Application Note

MSPM0 MCU のサイバーセキュリティ・イネーブラ

概要

MSPM0 マイコンは、コード、データ、キーといった資産を保護するためのセキュリティ対策を実装できるよう、さまざまなセキュリティ有効化技術を提供します。MSPM0 デバイス内には、セキュア ブートとセキュア ストレージを実現するためのハードウェアとソフトウェアの組み合わせソリューションが提供されています。このドキュメントでは、これらのデバイスに搭載されているイネーブラ、それらの機能と制限、動作方法、基本的な使用事例に合わせた構成方法について説明します。

目次

商標

English Document: SLAAE29

1 はじめに

産業用、車載、パーソナル・エレクトロニクスのアプリケーションがネットワーク接続され、攻撃者が利用できるツールが増加し続けるのにつれて、組込みアプリケーションでのデバイス・セキュリティの重要性も高まっていきます。テキサス・インスツルメンツの MSPM0 マイクロコントローラには、さまざまなハードウェアおよびソフトウェアのセキュリティ保護を実現するテクノロジーが搭載されており、セキュリティを考慮してアプリケーションを開発する際に活用できます。

1.1 主な概念

表 1-1. 主な概念

用語	意味
NONMAIN	デバイスのブート関連パラメータを構成する専用フラッシュ メモリ領域。NONMAIN の操作ガイドについては、MSPM0 NONMAIN フラッシュ操作ガイドを参照してください。
セキュア ブート	実行に先立つ前提条件として、更新可能なファームウェアおよびソフトウェア コンポーネントの完全性と真正性を検証および検証するプロセス。
INITDONE	INITDONE は、一部の MSPM0 デバイスに存在するレジスタで、特権状態と非特権状態を分離するために使用されます。INITDONE は、CSC によって特権状態の終了時にトリガされ、CSC で設定されたすべての非静的なセキュリティ ポリシは INITDONE の時点で有効になります。
カスタマ セキュア コード (CSC)	INITDONE 機構を備えたデバイス向けに、MSPM0 SDK で提供されているセキュア ブート ソリューション。これは信頼の起点の一部として機能し、製造後は不変のまま保持され、アプリケーション イメージの完全性および真正性の検証に加えて、その他のセキュリティ ポリシーの設定を実現します。CSC は、MSPM0 のハードウェア機能を指す場合もあり、この場合、MSPM0 デバイスが INITDONE メカニズムをサポートしていることを意味します。
ブート イメージ マネージャ (BIM)	INITDONE 機構を持たないデバイス向けに、MSPM0 SDK で提供されているセキュア ブート ソリューション。
信頼の基点 (RoT)	特に、デバイス上で最も信頼されているセキュリティコンポーネントである不変の信頼の起点を指します。製造後に変更することはできないため、本質的に信頼されています。より深いレベルでは、本物で変更されていないことを検証できるソフトウェアはありません。CSC ソリューションには、ROM ブートコードと静的書き込み保護付き CSC が含まれます。
キーストア	AES キー向けセキュア ストレージ。キーを Keystore に設定できるのは CSC のみであり、メインアプリケーションは、保存されているキーのうち 1 つを使用するように暗号エンジン (AES) を設定することはできますが、保存されているキー自体にアクセスすることは一切できません。
ファイアウォール	フラッシュメモリの特定領域に対する動的な保護メカニズムで、書き込み保護、読み取り、実行保護、IP 保護を含みます。
バンク スワップ	MSPM0 のデュアル バンク対応デバイスにおいて、フラッシュ バンクのアドレス マッピングを設定するためのメカニズムです。これは CSC で設定され、INITDONE 後に有効になります。
静的書き込み保護	NONMAIN 構成でイネーブルされる静的書き込み保護メカニズムです。NONMAIN 構成が再度書き込みをイネーブルにするように変更されない限り、ROM ブート コードの完了後に保護領域を変更できません。
SHA2-256	ハッシュ アルゴリズムはメッセージ全体を受け取り、固定長 (256 ビット) のダイジェストにまとめます。これは、メッセージの整合性を検証するために使用されます。MSPM0 デバイスのソフトウェアによってのみサポートされています。
ECDSA P256	メッセージの真正性を確認するための非対称アルゴリズムです。MSPM0 デバイスのソフトウェアによってのみサポートされています。
AES	高度暗号化標準では、一部の MSPM0 デバイスに AES 用のハードウェア アクセラレータが搭載されています。
TRNG	真性乱数生成器。一部の MSPM0 デバイスは、TRNG 向けのハードウェア アクセラレータを搭載しています。

1.2 サイバー・セキュリティの目標

一般に、組込みアプリケーションにおけるサイバー・セキュリティの主な目的は、以下の方法で重要な資産を保護することです。

- 機密性 (機密データへのアクセスを制限)
- 完全性 (データを変更から保護)
- 真正性 (すべての当事者が本人であることを保証)
- 可用性 (データや機能を必要なときに使用可能)
- 否認防止 (データの出所や ID を追加の当事者に提示可能)

これらは、多くの場合、以下の状態にある資産に適用されます。

- 休止時 (マイクロコントローラ上のコード、データ、またはキーはアクティブに使用されていない)
- 使用中 (マイクロコントローラ上のコード、データ、またはキーがアプリケーションでアクティブに使用されている)
- 転送中 (マイクロコントローラ上のコード、データ、またはキーが MCU と他のエンティティの間を移動中)

1.3 プラットフォームのセキュリティ イネーブラ

表 1-2 に、MSPM0 デバイスに搭載されているセキュリティ イネーブラを示します。デバッグ セキュリティ機能と MAIN フラッシュ メモリの整合性検証機能については、デバイス シリーズ テクニカル リファレンス マニュアルの「NONMAIN レイアウト タイプ」および「NONMAIN レジスタ」セクションを参照してください。セキュア ブート、セキュア ストレージ、暗号化アクセラレータの各機能は、各デバイス固有のデータシートに掲載されています。

表 1-2. MSPM0 MCU プラットフォームのセキュリティ イネーブラ

セキュリティ イネーブラ	デバイスの機能	M0C1103/4	M0C1105/6	M0L1x0x/M0L134x	M0G110x	M0G3x0x/M0G150x	M0H321x	M0L111x	M0Lx22x	M0Gx51x
デバッグのセキュリティ	パスワード認証を使用したデバッグ アクセス		ハッシュ	あり	あり	あり	ハッシュ	ハッシュ	ハッシュ	ハッシュ
	パスワード認証を使用したブートストラップ ローダ アクセス	ROM BSL なし	ROM BSL なし	あり	あり	あり	ROM BSL なし	ハッシュ	ハッシュ	ハッシュ
	パスワード認証を使用した MAIN フラッシュ メモリの一括消去		ハッシュ	あり	あり	あり	ハッシュ	ハッシュ	ハッシュ	ハッシュ
	パスワード認証を使用した完全な工場出荷時リセット		ハッシュ	あり	あり	あり	ハッシュ	ハッシュ	ハッシュ	ハッシュ
	テキサス・インスツルメンツ故障解析 (FA) のイネーブル / ディセーブル		あり	あり	あり	あり	あり	あり	あり	あり
	シリアル ワイヤ デバッグ (SWD) インターフェイスの完全なハードウェア ディセーブル	あり	あり	あり	あり	あり	あり	あり	あり	あり
	デバイス構成データを永続的にロック可能	あり	あり	あり	あり	あり	あり	あり	あり	あり
	エラー耐性のあるデバイス構成データ		あり	あり	あり	あり	あり	あり	あり	あり
	パスワードはハッシュ形式でのみ保存されます (SHA2-256)		あり				あり	あり	あり	あり

表 1-2. MSPM0 MCU プラットフォームのセキュリティ イネーブラ (続き)

セキュリティ イネーブラ	デバイスの機能	M0C1103/4	M0C1105/6	M0L1x0x/M0L134x	M0G110x	M0G3x0x/M0G150x	M0H321x	M0L111x	M0Lx22x	M0Gx51x
セキュア ブート	CSC が存在します		あり				あり	あり	あり	あり
	メイン フラッシュ メモリを永続的にロック可能	あり	あり	あり	あり	あり	あり	あり	あり	あり
	CRC-32 検証を使用した MAIN フラッシュ領域		あり	あり	あり	あり	あり	あり	あり	あり
	SHA2-256 検証を使用した MAIN フラッシュ メモリ領域		あり				あり	あり	あり	あり
	ブート時に MAIN フラッシュ アプリケーションへのエントリ ポイントを 1 つに制限	あり	あり	あり	あり	あり	あり	あり	あり	あり
	非対称ファームウェア イメージ認証ルーチン(ECDSA P-256、SHA2-256、いずれもソフトウェア実装)		あり	あり	あり	あり	あり	あり	あり	あり
	対称型ファームウェア イメージ認証ルーチン (ハードウェア ベースの AES-CMAC)							あり	あり	あり
	ECDSA 公開鍵の失効およびロールバック保護のためのロック可能なフラッシュ		あり				あり	あり	あり	あり
	SRAM の書き込み・実行相互排他 (W^X) 境界		あり	あり	あり	あり	あり	あり	あり	あり
セキュア ストレージ	フラッシュ書き込み保護ファイアウォール		あり				あり	あり	あり	あり
	フラッシュの読み取り / 実行 (RX) 保護ファイアウォール		あり				あり	あり	あり	あり
	フラッシュ IP 保護領域 (実行専用、読み取りアクセスなし)		あり				あり	あり	あり	あり
	フラッシュ バンクの書き込み実行相互排他 (W^X)							あり	あり	あり
	データ バンク書き込み読み取り保護									あり
	キー ストア (最大 4 つの 128 ビット キーまたは 2 つの 256 ビット キーと 1 つのセッション キー)							あり	あり	あり
	ハードウェア単調カウンタ							あり		あり

表 1-2. MSPM0 MCU プラットフォームのセキュリティ イネーブラ (続き)

セキュリティ イネーブラ	デバイスの機能	M0C1103/4	M0C1105/6	M0L1x0x/M0L134x	M0G110x	M0G3x0x/M0G150x	M0H321x	M0L111x	M0Lx22x	M0Gx51x
暗号化アクセラレーション機能	セルフテスト機能を備えた真性乱数生成器 (TRNG)					あり		あり	あり	あり
	基本的な AES アクセラレータ (GCM/CMAC/GHASH をサポートなし)					あり		あり	あり	あり
	高度な AES アクセラレータ (GCM/CMAC/GHASH をサポート)							あり	あり	あり
デバイス ID	固有のデバイス識別子 (96 ビット)	あり	あり	あり	あり	あり	あり	あり	あり	あり
認証	ARM PSA レベル						L1 予定	L1 予定	L1	L1 予定
	EVITA 機能					EVITA-Light		EVITA-Light	EVITA-Light	EVITA-Light
	ISO 21434 プロセス準拠							予定		予定
攻撃抵抗分析	3P ファームウェアの脆弱性分析						あり	あり	あり	あり
	ブート構成ルーチンによるフォルト注入攻撃への対策						あり	あり	あり	あり

2 デバイス セキュリティ モデル

MSPM0 ファミリには、次の 2 つの大規模なセキュリティ機能段階があります:

1. フラッシュ アプリケーションに入る前に、ユーザー定義の構成によって制御される、TI が作成した静的なブートコード
2. MAIN フラッシュに格納され、静的に書き込み保護された、ユーザー作成のカスタマ セキュア コード (CSC)。BCR の後に実行され、アプリケーションへジャンプする前に、追加のポリシーを適用し、またはアプリケーションの真正性および完全性を検証します

すべてのデバイスが追加の CSC ステップをサポートしているわけではありません。ファイアウォールなどの追加のセキュリティ ポリシーを適用できるのは CSC のみですが、CSC 非対応デバイスであってもアプリケーションの検証を行うことは可能です。アプリケーションの検証についてはセキュア ブートのセクションで詳しく説明するため、本章では追加のポリシーの適用に議論を限定し、この段階を CSC と呼ぶことにします。

このセクションでは、デバイスのブート プロセスの概要と、幅広いユース ケースを可能にするために設定できる、両カテゴリにおけるユーザー指定ポリシーについて説明します。

2.1 デバイス ID

すべての MSPM0 デバイスには、96 ビットのユニット固有識別コード (デバイス ID) が含まれており、アプリケーション・ソフトウェアで読み取ることができます。デバイス ID の詳細については、テクニカル・リファレンス・マニュアルとデバイスのデータシートを参照してください。

デバイス ID は、出荷される各ユニットで固有であり、特定のユニットと他のユニットを識別または区別するために使用できます。デバイス ID は固有ですが、一部のビットは部品番号や製品リビジョンなどのデバイス特性に対応しているため、暗号論的にランダムではありません。

2.2 ブート時の初期条件

コールド・パワーアップ (POR) 中、デバイスはセキュア状態にリセットされます。デジタル IO ピンは高インピーダンスとなり、すべてのペリフェラル機能が切断され、NRST ピンは NRST モード、シリアル・ワイヤ・デバッグ (SWD) インターフェイス・ピンは SWD モードになります。ブラウンアウト・リセットの解放後、シリアル・ワイヤ・デバッグ・ポート (SW-DP) が最初にイネーブルになり、デバッグ・プローブからデバッグ・サブシステムへの初期接続が確立されます。

ブート・プロセスのこの時点で、デバッグ・プローブからアクセス可能なデバッグ・アクセス・ポート (DAP) は、構成アクセス・ポイント (CFG-AP) とセキュア・アクセス・ポイント (SEC-AP) のみです。CFG-AP は、接続されたデバッグ・プローブで汎用デバイス情報 (デバイスの汎用部品番号など) を読み取るために使用できます。SEC-AP は、ブート構成ルーチンにコマンド・メッセージを渡すために使用できます。デバイスへのアプリケーション・デバッグ・アクセス (AHB-AP、ET-AP、および PWR-AP DAP 経由) は、ハードウェア・ファイアウォールによってブロックされたままです。そのため、デバイスの電源投入時には、デバイス・ハードウェアでプロセッサ、EnergyTrace 状態、または電源構成へのデバッグ・アクセスは許可されません。

ブラウンアウト・リセット (BOR) 後は、ブート・リセット (BOOTRST) が生成され、ブート構成ルーチンの実行が開始されます。

2.3 ブート構成ルーチン (BCR)

MSPM0 デバイスには、読み取り専用メモリ (ROM) に変更不可能な信頼ルート ブート構成ルーチンが含まれています。ブート構成ルーチン (BCR) は、デバイスの BOOTRST に続いて Cortex-M0+ プロセッサで実行される最初のコードです。BCR は、BSL エントリを認証するために必要なブートストラップ ローダ (BSL) のソフトウェア起動時にも実行されます。BCR の主な機能は次のとおりです。

1. 適切なデバイス動作に必要な TI の工場出荷時データを FACTORY フラッシュ メモリ領域からロジックにロードし、CRC-32 を使用して工場出荷時データ (デバイスのトリム データを含む) の整合性を検証
2. ユーザー指定のデバイス構成 (セキュリティ ポリシーを含む) を NONMAIN フラッシュ メモリ領域からロジックにロードし、CRC-32 を使用してユーザー構成データの整合性を検証
3. シリアル ワイヤ デバッグ (SWD) インターフェイス経由で送信されたブート コマンドを確認し、(該当する場合は) それらを認証して、(認証された場合は) 処理

4. ブートストラップ ローダ (BSL) がイネーブルになっている場合は BSL の呼び出し条件を確認し、有効な呼び出しが発生した場合に BSL を開始
5. ユーザー アプリケーションを起動する前に、CRC-32 または SHA-256 を使用して、ユーザー アプリケーション コードを含む MAIN フラッシュ メモリ領域のユーザー定義部分の整合性を確認
6. BCR の終了時、CSC が INITDONE を発行した後 (詳細はセクション FIX 参照)、またはまったく行わない、のいずれかとして、AHP-AP、ET-AP、および PWR-AP の DAP を解放するかどうかを決定します。
7. すべてのブート エラーを CFG-AP に記録
8. MAIN フラッシュ内の 0x0000.0000 からスタック ポインタを、0x0000.0004 からリセット ベクタをフェッチすることで、MAIN フラッシュ内の単一のエントリ ポイントから実行を開始するようにハードウェアをトリガします

2.4 ブートストラップ・ローダ (BSL)

MSPM0 デバイスには、読み取り専用メモリ (ROM) に変更不可能なブートストラップ・ローダ (BSL) が含まれていることもあります。BSL では、シリアル・ワイヤ・デバッグ (SWD) インターフェイスではなく、標準のシリアル・インターフェイス (UART または I2C) を使用して、デバイス・メモリの内容をプログラムおよび検証できます。

BSL は BCR でのみ起動できます。BCR は、有効な BSL 呼び出し条件 (ソフトウェア起動、IO ピン起動、ブランク・デバイス起動) をチェックし、BSL を起動する前に BSL がイネーブルになっていることを検証します。BSL が終了すると、BCR が再実行され、現在のデバイス・セキュリティ・ポリシーがロードされて、ユーザー・アプリケーションが起動します。

BSL は 256 ビットのユーザー指定パスワードで保護されており、BSL セッションを開始するときに、パスワードを UART または I2C インターフェイス経由で BSL に渡す必要があります。BSL を使用しない場合は、ディセーブルにできます (BSL イネーブル / ディセーブル・ポリシーを参照)。

2.5 ブート フロー

MSPM0 デバイスにおける高レベルのブート フローを、以下の図に示します。

BOOTRST では、TI ブートコードの実行が開始されます。ブートが成功すると、ブートコードが BOOTDONE を発行します。この時点で、SYSCTL はデバイスに SYSRST を発行し、フラッシュ メモリからの実行をトリガします。ブート構成レコードに応じて、これにより、メインアプリケーションの起動 (この構成で CSC が存在しない場合) または CSC の起動 (CSC が設定されている場合) のいずれかに進みます。

