

## AN89659

### SPI F-RAM を PSoC® 4 でインターフェース

作成者: Shivendra Singh

関連プロジェクト: あり

関連製品ファミリ: SPI F-RAM – FM25XXX

ソフトウェア バージョン: 3.0 コンポーネント パック 7 以降

関連アプリケーション ノート: [AN304](#)、[AN79953](#)、[AN87352](#)

AN89659 では、回路例、タイミング図、および疑似コードの助けを借りてシリアル ペリフェラル インターフェース (SPI) F-RAM をサイプレスの PSoC® 4 (プログラマブル システムオンチップ) デバイスでインターフェースする方法を示します。このアプリケーション ノートは、SPI F-RAM を別の標準的な SPI マスター コントローラとインターフェースする参照設計ガイドとしても使用できます。このアプリケーションは、関連する PSoC 4 のサンプル プロジェクトを含んでいます。

## 目次

SPI F-RAM を PSoC® 4 でインターフェース .....	1
目次 .....	1
はじめに .....	1
SPI F-RAM インターフェース .....	2
SPI 動作モード .....	3
PSoC 4 で SPI マスターの実装 .....	3
UDB を使用して SPI マスターの実装 .....	3
SCB を使用して SPI マスターの実装 .....	3
ビット パンギングを使用した SPI マスターのコンフィギュレーション .....	5
SPI F-RAM の入力ピンのコンフィギュレーション .....	5
SPI F-RAM の動作電圧 .....	6
SPI F-RAM のオペコード .....	6
SPI F-RAM でのアドレス指定 .....	8
メモリ容量のアップグレード .....	8
SPI F-RAM 動作の例 .....	8
ステータス レジスタの動作 .....	9
F-RAM の読み書き処理 .....	10
まとめ .....	14
付録 A: PSoC 4 のサンプル プロジェクト .....	15
サンプル プロジェクトのピン配置 .....	15
SPI F-RAM コンポーネントをプロジェクトに統合 .....	15
改訂履歴 .....	20
ワールドワイドな販売と設計サポート .....	21
製品 .....	21
PSoC®ソリューション .....	21

## はじめに

SPI F-RAM は高度な強誘電体プロセスを使用するシリアル不揮発性メモリです。強誘電体ランダム アクセス メモリ (F-RAM) は、シリアル EEPROM と他の不揮発性メモリに起因する複雑性、オーバーヘッド、およびシステム レベルの信頼性の問題を取り除いた不揮発性 RAM です。シリアル EEPROM やフラッシュ メモリと違って、F-RAM は書き込み遅延 (NoDelay™) を起こさずに、バス速度で書き込み動作を実行します。データはメモリ アレイに直接書き込まれ、次のバス サイクルはデータ ポーリングを必要としないで、すぐに開始できます。F-RAM 製品は、EEPROM とフラッシュ メモリより桁違いに多い  $10^{14}$  回の書き換え回数が可能です。また、F-RAM はシリアル EEPROM やフラッシュ メモリより消費電力も低いです。シリアル EEPROM と比べたシリアル F-RAM の利点についての詳細は、アプリケーション ノート [AN87352 – F-RAM for Smart E-Meters](#) を参照してください。

サイプレスの PSoC は、単一チップ上に設定可能なアナログとプログラム可能なデジタル周辺機能、メモリ、およびマイクロコントローラを集積した、真のプログラマブル組込みシステムオンチップです。PSoC 4 は ARM Cortex-M0 ベースの PSoC ファミリのデバイスです。PSoC 4 の詳細については、[AN79953 – Getting Started with PSoC® 4](#) を参照してください。

このアプリケーションノートでは、PSoC 4 内での SPI F-RAM の接続と機能を詳しく説明します。ここでのハードウェアの推奨事項は必須ではありませんが、それらを適用することは、より強固な設計につながります。

このアプリケーション ノートでは、システム レベルで SPI F-RAM の動作を説明するために、タイミング図と PSoC 4 ベースの疑似コードの助けを借りて幾つかのオペコードを説明します。

このアプリケーション ノートは、PSoC Creator™ で作成された PSoC 4 SPI F-RAM コンポーネントのような関連プロジェクトを含んでいます。付録 A は、SPI F-RAM コンポーネントを PSoC Creator での新規プロジェクトに統合する方法を説明します。PSoC Creator は、PSoC 3、PSoC 4、PSoC 5LP に対応したサイプレスの強力な統合開発環境 (IDE) です。

SPI F-RAM に関する全ての接続の詳細と推奨事項は、SPI F-RAM メモリにインターフェースする他の標準的な SPI マスター コントローラにも適用可能です。

このアプリケーション ノートは次の議題について説明しています：

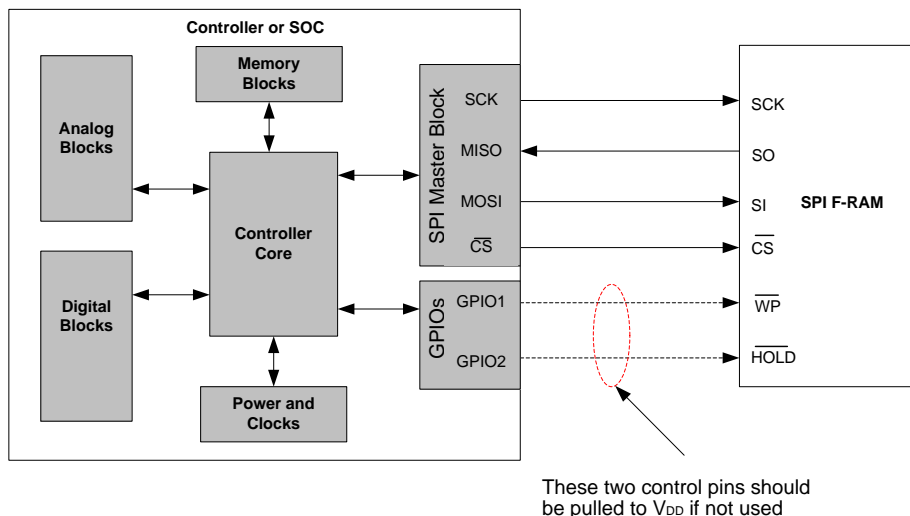
- SPI F-RAM のインターフェース
- SPI 動作モード
- PSoC 4 で SPI マスターの実装
- SPI F-RAM のオペコード
- SPI F-RAM でのアドレス指定
- SPI F-RAM 動作の例

## SPI F-RAM インターフェース

SPI ホスト コントローラ (マスター) と SPI F-RAM (スレーブ) 間のハードウェア接続方法は全てのメモリ領域で同じです。しかし、ファームウェアは SPI デバイスが提供するメモリ容量のオプションと機能に応じていくつかの相違があります。例えば、1M バイト以上のメモリ容量を備えた SPI F-RAM は、3 バイトのアドレス指定を必要とするのに対して、512K バイト以下のメモリ容量では 2 バイトだけのアドレス指定を必要とします。この場合、2 つのメモリ容量オプション間でのファームウェアの変更は、3 バイトか 2 バイトのアドレス指定のいずれかに組み入れるためであり、ハードウェアの接続は変更しません。同様に、シリアル番号読み出し (SNR) 機能は、全ての SPI F-RAM デバイス間で使えません。

図 1 は SPI F-RAM デバイスの一般的なシステム レベルのコンフィギュレーションを図示します。

図 1. コントローラ付きの標準的な SPI F-RAM のインターフェース



## SPI 動作モード

標準的な SPI は、SPI クロック (SCK) 極性 (CPOL) とクロック位相 (CPHA) で決められた 4 つの異なる動作モードに対応しています。SPI クロック (SCK) 極性 (CPOL) は、SPI クロック (LOW または HIGH) の基準値であり、クロック位相 (CPHA) は、立ち上がりか立ち下がりのどちらかの SPI クロック エッジです。SPI マスターは SPI 通信を開始する前に SPI モードを設定します。表 1 は、SPI クロックとデータ駆動、およびマスターアウト、スレーブイン (MOSI) とマスターイン、スレーブアウト (MISO) のライン上でのキャプチャエッジに対して、4 つ全ての SPI モードをまとめたものです。

表 1. SPI 動作モード

SPI データ駆動およびキャプチャエッジ	モード 0 (CPOL=0; CPHA=0)	モード 1 (CPOL=0; CPHA=1)	モード 2 (CPOL=1; CPHA=0)	モード 3 (CPOL=1; CPHA=1)
SPI クロック (SCK) 開始ロジックレベル	LOW	LOW	HIGH	HIGH
MOSI で F-RAM によってラッチされたデータ	SCK 立ち上りエッジ (↑)	SCK 立ち下がりエッジ (↓)	SCK 立ち下がりエッジ (↓)	SCK 立ち上りエッジ (↑)
MISO で F-RAM によって駆動されたデータ	SCK 立ち下がりエッジ (↓)	SCK 立ち上りエッジ (↑)	SCK 立ち上りエッジ (↑)	SCK 立ち下がりエッジ (↓)
SPI F-RAM のサポート	有	無	無	有

## PSoC 4 で SPI マスターの実装

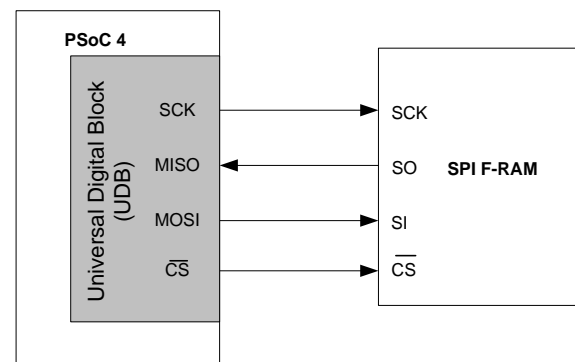
柔軟な自動配線機能付き PSoC 4 のプログラム可能で再設定可能なデジタルブロックは、SPI インターフェースとして使用される 4 つの汎用 I/O (GPIO) を選択できます。PSoC 4 は以下の方法のいずれかを使用して SPI マスターを実装できます:

