参考資料

User's Guide

MSPM33C セキュリティ ユーザー ガイド

TEXAS INSTRUMENTS

概要

このドキュメントは、アプリケーション開発時に MSPM33C デバイス ファミリに組み込まれているセキュリティ機能を利用するためのガイドとして活用できます。セキュリティ機能には TrustZone® と複数のセキュリティ アクセラレータが含まれます。セキュア ブートまたはセキュア デバッグの実装の詳細については、M33 TRM を参照してください。

目次

1 セキュリティ機能の概要 2
1.1 用語 2
2 セキュアな実行環境 3
2.1 TrustZone® 3
2.1.1 実装定義属性ユニット 3
2.1.2 セキュリティ属性単位 3
2.1.3 TrustZone® ソフトウェア開発 4
2.2 メモリ保護ユニット 4
2.2.1 TrustZone® および MPU 5
2.3 グローバル セキュリティ コントローラ 5
2.3.1 GSC メモリ構成 5
3 セキュリティ モジュール 7
3.1 AES 7
3.1.1 AES の概要 7
3.1.2 AES の使用法 8
3.2 キーストア 9
3.2.1 概要 9
3.2.2 キーストアの使用方法 9
3.3 SHA2 9
3.3.1 SHA の概要 10
3.3.2 SHA の性能 10
3.3.3 SHA の使用法 10
3.4 PKA 11
3.4.1 PKA の概要 11
3.4.2 PKA の使用法 11
3.5 PQC 12
3.5.1 ML-DSA 12
4 改訂履歴 12

商標

TrustZone®, Arm®, and Cortex® are registered trademarks of Arm Limited.
すべての商標は、それぞれの所有者に帰属します。

Copyright © 2026 Texas Instruments Incorporated

1 セキュリティ機能の概要

MSPM33C デバイス ファミリが搭載しているセキュリティ機能は、デバイスに展開したセキュア データの整合性と真正性を維持するためのフレームワークを構築します。

MSPM33C デバイス ファミリに搭載されている主なセキュリティ機能は次のとおりです。

- セキュアな実行環境
  - TrustZone®
    - 実装定義属性ユニット (IDAU)
    - セキュリティ属性ユニット (SAU)
  - メモリ保護ユニット (MPU)
    - TrustZone® および MPU
  - グローバル セキュリティ コントローラ (GSC)
    - SRAM 保護コントローラ (SPC)
    - フラッシュ保護コントローラ (FPC):
    - ペリフェラル保護コントローラ (PPC)
- セキュリティ アクセラレータ
  - 高度暗号化標準 (AES)
  - キーストア
  - セキュア ハッシュ アルゴリズム (SHA)
  - 公開鍵アクセラレータ (PKA)
  - ポスト量子暗号 (PQC)

1.1 用語

略称 / 用語	定義
GSC	グローバル セキュリティ コントローラ
PKA	公開キー アクセラレータ
SRAM	シンクロナス ランダム アクセス メモリ
MMR	メモリ マップト レジスタ

Copyright © 2026 Texas Instruments Incorporated

2 セキュアな実行環境

TrustZone ®アーキテクチャは、Arm® Cortex®-M33 によって実現されます。TrustZone® を使用すると、プログラマはメモリのセクションをセキュアと非セキュアに分割でき、権限のないユーザーによるメモリのセクションへのアクセスを制限できます。さらに、MSPM33C デバイスにはグローバル セキュリティ コントローラ (GSC) が搭載されており、CPU サブシステム (CPUSS) の外部に追加のセキュリティを提供できます。これにより、DMA などの外部ペリフェラルが、非セキュア プロセッサ状態からメモリのセキュア領域にアクセスすることを防止できます。

2.1 TrustZone®

Cortex®-M33 TrustZone® セキュリティを使用すると、プログラマはメモリをセキュア、非セキュア、および非セキュア呼び出し可能の 3 種類の領域に分割できます。メモリのセキュア領域では、ユーザーは、メモリのすべての領域に制限なくアクセスできます。メモリの非セキュア領域では、プログラマは他の非セキュア領域または非セキュア呼び出し可能領域にのみアクセスするよう制限されます。メモリの非セキュア呼び出し可能領域は固有の領域です。ユーザーが SG instruction を呼び出してプログラマがメモリの安全な領域に遷移できるようにしない限り、メモリの非セキュア領域にのみアクセスが限定されます。詳細については、Armv8-M アーキテクチャの TrustZone® テクノロジーのドキュメントを参照してください。

セキュアおよび非セキュアなメモリ領域を定義するために、セキュリティ属性ユニット (SAU) および実装定義属性ユニット (IDAU) が使用されます。IDAU は RTL レベルに設定され、TI によって構成されています。IDAU の構成方法については、『MSPM33C3 シリーズ 160MHz マイコン テクニカル リファレンス マニュアル』の「CPU」セクションを参照してください。

