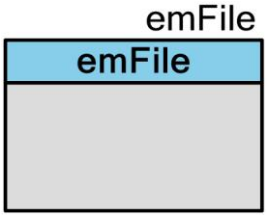
ファイル システム ライブラリ (emFile)

1.0

特長



- 最大 4 つまでのセキュア デジタル (SD) カード (SPI モード)
- FAT12/16 または FAT32 フォーマット
- オペレーティング システム (OS) と統合可能
- ロングファイルネーム (LFN) の取り扱い可能

概要説明

emFile コンポーネントは FAT ファイルシステムでフォーマットされた SD カードとのインターフェースを提供します。SD カードの仕様には、SD カードとの通信に使用する複数のハードウェア インタフェース オプションが含まれます。このコンポーネントは通信に SPI インタフェース手段を使用します。1 つの SD カードとの通信に、それぞれ最大 4 つまでの独立した SPI インタフェースが使用できます。FAT12/16 および FAT32 ファイル システム形式の両方がサポートされます。このコンポーネントには、SD カードの物理インタフェースがあり、SEGGER Microcontroller によりライセンス付与されている emFile ライブラリで動作します。これは、FAT ファイル システムの操作のための機能のライブラリを提供します。

emFile を使用するタイミング

emFile コンポーネントを使用して、FAT12/16 または FAT32 ファイル システム形式を使用してフォーマットされた SD カードにアクセスします。

はじめに

emFile ライブラリのインストール

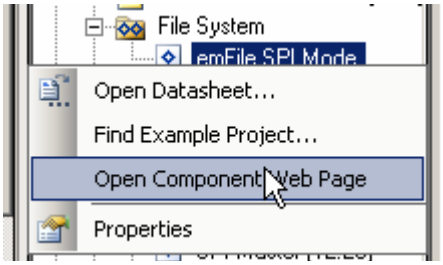
emFile ファイル システムの実装は 2 つの部分から成ります。1 つ目の部分は、PSoC Creator 同梱の emFile コンポーネントです。2 つ目の部分は、SEGGER Microcontroller によりライセンス付与されている emFile ファイル システム ライブラリです。ライブラリは zip ファイル形式で提供されており、サイプレスのウェブサイトの emFile コンポーネントのページからダウンロード可能です。

emFile ライブラリをインストールするには:

1. PSoC Creator を開き、[Component Catalog] ウィンドウに進みます。emFile コンポーネントを検索します。これは、**Cypress** タブ (**Communications > File System > emFile SPI Mode**) にあります。
2. emFile コンポーネントを右クリックします。

   図 1 に表示されているメニューが表示されます。

図 1. emFile Open Component Web Page



3. **Open Component Web Page** をクリックします。

   コンポーネントのランディング ページに移動します。ここには、最新のファイル システム ライブラリの zip ファイルがあります。
4. zip ファイルをダウンロードして、zip ファイルのディレクトリ構造はそのまま保ち、希望するフォルダで解凍します。解凍後のファイルが読み取り専用になっていないことを確認します。

emFile ライブラリ ディレクトリの構成

zip ファイルを解凍した後の emFile ライブラリのディレクトリ構造は、図 2 に表示されるものと類似しています。

図 2. emFile ディレクトリの構造



Code ディレクトリには、ソース コード形式で提供されるライブラリの一部が含まれます。このディレクトリには 2 つのサブディレクトリがあります。Include および Source です。Include ディレクトリには、ライブラリのヘッダー ファイルがあります。Source ディレクトリには、実行可能な一部のライブラリがあり、ソース形式になっています。

