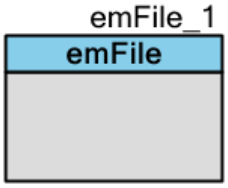
文件系统库（emFile）

1.20

特性



- 多达四个 SPI 模式中的安全数字（SD）卡
- 支持 FAT12/16 或 FAT32 格式
- 可以选择与操作系统（OS）集成
- 可选长文件名（LFN）处理

概述

emFile 组件提供了与使用 FAT 文件系统格式化的 SD 卡之间的接口。SD 卡规范包括用于与 SD 卡通信的多个硬件接口。该组件使用了 SPI 接口方式进行通信。可以将最多四个独立 SPI 接口用于与每个 SD 卡通信。同时支持 FAT12/16 和 FAT32 文件系统格式。该组件提供与 SD 卡之间的物理接口，使用 SEGGER Microcontroller 授权的 emFile 库提供的函数库，以操控 FAT 文件系统。

何时使用 emFile

使用 emFile 组件可访问格式化为 FAT12/16 或 FAT32 文件系统格式的 SD 卡。

入门

安装 emFile 库

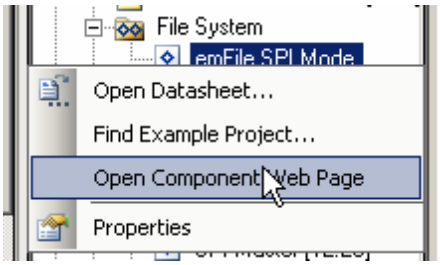
emFile 文件系统实现包含两个部分。第一部分是随 PSoC Creator 附带的 emFile 组件。第二部分是 SEGGER Microcontroller 授权的 emFile 文件系统库。该库以 zip 文件形式提供，可以从赛普拉斯网站 emFile 组件页面下载。

请按照以下步骤安装 emFile 库：

1. 打开 PSoC Creator 并转到 Component Catalog（组件目录）窗口。搜索 emFile 组件。该组件位于 **Cypress**（赛普拉斯）选项卡下的下面路径：**Communications > File System > emFile SPI Mode**。
2. 右键单击 emFile 组件。

这时将出现如图 1 中所示的菜单。

图 1. emFile 的打开组件网页



3. 单击“Open Component Web Page”（打开组件网页）。

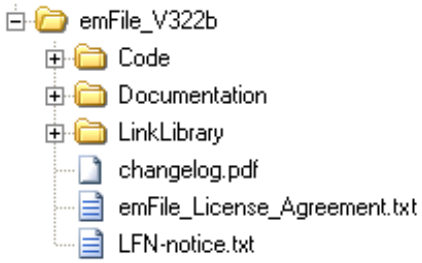
   进入含有最新文件系统库的 zip 文件的登录页面。

4. 下载该 zip 文件，并将其解压到所选的文件夹中（保持该 zip 文件的目录结构）。确保文件在解压后不是只读格式。

emFile 库目录组织

解压后的 emFile 库的目录结构类似于图 2 所示。

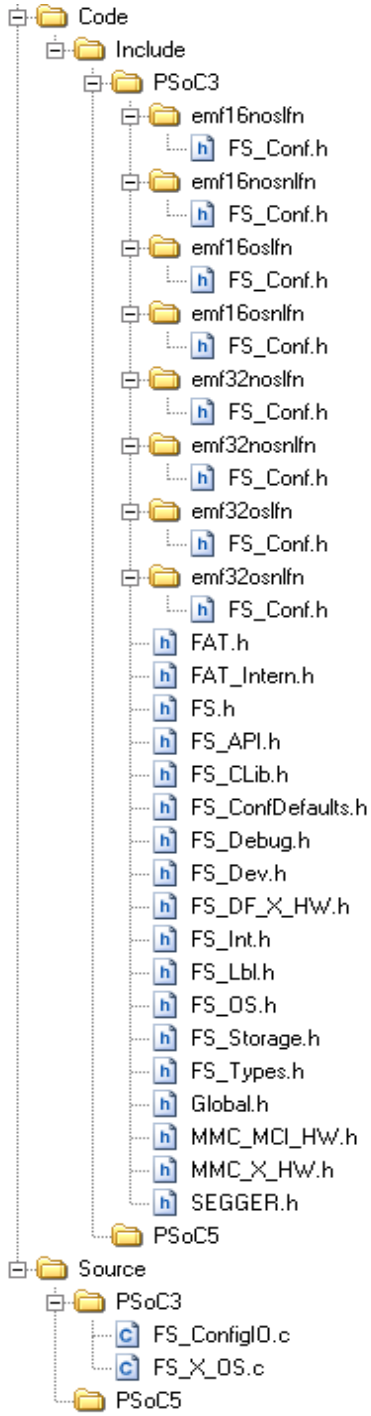
图 2. emFile 目录组织



Code（代码）目录包含以源代码格式提供的部分库。该目录具有两个子目录：Include（包括）和 Source（源代码）。Include（包括）目录提供库的头文件。Source（源代码）目录提供库的可执行部分，该库采用源代码格式。

图 3 显示的是 Code（代码）目录中的文件。该图显示了 PSoC 3 文件的目录组织。PSoC 5 的目录包含了 PSoC 5 器件系列（如 PSoC 5LP）的文件。PSoC 5 的目录类似于 PSoC 3 的文件目录组织。

图 3. emFile 的“Code”目录文件列表



Include（包括）部分划分为始终可应用于目录顶层的头文件，以及为特定 LinkLibary 选择的选项而使用的子目录中的头文件。子目录的名称指定为 emf<选项>。表 1 中介绍了所有选项。

