are some configuration files that need to be changed for the target system to work properly. For more details about emWin configuration, see the chapter “Configuration” in the *emWin User Manual*.

emWin files Required for a Graphic LCD Interface Application

The following files from the *Code\Source\PSoC3\Graphics LCD Interface Parallel* directory are required to build a PSoC 3 application using the GraphicLCDIntf component:

- *GUI_X.c*: You must edit this file to provide the timing routines (the emWin library requires the knowledge of time) and the debugging routines (optionally).
- *GUI_X_Touch.c* (optional): Provides the hardware routines to implement the resistive touch interface. You do not need to edit this file.
- *LCD667XX.c*: Display driver. You do not need to edit this file.
- *LCDConf.h*: You must edit this file to provide general configuration options required for compiling the display driver. For details about the available configuration options, see the "Display drivers" chapter in the *emWin User Manual*.
- *LCDConf.c*: You must edit this file to provide the initialization routine of the specific display controller used.

The following files from the *Code\Source\PSoC5\Graphics LCD Interface Parallel* folder are required to build a PSoC 5 application using the GraphicLCDIntf component:

- *GUI_X.c*: You must edit this file to provide the timing routines (the emWin library requires the knowledge of time) and the debugging routines (optionally). You should include it in the project if your application does not use an OS. If you are using an OS, then you should use an OS specific version of this file instead.
- *GUI_X_embOS.c*: This file provides the GUI kernel interface routines adapted for the embOS RTOS. Include this file in your project if your application uses the embOS real-time kernel. You do not usually need to edit this file.
- *GUI_X_uCOS.c*: This file provides the GUI kernel interface routines adapted for the uC/OS RTOS. This file should be included in your project if your application uses the uC/OS real-time kernel. You do not usually need to edit this file.
- *GUI_X_Touch.c* (optional): Provides the hardware routines to implement the resistive touch interface. You do not need to edit this file.
- *GUIConf.c*: Allocates memory for emWin operations. You can adjust this file for the specific chip and application. The default setting is for 8,192 bytes of SRAM for emWin to use.
- *GUIDRV_CompactColor_16.c*: Display driver. You do not need to edit this file.
- *LCDConf.h*: Indicates that you are using the Compact Color Driver. You do not need to edit this file.
- *LCDConf_CompactColor_16.h*: You must edit this file to provide general configuration options required to compile the display driver. For details about the available configuration options, see the "Display drivers" chapter in the *emWin User Manual*.
- *LCDConf.c*:You must edit this file to provide the initialization routine of the specific display controller you are using.

Graphics LCD Controller (GraphicLCDCtrl) Component

The GraphicLCDCtrl component provides the interface to an LCD panel that has an LCD driver, but not an LCD controller. This type of panel does not include a frame buffer. The frame buffer must be provided externally. This component directly drives the control signals and manages the frame buffer in an external SRAM.

The following steps are required to create a GraphicLCDCtrl application

1. Place the Graphic LCD Controller Macro onto the schematic.
2. Configure the GraphicLCDCtrl component to satisfy the screen refresh requirements and data transaction timing parameters of your LCD panel. For details about the configuration, refer to the GraphicLCDCtrl datasheet.
3. Make sure that all emWin files specific to the GraphicLCDCtrl component are included in the project. The files are located in the subdirectory Graphics LCD Controller which is located in the *Code\Source\PSoC3 or Code\Source\PSoC5* directory depending on the PSoC device you are planning to use.
4. Change required source files in accordance with your LCD display hardware. Most of the source files can be used unchanged from the emWin distribution, but there are some configuration files that you will need to edit for the target system to work properly. For more details about emWin configuration, see the “Configuration” chapter of the *emWin User Manual*.

emWin files Required for a Graphic LCD Controller Application

The following files from the *Code\Source\PSoC3\Graphics LCD Controller* directory are required to build a PSoC 3 application using the GraphicLCDCtrl component:

- *GUI_X.c*: You must edit this file to provide the timing routines (the emWin library requires the knowledge of time) and the debugging routines (optionally).
- *GUI_X_Touch.c* (optional): Provides the hardware routines to implement the resistive touch interface. You do not need to edit this file.
- *GUIDRV_Control.c*: Display driver. You do not need to edit this file.
- *LCDConf.h*: You must edit this file to provide general configuration options required to compile the display driver. For details about the available configuration options, see the “Display drivers” chapter of the *emWin User Manual*.
- *LCDConf.c*: You must edit this file to provide the initialization routine of the specific LCD panel you are using.