### UDB を使用して SPI マスターの実装

PSoC 4 の汎用デジタルブロック (UDB) は SPI マスターとして設定できます。UDB は全ての GPIO でアクセスできるため、PSoC 4 の 4 つの GPIO を SPI マスター制御ピンとして使用可能です。I/O ピンのコンフィギュレーションと UDB の詳細については、[PSoC 4 Architecture TRM](#) およびデバイスのデータシートを参照してください。図 2 には、UDB を使って実装された PSoC 4 の SPI マスターと SPI F-RAM のインターフェースを図示します。

添付されているサンプルプロジェクトでは、UDB を使用して PSoC 4 に SPI マスターを実装します。

図 2. PSoC 4 UDB に SPI マスターを実装



### SCB を使用して SPI マスターの実装

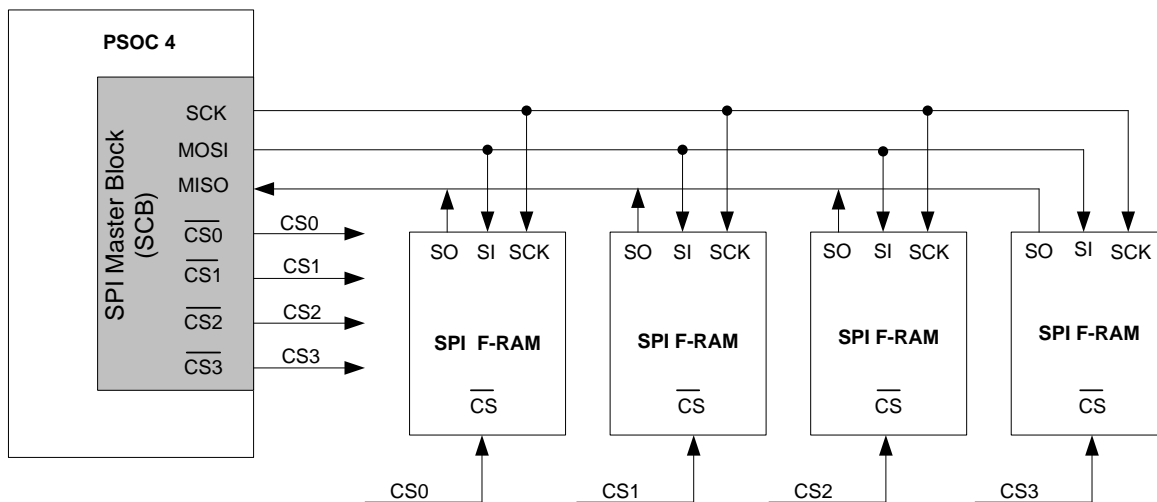
PSoC 4 は、SPI マスターとして設定できる 2 個の専用シリアル通信ブロック (SCB) を備えています。しかし、各 SCB は、特定の制御ピン用の固定 GPIO のセットを割り当て、これらのピンのみが PSoC 4 と周辺機器間のシリアル通信リンクを設定することができます。表 2 は、各 SCB 用に利用可能な専用 PSoC 4 の SPI 制御ピンを示します。

表 2. PSoC 4 の SCB 内の SPI ピンをコンフィギュレーション

PSoC 4 のピン		SCB ブロック	ピンの機能	ピンの説明
ピン名	タイプ			
P4.0	GPIO	SCB0	MOSI	SCB0 のマスター出力、スレーブ入力
P4.1	GPIO	SCB0	MISO	SCB0 のマスター入力、スレーブ出力
P4.2	GPIO	SCB0	SCK	SCB0 の SPI クロック
P4.3	GPIO	SCB0	CS0	SCB0 の SPI スレーブ選択 0
P0.0	GPIO	SCB0	CS1	SCB0 の SPI スレーブ選択 1
P0.1	GPIO	SCB0	CS2	SCB0 の SPI スレーブ選択 2
P0.2	GPIO	SCB0	CS3	SCB0 の SPI スレーブ選択 3
P0.4 / P3.0	GPIO	SCB1	MOSI	SCB1 のマスター出力、スレーブ入力
P0.5 / P3.1	GPIO	SCB1	MISO	SCB1 のマスター入力、スレーブ出力
P0.6 / P3.2	GPIO	SCB1	SCK	SCB1 の SPI クロック
P0.7 / P3.3	GPIO	SCB1	CS0	SCB1 の SPI スレーブ選択 0
P3.4	GPIO	SCB1	CS1	SCB1 の SPI スレーブ選択 1
P3.5	GPIO	SCB1	CS2	SCB1 の SPI スレーブ選択 2
P3.6	GPIO	SCB1	CS3	SCB1 の SPI スレーブ選択 3

各 SCB は 1 個の SPI マスター ブロックを実装できます。SPI マスターは、同じ SPI バスに接続された最大 4 個のスレーブ SPI デバイスで通信することができます。スレーブ選択 ( $\overline{CS}$ ) ピンは、SPI バスに接続された個別の SPI スレーブを選択します。図 3 は SCB を使って設定された PSoC 4 の SPI と SPI の F-RAM とのインターフェースを図示します。この図は同じ SPI バスで 1 個以上の SPI スレーブをインターフェースする方法も示します。

図 3. SCB を使用して PSoC 4 で SPI マスターを実装



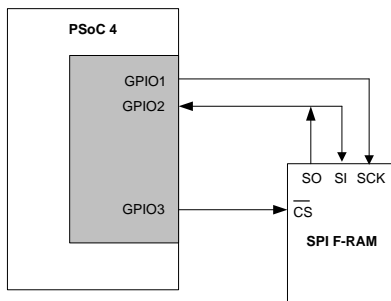
## ビット バンギングを使用した SPI マスターのコンフィギュレーション

三番目で最も少ない実施方法は、I/O のビット バンギングを使用して PSoC 4 内で SPI マスター コントローラを実装することです。ビット バンギング実装では、SPI プロトコルがファームウェアを介して I/O ピンを駆動することで生成されます。ファームウェアの実装は、UDB または SCB のいずれかを使用するハードウェア実装に比べて、コードのオーバーヘッドを大幅に追加します。SPI インターフェースのビット バンギング方法では、合計のデータ スループットを低下させ、CPU リソースを多く使います。そのことで CPU が他の機能用に少ししか使用できなくなってしまう。

ビット バンギング方法の唯一の利点は、4 つの GPIO ピンの代わりに 3 つの GPIO ピンだけを使用して SPI インターフェースを実装できることです。これは SPI インターフェースの半二重型とも呼ばれています。SPI インターフェースの半二重型は、PSoC 4 コントローラが読み書き処理のどちらかを任意の時点で実行するよう要求します。したがって、データが同時に送受信される標準的な SPI インターフェースの場合とは異なり、単一の GPIO はデータの送受信のいずれかに使用されます。

図 4 には、SPI F-RAM をその GPIO を使った PSoC 4 とのインターフェースを図示します。このアプリケーション ノートでは、ビット バンギング方法を使用する PSoC 4 内での SPI マスター コントローラの実装を扱っていません。

図 4. PSoC 4 のビット バンギングを使用する半二重 SPI マスターの実装

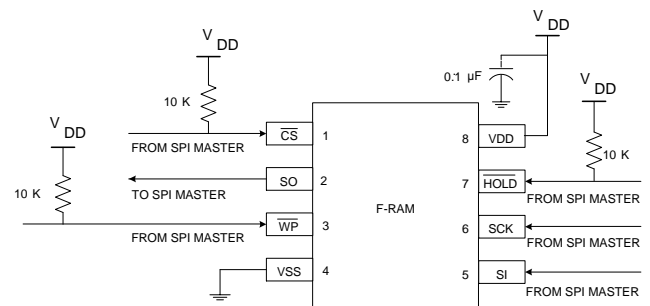


## SPI F-RAM の入力ピンのコンフィギュレーション

SPI F-RAM は、デバイスの正常な動作のために、固定された論理状態 (HIGH または LOW) に適切にバイアスされる必要がある制御入力ピンを備えています。入力ピンが適切にバイアスされずに、フローティングしていると、高いデバイス電流となってしまう中間の論理状態を想定するか、または望ましくない動作をトリガできる論理レベル LOW または HIGH にフロートしています。フローティングの入力信号がその論理状態を想定する方向は、システム内のノイズ、静電容量結合、およびリークなどいくつかの要因に依存します。

このため、フローティングしている入力回路からみえる論理レベルは相対的にランダムであり、動作中に変わることもあります。このような予測不能な入力レベルはデバイスの動作に大幅に影響する可能性があります。そのため、未使用の入力ピンは、アクティブ LOW 入力に対して HIGH などの適当な論理レベルに常に接続する必要があります。10kΩ の抵抗は未使用の入力ピンをプルアップまたはプルダウンするのに使用できます。

図 5. SPI F-RAM と標準的な SPI マスターとのインターフェース



**HOLD ピン:** HOLD ピンはアクティブ LOW 入力であり、ユーザーがクロックの途中で進行中の通信を一時停止させることができます。このピンが LOW の場合、デバイスは受信されたクロック パルスにตอบสนองしなくなり、通信は中断させられ、データは損失または破損する恐れがあります。使用されない場合、このピンは望ましくないイベントを回避するために 10kΩ のプルアップ抵抗に接続する必要があります。

