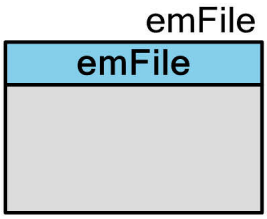
File System Library (emFile)
1.0

Features



- Up to four Secure Digital (SD) cards in SPI mode
- FAT12/16 or FAT32 format
- Optional integration with an Operating System (OS)
- Optional Long File Name (LFN) handling

General Description

The emFile component provides an interface to SD cards formatted with a FAT file system. The SD card specification includes multiple hardware interface options for communication with an SD card. This component uses the SPI interface method for communication. Up to four independent SPI interfaces can be used for communication with one SD card each. Both FAT12/16 and FAT32 file system formats are supported. This component provides the physical interface to the SD card and works with the emFile library licensed from SEGGER Microcontroller to provide a library of functions to manipulate a FAT file system.

When to Use emFile

Use the emFile component to access SD cards formatted using the FAT 12/16 or FAT32 file system formats.

Getting Started

Installing the emFile Library

The emFile file system implementation consists of two parts. The first part is the emFile component which is shipped with PSoC Creator. The second part is the emFile file system library licensed from SEGGER Microcontroller. The library is delivered as a zip file that can be downloaded from the Cypress website emFile component page.

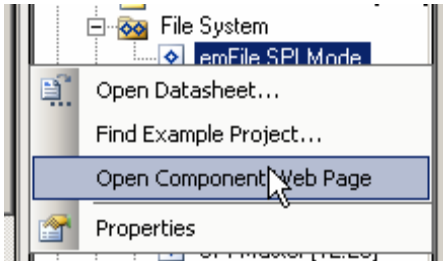
To install the emFile library:

1. Open PSoC Creator and go to the Component Catalog window. Search for the emFile component. It is located under the **Cypress** tab at **Communications > File System > emFile SPI Mode**.

2. Right-click on the emFile component.

   The menu shown in Figure 1 will appear.

Figure 1. emFile Open Component Web Page



3. Click on **Open Component Web Page**.

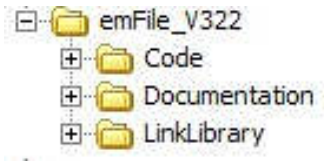
   It will direct you to the component landing page where the zip file with latest file system libraries is located.

4. Download the zip file and extract it to a folder of your choice, preserving the directory structure of the zip file. Make sure the files are not read-only after extracting.

emFile Library Directory Organization

The directory structure of the unzipped emFile library will be similar to that shown in Figure 2.

Figure 2. emFile Directory Organization



The Code directory contains the portion of the library that is provided in source code format. This directory has two subdirectories: Include and Source. The Include directory provides header files for the library. The Source directory provides the executable portion of the library that is delivered in source format.

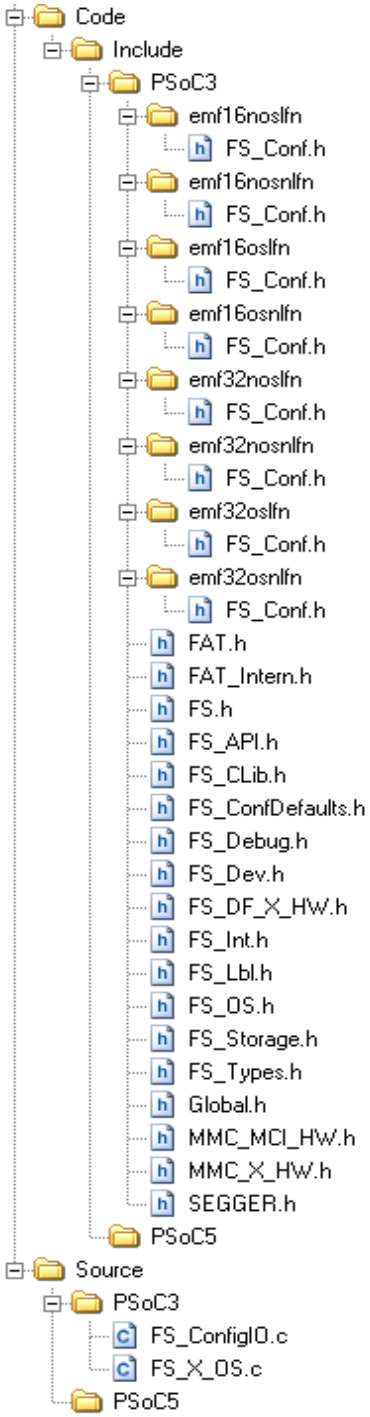
The Include section is divided into header files that are always applicable at the top level of the directory and header files in subdirectories that are used depending on the options chosen for the particular LinkLibrary. The names of the subdirectories are specified as emf<options>. All of the options are described in Table 1.