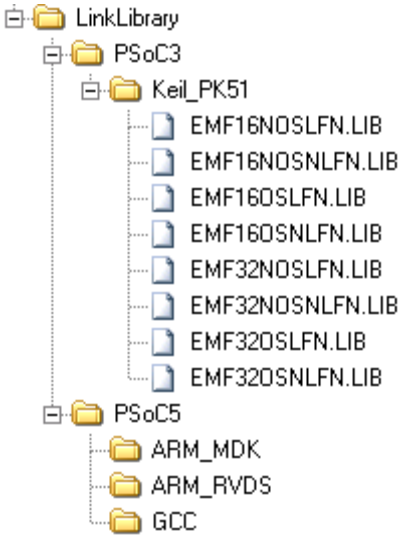
表 1. 选项

选项	说明
16	FAT 12/16 格式
32	FAT 32 格式
os	支持操作系统
nos	不支持操作系统
lfn	支持长文件名
nlfn	不支持长文件名

Documentation（文档）部分包含 SEGGER 微控制器 emFile 文件系统库的用户指南。

LinkLibrary 部分被划分为 PSoC 3 和 PSoC 5 两个目录。每个目录包含了用于每个受支持的工具链的子目录。这些目录是为所选的特定选项提供文件系统实现的对象库。使用本数据手册后面介绍的 Build Settings（构建设置）选项，该库被添加到工程中。扩展的 PSoC 3/Keil PK51 部分包含图 4 中显示的对象库。

图 4. LinkLibrary 结构



LinkLibrary 的命名约定为：<前缀>emf<选项>.<扩展名>。这里的选项与 include（包括）文件目录名称中使用的选项相同。前缀和扩展名特定于使用的工具链。请参见表 2。

表 2. 命名约定

工具链	前缀	扩展名
Keil PK51		lib
GCC	lib	a
ARM_MDK		lib
ARM_RVDS		lib

解压后的 emFile 目录中的文件为 *changelog.pdf*、*emFile_License_Agreement.txt* 以及 *LFN-notice.txt*。*changelog.pdf* 文件内容包含了在库中所进行的更改历史记录。*emFile_License_Agreement.txt* 文件包含了 emFile 库使用条款的最终用户许可协议。*LFN-notice.txt* 文件介绍了有关使用长文件名的注意内容。

选择文件系统（FS）库

当选择该库时，主要应考虑到 SD 卡的存储能力。FS_FormatSD()函数将分析该卡的存储容量，然后格式化为 FAT 16 或 FAT 32。当 SD 的存储容量小于或等于 2 GB 时，该卡将被格式化为 FAT 16；如果它超过 2 GB，则将被格式化为 FAT 32。因此，对于存储容量不大于 2 GB 的 SD 卡，应使用 FAT 16 库；则其余情况下应使用 FAT 32 库。

其次，还要考虑是否需要支持长文件名（LFN）。如果使用了“no LFN”（不支持长文件名）或短文件名的库，则无法构建不符合“8.3 filename”的文件名。“8.3 filename”意味着最多 8 个字符可用于实际文件名，以及最多 3 个字节可用于文件扩展。如果 SD 卡包含了长文件名的文件，并且使用“no LFN”库，则该库将为它生成一个短文件名。例如，名称为“LongNameFile.txt”的文件将变成“LONGNA~1.TXT”的短文件名。“LFN”库允许创建最大 256 个字节的文件名。

请注意，使用 LFN 时会涉及到法律问题。请参见 *LFN-notice.txt*，了解更多信息。

最后，还要决定是否需要使用一个外部的嵌入式操作系统（OS）。如果使用，OS 将控制代码流。特定的 OS 使用了一组具有不同的优先级的任务，那些任务之间会互相中断。由于 OS 的多任务的性质，因此在 OS 尝试同时执行多项使用文件系统资源的任务时，可能会有冲突。

要避免这种状况，必须实现若干 API 函数。否则，您将无法创建工程。这些 API 函数会锁定文件系统资源。每当任何文件系统 API 函数启动其操作时，都会锁定文件系统资源；则当文件系统 API 函数完成其操作时，将解锁文件系统资源。关于锁定/解锁功能的详细说明，请参考 *emFile 用户指南*中第 8 章“操作系统（OS）的集成”的内容。

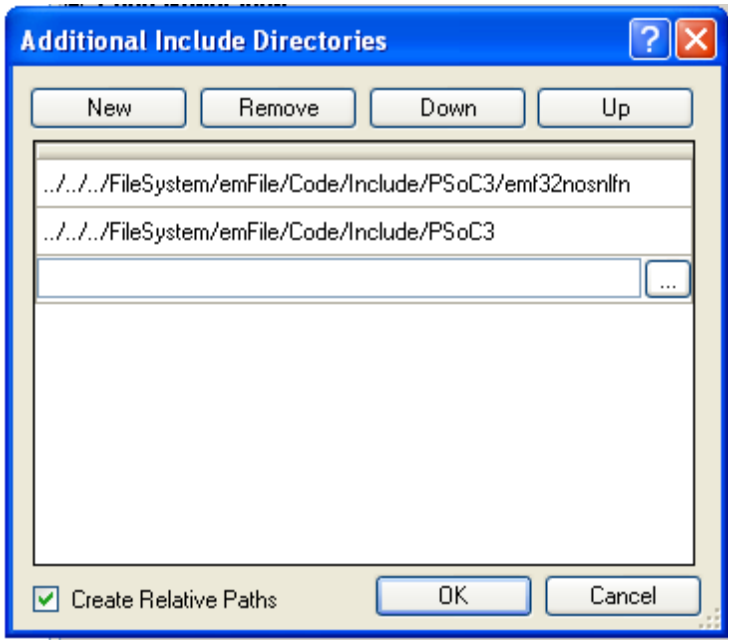
因为“no OS”（不支持操作系统）库采用了无限循环的代码流而不是多项任务，所以无需实现任何额外的 API 函数。