CSC は、実行バンクの決定、メモリ領域の保護設定、キーストアへのセキュア キーの初期化などを担当します。顧客セキュア コードが SYSCTL.SECCFG.INITDONE MMR に書き込んで INITDONE を発行すると、SYSCTL は 2 回目の SYSRST を発行します。デバイスは再び、フラッシュにマップされた 0x0 から実行を開始し、CSC が 2 回目に実行されます。今回は、CSC が INITDONE がすでに以前に発行されていることを検出し (これは

SYSCTL.SECCFG.SECSTATUS.INITDONE ビットを読み取ることで判定されます)、メイン アプリケーションを直接呼び出します。

BOOTRST からメイン アプリケーションまでの大まかなブート フロー:

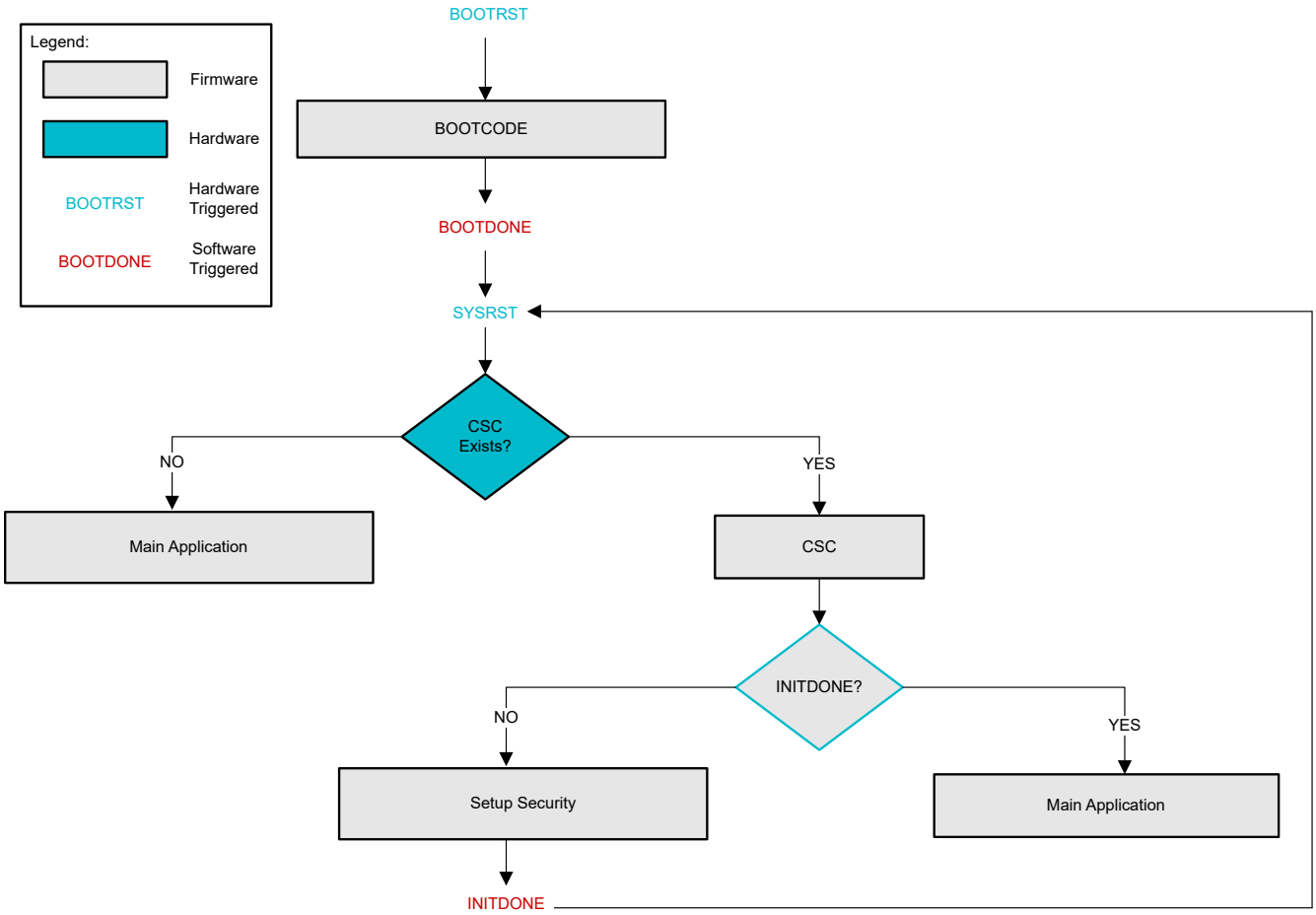

図 2-1. 大まかなブート フロー

注

セキュリティが有効化されていないデバイスとのブート フロー互換性を保つため、ブート構成のデフォルト設定は『CSC が存在しない』状態に設定されています。

セキュアな実行フローは、CSC_EXISTS = YES のパスです。この場合、BOOTRST の後、メイン アプリケーションが起動する前に 2 つの SYSRST が発行されることがあります。最初の SYSRST の後、顧客のスタートアップ コードが実行されます。セキュリティを設定し、INITDONE を発行します。この時点で、セキュリティ設定はロックされ、強制されます。この時点で 2 番目の SYSRST が発行され、スタートアップ コードの実行が再開されます。2 番目の SYSRST では、INITDONE が YES であるため、メイン アプリケーションが起動します。

BCR と BSL のどちらにも、ロック可能な NONMAIN フラッシュ メモリ領域にユーザー指定の構成データ構造が格納されています。これらのデータ構造で指定されているセキュリティ ポリシーについては、セクション 2.6 を参照してください。

2.6 ユーザー指定のセキュリティ ポリシー

MSPM0 デバイスには、ユーザー指定のセキュリティおよびデバイス構成ポリシーを保存するための専用のフラッシュ メモリ領域があります。この領域を NONMAIN フラッシュ領域と呼びます。ブート構成ルーチン (BCR) およびブートストラップ ローダ (BSL) は、デバイスを動作用に構成するため、NONMAIN フラッシュ領域に保存されているユーザー指定のデータを参照します。

製造時に、デバイスの NONMAIN フラッシュ メモリ領域に必要なポリシーをプロビジョニングする必要があります。このセクションでは、NONMAIN 構成メモリを使用して構成可能なセキュリティ ポリシーについて説明します。

NONMAIN フラッシュ領域は、次の 2 つの異なるデータ構造に分割されています。

- BCR 構成 (セクション 2.3 を参照):ブート構成のセキュリティ ポリシーを設定
- BSL 構成 (セクション 2.6.1.4 を参照):ブートローダのセキュリティ ポリシーを設定

どちらのデータ構造も、独自の 32 ビット CRC ダイジェストによってバックアップされ、構成データのエラー耐性機能の一部として使用されます。

注

BCR および BSL 構成構造には、このドキュメントに記載されているパラメータ以外のパラメータも含まれています。このドキュメントでは、セキュリティに関連するパラメータに焦点を当てています。NONMAIN フラッシュ メモリ領域における BCR および BSL 構成構造の詳細な説明は、該当するテクニカル リファレンス マニュアルのアーキテクチャの章にあるブート構成のセクションを参照してください。

2.6.1 ブート構成ルーチン (BCR) ポリシー

以下のセクションでは、フラッシュの NONMAIN 領域に静的に定義され、ブート構成ルーチン (BCR) によって適用されるポリシーについて説明します。特に記述のない限り、すべての MSPM0 デバイスは、これらのポリシーを強制できます。

2.6.1.1 シリアル ワイヤ デバッグ関連のポリシー

シリアル ワイヤ デバッグ関連のポリシーは、デバイスの物理的デバッグ インターフェイス (SWD) 経由で利用できる機能を構成します。デフォルトでは、MSPM0 デバイスは テキサス・インスツルメンツから無制限の状態で出荷されます。この状態により、製造時のプログラミング、評価、開発が容易になります。ただし、この無制限の状態では、大きな攻撃対象領域が存在したままになるため、量産では推奨されません。構成プロセスをシンプルに保ちながら各種のニーズに対応するため、MSPM0 デバイスは、制限なし (レベル 0)、カスタム制限 (レベル 1)、完全制限 (レベル 2) という 3 つの汎用セキュリティ レベルをサポートしています。表 2-1 に、最も制限の少ないものから最も制限の厳しいものの順に、3 つの汎用セキュリティ レベルを示します。

SWD インターフェイスには、保護することを考慮する必要がある 4 つの主な用途があります。

- アプリケーション デバッグ アクセス。以下のものが含まれます。
  - AHB-AP 経由でプロセッサ、メモリ マップ、ペリフェラルにフル アクセス
  - ET-AP 経由でデバイス EnergyTrace+ 状態情報にアクセス
  - PWR-AP 経由でデバッグを行うためのデバイス電源状態制御へのアクセス
- 以下を含む一括消去アクセス:
  - SWD 経由でコマンドを送信して MAIN メモリ領域を消去し、NONMAIN デバイス構成メモリはそのまま保持する機能
- 以下を含む工場出荷時リセットへのアクセス:
  - SWD 経由でコマンドを送信して MAIN メモリ領域を消去し、NONMAIN デバイス構成メモリを テキサス・インスツルメンツの工場出荷時デフォルト (レベル 0) にリセットする機能
- 以下を含む テキサス・インスツルメンツ故障解析アクセス:
  - テキサス・インスツルメンツが SWD 経由で故障解析の返却フローを開始する機能 (テキサス・インスツルメンツに FA アクセスが付与される前に、テキサス・インスツルメンツ FA フローにより強制的に工場出荷時リセットが実行されます。これにより、故障解析フローが開始されたときに、デバイスのフラッシュ メモリに保存されていた独自の顧客情報を テキサス・インスツルメンツが読み取ることができないことが確実になります)。

表 2-1. 汎用セキュリティ レベル

レベル	シナリオ	SW-DP ポリシー	アプリケーション デバッグ ポリシー	一括消去ポリシー	工場出荷時のリセット ポリシー	テキサス・インスツルメンツ FA ポリシー
0	制限なし	EN	EN	EN	EN	EN
1	カスタム制限	EN	イネーブル、ディスエーブル	ディスエーブル	イネーブル、ディスエーブル	イネーブル、ディセーブル

表 2-1. 汎用セキュリティ レベル (続き)

レベル	シナリオ	SW-DP ポリシー	アプリケーション デバッグ ポリシー	一括消去ポリシー	工場出荷時のリセット ポリシー	テキサス・インスツルメンツ FA ポリシー
2	完全制限	DIS	無関係 (SW-DP が無効な場合はアクセス不可)(1)			

(1) SW-DP ポリシーが **SW-DP ディスエーブル**の場合、SWD インターフェイスの観点からは、一括消去と工場出荷時リセットのポリシーは関係ありません。ただし、ブートストラップ ローダ (BSL) がイネーブルになっている場合、一括消去および工場出荷時リセットのポリシーは、BSL を使用して利用可能な機能に影響を及ぼします。BSL のセキュリティ保護の詳細については、BSL セキュリティのセクションを参照してください。

2.6.1.1.1 SWD セキュリティ・レベル 0

SWD セキュリティ・レベル 0 は、SWD セキュリティ状態のうち最も制限の少ないレベルです。これはテキサス・インスツルメンツの新しいデバイスのデフォルト状態であり、工場出荷時リセットが成功した後のデバイスの状態でもあります。この状態で故障解析を行うため、アプリケーションのデバッグ・アクセス、一括消去、工場出荷時リセットに制限はありません。

この状態を使用する状況

レベル 0 は、デバイス・メモリのプログラミングや、プロセッサとペリフェラルのデバッグが可能なため、プロトタイプ作成や開発に最適です。

この状態を使用するべきでない状況

量産ではレベル 0 を使用しないでください。攻撃者が自由にデバイス・メモリの内容を読み取り、デバイスの実行を操作し、(フラッシュ・メモリの書き込み保護方式に応じて) フラッシュ・メモリの内容を変更できてしまいます。

2.6.1.1.2 SWD セキュリティ・レベル 1

SWD セキュリティ・レベル 1 では、セキュリティ構成をカスタマイズできます。物理的デバッグ・ポート (SW-DP) はイネーブルのままにし、各機能 (アプリケーションのデバッグ、一括消去コマンド、工場出荷時リセット・コマンド、テキサス・インスツルメンツ故障分析) は個別にイネーブル、ディセーブル、または (場合によって) パスワード認証を使用してイネーブルにでき、特定の使用事例に合わせてデバイスの動作をカスタマイズすることが可能です。

この状態を使用する状況

レベル 1 は、制限されたプロトタイプ製作 / 開発シナリオや、特定の SWD 機能 (工場出荷時リセット、テキサス・インスツルメンツ故障解析など) を保持しながら、アプリケーションのデバッグなどの機能をディセーブルにする量産シナリオに最適です。表 2-2 に、レベル 1 のカスタマイズ構成の一般的な例を示します。

表 2-2. レベル 1 構成の例

レベル 1 シナリオ	構成			
	アプリケーションのデバッグ	一括消去	工場出荷時リセット	テキサス・インスツルメンツ FA
このシナリオでは、ユーザー指定のパスワードを使用してデバッグ・アクセスを制限するが、工場出荷時リセットとテキサス・インスツルメンツ障害解析は使用可能。この構成ではフィールド・デバッグ (パスワードを使用) が可能で、工場出荷時リセットによりデバイスをデフォルトのレベル 0 状態に戻すことも可能。	パスワードでイネーブル	ディセーブル	イネーブル	イネーブル
このシナリオではデバッグは不可。工場出荷時リセットは可能だが、ユーザー指定のパスワードが必要。これにより、MAIN メモリの内容をクリアし、パスワードがわかっている場合はデバイスをレベル 0 状態に戻すことにより、現場でデバイスを開けることが可能。工場出荷時リセット用のパスワードが漏洩した場合でも、攻撃者は MAIN フラッシュ・メモリ内の独自の情報を読み取ることは不可。	ディセーブル	ディセーブル	パスワードでイネーブル	イネーブル
このシナリオでは、デバッグとテキサス・インスツルメンツ故障解析は不可。これにより、ユーザーが FA 用にデバイスをテキサス・インスツルメンツに返却する場合に、ユーザー指定のパスワードを使用して工場出荷時リセットを実行しない限り、テキサス・インスツルメンツがデバイスに対して工場出荷時リセットおよび FA 作業を実行することは不可。	ディセーブル	ディセーブル	パスワードでイネーブル	ディセーブル

注

レベル 1 は、ほとんどの標準的な製造使用事例で推奨される構成です。セキュア・ブートを必要としないアプリケーションでは、製造にレベル 1 を使用し、工場出荷時リセットを (パスワードを指定して) イネーブルにして、テキサス・インスツルメンツ故障解析はイネーブルのままにすることをお勧めします。このような構成では、ユーザー (パスワードを使用) またはテキサス・インスツルメンツ (障害解析の返却フロー) のいずれかによるプロビジョニング後に、デバイスをより制限の少ない状態に回復できます。最大限のセキュア・ブート保護を必要とする使用事例では、製造用により制限の厳しいレベル 1 またはレベル 2 を使用できます。ただし、デバイスをプロビジョニングした後は、より制限の低い状態に回復できない可能性があるというトレードオフがあります。

この状態を使用するべきでない状況

プロトタイプ作成時にデバイスへの完全なアクセスが必要な場合は、レベル 1 を使用せず、レベル 0 を使用してください。

また、最大限の制限が必要で、SWD 機能をイネーブルにしない量産シナリオでも、レベル 1 を使用しないでください。このような場合はレベル 2 を使用して、SWD 物理インターフェイス全体を直接ディセーブルし、誤設定の可能性を最小限に抑える必要があります。

注

アプリケーション・デバッグと工場出荷時リセットをでディセーブルしてデバイスを構成した場合、ユーザーがデバイスへのデバッグ・アクセスを回復する唯一の方法は、ユーザー・アプリケーション・コードで NONMAIN 構成をより制限の少ない状態に変更するメカニズムを提供することです。NONMAIN が静的な書き込み保護によってロックされている場合は、状態を変更できず、ユーザーがデバッグ・アクセスを回復することはできません。

2.6.1.1.3 SWD セキュリティ レベル 2

SWD セキュリティ レベル 2 では、デバイスは最大制限状態に設定されます。物理デバッグ ポート (SW-DP) は完全にディセーブルになり、SWD でアクセス可能なすべての機能 (アプリケーションのデバッグ、一括消去、工場出荷時リセット、テキサス・インスツルメンツ故障解析) には、個別の構成に関係なく、SWD からアクセスできません。

レベル 2 を選択した場合 (SW-DP がディセーブル)、アプリケーションのデバッグ構成および テキサス・インスツルメンツ故障解析構成フィールドはデバイス構成には影響を与えません。

BSL がディセーブルの場合、一括消去および工場出荷時リセットの構成フィールドも無関係なフィールドになります。BSL がイネーブルの場合は、BSL インターフェイスから送信される一括消去または出荷時リセットのコマンドを認証するため、BSL は引き続き一括消去および工場出荷時リセットの構成フィールドを使用します。

この状態を使用する状況

レベル 2 は、SWD 機能へのいかなる追加アクセスも不要で、デバイスを最大限にセキュアな状態にしたい量産時にのみ使用します。

この状態を使用するべきでない状況

次の場合は、レベル 2 を使用しないでください:

- 将来的に、SWD を介したアプリケーションのデバッグまたは再プログラミングが必要です
- これにより、TI はデバイスの故障分析を実行できます
- SWD を介して一括消去または工場出荷時リセット コマンドを送信し、フラッシュ メモリから機密情報を削除します