Table 1. Options

Option	Description
16	FAT 12/16 format
32	FAT 32 format
os	Operating System support
nos	No Operating System support
lfn	Long File Name support
nlfn	No Long File Name support

Figure 3 shows files in the Code directory. This figure shows the directory organization of files for PSoC 3. PSoC 5 has a similar file organization.

Figure 3. emFile "Code" Directory File Listing

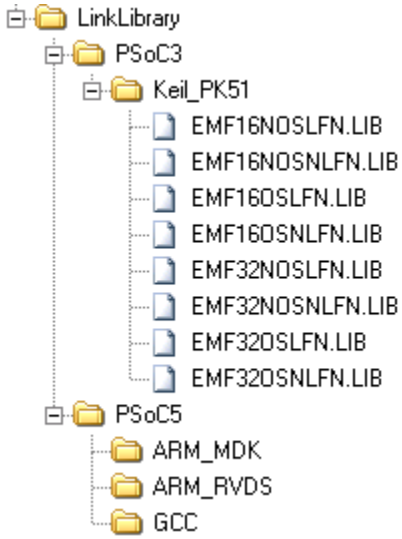


The Documentation section contains the User Guide for the SEGGER Microcontroller emFile file system library.

The LinkLibrary section is divided into PSoC 3 and PSoC 5 directories. Each directory contains a subdirectory for each of the supported tool chains. Within those directories are object libraries that provide the implementation of the file system for the specific options chosen. The library is

added to a project using the Build Settings options described later in this datasheet. The expanded PSoC 3/Keil PK51 section contains the object libraries shown in Figure 4.

Figure 4. LinkLibrary Structure



The naming convention for the LinkLibrary is: <prefix>emf<options>.<extension>. The options are the same options used for the include file directory names. The prefix and extension are specific to the tool chain used. See Table 2.

Table 2. Naming Convention

Toolchain	Prefix	Extension
Keil PK51		lib
GCC	lib	a
ARM_MDK		lib
ARM_RVDS		lib

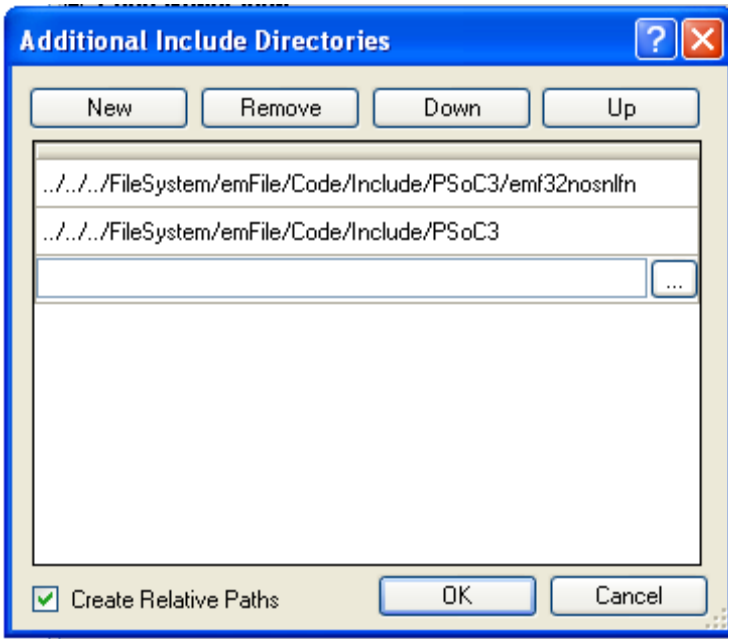
Creating an emFile Project for a PSoC 3 Application

