AN304

F-RAM™向けの SPI ガイド

作成者: Harsha Medu
関連プロジェクト: CE204087
関連製品ファミリ: FM25xxx
ソフトウェア バージョン: なし
関連アプリケーションノート: ここをクリック

AN304 は、SPI F-RAM の機能説明、タイミング、およびサンプル コードを提供します。また、SPI F-RAM 用の PSoC 3 ベースのユーザー モジュールも含まれます。

目次

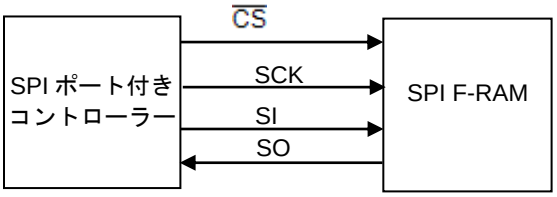
はじめに

FM25xxx F-RAM 製品ファミリは、業界標準の 4 線式 SPI インターフェースを備えています。これらは、高速 (最大 40MHz)、低消費電力、不揮発性メモリ デバイスです。SPI F-RAM の容量は 4K ビット～4M ビットです。本アプリケーション ノートは、これらデバイスの機能とタイミングについて説明します。

SPI の用途

シリアル ペリフェラル インターフェース (SPI) は、Motorola 社 (現、Freescale 社) によって開発されたシリアル バスであり、それら会社および Cypress、TI、Atmel、Microchip、Analog Devices などその他の半導体サプライヤの MCU 上の専用インターフェースとして備えられています。SPI ポートは DSP、ネットワーク プロセッサー、FPGA などでも備えられています。SPI は、高いシリアル データ レートを必要とする、マイクロコントローラー ベースのシステムには最適です。シリアル データ スループットはシリアル クロック速度 (図 1 の SCK 信号) との相互関係があります。サイプレスの全てのシリアル F-RAM は 40MHz までクロック可能です。SPI ポートを備えていない幾つかのマイクロコントローラーは、ビット バンギングを使用して GPIO ピンを SPI 動作用に使用することができます。この方法には、I/O ポートを制御または「バンギング」を行うソフトウェアが必要です。図 1 には、基本的な SPI インターフェースを示します。

図 1. 基本的な SPI インターフェース

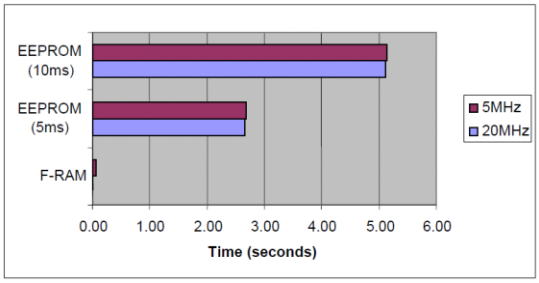


速度の利点

F-RAM メモリ技術により、等価の EEPROM かフラッシュよりはるかに大きなデータ ブロックを速く書き込むことができます。EEPROM かフラッシュとは違って、F-RAM デバイスはページ バッファを使用しません。F-RAM は、受信した各バイトの 8 番ビットの直後に各データ バイトを書き込みます。書き込み遅延なしと高いクロック速度の組み合わせにより、F-RAM は多くのデータ量を速く書き込む必要のあるアプリケーションに説得力のある選択になります。設計者は、SPI F-RAM に書き込むバイト数を自由に決定することができます。F-RAM 内のランダムな位置に 1 個か 2 個のバイトを書き込む際、書き込みサイクルは約 1µs であるー方、EEPROM またはフラッシュの書き込みサイクルは 5ms～10ms もかかります。また、システムが次世代メモリの容量の変化を必要としてもページ バッファ サイズの変化について心配することはありません。

図 2 は、F-RAM と EEPROM に対応する 256K ビット アレイへ書き込むのに必要な時間の比較を示すグラフです。同じクロック レートでは、F-RAM デバイスは、64 バイト ページ バッファを備えた EEPROM よりも桁違いに速いです。これは、システム設定をプログラムする時間が限られている生産ラインには特に重要です。

図 2. 256K ビット SPI メモリ アレイへの書き込みに必要な時間



256K ビット EEPROM メモリへの書き込みに必要な時間は、クロック周波数を 5MHz から 20MHz に増加することで大幅に改善されないことに注意してください。これは、各ページ書き込みに必要な長い書き込み遅延が支配的だからです。20MHz F-RAM メモリでは、32K バイト アレイ全体はわずか 13ms で書き込まれます。この時間は小さい値であり、図 2 のグラフに表示されません。

SPI バス

図 1 に示されるように、SPI インターフェースは 4 本のピンを持っています。全てのトランザクションは$\overline{CS}$が LOW になっている間に実行され、アドレス、制御、およびデータはデバイスにバイト サイズ ブロック単位でシリアル入力されます。アドレス、制御、および入力データは SI ピンを介して入力され、出力データは SO ピンを介して出力されます。

オペコードは、デバイスに対する制御を提供します。読み出しと書き込みトランザクションは、オペコード - アドレス - データというシーケンスに従います。データ転送の必要ない他のトランザクションは、ステータス レジスタの書き込みイネーブル ラッチ (WEL) ビットをセット／クリアするために使用されます。ステータス レジスタは表 4 に示され、本書の後半で記載されます。EEPROM およびフラッシュ ベースの SPI メモリでは、ステータス レジスタは$\overline{RDY}$という重要なビットも含んでいます。これは、書き込みサイクルが完了したかどうかを SPI コントローラーに通知するレディー フラグです。EEPROM およびフラッシュ メモリは通常、書き込み後にデバイスにアクセスできる前に 5ms～10ms の遅延を必要とします。F-RAM では、遅延がなく、内部書き込みが完了するのを待機する必要がありません。そのため、$\overline{RDY}$ビットは常に論理「0」です。F-RAM ステータス レジスタは、遅い EEPROM およびフラッシュ メモリと動作するファームウェアを備えたコントローラーが速い F-RAM 製品に素早く対応できるようこの$\overline{RDY}$ = 0 ビットを含みます。即ち、コントローラーは、F-RAM メモリの読み出し／書き込みを RAM の実行速度で実行することができます。

