

サイプレスはインフィニオン テクノロジーズになりました

この表紙に続く文書には「サイプレス」と表記されていますが、これは同社が最初にこの製品を開発したからです。新規および既存のお客様いずれに対しても、引き続きインフィニオンがラインアップの一部として当該製品をご提供いたします。

文書の内容の継続性

下記製品がインフィニオンの製品ラインアップの一部として提供されたとしても、それを理由としてこの文書に変更が加わることはありません。今後も適宜改訂は行いますが、変更があった場合は文書の履歴ページでお知らせします。

注文時の部品番号の継続性

インフィニオンは既存の部品番号を引き続きサポートします。ご注文の際は、データシート記載の注文部品番号をこれまで通りご利用下さい。

Traveo II ファミリでの CRYPTO モジュールの使用方法

著者: Christian Eyrich

関連製品ファミリ: Traveo II ファミリ

AN220253 では、Traveo™ II ファミリの暗号ブロック (CRYPTO) モジュールの設定および使用方法について説明します。このアプリケーションノートでは、非対称/対称暗号化/復号化、ハッシュ値計算、真性乱数と擬似乱数生成など、さまざまな暗号関連機能の使用方法について説明します。

Contents

1 はじめに	1	7 Advanced Encryption Standard(AES)を使用した対称鍵暗号化	8
2 暗号化機能概要と特徴	2	7.1 ユースケース	9
2.1 乱数生成器(Random Number Generator: RNG)	2	7.2 ドライバ関数	9
2.2 対称暗号化	2	7.3 フローチャート	9
2.3 非対称暗号化	2	7.4 他の対称鍵アルゴリズム	9
2.4 ハッシュ関数	2	8 SHA 暗号ハッシュ関数ファミリ	10
2.5 メッセージ認証コード(MAC)	2	8.1 ユースケース	10
2.6 デジタル署名	2	8.2 ドライバ関数	10
3 Crypto ドライバ	3	8.3 フローチャート	10
3.1 ドライバアーキテクチャ	3	9 ハッシュベースおよび暗号ベースのメッセージ認証コード(HMAC, CMAC)	10
3.2 ドライバの初期化	3	9.1 ユースケース	11
3.3 ドライバの使い方	4	9.2 ドライバ関数	11
4 巡回冗長検査(CRC)	5	9.3 フローチャート	11
4.1 ユースケース	5	10 RSA を使用した非対称鍵暗号化	11
4.2 ドライバ関数	5	10.1 ユースケース	11
4.3 フローチャート	5	10.2 ドライバ関数	11
5 擬似乱数生成器(PRNG)	6	10.3 フローチャート	12
5.1 ユースケース	6	11 RSA と SHA を使用したデジタル署名検証	14
5.2 ドライバ関数	6	11.1 ユースケース	14
5.3 フローチャート	7	11.2 ドライバ関数	14
6 真性乱数生成器(TRNG)	7	11.3 フローチャート	14
6.1 ユースケース	7	改版履歴	16
6.2 ドライバ関数	7	ワールドワイド販売と設計サポート	17
6.3 フローチャート	7		

1 はじめに

今日の組込みアプリケーションでは、セキュリティ機能に関する要件がますます高まっています。車載アプリケーションでは、一般的に次の機能を実行するためにセキュリティ機能が使用されます。

- 知的財産の保護
- 車内の他の電子制御ユニットまたは外部の電子制御ユニットとの間のメッセージの認証

- 改ざん防止 (例えば、エンジンパラメータの調整やトリップレコーダの変更防止など)
- アフターマーケットでの機能有効化の許可 (例えば、スピードリミッタの有効化など)
- OEM(オリジナルの機器メーカー)のスペアパーツのみ使用を許可する

Traveo II ファミリのマイクロコントローラは、このような使用事例を可能にするため特別に開発されたセキュリティ機能を提供します。デバッグインタフェースまたはマイクロコントローラで実行されるソフトウェアのアクセス許可を制御するライフサイクルおよび保護スキームに加え、このアプリケーションノートで説明する暗号ブロック (CRYPTO) も重要なコンポーネントの1つです。本アプリケーションノートでは、サイプレスのサンプルドライバライブラリ (SDL) の一部である "Crypto" ドライバを用いて暗号機能について説明します。

2 暗号化機能概要と特徴

2.1 乱数生成器 (Random Number Generator: RNG)

擬似乱数生成器 (Pseudo Random Number Generator: PRNG) と真性乱数生成器 (True Random Number Generator: TRNG) の 2 種類があります。

PRNG は、多項式と与えられた開始値(以下「シード」と記載します)に基づいて、ランダムに見える数列を出力します。PRNG に関する知識をもつ攻撃者は、将来の乱数を予測できる場合があります。

TRNG は、真性のランダム性の根源として物理的性質(例えば熱ノイズ)を使用します。通常、後処理ステップが実行され、TRNG の出力を無効にしようとする攻撃者によって引き起こされた TRNG の誤動作を検出するために、監視機能が実装されます。

PRNG の利点は、通常 TRNG よりも乱数生成が高速な点です。さらに、物理的な実装に応じて、アクティブな TRNG は特定のアプリケーションにおいては、電力損失を引き起こすことがあります(例えば、リングオシレータがダイナミックスイッチング電流を発生させるなど)。これらは、多くのアプリケーションが TRNG を使用して PRNG の真性のランダムシードを生成する理由です。

2.2 対称暗号化

対称暗号化は、メッセージの送信者と受信者の間の共有秘密情報に依存します。同じ秘密鍵が暗号化と復号化に使用されます。

2.3 非対称暗号化

非対称暗号化の場合、各当事者はプライベート鍵と公開鍵を持っています。

非対称暗号化の原則は、送信者が受信者の公開鍵を使用して受信者が秘密鍵で解読できるメッセージを暗号化できるようにします。または、送信者はその秘密鍵を使用してメッセージのデジタル署名を生成でき、受信者は送信者の公開鍵を使用してこのメッセージを検証できます。

2.4 ハッシュ関数

ハッシュ関数は、可変長入力に対して固定長出力(“ハッシュ値”または“ダイジェスト”)を生成します。暗号ハッシュ関数は、その特性に関連してより強い要件を持つ特別な種類のハッシュ関数です。