To use the emFile library in a PSoC 3 project:

1. Decide which library you need. This decision is based on whether you need FAT12/16 or FAT32, whether your application uses an OS, and whether you need long file name support. This example uses emf32nosnlfn.lib (FAT32, no OS, and no long file name support).
2. Select the necessary include file directory. Go to **Project > BuildSettings > Compiler > General**. Click the "**…**" button in the **Additional Include Directories** property field. The **Additional Include Directories** dialog will display. Click the **New** button and select an

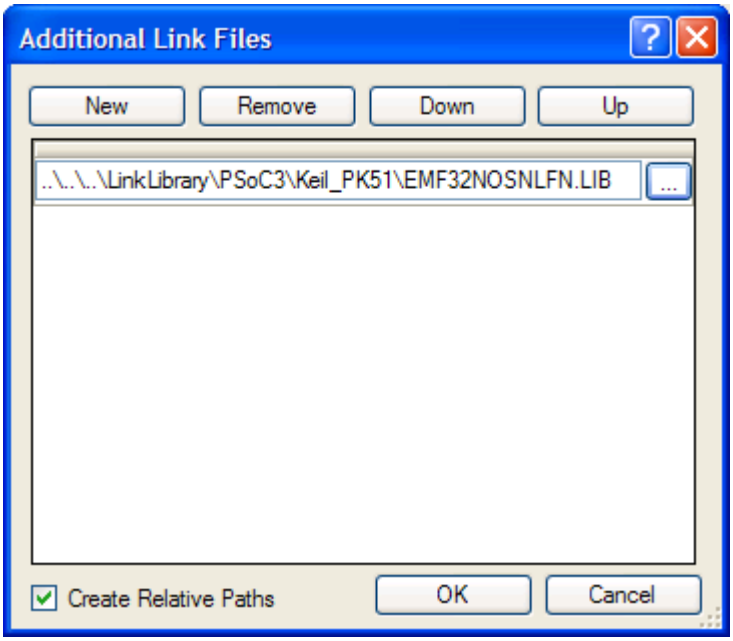
include directory for PSoC 3 and an include directory for the specific options you want. When complete, the dialog should appear as shown in Figure 5.