SAU は、CPU サブシステム (CPUSS) によって使用され、メモリ領域をセキュアまたは非セキュアとして定義します。このペリフェラルにより、プログラマはコードをセキュアまたは非セキュアとして分割できます。SAU および IDAU がどのように連携して領域をセキュアまたは非セキュアとして定義するかについては、Armv8-M アーキテクチャの TrustZone® テクノロジーのドキュメントを参照してください。

2.1.1 実装定義属性ユニット

実装定義属性ユニット (IDAU) は TI によって定義されており、『MSPM33C3 シリーズ 160MHz マイコン テクニカル リファレンス マニュアル』の「CPU」セクションに説明があります。IDAU の構成は SAU と一緒に使用され、領域の最終的なセキュリティ レベルが構成されるため、IDAU の構成方法を知ることが重要です。たとえば、MSPM33C32 デバイスでは、アドレスの 8 番目のビットが 1 のとき、メモリ領域は非セキュア呼び出し可能として定義され、0 のときは非セキュアとしてのみ定義されるように IDAU が構成されています。たとえば、アドレス 0x1000.0000 は NSC として定義されます。

IDAU と SAU の両方を使用して、最終的なセキュリティ レベルが決定されます。この 2 つのうち、より高いセキュリティ レベルが、最終的なセキュリティ属性レベルとして設定されます。MSPM33C321A における例については、図 2-1 を参照してください。

Memory Location	IDAU	SAU	Final Security Level
0x0000.0000 to 0x00FF.FFFF	Non-Secure	Non-Secure	Non-Secure
0x1000.0000 to 0x10FF.FFFF	Non-Secure Callable	Non-Secure Callable	Non-Secure Callable
0x2000.0000 to 0x203F.FFFF	Non-Secure	Secure	Secure
0x3000.0000 to 0x303F.FFFF	Non-Secure Callable	Non-Secure	Non-Secure Callable
0x4000.0000 to 0x4FFF.FFFF	Non-Secure	Non-Secure callable	Non-Secure callable
0x5000.0000 to 0x5FFF.FFFF	Non-Secure Callable	Secure	Secure

図 2-1. MSPM33C321A の IDAU および SAU 構成例

2.1.2 セキュリティ属性単位

セキュリティ属性単位 (SAU) は、プログラマがメモリ領域のセキュリティを定義するために使用できます。最終的なセキュリティ レベルでは、SAU 構成と IDAU 構成の両方が使用されます。メモリのセクションのセキュリティ レベルを決定するには、SAU と IDAU のうち、より高いセキュリティ レベルを採用します。たとえば、SAU がある領域をセキュアとして定義し、IDAU がその領域を非セキュアとして定義している場合、その領域はセキュアとして定義されます。

Copyright © 2026 Texas Instruments Incorporated

MSPM33 デバイスでは、SAU は、8 つの領域をセキュア、非セキュア、または非セキュア呼び出し可能として定義するように構築されています。

領域をセキュアとして定義する

SAU の主要な用途の 1 つは、領域をセキュアまたは非セキュアとして定義することです。領域を定義するには、次の手順で SAU のレジスタに書き込みます。

1. SAU->RNR レジスタを使用して、構成する領域を選択します。
2. SAU>RBAR レジスタを使用して、領域のベース アドレスを選択します。
3. SAU->RLAR を設定して、領域のサイズを構成します。
4. すべての領域について、手順 1 ～ 3 を繰り返します。
5. SAU->CTL レジスタを構成して、構成したすべての領域に伝播します。

2.1.3 TrustZone® ソフトウェア開発

プログラマのコード開発に SAU の使用を統合するには、CMSE ライブラリを使用することをお勧めします。このライブラリは Arm® によって設計されており、ユーザーは関数をセキュアまたは非セキュアとして簡単に定義できます。このガイドラインの詳細については、ARMv8-M セキュア ソフトウェア ガイドラインを参照してください。

ARMv8-M instruction は、BXNS (Branch and Exchange Non-Secure) および SG (Secure Gateway) を使用して、非セキュア コードとセキュア コード間の遷移を行います。BXNS はセキュア コードから非セキュア コードへの移行に使用され、SG は非セキュアからセキュアへの遷移に使用されます。SG コマンドは、NSC 領域を使用してこの遷移を行います。これらの instruction をプログラミング環境で簡単に利用するために、CMSE ライブラリではこれらの関数を関数呼び出しに統合しています。

関数をセキュアおよび非セキュアとして定義する

CMSE ライブラリの主な機能の 1 つは、プログラマが関数をセキュアまたは非セキュアとして定義できることです。これにより、CPU 状態が決定され、プログラマはセキュア領域と非セキュア領域の間を遷移できます。CMSE ライブラリをプログラムに追加するには、次のヘッダーを使用します。

```
#include <arm_cmse.h>
```

コードに SG コマンドを実装するには、関数に cmse_nonsecure_entry 属性を指定します。この関数定義は、コードの先頭でセキュア ゲートウェイ ベニヤを呼び出します。これは、非セキュア コードがアクセス可能な、セキュア メモリの情報を使用する関数を定義する際に便利です。

```
__attribute__((cmse_nonsecure_entry)) secure_fxn()
{

}
```