注

デバイスをレベル 2 (SW-DP ディスエーブル) で構成した後は、SWD 経由でデバイスにアクセスすることは**できなくなります**。デバイスを SWD アクセスが可能なレベル 0 またはレベル 1 の状態に戻すことができるのは、BSL と工場出荷時リセットの両方がイネーブル (BSL で工場出荷時リセット コマンドを送信可能) になっている場合か、またはユーザー アプリケーション コードに含まれるメカニズムにより、NONMAIN 構成を制限の少ない状態に変更できる場合のみです。いずれの場合も、NONMAIN が静的な書き込み保護によってロックされている場合は、レベル 2 状態から変更できず、SWD アクセスを回復することはできません。

2.6.1.2 ブートストラップ ローダ (BSL) のイネーブル / ディセーブル ポリシー

ブートストラップ ローダ (BSL) では、シリアル ワイヤ デバッグ インターフェイスではなく、標準のシリアル インターフェイス (UART または I2C) を使用して、デバイス メモリをプログラムおよび検証できます。BSL には独自の構成ポリシーがありますが、BSL の起動をイネーブルにするか、ディセーブル (起動不可) にするかは、BCR が決定します。

BSL には追加の攻撃対象領域が存在するため、アプリケーションで BSL を使用しない場合は、ユーザー指定のブート セキュリティ ポリシーでディセーブルにできます。BSL をアプリケーションで使用する場合、BSL セキュリティ設定 (BSL アクセス用のパスワードを含む) は BSL 構成ポリシーで管理されます。

2.6.1.3 フラッシュ・メモリの保護と整合性ポリシー

フラッシュ・メモリの保護および整合性ポリシーでは、変更されないようにロックするフラッシュ・メモリのセクタと、ユーザー・アプリケーションを開始する前のブート・プロセス中に整合性をチェックするセクタを指定します。

2.6.1.3.1 アプリケーション (MAIN) フラッシュ メモリのロック

MSPM0 マイコンには、静的書き込み保護方式が実装されており、MAIN フラッシュ領域のユーザー定義セクタを実行時のプログラム / 消去操作からロックアウトできます。目的の静的書き込み保護方式は、NONMAIN フラッシュ領域のブート セキュリティ ポリシーの一部として構成されます。

目的

静的書き込み保護により、次の特性を持つ固定のユーザー定義アプリケーションをフラッシュ メモリに配置できるようになります:

- 一度プログラムされてロックされると、そのアプリケーションはアプリケーション コードや ROM ブートローダーによって変更することはできません
- アプリケーションがフラッシュ メモリの先頭に配置されている場合、ROM ブート構成ルーチンが実行をユーザー アプリケーションに移した際に、このアプリケーションが最初に実行されるコードになります

MSPM0 静的書き込み保護は両方の特性をサポートしており、セキュア ブート イメージ マネージャを実装するにはこれらの特性を満たす必要があります。

機能

NONMAIN のセクタが書き込みロックされている場合、ブート構成ルーチンがブートストラップ ローダまたは MAIN フラッシュ内のユーザー アプリケーション コードに実行を移行する場合、機能を変更することはできません。静的に保護されたセクタをアプリケーション コードまたはブートストラップ ローダでプログラムまたは消去しようとすると、ハードウェア フラッシュ動作エラーが発生し、セクタは変更されません。

静的書き込み保護により、アプリケーション コードやブートローダーからの変更は防止されますが、SWD インターフェイスを介して送信される一括消去や工場出荷時リセット コマンドは受け付けられます。この動作が望ましくない場合は、一括消去または工場出荷時リセットを行う SWD コマンドを固有のパスワードを使用して保護するか、ディセーブルできます (SWD ポリシーを参照)。静的に書き込み保護された MAIN フラッシュ セクタを変更する手段を完全に削除するには、一括消去および工場出荷時リセット コマンド (または SW-DP) をディセーブルする必要があります。また、NONMAIN ブート構成メモリを静的に書き込み保護し、アプリケーション コードが NONMAIN 領域の内容を変更して下位の書き込み保護方式を変更するのを防止する必要があります。これについては、次のセクションで説明します。

2.6.1.3.2 構成 (NONMAIN) フラッシュ メモリのロック

MSPM0 MCU には、静的書き込み保護機能が実装されており、NONMAIN フラッシュ領域を実行時のプログラム / 消去操作からロックアウトできます。書き込み保護機能は、NONMAIN フラッシュ領域のブート セキュリティ ポリシーの一部として構成されます。

目的

デフォルトでは、NONMAIN 構成メモリ (ユーザー指定のブート セキュリティ ポリシーとブートストラップ ローダ ポリシーを含む) は書き込み保護の状態になっていません。これにより、プロビジョニング中にユーザーが NONMAIN を消去し、量産時に使用されるユーザー指定のポリシーで再プログラミングすることができます。

多くの場合、構成メモリはプロビジョニング後にロックするのが適切です。構成メモリをロックすると、セキュリティ ポリシー、ブートストラップ ローダ ポリシー、静的書き込み保護ポリシーが、ブートストラップ ローダまたはアプリケーション コードによって不正に変更されるのを防止できます。ほとんどのアプリケーションでは、量産デバイスの構成メモリの変更は、デバイスのファームウェアが更新された場合でも必要ありません。

機能

保護するように構成した場合、NONMAIN 領域全体が書き込みロックされ、ブート構成ルーチンがブートストラップ ローダまたは MAIN フラッシュ内のユーザー アプリケーション コードに実行を移行したときに、機能を変更することはできません。NONMAIN 領域をアプリケーション コードまたはブートストラップ ローダでプログラムまたは消去しようとすると、ハードウェア フラッシュ動作エラーが発生し、セクタは変更されません。

静的書き込み保護は、アプリケーション コードまたはブートローダによる変更は防止しますが、SWD インターフェイス経由で送信される工場出荷時リセット コマンドは実行されます。この動作が望ましくない場合は、工場出荷時リセットを行う SWD コマンドを固有のパスワードを使用して保護するか、ディセーブルできます (SWD ポリシーを参照)。NONMAIN 構成メモリを変更する手段を完全に削除するには、工場出荷時リセット コマンドと テキサス・インスツルメンツ FA (または SW-DP) をディセーブルする必要があります。

注

NONMAIN が静的に書き込み保護されており、工場出荷時リセット コマンドと テキサス・インスツルメンツ FA (または SW-DP) がディセーブルの場合、NONMAIN は変更不可能な読み取り専用メモリと同等であり、いかなる方法でもデバイス構成を変更することはできません。さらに、MAIN メモリ領域セクタのいずれかが静的保護機能を使用して構成されている場合、これらのセクタはいかなる方法でも変更できず、変更不可能とみなすことができます。

2.6.1.3.3 アプリケーション (MAIN) フラッシュ メモリの整合性の検証

BCR は、BCR (ROM 内) からユーザー アプリケーション (MAIN フラッシュ メモリ内) に実行を移行する前に、MAIN フラッシュ メモリ内のユーザー指定アドレス範囲のデータ整合性をチェックすることをサポートしています。

目的

整合性チェックは、ブート ROM (通常はセキュア ブート イメージ マネージャ) の後に最初に実行されるコードに対して、予測される値と一致する CRC/SHA256 ダイジェストが含まれていることを確認するための追加ステップとして使用できます。この整合性チェックにより、フラッシュ メモリ内の重要なコード (残りのユーザー アプリケーション ソフトウェア イメージの認証を実行) が破損しているためにセキュリティの脆弱性が発生する可能性が低減されます。

機能

開始アドレス、長さ、および ISO-3309 CRC-32 または SHA2-256 ダイジェストを、NONMAIN 構成メモリにプロビジョニングできます。ブート プロセス中に、BCR は MAIN フラッシュ メモリ内の指定された範囲の CRC-32 ダイジェストを計算し、計算されたダイジェストをプロビジョニングされた (予測される) ダイジェストと比較して検証します。値が一致している場合は、ユーザー アプリケーションが開始します。値が一致しない場合は、ユーザー アプリケーションは開始せず、重大なブート エラーが発生します。

2.6.1.4 ブートストラップ・ローダ (BSL) のセキュリティ・ポリシー

BSL セキュリティ・ポリシーは、ブートローダが呼び出されたときにブートローダーにより解釈され、次のパラメータが含まれます。

- BSL アクセス・パスワード (セクション 2.6.1.4.1 を参照)
- BSL 読み出しポリシー (セクション 2.6.1.4.2 を参照)
- BSL セキュリティ・アラート・ポリシー (改ざん検出) (セクション 2.6.1.4.3 を参照)

2.6.1.4.1 BSL アクセス・パスワード

BSL へのアクセスは、ユーザーが指定した 256 ビットのパスワードで保護されます。パスワードをディセーブルするオプションはありません。ほとんどの BSL 機能にアクセスできるようにするには、BSL を呼び出した後にパスワードを入力する必

要があります。パスワードを入力しない場合、使用できる BSL コマンドは Get Identity および Start Application のみです。

BSL に誤ったパスワードを供給すると、BSL は 2 秒間停止し、その後パスワードを入力し直すことができます。パスワード入力に 3 回失敗すると、セキュリティ・アラート機能がアクティブになります (セクション 2.6.1.4.3 を参照)。

2.6.1.4.2 BSL 読み出しポリシー

BSL はオプションで、デバッグや診断の目的でデバイス・メモリの読み出しをサポートできます (パスワードが一致して BSL にアクセスできるようになった後)。デフォルトでは、デバイスから機密コードやデータが抽出されるのを防止するため、この機能はセキュリティ上ディセーブルになっています。BSL 読み出しポリシーがディセーブルの場合、BSL インターフェイス経由でホストに送信できる情報は、最小セグメント長が 1KB であるメモリ・セグメントの CRC32 ダイジェストのみです。デバイス・メモリを直接読み出す必要がある場合は、BSL 構成でイネーブルにできます。

2.6.1.4.3 BSL セキュリティ・アラート・ポリシー

BSL には、改ざんの疑いがある場合に対処するためのアラート・メカニズムがあります。具体的には、1 つの BSL セッション中に誤ったパスワードが 3 回 BSL に渡された場合、セキュリティ・アラートがアクティブになり、BSL は指定されたセキュリティ・アラート・ポリシーに基づいて、次の 3 つの方法のいずれかで応答します。

1. 工場出荷時リセットの発行 (MAIN フラッシュを消去し、NONMAIN フラッシュ領域をリセット)
2. BSL をディセーブル (MAIN フラッシュはそのままにし、NONMAIN を再構成して BSL アクセスをブロック)
3. 無視 (構成は変更せず、パスワードの試行を継続的に許可)

注

オプション 1 および 2 を選択するには、NONMAIN フラッシュ領域が静的に書き込み保護されていることが必要です (セクション 2.6.1.3.2 を参照)。

オプション 1 を選択した場合、静的に書き込み保護されている MAIN メモリ領域 (セクション 2.6.1.3.1 を参照) は、工場出荷時リセット時に消去されません。

2.6.2 カスタマ セキュア コード (CSC) セキュリティ ポリシー

以下のセクションでは、対応デバイスにおいてカスタマ セキュア コード (CSC) によって強制されるポリシーについて説明します。すべてのポリシーは、BOOTRST から抜けた後に実行される CSC の処理中に設定され、デバイスで INITDONE 信号が発行される前にのみ変更できます。INITDONE が発行されて SYSRST が発生した後も、ポリシーは有効なままで、BOOTRST または POR まで変更できません。

2.6.2.1 CSC によるバンク スワップ強制

複数の MAIN フラッシュ バンクを備えるデバイスでは、システム内でアプリケーションを 2 つのバージョンでサポートでき、各バンクに 1 つずつ配置できます。この場合、CSC はイメージのバージョンおよび正当性に基づいて、いずれか一方を実行することを選択できます。決定プロセスについては、セクション 3 セクションで詳しく説明します。

マルチバンク MSPM0 デバイスでのバンク スワップ機能のガイダンスは、MSPM0 ファミリのフラッシュ マルチ バンク機能に記載されています。

2.6.2.2 CSC が適用したファイアウォール

CSC 対応デバイスには、デバイス上でアクティブ化できるさまざまなファイアウォールが含まれています:

- MAIN フラッシュ書き込み保護ファイアウォール:INITDONE 以降、指定されたフラッシュのセクタが書き込みおよび消去不可になる仕組みです。これは、バンクの書き込み / 実行の除外に加えて行われます。
- MAIN フラッシュ読み取り・実行保護ファイアウォール:アプリケーションから読み取りおよび実行ができないようにする、フラッシュの指定領域です。この領域に読み出すと、すべて 0 が返されます。
- MAIN フラッシュ IP 保護ファイアウォール:フラッシュ バスやデータ バスからは読み取り不可だが、CPU によるフェッチは許可されるフラッシュの指定領域です。コードの特定の実行可能部分を読み取り不可にし、機密性の高いアルゴリズムを読み取れるようにします。これは、バンクの書き込み / 実行の除外に加えて行われます。

- DATA フラッシュ書き込み保護ファイアウォール：DATA バンクが存在する場合、指定されたセクタはアプリケーションから書き込みおよび消去ができなくなります
- DATA フラッシュ読み取り保護ファイアウォール：DATA バンクが存在する場合、指定されたセクタはアプリケーションから読み取ることができません。この領域に読み出すと、すべて 0 が返されます

読み取り保護ファイアウォールは、アプリケーションから機密情報を隠すために使用でき、書き込み保護ファイアウォールは、後から変更できない情報をアプリケーションに渡すために使用できるため、その情報は信頼できるものと見なすことができます。

複数のバンクを持つデバイスでは、ファイアウォールもバンク間でミラーリングされます。これは、フラッシュサイズが 0x4.0000 の 2 バンク デバイス(バンク開始アドレスが 0x0000000 と 0x2.0000)において、0x5000～0x6000 に読み取り保護ファイアウォールを設定すると、アドレス範囲 0x5000～0x6000 と 0x2.5000～0x2.6000 の両方から、すべて 0 が返されることを意味します。

2.6.2.3 KEYSTORE への CSC キー書き込み

AES キーは KEYSTORE に安全に格納できます。KEYSTORE は、INITDONE が発行される前の CSC 実行中にのみアクセス (読み出しおよび書き込み) できます。アプリケーション プログラムの実行中 (INITDONE 後) は、アプリケーションが AES エンジンにロードされる KEYSTORE のキーを制御できますが、ロード処理の全過程において、キーはアプリケーションからは可視ではありません。詳細については、セクション 4.5 を参照してください。

2.6.3 構成データのエラー耐性

MSPM0 デバイスには、NONMAIN 構成メモリにデータ・エラーが発生してセキュリティが失われる可能性を削減するため、複数のメカニズムが備えられています。

2.6.3.1 CRC で保護された構成データ

NONMAIN メモリの BCR 構成データと BSL 構成データ構造には、それぞれの構造の CRC ダイジェストに対応する CRC 値が含まれています。デバイスのブート プロセス中に、BCR はデータ構造の CRC ダイジェストを計算し、格納されている CRC 値と比較して、構造体内に含まれるデータが信頼できるものかを確認します。

BCR 構成の CRC エラーの処理

ブート中に BCR 構成データ (SWD ポリシー、BSL イネーブル / ディセーブル ポリシー、フラッシュ メモリ保護および整合性チェック ポリシーを含む) の CRC チェックがエラーとなった場合、致命的なブート エラーが発生し、以下の制限が課されます。

- エラーの原因をブート診断として CFG-AP に記録
- BSL はイネーブルに設定されていても起動しない
- ユーザー アプリケーションを開始しない
- アプリケーションのデバッグ アクセスはすべてディセーブル
- 保留中の SWD 工場出荷時リセット コマンドがイネーブルの場合、またはパスワードを使用してイネーブルになった場合は、それを実行
- 保留中の テキサス・インスツルメンツ故障解析フロー エントリがイネーブルの場合は適用
- ブート プロセスを最大 3 回再試行
  - 2 回目または 3 回目の試行で成功した場合、デバイスを通常どおり起動
  - 3 回目の試行に失敗した場合、次に BOR または POR が実行されるまで、それ以上のブート試行は許可しない

この CRC チェックの利点は、静的書き込み保護構成 (セキュア ブートの中核) などの構成データに反転ビットがある場合に、それをブート プロセス中に高い信頼性で検出できることです。エラー処理手順では、BSL およびユーザー アプリケーションの実行が明示的に防止され、サポートされているオプション (SWD 工場出荷時リセットおよび TI FA) のみが 16 ビットのパターン一致フィールドによって保護されます。

BSL 構成の CRC エラーの処理

BSL 呼び出し中に BSL 構成データ (BSL パスワードおよび BSL ポリシーを含む) の CRC チェックがエラーとなった場合、致命的なブート エラーが発生し、以下の制限が課されます。

- このエラーの原因をブート診断として CFG-AP に記録

- BSL はイネーブルに設定されていても起動しない
- ユーザー アプリケーションを開始しない
- アプリケーションのデバッグ アクセスはすべてディセーブル
- ブート プロセスを最大 3 回再試行
  - 2 回目または 3 回目の試行で成功した場合、デバイスを通常どおり起動
  - 3 回目の試行に失敗した場合、次に BOR または POR が実行されるまで、それ以上のブート試行は許可しない

この CRC チェックの利点は、BSL 構成データ内に反転ビットがある場合に、それを起動プロセス中に高い信頼性で検出できることです。このエラー処理手順により、BSL が無効なデータから開始するのを防止できます。無効なデータから開始すると、セキュリティが失われる可能性があります。

テキサス・インスツルメンツの工場出荷時トリム データの CRC エラー処理

ユーザーが指定した構成データに加えて、ブート中に テキサス・インスツルメンツの工場出荷時トリムの CRC チェックがエラーとなった場合も、致命的なブート エラーが発生し、以下の制限が発生します。