Figure 5. Adding include Directories



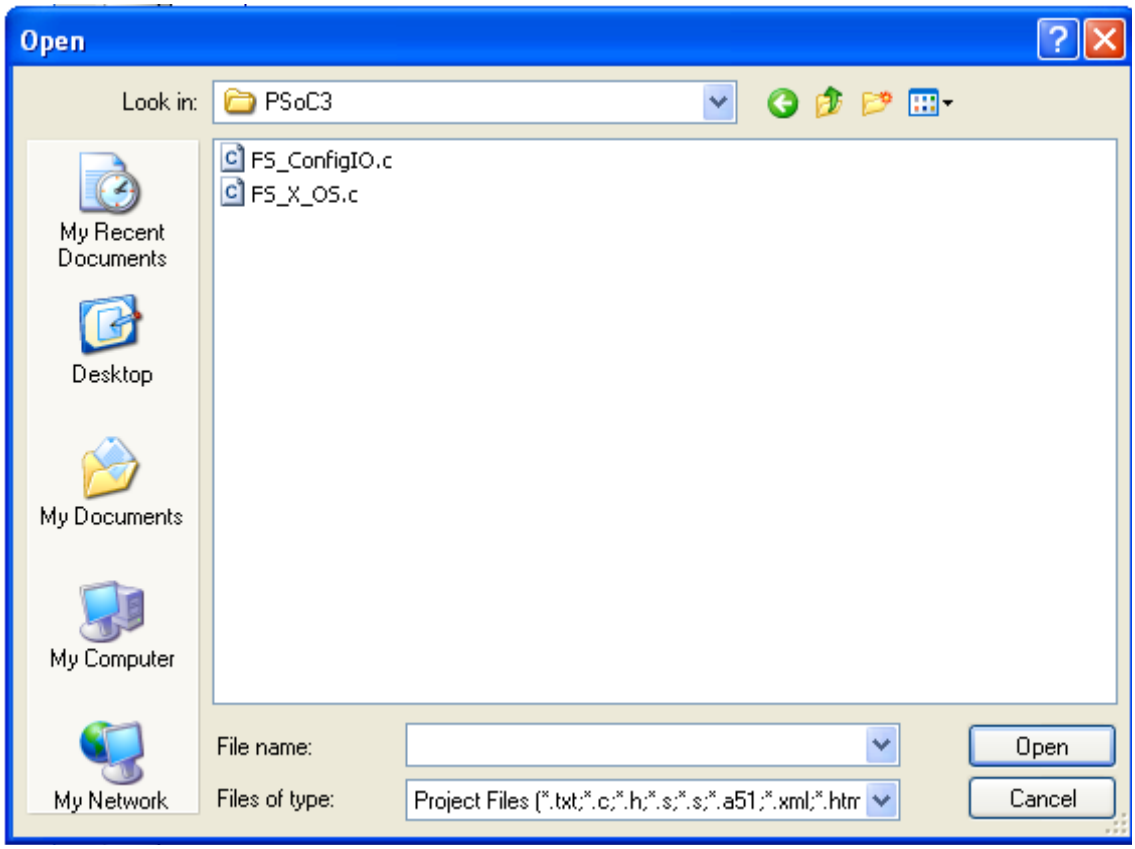
3. Select the Link library file you need. Go to **Project > BuildSettings > Linker > General**. Click the "**…**" button in the **Additional Link Files** property field. The **Additional Link Files** dialog will display. Click the **New** button and select the library file based on the specific options you want. When complete, the dialog should appear as shown in Figure 6.

Figure 6. Adding a Link Library



4. Add the source file *FS_ConfigIO.c* into the project. If you are using an OS, also add *FS_X_OS.c*. These files can be added directly into your project from the Code/Source/PSoC3 directory or copied first to your project directory and added from there. The files will usually be edited, so you need to decide whether to change the original files or a copy that is specific to the project. The files are added to the project using ***Project > Existing Item***. The dialog shown in Figure 7 will open to allow you to select the files.

Figure 7. Adding Source Files



The emFile library is now included in your project.

Creating an emFile Project for a PSoC 5 Application

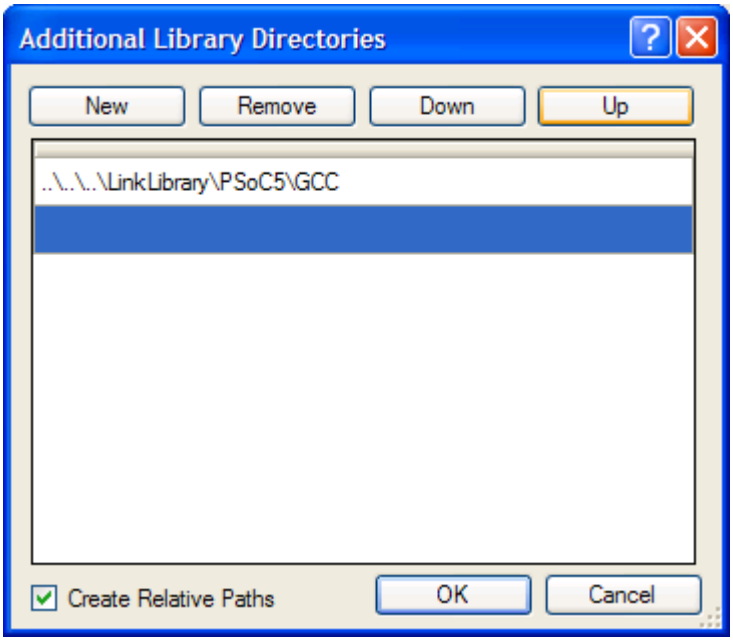
The steps to create an emFile project for PSoC 5 are the same as for PSoC 3 except when using the GCC toolchain. When using the GCC toolchain, instead of adding the link library as a file, you must specify the directory where the library file is located.