セキュア コードから非セキュア関数を呼び出すには、BXNS 関数を使用できます。CMSE ライブラリを使用してこれをプログラミング環境に簡単に統合するには、関数に cmse_nonsecure_call 属性を指定してください。例として以下のコードブロックを参照してください。

```
__attribute__((cmse_nonsecure_call)) nonsecure_fxn()
{

}
```

2.2 メモリ保護ユニット

MSPM33C デバイスには、CPUSS にメモリ保護ユニット (MPU) が搭載されています。MPU では、プログラマはメモリ領域を特権付きまたは特権なしとして定義できます。これは、SAU が領域をセキュアおよび非セキュアとして定義する方法と同様に動作します。主な違いは、デバイスがスレッド モードかハンドラ モードかに基づいて特権モードが変わることです。

Copyright © 2026 Texas Instruments Incorporated

リセット状態から復帰すると、デバイスはスレッド モードになり、プロセッサまたはペリフェラルによって例外が発行されるとハンドラ モードに移行します。例外は常に特権モードで動作し、スレッド モードでは MPU を使用して特権レベルを決定できます。詳細については、『MSPM33C3 シリーズ 160MHz マイコン テクニカル リファレンス マニュアル』を参照してください。

2.2.1 TrustZone® および MPU

MSPM33C CPUSS では、MPU、SAU、IDAU を使用して、さまざまな権限とセキュリティ レベルを組み合わせた属性をメモリ領域に割り当てます。これにより、プログラマは特権とセキュリティ制限を使用して、アクセス権限を多層的に設定できます。図 2-2 に、プログラマが使用できる組み合わせの例を示します。デバイスがハンドラ モードの場合、プロセッサは常に特権状態になります。

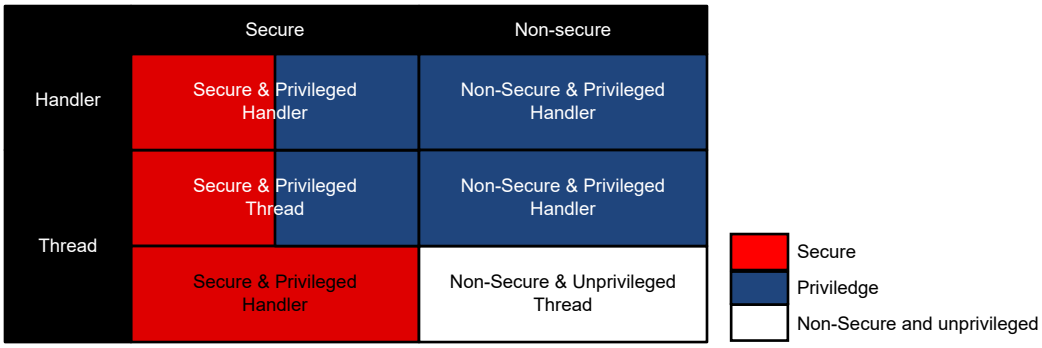


図 2-2. セキュリティと特権メモリの属性

2.3 グローバル セキュリティ コントローラ

グローバル セキュリティ コントローラ (GSC) は、MSP に対する独自のペリフェラルであり、TrustZone® アーキテクチャに、さらにセキュリティ層を追加します。GSC は、ペリフェラル、フラッシュ、および SRAM のセキュア属性と特権属性の両方を構成します。これにより、プログラマは 1 つの IP を使用してすべてのリソースを保護でき、セキュアでないペリフェラルがメモリのセキュア領域にアクセスするのを防ぎます。1 つの例として、非セキュアな DMA 読み取りセキュア メモリが挙げられます。GSC を使用することで、セキュアでない DMA がメモリのセキュア領域にアクセスしようとしたときにフォルトを発生させ、これらの攻撃からシステムを保護できます。GSC の詳細については、『MSPM33C3 シリーズ 160MHz マイコン テクニカル リファレンス マニュアル』の「グローバル セキュリティ コントローラ」の章を参照してください。

2.3.1 GSC メモリ構成

GSC は MPU と SAU の両方を利用し、セキュリティ属性および特権属性を用いてメモリを構成します。この構成を行うため、GSC はペリフェラル保護コントローラ (PPC)、SRAM 保護コントローラ (SPC)、フラッシュ保護コントローラ (FPC) の 3 つの個別のコントローラを備えています。レジスタの構成および使用方法の詳細については、『MSPM33C3 シリーズ 160MHz マイコン テクニカル リファレンス マニュアル』を参照してください

GSC は、SAU および MPU と同様の特性を持っています。これにより、メモリの保護にどのペリフェラルを使用すべきか混乱が生じる可能性があります。SAU および MPU は、構成可能な領域の数またはアドレスの粒度に制限されます。たとえば、M33 デバイスの SAU では、非セキュア属性とセキュア属性を 8 つのメモリ領域にのみ設定できます。未構成のメモリ領域は、セキュア属性のままとなります。ただし、GSC は 2kB サイズの領域に属性を割り当てます。1MB デバイス上で 255 の異なるセキュリティ領域を設定可能です。これらの 2kB 領域のそれぞれに、セキュア アクセスまたは特権アクセスのいずれかの属性が割り当てられます。