The following files from the *Code\Source\PSoC5\Graphics LCD Controller* directory are required to build a PSoC 5 application using GraphicLCDCtrl component:

- *GUI_X.c*: You must edit this file to provide the timing routines (the emWin library requires the knowledge of time) and the debugging routines (optionally). You should include it in the project if you application does not use an OS. If you are using an OS, then you should use an OS specific version of this file instead.
- *GUI_X_embOS.c*: This file provides the GUI kernel interface routines adapted for the embOS RTOS. Include this file in your project if your application uses the embOS real time kernel. You do not usually need to edit this file.
- *GUI_X_uCOS.c*: This file provides the GUI kernel interface routines adapted for the uC/OS RTOS. Include this file in your project if your application uses the uC/OS real time kernel. You do not usually need to edit this file.
- *GUI_X_Touch.c* (optional): Provides the hardware routines to implement the resistive touch interface. You do not need to edit this file.
- *GUIConf.c*: Allocates memory for emWin operations. You can adjust this file for the specific chip and application. The default setting is for 8,192 bytes of SRAM for emWin to use.
- *GUIDRV_Control.c*: Display driver. You do not need to edit this file.
- *LCDConf.h*: Indicates the driver to be used and contains some general configuration options including display orientation and color depth. You can edit this file if you need to.
- *LCDConf.c*: You must edit this file to provide the initialization routine of the specific LCD panel you are using.

Setting Up the Analog Touchscreen

The most common way to connect a touchscreen is the 4-pin analog interface. A driver is supplied with emWin for this. The emWin touchscreen driver manages analog input (from an A/D converter), debouncing, and calibration of the touchscreen.

The touchscreen driver continuously monitors and updates the touch panel using the function GUI_TOUCH_Exec(). This function calls the appropriate generic touchscreen API routines when it recognizes that an action has been performed or something has changed.

Add touchscreen functionality to your project using the following steps:

1. Make sure that the project settings are made to use the emWin library with touchscreen support. For details about how to select emWin libraries, see the Integrating the emWinGraphics Library into PSoC Creator section of this document.
2. Add the *GUI_X_Touch.c* file to your project.
3. Place the Resistive Touch component onto the schematic. The component is located in the Display folder of the Component Catalog.
4. Implement regular calls to GUI_TOUCH_Exec(). Your application should call it about 100 times/second. If you are using a real-time operating system, the easiest way to make

sure this function is called is to create a separate task. When not using a multitasking system, you can use an interrupt service routine.

5. Determine the minimum and maximum values returned from the A/D converter for your panel and change the parameters of the function GUI_TOUCH_Calibrate() with these values. emWin needs these values to convert the measurement result to the touch position in pixels. For details about touchscreen configuration, see Chapter 19.4.2 “The analog touch screen driver” in the *emWin User Guide*. The files *TOUCH_Sample.c* and *TOUCH_Calibrate.c* referenced in the User Guide can be found in the *Resources\Examples* directory.

EmWin Sample Files

The emWinGraphics component is shipped with many emWin sample files available for PSoC 3 and PSoC 5. They are placed in the PSoC 3 or PSoC 5 subdirectories of the *Resources\Samples* directory. If you are not familiar with emWin, Cypress recommends that you compile, link, and test sample programs before creating your own applications.

To get a sample file running:

1. Add one of the demo sample files to your project.
2. Edit the main source file (*main.c*) to include *GUI.h*. Enable interrupts and then call MainTask() as shown in the code that follows. All of the emWin examples use MainTask() as the entry point.

```
#include <device.h>
#include "GUI.h"

extern void MainTask(void);

void main()
{
    CyGlobalIntEnable;
    MainTask();

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

3. Build and run the project.

Available Libraries

The emWin GUI can be used with the Keil_PK51, GNU CC, KEIL MDK, or KEIL RVDS toolchain. For every toolchain, except Keil_PK51, the libraries are available for two options – OS (os) support and touchscreen (ts) support. The OS support provides the necessary hooks during

initialization of the Graphics library and hooks for locking and unlocking resources within the library by the OS. The options supported by the specific library names are shown here:

Option	OS support	TS support
nosnts	NO	NO
nosts	NO	YES
osnts	YES	NO
osts	YES	YES

emWin Libraries for the Keil_PK51 Toolchain

The libraries are located in the *LinkLibrary\PSoC3\Keil_PK51* directory and are named as follows:

emWin<options>.lib

The options available are nosnts and nosts.

Example:

libemWinnosts.a is the library without OS support and with touchscreen support.

emWin Libraries for the GCC Toolchain

The libraries are located in *LinkLibrary\PSoC5\GCC* directory and are named as follows:

libemWin<options>.a

The options available are nosnts, nosts, osnts and osts.

Example:

libemWinosnts.a is the library with OS support and without touchscreen support.

Note that when using the GCC toolchain, the ‘lib’ prefix and ‘.a’ file extension of the library name must be omitted in the **Additional Libraries** field of the **Linker** options.

emWin Libraries for the Keil MDK Toolchain

The libraries are located in *LinkLibrary\PSoC5\ARM_MDK* directory and are named as follows:

emWin<options>.lib

The options available are nosnts, nosts, osnts and osts.

Example:

emWinosts.lib is the library with OS and touchscreen support.

emWin Libraries for the ARM RVDS Toolchain

The libraries are located in *LinkLibrary\PSoC5\ARM_RVDS* directory and are named as follows:

emWin<options>.lib

The options available are nosnts, nosts, osnts and osts.

Example:

emWinnosnts.lib is the library without OS and touchscreen support.

Placement

The implementation is entirely software.

Performance and Resource Usage

The actual performance and resource usage depends on many factors (CPU, compiler, memory model, optimization, configuration, interface to LCD controller, and so on). This section contains performance values of image drawing operations and information about memory resource usage for a typical configuration.

Performance

Product	CPU Speed (MHz)	Graphic LCD Component	bpp	Filling[1]	Small Fonts[2]	Large Fonts[2]	Bitmap 1 bpp	Bitmap 8 bpp	Bitmap 16 bpp
PSoC 3	24	Graphics LCD Interface (16-bit)	16	112K	24K	30K	36K	14K	NA
PSoC 3	60	Graphics LCD Interface (16-bit)	16	279K	56K	70K	82K	33K	NA
PSoC 3	24	Graphics LCD Interface (8-bit)	16	63K	20K	24K	28K	13K	NA

[1] Based on the performance value listed in the table and the resolution of your display, the performance values in frames per second (fps) can be calculated as follows:

fps = pixels per second / (LCD_XSIZE × LCD_YSIZE)

[2] Based on these values, an approximate number of characters per second can be calculated as follows:

Characters per second = pixels per second / (character width × character height)

For details about character size, se the “Fonts” section of the *emWin User Manual*.

Product	CPU Speed (MHz)	Graphic LCD Component	bpp	Filling[1]	Small Fonts[2]	Large Fonts[2]	Bitmap 1 bpp	Bitmap 8 bpp	Bitmap 16 bpp
PSoC 3	60	Graphics LCD Interface (8-bit)	16	156K	46K	56K	65K	30K	NA
PSoC 3	24	Graphics LCD Controller	16	34K	21K	23K	26K	12K	NA
PSoC 3	60	Graphics LCD Controller	16	75K	48K	54K	61K	29K	NA
PSoC 5	24	Graphics LCD Interface (16-bit)	16	662K	156K	177K	195K	150K	589K
PSoC 5	60	Graphics LCD Interface (16-bit)	16	1.64M	315K	357K	400K	306K	1.40M
PSoC 5	24	Graphics LCD Interface (8-bit)	16	300K	124K	136K	149K	117K	178K
PSoC 5	60	Graphics LCD Interface (8-bit)	16	570K	231K	252K	277K	219K	335K
PSoC 5	24	Graphics LCD Controller	16	147K	131K	130K	135K	114K	152K
PSoC 5	60	Graphics LCD Controller	16	224K	200K	192K	207K	184K	229K

Memory Requirements

The memory usage for the emWin library varies significantly, depending primarily on the application and features used. The following table provides the overall size (everything including code, fonts, boot, images, and so on) of some sample applications to give you a basic idea about the memory requirements. To do more precise calculations, see the SEGGER documentation for the emWin library.

Sample	Keil 8051		GCC-4.4.1		Description
	Flash	SRAM	Flash	SRAM	
Hello World application	19.4 KB	3.6 KB	24.5 KB	2.9 KB	This sample project uses only the basic functions for displaying text.
2D Graphics application	33 KB	3.7 KB	34 KB	2.9 KB	This sample displays text using different fonts and performs some drawing operations of a 2D Graphics Library.
Window application	N/A	N/A	84 KB	11 KB	This project is a “typical” application using the window manager and different widgets.

Application Programming Interface

The emWinGraphics component does not include any API functions that are not present in the standard emWin library. See to the SEGGER emWin documentation for a detailed description of available emWin routines.

Functional Description

The detailed functionality is documented in the SEGGER emWin documentation. This section identifies each of the major categories of SEGGER functionality with respect to PSoC 3 and PSoC 5.

SEGGER emWin Library feature	PSoC 3	PSoC 5
2-D graphics library	Full support	Full support
Displaying bitmap files	No support for formats other than compiled bitmap format. No support for 16-bit bitmaps. Eight and smaller bit-count bitmaps are fully supported.	Support for all compiled bitmap formats Other formats (JPEG, GIF, PNG) may be supportable, but may be impractical because of RAM requirements
Fonts	Support for proportional fonts only	Support for all font types
Bitmap converter	Compiled windows application shipped as a binary.	
Color support	Full support	Full support
Memory devices	No support	Full support
Multitask (RTOS)	No support	Full support
Window manager	No support	Full support
Window objects (widgets)	No support	Full support
Virtual screens/Virtual pages	Full support	Full support
Multilayer/Multidisplay	No support	No support
Pointer input devices	Full support of touchscreens only	Full support of touchscreens only
Sprites and cursors	No support	Full support
Antialiasing	No support	Full support
Foreign language	No support	No support
VNC	No support	No support
Drivers	Compact color driver and Cypress custom driver	

Component Changes

Version 1.0 is the first release of the emWinGraphics library.

© Cypress Semiconductor Corporation, 2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.