**WP ピン:** 書き込み保護 (WP) ピンはアクティブ LOW 入力であり、このピンを内部で LOW にプルして、ステータス レジスタへの書き込みを防止するのに使用されます。ステータス レジスタの WPEN ビットは、WP ピンの機能を決定します。

WPEN ビットが「1」に設定された場合は、WP ピン制御を有効にします; 「0」にクリアされた場合は、WP ピンを無効にします。このピンは、望ましくないイベントを回避するために 10kΩ のプルアップ抵抗に接続する必要があります。

**CS ピン:** SPI マスターは通常動作中にチップ選択 ( $\overline{CS}$ ) ピンを駆動します。 $\overline{CS}$ 上の外部プルアップはオプションです。しかし、SPI マスターがこのラインを駆動しない場合、 $\overline{CS}$ ピンを HIGH に維持するために、10k $\Omega$  のプルアップ抵抗が使用されます。

電源投入時、電源遮断時および通常動作時の入力制御ピンの機能およびそれらの動作の詳細については、デバイスのデータシートを参照してください。

## SPI F-RAM の動作電圧

SPI F-RAM のデバイスは 2.0V~5.5V の広い範囲の動作電圧 ( $V_{DD}$ ) を備えています。しかし、この広い範囲の動作電圧はすべての F-RAM デバイスには適用していません。最も一般的な F-RAM 動作電圧の範囲は 2.0V~3.6V、2.7V~3.6V、および 4.5V~5.5V です。その  $V_{DD}$ ピンとその I/O に印加されている電圧はデータシートに記載された限界値を超えないことを確認するために、デバイスのデータシートに規定されている  $V_{DD}$  を印可した入力ピン上での許容電圧レベルを参照してください。

## SPI F-RAM のオペコード

全ての SPI オペコード、アドレスおよびデータは 8 ビットのデータ転送です。全てのトランザクションは  $\overline{CSLOW}$  で発生します。オペコード、アドレスおよび入力データは SI ピンでクロック入力され、出力データは SO ピン上でクロック出力されます。オペコードは、デバイスに対する制御を提供します。SPI F-RAM は、全ての読み書き処理用に業界標準のオペコードを提供しています。ユニークなオペコードは、SPI F-RAM の特定の動作ごとに割り当てられます。表 3 はオペコードのリストを示します。表 4 は、各オペコードと正常な動作に必要な関連データ バイトを説明します。

表 3. SPI F-RAM のオペコード

命令カテゴリ	命令名	説明	オペコード	
			16 進数	2 進数
状態レジスタ制御命令	WREN	ステータスレジスタで書き込みイネーブル ラッチ (WEL) を設定	06h	0000 0110b
	WRDI	ステータスレジスタで WEL ビットをクリア; これは全ての書き込み動作を無効にする	04h	0000 0100b
	RDSR	ステータスレジスタの読み出し	05h	0000 0101b
	WRSR	ステータスレジスタの書き込み	01H	0000 0001b
F-RAM 読み出しと書き込み命令	READ (16K バイト以上)	メモリ データの読み出し	03H	0000 0011b
	READ (4Kb) <sup>(1)</sup>	メモリ データの読み出し	03H or 0BH	0000 A011b
	FSTRD <sup>(2)</sup>	メモリ データの高速読み出し	0BH	0000 1011b
	書き込み	メモリ データの書き込み	02H	0000 0010b
	WRITE (4Kb) <sup>(1)</sup>	メモリ データの書き込み	02H or 0AH	0000 A010b
スリープ命令	スリープ	スリープ モードへの移行	B9H	1011 1001b
デバイス ID とシリアル番号の命令	RDID	デバイス ID の読み出し	9FH	1001 1111b
	SNR	シリアル番号の読み出し	C3H	1100 0011b

**注 1** 4K バイトの SPI F-RAM は、読み書き用にシングル バイトのアドレス指定を使用します。アドレス ビットの MSB は、LSB (ビット 3) から 4 番目のビットを使用しているオペコード バイトを介して送信されます。

**注 2** FSTRD コマンドが追加のダミー アドレス サイクルを要することを除き、SPI の FSTRD は SPI の READ コマンドと同様です。FSTRD コマンドは、FSTRD コマンドに対応する SPI フラッシュ メモリのためのドロップイン置換えとして SPI F-RAM を有効にします。SPI F-RAM では、FSTRD と READ コマンドは同じ周波数で動作します。FSTRD コマンドの詳細については、SPI F-RAM のデータシートを参照してください。



表 4. SPI F-RAM のデータフロー

命令名	オペコード	SI でのマスター転送	SO での F-RAM 転送	コメント
WREN	06H	06h	–	このコマンドはステータスレジスタの WEN ビットを設定する。WEL ビットは、この命令後に、チップ選択 ( $\overline{CS}$ ) の立ち上がりエッジで設定
WRDI	04h	04h	–	WRDI コマンドはステータスレジスタの WEL ビット (設定された場合) をクリア
RDSR	05H	05h	StatusReg_Data	RDSR コマンドはステータスレジスタの内容を読み出す
WRSR	01H	01H , StatusReg_Data	–	WEL ビットはステータスレジスタに書き込む前に設定される必要がある。 $\overline{CS}$ が HIGH になると、WEL ビットはクリア
READ	03H	03H, Add1, Add2, Add3	Data1, Data2, Data3, ..., DataN	読み出し処理ではデータ長は 1~N に設定され、N は任意の整数値。READ コマンドを実行すると、F-RAM の内部アドレスは自動的に 1 ずつインクリメントし、デバイスは増分されたメモリアドレスから次のデータバイトを送信する準備をする
FSTRD	0BH	0BH, Add1, Add2, Add3, Dummy_Byte	Data1, Data2, Data3,..., DataN	内部カウンタがその最大の読み出し可能なアドレスに達すると、開始アドレス 00H に戻り、デバイスが読み出しモードで、シリアル (SPI) クロックが利用可能である限り、そこからデータを読み出し続ける。 $\overline{CS}$ が HIGH にトリグすると、読み出しが終了 FSTRD コマンドは追加のダミー アドレス サイクルを必要とする。ダミー サイクルの後に SO を有効なデータで駆動し開始
書き込み	02H	02H, Add1, Add2, Add3, Data1, Data2, Data3,..., DataN	–	WEL ビットは WRITE オペコードを送信する前に設定される必要がある。書き込み処理のためにデータ長は 1~N に設定され、N は任意の整数値。WRITE コマンドが実行すると、F-RAM 内部アドレスは自動的に 1 ずつインクリメントし、デバイスは増分されたメモリ位置に書き込む次のバイトを受信する準備をする 内部カウンタがその最大の書き込み可能なアドレスに達すると、開始アドレス 00H に戻り、以前書き込まれたデータを上書きすることでそこからデータを書き込み続ける。デバイスが書き込みモードであり、シリアル (SPI) クロックが利用可能である限り、データの書き込みを継続する コントローラ ファームウェアは、書き込み処理中にロールオーバーするメモリカウンタによるデータの上書き処理をする必要がある。 $\overline{CS}$ が HIGH にトリグすると、書き込みが終了し、WEL がクリア
スリープ	B9H	B9H	–	$\overline{CS}$ が HIGH になる場合、デバイスはスリープモードに移行し、スリープモードの電流 (IZZ) を消費する。WEL ビットは、SLEEP コマンドを開始する前に設定される必要がある $\overline{CS}$ が HIGH にトリグすると、WEL がクリア
RDID	9FH	9FH	Data1, Data2, Data3,..., Data9	RDID コマンドはデバイス ID を読み出す (9 バイト)
SNR	C3H	C3H	Data1, Data2, Data3,..., Data8	SNR コマンドはシリアル番号を読み出す (8 バイト)

注 1M バイト以上のメモリ容量の SPI F-RAM は 3 バイトのアドレスを使用する; それより少ないメモリ容量の F-RAM (512K バイト以下、最低 16K バイト) は 2 バイトのアドレスを使用します。4K バイトの F-RAM は 1 バイトアドレスのみを使用します。

## SPI F-RAM でのアドレス指定

SPI ホスト コントローラは、バイト単位で SPI F-RAM と通信し、バイト送信の間は常に最初のクロック サイクルで最上位ビット (MSb) を、8 番目のクロック サイクルで最下位ビット (LSb) を送信します。これは、コマンド、アドレス、およびデータ バイトを含むすべての SPI 通信に該当します。

同様に、SPI F-RAM が読み出し処理中にデータ バイトを送信する場合、常に最上位ビットを最初に送信し、最下位ビットを最後に送信します。

図 6 は、SPI マスターで 3 つのアドレス バイトを送信している間に、SPI MOSI ラインを介して送信されているアドレス ビットの例を示します。

最上位バイト (MSB) の未使用ビットは F-RAM が無視する「ドントケア」ビットです。ただし、ファームウェアで未使用のアドレス ビットを「0」にクリアするのは良い方法です。このアプローチは同じソケットで、もっと多いメモリ容量のデバイスに移動する場合、ファームウェアを簡単にアップグレードできるようになります。