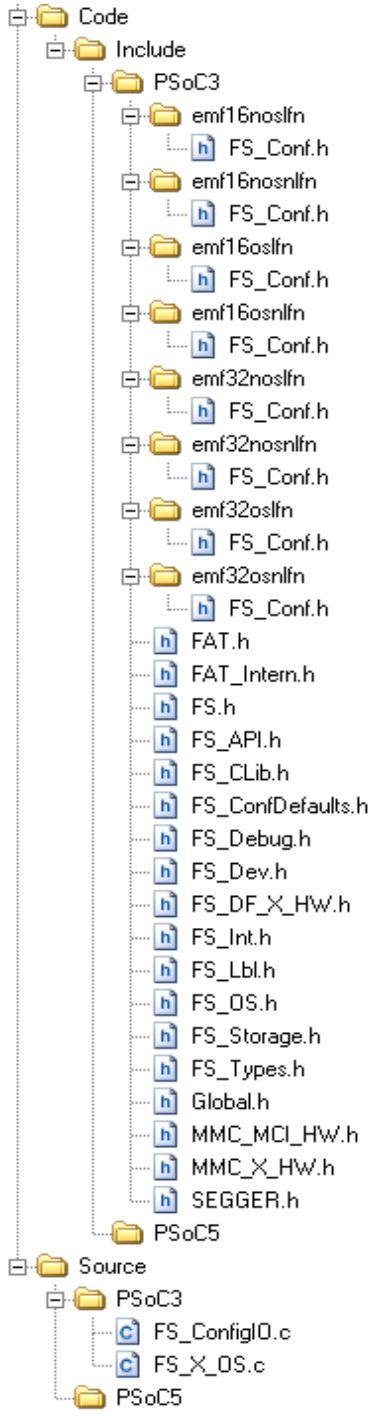
Include セクションは、ディレクトリのトップ レベルに常に適用されるヘッダー ファイルと、特定の LinkLibrary に対して選択したオプションによって使用される場合があるサブディレクトリのヘッダー ファイルに分類されます。サブディレクトリの名前は emf<options> と指定されます。すべてのオプションは表 1 で説明されています。

表 1. オプション

オプション	説明
16	FAT 12/16 形式
32	FAT 32 形式
OS	オペレーティング システムのサポート
nos	オペレーティング システムのサポートなし
lfn	ロングファイルネームサポート
nlfn	ロングファイルネームサポートなし

図 3 は Code ディレクトリのファイルを表示しています。図では、PSoC 3 のディレクトリ構成を表示しています。PSoC 5 も同様の構成になります。

図 3. emFile「Code」ディレクトリ ファイルのリスト

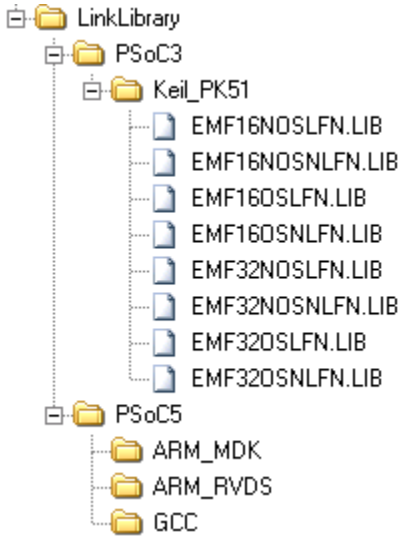


ドキュメント セクションには『SEGGER Microcontroller emFile ファイル システム ライブラリのユーザ ガイド』が含まれます。

LinkLibrary セクションは PSoC 3 ディレクトリと PSoC 5 ディレクトリに分類されます。各ディレクトリには、それぞれでサポートされるツールチェーンのサブディレクトリが含まれます。これらのディレクトリ内にはオブジェクト ライブラ

リがあり、ファイル システムを選択した特定のオプションで実装できます。このデータシート後半で説明されている Build Settings オプションを使用して、ライブラリがプロジェクトに追加されます。拡張した PSoC 3/Keil PK51 セクションには、図 4 に表示されるオブジェクト ライブラリが含まれます。

図 4. LinkLibrary の構成



LinkLibrary の命名規則:<prefix>emf<options>.<extension>. オプションは、ファイル ディレクトリ名に含む場合に使用されるオプションと同様です。プリフィックスおよび拡張子は、使用するツールチェーンにより異なります。表 2を参照してください。