为 PSoC 3 应用创建 emFile 工程

在 PSoC 3 工程中使用 emFile 库，请按照下列步骤：

1. 选择所需的库。该选择基于是否需要 FAT12/16 还是 FAT32、应用程序是否使用操作系统以及是否需要长文件名支持等。该示例使用 emf32nosnlfn.lib（FAT32、不支持操作系统以及长文件名）。
2. 选择所需的 include （包括） 文件目录。依次选择 **Project > Build Settings > DP8051 Keil 9.51 > Compiler > General**。单击 **Additional Include Directories**（附加的包括目录）属性字段中的“**…**”按键。这时会显示 **Additional Include Directories** 对话框。单击 **New** 按键，然后为 PSoC 3 选择 include （包括） 目录并为所需特定选项选择 include （包括）目录。完成时，将显示如图 5 所示的对话框。

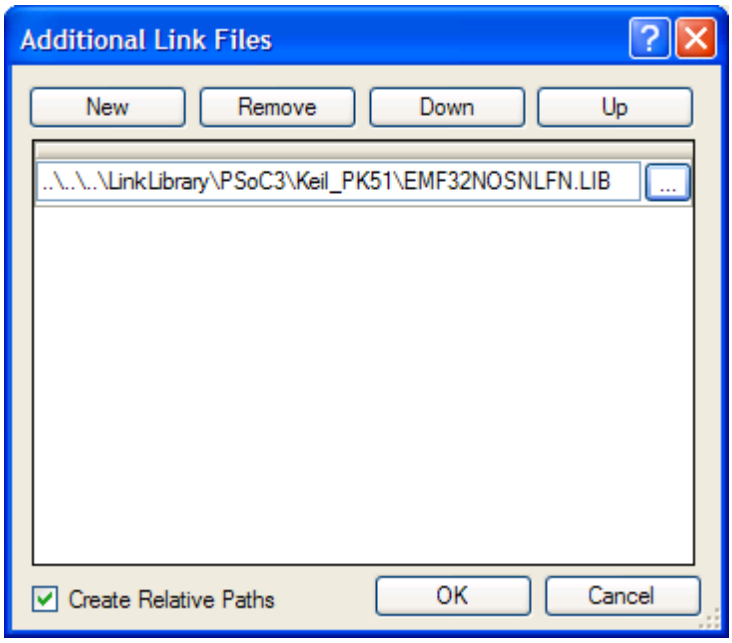
图 5. 添加 include（包括）目录



请注意，如果想在“Debug”（调试）和“Release”（释放）配置中运行工程，您需要向这两个配置添加 include（包括）目录。

3. 选择所需的链接库文件。依次选择 **Project > Build Settings > DP8051 Keil 9.51 > Linker > General**。单击 **Additional Link Files**（附加的链接文件）属性字段中的“**…**”按键。这时会显示 **Additional Link Files** 对话框。单击 **New** 按键，并根据所需特定选项来选择库文件。完成时，将显示如图 6 所示的对话框。

图 6. 添加链接库



注意：如果想在“Debug”（调试）和“Release”（释放）配置中运行工程，您需要向这两个配置添加链接库。

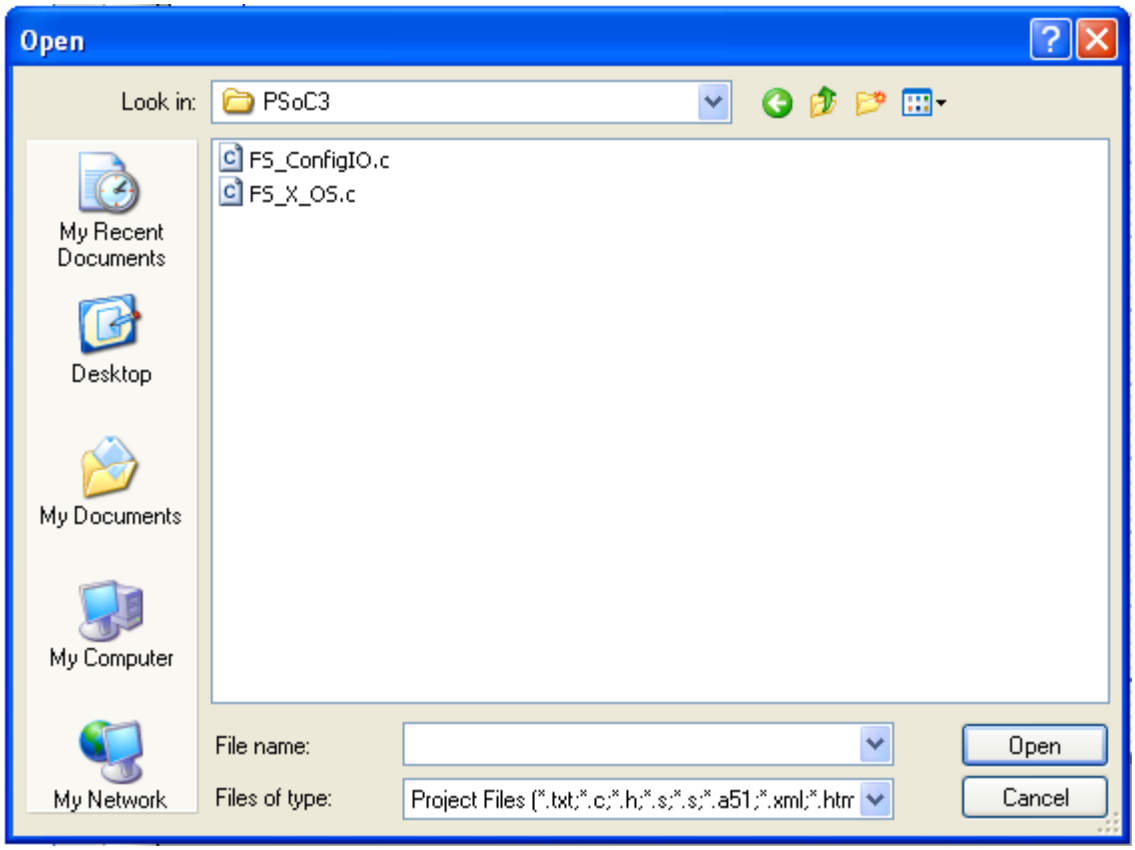
4. 如果想使用文件系统库的调试记录特性，请在工程中添加源文件 *FS_ConfigIO.c*。欲了解有关调试的详细信息，请参见 emFile 用户指南中第 9 章的内容。