SPI F-RAM デバイスを制御するのは 6 つのオペコードです。いくつかの SPI F-RAM デバイスは高速読み出し、スリープ遷移、デバイス ID およびシリアル番号読み出し機能用の他のオペコードを追加します。各オペコードは、動作するようメモリまたはステータス レジスタに指示する 8 ビット コマンドです。アクティブな$\overline{CS}$サイクル毎に 1 つのオペコードのみが発行されます。表 1 では、全てのオペコードを説明します。

表 1. オペコードの説明

名称	オペコード	アドレス	ダミー バイト	データ	動作
WREN	0000_0110b	-	-	-	WEL をセット
書き込み	0000_0010b[2]	3 バイト[1]	-	メモリの入力データ	WEL = 1 の場合、F-RAM アレイにデータを書き込む $\overline{CS}$が HIGH になると、WEL がクリア
読み出し	0000_0011b[2]	3 バイト[1]	-	メモリの出力データ	F-RAM アレイからデータを読み出す
WRDI	0000_0100b	-	-	-	WEL をクリア

名称	オペコード	アドレス	ダミー バイト	データ	動作
RDSR	0000_0101b	-	-	ステータス レジスタの出力データ	WPEN、BP(1:0)、WEL ビットを読み出す
WRSR	0000_0001b	-	-	ステータス レジスタの入力データ	WPEN および BP(1:0)ビットを書き込む
SLEEP[3]	1011_1001b	-	-	-	スリープ モードに移行
FSTRD[3]	0000_1011b	3 バイト[1]	1 バイト	メモリの出力データ	40MHz で F-RAM アレイからデータを読み出す
RDID	1001_1111b	-	-	9 バイト デバイス ID データ出力	9 バイト デバイス id を読み出す
SNR[3]	1100_0011b	-	-	8 バイトの連番データ出力	8 バイト連番を読み出す

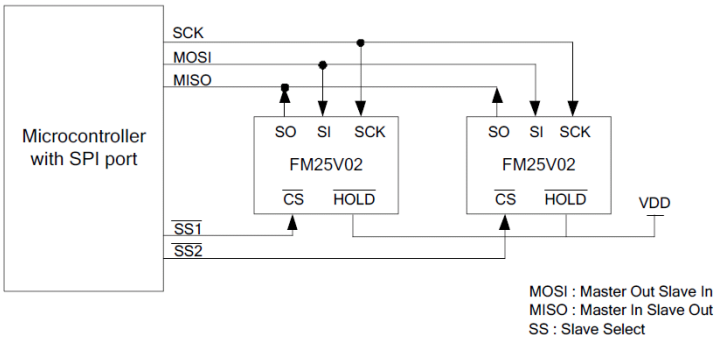
注

1. 容量に応じて、1 バイトまたは 2 バイトのアドレス指定を使用する SPI デバイスもあります。表 2 を参照してください。
2. 4-K ビット デバイスには、書き込みと読み出しオペコードのビット 3 は上位アドレス ビット (A8) に対応します。
3. すべての SPI デバイスはこのコマンドをサポートしない場合があります。

システムの接続

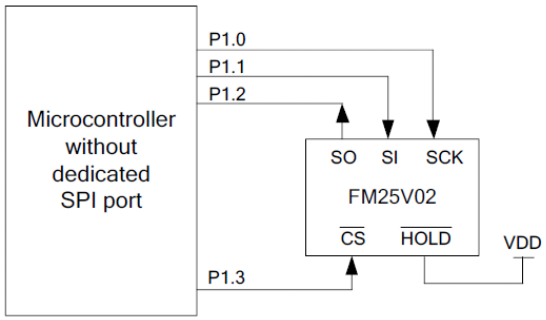
コントローラーがチップ セレクトをそれぞれの F-RAM デバイスに駆動するための追加のピンを持っている限り、複数のデバイスを使用することができます。図 3 には、マイクロコントローラーの標準 SPI ポートと連結する 2 個の F-RAM デバイスのシステム構成を示します。

図 3. 2 個の F-RAM デバイスのシステム構成



専用 SPI バスを持たないマイクロコントローラーでは、汎用ポートが図 4 の用に使用されることもあります。ビット バンギング コードはこのインターフェースを駆動します。

図 4. GPIO を使用したマイクロコントローラーと単一 F-RAM のシステム構成



単独型の SPI F-RAM 製品

下表では、単独型の SPI F-RAM 製品の基本特性をまとめます。

表 2. SPI F-RAM 製品ラインアップ

		3V									5V			
	FM25L04B	FM25L16B	FM25CL64B	FM25V01	FM25V02	FM25V05	FM25V10	FM25V20 FM25V20A	FM25H20	FM25V40	FM25040B	FM25C160B	FM25640B	FM25W256
容量	4K ビット	16K ビット	64K ビット	128K ビット	256K ビット	512K ビット	1M ビット	2M ビット	2M ビット	4M ビット	4K ビット	16K ビット	64K ビット	256K ビット
内部構成	512 x 8	2K x 8	8K x 8	16K x 8	32K x 8	64K x 8	128K x 8	256K x 8	256K x 8	512K x 8	512 x 8	2K x 8	8K x 8	32K x 8
アドレス ビット数	9	11	13	14	15	16	17	18	18	19	9	11	13	15
アドレス バイト数	1	2	2	2	2	2	3	3	3	3	1	2	2	2
動作電圧	2.7 V～ 3.6V	2.7 V～ 3.6V	2.7 V～ 3.65V	2.0 V～ 3.6V	2.0 V～ 3.6V	2.0 V～ 3.6V	2.0 V～ 3.6V	2.0 V～ 3.6V	2.7 V～ 3.6V	2.0 V～ 3.6V	4.5 V～ 5.5V	4.5 V～ 5.5V	4.5 V～ 5.5V	2.7 V～ 5.5V
最大クロック周波数	20MHz	20MHz	20MHz	40MHz	40MHz	40MHz	40MHz	40MHz	40MHz	40MHz	20MHz	20MHz	20MHz	20MHz
サポートするクロックモード	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3	0、3
スリープモード				✓	✓	✓	✓	✓	✓	✓				
ユニークな S/N							✓							
デバイス ID				✓	✓	✓	✓	✓		✓				
パッケージ	SOIC8 DFN8 (4x4.5)	SOIC8 DFN8 (4x4.5)	SOIC8 DFN8 (4x4.5)	SOIC8	SOIC8 DFN8 (4x4.5)	SOIC8	SOIC8	幅広い SOIC8 DFN8[1] (5x6)	幅広い SOIC8 DFN8[1] (5x6)	幅広い SOIC8 DFN8[1] (5x6)	SOIC8	SOIC8	SOIC8	SOIC8