2.3.1.1 GSC および SAU によるセキュリティ例外

GSC と SAU の両方に、メモリ領域にセキュリティ属性を割り当てる機能があります。両者の主な違いの 1 つは、GSC がセキュア NMI を生成するのに対し、SAU はセキュア フォルトを生成することです。セキュア NMI はエラー集約モジュール (EAM) によって処理されるため、プログラマはエラーに関する特定の詳細を参照できます。エラーを引き起こしたペリフェラルや、エラーが発生したアドレスなどです。また、GSC は 8 つの領域に限定されません。フラッシュ、SRAM、およびペリフェラルは、セキュリティ属性に対して独自の粒度を持っています。詳細については、『MSPM33C3 シリーズ 160MHz マイコン テクニカル リファレンス マニュアル』を参照してください。

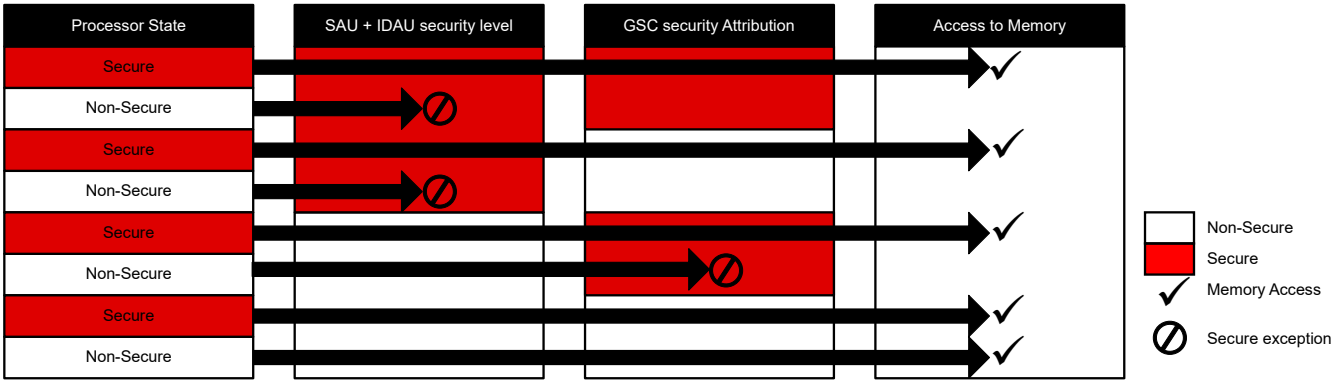
Copyright © 2026 Texas Instruments Incorporated

GSC および SAU を使用して、メモリの領域にセキュアまたは非セキュアの属性を割り当てるときに発生するエラーについては、表 2-1 を参照してください。また、これについての画像を「GSC および SAU でのセキュリティ例外」に示します。

表 2-1. GSC および SAU のセキュリティ属性割り当てに基づくセキュリティ割り込み

プロセッサの状態	SAU + IDAU セキュリティ属性割り当て	GSC セキュリティ属性割り当て	結果の割り込み	アクセス違反
セキュアな機能	セキュアな機能	セキュアな機能	なし	なし
非セキュア	セキュアな機能	セキュアな機能	セキュリティ故障	あり (CPU によってブロック)
セキュアな機能	セキュアな機能	非セキュア	なし	なし
非セキュア	セキュアな機能	非セキュア	セキュリティ故障	あり (CPU によってブロック)
セキュアな機能	非セキュア	セキュアな機能	なし	なし
非セキュア	非セキュア	セキュアな機能	セキュリティ NMI	あり (GSC によってブロック)
セキュアな機能	非セキュア	非セキュア	なし	なし
非セキュア	非セキュア	非セキュア	なし	なし

GSC および SAU でのセキュリティ例外



2.3.1.2 GSC および MPU による特権例外

GSC と MPU を組み合わせて使用すると、メモリ領域を特権または非特権として定義できます。これは、GSC および SAU がメモリ領域をセキュアまたは非セキュアとして定義する方法と同様です。これにより、プログラマは特権アクセスを使用してメモリ領域に別の保護層を追加することができます。特権状態は例外の発生によっても変化するため、割り込みが発生したときに期待される特権アクセスが正しく設定されていることを確認してください。

GSC および MPU を使用してメモリの領域に特権または非特権の属性を割り当てるときに、どちらのフォルトが優先的に発生するかを判断するには、表 2-2 を使用します。