   如果使用操作系统，还要添加 *FS_X_OS.c*。请参见 emFile 用户指南中第 8 章的内容，了解有关操作系统集成的更详细信息。

   这些文件可以从 Code（代码）/Source（源代码）/PSoC3 目录直接添加到工程中，或先复制到工程目录下，然后从那里添加。这些文件通常将进行编辑，因此您需要决定是更改原始文件还是特定工程的副本。

   通过 ***Project> Existing Item***，将这些文件添加到工程中。会打开图 7 中显示的对话框，供您选择文件。

图 7. 添加 PSoC 3 源文件



5. 向 main.c 函数中添加<FS.h>头文件。

现在 PSoC 3 emFile 库已经包含在工程中。

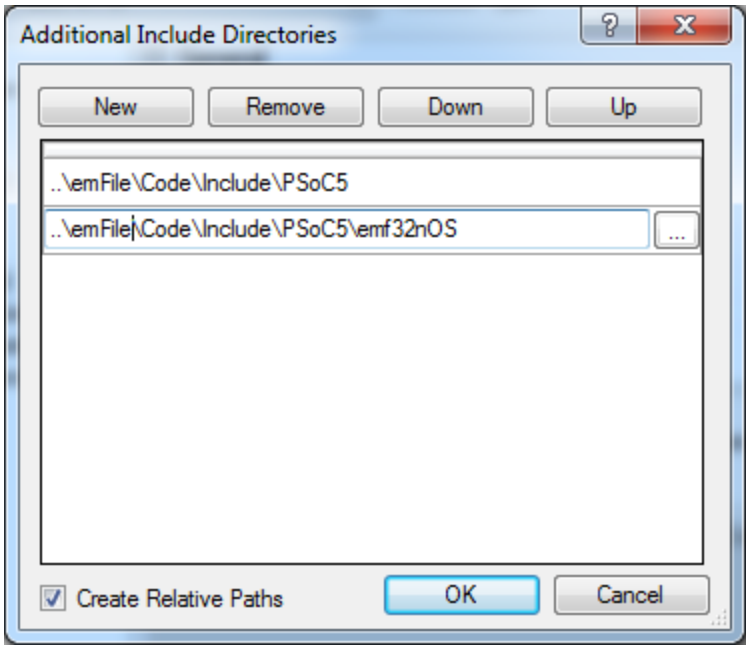
使用 GCC 工具链为 PSoC 5LP 应用创建 emFile 工程

为 PSoC 5LP 创建 emFile 工程的步骤与 PSoC 3 的步骤相同（除了在使用 GCC 工具链的情况下）。使用 GCC 工具链时，必须指定库文件所在的目录，而非以文件形式添加链接库。使用 GCC 工具链为 PSoC 5LP 应用创建 emFile 工程的步骤如下：

1. 选择所需的库。该选择基于是否需要 FAT12/16 还是 FAT32、应用程序是否使用操作系统以及是否需要长文件名支持等。该示例使用 emf32nosnlfn.lib（FAT32、不支持操作系统以及长文件名）。
2. 选择所需的 include（包括） 文件目录。依次选择 **Project > Build Settings > ARM GCC 4.7.3 > Compiler > General**。单击 **Additional Include Directories**（附加的包括目录）属性字段中的“**…**”按键。这时会显示 **Additional Include Directories** 对话框。单击