注:
1. 5 x 6mm DFN8 は SOIC8 フットプリントに準拠しています。

読み出し／書き込みトランザクション

SPI インターフェースは、コントローラーが駆動するクロックに同期しています。全ての F-RAM SPI デバイスは、SCK の立ち上がりエッジで入力データを登録し、SCK の立ち下がりエッジでデータをコントローラーに返します。このタイミングに従うために、コントローラーは通常、SCK の立ち下がりエッジで信号をメモリに駆動します。これは、信号が伝播する時間があり、メモリ デバイスのセットアップタイミング仕様を満たせます。

メモリ読み出し

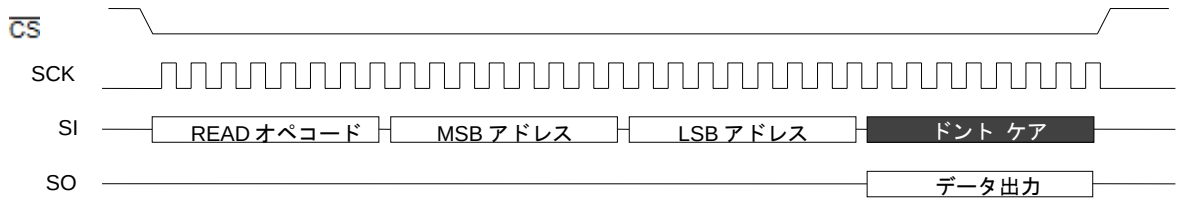
フォーマット: *READ op-code, MSB Address, LSB Address, Data-out, (Data-out, Data-out ...)*

読み出しサイクル中に、コントローラーは READ オペコードとアドレスを発行します。データは SO ピンを介して出力されます (例えば、data-out(0)、data-out(1)、data-out(2) など)。$\overline{CS}$ ピンは、サイクル全体にわたって LOW のままでなければなりません。$\overline{CS}$ が HIGH にデアサートされると、データ出力が停止し、SO は HI-Z 状態になります。クロック入力されたアドレスは、最初のデータ バ

イトの開始アドレスです。後続のデータ バイトには、単にデータ バイトを連続的にクロック出力しながら$\overline{CS}$をLOW に維持することでアクセスできます。各バイトは、SPI F-RAM デバイスによって増分されるアドレスから読み出されます。

図 5 には、16K ビット～512K ビット容量用に使用される 2 バイト アドレスを示します。

図 5. SPI の読み出しタイミング



メモリ書き込み

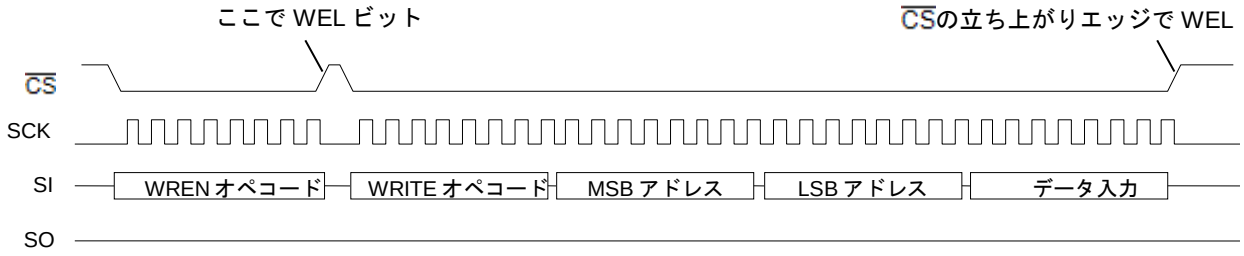
フォーマット: *WREN op-code, WRITE op-code, MSB Address, LSB Address, Data-in, (Data-in, Data-in, ...)*

書き込みサイクルでは、コントローラーは、次のシーケンスに従って 2 つのオペコード (WREN と WRITE) を発行します。各オペコードは$\overline{CS}$が LOW の間以内です。WREN オペコードに続いて WRITE オペコード、アドレスおよびデータを発行します (例えば、data-in(0)、data-in(1)、data-in(2) など)。クロック入力されたアドレスは、最初のデータ バイトの開始アドレスです。後続のデータ バイトには、単にデータ バイトを連続的にクロック入力しながら$\overline{CS}$を LOW に維持することで書き込むことができます。各バイトは、SPI F-RAM デバイスによって増分されるアドレスに書き込まれます。各データ バイトは、そのバイトの 8 番目のクロック エッジで F-RAM アレイに書き込まれます。ページ バッファも書き込み遅延もありません。

ステータス レジスタの WEL ビットは、内部で SPI F-RAM デバイスによってセットされ、クリアされます。これは、WREN オペコードをクロック入力した後にセットされ、書き込み動作の終了時に$\overline{CS}$の立ち下がりエッジでクリアされます。WREN と WRITE オペコードの間でステータス レジスタを読み出すこと (RDSR オペコード) は WEL ビットをクリアしません。WEL ビットがセットされたかをチェックするために、WREN の直後にステータス レジスタを読み出すユーザーもいます。しかし、WEL ビットの読み出しは、書き込み動作を完了するのに必要ありません。

図 6 は 1 バイトの完全な書き込みトランザクションを示します。これは、16K ビット～512K ビット容量用に使用される 2 バイト アドレスを示します。

図 6. SPI の書き込みタイミング



ステータス レジスタ書き込み

フォーマット: *WREN op-code, WRSR op-code, Data-in*

以下に示されるように、ステータス レジスタは$\overline{WP}$イネーブル (WPEN) ビットおよびブロック保護 (BP1、BP0) ビットを含んでいます。

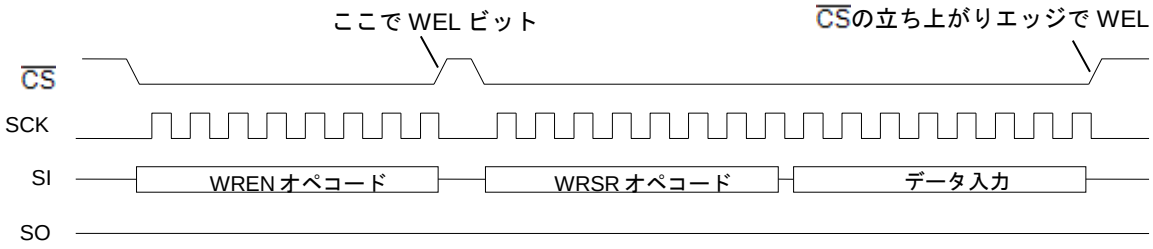
Status Register							
7	6	5	4	3	2	1	0
WPEN	0	0	0	BP1	BP0	WEL	0