ハッシュ関数への入力データとともにハッシュ値を使用して、入力データまたはハッシュ値の完全性をチェックできます。

2.5 メッセージ認証コード (MAC)

MAC は、暗号ハッシュ関数または対称暗号に基づいています。メッセージの完全性に加えて、送信者と受信者の間で共有される秘密のために、メッセージの信頼性を保証できます。

2.6 デジタル署名

デジタル署名は、暗号ハッシュ関数と組み合わせて非対称暗号を使用することによって生成されます。これらは、完全性、真正性、否認防止の3つのセキュリティ機能を提供します。つまり送信者は、送信者のプライベート鍵を使用して署名を生成し、受信者が送信者の公開鍵を使用して、これを証明できます。

3 Crypto ドライバ

3.1 ドライバアーキテクチャ

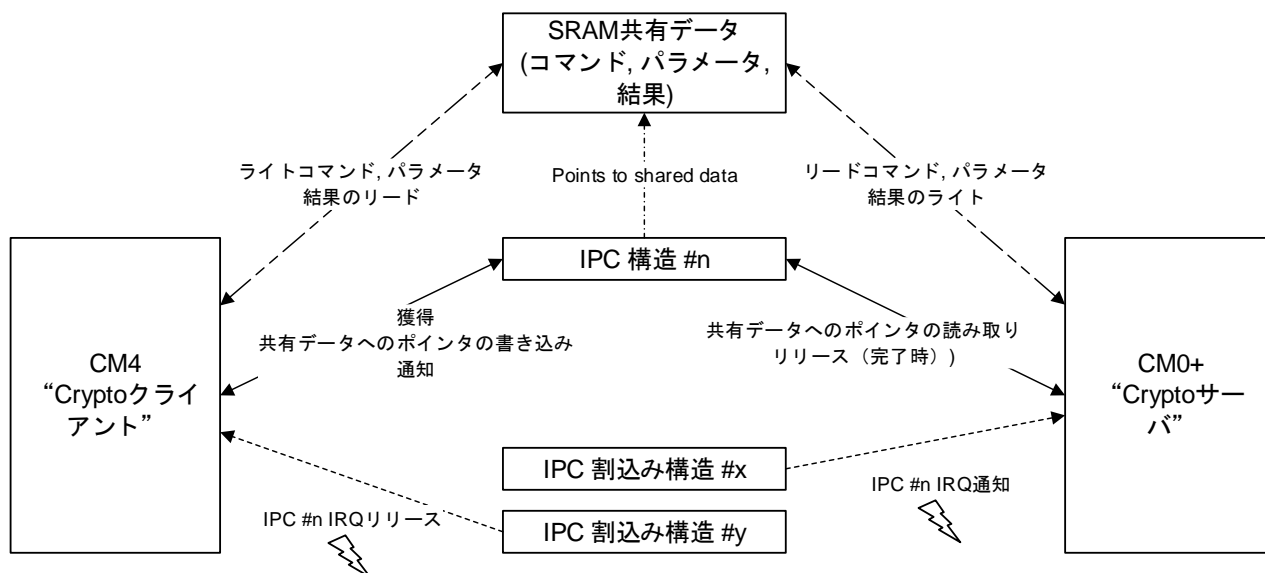
Crypto ドライバの設計は、クライアントサーバアーキテクチャに基づいています。Crypto サーバは CM0+コアでのみ動作し、適切な Crypto Core 関数を実行することで Crypto ハードウェアと連携します。Crypto クライアントはどちらのコアでも実行できますが、このアプリケーションノートでは、クライアントが CM4 コアで動作するケースについてのみ説明します。CM4 を参照するすべての説明と図は、CM7 を搭載した Traveo II デバイスにも適用されます。クライアントとサーバは、プロセッサ間通信 (IPC) を介して通信します。通信に IPC を使用すると、異なるコアからの同時要求を処理するシンプルな同期メカニズムが提供されます。

Crypto ドライバは、CRYPTO ブロック以外の次のハードウェアリソースを使用します。

- クライアントとサーバ間のデータ交換用の 1 つの IPC チャンネル構造
- 通知のための 2 つの IPC 割込み構造
- ハードウェアエラーのハンドリング用に 1 つの割込み

Crypto サーバとクライアント間の一般的な通信の概念を図 1 に示します。

図 1. IPC 経由のサーバ/クライアント通信例



3.2 ドライバの初期化

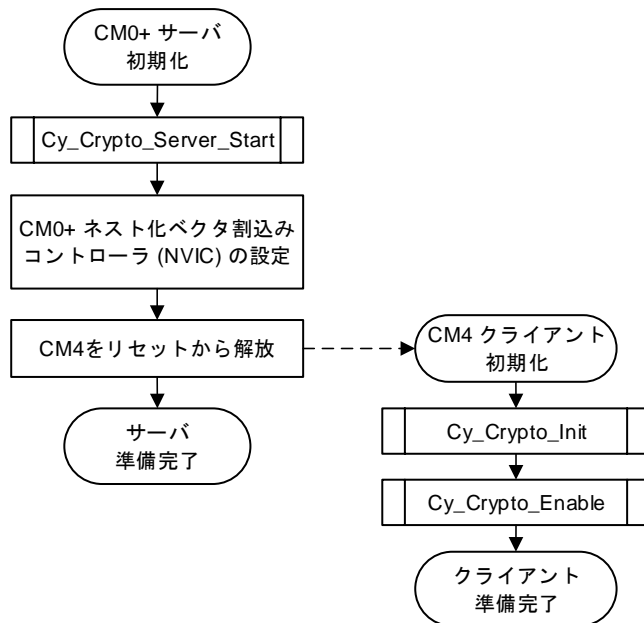
ドライバを暗号操作に使用する前に、正しい順序でドライバを初期化する必要があります。

1. Crypto サーバは CM0+で初期化する必要があります。
2. Crypto サーバの初期化が完了すると、クライアントは CM4 によって正常に初期化されます。

Note: Crypto サーバは CM0+上で動作し、クライアントは CM4 上で実行できます。そして、両方のコアは互いに異なる周波数で独立して実行されるため、クライアントの初期化時にサーバの初期化が完了しないことがあります。理想的には、この状況を回避するために、CM0+は Crypto サーバ初期化後、CM4 のリセットを解除します。あるいは、クライアント初期化の前に、サーバの状態を確認できます。