New 按键，然后为 PSoC 3 选择 include （包括）目录并为所需特定选项选择 include （包括）目录。完成时，将显示如图 8 所示的对话框。

图 8. 添加 include（包括）目录



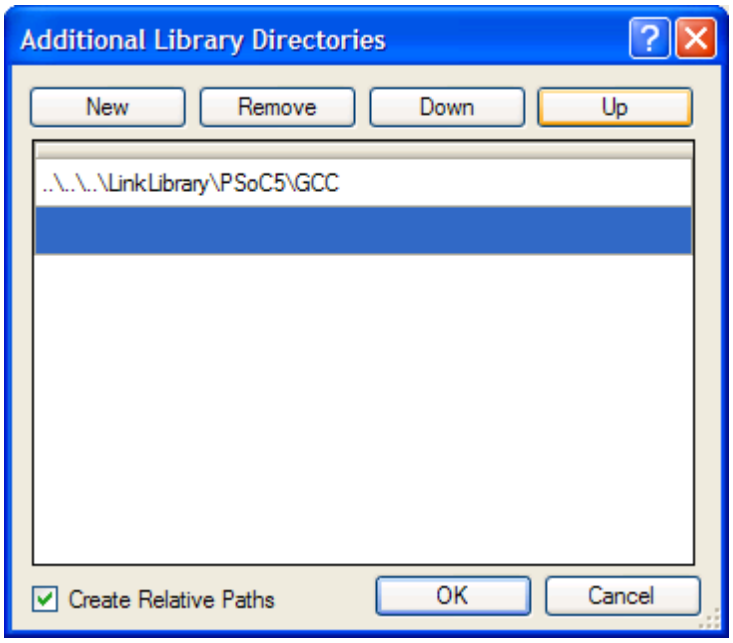
注意：如果想在“Debug”（调试）和“Release”（释放）配置中运行工程，您需要向这两个配置添加 include（包括）目录。

3. 依次选择 **Project > Build Settings > ARM GCC 4.7.3 > Linker > General > Additional Library Directories**。单击“Additional Library Directories”（附加的库目录）属性字段中的“…”按键。

   这时会显示 Additional Library Directories 对话框。

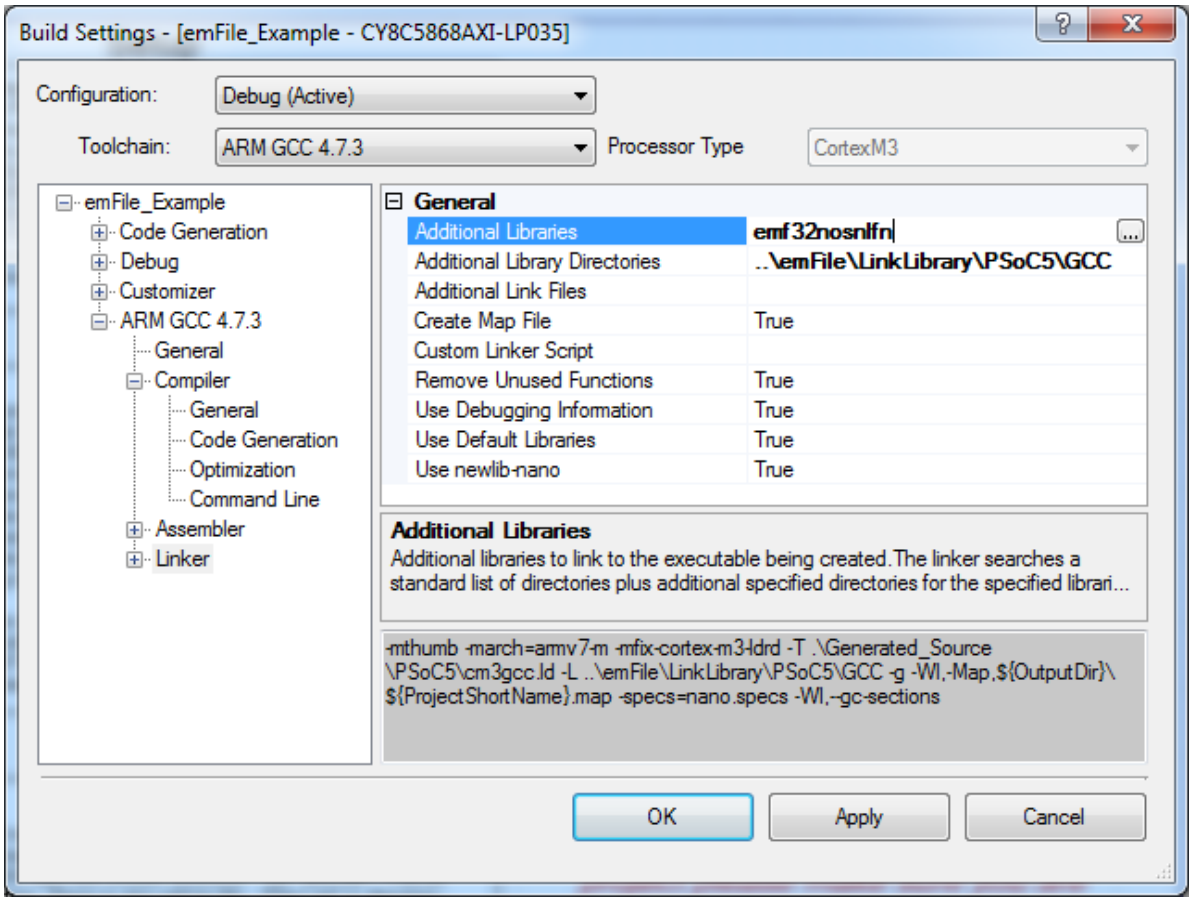
4. 单击 **New** 按键并为 GCC 库选择库目录。完成时，将显示如图 9 所示的对话框。

图 9. 添加库目录



5. 在库目录中指定库。
   a. 依次选择 **Project > Build Settings > ARM GCC 4.7.3 > Linker > General > Additional Libraries**。
   b. 键入库名称（从名称中去除“lib”前缀和“.a”后缀）。
   c. 假设使用 libemf32nosnlfn.a 库文件，则库名称应为 emf32nosnlfn，请参见图 10。