Go to **BuildSettings > Linker > General > Additional Library Directories**. Click the "**…**" button in the **Additional Library Directories** property field.

The **Additional Library Directories** dialog will display.

Click the **New** button and select the library directory for the GCC library. When complete, the dialog should appear as shown in Figure 8.

Figure 8. Adding Library Directory



Specify the library in the library directory.

a. Go to **BuildSettings > Linker > General > Additional Libraries**.

b. Type in the library name with the “lib” prefix and “.a” suffix excluded from the name.

   Assuming that you are using the libemf32nosnlfn.a library file, the library name should be emf32nosnlfn.

Input/Output Connections

This section describes the various input and output connections for the emFile component. An asterisk (*) in the list of I/Os means that the I/O may be excluded from the component under the conditions listed in the description of that I/O.

There are no visible connections for the component on the symbol. All connections shown are for pin connections that are included internal to the component. These pins will all appear in the Design-Wide Resources Pin Editor. They must be assigned to the appropriate physical pins using the Pin Editor. For each of the connections there are four pins named with indexes 0 to 3. These represent the four independent SD cards that the component can support. One to four of these pins will be present in the design, depending on the number of SD cards selected. The following is a description of these pins.

SPI0_WP – SPI3_WP*

Optional input pins based on **SD Card [0-3] Write Protection** options, which configure the write protect for SD cards [0-3]. These pins are present if these options are checked. The default state for these pins is not present, which indicates that the SD cards are not write protected.

mosi0 – mosi3

Master Out Slave In pin of the SPI Master. This pin should be connected to the DI/CMD pin of the SD card.

miso0 – miso3

Master In slave Out pin of the SPI Master. This pin should be connected to the DO/DAT0 pin of the SD card.