表 2-2. GSC および MPU による特権属性割り当てに基づく特権例外

プロセッサの状態	MPU 特権属性割り当て	GSC 特権属性割り当て	結果の割り込み	アクセス違反
特権	特権	特権	なし	なし
非特権	特権	特権	MemMange フォルト	あり (CPU によってブロック)
特権	特権	非特権	なし	なし
非特権	特権	非特権	MemMange フォルト	あり (CPU によってブロック)
特権	非特権	特権	なし	なし
非特権	非特権	特権	特権 NMI	あり (GSC によってブロック)
特権	非特権	非特権	なし	なし
非特権	非特権	非特権	なし	なし

Copyright © 2026 Texas Instruments Incorporated

3 セキュリティ モジュール

M33C デバイス ファミリに搭載されている複数のモジュールを利用することで、セキュアな実行環境を実装できます。セキュリティ モジュールには以下が含まれています。

- 高度暗号化標準 (AES)
- キーストア
- セキュア ハッシュ アルゴリズム (SHA)
- 公開鍵アクセラレータ (PKA)
- ポスト量子暗号 (PQC)
  - ML-DSA

3.1 AES

AES アクセラレータ モジュールは、FIPS PUB 197 AES に基づく暗号化および復号化処理をハードウェアで高速化します。

3.1.1 AES の概要

AES アクセラレータ モジュールは、ハードウェアで 128 ビットまたは 256 ビットのキーを使用して、128 ビットのデータ ブロックの暗号化と復号化を実行します AES は、FIPS PUB 197 で規定されている対称キー ブロック暗号アルゴリズムです。

AES アクセラレータには、次のような機能があります。

- AES 128 ビット ブロックの暗号化および復号化
- ハードウェア内でのキー スケジューリング
- ENC /復号化のみのモード:CBC、CFB-1、CFB-8、CFB-128、OFB-128、CTR/ICM
- 認証専用モード:CBC-MAC、CMAC
- AES-CCM
- AES-GCM
- AES-CCM および AES-GCM モードは、ペイロード データのホールド / レジュームによる継続をサポートしています
- 32 ビット ワードのアクセスにより、キー データ、入力データ、および出力データを供給
- AES 準備完了割り込み
- 入出力データの DMA トリガ
- RUN モードと SLEEP モードをサポート (デバイスのテクニカル リファレンス マニュアルの「動作モード」セクションを参照)

図 3-1 に、AES エンジンのブロック概略図を示します。AES エンジンは、暗号化 / 復号処理に加えてガロア体乗算も実行する処理コアで構成されています。このコアは、ソフトウェアがメモリ マップト レジスタを介して設定する構成入力およびデータ入力によって駆動されます。

Copyright © 2026 Texas Instruments Incorporated

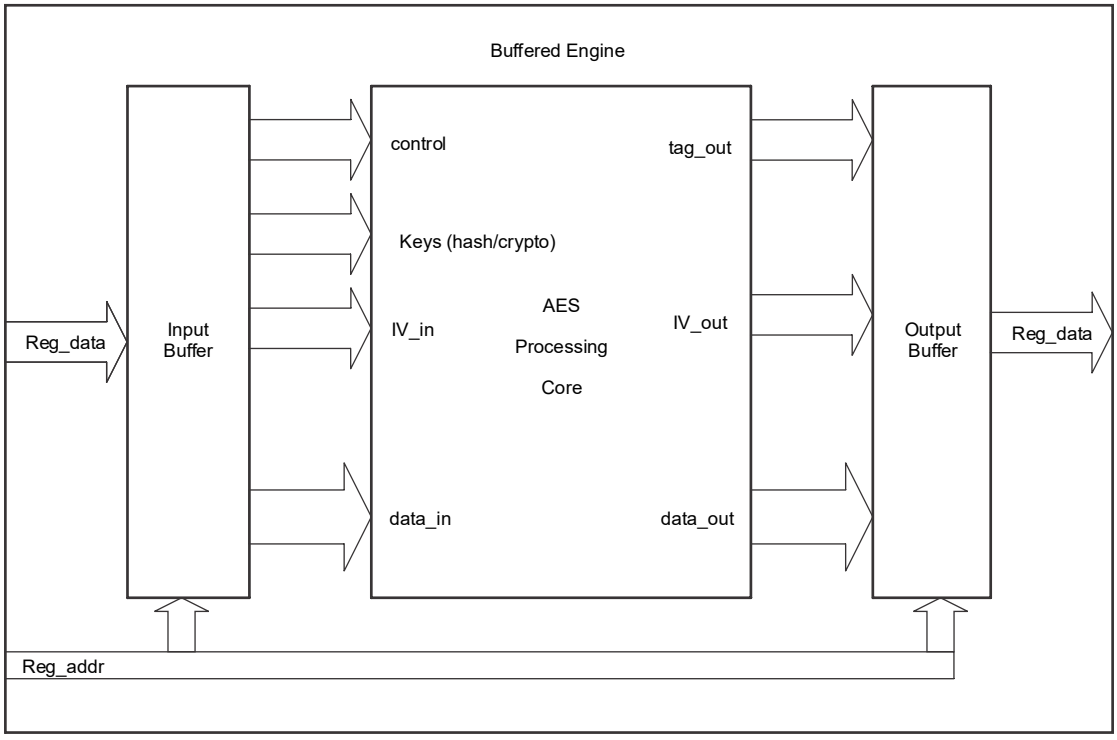


図 3-1. AES のブロック図

3.1.2 AES の使用法

AES モジュールの使用方法は、テキサス・インスツルメンツの MSPM33 ドライバ ライブラリ (DriverLib) にわかりやすく説明されています。AES DriverLib API を活用するサンプルは、MSPM33 SDK の最新リリース内の、評価基板固有の非 RTOS のサンプル サブフォルダにあります。