图 10. 指定库名



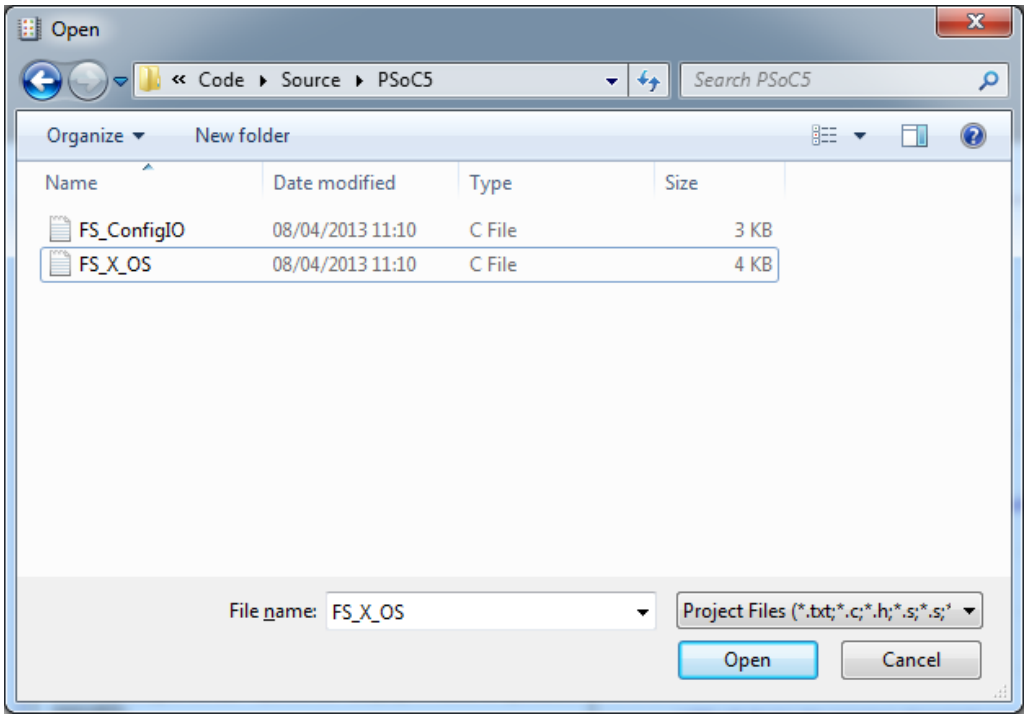
注意：如果想在“Debug”（调试）和“Release”（释放）配置中运行工程，您需要向这两个配置添加链接库。

6. 要想使用操作系统或记录特性，则应向工程中添加 *FS_ConfigIO.c* or *FS_X_OS.c*。欲了解有关使用操作系统集成和记录特性的详细信息，请分别参阅 emFile 用户指南中第 8 和第 9 章的内容。

   所述的文件可以从 Code（代码）/Source（源代码）/PSoC5 目录直接添加到工程中，或先复制到工程目录下，然后从那里添加。这些文件通常将进行编辑，因此您需要决定是更改原始文件还是特定工程的副本。

   通过选择 **Project> Existing Item**，将这些文件添加到工程中。此时会打开如图 11 中显示的对话框，供您选择文件。

图 11. 添加 PSoC 5 源文件



7. 向 main.c 函数中添加<FS.h>头文件。

现在 PSoC 5LP emFile 库已经包含在工程中。

使用 MDK 和 RVDS 工具链为 PSoC 5LP 应用创建 emFile 工程

*.zip 文件中也是 ARM MDK 和 ARM RVDS 编译器的库。ARM RVDS 和 ARM MDK 库的工程创建流程和 PSoC 3 应用的创建流程相同（除了需要使用 PSoC 5 目录中的文件和库的情况下）。

输入/输出接口

本节介绍 emFile 组件的各种输入和输出接口。I/O 列表中的星号(*)表示该 I/O 可能在 I/O 说明中列出的情况下从组件中排除。

在符号上，组件没有可见的连接。显示的所有连接是用于组件内部包含的引脚连接。这些引脚都会显示在设计范围资源引脚编辑器中。必须使用引脚编辑器将它们分配给合适的物理引脚。每个

接口具有四个引脚（使用索引 0 到 3 命名），表示组件支持四个独立的 SD 卡。这四个引脚中的一个会出现在设计中，具体情况取决于所选用的 SD 卡数量。下面是对这些引脚的说明。

SPI0_WP – SPI3_WP*

可选输入引脚，基于 **SD Card [0-3] Write Protection**（SD 卡[0-3]写保护）选项，这些选项配置了 SD 卡[0-3]的写保护。如果选中这些选项，这些引脚将显示。这些引脚的默认情况是不显示的，这表示不对 SD 卡进行写保护。