sclk0 – sclk3

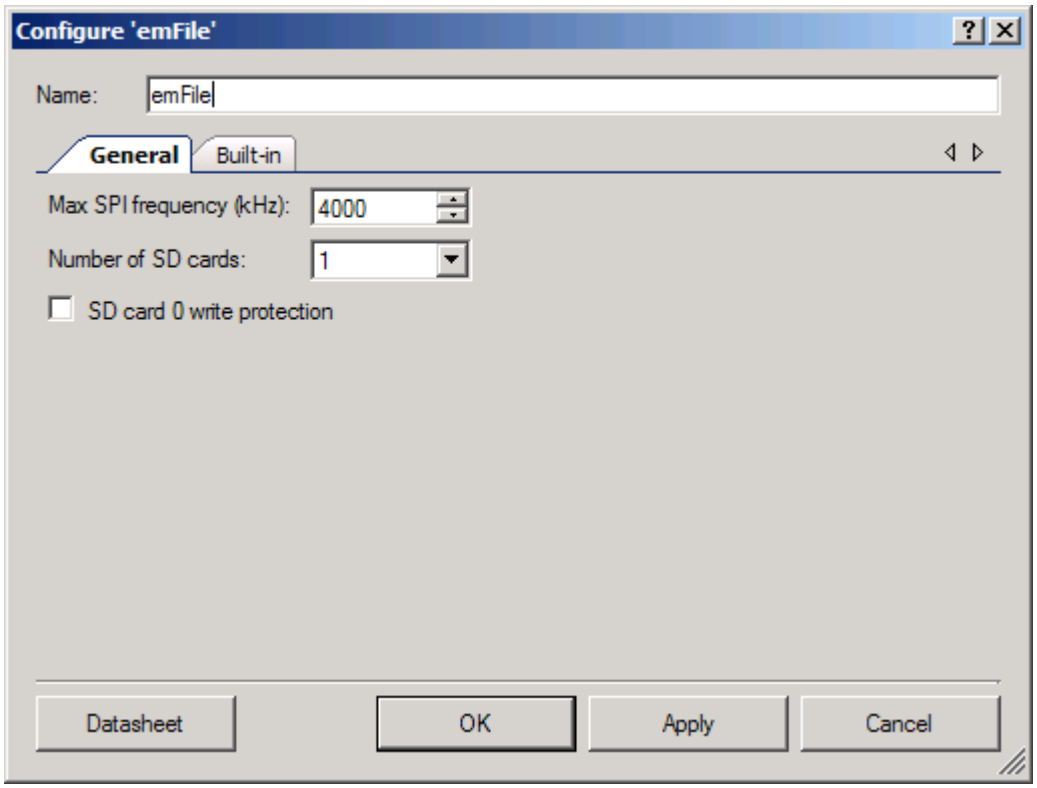
Serial Clock pin from the SPI Master. This pin should be connected to the clock pin of the SD card.

SPI0_CS – SPI3_CS

Card Select output pin. This pin should be connected to the nCS pin of the SD card.

Component Parameters

Drag an emFile onto your design and double-click it to open the Configure dialog.



The emFile provides the following parameters.

Max SPI frequency

Defines the maximum frequency at which the SPI Master components will clock an SD card. The value is specified in kHz. Possible values are in the range 400 to 25,000 kHz, the default setting is **4000**.

The range of frequencies is based on the specification for SD cards. Note that there are limitations for maximum frequency for the SPI Master (v2.20) component that is used in emFile. Refer to the SPI Master v2.20 datasheet "Timing Characteristics" for detailed information about the maximum frequency that this component can support.

Number of SD cards

Defines the number of SD cards in the emFile system. The value can be set between 1 and 4. The default setting is **1**.

SD card *n* write protection

Defines the write protect enable for each of *n* SD cards. It is disabled by default.

Placement

The emFile component is placed throughout the UDB array and all placement information is provided to the API through the *cyfitter.h* file.

Performance and Resource Usage

The performance of emFile depends on a set of parameters (CPU, compiler, optimization, size of payload data) and it is limited by the maximum speed of the SPI Master component. The following table contains read/write speed values depending on a variety of factors that affect emFile component performance.

Device	CPU Speed	SPI Clock	Mode	Performance (KBps)							
				1 Byte		256 Bytes		1 KB		8 KB	
				Write	Read	Write	Read	Write	Read	Write	Read
PSoC 3	24 MHz	8 MHz	Release	0.02	0.14	6.56	28.44	22.73	76.92	32.26	68.96
PSoC 3	48 MHz	8 MHz	Release	0.04	0.26	10.93	49.23	38.46	119.0	54.05	117.6
PSoC 3	24 MHz	4 MHz	Release	0.02	0.13	5.98	25.6	21.28	66.67	30.07	64.52
PSoC 3	48 MHz	4 MHz	Release	0.04	0.21	9.55	42.2	32.47	108.7	45.44	100
PSoC 5	24 MHz	8 MHz	Release	0.05	0.41	13.61	106.6	47.17	208.3	87.92	205.12
PSoC 5	48 MHz	8 MHz	Release	0.06	0.5	16.2	142.2	55.56	250	103.92	242.4
PSoC 5	24 MHz	4 MHz	Release	0.04	0.29	9.99	75.29	33.78	138.9	62.96	145.44
PSoC 5	48 MHz	4 MHz	Release	0.05	0.38	12.59	98.45	42.37	192.3	80	186.08