- エラーの原因をブート診断として CFG-AP に記録
- BSL はイネーブルに設定されていても起動しない
- ユーザー アプリケーションを開始しない
- アプリケーションのデバッグ アクセスはすべてディセーブル
- 保留中の テキサス・インスツルメンツ故障解析フロー エントリがイネーブルの場合は適用
- ブート プロセスを最大 3 回再試行
  - 2 回目または 3 回目の試行で成功した場合、デバイスを通常どおり起動
  - 3 回目の試行に失敗した場合、次に BOR または POR が実行されるまで、それ以上のブート試行は許可しない

2.6.3.2 クリティカル・フィールドの 16 ビット・パターン一致

SWD セキュリティ・ポリシーなどの BCR 構成メモリ内の重要なポリシーは、NONMAIN メモリ内の 16 ビット・パターン一致フィールドとして実装され、以下の特徴があります。

- より低いセキュリティ状態をイネーブルにするにはパターンが正確に一致することが必要
- 16 ビット・フィールドのいずれかの値が定義されたパターンに正確に一致しない場合、対応するパラメータが最大のセキュア状態になる

これにより、1 ビットの反転によって、デバイスが指定されていたものより低いセキュリティ状態に移行してしまうことが防止されます。

3 セキュア ブート

MSPM0 デバイスは、ハードウェア機能とソフトウェア機能の組み合わせにより、アプリケーション ソフトウェア (セキュア ブート) の認証をサポートしています。非対称型および対称型の認証方式がサポートされていますが、すべての MSPM0 デバイスにソフトウェアの悪用から対称キーを保護するためのセキュア ストレージが搭載されているわけではありません。

MSPM0 アーキテクチャには、セキュア ブートを実現するために必要ないくつかの重要なハードウェア機能が含まれています。

- 固定認証ファームウェアと認証キーを格納するためのロック可能なフラッシュ メモリ
- ブート中のエントリ ポイントを 1 つにし、セキュア ブート イメージ マネージャが常に BCR の後実行される最初のアプリケーションであることを保証

MSPM0 ソフトウェア開発キット (SDK) には、MSPM0 マイコン上でセキュア ブートを実装するためのブート イメージ マネージャ (BIM) とカスタマ セキュア コード (CSC) のリファレンス アプリケーションが含まれています。このリファレンス アプリケーションは、MSPM0 デバイスに簡単に構成およびプロビジョニングできます。

3.1 セキュア処理環境の分離

一部のセキュアブートプロセスでは、セキュア処理環境 (SPE) を非セキュア処理環境 (NSPE) から分離するためのハードウェア機構が必要とされ、アプリケーション ファームウェアに対するあらゆる更新は、実行直前に信頼の起点 (RoT) によって完全性および真正性を検証する必要があります。

MSPM0 ファミリの一部のデバイスでは、このような分離メカニズムがハードウェアとして提供されており、INITDONE 以前は CPU が信頼された環境 (特権状態) で実行され、INITDONE 以降は信頼されていない環境 (非特権状態) で実行されることを保証します。プログラムが特権状態 (INITDONE より前) で実行された場合、CPU には以下の権限があり、INITDONE 後にこれらの設定を変更することはできません。

- セクション 4.5 で AES キーを設定します
- 銀行スワップ ポリシーを設定して
- データの書き込み保護、読み取り実行保護、または IP 保護のためにセクション 4 を設定。
- アプリケーション プログラムセクション 2.6.1.3.3。

注

アプリケーション プログラムの完全性および真正性の検証は、INITDONE に直接ひも付いたハードウェア機能ではなく、CSC ソリューションにおける特権状態で実現されます。

他の MSPM0 デバイスにはこのような絶縁メカニズムが搭載されていないため、すべての MAIN フラッシュ プログラムが同じ権限で実行されます。デバイスにハードウェア絶縁メカニズム (INITDONE) が搭載されているかどうかに応じた、MSPM0 デバイスのセキュア ブート機能の概要については、表 3-1 を参照してください。

表 3-1. MSPM0 のセキュア ブート機能の比較

デバイス	MSPM0Gx10x、MSPM0Gx50x、MSPM0L130x	MSPM0L111x、MSPM0Lx22x、MSPM0Gx51x
INITDONE	なし	あり
セキュア ブート ソリューション	ブート イメージ マネージャ (BIM)	カスタマ セキュア コード (CSC)
キーストア	なし	あり
バンク スワップ	なし	あり
ファイアウォール	なし	あり
CMAC	なし	あり
ECDSA+SHA256	ソフトウェアでサポートされます	ソフトウェアでサポートされます

3.2 カスタマ セキュア コード (CSC)

カスタマ セキュアコード (CSC) は、ハードウェア分離メカニズム (INITDONE) を搭載した MSPM0 デバイス用のセキュア ブート ソリューションです。図 3-1 に、CSC のブートおよびスタートアップ シーケンスを示します。BOOTRST では、TI

ROM ブートコードの実行が開始されます。正常にブートが完了すると、ブートコードは BOOTDONE を発行します。この時点で、SYSCTL は SYSRST をデバイスに発行し、MAIN フラッシュ メモリからの実行をトリガします。MAIN フラッシュ プログラムは、ブートコード完了後、常に物理アドレス 0x0004 ベクタ (リセット ハンドラ) から開始されます。NONMAIN フラッシュの BCR における CSCEXISTS の設定に応じて、BOOTDONE の後には 2 つの実行フローがあります:

- CSCEXISTS が設定されている場合、CSC ブート シーケンスが有効となり、MAIN フラッシュのプログラムは INITDONE がクリア状態のまま開始されます。この場合、ユーザーは CSC ファームウェア (MSPM0 SDK CSC の例) を MAIN フラッシュの 0x0000 アドレスに配置する必要があります。CSC ファームウェアは、NONMAIN BCR の構成によって静的書き込み保護される必要があります。
- CSCEXISTS クリア:CSC ブート シーケンスは許可されず、MAIN フラッシュ プログラムは INITDONE が設定状態で開始されます。セキュリティ関連のポリシーは設定できず、この場合にはアプリケーション ファームウェアを MAIN フラッシュの 0x0000 アドレスに配置する必要があります。

注

MAIN フラッシュ プログラムは、BOOTDONE の後の物理アドレス 0x0004 から常に開始されます。バンク スワップのポリシーは BOOTRST の際にリセットされるため、BOOTDONE の後、MAIN フラッシュのプログラムは常にバンク スワップなしで開始されます。バンク スワップは、NONMAIN の設定で CSCEXISTS と FLASHBANKSWAPPOLICY の両方が有効になっており、かつ INITDONE の後にのみ有効になります。

CSC が存在する場合、実行するバンクの決定、メモリ領域の保護設定、KEYSTORE へのセキュア キー初期化、アプリケーション プログラムの完全性および真正性の検証などは CSC が担当します。このときデバイスは、これらのセキュリティ ポリシを設定する権限を持つ特権状態で動作します。INITDONE が発行されます (SYSCTL.SECCFG.INITDONE へ書き込み、レジスタ定義についてはデバイス固有のテクニカル リファレンス マニュアルを参照)。CSC の最後に、SYSCTL が 2 番目の SYSRST を発行します。以下に示すすべてのセキュリティ ポリシーは INITDONE 中に有効になり、次の BOOTRST まで変更することはできません:

- ファイアウォール保護ポリシー
- バンク スワップ ポリシー
- キーストア保護

INITDONE の後、デバイスは特権なし状態になり、MAIN フラッシュのアドレス 0x0004 から再度実行を開始し、CSC が再度実行します。今回、CSC は INITDONE がすでに以前に発行されていることを検出し(これは SYSCTL.SECCFG.SECSTATUS.INITDONE ビットを読み取ることで判定されます)、そのままメイン アプリケーションへ直接ジャンプします。特権ステート (pre-INITDONE) および非特権ステート (post-INITDONE) での CSC 実行フローについては、図 3-2 を参照してください。

ブートおよび起動シーケンスの詳細については、MSPM0 G シリーズ 80MHz マイクロコントローラ テクニカル リファレンス マニュアル (Rev. C) の「セキュリティ」の章を参照してください。

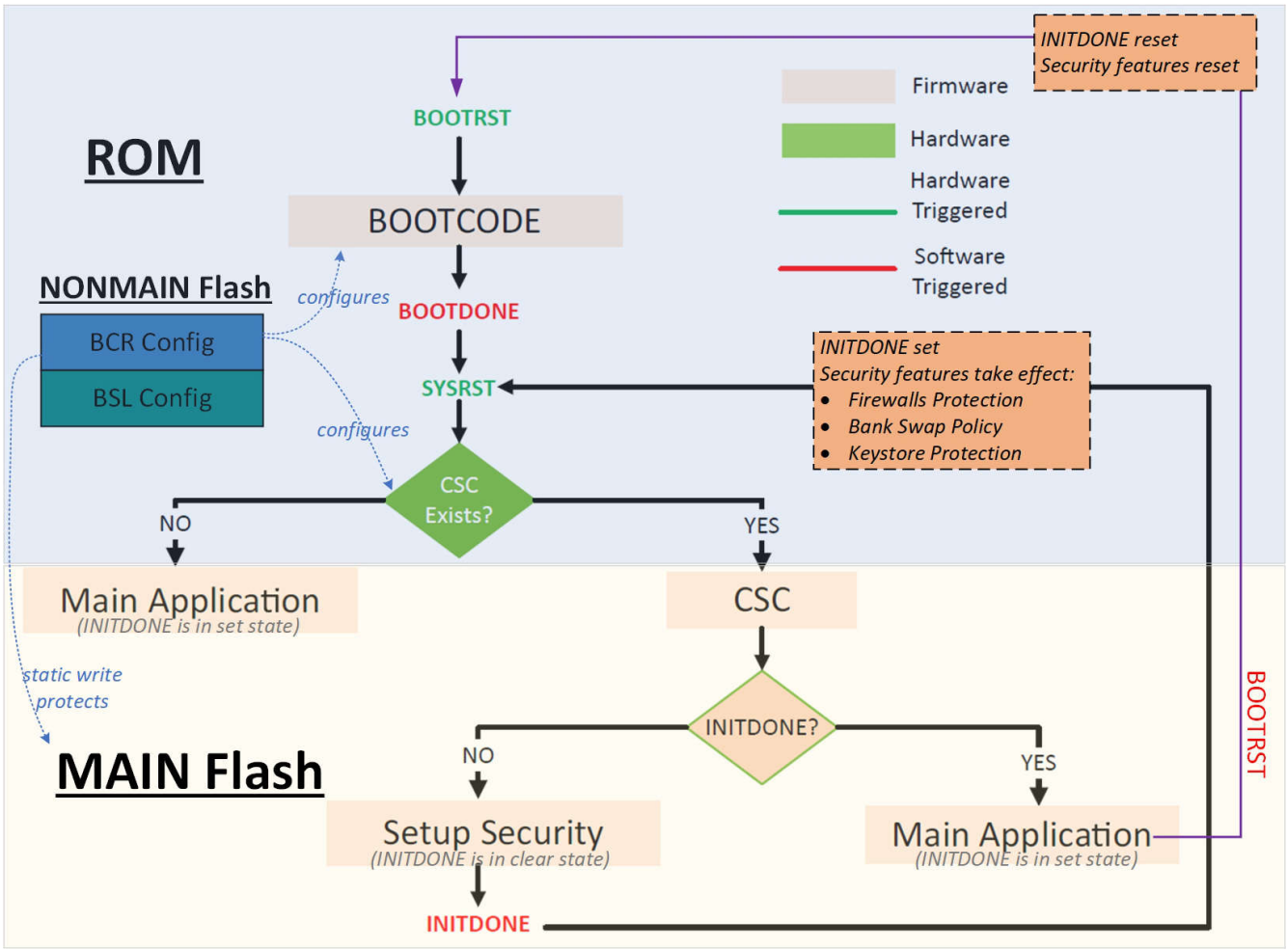

図 3-1. CSC のブートおよびスタートアップ シーケンス

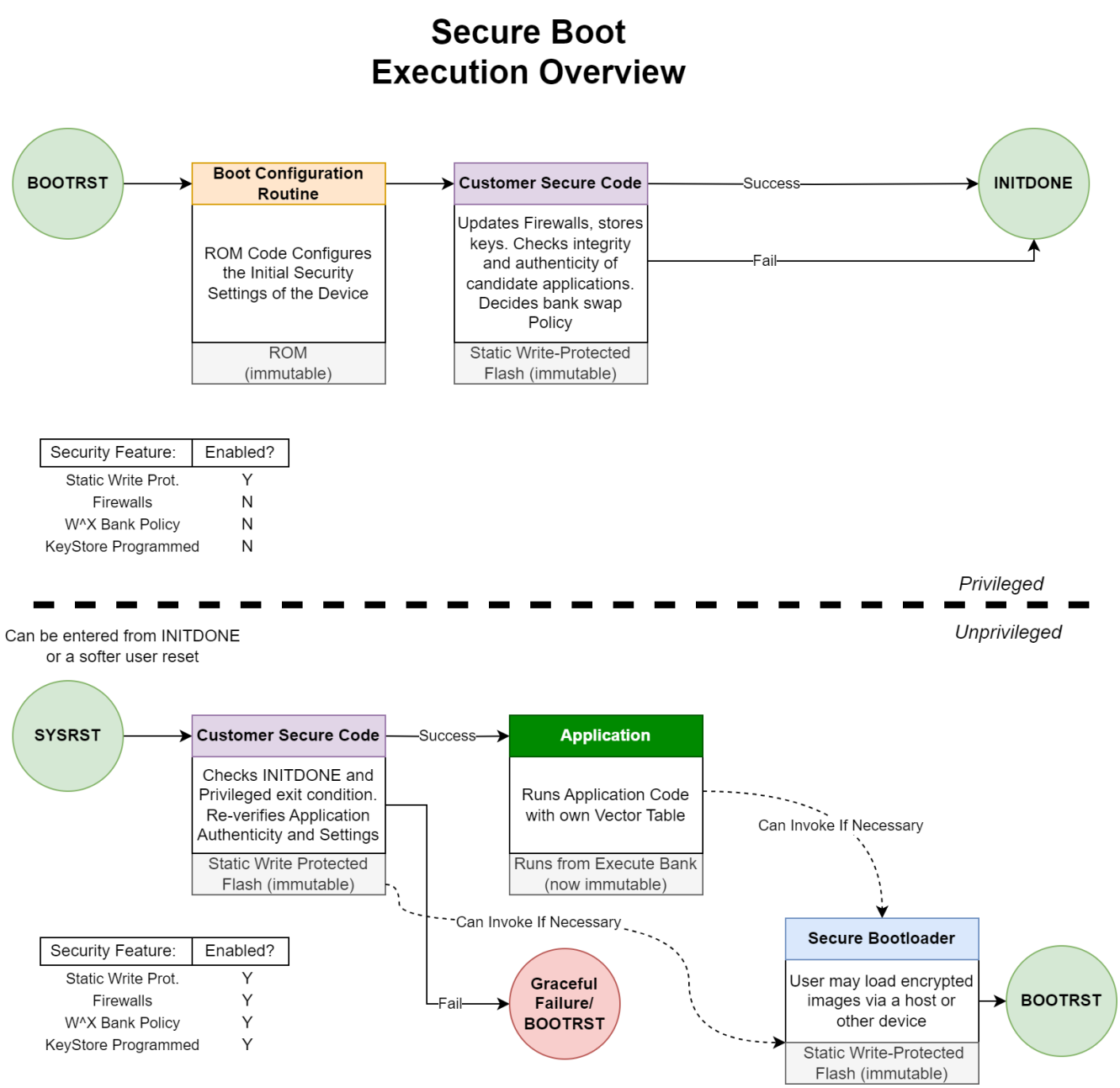

図 3-2. CSC 実行の概要

3.2.1 セキュア ブートのフロー

このセクションでは図 3-3、に示すように、MSPM0 SDK CSC の例 (MSPM0 SDK 2.08.00.03) に基づく CSC ソリューションのブートフローの詳細について説明します。実行フロー全体は、図 3-1 および図 3-2 に示すフローチャートとほとんど互換性があります。

ROM ブートコードの実行が完了すると、最初にプログラムが CSC ファームウェアに移行するときに、INITDONE (SYSCTL.SECCFG.SECSTATUS.INITDONE) はクリア状態になります。CSC は最初に特権状態で動作します。両方のフラッシュ バンクから最も高いバージョンのイメージを検索し，バージョンのロールバックをチェックした後，対称方式 (ハードウェア AES-CMAC) または非対称方式 (ソフトウェア SHA256+ECDSA) によってアプリケーション イメージの正当性と整合性を検証します。検証が渡されると、CSC はロールバック カウンタ、CMAC タグ、秘密キー、および KEYSTORE を更新します。その後，SECRET フラッシュ領域およびロック可能なフラッシュ領域にファイアウォールを設定し，バンク スワップ ポリシーを決定します。CSC は INITDONE を発行して SYSRST をトリガし、デバイスは特権なし状態になります。

デバイスは、INITDONE がセット状態になった状態で、再度 CSC ファームウェアから実行されます。前回のブート ステータスが正常であることを確認した後，CSC はアプリケーション イメージにジャンプしてアプリケーションを開始します。

CSC の実行フローの例に関連する重要な注意事項がいくつかあります:

- NONMAIN フラッシュの CSCEXISTS および FLASHBANKSWAPPOLICY フィールドをイネーブルにして、CSC シーケンス全体をイネーブルにする必要があります。
- PB0 は物理バンク 0 を表します。バンク スワップ ポリシーは特権状態 (INITDONE より前) では有効ではないため、特権状態 CSC で使用されるフラッシュ アドレスは常に物理アドレスを参照します。
- PB0 と PB1 の 2 つのイメージが同じバージョンの場合、PB0 イメージが検証され、より高い優先度で実行されます。
- 最も高いバージョンのイメージが SHA256+ECDSA 検証に合格しない場合、もう一方のバンクのイメージ (存在する場合) はすぐに検証されます。
- 非対称認証の場合，まずアプリケーション コードのセキュア ハッシュ (SHA2-256) ダイジェストをソフトウェアで計算し，その後ソフトウェア ECDSA がファームウェア内の公開鍵に基づいてイメージ署名を検証します。
- 対称型 AES-CMAC アルゴリズムは、ファームウェアの更新が検出されない場合にアプリケーション イメージを検証するための時間節約メカニズムです。AES-CMAC はハードウェア アクセラレーションされているため，タグを単純にチェックし，非対称方式で検証された以降に変更されていないことを確認するだけで，処理が格段に高速になります。AES-CMAC 方式は BOOTRST が発生した場合にのみ適用され、前回の BOOTRST 以降、フラッシュ メモリにより高いバージョンのイメージが配置されていない場合に限って実行されます。
- SECRET フラッシュ領域は、ユーザーが指定する領域で、機密情報を格納し、ファイアウォールによって読み取りおよび実行が保護されています。ロック可能なフラッシュ領域は、変更しない情報を格納し、ファイアウォールによって書き込み保護されているユーザー指定の領域です。詳細については、フラッシュ メモリのマッピングを参照してください。

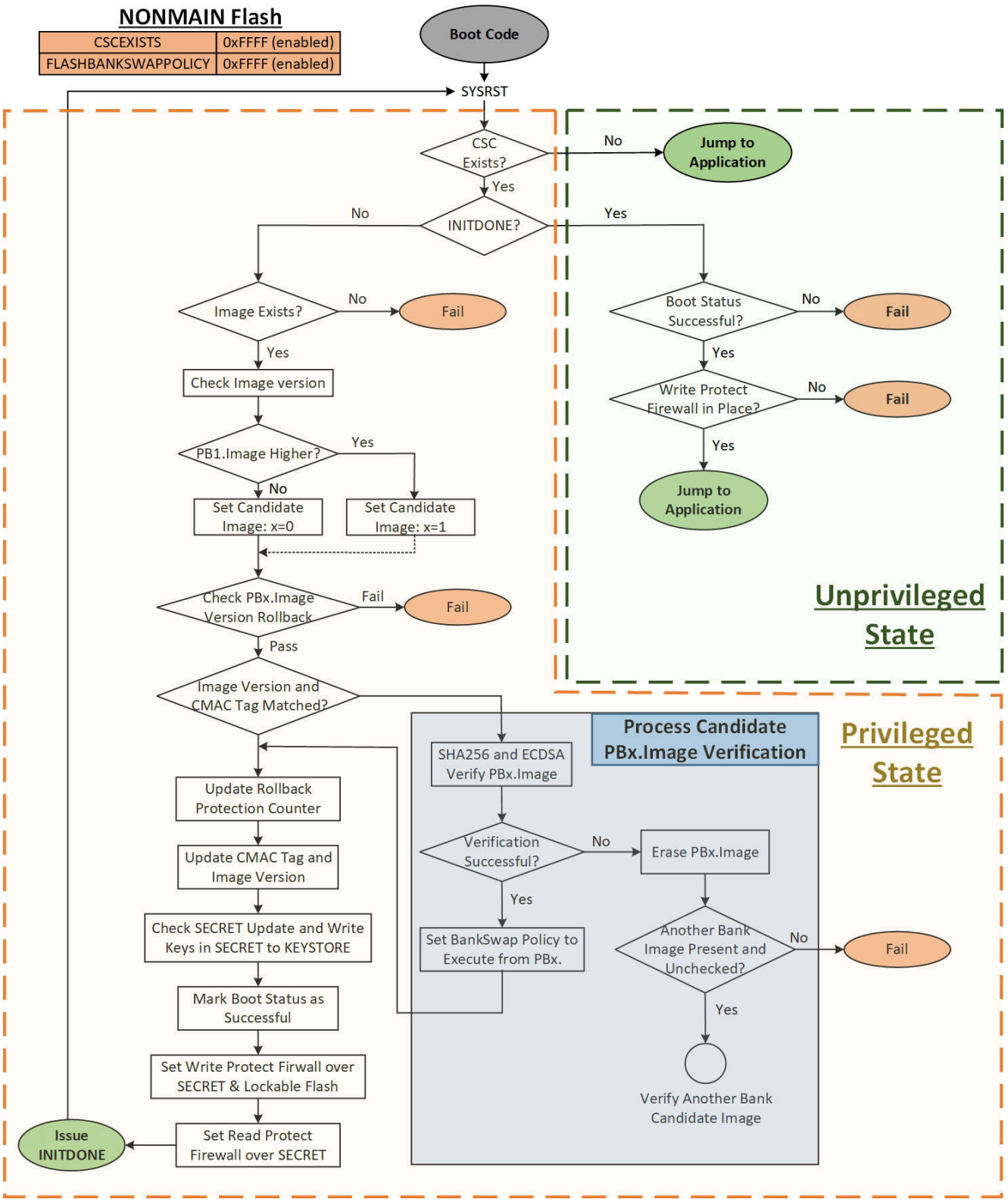

図 3-3. MSPM0 SDK CSC 実行フロー

3.2.2 フラッシュ メモリ マップ

図 3-4 に、CSC セキュア ブートの詳細なフラッシュ メモリ マップを示します。CSC のセクションの説明を次に示します:

- **シークレット**: SECRET は特権実行フローからは見えるが，読み取り保護ファイアウォールによって保護されるため，非特権フロー (CSC およびアプリケーション) からは一切見えなくなります。SECRET 領域は，実行時に KEYSTORE にロードされる不揮発性キーを格納するために使用できます。したがって、非特権コードはこれらのキーを使用できますが、読み取りアクセスはできません。また、CMAC タグやキーなどの追加情報を含めるようにユーザーがカスタマイズすることもできます。
- **ロック可能なフラッシュ**:ロック可能なフラッシュは、特権状態によって書き込まれ、特権なしフローによって読み取られるが変更されない重要な情報に対する動的な書き込み保護を提供します。この領域に格納される典型的なものとしては，セキュリティカウンタ (ロールバック保護)、キー ストア ハッシュ テーブル，およびイメージ ハッシュがあります。ロック可能なコンテンツは、フラッシュ バンク 0 とバンク 1 の両方で CSC 領域にプログラムされ、両方のバンク アプリケーション プログラムが同じ方法でこの領域にアクセスできるようになります。
- **CSC 割り込みベクタ**:これらはカスタマ セキュア コード用の割り込みベクタです。この割り込みベクタ テーブルは，BOOTRST または SYSRST が発生した場合に，常にフラッシュから最初に実行されます。これは、VTOR が両方のリセットでクリアされるため、強制的に実行されます。つまり、0x0000 が使用されます (これは論理バンク 0 を指し、不変 CSC のコピーが存在します)。
- **CSC コード**:メイン コードとセキュリティ プリミティブが，カスタマ セキュア コードの大部分を占めます。これは割り込みベクトルとともに、両方のバンクに複製されます。プライマリ バンクとセカンダリ バンクの両方のイメージは同一であるべきで，コードが 0x0000 から実行されているかのようにコードへの参照が行われます。バンク スワップ中は，INITDONE により SYSRST が発生した後，プログラムは常に論理バンク 0 (論理アドレス 0x0000) から実行されます。FLASHCTL は，バンク スワップ ポリシーの設定に応じて，アドレス 0x0000 を PB0 の開始アドレス 0x0000 または PB1 の開始アドレス 0x0000 にマッピングします。

注

バンク スワップ可能な構成では、ファイアウォール保護は自動的に両方のバンクにミラーリングされます。

アプリケーション イメージのセクションは次のとおりです:

- 画像ヘッダ、画像 TLV、画像トレーラ:これらの部品は、MCUBOOT が提供する署名ツール imgtool によって生成されます (*<mspm0_sdk_path>\source\third_party\mcuboot\scripts* の Python スクリプトを参照)。これらの内容は生成され，CCS のポスト ビルド ステップでコンパイル済みアプリケーション イメージに統合されます。MSPM0 SDK には、署名済みイメージが CCS 内でどのように構築されているかを示す customer_secure_sample_image の例があります。これらの画像部分の説明を以下に示します:
  - **画像ヘッダ**: アプリケーション イメージのヘッダ情報には，ヘッダ マジック (0x96F3B83D)、イメージ サイズ，イメージ バージョンが含まれます。これは、アプリケーション割り込みベクタの前のアドレス 0x100 バイト (デフォルト) に配置されています。
  - **画像 TLV**:MCUBOOT は，イメージのメタ データを含むタイプ・長さ・値 (TLV) レコードを定義しており、イメージの末尾の後に配置されます。MSPM0 CSC で定義されている TLV には以下が含まれます:TLV マジック (0x6907)、イメージ ハッシュ、ECDSA 公開鍵ハッシュ、ECDSA 署名。詳細については、main·MCU-tools/mcuboot·GitHub の mcuboot/docs/design.md を参照してください。
  - **画像トレーラ**:イメージ フラッシュ領域の末尾に配置される 16 バイトのマジック コンテンツ。

注

SHA256 検証は、アプリケーション割り込みベクタの開始アドレスと画像 TLV の開始アドレスからの画像内容に対してのみ実行されます。

- **アプリケーション割り込みベクタ**:これらはアプリケーションプログラムが使用する個別の割り込みベクタです。CSC がアプリケーションにジャンプする際，ベクタ テーブル オフセット レジスタ (VTOR) はメモリ内のこの位置を指しますす。したがって，以降のすべての割り込みは，この割り込みベクタ セットにリセット リンクを持たずに発生します。開始アドレスは 32 バイト境界に揃える必要があります。

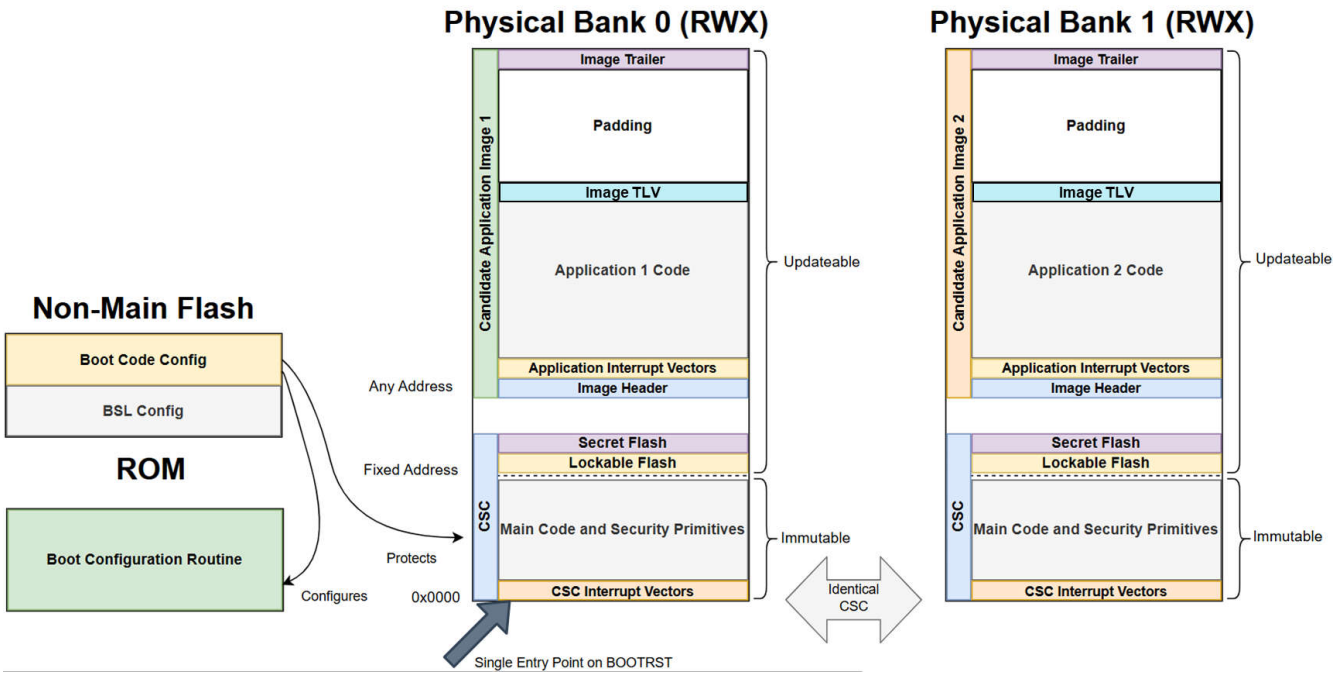

図 3-4. CSC フラッシュマップ

3.2.3 特長

3.2.3.1 CMAC アクセラレーション

CMAC (暗号ベースのメッセージ認証コード) は、データの整合性と真正性を検証するために設計された暗号化アルゴリズムです。CMAC キーを用いて CMAC タグを生成することで動作し、計算速度は処理するイメージの長さに応じて変化します。

これは AES (高度暗号化標準) アルゴリズムを利用しており、ハードウェア アクセラレーションを用いて実装した場合、CMAC は高いセキュリティと高速な処理速度の両方を提供します。これにより、セキュア ブートのシナリオや効率的なメッセージ認証が必要な環境に特に適しています。

完全で時間のかかる完全性および真正性の検証プロセスが必要なのは、新しいイメージの場合のみです。ただし、検証されたイメージが変更されない場合は、前回の認証時に保存された状態情報を利用できます。AES ハードウェア アクセラレーションと組み合わせて CMAC を使用することで、変更されていないイメージの検証プロセスは非常に高速かつ効率的になり、セキュア ブートに要する時間を大幅に短縮し、迅速なシステム起動を可能にします。

3.2.3.2 非対称検証

新しいイメージの整合性と真正性は暗号化アルゴリズムで検証されます。検証済みの画像のみが安全であると見なされ、実行できます。

SHA-256

SHA-256 (Secure Hash Algorithm 256) は、広く使用されている暗号化ハッシュ関数で、任意の入力メッセージから固定長の 256 ビット ダイジェストを生成します。このアルゴリズムは、入力メッセージにわずかな変更が加えられただけでも、出力されるハッシュが大きく変化するように設計されており、入力の変化に対する高い感度を保証します。

その中核的な特徴の 1 つが耐衝突性であり、異なる 2 つのメッセージが同じハッシュ値を生成する可能性は極めて低いことを意味します。この特性により、SHA-256 はデータの完全性を検証する用途において非常に信頼性が高く、いかなる変更も容易に検出できます。

実際には、SHA-256 はデジタル署名やデータ完全性チェックで一般的に使用されています。SHA-256 は、イメージ全体を一意のダイジェストに要約することで、ECDSA アルゴリズムの次の段階に向けた堅牢な基盤を提供します。

ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) は、楕円曲線数学に基づくデジタル署名に使用される暗号アルゴリズムです。これは、RSA のような従来のアルゴリズムと比べて、はるかに短い鍵長で高いセキュリティを提供できるため、効率的で、リソース制約のある環境にも適しています。

ECDSA は現在、MSPM0 セキュア ブートでサポートされている唯一の非対称認証方法です。ECDSA は非対称アルゴリズムで、公開鍵と秘密鍵が別々に存在します。公開鍵はデバイスのフラッシュメモリに保存され、秘密鍵は開発者が安全に管理します。CSC では、セキュアな秘密鍵管理は提供されていません。

ECDSA は、画像のハッシュと公開鍵を使用してデジタル署名を検証し、データの真正性を確保します。このプロセスは対称暗号化オプションに比べてはるかに遅くなりますが、デバイスに重要な脆弱性はありません。

注

最終的な製品展開におけるブートフローを維持するためには、秘密鍵を安全に保管して、管理し、イメージ署名に容易にアクセスできないようにしておくことが非常に重要です。ローカル共有ドライブにキーを保持することは安全な場所ではありません。MSPM0 では現在、セキュアな秘密キー管理は提供されていません。

表 3-2 に、2 つの方式のトレードオフを示します。

表 3-2. セキュア ブート アルゴリズムの比較

パラメータ	非対称型 (SHA2 + ECDSA)	対称型 (CMAC)
認証時間	長い (ソフトウェア ハッシュの計算と公開キーの演算のため)	短い (アルゴリズムが簡素で、ハードウェア AES アクセラレーションが利用可能な場合に活用できるため)
コードのサイズ	大きい (SHA および ECDSA アルゴリズムのため)	小さい (特にターゲット デバイスで AES アクセラレーションを利用できる場合)
キーの整合性	公開キーがデバイスにプロビジョニングされ、変更不可能であることが必要	共有キーがデバイスにプロビジョニングされ、変更不可能であることが必要
キーの機密性	公開キーには機密性の要件はなく、公開キーをアプリケーション コードの脆弱性から保護する必要なし	共有キーは機密情報として保護し、使用しないときには、アプリケーション コードの脆弱性から保護するため、ラップして静的読み取りファイアウォール (ターゲット デバイスでサポートされる場合) で保護する必要あり

ほとんどの状況では、非対称型の実装を推奨します。コード サイズが限られている場合や、認証時間を最小限に抑える必要がある場合には、対称型の実装を使用できます。ただし、共有キーを注意深く管理する必要があります。すべてのデバイスに、ソフトウェアの脆弱性から共有対称キーを保護するためのセキュアなストレージ (KEYSTORE) が搭載されているわけではありません。詳細については、プラットフォーム セキュリティ イネーブラを参照してください。

3.2.3.3 KEYSTORE とファイアウォール

KEYSTORE は AES キーを安全に格納できる保護された SRAM メモリであり、キーは INITDONE の前に CSC で設定されます。アプリケーションは INITDONE 後、KEYSTORE から AES エンジンへのキー転送をトリガできますが、これらのキーに直接アクセス (読み取りまたは書き込み) することはできません。

ファイアウォールは、フラッシュ書き込み保護、フラッシュ読み取り、実行保護、フラッシュ IP 保護などのフラッシュ保護メカニズムで構成されており、これらは CSC で設定され、INITDONE 後に有効になります。