図 7 は異なる F-RAM メモリ容量用のアドレス スキームを示します。A0 はアドレスでは LSb になります。

図 6. SPI F-RAM でのアドレス ビット送信

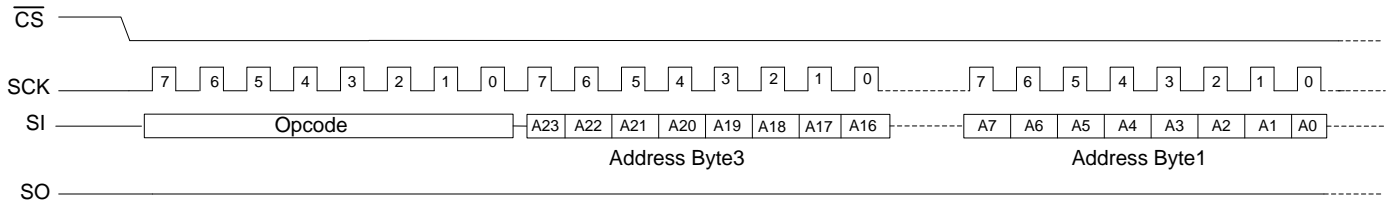


図 7. SPI F-RAM のオペコードとアドレス指定

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4 Kbit <sup>[Note 3]</sup>	op	op	op	op	A	op	op	op	Not Applicable (1 Byte Addressing Only)										A7	A6	A5	A4	A3	A2	A1	A0						
16 Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)							0	0	0	0	0	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
64 Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)							0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
128 Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)							0	0	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
256 Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)							0	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
512 Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)							A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1 Mbit	op	op	op	op	op	op	op	op	0	0	0	0	0	0	0	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
2 Mbit	op	op	op	op	op	op	op	op	0	0	0	0	0	0	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
4 Mbit	op	op	op	op	op	op	op	op	0	0	0	0	0	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

注 4K バイトの SPI F-RAM は 1 バイト アドレスのみを使用します。最上位のアドレス ビット A8 (9 番目のビット) は READ と WRITE オペコードの一部です。READ オペコードと WRITE オペコードのビット 3 はメモリ アドレスの MSb として使用されます。

## メモリ容量のアップグレード

SPI 規格は、シングル マスターのマルチ スレーブ接続形態に対応しています。これにより、同じ SPI バスに 1 つ以上の SPI デバイスを接続できるようになります。SPI スレーブと通信するために、SPI マスターは最初にそのスレーブ選択 (CS) ピンを LOW にプルすることで SPI スレーブを選択して、次に SPI コマンドを送信します。1 つ以上の SPI スレーブをサポートするために、マスターは各 SPI スレーブ専用の 1 個のスレーブ選択制御ピンを持つ必要があります。図 3 は、同じ SPI マスターに接続する複数の SPI スレーブを示します。この接続形態は 1 つ以上の SPI スレーブを制御するのに使用されます。例えば、複数の SPI メモリ デバイスを用いてシステム メモリの容量を拡張します。

## SPI F-RAM 動作の例

この節では、タイミング図と PSoC 4 特有の擬似コードを使用して F-RAM の動作について説明します。プレフィックス NVRAM\_SPI\_1 で始まるすべての関数は、PSoC 4 特有の関数です。NVRAM\_SPI\_1 は、添付されている PSoC Creator サンプル プロジェクトでインスタンス化された SPI F-RAM コンポーネントの名前です。

この節では、SPI マスターと SPI F-RAM 間の SPI 通信中の SPI F-RAM のデータ フローを示すサンプルとして幾つかのオペコードのみを扱っています。オペコードの詳細については、SPI F-RAM のデータシートを参照してください。



## ステータス レジスタの動作

この節では、タイミング図と PSoC 4 のコードをサンプルとして使用して F-RAM のステータス レジスタの読み書き処理を説明します。

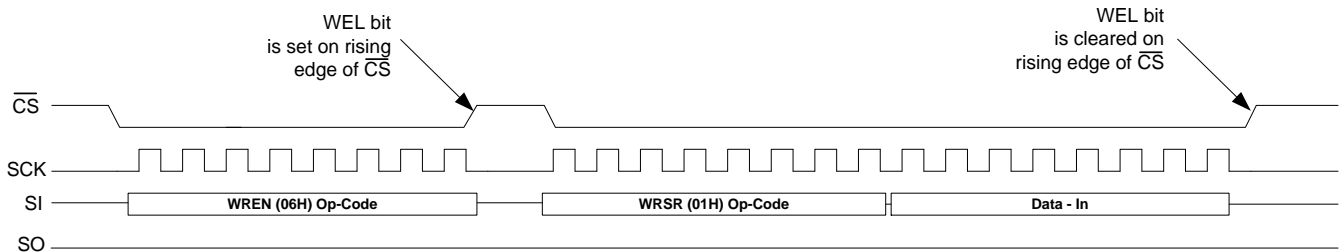
### ステータスレジスタの書き込み

WRSR コマンドは、ステータス レジスタでユーザーが書き込み可能なビットを設定するために使用されます。他の書き込み不可のビットは、内部的にデバイスで設定されるか、またはそれらビットが予約され、固定値 (「0」か「1」) を返します。ステータス レジスタの詳細については、デバイスのデータシートを参照してください。

F-RAM ステータス レジスタで書き込みを開始するには以下のシーケンスが必要です:

- 書き込みイネーブル ラッチ (WEL) ビットを設定するために、WREN オペコードを送信します。
- ステータス レジスタへ書き込まれるデータ バイトに続く書き込みステータス レジスタのオペコード (WRSR) を送信します。ステータス レジスタの読み出し専用ビットが WRSR 動作によって影響を受けないことに注意してください。ステータス レジスタの詳細については、デバイスのデータシートを参照してください。
- 図 8 は、ステータス レジスタへ書き込むタイミング図を示します。

図 8. ステータス レジスタへの書き込み (WRSR オペコード)



```

/*****PSoC 4 pseudocode for Status Register write*****/

void NVRAM_SPI_1_Status_Reg_Write ( uint8 data_byte )
{
    NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW
    NVRAM_SPI_1_SPIM_ClearTxBuffer(); //Clear SPI transmit buffer before sending command
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WREN); // Set the write enable (WEL) bit prior to write

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) // Wait till SPI_DONE flag is
        != NVRAM_SPI_1_SPIM_STS_SPI_DONE); // cleared
    NVRAM_SPI_1_CS_Reg_Write(1); // WEL is set high when CS is switched high
    NVRAM_SPI_1_CS_Reg_Write(0); // Re-enable the SPI slave
    NVRAM_SPI_1_SPIM_ClearTxBuffer();
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WRSR_CMD); //Send Write Status Register instruction
    NVRAM_SPI_1_SPIM_WriteTxData(data_byte); //Send data

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);
    NVRAM_SPI_1_CS_Reg_Write(1); // Terminate the write operation by toggling chip select HIGH
}

```

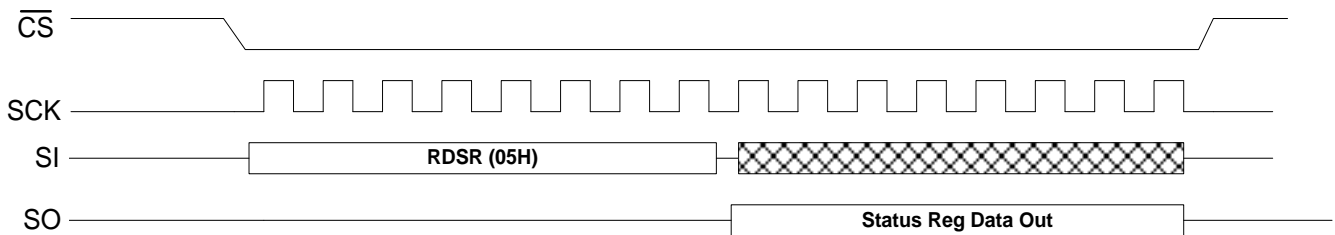
### ステータス レジスタの読み出し

RDSR コマンドはステータス レジスタの現在の値を読み出します。例えば、WREN コマンドに続く RDSR コマンドを実行すると、書き込みイネーブル ラッチ (WEL) ビットが「1」に設定され、ステータス レジスタの値を返します。

F-RAM ステータス レジスタから読み出しを開始するには以下のシーケンスが必要です:

- RDSR オペコードを送信します。
- SO ライン上でステータス レジスタの値を読み出します。
- ホストは、チップ選択を HIGH にトグルすることでステータス レジスタのコマンドを終了できます。図 9 は SPI F-RAM ステータス レジスタから読み出すタイミング図を示します。

図 9. ステータス レジスタからの読み出し (RDSR オペコード)



```

/***** PSoC 4 pseudocode for Status Register read *****/

uint8 NVRAM_SPI_1_Status_Reg_Read ( void )
{
    uint8 data_byte;

    NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW
    NVRAM_SPI_1_SPIM_ClearTxBuffer(); //Clear SPI transmit buffer before sending command
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_RDSR_CMD); //Send read status register command

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) // Wait till SPI_DONE flag is
        != NVRAM_SPI_1_SPIM_STS_SPI_DONE); // cleared

    NVRAM_SPI_1_SPIM_ClearRxBuffer();
    NVRAM_SPI_1_SPIM_WriteTxData(0x00); //Dummy write for reading the status register data byte

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);
    while(!NVRAM_SPI_1_SPIM_GetRxBufferSize()); //Wait until there is data in the read buffer
    data_byte = NVRAM_SPI_1_SPIM_ReadRxData();
    NVRAM_SPI_1_CS_Reg_Write(1); // Terminate the read operation by toggling chip select HIGH

    return(data_byte);
}

```

### F-RAM の読み書き処理

この節では、タイミング図と PSoC 4 のコードをサンプルとして使用して F-RAM の読み書き処理を説明します。

#### F-RAM の書き込み処理

F-RAM への書き込みを開始するには以下のシーケンスが必要です:

- 書き込みイネーブル ラッチ (WEL) ビットを設定するために、WREN オペコードを送信します。
- 書き込みオペコードを送信します。
- 最上位アドレス バイトを送信します。
- 中間アドレス バイト (3 バイトのアドレス指定) を送信します。
- 最上位アドレス バイトを送信します。
- データ バイト/バイトを送信します。