表 2. 命名規則

ツールチェーン	接頭辞	拡張子
Keil PK51		lib
GCC	lib	a
ARM_MDK		lib
ARM_RVDS		lib

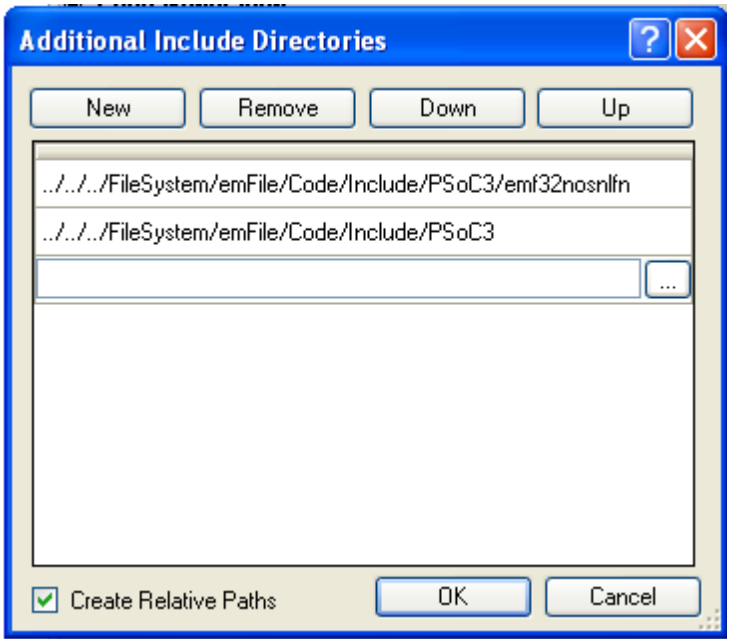
PSoC 3 アプリケーション用の emFile プロジェクトの作成

PSoC 3 プロジェクトで emFile ライブラリを使用するには:

1. 必要なライブラリを決定します。この決定は、FAT12/16 または FAT32 が必要か、アプリケーションで OS を使用するか、また長いファイル名サポートが必要かどうかに基づきます。この例では emf32nosnlfn.lib (FAT32、OS なし、長いファイル名サポートなし) を使用しています。

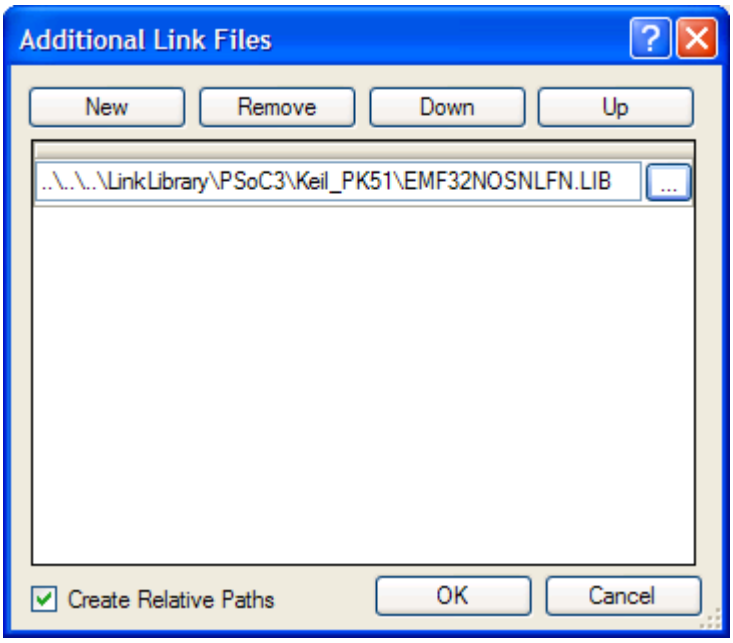
2. 必要なファイル ディレクトリを選択します。**Project > BuildSettings > Compiler > General** に進みます。「**…**」ボタン (**Additional Include Directories** プロパティ フィールド) をクリックします。 **Additional Include Directories** ダイアログが表示されます。**New** ボタンをクリックし、PSoC 3 に含まれるディレクトリおよびご希望の特定のオプションに含まれるディレクトリを選択します。完了したら、ダイアログは図 5 のように表示されるはずです。