図 2 は、Crypto サーバとクライアントの初期化を示します。

図 2. サーバ / クライアントの初期化フローチャート例



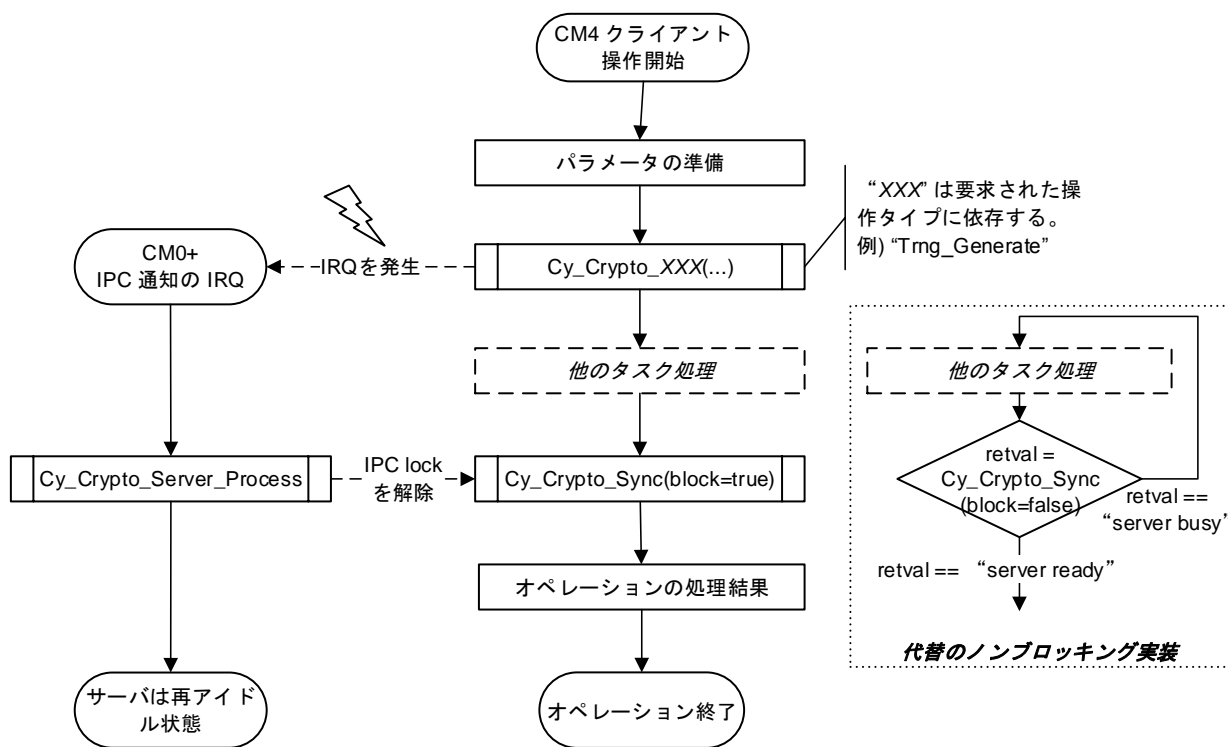
3.3 ドライバの使い方

サーバとクライアントの初期化が完了すると、クライアントはサーバから暗号操作を要求できます。CM0+で実行されているサーバが要求された操作を処理している間、CM4は他のタスクを処理できます。操作の終了は、3つの異なる方法で決定できます。

1. Crypto ドライバ関数 `Cy_Crypto_Sync` をブロッキングモードで呼び出します。関数が戻ると、クライアントは前の操作が完了したことを確認できます。
2. Crypto ドライバ関数 `Cy_Crypto_Sync` をノンブロッキングモードで定期的呼び出し、戻り値をチェックします。戻り値がサーバの準備ができていていることを示している場合、前操作は完了です。
3. クライアントは、IPC リリース割り込みを設定して、サーバ操作が完了した直後に通知を受け取れます。

このアプリケーションノートでは、第 1 の方法（`Cy_Crypto_Sync` をブロッキングモードで呼び出すこと）が使用され、関数パラメータと "block = true" 値によってもフローチャートに示されています。図 3 は、初期化された Crypto ドライバの一般的な使い方を示します。

図 3. 一般的なドライバの使用法のフローチャート



4 巡回冗長検査 (CRC)

4.1 ユースケース

メモリ領域の CRC 結果 (たとえば、フラッシュのパラメータ) を格納できます。ソフトウェアは、環境の影響やハードウェアの欠陥などの原因によって領域の内容が壊れていないことを保証します。

Note: CRC は暗号操作ではないため、このような領域を悪意のある変更から保護するために使用しないでください。攻撃者は変更された領域の CRC 結果を簡単に計算できます。

4.2 ドライバ関数

Crypto ドライバは、次の CRC 関連の関数を提供します。

4.2.1 Cy_Crypto_Crc_Init

この関数は、基本的な CRC 関連の設定を初期化し、後で複数の CRC 計算で再利用できます。広く使用されている CRC32 や CRC16-CCITT などの多くの CRC 標準は、使用されている多項式 (および長さ) と入力/出力設定が異なります。これらの設定は、Cy_Crypto_Crc_Init を呼び出すことによって定義されます。

4.2.2 Cy_Crypto_Crc_Run

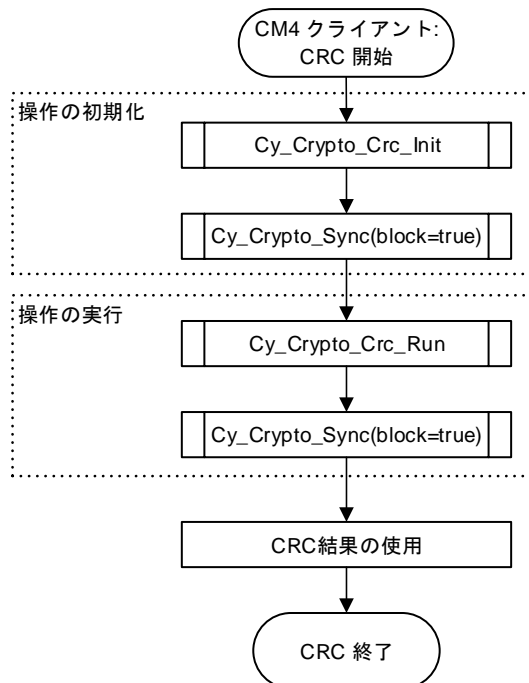
実際の CRC 計算は、この関数を呼び出すことによって要求されます。計算、開始アドレス、およびサイズの初期値が関数に渡され、CRC 結果が返されます。