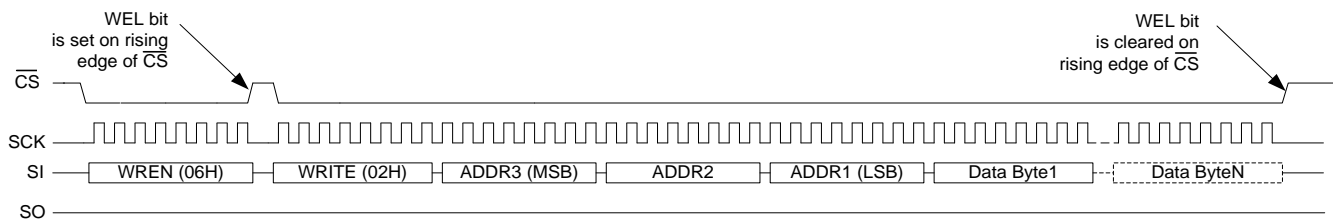
F-RAM への任意の書き込みコマンドは、書き込みイネーブル (WREN) 命令が先行する必要があります。デバイスは書き込みイネーブルではない場合 (WEL = 「0」) は書き込み命令を無視します。新しいCSの立ち上がりエッジが、SPI 通信を再開させるために必要です。

書き込み命令 (WRSR または WRITE) が完了した後、ステータス レジスタの WEL ビットはチップ選択 ( $\overline{CS}$ ) の立ち上がり

エッジで「0」にクリアされます。これにより、SPI F-RAM は書き込みモードを終了し、意図しない書き込みを防止できます。

ステータス レジスタ (RDSR オペコード) の読み出しが WEL ビットをクリアしないことに注意してください。一部のユーザーは WREN を実行した直後に、ステータス レジスタを読み出して書き込み処理を開始する前に WEL ビットが設定されているかどうかを確認しています。図 10 は F-RAM へ書き込むタイミング図を示します。

図 10. F-RAM への書き込み (書き込みオペコード)



/\* PSoC 4 pseudocode for 1 Mb (128kx8) F-RAM write in burst mode. By passing in total\_data\_count =1, the user can write 1 byte at a given address location\*/

```
void NVRAM_SPI_1_Write ( uint32 addr, uint8 *data_write_ptr, uint32 total_data_count )
{
    uint32 i;

    NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW
    NVRAM_SPI_1_SPIM_ClearTxBuffer(); // Clear SPI transmit buffer
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WREN); // Set the write enable (WEN) bit prior to write

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) // Wait till SPI_DONE flag is
        != NVRAM_SPI_1_SPIM_STS_SPI_DONE); // cleared

    NVRAM_SPI_1_CS_Reg_Write(1); // WEL is set here
    NVRAM_SPI_1_CS_Reg_Write(0);
    NVRAM_SPI_1_SPIM_ClearTxBuffer();
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_WRITE_CMD); //Send memory write command

    if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
    {
        NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16)); //Transmits most significant address byte (1 Mb and above)
    }
    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8)); // Transmits intermediate address byte in 3 byte addressing,
        //or, Transmits most significant address byte in 2 byte addressing
    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr)); // Transmits least significant address byte in 2/3 byte addressing,
    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
        !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);

    for(i = 0; i < total_data_count; i++ )
    {
        NVRAM_SPI_1_SPIM_WriteTxData ((uint8) (data_write_ptr[i]));
        while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
            != NVRAM_SPI_1_SPIM_STS_SPI_DONE);
    }
    NVRAM_SPI_1_CS_Reg_Write(1); // Terminate the write operation by toggling chip select HIGH
}
```

## F-RAM の読み出し処理

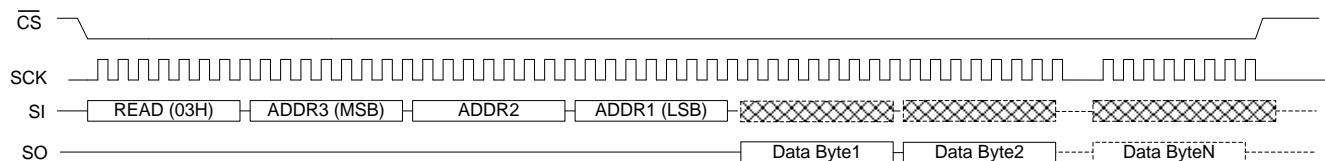
F-RAM からの読み出しを開始するには以下のシーケンスが必要です:

- 読み出しオペコードを送信します。
- アドレス サイクル中に、最上位アドレス バイトを最初に送信し、最下位アドレス バイトを最後に送信します。
- SPI F-RAM デバイスが読み出しコマンドを受信すると、SO ライン上でデータ バイトを送信し始めます。ホスト コントローラは、シングル バイトの読み出しまたはバースト読み出し (1 バイト以上) のいずれかを開始することができます。

バースト読み出しは、単一の読み出しコマンドを開始することで連続するメモリ位置を読み出すことができます。バースト読み出しでは、F-RAM デバイスは、内部アドレス カウンタを自動的にインクリメントし、SO ライン上にデータ バイトを送信し続けます。チップ選択  $\overline{CS}$  が LOW にアサートされたままで、SPI クロックが存在する限りこれは継続します。バースト読み出しは循環方式でメモリを介してサイクルし続けます。

チップ選択  $\overline{CS}$  がデアサートされると、データ出力が停止し、SO が高インピーダンス (Hi-Z) 状態に移行します。図 11 は F-RAM から読み出すタイミング図を示します。

図 11. F-RAM からの読み出し (読み出しオペコード)



/\* PSoC 4 pseudocode for 1 Mb (128kx8) F-RAM Read in burst mode. By passing in total\_data\_count =1, user can read only 1 byte from a given address location\*/

```
void NVRAM_SPI_1_Read ( uint32 addr, uint8 *data_read_ptr, uint32 total_data_count )
{
    uint32 i;

    NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW
    NVRAM_SPI_1_SPIM_ClearTxBuffer(); // Clear SPI transmit buffer
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_READ_CMD); // Send memory read command

    if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
    {
        NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16)); //Transmits most significant address byte (1 Mb and above)
    }
    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8)); // Transmits intermediate address byte in 3 byte addressing,
    //or, Transmits most significant address byte in 2 byte addressing
    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr)); // Transmits least significant address byte in 2/3 byte addressing,

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
        !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
    for(i = 0; i < total_data_count; i++ )
    {
        NVRAM_SPI_1_SPIM_ClearRxBuffer();
        NVRAM_SPI_1_SPIM_WriteTxData((uint8) 0x00); //dummy write for reading the F-RAM memory data byte
        while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
            !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
        while(!NVRAM_SPI_1_SPIM_GetRxBufferSize());
        data_read_ptr[i] = NVRAM_SPI_1_SPIM_ReadRxData();
    }
    NVRAM_SPI_1_CS_Reg_Write(1); // Terminate the read operation by toggling chip select HIGH
}
```

## F-RAM の高速読み出し処理

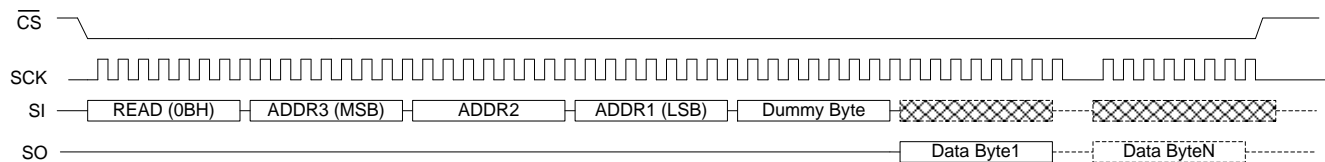
F-RAM から高速読み出しを開始するには以下のシーケンスが必要です:

- FSTRD オペコードを送信します。
- アドレス サイクル中に、最上位アドレス バイトを最初に送信し、最下位アドレス バイトを最後に送信します。
- ダミー アドレス バイトを送信します。
- SPI F-RAM デバイスが FSTRD コマンドを受信すると、SO ライン上にデータ バイトを送信し始めます。ホスト コントローラはシングル バイトの読み出し、またはバースト読み出し (1 バイト以上) のいずれかを開始することができます。

バースト読み出しは、単一の読み出しコマンドを開始することで連続するメモリ位置を読み出すことができます。バースト読み出しでは、F-RAM デバイスが内部アドレス カウンタを自動的にインクリメントし、SO ライン上にデータ バイトを送信し続けます。チップ選択  $\overline{CS}$  が LOW にアサートされたままで、SPI クロックが存在する限りこれは継続します。バースト読み出しは循環方式でメモリを介してサイクルし続けます。

チップ選択  $\overline{CS}$  がデアサートされると、データ出力が停止し、SO が高インピーダンス (HI-Z) 状態に移行します。図 12 は、高速読み出し (FSTRD) コマンドを使用して F-RAM から読み出すタイミング図を示します。