3.1.2.1 構成

AES モジュールの構成は、SysConfig を使用して実行できます。基本構成では、ユーザーは次のことができます。

- aesadvhp キー長の設定 (128/256 ビットのキー長)
- 操作タイプ (暗号化 / 復号化) の構成
- 暗号モードの構成

詳細については、選択したデバイスのデータシートを参照してください。

3.1.2.2 構成

AES モジュールの標準セットアップでは、次の 4 つの手順を実行します。

1. AES 状態変化監視のセットアップ
   a. CPU は、2 つの方法 (トリガされた割り込みとレジスタ ポーリング) を使用して AES 状態の変化について通知を受けることができます。AES 割り込みを使用して状態の変化を監視する場合、AES モジュールを初期化する前に、これらの割り込みを構成して有効化する必要があります。
2. AES キーのセットアップ
   a. モジュールを初期化する前に、AES キーを AES モジュールで設定する必要があります。
3. AES 暗号モードのセットアップ
   a. SysConfig 構成で選択された暗号モードは、手順 1 および 2 を実行した後に初期化できます。
4. AES モジュールの準備が完了するまで待ちます。

Copyright © 2026 Texas Instruments Incorporated

a. 手順 1 で選択した状態変更監視方法を使用して、AES モジュールの準備が完了したことが CPU に通知されます。

3.1.2.3 動作

構成およびセットアップが完了した後、データを AES モジュールにロードし、暗号化または復号化できます。データがモジュールにロードされたら、CPU は、モジュールがデータの処理を完了するまで待機してから、暗号化または復号化された出力データを読み取る必要があります。この待機は、以前に選択した状態監視方法で実行できます。いずれかの時点で動作タイプを変更した場合は、AES モジュールを再初期化する必要があります。

3.2 キーストア

3.2.1 概要

キーストア コントローラは、Advanced Encryption Engine (AES) キーの安全な管理を提供します。キーストア コントローラの使用モデルは、顧客のセキュア コードの実行中にキーを安全に格納し、その後 AES エンジンがオブザーバーにキーデータを漏らさずに安全な方法でそれらにアクセスすることです。128 ビットと 256 ビットのキーは、キーストアのキー スロットに格納できます。キーストアと AES エンジンとの相互作用は、部分的なキー変更攻撃を阻止するなど、安全な操作を可能にするように設計されています。

3.2.2 キーストアの使用方法

キーストア モジュールの使用方法は、テキサス・インスツルメンツの MSPM33 ドライバ ライブラリ (DriverLib) にわかりやすく説明されています。Keystore DriverLib API を使用したサンプルは、評価基板固有の非 RTOS サンプル サブフォルダにあります。

3.2.2.1 構成

キーストア モジュールの構成は、SysConfig を使用して行うことができます。基本構成では、ユーザーは次のことができます。

- キー ストレージ スロット (0 ～ 3) またはスロット (256 ビット キー用) の構成
- キー長の設定 (128/256 ビットのキー長)
- キー暗号転送先 (AES モジュール) の構成
- キーのロード場所の構成

詳細については、選択したデバイスのデータシートを参照してください。

3.2.2.2 構成

キーストア モジュールの標準セットアップでは、次の 2 つの手順を実行します。

1. 256 ビット キーの数のセットアッ
   a. 使用可能なキース ロットを調整できるように、キーストア モジュールに 256 ビット キーの数を通知する必要があります。すべての 256 ビット キーは、使用可能な最小のスロットに書き込まれます。
2. キー書き込みのセットアップ
   a. キーを転送する前に、キーストア モジュールのキー書き込み構成を実行する必要があります。

3.2.2.3 動作

構成とセットアップが完了後、キーストア モジュール ポインタとキー転送設定を指定することで、キー転送を開始できます。

3.3 SHA2

SHA2 (セキュア ハッシュ アルゴリズム–2) は、NIST 標準に準拠した一連の暗号化ハッシュ関数を提供します。この章では、SHA モジュールの動作について説明します。

Copyright © 2026 Texas Instruments Incorporated

3.3.1 SHA の概要

SHA 暗号化ペリフェラルは、メッセージ認証アプリケーションに使用できる FIPS 準拠のセキュア ハッシュ アルゴリズム (SHA-224、SHA-256) とハッシュベースのメッセージ認証 (HMAC) をサポートしています。