CRC 計算の初期値は、使用する CRC 標準によって定義され、新しい CRC 計算の開始に適用されます。CRC が複数の非連続メモリ領域にわたって計算される場合、Cy_Crypto_Crc_Run は複数回呼び出されなければならず (現在の操作が終了した後)、前のブロックの結果を次のブロックの初期値として使用する必要があります。

4.3 フローチャート

図 4 は、Crypto ドライバを使用して CRC 値を計算する一般的なフローを示します。

図 4. CRC 計算のフローチャート



Note: CRC 演算の繰り返し使用のために、関連パラメータ (例えば多項式) の 1 つが変化しない限り、Cy_Crypto_Crc_Init を再度呼び出す必要はありません。

5 擬似乱数生成器 (PRNG)

5.1 ユースケース

擬似乱数生成器を使用して、以下の生成が可能です。

- 通信セッションの対称キー
- "暗号ソルト(cryptographic salt)"、辞書内のすべての単語に対してあらかじめ計算されたハッシュ値の巨大なデータベースを使用する辞書攻撃を防ぐためのハッシュ関数用の追加のランダム入力用データ
- チャレンジレスポンス認証プロトコルにおける"チャレンジ"値

Note: PRNG シード値は、強力なセキュリティ特性を提供するために、エントロピーが高い真性のランダム値で初期化する必要があります。その目的で TRNG を使用できます。非ランダムまたは一定のシード値は、ソフトウェアの開発やテストなどの一時的な状況で再現可能な PRNG 出力などの決定的な動作が必要な場合にのみ使用してください。

5.2 ドライバ関数

Crypto ドライバは、PRNG に関連する次の関数を提供します。

5.2.1 Cy_Crypto_Prng_Init

この関数は、PRNG を構成する 3 つの線形フィードバックシフトレジスタを初期化するためにシード値を決定します。

5.2.2 Cy_Crypto_Prng_Generate

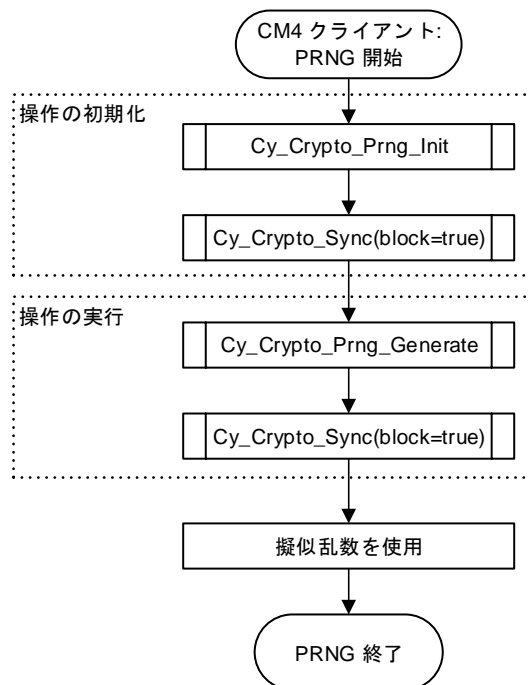
この関数を呼び出すことによって擬似乱数が生成されます。擬似乱数の上限は、関数が 0 から指定された制限の間の数値を返すように指定できます。上限は $2^{32}-1$ です。

Note: `Cy_Crypto_Prng_Init` は、後でシード値を変更する場合にのみ呼び出す必要があります。複数の乱数が必要な場合は、`Cy_Crypto_Prng_Generate` を呼び出すだけで十分です。

5.3 フローチャート

図 5 は、Crypto ドライバを使用して疑似乱数を生成する一般的なフローを示します。

図 5. 疑似乱数生成のフローチャート



Note: `Cy_Crypto_Prng_Init` は、シード値が変更された場合にのみ呼び出される必要があります。PRNG 値の繰り返し生成のためにこの関数を呼び出す必要はありません。

6 真性乱数生成器 (TRNG)

6.1 ユースケース

TRNG は通常、PRNG のエントロピーが高い真性のランダムシード値を生成するためにのみ使用されます。これは消費電流が比較的高く、乱数ビットがかなりゆっくりと生成されるためです。

6.2 ドライバ関数

Crypto ドライバは、次の TRNG 関連の関数を提供します。

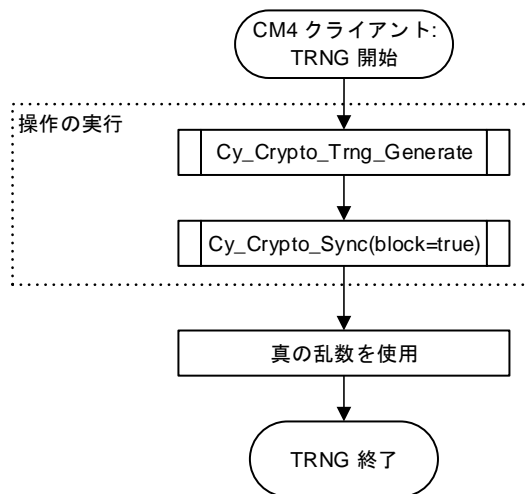
6.2.1 `Cy_Crypto_Trng_Generate`

この関数は、指定されたビット長の真性の乱数を生成します。この関数を呼び出すときには、物理的な構成 (リングオシレータ) に関連するいくつかのパラメータを記述する必要があります。

6.3 フローチャート

図 6 は、真性の乱数を生成するために Crypto ドライバを使用する一般的なフローを示します。

図 6. 真性乱数生成のフローチャート



7 Advanced Encryption Standard(AES)を使用した対称鍵暗号化

Crypto ブロックはいくつかの対称鍵暗号化規格をサポートし、Advanced Encryption Standard (AES) は最も広く普及しています。

AES 動作自体は、128 ビットの単一のデータブロックに基づいています。複数の関連データブロックが暗号化されている場合、NIST (National Institute of Standard and Technology) Special Publication 800-38A などの科学文献に記載されているブロックチェーンモードの 1 つを使用することをお勧めします。それぞれのブロックを独立して暗号化することは、通常、不十分なセキュリティレベルを提供します。たとえば、同じプレーンテキストのブロックも同じ暗号テキストを持つ場合があります。