図 12. F-RAM からの高速読み出し (FSTRD オペコード)





```

/* PSoC 4 pseudocode for 1 Mb (128kx8) F-RAM fast read in burst mode. By passing in total_data_count =1, the user
can read only 1 byte from a given address location*/

void NVRAM_SPI_1_FastRead ( uint32 addr, uint8 *data_read_ptr, uint32 total_data_count )
{
    uint32 i;

    NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW
    NVRAM_SPI_1_SPIM_ClearTxBuffer();// Clear SPI transmit buffer
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_READ_CMD); // Send memory read command

    if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
    {
        NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16)); //Transmits most significant address byte (1 Mb and above)
    }
    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8)); // Transmits intermediate address byte in 3 byte addressing,
                                                    //or, Transmits most significant address byte in 2 byte addressing
    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr)); // Transmits least significant address byte in 2/3 byte addressing,

    NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr)); // Transmits least significant address byte as dummy byte for fast
                                                    // read operation

    while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
    !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
    for(i = 0; i < total_data_count; i++)
    {
        NVRAM_SPI_1_SPIM_ClearRxBuffer();
        NVRAM_SPI_1_SPIM_WriteTxData((uint8) 0x00); //dummy write for reading the F-RAM memory data byte
        while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
        !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
        while(!NVRAM_SPI_1_SPIM_GetRxBufferSize());
        data_read_ptr[i] = NVRAM_SPI_1_SPIM_ReadRxData();
    }
    NVRAM_SPI_1_CS_Reg_Write(1); // Terminate the read operation by toggling chip select HIGH
}

```

## まとめ

サイプレスの SPI F-RAM は、SPI EEPROM、フラッシュおよび MRAM などの他の不揮発性 SPI メモリ製品と同様に、業界標準の SPI アクセス プロトコルに対応しています。これにより、全ての標準的な SPI マスター コントローラと F-RAM の互換性が高まります。SPI F-RAM のオペコードが標準的な SPI メモリ製品とマッチしているため、SPI F-RAM は簡単なドロップイン代替品になります。このアプリケーション ノートでは回路図、タイミング図およびコード例を使用するサイプレスの PSoC 4 のコントローラ付きの SPI F-RAM をインターフェースする方法を説明しました。

## 付録 A: PSoC 4 のサンプル プロジェクト

このアプリケーション ノートと共に提供されるプロジェクトは、PSoC 4 デバイスで SPI F-RAM のインターフェースを説明するサンプル プロジェクトです。AN89659.zip のアプリケーション ノートからプロジェクト ファイルをダウンロードできます。PSoC 4 のプログラム可能性と柔軟性により、目標のアプリケーションに応じて、より多くの機能を追加したりプロジェクトを変更したりできます。ハードウェアの設定時にサンプル プロジェクトを実行するには、SPI F-RAM に接続した PSoC 4 を持つ必要があります。

### サンプル プロジェクトのピン配置

表 5 は、添付されているサンプル プロジェクトにおける接続の詳細を示します。サンプル プロジェクト用に以下のソフトウェアとハードウェアのコンポーネントが使用されます:

- PSoC Creator 3.0 のコンポーネント パック 7
- ハードウェア キット – CY8CKIT-042
- PSoC 4 の製品番号 CY8C4245AXI-483 がプロジェクトを構築するのに選択されます。
- $V_{DD}$  が USB 経由で PSoC 4 コントローラに電源供給します。

表 5. サンプル プロジェクトでの PSoC 4 のポート コンフィギュレーション

F-RAM の信号名	PSoC (マスター) 信号名	PSoC 4 の I/O 割り当て	信号方向
$\overline{CS}$	CS	P2[0]	PSoC 4 の出力
SI	MOSI	P2[3]	PSoC 4 の出力
SO	MISO	P1[4]	PSoC 4 の入力
SCK	SCK	P2[2]	PSoC 4 の出力
$\overline{HOLD}$	HOLD	P2[1]	PSoC 4 の出力

サンプル プロジェクトは以下のことを行います:

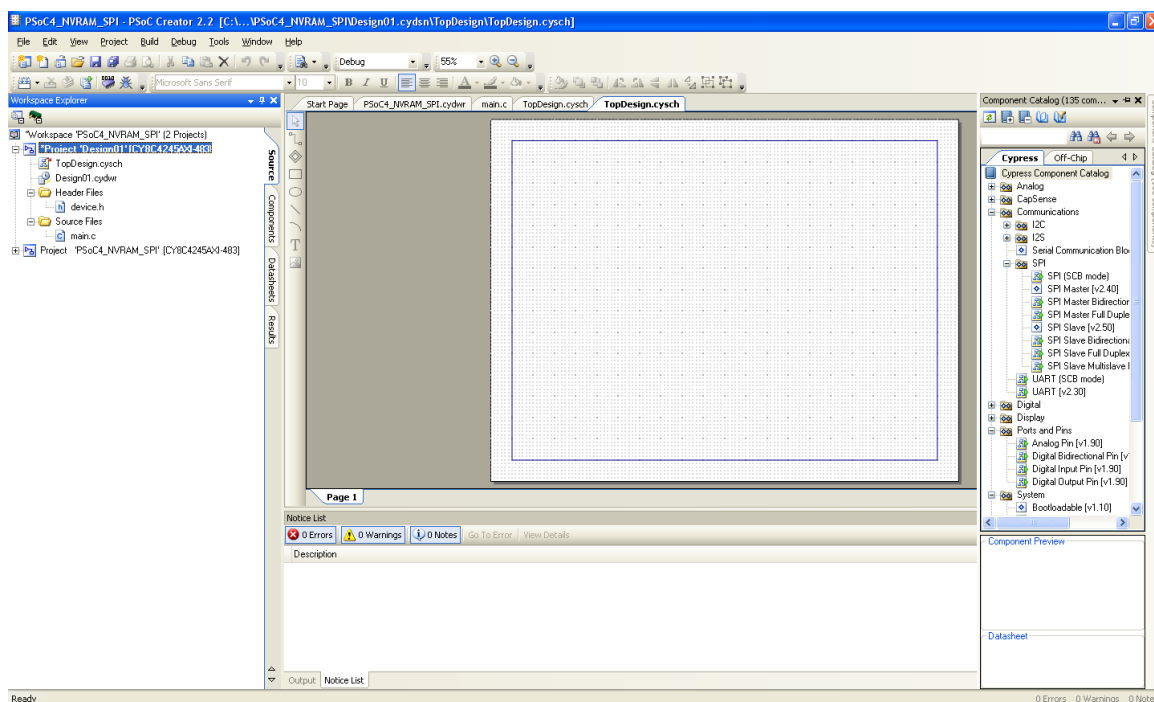
1. 4 バイトのデータをメモリに書き込み、書き込まれた 4 バイトのデータを読み戻します。データが内部レジスタに格納されます。
2. 1 バイトのデータをステータス レジスタに書き込み、その書き込まれたデータを読み戻します。読み出しデータ バイトが内部レジスタに格納されます。

### SPI F-RAM コンポーネントをプロジェクトに統合

この節では、F-RAM コンポーネントを PSoC 4 プロジェクトに組み込む方法について説明します。圧縮された AN89659.zip ファイルを展開すると、「PSoC4\_NVRAM\_SPI」のフォルダ名が作成されます。このフォルダはサンプル プロジェクトと NVRAM\_SPI コンポーネントを含みます。NVRAM\_SPI コンポーネントを PSoC 4 の設計に組み込んで使用する方法を以下の手順で説明します。

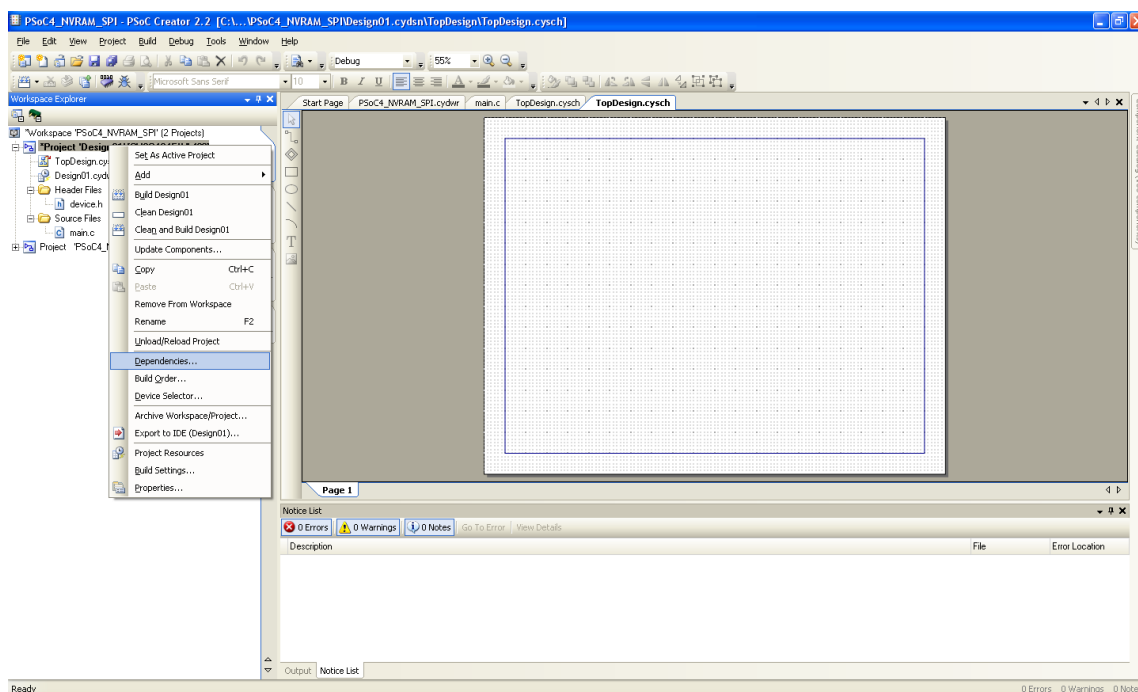
1. PSoC Creator を開き、**File > New > Project** を選択し、新規プロジェクトを作成します。「Empty Template」の次に「Empty PSoC 4 Design」を選択します。「Name」のフィールドにプロジェクトの名前を記入します。図 13 に示すように、「Design01」の新規プロジェクト (ワークスペース) が作成されます。