3.3.1.1 SHA 機能

- FIPS PUB 180-2 および FIPS PUB 180-3 の Secure Hash Standard に準拠した実装
- FIP PUB 198-1 に準拠するすべてのアルゴリズムをサポートする HMAC
- クロック サイクル当たり 1 回のハッシュ反復 (7.88 ビット) の性能を備え、高速なスループットを実現する高性能ハッシュ
- SHA-224 および SHA-256 の HMAC と基本ハッシュ演算をサポート
- ハッシュおよび HMAC コンテキスト スイッチング
- MAC キーの XOR およびメッセージ パディングをサポート
- アルゴリズム ブロック サイズよりも短い、等しい、または大きい MAC キーをサポートし
- メッセージ データの自動スケジューリングをサポートし
- 8 ビット単位で最大 2^{64} ビットのメッセージ サイズをサポート
- メッセージ長に基づいて DMA 要求を自動生成し、CPU のオーバーヘッドを低減
- 入力メッセージ解析用の定数アドレスと増分アドレスの両方をサポート

3.3.2 SHA の性能

表 3-1 は 32MHz で動作する MSPM33C321A を使用して、マルチブロック イメージの性能指標を提供します。

表 3-1. SHA ハードウェア アクセラレータの主な性能指標 (マルチブロック)

画像のサイズ	SHA 224 HASH	SHA 224 HMAC	SHA 256 HASH	SHA 256 HMAC
1kb	0.016ms	0.018ms	0.018ms	0.018ms
32kb	0.436ms	0.436ms	0.436ms	0.436ms
64kb	0.867ms	0.866ms	0.866ms	0.868ms
128kb	1.811ms	1.811ms	1.802ms	1.809ms

3.3.3 SHA の使用法

SHA モジュールの使用方法は、テキサス・インスツルメンツの MSPM33 ドライバ ライブラリ (DriverLib) にわかりやすく説明されています。SHA DriverLib API を活用するサンプルは、MSPM33 SDK の最新リリース内の、評価基板固有の非 RTOS のサンプル サブフォルダにあります。

3.3.3.1 構成

SHA モジュールの構成は、SysConfig を使用して実行できます。基本構成では、ユーザーは次のことができます。

- SHA モードの構成 (HASH/HMAC)
- SHA アルゴリズム (224/256) の構成

詳細については、選択したデバイスのデータシートを参照してください。

3.3.3.2 構成

SHA モジュールの標準セットアップでは、次の 4 つの手順を実行します。

1. SHA 状態監視のセットアップ
   a. CPU は、2 つの方法 (トリガされた割り込みとレジスタ ポーリング) を使用して SHA 状態の変化について通知を受けることができます。SHA 割り込みを使用して状態の変化を監視する場合、SHA モジュールを使用する前に、これらの割り込みを設定して有効化する必要があります。
2. SHA データ長のセットアップ
   a. SHA モジュールを使用する前に、SHA データの長さを設定する必要があります。
   b. データ長については、SHA モードに含まれるデータ長をバイト単位で指定する必要があります。
   c. その後、モードとデータ長が利用可能であることが SHA モジュールに通知されます。
3. MAC キーのセットアップ (HMAC モードのみ)

Copyright © 2026 Texas Instruments Incorporated

a. SHA HMAC モードを使用する場合、SHA モジュールを使用する前に、暗号化に使用する MAC キーをモジュールに渡す必要があります。
b. MAC キーは、合計キー サイズを含めて、SHA モジュールにワードで書き込む必要があります。
c. その後、キーが利用可能であることが SHA モジュールに通知される必要があります。
d. キーの処理を開始できることが SHA モジュールに通知される必要があります。
e. 最後に、CPU は SHA モジュールの処理が完了するまで待機する必要があります。

4. ダイレクト メモリ アクセス (DMA) のセットアップ
   a. SHA モジュールはデータ ブロックで動作するため、モジュールがダイジェストするにはデータを手動で供給する必要があります。これを実行するには、CPU 経由または DMA 転送によってデータを供給できます。DMA 転送を使用する場合は、SHA モジュールを使用する前に DMA チャネルを設定する必要があります。
   b. DMA ソース アドレスは、ハッシュするメッセージの先頭を指す必要があります。
   c. DMA 宛先アドレスは、SHAW_DATA_FIXED レジスタを指す必要があります。
   d. DMA メッセージの長さは、メッセージ長をワードで設定する必要があります。
   e. DMA のソースおよび宛先の幅は、ワードのサイズである必要があります。
   f. DMA ソースはシングル インクリメント モードに構成する必要があり、宛先は変更できません。
   g. DMA トリガは SHAW_TRIGGER に設定する必要があります。
   h. 最後に、DMA チャネルを有効化します。

3.3.3.3 動作

構成とセットアップが完了した後、DMA または CPU によりデータを SHA モジュールにロードできるようになります。データがモジュールにロードされたら、CPU は、モジュールがデータの処理を完了するまで待機してから、出力のハッシュされたデータを読み取る必要があります。この待機は、以前に選択した状態監視方法で実行できます。ハッシュされたデータが SHA モジュールから読み出された後、そのデータを解放して、次のハッシュ要求に使用する必要があります。