mosi0 – mosi3

SPI 主控的主出从入引脚。此引脚应连接到 SD 卡的 DI/CMD 引脚。

miso0 – miso3

SPI 主控的主入从出引脚。该引脚应连接到 SD 卡的 DO/DAT0 引脚。

sclk0 – sclk3

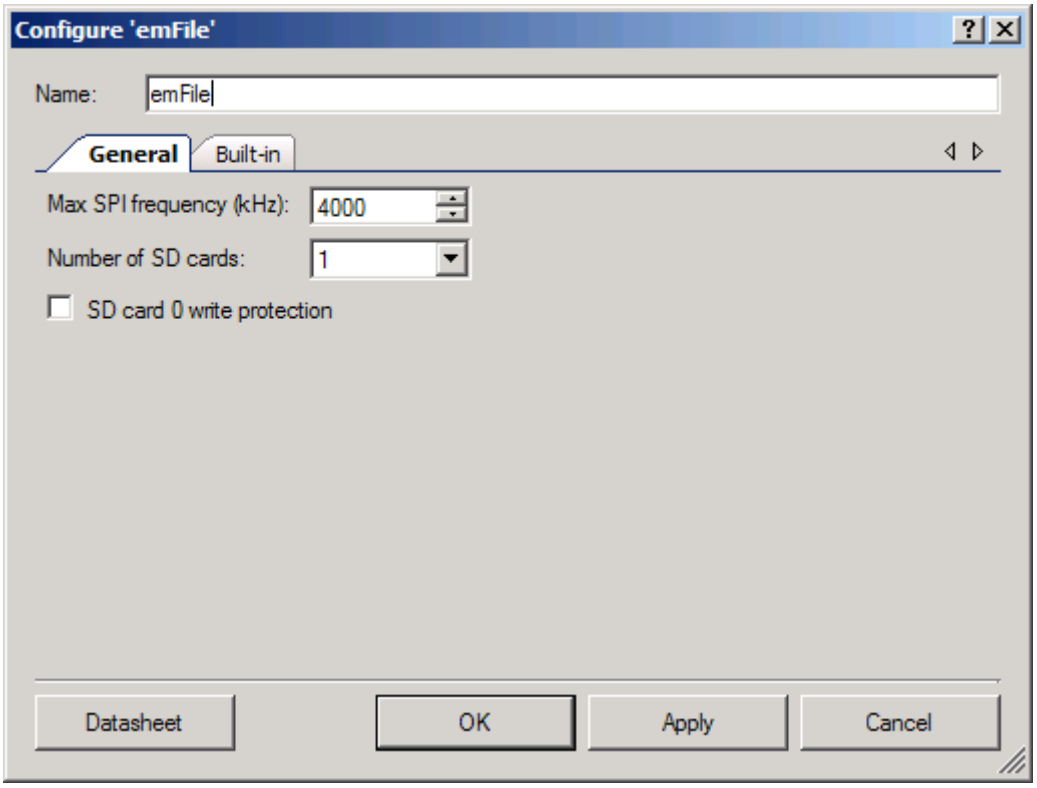
SPI 主控的串行时钟引脚。此引脚应连接到 SD 卡的时钟引脚。

SPI0_CS – SPI3_CS

卡选择输出引脚。该引脚应连接到 SD 卡的 nCS 引脚。

组件参数

将 emFile 拖动到您的设计中，双击打开 Configure（配置）对话框。



emFile 提供以下参数。

Max SPI frequency（最大 SPI 频率）

定义了 SPI 主控组件为 SD 卡提供时钟脉冲的最大频率。该值以 kHz 为单位。取值范围为 400 到 25,000 kHz，默认设置为 **4000**。

频率范围是基于 SD 卡规范的。请注意，对于 emFile 中使用的 SPI 主控（v2.20），组件的最大频率有所限制。有关此组件可以支持的最大频率的详细信息，请参考 SPI 主控 v2.20 数据手册“时序特性”。

Number of SD cards（SD 卡的数量）

定义 emFile 系统中的 SD 卡的数量。取值范围为 1 到 4。默认设置为 **1**。

SD card *n* write protection（SD 卡 n 写保护）

定义每个 *n*SD 卡的写保护使能。默认情况下，它被禁用。

应用编程接口（API）

通过应用编程接口（API），您可以使用软件对组件进行配置。下表列出并说明了每个函数的接口。以下各节将对每个函数加以说明。

默认情况下，PSoC Creator 将给设计中的第一个组件命名为“emFile_1”。 您可以将其重新命名为遵循标识符语法规则的任何唯一值。实例名称会成为每个全局函数名称、变量和符号常量的前缀。出于可读性考虑，下表中使用的实例名称为“emFile”。

emFile 库子程序会自动初始化并使能 emFile 组件。为 emFile 组件提供的 APIs 是为了支持功耗模式之间的切换。

本节不包括 emFile 库的文件系统 API 的说明，因为在 *emFile 用户指南*的第 4 章中已经介绍了这些 API。

注意： 若要在 PSoC 5LP 器件上使用长文件名（LFN）支持，必须调用 FS_FAT_SupportLFN()。对于 PSoC 3 器件，默认使能该特性。

函数	说明
emFile_Sleep()	准备emFile进入睡眠模式。
emFile_Wakeup()	在退出睡眠模式之后恢复emFile。
emFile_SaveConfig()	保存硬件驱动器使用的SPI主控配置。
emFile_RestoreConfig()	恢复硬件驱动器使用的SPI主控配置。

void emFile_Sleep(void)

说明： 准备emFile进入睡眠模式。

参数： 无

返回值： 无

其他影响： 无

void emFile_Wakeup(void)

说明： 在退出睡眠模式之后恢复emFile。

参数： 无

返回值： 无

其他影响： 调用emFile_Wakeup()函数前未调用emFile_Sleep()或emFile_SaveConfig()函数可能会产生意外行为。

void emFile_SaveConfig(void)

说明： 保存硬件驱动器使用的SPI主控配置。此函数由emFile_Sleep()调用。

参数： 无

返回值： 无

其他影响： 无

void emFile_RestoreConfig(void)