図 5. 追加のディレクトリを含む



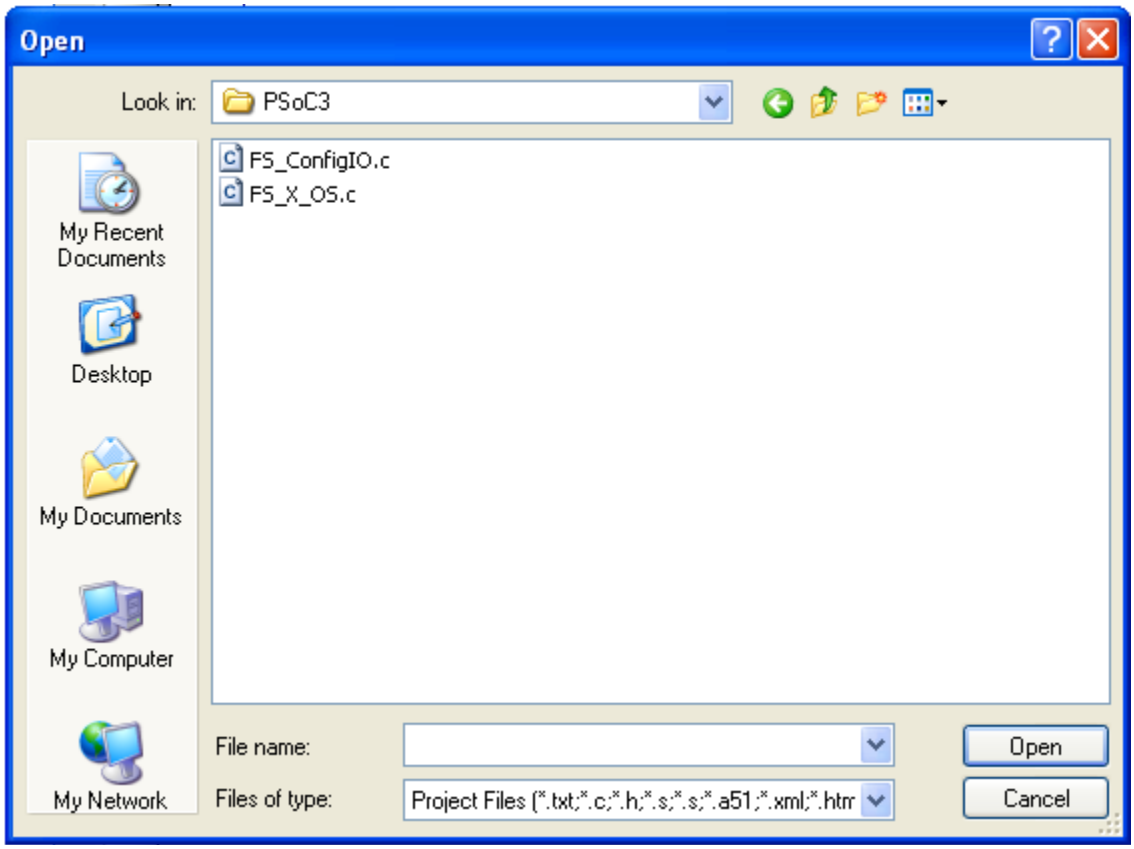
3. 必要なリンク ライブラリ ファイルを選択します。**Project > BuildSettings > Linker > General** へ進みます。「**…**」ボタン (**Additional Include Directories** プロパティ フィールド) をクリックします。**Additional Link Files** ダイアログが表示されます。**New** ボタンをクリックし、希望する特定のオプションに基づいたライブラリ ファイルを選択します。完了後、ダイアログは図 6 のように表示されるはずです。

図 6. リンク ライブラリの追加



4. ソース ファイル *FS_ConfigIO.c* をプロジェクトに追加します。OS を使用している場合、*FS_X_OS.c* も追加します。これらのファイルは、Code/Source/PSoC3 ディレクトリから直接プロジェクトに追加するか、まずプロジェクト ディレクトリにコピーしてから、追加することができます。通常、ファイルは編集されるため、元のファイルを変更するか、プロジェクト専用のファイルをコピーするかを決定する必要があります。ファイルは、***Project > Existing Item*** を使用してプロジェクトに追加されます。図 7 で表示されているダイアログが開き、ファイルを選択することができます。