Crypto ドライバは、次の AES 動作モードをサポートしています。

- ECB (Electronic Code Book Mode)
このモードでは、連鎖することなく独立してブロックを暗号化/復号化するため、関連するデータの複数のブロックに低レベルのセキュリティが提供されます。
- CBC (Cipher Block Chaining Mode)
1 つのブロックの暗号文は、次のブロックのプレーンテキストと結合されます。最初のブロックに先行するブロックがないので、いわゆる初期化ベクトル (IV) が必要であり、それが最初のブロックのプレーンテキストと組み合わせられるためです。
- CFB (Cipher Feedback Mode)
このモードは、任意の長さのプレーンテキストのストリーム暗号として使用できます。プレーンテキストは AES 暗号化操作の出力と XOR されます。暗号文は、後続のブロックの暗号化入力として使用されます。最初のブロックには、CBC モードと同様の初期化ベクトルが必要です。
- CTR (Counter Mode)
カウンタ動作モードは、ストリーム暗号としても使用されます。ここでも、プレーンテキストは AES 暗号化操作の出力と XOR されます。AES 暗号化への入力は、いわゆるノンスとカウンタの組み合わせ、または連結です。カウンタはブロックごとに変化し、最も単純なケースでは、それは単に増分されているだけです。暗号文脈におけるノンスは、通常、暗号操作のために一度だけ使用される任意の数です。

7.1 ユースケース

AES の性能要件は非対称鍵暗号の場合と比べはるかに低いため、通常、暗号鍵を持つデバイス間の安全な通信に使用されます。対照的に、非対称鍵暗号化の多くは通信プロトコルによって関係するすべての当事者間で鍵を交換することで通信セッションを確立するためだけに使用されます。この鍵は、対称暗号化と復号化に使用されます。

7.2 ドライバ関数

Crypto ドライバは、以下の AES 関連の関数を提供します。AES の初期化は不要です。

7.2.1 Cy_Crypto_Aes_Ecb_Run

この関数は、単一の 128 ビットデータブロックを暗号化または復号化します。ユーザは、方向（暗号化/復号化）、鍵の場所とサイズ、および送信元と送信先のブロックの場所を指定するよう要求されます。

7.2.2 Cy_Crypto_Aes_Cbc_Run

Cy_Crypto_Aes_Ecb_Run に加えて、この関数は初期化ベクトルと暗号化/復号化されるデータの合計サイズを求めます。

7.2.3 Cy_Crypto_Aes_Cfb_Run

この関数は、Cy_Crypto_Aes_Cbc_Run と同じ引数をとります。

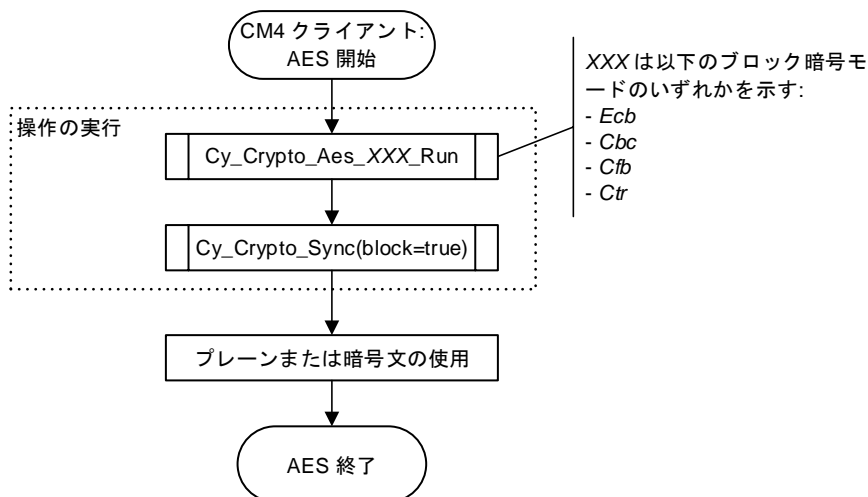
7.2.4 Cy_Crypto_Aes_Ctr_Run

この関数は、Cy_Crypto_Aes_Cbc_Run と同じ引数をとります。初期化ベクトルは、開始カウンタ値と組み合わせたノンスです。ドライバは、新しいブロック暗号化ごとにカウンタ部分を内部的にインクリメントします。

7.3 フローチャート

図 7 は、Crypto ドライバを使用して AES 操作を実行する一般的なフローを示します。

図 7. AES 操作のフローチャート



7.4 他の対称鍵アルゴリズム

一般的な AES アルゴリズムに加えて、Crypto ドライバとハードウェアは次のアルゴリズムもサポートしています。

- DES(データ暗号化標準)
Cy_Crypto_Des_Run 関数は、Crypto ドライバによって提供されます。
- TDES(トリプル DES, 3DES とも略されます)
対応するドライバ関数は Cy_Crypto_Tdes_Run です。
- Chacha
Chacha は、Cy_Crypto_Chacha_Run ドライバ関数によってサポートされています。

8 SHA 暗号ハッシュ関数ファミリ

SHA は Secure Hash Algorithm の略で、標準化された暗号ハッシュ関数です。Crypto ドライバは、SHA-1, SHA-256, SHA-512, SHA-512/256, SHA3-256, SHA3-512, SHAKE256 などの標準化されたすべてのバリエーションをサポートしています。

8.1 ユースケース

SHA は、実際のプレーンテキストパスワードを直接格納する代わりに、ユーザパスワード(理想的には「暗号ソルト値」と組合せ)からハッシュ値を導き出し、このハッシュ値(ソルトと共に)をデータベースに格納するためによく使用されます。セキュリティ違反が発生した場合、データベースは他のシステムへのログインに使用されるユーザパスワードを明らかにしません。

SHA は、他の暗号化ブロックと組み合わせて使用して、より高いレベルのセキュリティ目標を達成します。例えば、ハッシュベースのメッセージ認証コード(9を参照)またはデジタル署名(11を参照)です。