詳細については、セクション 4 を参照してください。

注

バンクスワップ可能構成では、ファイアウォール保護は自動的に両方のバンクにミラーリングされます。

3.2.3.4 CSC 性能

MSPM0 CSC のコード サイズは、コンパイラおよび最適化レベルに関連しています。デフォルトでは、CSC コード サイズ情報は以下のとおりです:

- CSC リージョン サイズ: 18KB
- CSC メイン コード サイズ: 13KB
- SECRET サイズ: 1KB
- ロック ストレージ サイズ: 1KB

CSC サンプルは、メイン コードとシークレットまたはロック ストレージのサイズを変更するようにカスタマイズできます。CSC リージョン サイズを変更する必要がある場合は、それに応じてアプリケーション ファームウェアの開始アドレスを変更する必要があります。詳細については、セクション 3.2.4.4 を参照してください。

さまざまなアルゴリズムに対する CSC タイミング性能を表 3-3 で確認できます。表から分かるように、ハードウェアベースの CMAC 対称方式は、ソフトウェア ベースの SHA256-+ECDSA アルゴリズムと比べてはるかに高速です。そのため、MAIN フラッシュにファームウェア更新がない場合、MCU を非常に効率よくブートできます。

表 3-3. CSC タイミング性能

ECDSA 検証 (SW)	SHA256 (SW)	CMAC (アクセル)
32MHz で約 1.9 秒	~ 5ms/kByte	~ 0.6ms/kByte

3.2.4 クイック スタート ガイド

このセクションでは、本ガイド ドキュメントおよび、イメージ暗号化機能を使用しない MSPM0 SDK の customer_secure_sample_image 例を基に、簡単なステップ バイ ステップのガイダンスを提供します。イメージ暗号化機能を使用した customer_secure_image_with_bootloader の例に関するガイダンスについては、セキュア ブート ユーザー ガイドに含まれる MSPM0 カスタマ セキュア コードおよびブートローダー (CSC) ユーザー ガイドの「バイナリ イメージのロード」セクションを参照してください。

3.2.4.1 環境設定

初期セットアップを実行するには、最新の pip パッケージを備えた Python 3.7 以降がインストールされていることを確認し、必要な要件をダウンロードするために以下の手順を実行してください。

1. コマンド ライン ウィンドウを開き、以下のコメントを実行して、Python が環境にインストールされているかどうかを確認します:

```
python --version
```

2. MSPM0 SDK のインストール パス (*C:\ti\mspm0_sdk_2_08_00_03*) に移動し、必要な要件に応じてコマンド ラインで以下のコマンドを実行します:

```
python -m pip install --user -r source/third_party/mcuboot/scripts/requirements.txt
```

3. これらの Python ライブラリは、*customer_secure_sample_image* プロジェクトのビルド後のステップで、*<mspm0_sdk_path>/source/third_party/mcuboot/scripts* フォルダの下の Python スクリプトによってアプリケーション イメージに署名するために適用されます。

3.2.4.2 ステップ バイ ステップ ガイダンス

Python 環境が適切に設定されている場合は、以下の手順に従って MSPM0 SDK CSC の例で実践します (MSPM0G351x デバイスを例として使用します):

1. SDK パス *<mspm0_sdk_path>\examples\nortos\<mspm0_device>\boot_manager* から、customer_secure_code の例と customer_secure_sample_image の例の両方を CCS ワークスペースにインポートしてください。
2. *customer_secure_code* サンプルと *customer_secure_sample_image* サンプルの両方を作成します。サンプル イメージの例では、EITHER_SLOT_BLUE と EITHER_SLOT_GREEN の両方の構成でビルドすると、プロジェクト内にそれぞれ対応するデバッグ フォルダが生成されているのを確認できます。

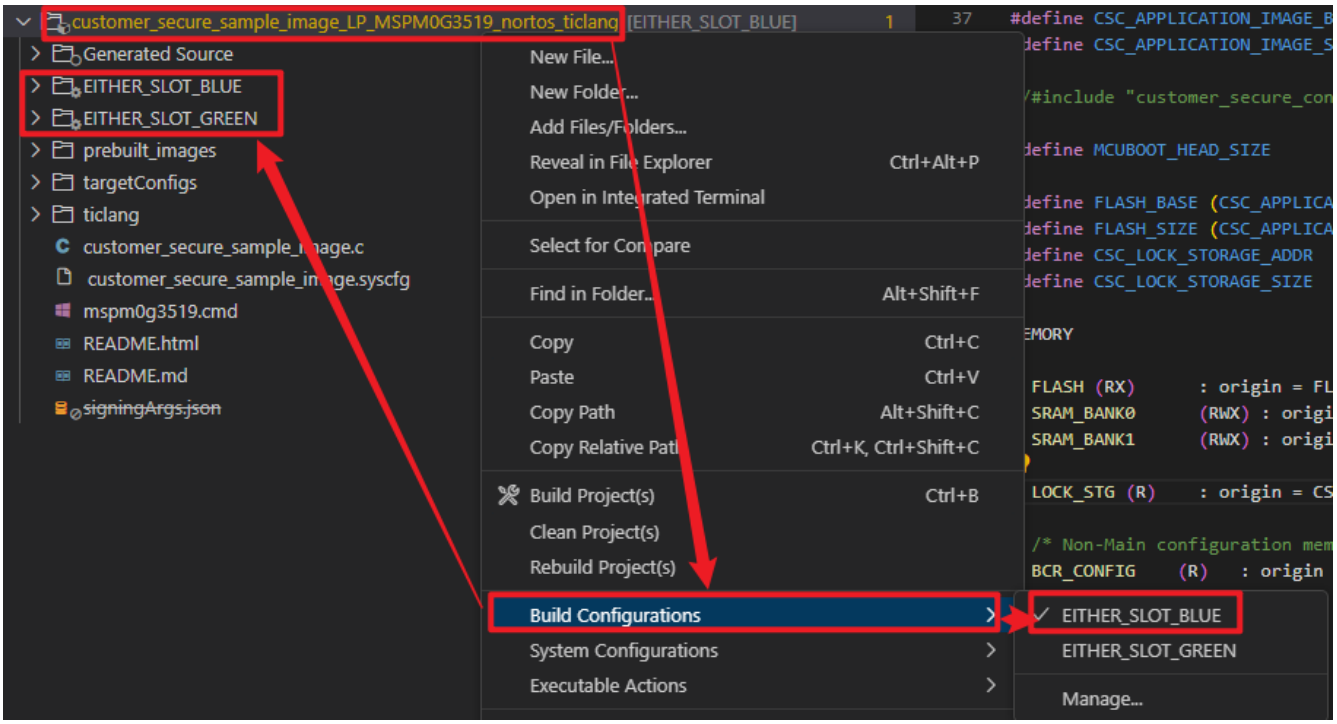

図 3-5. CSC セキュア サンプル イメージ ビルド構成

3. UNIFLASH ソフトウェア プログラミング ツール | TI.com ツールを開き、PC を MSPM0 ローンチパッドに接続して、工場出荷時リセットを実行します。

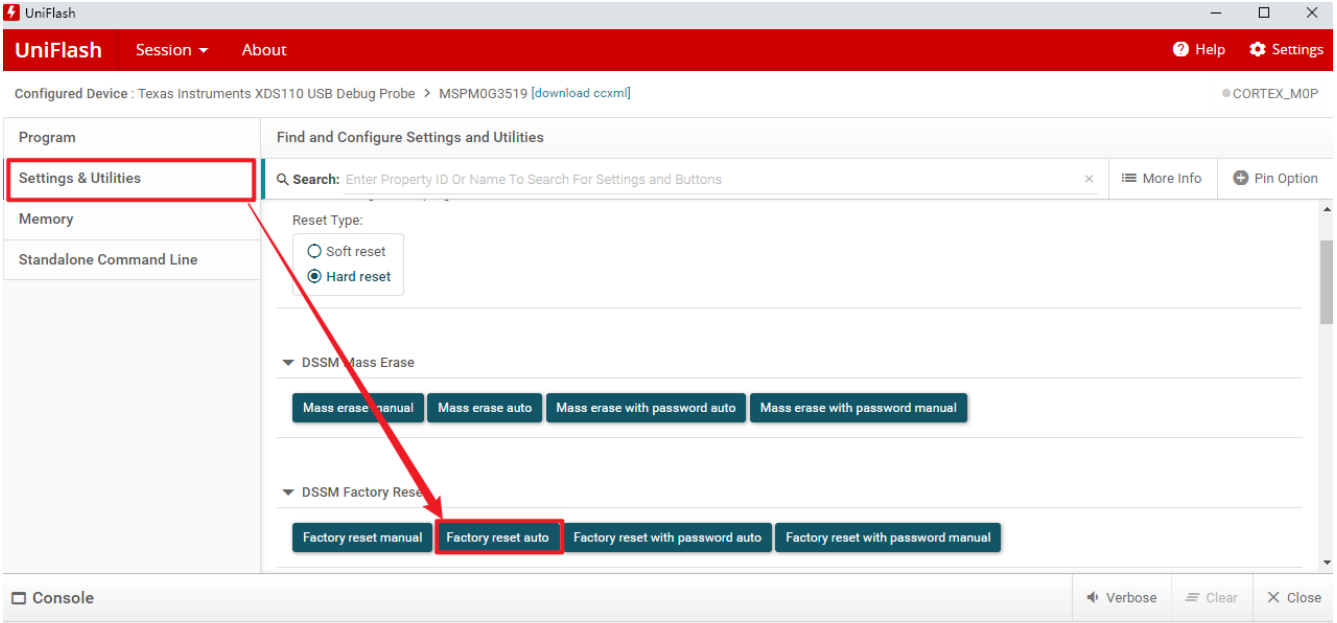

図 3-6. Uniflash による工場出荷時リセット

4. フラッシュ設定を **MAIN および NONMAIN 必要セクタのみを消去**するように構成します。このオプションは画像出力ファイルで使用されているフラッシュ領域のみを消去します。

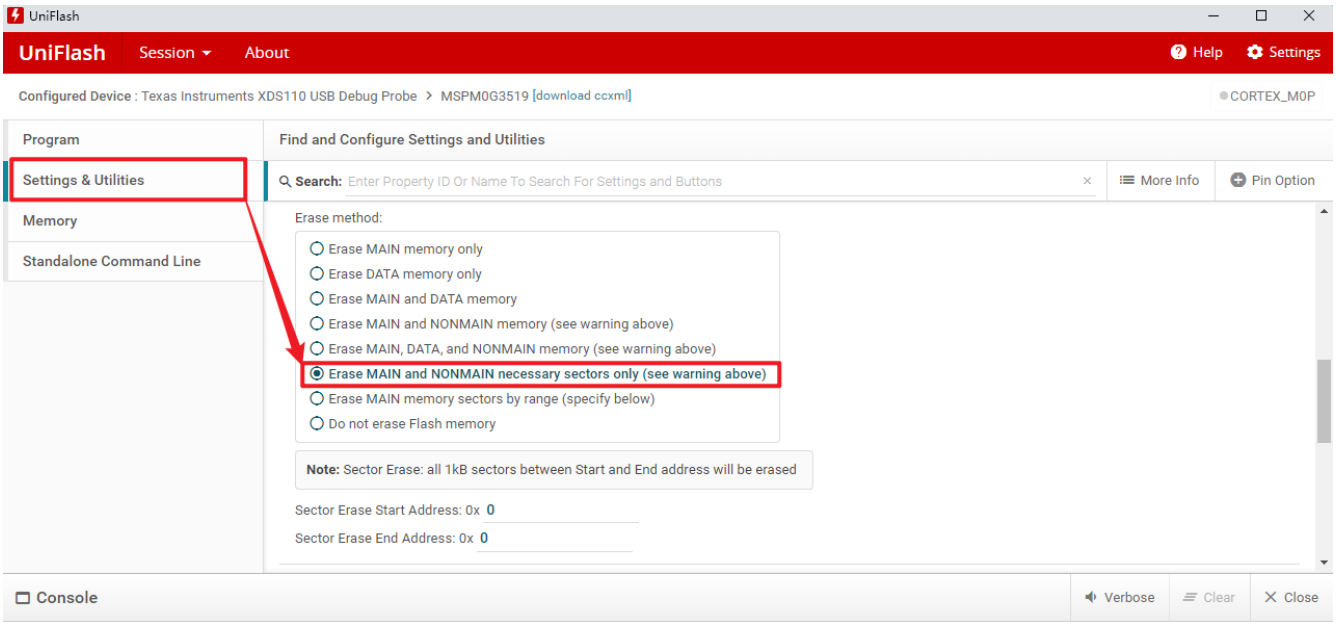

図 3-7. Uniflash の CSC フラッシュ設定

5. 以下のファイルを MSPM0 デバイスにロードし、その後デバイスをパワー サイクルする(または launchpad の NRST ボタンを押す)と、赤色 LED が 2 秒間点灯します(イメージ検証のために CSC を実行します)。その後、青色 LED が点滅します (物理バンク 1 でプログラムが実行されていることを示します)。

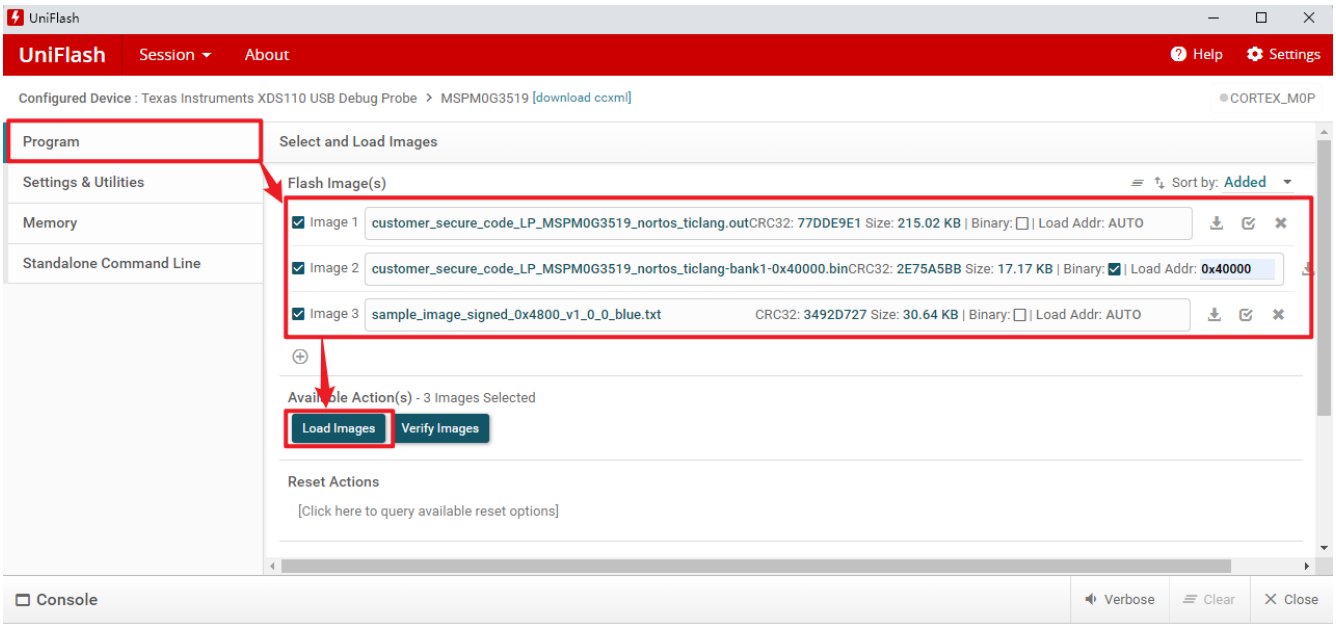

図 3-8. Uniflash で CSC イメージをロードする

- *\customer_secure_code_LP_MSPM0G3519_nortos_ticlang\Debug\customer_secure_code_LP_MSPM0G3519_nortos_ticlang.out*: 物理的な Bank0 に対応する CSC ファームウェアと、必要な NONMAIN 構成に対応するものが付属しています。
- *\customer_secure_code_LP_MSPM0G3519_nortos_ticlang\Debug\customer_secure_code_LP_MSPM0G3519_nortos_ticlang-bank1-0x40000.bin*: これは物理バンク 1 用の CSC ファームウェアのコピーであり、物理バンク 1 の開始アドレス(MSPM0G3519 デバイスでは 0x40000)に配置する必要があります。
- *\customer_secure_sample_image_LP_MSPM0G3519_nortos_ticlang\EITHER_SLOT_BLUE\sample_image_signed_0x4800_v1_0_0_blue.txt*: これは、物理バンク 1 0x44800 アドレスから始まるアプリケーション イメージです。

6. Uniflash を使用して、より高いバージョンの GREED アプリケーション イメージをデバイスに更新し、NRST ボタンを押します。赤色の LED が 2 秒間点灯した後、緑色の LED が点滅していることがわかります。CSC は、イメージがどのような方法でフラッシュに書き込まれたかは関知せず、この手順は、Uniflash を使って MCU のロジック バンク 1 にファームウェアを直接ロードする方法の一例を示しているだけです。