ステータス レジスタに書き込むことで、メモリ ブロックを書き込みから保護し、$\overline{WP}$ピンを有効にすることができます。BP1 および BP0 という 2 個のブロック保護ビットがあります。これらにより、上位 1/4、上位 1/2、または

全体のアレイを書き込みから保護することができます。BP0、BP1 および WPEN ビットは黄色で示されます。これらのビットは不揮発性であり、書き込まれた値が電源切断後に再投入しても保持されます。WPEN は、外部$\overline{WP}$ピンを有効か無効にします。システムの不正使用の時に$\overline{WP}$ピンをオーバーライドするために、ソフトウェアを使用することができます。WEL は、書き込みイネーブル ラッチがセットされたかを単にユーザーに通知する読み出し専用ビットであり、ステータス レジスタまたはメモリへの書き込みを可能にします。

図 7 には、ステータス レジスタの完全な書き込みトランザクションを示します。

図 7. ステータス レジスタの書き込みタイミング

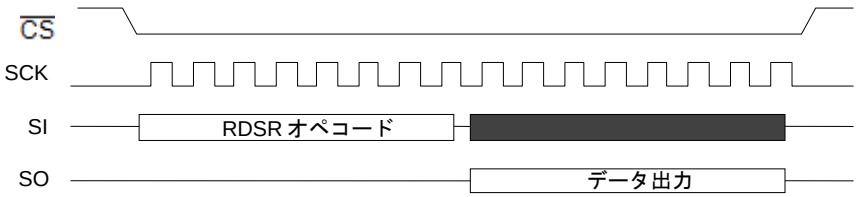


ステータス レジスタ読み出し

フォーマット: *RDSR op-code, Data-out*

ステータス レジスタを読み出すことで、WPEN ビット、BP (1:0) 書き込み保護ビット、WEL ビットの状態をチェックできます。図 8 には、ステータス レジスタの完全な読み出しトランザクションを示します。

図 8. ステータス レジスタの読み出しタイミング



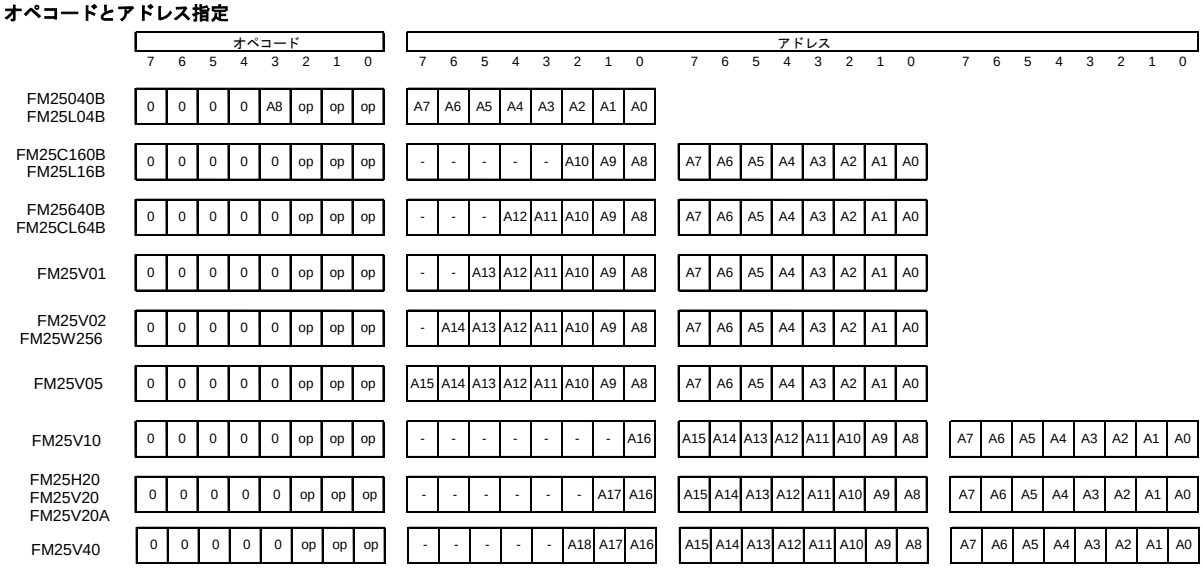
SPI F-RAM アドレス指定

単独型の SPI F-RAM デバイスは、容量に応じて 1 バイト、2 バイト、または 3 バイト アドレスを必要とします。オペコードに続いて、開始アドレスは MSB ファーストでシフトインされます。最下位アドレス バイト (LSB) の直後に、書き込みの場合は期待されるデータがマスタから入力され、読み出しの場合はデータがメモリから出力されます。SCK が引き続きトグルする限り、内部アドレスは自動的に増分され、データ入力／出力は$\overline{CS}$がデアサートされるまで継続します。

注: 4K ビット デバイスは、1 バイトのアドレスのみ必要とします。

図 9 には、SPI F-RAM デバイスの異なる容量に対応するアドレス指定を示します。

図 9. 異なる容量間のアドレス指定の違い



高容量 F-RAM デバイスを低容量ソケットに配置

特定のデバイスがなく、基板またはシステムが低い容量の F-RAM デバイス向けに設計された場合、より高い容量のデバイスと置き換えることができます。例えば、16K ビット デバイスを使用するよう設計されたシステムは、64K ビット、128K ビット、256K ビット、または 512K ビット デバイスも使用できます。図 9 には、これらデバイスは共通の 2 バイト アドレスを読み出しと書き込み動作に使用することを示します。指定された動作電圧とタイミング要件が満たされることを前提にすると、これら容量の範囲内のデバイスはピン配置、パッケージ (SOIC)、読み出し／書き込み機能において同じです。2 つの潜在的な問題があります: システムがデバイス内でアドレス ラップ機能を使用する場合、またはシステムがブロック保護機能を使用する場合。16K ビット デバイスの終端アドレスは 0x800、64K ビット デバイスの終端アドレスは 0x2000、128K ビット デバイスの終端アドレスは 0x4000 などです。容量が 2 倍違うデバイスを比較する時、ブロック保護境界はアドレスの 2 倍の間隔で開けられます。