8.2 ドライバ関数

Crypto ドライバは、次の SHA 関連の関数を提供します。SHA の初期化は必要ありません。

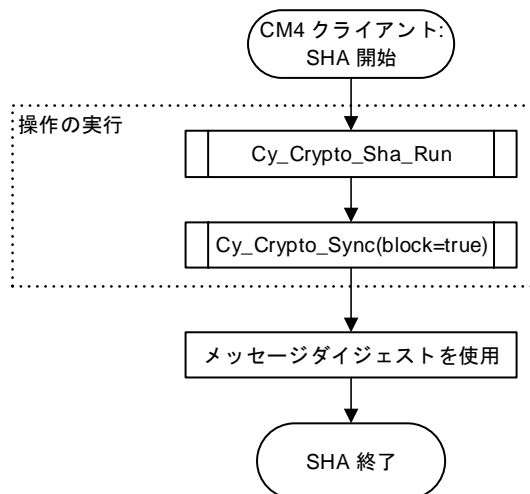
8.2.1 Cy_Crypto_Sha_Run

この関数は、選択された SHA モードごとにメッセージダイジェストを生成します。これは、関数を呼び出すときに指定する必要があります。さらに、メッセージの場所、サイズ、およびダイジェストを格納する場所は、関数によって求められます。

8.3 フローチャート

図 8 は、SHA 操作を実行するために Crypto ドライバを使用する一般的なフローを示します。

図 8. SHA 操作のフローチャート



9 ハッシュベースおよび暗号ベースのメッセージ認証コード (HMAC, CMAC)

メッセージ認証コード(MAC)は、暗号化ハッシュ関数に似ていますが、セキュリティ要件は異なります。これらのセキュリティ要件は、MAC を生成するための基礎として、暗号ハッシュ関数またはブロック暗号のいずれかと共に秘密暗号鍵を使用することによって達成できます。この操作は、ハッシュベースの MAC(HMAC)または暗号ベースの MAC(CMAC)と呼ばれます。Crypto ドライバは、HMAC 操作には暗号ハードウェアの SHA ブロックを使用し、CMAC 操作には AES ブロックを使用します。

9.1 ユースケース

MAC は、メッセージの受信者がそのメッセージが本物の送信者から発信されたことを知ることが重要であるときに使用されます。一例は、自動車車内のネットワーク中の分散された電子制御ユニットが、すべての重要なメッセージに対して MAC を使用して攻撃者がメッセージをネットワークに注入できないことを保証するケースです。これは、MAC 生成および検証プロセスで使用される送信側と受信側の間の共有秘密鍵によって実現されます。

9.2 ドライバ関数

Crypto ドライバは、次の MAC 関連の関数を提供します。MAC 操作の初期化は必要ありません。

9.2.1 Cy_Crypto_Hmac_Run

この関数を呼び出す場合、目的の SHA モード、キーの格納先とサイズ、メッセージの場所とサイズ、計算された MAC が格納される場所を渡す必要があります。

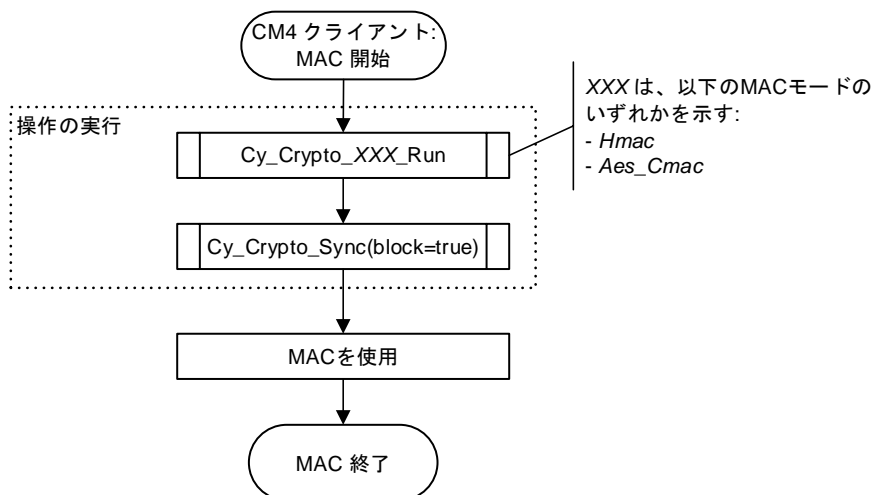
9.2.2 Cy_Crypto_Cmac_Run

この関数は、Cy_Crypto_Hmac_Run と同様のパラメータを求めますが、SHA モードを使用します。

9.3 フローチャート

図 9 は、Crypto ドライバを使用して MAC 操作を実行する一般的なフローを示します。

図 9. MAC 操作のフローチャート



10 RSA を使用した非対称鍵暗号化

非対称鍵暗号化(公開鍵暗号としても知られている)のための様々なアルゴリズムが過去数十年に提案され、一般に使用されています。最も一般的なものの 1 つは、その発明者 Rivest, Shamir, Adleman によって命名された RSA です。Crypto ドライバは RSA アルゴリズムをサポートしています。

RSA と SHA に基づいてデジタル署名を検証するには、RSA と SHA に関連する Crypto ドライバ API を組み合わせて使用する必要があります。RSA と SHA を使用したデジタル署名検証を参照してください。

10.1 ユースケース

非対称暗号化を使用する暗号化と復号化は、通常、対称暗号化または復号化よりもはるかに低速です。これは、関係するすべての当事者間で共通鍵を交換するための通信チャネルの確立中にのみ使用されることが多い理由です。

10.2 ドライバ関数

Crypto ドライバは、RSA 暗号化および復号化に関連する次の関数を提供します。

10.2.1 Cy_Crypto_Rsa_InvertEndianness

プレーンテキスト, 暗号文, モジュラス, パブリックおよびプライベート指数などの RSA 操作に必要なマルチバイトパラメータは、通常ビッグエンディアン形式で定義されます。Crypto ドライバは内部においてリトルエンディアン形式で動作するため、すべての入力パラメータをリトルエンディアン形式で使用可能にする必要があります。この変換は、実行時に Cy_Crypto_Rsa_InvertEndianness を呼び出してソフトウェアで行えます。Crypto ドライバは、リトルエンディアン形式の Cy_Crypto_Rsa_Proc の出力(プレーンテキストまたは暗号テキスト)を格納します。この関数を使用して、必要に応じて出力をビッグエンディアン形式に変換できます。