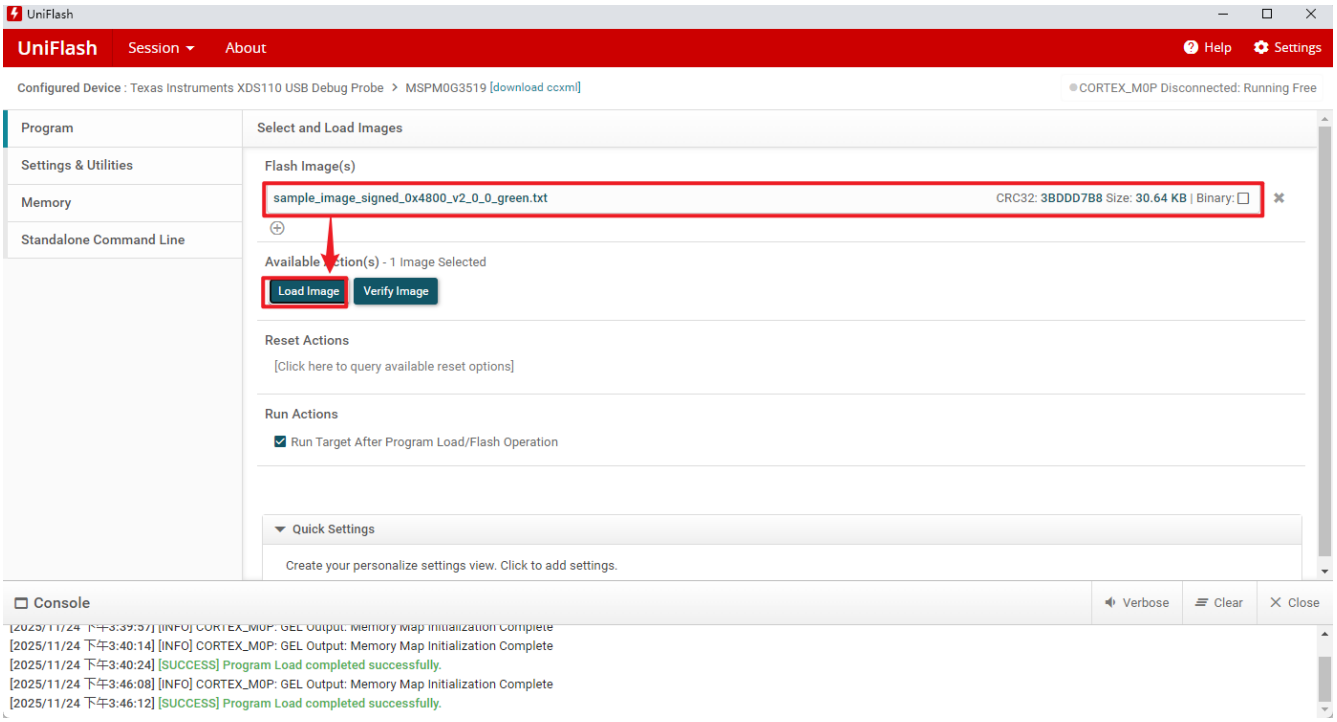

図 3-9. Uniflash でファームウェアを更新

3.2.4.3 CSC NONMAIN の設定

CSC プロセスを有効にするには、いくつかの NONMAIN 設定が必要です。CSC には、推奨される NONMAIN 構成がいくつかあります:

1. **デバッグ ポート保護**:MSPM0 デバッグ ポート (SWD) は、製造後に完全にディスエーブルにするか、パスワードによってイネーブルにするかを選択できます。MSPM0Gx51x、MSPM0Lx22x、MSPM0L111x などの CSC 対応デバイスでは、デバッグ ポートにアクセスするためのパスワードとして、256 ビットのハッシュ化されたパスワードが提供されます。詳細については、プラットフォーム セキュリティ イネーブラを参照してください。
2. **CSC 静的書き込み保護**:CSC 領域を変更不可能かつワンタイム プログラマブル (OTP) にするため、Bank0 と Bank1 の両方の CSC ファームウェアは、製造後に静的書き込み保護する必要があります。
3. **NONMAIN 静的書き込み保護**:NONMAIN には、静的書き込み保護とデバッグ アクセスに対するこれらの重要な構成がすべて含まれているため、アプリケーション プログラムによる消去または書き込み動作を防ぐために、NONMAIN のコンテンツ自体にも静的書き込み保護が必要です。
4. **パスワードを使用した工場出荷時リセット**:工場出荷時リセットでは、SWD にアクセスできるときにすべての NONMAIN 構成をデフォルト設定に回復し、MAIN フラッシュ内のすべての内容を消去できます。MSPM0 を工場出荷時リセットにしたくない場合は、パスワードを使用して工場出荷時リセットをイネーブルにできます。
5. **CSCEXISTS と FLASHBANKSWAPPOLICY**:CSCEXISTS は CSC ブート シーケンスを有効にし、FLASHBANKSWAPPOLICY はバンク スワップ ポリシーを有効にします。これらは、SDK CSC の例でイネーブルになっています。

注

CSC の開発段階では構成 5 を有効にする必要があり、構成 1～4 は、すべてのファームウェア開発が完了し、デバイスが量産段階に入るときにのみ有効にできます。

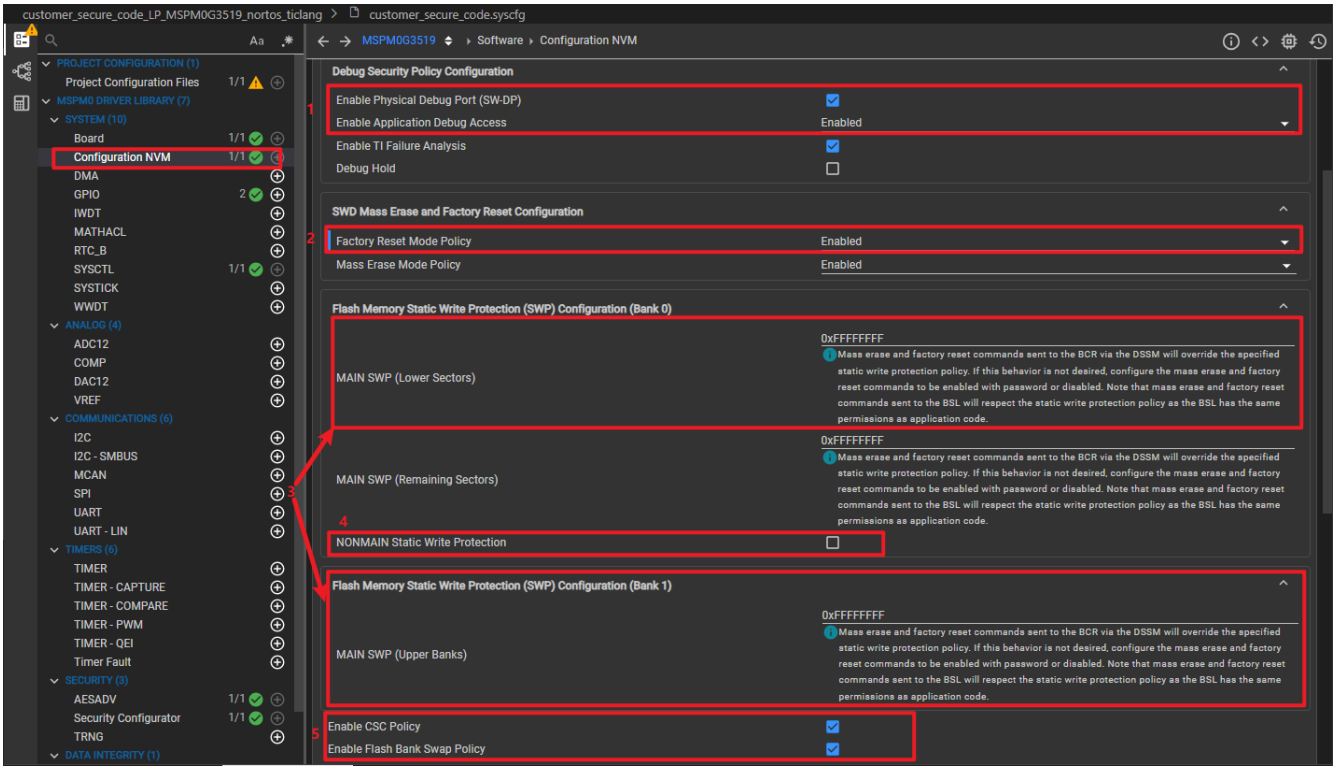

図 3-10. NONMAIN のセキュリティ構成

3.2.4.4 CSC の例で変更をカスタマイズ

アプリケーションの開始アドレスを変更

このセクションでは、MSP0 SDK の CSC サンプルで使用されているフラッシュ アドレス関連のいくつかのパラメータを紹介し、アプリケーションの開始アドレスを変更する方法をより理解しやすくすることを目的としています。

図 3-11 を参照して、図 3-12 の左側に表示されるパラメータは CSC サンプル Sysonfig で定義されています。これらのパラメータはすべて、*customer_secure_code* サンプルおよび *customer_secure_sample_image* のサンプル リンカ ファイル (.cmd) の対応するパラメータ定義と同じでなければなりません。

- CSC ロック ストレージ アドレス: このアドレスは、CSC コード サイズよりも大きく定義する必要があります。
- CSC ロック ストレージ サイズ: ロック ストレージ領域のサイズ。
- CSC シークレット アドレス: ロック ストレージ領域の直後にあるシークレット領域の開始アドレス。
- CSC シークレット サイズ: シークレット リージョン サイズ。
- CSC アプリケーション イメージ ベース アドレス:イメージ ヘッダが配置される開始アドレス。アプリケーションの割り込みベクタは、このアドレスから 0x100 バイト(イメージ ヘッダ サイズ)後に配置されます。
- CSC アプリケーション イメージ サイズ:元の未署名アプリケーション コードのサイズに、イメージ ヘッダ サイズとイメージ TLV サイズ (CSC サンプルでは約 160 バイト) を加えた合計よりも大きくする必要があります。これはイメージ トレーラが配置される位置を決定し、未使用の領域は 0xFF で埋められます。

注

CSC sysconfig で「Security Configurator」(セキュリティ コンフィギュレータ) がイネーブルになっていない場合、CSC アドレスとサイズ パラメータは、さまざまなデバイス ファミリの flash_mem_backend.c ファイルで定義されます。アプリケーション アドレスを変更するには、このソース ファイルを変更する必要があります。リンカ ファイルと signingArgs.json ファイルについても同じ変更が必要です。

図 3-12 の右側は、*customer_secure_sample_image* の *signingArgs.json* ファイルで定義されています。例:

- slotSize: CSC アプリケーション イメージ サイズと同じである必要があります。

- オフセット: CSC アプリケーション イメージ ベース アドレスと同じである必要があります。

アプリケーションの開始アドレス (またはシークレット アドレスやロック ストレージ アドレスなどの他のアドレス) を変更する場合、変更を有効にするために、*customer_secure_code* サンプル sysconfig ファイルとリンカ ファイル、*customer_secure_sample_image* サンプル リンカ ファイル、*signingArgs.json* ファイルを一緒に変更する必要があります。

図 3-11. Sysconfig CSC コンフィギュレータ

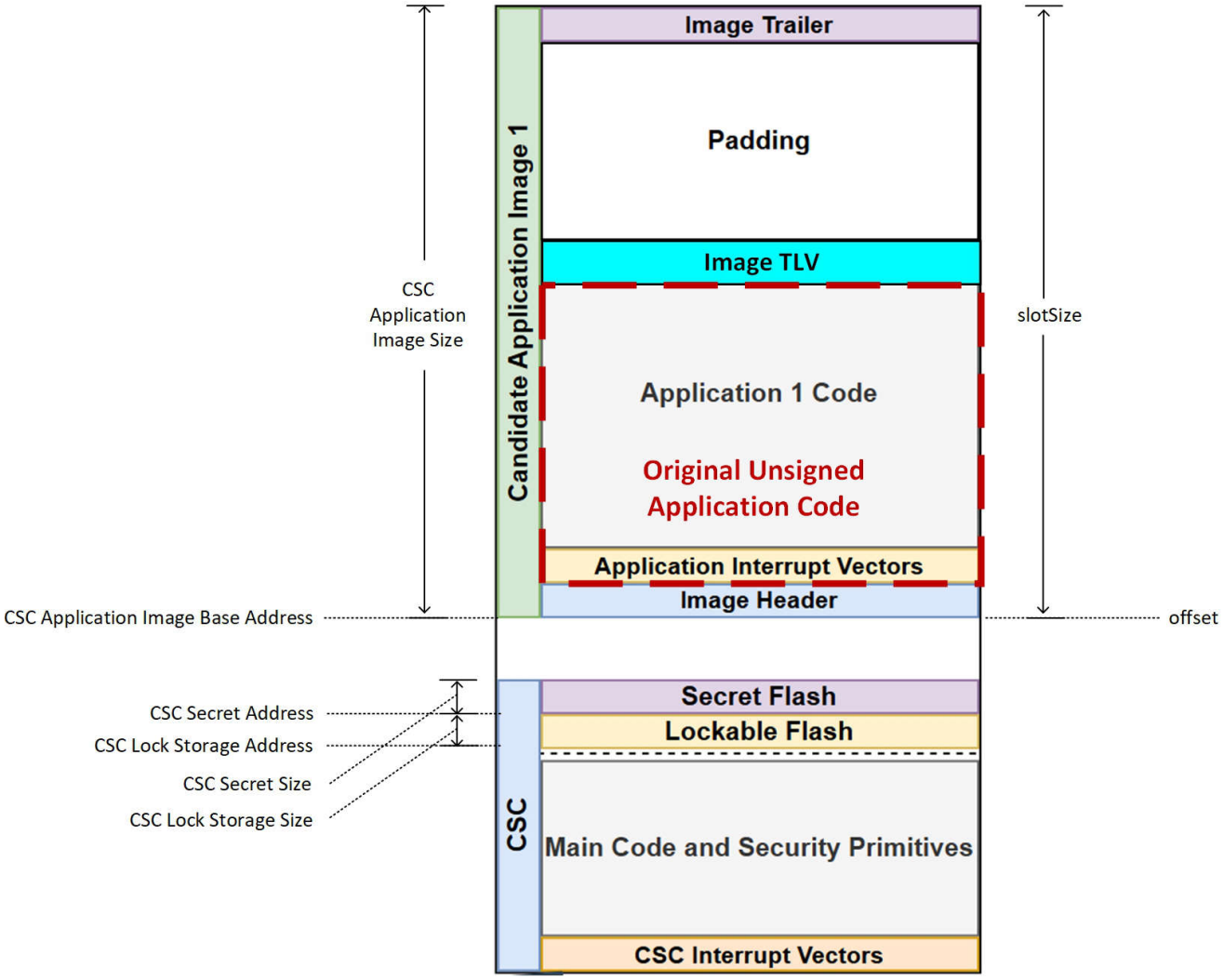

図 3-12. CSC フラッシュ マップ パラメータ

新しい **ECDSA** キーを生成し

新しい ECDSA キーの作成と、Python スクリプトを使用して新しいキーを使用したアプリケーション イメージの署名については、セキュア ブート ユーザーガイドの「MSPM0 カスタマ セキュア コードおよびブートローダー (CSC) ユーザー ガイド」の「顧客セキュア コードを使用した開発」セクションを参照してください。

3.3 ブート イメージ マネージャ (BIM)

ブート イメージ マネージャ (BIM) は、プロセスフローの側面から見たカスタマ セキュア コード (CSC) のサブセットです。このソリューションでは、特権状態と非特権状態を分離するハードウェア分離機構がないため、ブート フロー全体は CSC よりも簡素であり、ファイアウォール、KEYSTORE、バンク スワップ、CMAC といった機能は含まれていません。BIM と CSC の比較については、表 3-1 を参照してください。

3.3.1 セキュア ブートのフロー

BIM のセキュア ブート フローを、図 3-13 に示します。ソフトウェア ベースの非対称 SHA256 および ECDSA は、CSC と同じ機能および実行フローを維持しています。BIM でのセキュア ブートのために、次のプロビジョニング手順が実行されます:

1. ブート イメージ マネージャ ファームウェアは、MAIN フラッシュ メモリに構成およびプログラムする必要があります。リセット ベクトルは 0x0000.0004 で、ブート イメージ マネージャの開始を指しています。
2. ブート イメージ マネージャが必要とするすべての認証キー マテリアルは、ブート イメージ マネージャに隣接する MAIN フラッシュ メモリにプログラムする必要があります。
3. デバイスの NONMAIN 構成メモリは、次のようにプログラムする必要があります。
    a. ブート イメージ マネージャ ファームウェアと認証キー マテリアルを含む MAIN フラッシュ セクタは、変更を防止するため静的書き込み保護として構成する必要があります。
    b. NONMAIN フラッシュ セクタは、変更を防止するため静的書き込み保護として構成する必要があります。
    c. 一括消去および工場出荷時リセット コマンドは、パスワード保護またはディセーブルすることを推奨します。上記の構成設定で工場出荷時リセットをディセーブルすると、NONMAIN 構成がブート イメージ マネージャおよび認証キーを含むセクタと共に永続的にロックされます。
    d. MAIN フラッシュ メモリの整合性チェックはイネーブルにし、アドレス範囲をブート イメージ マネージャと認証キーを含めるように設定することを推奨します。