図 7. ソース ファイルの追加



これで、emFile ライブラリがプロジェクトに含まれました。

PSoC 5 アプリケーション用の emFile プロジェクトの作成

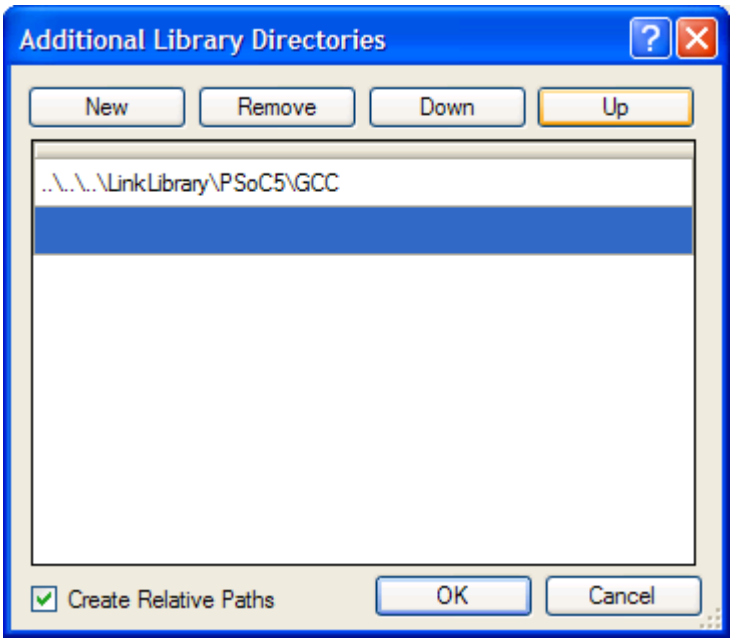
PSoC 5 用に emFile プロジェクトを作成するためのステップは、GCC ツールチェーンを使用する場合以外は、PSoC 3 と同様です。GCC ツールチェーンを使用する場合、リンク ライブラリをファイルとして追加するのではなく、ライブラリ ファイルが置かれているディレクトリを指定する必要があります。

BuildSettings > Linker > General > Additional Library Directories へ進みます。「**…**」ボタン (**Additional Library Directories** プロパティ フィールド) をクリックします。

Additional Library Directories ダイアログが表示されます。

New ボタンをクリックし、GCC ライブラリのライブラリ ディレクトリを選択します。完了後、ダイアログは図 8 のように表示されるはずです。

図 8. ライブラリ ディレクトリの追加



ライブラリ ディレクトリ内のライブラリを指定します。

a. **BuildSettings > Linker > General > Additional Libraries** と移動します。

b. プリフィックスの「lib」およびサフィックスの「.a」を除いたライブラリ名を入力します。

   libemf32nosnlfn.a ライブラリ ファイルを使用していると仮定すると、ライブラリ名は emf32nosnlfn となります。

入出力接続

このセクションでは、emFile コンポーネントの様々な入力および出力の接続について説明します。I/O のリストでアスタリスク (*) が付いている場合、その I/O の説明で一覧されている条件で、コンポーネントから I/O が除外されている可能性があることを示しています。

シンボル上のコンポーネントの接続は表示されません。表示される接続はすべてピン接続であり、コンポーネント内部に含まれます。これらのピンは [Design-Wide Resources Pin Editor] で表示されます。[Pin Editor] を使用して、適切な物理ピンに割り当てる必要があります。それぞれの接続には、インデックス 0 ～ 3 と名付けられた 4 つのピンがあります。これらは、コンポーネントがサポートする 4 つの独立した SD カードを示しています。選択した SD カード数により、これらのピンのうち 1 ～ 4 つをデザインに含めることができます。以下は、これらのピンの詳細です。