说明： 恢复硬件驱动器使用的SPI主控配置。此函数由emFile_Wakeup()调用。

参数： 无

返回值： 无

其他影响： 调用该函数前未调用emFile_Sleep()或emFile_SaveConfig()函数可能会产生意外行为。

MISRA 符合性

本节介绍了MISRA-C:2004符合性和本组件的偏差情况。定义了下面两种类型的偏差：

- 工程偏差 — 适用于所有 PSoC Creator 器件的偏差
- 特定偏差 — 仅适用于该组件的偏差

本节介绍了有关组件特定偏差的信息。*系统参考指南*的“MISRA 符合性”章节中介绍了工程偏差以及有关 MISRA 符合性验证环境的信息。

尚未根据 MISRA-C:2004 编码准则合规性，验证 emFile 组件源代码。

固件源代码示例

PSoC Creator 在“Find Example Project”（查找示例工程）对话框中提供了很多包括原理图和代码示例的示例工程。要查看特定组件实例，请打开“Component Catalog”中的对话框或原理图中的组件示例。要查看通用示例，请打开‘Start Page’或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的工程列表。

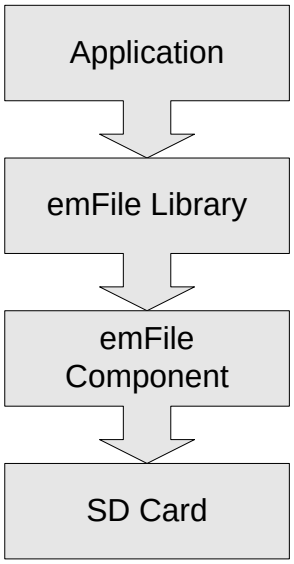
更多有关信息，请参考《PSoC Creator 帮助》部分中主题为“查找示例工程”的内容。

功能说明

文件系统库结构

emFile 文件系统实现由两个部分组成：emFile 组件和 SEGGER Microcontroller 授权的 emFile 文件系统库。文件系统应用会使用 emFile 库中提供的 API。该库会使用 emFile 组件提供以使用 SPI 接口的 SD 卡之间的物理接口。emFil 文件系统结构如图 12 所示。

图 12. emFile 结构



放置

emFile 组件放置于整个 UDB 阵列中，并且所有放置信息通过 *cyfitter.h* 文件提供给 API。

性能和资源使用情况

emFile 的性能取决于一组参数（CPU、编译器、优化、负载数据大小），并且受 SPI 主控组件的最大速度限制。下表包含读/写速度值（取决于各种影响 emFile 组件性能的因素）。

器件	CPU的速度	SPI时钟	模式	性能（KBps）							
				1字节		256个字节		1 KB		8 KB	
				写	读	写	读	写	读	写	读
PSoC 3	24 MHz	12 MHz	释放	0.02	0.09	4.46	20.00	15.38	47.17	22.04	47.34
PSoC 3	48 MHz	12 MHz	释放	0.03	0.18	8.15	38.20	28.33	87.72	41.67	93.02

器件	CPU的速度	SPI时钟	模式	性能（KBps）							
				1字节		256个字节		1 KB		8 KB	
				写	读	写	读	写	读	写	读
PSoC 3	24 MHz	8 MHz	释放	0.02	0.09	4.46	19.69	15.38	47.17	22.04	47.33
PSoC 3	48 MHz	8 MHz	释放	0.03	0.17	7.83	36.05	27.10	84.03	39.70	87.91
PSoC 3	24 MHz	4 MHz	释放	0.02	0.10	4.27	19.40	14.70	44.64	16.70	44.94
PSoC 3	48 MHz	4 MHz	释放	0.03	0.15	6.98	32.40	23.98	74.07	35.08	77.67

存储器使用情况

emFile 组件存储器的使用情况因应用而异。SEGGER 的文档提供了有关如何计算存储器资源的详细说明。下表包含文件系统一些常用功能的存储器使用情况。所有值都以字节为单位。

emFile模块	Keil_PK51				GCC-4.7.3			
	释放		调试		释放		调试	
	Flash	SRAM	Flash	SRAM	Flash	SRAM	Flash	SRAM
文件系统内核（SPI驱动器）	28035	4762	28799	4849	10872	4272	12376	4272
读取文件	2668	6	2672	3	832	0	832	0
写入文件	1927	0	1932	0	808	0	816	0
打开目录	2714	47	2733	47	408	0	424	0
创建目录	898	0	898	0	464	0	464	0
删除文件	75	0	75	0	56	0	64	0
支持长文件名 [1]	–	–	–	–	2216	0	2216	0
低级格式	769	0	769	0	248	0	256	0
格式化SD卡	6063	0	6063	0	2288	0	2288	0

1. 长文件名支持代码会自动包含在 PSoC 3 的文件系统内核中。

组件更改

版本	更改说明	更改原因/影响
1.20.b	已将**选择文件系统库**一节添加到本应用手册中。	本节为根据工程所需的特性选择正确的文件系统（FS）库提供了指南。
	已向本数据手册中添加了“使用MDK和RVDS工具链为PSoC 5LP应用创建emFile工程”的新一节。	
	已向“使用MDK和RVDS工具链为PSoC 5LP应用创建emFile工程”一节中添加了更详细的说明。	为创建PSoC 5LP的emFile工程提供更详细信息。
	已将“创建PSoC 5应用的emFile工程”改名为“使用GCC工具链创建PSoC 5LP应用的emFile工程”。	
1.20a	更新了图2，即emFile目录组织图	
1.20	添加了MISRA符合性一节。	该组件未进行MISRA符合性验证。
	将emFile组件中的SPI主控、时钟以及引脚组件更新为最新版本。	
1.10	更新了性能表中的值。	
	更新了emFile目录组织图。	emFile文件系统库（.zip）已被更新，因此要在数据手册中反映该更新内容。
	将emFile的SPI主控组件更新为最新版本。	
1.0	首次发行版本	