図 13. 「Design01」プロジェクトを作成



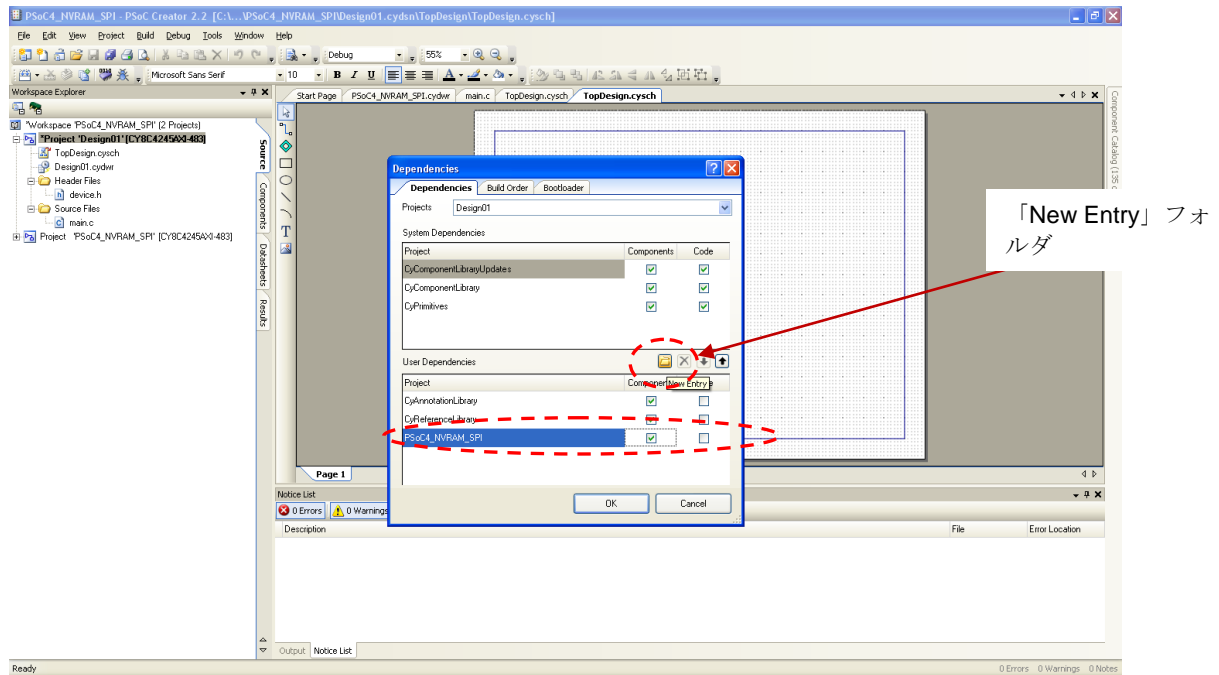
- プロジェクトを右クリックして、**Workspace Explorer** で **Dependencies** タブを選択します。図 14 に示すように、NVRAM\_SPI コンポーネントをユーザーの設計に組み込みます。

図 14. 「Dependencies」を開き



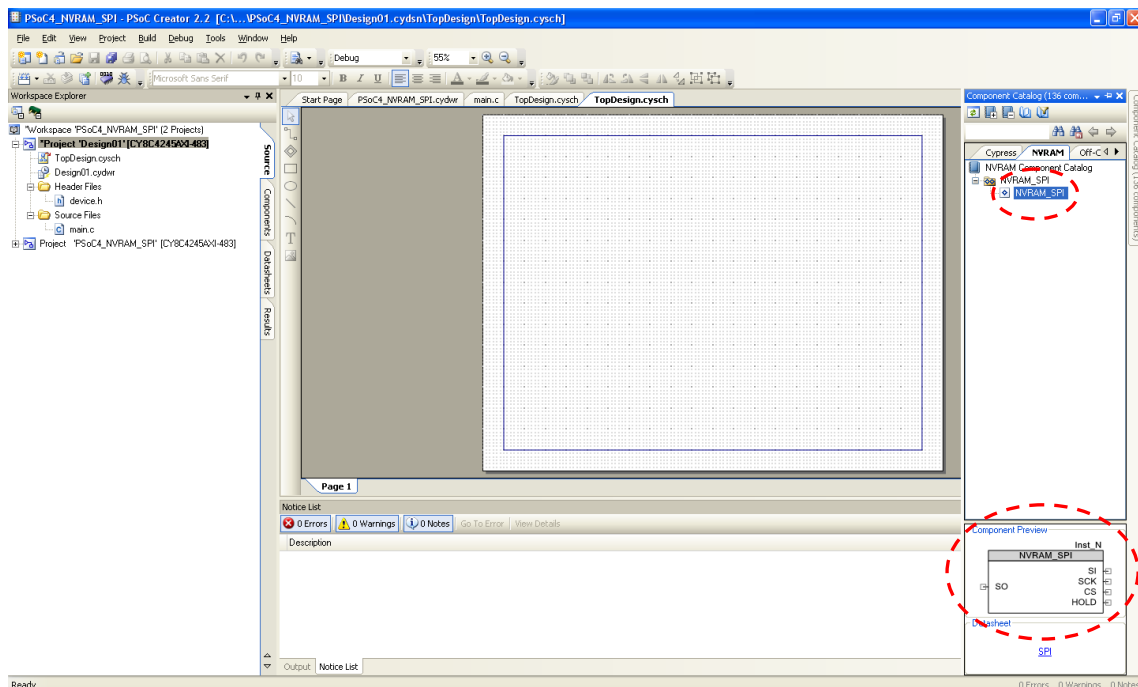
3. 図 15 に示すように、**New Entry** (User Dependencies) をクリックして、**PSoC4\_NVRAM\_SPI.cydsn** フォルダから **PSoC4\_NVRAM\_SPI.cypri** を選択します。

図 15. 「Dependencies」を追加



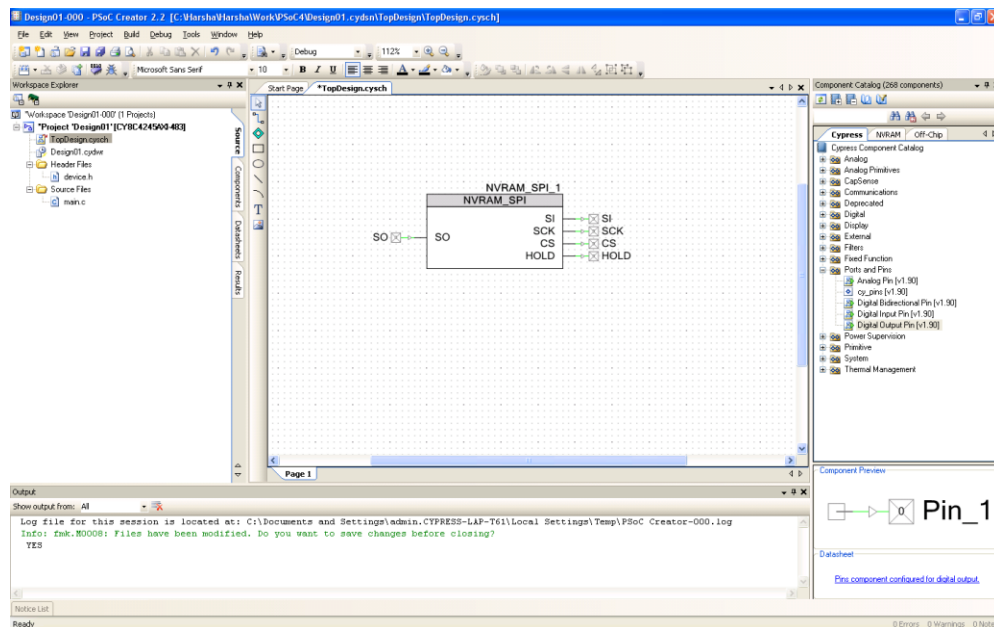
4. 図 16 に示すように、NVRAM\_SPI コンポーネントは **NVRAM** タブと「NVRAM\_Component Catalog」の下に表示します。

図 16. カタログの NVRAM\_SPI のコンポーネント



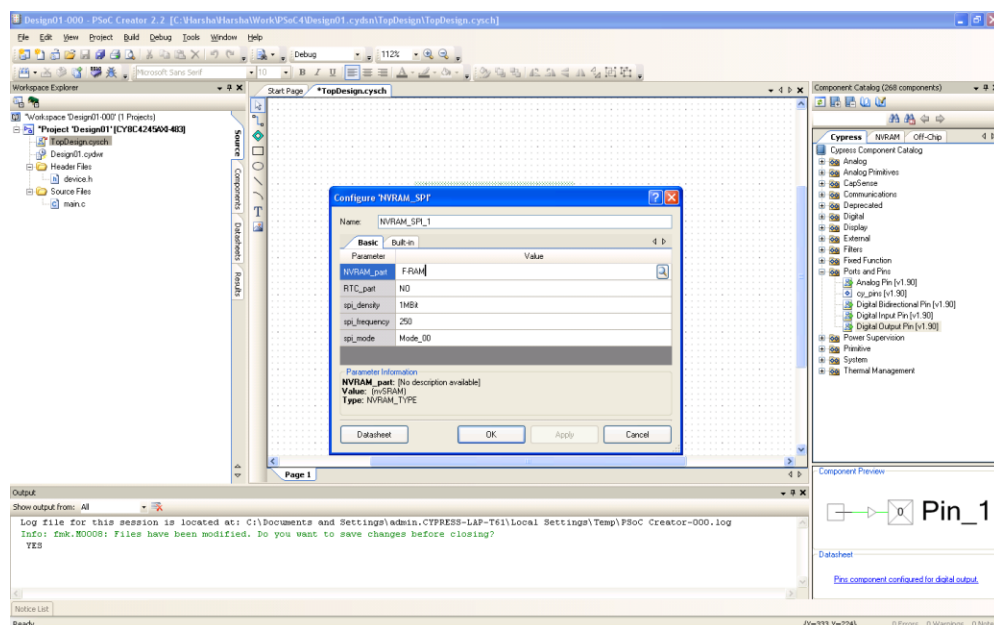
- NVRAM\_SPI コンポーネントを *TopDesign.cysch* にドラッグ&ドロップして、デジタル I/O を割り当てます。コンポーネント入力／出力ピンを PSoC 4 の適切なポート ピンに接続します。表 5 はサンプル プロジェクトでのピン配置を識別します。