SPI0_WP – SPI3_WP*

SD Card [0-3] Write Protection オプションに基づいたオプションの入力ピンであり、SD カード [0-3] の書き込み保護を設定します。オプションにチェックが入っていると、これらのピンが含まれます。デフォルトの状態では、これらのピンは含まれておらず、SD カードは書き込み保護されていないことを示します。

mosi0 – mosi3

SPI マスタのマスタ アウト スレーブ イン端子。この端子は SD カードの DI/CMD 端子に接続する必要があります。

miso0 – miso3

SPI マスタのマスタ イン スレーブ アウト端子。この端子は SD カードの DO/DAT0 端子に接続する必要があります。

sclk0 – sclk3

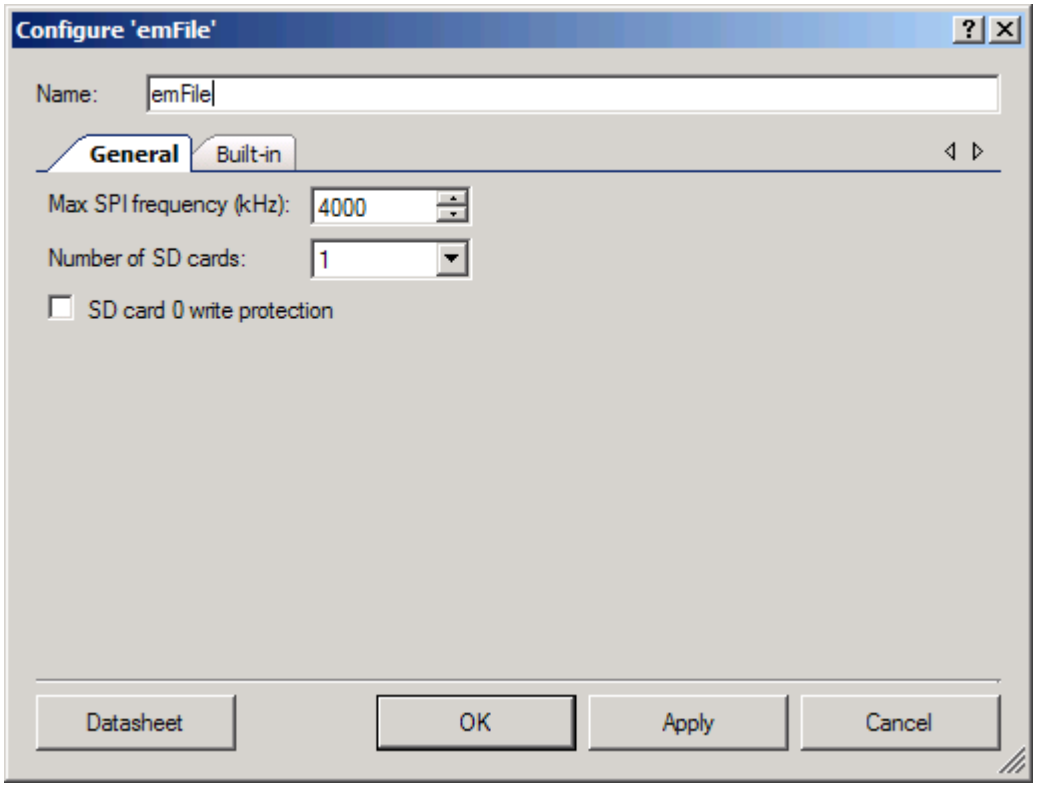
SPI マスタのシリアル クロック端子。この端子は SD カードのクロック端子に接続する必要があります。

SPI0_CS – SPI3_CS

カード選択出力端子。この端子は SD カードの nCS 端子に接続する必要があります。

コンポーネント パラメータ

emFile をデザイン上にドラッグし、ダブルクリックして [Configure] ダイアログを開きます。



emFile には、次のパラメータがあります。

最大 SPI 周波数

SPI マスタ コンポーネントが SD カードをクロックする最大周波数を定義します。指定する値は kHz 単位です。400 ～ 25,000 kHz の範囲の値を設定できます。デフォルト値は **4000** です。