Memory Usage

emFile component memory usage varies depending on the application. The documentation from SEGGER gives a detailed description of how to calculate the memory resources. The following table contains memory usage values for some commonly used features of the file system. All values are in bytes.

emFile Module	Keil_PK51				GCC-4.4.1			
	Release		Debug		Release		Debug	
	Flash	SRAM	Flash	SRAM	Flash	SRAM	Flash	SRAM
File system core (SPI driver)	28035	4762	28799	4849	10440	4272	12432	4272
Read file	2668	6	2672	3	824	0	832	0
Write file	1927	0	1932	0	808	0	816	0
Open directory	2714	47	2733	47	408	0	416	0
Create directory	898	0	898	0	464	0	480	0
Remove file	75	0	75	0	48	0	48	0
Long file name support*	-	-	-	-	2232	0	2232	0
Low-level format	769	0	769	0	240	0	232	0
Format SD card	6063	0	6063	0	2296	0	2304	0

* Long file name support code is included automatically as part of the file system core for PSoC 3.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name “emFile_1” to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is “emFile.”

The emFile library routines automatically initialize and enable the emFile component. The only APIs provided for the emFile component are to support transitioning between power modes.

This section does not include a description of file system APIs of the emFile library because these APIs are described in the *emFile User Guide*, Chapter 4. You must include the *FS.h* header file in your *main.c* file to use file system APIs.

Note To use long file names (LFN) support on PSoC 5 devices, you must call FS_FAT_SupportLFN(). For PSoC 3 devices, this feature is enabled by default.

Function	Description
emFile_Sleep()	Prepares emFile to enter sleep mode.
emFile_Wakeup()	Restores emFile after coming out of sleep mode.
emFile_SaveConfig()	Saves the SPI Master configuration used by the HW driver.
emFile_RestoreConfig()	Restores the SPI Master configuration used by the HW Driver.

void emFile_Sleep(void)

Description: Prepares emFile to go to sleep.

Parameters: None

Return Value: None

Side Effects: None

void emFile_Wakeup(void)

Description: Restores emFile after coming out of sleep mode.

Parameters: None

Return Value: None

Side Effects: Calling the emFile_Wakeup() function without first calling the emFile_Sleep() or emFile_SaveConfig() function may produce unexpected behavior.

void emFile_SaveConfig(void)

Description: Saves the SPI Master configuration used by the HW driver. This function is called by emFile_Sleep().

Parameters: None

Return Value: None

Side Effects: None

void emFile_RestoreConfig(void)

Description: Restores the SPI Master configuration used by the HW Driver. This function is called by emFile_Wakeup().

Parameters: None

Return Value: None

Side Effects: Calling this function without first calling the emFile_Sleep() or emFile_SaveConfig() function may produce unexpected behavior.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

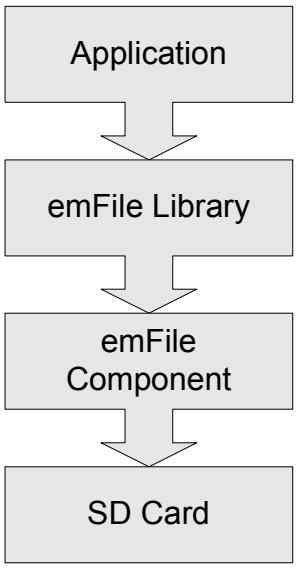
Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

Functional Description

File System Library Structure

The emFile file system implementation consists of two parts: the emFile component and the emFile file system library licensed from SEGGER Microcontroller. A file system application will use the APIs provided in the emFile library. That library will use the emFile component to provide the physical interface to the SD card using a SPI interface. The structure of the emFile file system is shown in Figure 9.

Figure 9. emFile Structure



Component Changes

Version 1.0 is the first release of the emFile component.

© 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® and CapSense® are registered trademarks, and SmartSense™, 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.