例えば、図 10 および図 11 には、16K ビットと 128K ビット デバイスのシリアル アドレス ストリーム間の違いを示します。

図 9 に示された 16K ビットと 128K ビット デバイスのアドレス要件を比較すると、3 つの追加のアドレス ビット位置 (図 11 に示す、赤枠で囲んだ A13、A12、A11) が 128K ビット デバイスにおいて使用されることが明らかになります。コントローラーがこれら 3 個のアドレス ビットを読み出しと書き込み用に一貫して駆動する限り、高い容量デバイスは、低い容量の製品に設計されたシステムで動作することができます。1M ビット容量 (またはそれ以上) デバイスは 3 バイト アドレスを使用し、より低い容量に設計されたシステムで置き換え製品として使用することができません。例えば、2 バイト アドレスを発行するシステムは、3 バイト メモリが使用されている場合は正常に動作しません。容量に応じたアドレス バイト数については、表 2 を参照してください。

図 10. FM25L16B 書き込みサイクル (WREN が示されない)

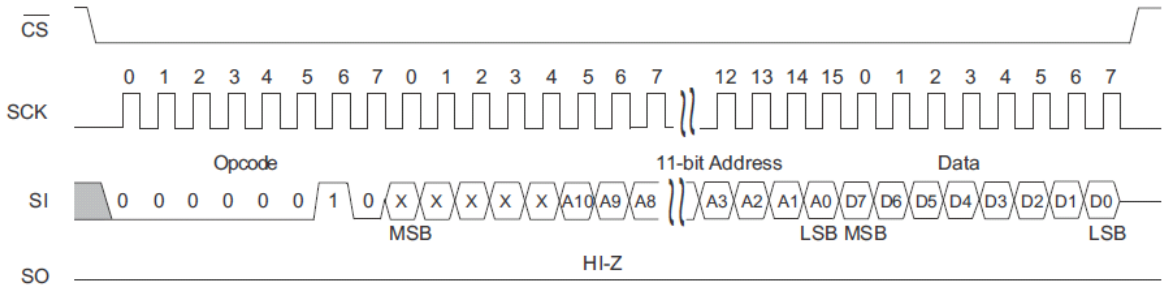
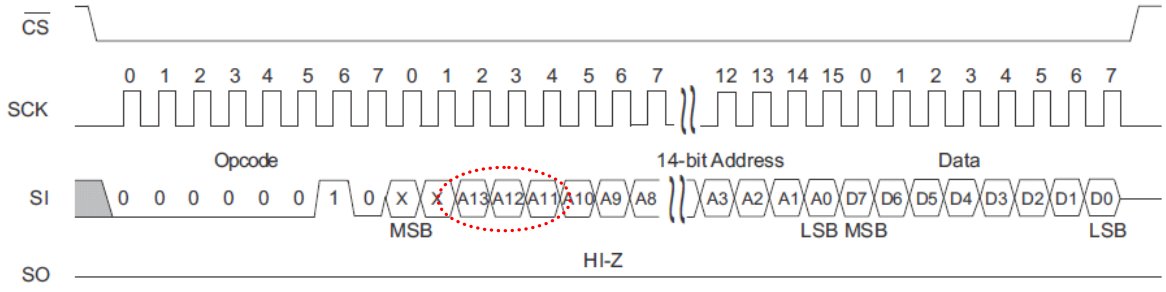


図 11. FM25V01 書き込みサイクル (WREN が示されない)



クロック モード

FM25xxx デバイス ファミリは、4 つの SPI 標準クロック モードの中で、モード 0 とモード 3 をサポートしています。全ての F-RAM 製品は、モードに関係なく、SCK エッジの立ち上がりエッジでデータをデバイスにクロック入力し、SCK の立ち下がりエッジでデータをクロック出力することに注意してください。モード 0 とモード 3 の相違点は、$\overline{CS}$が LOW にアサートされる時に SCK が LOW か HIGH で始まるかだけです。異なる SPI モードは、表 33 に示します。

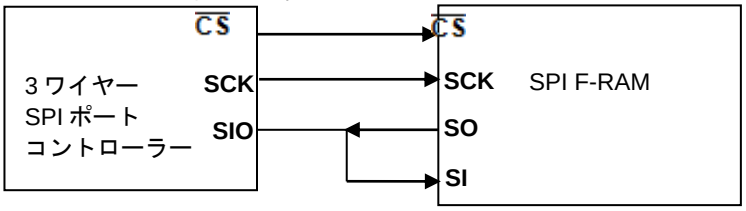
表 3. SPI モード

	モード 0	モード 1	モード 2	モード 3
SCK が開始した時の状態	LOW	LOW	HIGH	HIGH
SI でのデータ入力がラッチされるエッジ	↑	↓	↓	↑
SO でのデータ出力が駆動されるエッジ	↓	↑	↑	↓

半二重動作

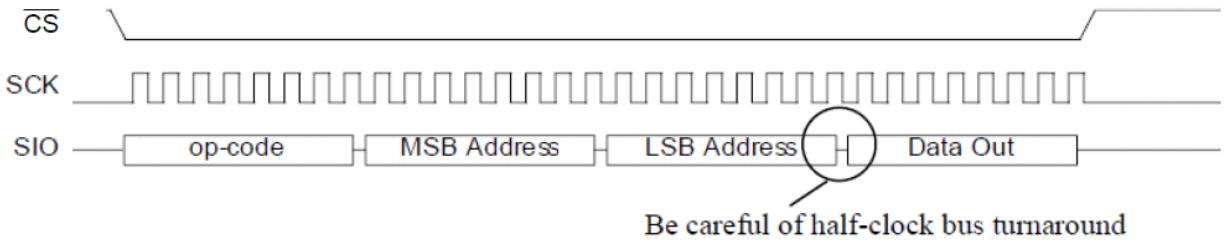
SPI インターフェースのピン数を減らすために、データ ラインを互いに結んで共通データ I/O ラインを作ることができます。図 12 に示されるように、この 3 線式インターフェースは SPI の最小ピン数の構成です。ここで、コントローラーは、SIO ラインが読み出しサイクル中に hi-Z であること保証する必要があります。そうしないと、バスの競合が発生します。2 番目の問題は、データ バスが半二重になったため、データ帯域幅が減少する可能性があります。