周波数の範囲は SD カードの仕様に基づきます。emFile で使用される SPI マスタ (v2.20) コンポーネントの最大周波数には限度があることに注意してください。SPI マスタ v2.20 データシート「タイミング特性」で、このコンポーネントがサポートできる最大周波数の詳細情報を参照してください。

SD カードの数

emFile システムの SD カードの数を定義します。値は、1 から 4 の間で設定できます。デフォルトの設定は **1** です。

SD カード *n* 書き込み保護

それぞれの *n* SD カードについて、書き込み保護の有効を定義します。これは、デフォルトでは無効になっています。

配置

emFile コンポーネントは、UDB アレイ全体に配置され、すべての配置情報は、*cyfitter.h* ファイルを通して API に提供されます。

パフォーマンスとリソース利用

emFile のパフォーマンスはパラメータ (CPU、コンパイラ、最適化、ペイロード データのサイズ) により異なり、SPI マスタ コンポーネントの最大速度に制限されます。次の表には、emFile コンポーネントのパフォーマンスに影響を及ぼすさまざまな要素により異なる読み取り/書き込み速度の値が含まれます。

デバイス	CPU 速度	SPI クロック	モード	性能 (KBps)							
				1 バイト		256 バイト		1 KB		8 KB	
				書き込み	読み取り	書き込み	読み取り	書き込み	読み取り	書き込み	読み取り
PSoC 3	24 MHz	8 MHz	リリース	0.02	0.14	6.56	28.44	22.73	76.92	32.26	68.96
PSoC 3	48 MHz	8 MHz	リリース	0.04	0.26	10.93	49.23	38.46	119.0	54.05	117.6
PSoC 3	24 MHz	4 MHz	リリース	0.02	0.13	5.98	25.6	21.28	66.67	30.07	64.52
PSoC 3	48 MHz	4 MHz	リリース	0.04	0.21	9.55	42.2	32.47	108.7	45.44	100
PSoC 5	24 MHz	8 MHz	リリース	0.05	0.41	13.61	106.6	47.17	208.3	87.92	205.12
PSoC 5	48 MHz	8 MHz	リリース	0.06	0.5	16.2	142.2	55.56	250	103.92	242.4
PSoC 5	24 MHz	4 MHz	リリース	0.04	0.29	9.99	75.29	33.78	138.9	62.96	145.44
PSoC 5	48 MHz	4 MHz	リリース	0.05	0.38	12.59	98.45	42.37	192.3	80	186.08

メモリの使用

emFile コンポーネントのメモリの使用は、アプリケーションにより異なります。SEGGER のドキュメントには、メモリリソースを計算する方法について、詳しい説明が記載されています。次の表は、ファイル システムで一般的に使用される機能でのメモリ使用の値が含まれています。すべての値はバイト単位です。

emFile モジュール	Keil_PK51				GCC-4.4.1			
	リリース		デバッグ		リリース		デバッグ	
	フラッシュ	SRAM	フラッシュ	SRAM	フラッシュ	SRAM	フラッシュ	SRAM
ファイル システム コア (SPI ドライバー)	28035	4762	28799	4849	10440	4272	12432	4272
ファイルの読み取り	2668	6	2672	3	824	0	832	0
ファイルの書き込み	1927	0	1932	0	808	0	816	0
ディレクトリを開く	2714	47	2733	47	408	0	416	0
ディレクトリの作成	898	0	898	0	464	0	480	0
ファイルの削除	75	0	75	0	48	0	48	0
ロングファイルネームサポート*	-	-	-	-	2232	0	2232	0
低レベル フォーマット	769	0	769	0	240	0	232	0
SD カードのフォーマット	6063	0	6063	0	2296	0	2304	0

* ロングファイルネームサポート コードは、PSoC 3 のファイル システム コアの一部として、自動的に含まれています。

アプリケーション プログラミング インタフェース

アプリケーション プログラミング インタフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインタフェースとその説明を示しています。その次のセクションでは、各関数について詳しく説明します。