図 17. NVRAM\_SPI コンポーネントを使用して設計回路図を作成



- NVRAM\_SPI コンポーネントをダブルクリックして、コンポーネントのパラメータを次の通りに設定します。NVRAM\_Part を F-RAM として、RTC を NO として、およびアプリケーションで使用される F-RAM 容量当たりの SPI 容量を選択します。次に、図 18 に示すように、SPI 周波数とモード (モード 0 かモード 3) をアプリケーションの要求に従って設定します。

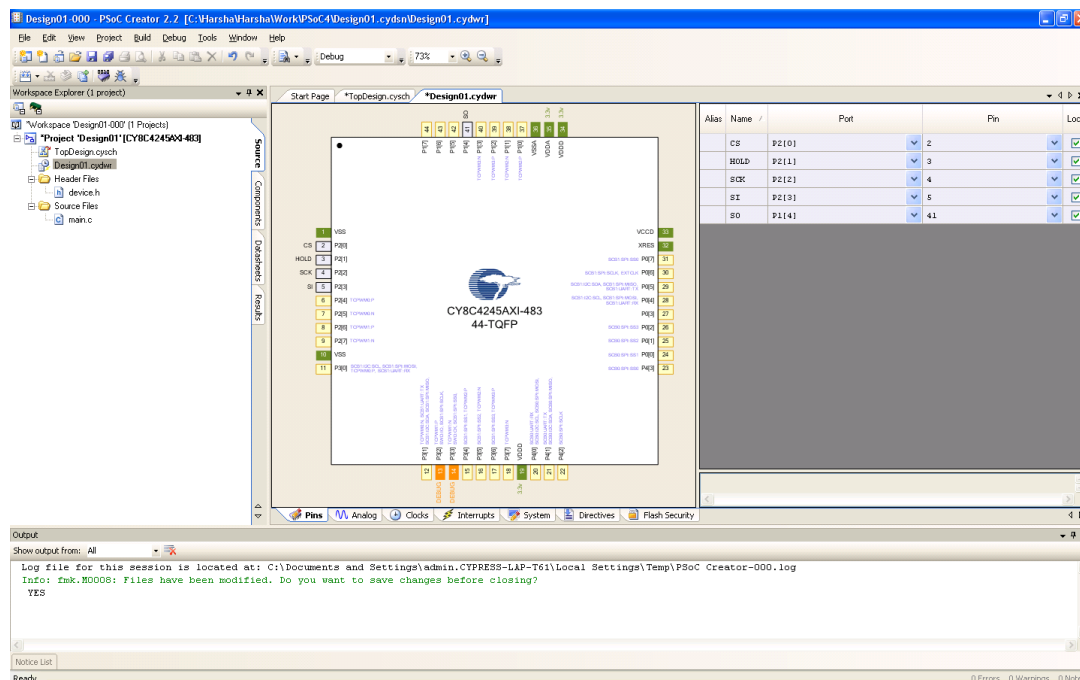
図 18. NVRAM\_SPI コンポーネントのパラメータを設定





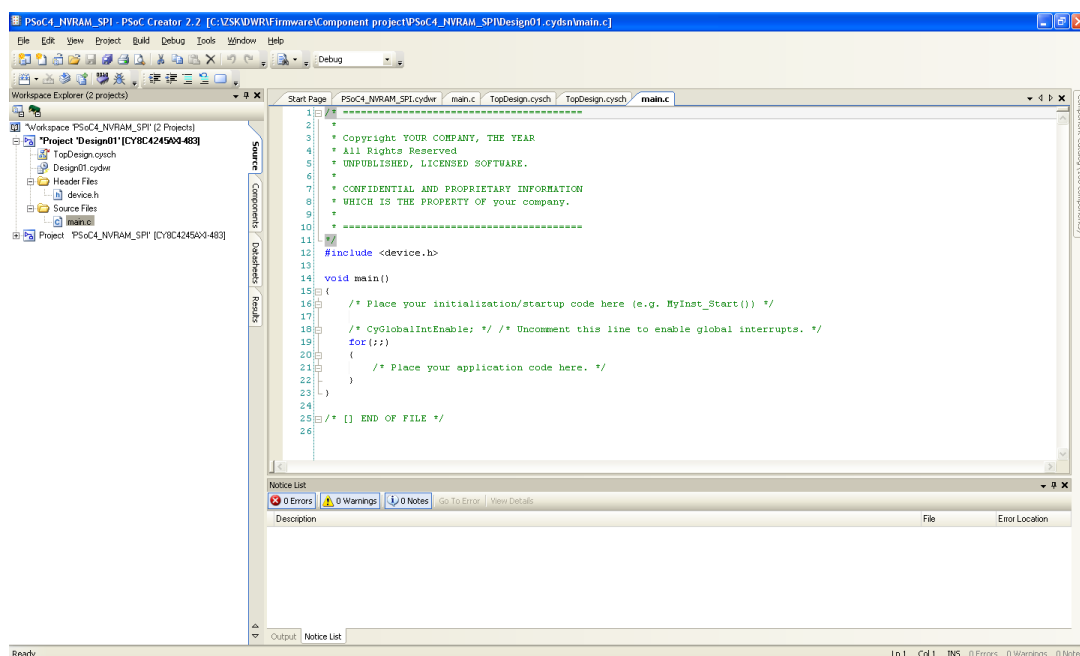
7. 図 19 に示すように、ユーザーの設計に応じて適切な入力／出力ピンを割り当て、プロジェクトを構築します。

図 19. PSoC 4 のピン配置



*main.c* ファイル内でコードを展開します。API をユーザーのプログラムに直接呼び出し、F-RAM 機能を実行できます。

図 20. プロジェクトの *main.c* ファイル



PSoC 4 と PSoC Creator の詳細については、[AN79953 – Getting Started with PSoC<sup>®</sup> 4](#) および関連するリンクを参照してください。

## 改訂履歴

文書名: SPI F-RAM を PSoC® 4 でインターフェース – AN89659

文書番号: 001-92728

版	ECN	変更者	提出日	変更内容
**	4395700	HZEN	06/20/2014	これは英語版 001-89659 Rev. **を翻訳した日本語版 Rev. **です。

## ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューションセンター、メーカー代理店および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、[サイプレスのロケーションページ](#)をご覧ください。

## 製品

車載用	<a href="http://cypress.com/go/automotive">cypress.com/go/automotive</a>
クロック & バッファ	<a href="http://cypress.com/go/clocks">cypress.com/go/clocks</a>
インターフェース	<a href="http://cypress.com/go/interface">cypress.com/go/interface</a>
照明 & 電源管理	<a href="http://cypress.com/go/powerpsoc">cypress.com/go/powerpsoc</a> <a href="http://cypress.com/go/plc">cypress.com/go/plc</a>
メモリ	<a href="http://cypress.com/go/memory">cypress.com/go/memory</a>
PSoC	<a href="http://cypress.com/go/psoc">cypress.com/go/psoc</a>
タッチセンシング	<a href="http://cypress.com/go/touch">cypress.com/go/touch</a>
USB コントローラ	<a href="http://cypress.com/go/usb">cypress.com/go/usb</a>
ワイヤレス / RF	<a href="http://cypress.com/go/wireless">cypress.com/go/wireless</a>

## PSoC®ソリューション

[psoc.cypress.com/solutions](http://psoc.cypress.com/solutions)  
PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

## サイプレス開発者コミュニティ

[コミュニティ](#) | [フォーラム](#) | [ブログ](#) | [ビデオ](#) | [トレーニング](#)

## テクニカルサポート

[cypress.com/go/support](http://cypress.com/go/support)

PSoC はサイプレス セミコンダクタ社の登録商標であり、PSoC Creator はサイプレス セミコンダクタ社の商標です。本書で言及するその他すべての商標または登録商標は、各社の所有物です。



Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709

電話番号 : 408-943-2600  
ファックス : 408-943-4730  
ウェブサイト: [www.cypress.com](http://www.cypress.com)

© Cypress Semiconductor Corporation, 2013-2014. 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 (サイプレス セミコンダクタ社) 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一切の責任を負いません。サイプレス セミコンダクタ社は、特許またはその他の権利に基づくライセンスを譲渡すること、または含意することはありません。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、または安全の用途のために使用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ト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。

このソースコード (ソフトウェアおよび/またはファームウェア) はサイプレス セミコンダクタ社 (以下「サイプレス」) が所有し、全世界の特許権保護 (米国およびその他の国)、米国の著作権法なら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り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り、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および/またはカスタム 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コピー、使用、変更して作成するためのライセンス、ならびにサイプレスのソースコードおよ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、または表示することは全て禁止します。

免責事項: サイプレスは、明示的または黙示的を問わず、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、商品性また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または回路を適用また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ト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。