図 12. 本当の 3 線式 SPI インターフェース



タイミング モード (0 か 3) にかかわらず、全ての SPI F-RAM は、SCK の立ち上がりエッジで入力データをラッチし、SCK の立ち下がりエッジで出力データを駆動します。SPI 読み出しトランザクションは、図 13 に示します。

図 13. 共通データ I/O ラインとバス ターンアラウンド



書き込み保護

SPI F-RAM デバイスは、WPハードウェア ピンを使用するか、またはステータス レジスタのビットをプログラムすることで書き込みから保護することができます。F-RAM メモリ アレイに加えて、ステータス レジスタその自身も保護されます。ステータス レジスタは、メモリ アレイの領域への書き込みを無効にするための不揮発性ブロック保護ビット BP (1:0)を含んでいます。BP0、BP1 および WPEN は不揮発性であり、電源切断後に再投入しても保持されますが、表 5 に黄色で示されます。WPEN はWP ハードウェア ピンを有効にします。ビット位置 0 は、EEPROM とシリアル フラッシュ用のRDYビットとして予約されています。このビットはそれらデバイスで使用され、ユーザーはステータス レジスタを読み出すことで、メモリが他のコマンドの受信の準備が完了したかどうかを判断することができます。書き込みサイクル後にチップが常に準備ができる (ゼロ遅延) ため、RDYビットは全ての SPI F-RAM デバイスの内部で LOW に接続されています。

表 4: ステータス レジスタおよびブロック保護の設定

Status Register							
7	6	5	4	3	2	1	0
WPEN	0	0	0	BP1	BP0	WEL	0

表 5 は、ステータス レジスタおよび F-RAM アレイの書き込み保護に関する全てのケースを含む書き込み保護の表を示します。WEL = 0 の場合、F-RAM アレイおよびステータス レジスタへの書き込みは全てブロックされます。

表 5. 書き込み保護

WEL	WPEN	$\overline{WP}$	Protected Blocks	Unprotected Blocks	Status Register
0	X	X	Protected	Protected	Protected
1	0	X	Protected	Unprotected	Unprotected
1	1	0	Protected	Unprotected	Protected
1	1	1	Protected	Unprotected	Unprotected

- WPEN=0、WPピン=1 でも、F-RAM アレイは、BP ビットが HIGH になることで保護される
- WPEN=1、WPピン=0 でも、F-RAM アレイは、BP ビットが LOW になることで保護されない
- WPEN=1、WPピン=0 の場合にのみ、ステータス レジスタは保護される

注: FM25040B および FM25L04B は WPEN ビットがありません。WPピン = 0 の時、全ての書き込み (メモリアレイおよびステータス レジスタ) はブロックされます。

パワー サイクル

F-RAM デバイスは高速の不揮発性メモリですが、読み出しまたは書き込みシーケンスで発生する電源グリッチは誤ってアレイデータをオーバーライト (破損) する可能性があります。例えば、チップ セレクトがアクティブ (LOW) の時に、デバイスが中レベル電源電圧で不注意にデータを書き込む可能性があります。SPI F-RAM データシートは、チップ セレクトが非アクティブ (HIGH) になっている時にデバイスの電源を切断することを指定 (推奨) しています。

SPI F-RAM デバイスは、簡単な内部パワーオン リセット回路以外に他の電源管理回路を備えていません。誤った動作を防ぐために V_{DD} がデータシートに記載された許容誤差以内であることを保証してください。V_{DD} 電源電圧を、しっかりと制御して立ち上げおよび立ち下げすることが推奨されています。スイッチング電源は、投入時と切断時に出力が制御されないという不利な点を持っています。

システム設計者は、パワー サイクル中のチップ イネーブルと V_{DD} の状態を理解する必要があります。データ保護の詳細については、「*AN302 - F-RAM SPI Read & Write Internal Operation and Data Protection*」を参照してください。

まとめ

本アプリケーション ノートは、異なる F-RAM SPI 製品の機能の詳細、タイミング、およびサンプル コードについて説明します。

関連アプリケーション ノート

SPI F-RAM デバイスのさらなる理解のために、以下のアプリケーション ノートを参照してください。

- AN302 - F-RAM SPI Read & Write Internal Operation and Data Protection
- AN408 - A Design Guide to SPI F-RAM Processor Companion - FM33256B

付録 A: 疑似サンプル コード (1 バイト アドレス、4K ビット デバイス)

```
#define WREN  0x06
#define WRITE 0x02 // Write opcode to access lower half of memory
#define WRITE 0x0A // Write opcode to access upper half of memory
#define READ  0x03  // Read opcode to access lower half of memory
#define READ  0x0B  // Read opcode to access upper half of memory
#define RDSR  0x05
#define WRSR  0x01
#define WRDI  0x04
```

以下の各事例では、左括弧は$\overline{CS}$ピンがLOWになること、右括弧はこのピンがHIGHになることを示します。

```
/****** Memory Write (single byte to location 0130h) *******/
WREN (0x06)         // Sets WEL bit. WREN must precede WRITE opcode.
WRITE (0x0A,        // 0x02 is WRITE opcode and A8 bit set
0x30,               // starting address
0x55)               // 0x55 is data written to location 0130h

/****** Memory Write (multiple bytes to starting location 01FCh) *******/
WREN (0x06)         // Sets WEL bit. WREN must precede WRITE opcode.
WRITE (0x0A,        // 0x02 is WRITE opcode and A8 bit set
0xFC,               // starting address
0x55,               // 0x55 is data written to location 01FCh
0xAA,               // 0xAA is data written to location 01FDh
0x55,               // 0x55 is data written to location 01FEh
0xAA)               // 0xAA is data written to location 01FFh

/****** Memory Read (single byte from location 01D3h) *******/
READ (0x0B,         // 0x03 is READ opcode
0xD3,               // starting address
0xAA)               // 0xAA is data read from location 01D3h

/****** Memory Read (multiple bytes from starting location 01FCh) *******/
READ (0x0B,         // 0x03 is READ opcode
0xFC,               // starting address
0x55,               // 0x55 is data read from location 01FCh
0xAA,               // 0xAA is data read from location 01FDh
0x55,               // 0x55 is data read from location 01FEh
0xAA)               // 0xAA is data read from location 01FFh

/****** Write Status Register (write protect upper half of memory) *******/
WREN (0x06)         // Sets WEL bit. WREN must precede WRSR opcode.
WRSR (0x01,         // 0x01 is WRSR opcode
0xF8)               // 0xF8 sets the BP1 bit which protects the upper
                    // half of the memory array. The upper nibble
                    // set to “F” attempts to write the upper bits
                    // to 1.

/****** Read Status Register *******/
RDSR (0x05,         // 0x05 is RDSR opcode
0x08)               // 0x08 tells us that the BP1 bit is set and that
                    // the upper half of the memory array is protected.
                    // The upper nibble returns “0” since they are
                    // hardwired low.
```