10.2.2 Cy_Crypto_Rsa_CalcCoefs

この関数は、RSA 処理を高速化する RSA 計算で使用する特定の係数とパラメータを事前に計算できます。計算係数は、RSA 関連の暗号化ドライバコンテキストに添付され、モジュラスなどの依存パラメータが変更されない限り、複数の RSA オペレーションにわたって再利用できます。

RSA 操作の前処理中に関数が呼び出されない場合、Cy_Crypto_Rsa_Proc は呼び出されるごとにこれらパラメータを内部的に計算します。

10.2.3 Cy_Crypto_Rsa_Proc

これは、RSA を使用してプレーンテキストを暗号化することや、暗号テキストを復号する主な関数です。

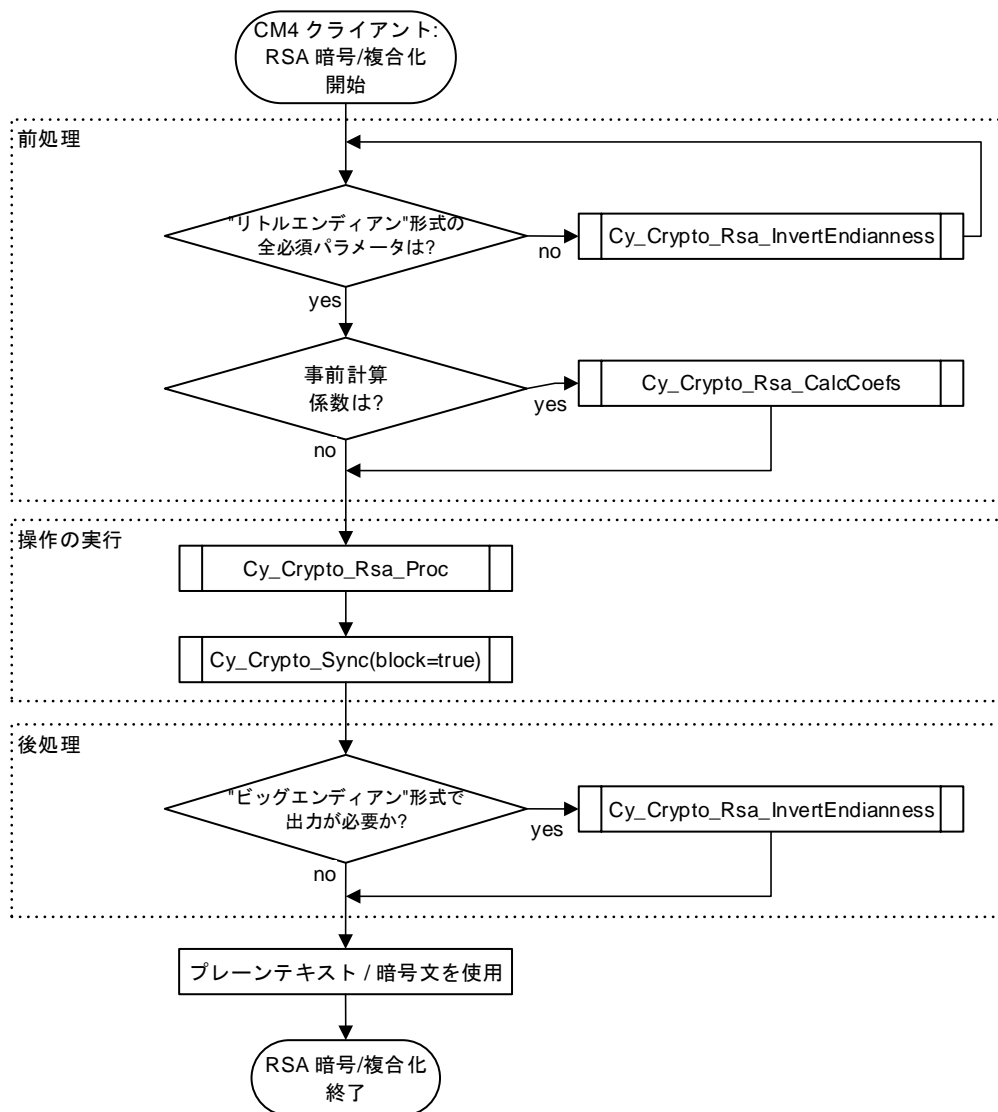
関数インターフェースは汎用です。入力メッセージの配置とそのサイズ、出力メッセージの位置、モジュラス、キー(すなわち指数)、およびオプションで Cy_Crypto_Rsa_CalcCoefs によって計算された係数の位置を要求します。

Cy_Crypto_Rsa_Proc 関数は、提供された入力メッセージをモジュロと指数で処理し、その出力を出力メッセージの場所に格納します。提供される入力メッセージのタイプ(プレーンテキストまたは暗号テキスト)と提供される指数の種類(公開または非公開)によって異なります。この関数は、暗号化の有無を問わず、内部的には操作を区別しません。

10.3 フローチャート

図 10 に、RSA の暗号化および復号化操作に Crypto ドライバを使用する一般的なフローを示します。

図 10. RSA 暗号化と復号化操作のフローチャート



11 RSA と SHA を使用したデジタル署名検証

デジタル署名は広く使用されており、しばしば RSA と暗号強度を備えたハッシュ関数に基づいています。

公開鍵暗号標準(PKCS)は、非対称鍵暗号のさまざまな側面を標準化しようとする仕様を集めたものです。これらの側面の 1 つは、デジタル署名の生成および検証方法を定義することです。Crypto ドライバは RSASSA-PKCS1-v1_5 に基づくデジタル署名の検証のみをサポートし、SHA 暗号化ハッシュ関数を使用します。より詳細な情報については、次のドキュメントを参照してください。

<http://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf>

11.1 ユースケース

デジタル署名のアプリケーションの 1 つは、インストールまたは実行前のファームウェアイメージの検証です。例えば、ファームウェア更新プロセスは、安全でない通信チャネルを介してファームウェアイメージおよび関連するデジタル署名を受信します。更新プロセスは、生産中に製造業者によってデバイスにプログラムされた公開鍵を使用してデジタル署名を検証することにより、ファームウェアイメージの信頼性および起点を証明できます。