デフォルトでは、PSoC Creator はインスタンス名「emFile_1」をデザイン上のコンポーネントの最初のインスタンスに割り当てます。コンポーネントのインスタンス名は、識別子の文法ルールに従って固有の名前に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名の接頭辞になります。読みやすいように、下表では「emFile」というインスタンス名を使用しています。

emFile ライブラリのルーチンは emFile コンポーネントを自動的に初期化し、有効にします。emFile コンポーネント用に提供される唯一の API は、電源モードの移行をサポートするためのものです。

このセクションには、emFile ライブラリのファイル システム API を含みません。これらの API は『*emFile ユーザ ガイド*』第 4 章で説明されています。ファイル システム API を使用するには、*FS.h* ヘッダファイルを *main.c* ファイル中でインクルードする必要があります。

注 ロングファイルネーム (LFN) サポートを PSoC 5 デバイスで使用するには、FS_FAT_SupportLFN() を呼び出す必要があります。PSoC 3 デバイスでは、この機能はデフォルトで有効になっています。

関数	説明
emFile_Sleep()	スリープ モードに入れるように emFile を準備します。
emFile_Wakeup()	スリープ モードから復帰した後 emFile を復元します。
emFile_SaveConfig()	ハードウェア ドライバーで使用した SPI マスタ構成を保存します。
emFile_RestoreConfig()	ハードウェア ドライバーで使用した SPI マスタ構成を復元します。

void emFile_Sleep(void)

説明: スリープに入れるように emFile を準備します。

パラメータ: なし

戻り値: なし

注意事項: なし

void emFile_Wakeup(void)

説明: スリープ モードから復帰した後 emFile を復元します。

パラメータ: なし

戻り値: なし

注意事項: 最初に emFile_Sleep() または emFile_SaveConfig() 関数を呼び出すことなく emFile_Wakeup() 関数を呼び出すと、予期しない動作を引き起こす場合があります。

void emFile_SaveConfig(void)

説明: ハードウェア ドライバーで使用した SPI マスタ構成を保存します。この関数は、emFile_Sleep() 関数に呼び出されます。

パラメータ: なし

戻り値: なし

注意事項: なし

void emFile_RestoreConfig(void)

説明: ハードウェア ドライバーで使用した SPI マスタ構成を復元します。この関数は、emFile_Wakeup() 関数に呼び出されます。

パラメータ: なし

戻り値: なし

注意事項: emFile_Sleep() または emFile_SaveConfig() 関数を呼び出す前に、この関数を呼び出した場合、予期しない動作を引き起こす場合があります。

ファームウェア ソースコードの例

PSoC Creator は、Find Example Project ダイアログに数多くのプロジェクト例を提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、Component Catalog または回路図に置いたコンポーネント インスタンスからダイアログを開きます。一般例については、「Start Page (スタート ページ)」または **File** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

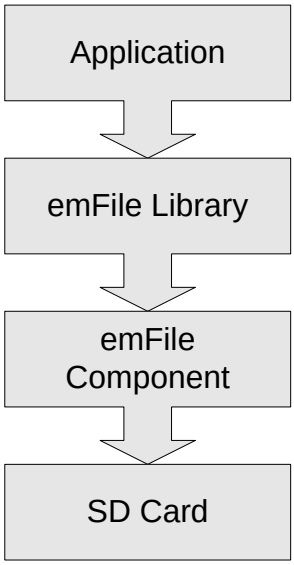
詳しくは、PSoC Creator ヘルプの「Find Example Project (サンプルプロジェクトを探す)」を参照してください。

機能説明

ファイル システム ライブラリ構造

emFile ファイル システムの実装は 2 つの部分から成ります。emFile コンポーネントと、SEGGER Microcontroller によりライセンス付与されている emFile ファイル システム ライブラリです。ファイル システム アプリケーションは、emFile ライブラリにある API を使用します。このライブラリでは、emFile コンポーネントを使用して、SPI インタフェースを使用した SD カードに物理インタフェースを提供します。emFile ファイル システムの構造は図 9 に表示されています。

図 9. emFile 構造



コンポーネントの変更

バージョン 1.0 は emFile コンポーネントの最初のリリースです。

Copyright © 2005-2012 Cypress Semiconductor Corporation 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporation は、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™及び Programmable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)は Cypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項:サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。