注: **青色**のテキストは、データがコントローラーによって送信されていることを示します。**赤色**のテキストは、データがコントローラーによって受信されていることを示します。

付録 B: 疑似サンプル コード (2 バイト アドレス、16K ビット～512K ビット デバイス)

```
#define WREN 0x06
#define WRITE 0x02
#define READ 0x03
#define RDSR 0x05
#define WRSR 0x01
#define WRDI 0x04
```

以下のどの場合でも、左括弧は$\overline{CS}$ピンがLOWになること、右括弧はこのピンがHIGHになることを示します。

```
/****** Memory Write (single byte to location 0F30h) *******/
WREN (0x06)  // Sets WEL bit. WREN must precede WRITE opcode.
WRITE (0x02, // 0x02 is WRITE opcode
0x0F,        // starting address MSB
0x30,        // starting address LSB
0x55)        // 0x55 is data written to location 0F30h

/****** Memory Write (multiple bytes to starting location 07FC) *******/
WREN (0x06)  // Sets WEL bit. WREN must precede WRITE opcode.
WRITE (0x02, // 0x02 is WRITE opcode
0x07,        // starting address MSB
0xFC,        // starting address LSB
0x55,        // 0x55 is data written to location 07FCh
0xAA,        // 0xAA is data written to location 07FDh
0x55,        // 0x55 is data written to location 07FEh
0xAA)        // 0xAA is data written to location 07FFh

/****** Memory Read (single byte from location 0F31h) *******/
READ (0x03,  // 0x03 is READ opcode
0x0F,        // starting address MSB
0x31,        // starting address LSB
0xAA)        // 0xAA is data read from location 0F31h

/****** Memory Read (multiple bytes from starting location 07FCh) *******/
READ (0x03,  // 0x03 is READ opcode
0x07,        // starting address MSB
0xFC,        // starting address LSB
0x55,        // 0x55 is data read from location 07FCh
0xAA,        // 0xAA is data read from location 07FDh
0x55,        // 0x55 is data read from location 07FEh
0xAA)        // 0xAA is data read from location 07FFh

/****** Write Status Register (write protect upper half of memory) *******/
WREN (0x06) // Sets WEL bit. WREN must precede WRSR opcode.
WRSR (0x01, // 0x01 is WRSR opcode
0x08)       // 0x08 sets the BP1 bit which protects the upper
            // half of the memory array.
/****** Read Status Register *******/
RDSR (0x05, // 0x05 is RDSR opcode
0x88)       // 0x88 tells us that the BP1 bit is set and that
            // the upper half of the memory array is protected.
            // The WPEN bit is also set which works with
            // the WP pin to protect the Status Register.
```

注: **青色**のテキストは、データがコントローラーによって送信されていることを示します。**赤色**のテキストは、データがコントローラーによって受信されていることを示します。

付録 C: 疑似サンプル コード (3 バイト アドレス、1M ビット～4M ビット デバイス)

```
#define WREN 0x06
#define WRITE 0x02
#define READ 0x03
#define RDSR 0x05
#define WRSR 0x01
#define WRDI 0x04
```

以下のどの場合でも、左括弧は$\overline{CS}$ピンがLOWになること、右括弧はこのピンがHIGHになることを示します。

```
/****** Memory Write (single byte to location 1BF30h) *******/
WREN (0x06)  // Sets WEL bit. WREN must precede WRITE opcode.
WRITE (0x02, // 0x02 is WRITE opcode
0x01,        // starting address MSB
0xBF,        // starting address 2nd byte
0x30,        // starting address LSB
0x55)        // 0x55 is data written to location 1BF30h

/****** Memory Write (multiple bytes to starting location 1B7FC) *******/
WREN (0x06)  // Sets WEL bit. WREN must precede WRITE opcode.
WRITE (0x02, // 0x02 is WRITE opcode
0x01,        // starting address MSB
0xB7,        // starting address 2nd byte
0xFC,        // starting address LSB
0x55,        // 0x55 is data written to location 1B7FCh
0xAA,        // 0xAA is data written to location 1B7FDh
0x55,        // 0x55 is data written to location 1B7FEh
0xAA)        // 0xAA is data written to location 1B7FFh

/****** Memory Read (single byte from location 1BF31h) *******/
READ (0x03,  // 0x03 is READ opcode
0x01,        // starting address MSB
0xBF,        // starting address 2nd byte
0x31,        // starting address LSB
0xAA)        // 0xAA is data read from location 1BF31h

/****** Memory Read (multiple bytes from starting location 1B7FCh) *******/
READ (0x03,  // 0x03 is READ opcode
0x01,        // starting address MSB
0xB7,        // starting address 2nd byte
0xFC,        // starting address LSB
0x55,        // 0x55 is data read from location 1B7FCh
0xAA,        // 0xAA is data read from location 1B7FDh
0x55,        // 0x55 is data read from location 1B7FEh
0xAA)        // 0xAA is data read from location 1B7FFh



/****** Write Status Register (write protect upper half of memory) *******/
WREN (0x06)  // Sets WEL bit. WREN must precede WRSR opcode.
WRSR (0x01,  // 0x01 is WRSR opcode
0x08)        // 0x08 sets the BP1 bit which protects the upper
             // half of the memory array.

/****** Read Status Register *******/
RDSR (0x05,  // 0x05 is RDSR opcode
0x88)        // 0x88 tells us that the BP1 bit is set and that
             // the upper half of the memory array is protected.
             // The WPEN bit is also set which works with
```

```
                // the WP pin to protect the Status Register.
```

注: **青色**のテキストは、データがコントローラーによって送信されていることを示します。**赤色**のテキストは、データがコントローラーによって受信されていることを示します。

付録 D: PSoC 3 ベースのユーザー モジュール向けのサンプル コード

以下のはPSoC 3ベースのユーザー モジュール向けのサンプル コードです。

SPI メモリ書き込み

```
/************ Memory Write (write 0x55 0xAA 0x55 0xAA from location 1B7FCh) **********/
data[0] = 0x55;
data[1] = 0xAA;
data[2] = 0x55;
data[3] = 0xAA;

WRITE (0x01B7FC,           // Sets the address pointer to memory location 1B7FCh
           data,           // Write data
           0x04);          // Number of bytes to be written
```