3.4 PKA

3.4.1 PKA の概要

PKA は、公開鍵アクセラレーションのための統合モジュールであり、多くのリソースを消費する公開暗号演算の計算負荷を軽減します。

3.4.1.1 PKA 機能

- PKA エンジンでは、次のような基本的な動作が行われます。
  - 大きいベクトルの加算、減算、および加算と減算の組み合わせ
  - 大きいベクトルの左または右へのシフト
  - 大きいベクトルの乗算と除算 (商の有無にかかわらず)
  - 大きいベクトルの比較とコピー
- PKA エンジンでは、次のような複雑な動作が行われます。
  - 大きいベクトルの符号なし値モジュラべき乗
  - 事前に計算された Q 逆ベクトルを用いた CRT 法による、大きいベクトルの符号なし値モジュラべき乗
  - モジュラ逆数
  - 2 種類の曲線上の ECC 演算：モンゴメリー曲線は Curve25519 や Curve448 のようなもので、$y^2 = x^3 + ax + b \pmod p$ という形の任意の曲線です。
  - 入力 / 出力としてアフィンまたは射影点を持つ楕円曲線上の ECC ポイント加算 / 2 倍算
  - 楕円曲線上の ECC 点の乗算
- 不正な状態およびタイミング攻撃の検出

3.4.2 PKA の使用法

PKA モジュールの使用方法は、テキサス・インスツルメンツの MSPM33 ドライバ ライブラリ (DriverLib) にわかりやすく説明されています。PKA DriverLib API を使用したサンプルは、評価基板固有の非 RTOS のサンプル サブフォルダにあります。

Copyright © 2026 Texas Instruments Incorporated

3.4.2.1 構成

PKA モジュールをプロジェクトに追加するだけで、SysConfig を使用して PKA モジュールの構成を行うことができます。

詳細については、選択したデバイスのデータシートを参照してください。

3.4.2.2 構成

初期化ステップが自動的に生成されるため、PKA モジュールのセットアップ要件は SysConfig によって処理されます。生成された *SYSCFG_DL_init()* 関数を呼び出すだけで十分です。

3.4.2.3 動作

構成およびセットアップが完了すると、PKA モジュールを使用して暗号操作を実行できるようになります。PKA 操作を開始してから、PKA ステータスをポーリングして操作が終了するのを待つことをお勧めします。PKA ステータスがクリアされると、PKA 操作の結果を取得できます。

3.5 PQC

ポスト量子暗号 (PQC) は、量子コンピュータによる脅威に対抗するために設計された暗号化標準のセットであり、米国国立標準技術研究所 (NIST) によって定義されています。これらの標準の 1 つに、Module-Lattice-Based Digital Signature Algorithm (ML-DSA) があります。

3.5.1 ML-DSA

3.5.1.1 ML-DSA の概要

ML-DSA は、量子コンピュータによる脅威から保護するために使用される、NIST が定義した PQC アルゴリズムの 1 つです。この標準は、顧客セキュア コード (CSC) と組み合わせて使用することで、組込みアプリケーションの真正性を確保できます。

3.5.1.2 ML-DSA の使用方法

ML-DSA 標準の使用方法は、テキサス・インスツルメンツの MSPM33 Crypto Library (暗号ライブラリ) にわかりやすく説明されています。ML-DSA 規格と MCUBoot を組み合わせて活用する例は、MSPM33 SDK の最新リリース内の、評価基板固有の非 RTOS サンプル サブフォルダにあります。

3.5.1.2.1 構成

ML-DSA を利用するための MCUBoot の構成は、SysConfig で行えます。基本構成では、次の操作が可能です。

- ML-DSA を使用して認証を有効にする
- ECDSA および ML-DSA を使用して認証を有効にする

詳細については、選択したデバイスのデータシートを参照してください。

3.5.1.2.2 構成

MCU ブートでの ML-DSA 使用のセットアップは、MLDSA キーとキー長を MCUBoot が使用するグローバル構造体 bootutil_keys に格納することで完了となりきます。以下にこの例を示します。

```
const struct bootutil_key bootutil_keys[] = {
    {
        .key = mldsa_pub_key,
        .len = &mldsa_pub_key_len,
    },
};
const int bootutil_key_cnt = 1;
```

3.5.1.2.3 動作

構成とセットアップが完了すると、MCUBoot は、boot_go が開始時にアプリケーションイ メージを検証するために、ML-DSA および事前に設定した公開 ML-DSA キーを使用します。

4 改訂履歴

資料番号末尾の英字は改訂を表しています。その改訂履歴は英語版に準じています。

Copyright © 2026 Texas Instruments Incorporated

日付	改訂	注
2025 年 12 月	*	初版リリース

Copyright © 2026 Texas Instruments Incorporated

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

Copyright © 2026, Texas Instruments Incorporated

最終更新日：2025 年 10 月