11.2 ドライバ関数

Crypto ドライバは、RSA および SHA を使用してデジタル署名を検証することに関連する次の関数を提供します。

11.2.1 Cy_Crypto_Rsa_InvertEndianness

10.2.1 を参照してください。

11.2.2 Cy_Crypto_Rsa_CalcCoefs

10.2.2 を参照してください。

11.2.3 Cy_Crypto_Rsa_Proc

10.2.3 を参照してください。

デジタル署名を検証するには、この関数と署名者の公開鍵を使用して署名を復号化する必要があります。

11.2.4 Cy_Crypto_Sha_Run

8.2.1 を参照してください。

署名されたメッセージのダイジェストを計算するには、デジタル署名の生成に使用されたものと同じ SHA モードを使用する必要があります。

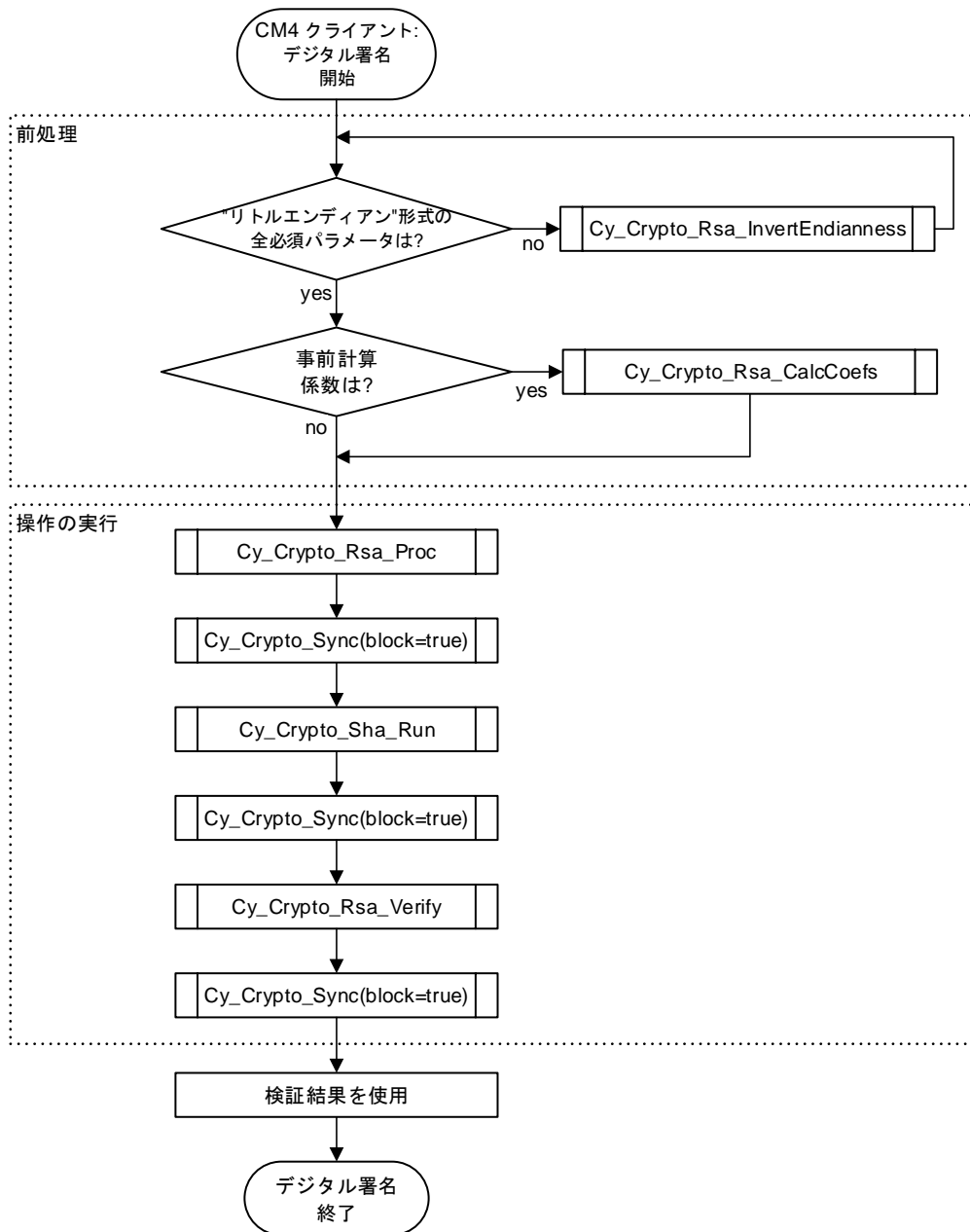
11.2.5 Cy_Crypto_Rsa_Verify

この関数は実際の検証を行います。 解読されたデジタル署名を受け取り、サポートされている RSASSA-PKCS1-v1_5 標準に従ってその署名の形式を検証し、提供されたメッセージダイジェストがデジタル署名に含まれているものと一致するかどうかをチェックします。

11.3 フローチャート

図 11 は、RSA および SHA 操作を使用してデジタル署名を検証するための、Crypto ドライバの使用法の一般的なフローを示します。

図 11. RSA および SHA 操作によるデジタル署名の検証のフローチャート



改版履歴

文書名: AN220253 - Traveo II ファミリでの CRYPTO モジュールの使用方法

文書番号: 002-24018

版	ECN 番号	変更者	発行日	変更内容
**	6233985	KUME	07/13/2018	これは英語版 002-20253 Rev. **を翻訳した日本語版 002-24018 Rev. **です。

ワールドワイド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	cypress.com/usb
ワイヤレス	cypress.com/wireless

PSoC® ソリューション

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

サイプレス開発者コミュニティ

[コミュニティ](#) | [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

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



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

© Cypress Semiconductor Corporation, 2018. 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、明示又は黙示を問わず、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。いかなるコンピューティングデバイスも絶対に安全ということはない。従って、Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず、Cypress は、Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて、本書面に記載された製品には、エラーと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり、公表された仕様とは異なる動作をする場合がある。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問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 製品の本来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, CapSense, 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。