SPI メモリ読み出し

```
/************ Memory Read (read 4 bytes from location 1B7FCh) **********/
READ (0x01B7FC,            // Sets the address pointer to memory location 1B7FCh
          data,            // Pointer to store the read data
          0x04);           // Number of bytes to be read
// data buffer contains 0x55, 0xAA, 0x55, 0xAA after read
```

SPI 書き込み関数

```
/**********PSoC3 Based Code for SPI Write **********/
uint8 WRITE (uint32 addr, uint8 *data_write_ptr, uint32 total_data_count)
{
   uint8 i;

   // Clear the Transmit Buffer
   NVRAM_SPI_1_SPIM_ClearTxBuffer();

   // Send Write Enable WREN command
   // Make chip select LOW CS = 0
   NVRAM_SPI_1_CS_Reg_Write(0);
   // Send Write Enable Command WREN
   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WREN);
   // Wait for the transfer to complete
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) !=
   NVRAM_SPI_1_SPIM_STS_SPI_DONE);
   // Make chip select High CS = 1
   NVRAM_SPI_1_CS_Reg_Write(1);

   // Delay
   CyDelay(1);

   // Clear the Transmit Buffer
   NVRAM_SPI_1_SPIM_ClearTxBuffer();

   // Make chip select LOW CS = 0
   NVRAM_SPI_1_CS_Reg_Write(0);
   // Send NVRAM Write Command
   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_WRITE_CMD);
   // Send NVRAM address
   if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
   {
      // Send MSB of 3-byte address for F-RAM densities of 1-Mbit and greater
      NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16));
   }
   // Send the second byte of 3-byte address or MSB of 2-byte address
   NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8));
   // Send LSB address byte
   NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr));

   // Wait for the transfer to complete
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) !=
   NVRAM_SPI_1_SPIM_STS_SPI_DONE);

   // Send NVRAM data
   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);
   }
   // Make chip select HIGH CS = 1
   NVRAM_SPI_1_CS_Reg_Write(1);
}
```

SPI 読み出し関数

```
/*****************PSoC3 Based Code for SPI Read**************/
uint8 READ (uint32 addr, uint8 *data_read_ptr, uint32 total_data_count)
{
   uint8 i;

   // Clear the Transmit Buffer
   NVRAM_SPI_1_SPIM_ClearTxBuffer();

   // Make chip select LOW CS = 0
   NVRAM_SPI_1_CS_Reg_Write(0);
   // Send NVRAM Read Command
   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_READ_CMD);
   // Send NVRAM Address
   if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
   {
      // Send MSB of 3-byte address for F-RAM densities of 1-Mbit and greater
      NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16));
   }
   // Send the second byte of 3-byte address or MSB of 2-byte address
   NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8));
   // Send LSB address byte
   NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr));

   // Wait for the transfer to complete
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) !=
   NVRAM_SPI_1_SPIM_STS_SPI_DONE);

   // Read the data and store in data_read_ptr
   for(i = 0; i < total_data_count; i++ )
   {
      // Clear receive buffer
      NVRAM_SPI_1_SPIM_ClearRxBuffer();
      // Send a dummy byte
      NVRAM_SPI_1_SPIM_WriteTxData((uint8)0x00);
      // Wait for the transfer to complete
      while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE) !=
      NVRAM_SPI_1_SPIM_STS_SPI_DONE);
      // Wait till the receive buffer has received a byte
     while(!NVRAM_SPI_1_SPIM_GetRxBufferSize());
      // Copy the read byte to data_read_ptr
      data_read_ptr[i] = NVRAM_SPI_1_SPIM_ReadRxData();
   }

   // Make chip select HIGH CS = 1
   NVRAM_SPI_1_CS_Reg_Write(1);
}
```

改訂履歴

文書名: F-RAM™向けの SPI ガイド - AN304

文書番号: 001-92166

版	ECN 番号	変更者	発行日	変更内容
**	4347534	HZEN	04/15/2014	これは英語版001-87196 Rev. **を翻訳した日本語版001-92166 Rev. **です。
*A	4722807	HZEN	04/27/2015	これは英語版001-87196 Rev. *Cを翻訳した日本語版001-92166 Rev. *Aです。
*B	5710922	MEDU	04/26/2017	CE204087へのリンクを追加。 これは英語版001-87196 Rev. *Dを翻訳した日本語版です。

セールス, ソリューションおよび法律情報

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

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

© Cypress Semiconductor Corporation, 2013-2017. 本書面は, Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。) を含む) は, アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し, 本段落で特に記載されているものを除き, その特許権, 著作権, 商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず, かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合, Cypress は, (1) 本ソフトウェアの著作権に基づき, (a) ソースコード形式で提供されている本ソフトウェアについて, Cypress ハードウェア製品と共に用いるためにのみ, かつ組織内部でのみ, 本ソフトウェアの修正及び複製を行うこと, 並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ, (直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること, 並びに (2) 本ソフトウェア (Cypress により提供され, 修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき, Cypress ハードウェア製品と共に用いるためにのみ, 本ソフトウェアの作成, 利用, 配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用, 複製, 修正, 変換又はコンパイルを禁止する。

適用される法律により許される範囲内で, Cypress は, 本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても, 明示又は黙示をとわず, いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。適用される法律により許される範囲内で, Cypress は, 別途通知することなく, 本書面を変更する権利を留保する。Cypress は, 本書面に記載のある, いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は, 参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計, プログラム, かつテストすることは, 本書面のユーザーの責任において行われるものとする。Cypress 製品は, 兵器, 兵器システム, 原子力施設, 生命維持装置若しくは生命維持システム, 蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム, 汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用, 又は装置若しくはシステムの不具合が人身傷害, 死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計, 意図又は承認されていない。重要な構成部分とは, それの不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ, 若しくは本目的外使用に関連するいかなる請求, 損害又はその他の責任についても, Cypress はその全部又は一部をとわず一切の責任を負わず, かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求, 費用, 損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, CapsSense, EZ-USB, F-RAM, 及び Traveo は, 米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは, cypress.com を参照すること。その他の名称及びブランドは, それぞれの権利者の財産として権利主張がなされている可能性がある。