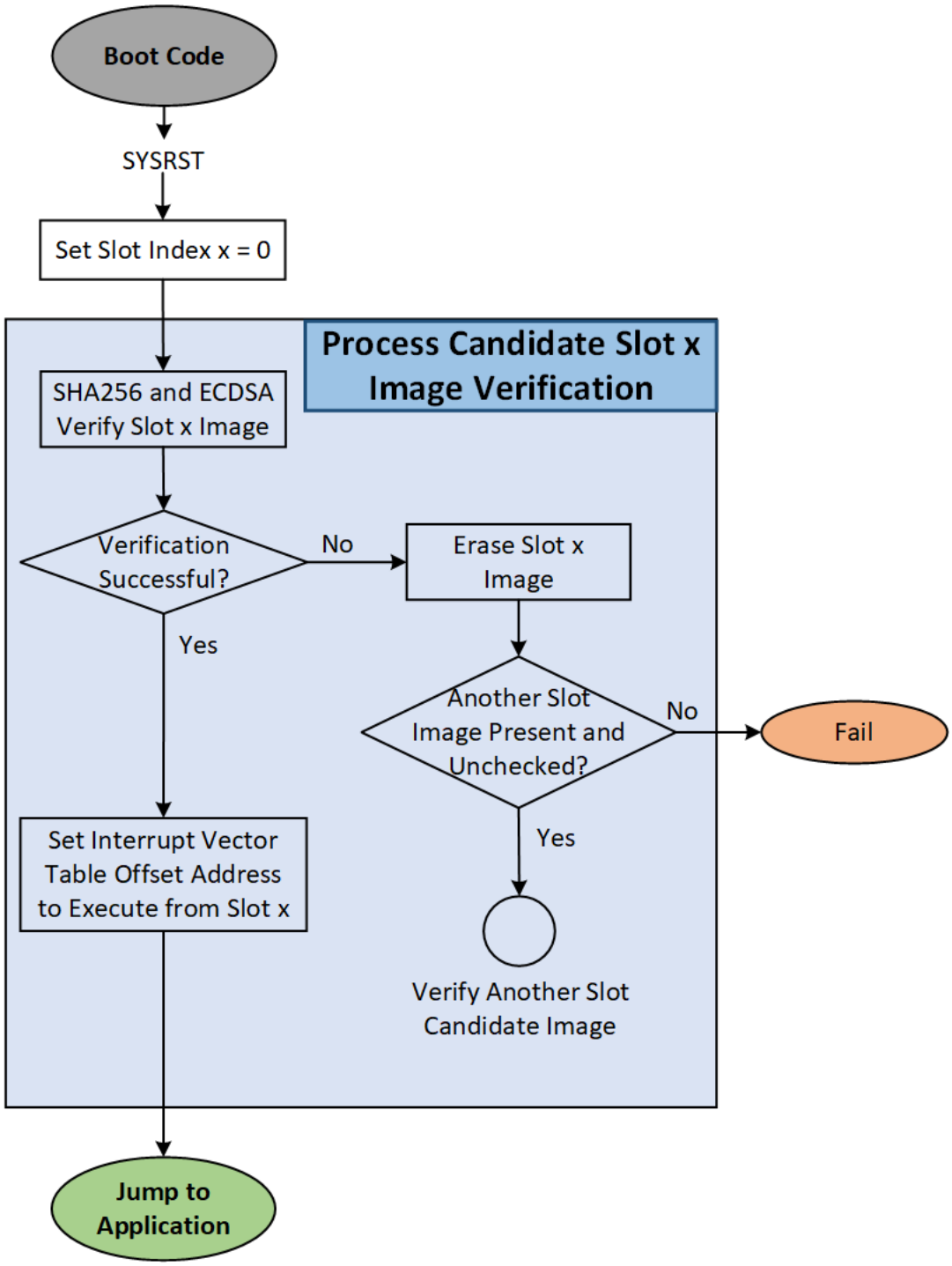

図 3-13. BIM 起動シーケンス

3.3.2 フラッシュ メモリ マップ

図 3-14 に、BIM のフラッシュ メモリ マップを示します。このソリューションにはバンク スワップ機能がないため、2 つのアプリケーション イメージは異なるフラッシュ アドレスにマップする必要があります。この点はプロジェクトのリンカ ファイルで示されています。

アプリケーション イメージのアドレスを変更する必要がある場合、対応するデバイス ファミリのメモリ マップ図に示されているアドレスおよびサイズのパラメータを再定義するために、flash_mem_backend.c ファイルを修正する必要があります。

各アプリケーション イメージ スロットには、MCUBOOT の署名ツール imgtool によって生成される Image Header、Image TLV、および Image Trailer が含まれています。詳細については、フラッシュ メモリ マップを参照してください。

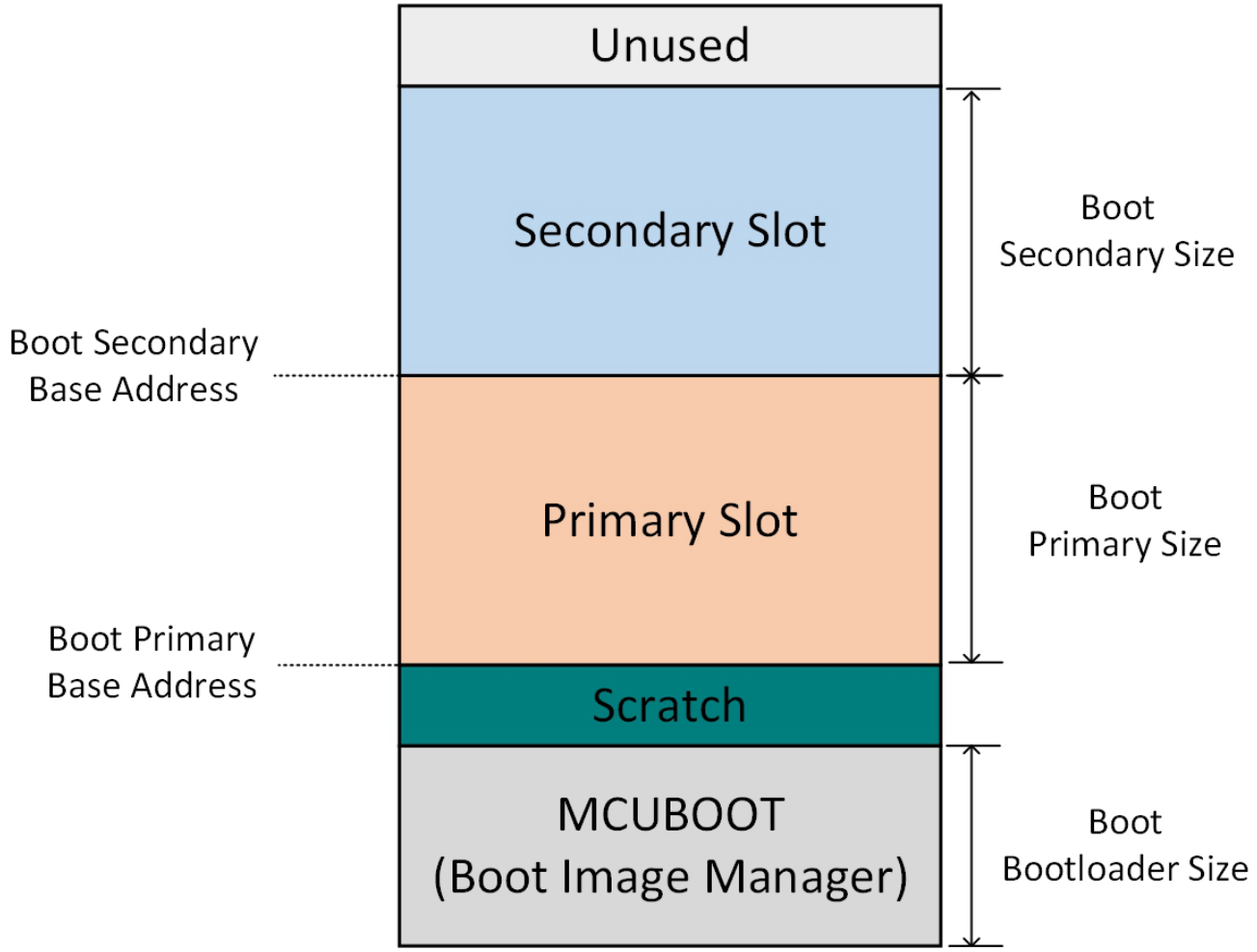

図 3-14. BIM フラッシュマップ

3.3.3 クイック スタート ガイド

ユーザーは、MSPM0 SDK のパス *<mspm0_sdk_path>\examples\nortos\<mspm0_device>\boot_manager* にある *bim_sample_image* の例および *boot_application* の例を参照することで、BIM ソリューションの評価を開始できます。また、セキュア ブート ユーザー ガイドに含まれる MSPM0 Boot Image Manager (BIM) ユーザー ガイドのガイダンスも参照してください。

イメージの署名には同じ Python 環境が必要であり、CSC のガイダンスを参照しつつ、BIM の例に示されているのと同様の手順に従って操作を行うことができます。詳細については、環境設定およびステップ バイ ステップ ガイダンスを参照してください。

4 セキュア ストレージ

MSPM0 は、フラッシュ、AES 秘密鍵、SRAM にさまざまな種類のメモリ保護メカニズムを搭載しており、多様なセキュリティ要件に対応できます。これらのメカニズムの構成の詳細については、MSPM0 G シリーズ 80MHz マイコ ンテクニカル リファレンス マニュアル (Rev. C) のセキュリティの章を参照し、これらの機能については MSPM0 デバイスシリーズのプラットフォーム セキュリティ イネーブラを参照してください。

4.1 フラッシュ書き込み保護

書き込み保護には、次の 3 つのレベルがあります:

1. TI のブートコードによって強制される静的書き込み保護。これは NONMAIN BCR 領域で設定され、ブート コード実行が完了した後に有効になります。CSC のブート シーケンスについては、図 3-1 を参照してください。この機能は、CSC を信頼の起点の拡張として扱い、かつ不変にする必要がある場合に有用です。
2. CSC によって強制される書き込み保護であり、更新を許可されていますが、アプリケーションからは変更されるべきでないデータをさらに保護するためのものです。この保護は CSC で設定され、INITDONE 後に有効になります。CSC によって追加の書き込み保護を持つように構成できるのは、フラッシュ メモリの最初の 32KB のみであることに注意してください。バンク スワップ可能な構成では、保護は自動的に両方のバンクにミラーリングされます。
3. バンク スワップ状況での書き込み保護。また、CSC で設定され、INITDONE 後に有効になります。バンク スワップがイネーブルの場合、ロジック下位のバンクは読み取り実行権限を取得し、書き込み / 消去権限を失います。もう一方のバンク (論理上位バンク) は読み取り可能で、書き込み可能ですが、実行可能ではありません。

4.2 フラッシュ読み取り実行保護

フラッシュ メモリの一部領域は読み取り、実行保護として設定でき、この領域への読み出しアクセスおよび命令フェッチ アクセスはいずれもエラーを返します。CPU、DMA、デバッガ アクセスはすべて同じように扱われます。

この仕組みは、CSC の再実行を防止したい場合や、アプリケーション プログラムから機密情報が読み出されるのを防ぐ必要がある場合に有用です。

バンク スワップ可能な構成では、保護は自動的に両方のバンクにミラーリングされます。

4.3 フラッシュ IP 保護

フラッシュ メモリの一部領域は読み取り保護として設定でき、この領域への読み取りアクセスはエラーを返しますが、命令フェッチによるアクセスは許可されます。CPU、DMA、デバッガ アクセスはすべて同じように扱われます。

この仕組みは、サードパーティ ベンダが提供するソフトウェア IP のコード読み取りを防止する必要がある場合に有用です。IP 保護を目的としたコードは、この領域に対する埋め込みデータ アクセス (リテラル フェッチ) が発生しないようにコンパイルされている必要がある点に注意してください。

バンク スワップ可能な構成では、保護は自動的に両方のバンクにミラーリングされます。

4.4 データ バンクの保護

フラッシュの DATA バンクの一部領域は読み書き保護として設定でき、読み取り、書き込み、またはその両方のアクセスのタイプをブロックできます。CPU、DMA、デバッガ アクセスはすべて同じように扱われます。

DATA バンクのうち、保護できるのは先頭の 4 KB のみで、セクタ (1 KB) 単位での保護となります。各セクタには次のものがあります:

- 読み取り保護
- 書き込み保護
- 両方
- どちらでもない

このメカニズムにより、CSC はデータ バンク内に秘密または機密データを保存時、それを単独で読み取り/ 変更できます。

スタートアップ時。

4.5 セキュアなキー ストレージ

秘密鍵 (非対称スキームの秘密鍵) は、機密性を確保するために保護する必要があります。信頼されたコードのみが、フラッシュ メモリから鍵を安全に格納し、暗号エンジンのみがアクセス可能な場所へ転送するための仕組みを備えるべきです (具体的には AES アクセラレータを指します)。

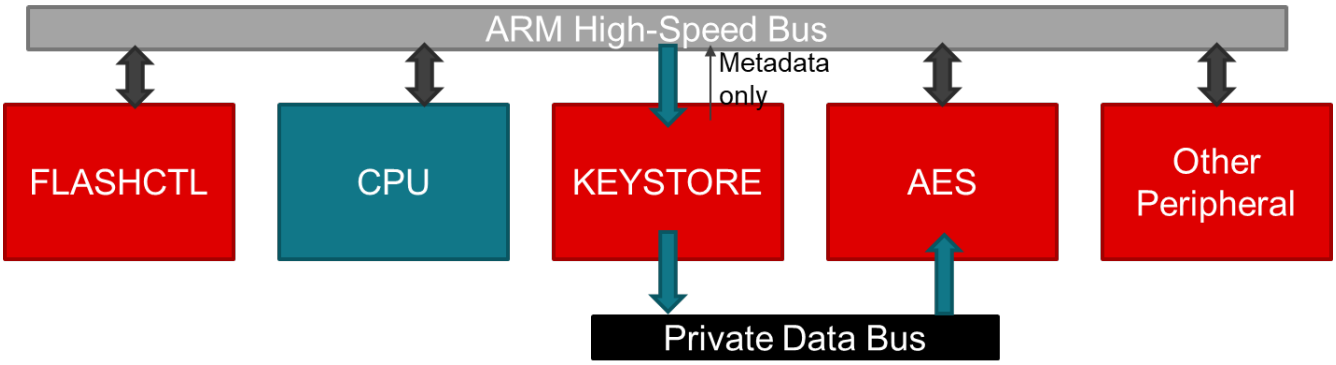

図 4-1. KEYSTORE の動作プロセス

INITDONE 以前に KEYSTORE に鍵を設定できるのは CSC のみです。その後、メイン アプリケーションは、保存されているキーのいずれかを使用するように暗号化エンジンを設定できますが、保存されているキーにはアクセス (読み取りまたは書き込み) できません。KEYSTORE から暗号化エンジンへのキー転送は安全に実行され、アプリケーション コードからは見えません。

KEYSTORE の内容は BOOTRST ごとに消去され、ストアは書き込み可能な状態でアンロックされます。KEYSTORE の内容は SYSRST およびそれより下位のリセットの影響を受けません。

4.6 SRAM 保護

バッファ オーバーフローは一般的な攻撃手法の原因であり、例えば復帰アドレスが破壊されると、実行が悪意のあるコードへジャンプしてしまう可能性があります。このような悪用を緩和するために、SRAM コード保護機能が利用できます。これにより、SRAM はアドレス A によって次の 2 つの領域に分割できます。

- 領域 1 (アドレス < A): 読み取り / 書き込み (RW)、命令フェッチの実行ではありません。
- 領域 2 (アドレス >= A): 読み取り実行 (RX)、書き込み不可。

A = 0 は SRAM 全体が RWX として扱われます。これは SRAM のリセット状態です。

4.7 ハードウェア単調カウンタ

一部のデバイスには、次の制限の下でアクセスされる 2 つ目の NONMAIN セクタ (1KB) が用意されています:

- このセクタは決して消去できません (工場出荷時リセットや一括消去操作を含みます)
- このセクタは CSC が INITDONE を呼び出すまでのみ書き込み可能で、その後は読み取り専用になります。

この保護機能は、ハードウェア ベースの単調カウンタ機能を実装します。このセクタは追記のみでプログラムされるため、実質的には不揮発性のアップ カウンタ (ソフトウェアでの解釈によってはダウン カウンタ) として動作します。CSC はこのセクタにリビジョン情報やロールバック保護情報を保持でき、その結果、カウンタ値より低いバージョンは有効化できないことが保証されます。ハードウェア単調カウンタ機能をサポートするデバイスについては、プラットフォーム セキュリティ イネーブラを参照してください。

5 暗号化アクセラレーション機能

一部の MSPM0 MCU には、AES (Advanced Encryption Standard) のハードウェア アクセラレーション機能と、暗号化目的に真の乱数を生成するためのハードウェア (TRNG) が搭載されています。デバイスに AES アクセラレータや TRNG が搭載されているかどうかは、各デバイス固有のデータシートを参照するか、*サブファミリ別セキュリティ有効化機能*を参照してください。

5.1 ハードウェア AES アクセラレーション

一部の MSPM0 デバイスには、AES (Advanced Encryption Standard) のハードウェア アクセラレーション機能が備わっています。AES アクセラレータには MSPM0 デバイスで定義されているという 2 種類があり、セクション 5.1.1 とセクション 5.1.2、異なる機能セットをサポートしています。AES と AESADV の比較については、表 5-1 を参照してください。

デバイスに AES アクセラレータが搭載されているかどうかを確認するには、該当するデバイスのデータシートを参照してください。

表 5-1. AES と AESADV の比較表

特長	基本的な AES (AES)	高度な AES (AESADV)
ECB	✓	✓
CBC	✓	✓
OFB	✓	✓
CFB	✓	✓
CTR	✓	✓
CBC-MAC	✓	✓
CMAC		✓
AES-CCM		✓
AES-GCM		✓

5.1.1 AES

AES アクセラレータ モジュールは、AES (Advanced Encryption Standard) に従って 128 ビットまたは 256 ビットのキーをハードウェアに配置し、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
- NIST SP 800-38 で定義されている ECB、CBC、OFB、CFB ブロック暗号モードを自動化するための DMA トリガのサポート
- 事前に計算された (nonce || counter) ブロックの暗号化による CTR 暗号モードおよび生成されたキー ストリームを使用するテキストの XOR の高速化をサポート
- CBC-MAC タグ計算の高速化 (ゼロ初期化ベクトルを使用した CBC DMA モード) をサポート
- 暗号化と復号化のためのオンザフライ方式キー拡張機能
- 復号用のオフライン キー生成
- シャドウ レジスタにすべてのキー長の初期キーを格納
- 8 ビット バイトまたは 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
- 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)

AES アクセラレータ ハードウェアは、128 ビットのステート メモリと関連の入出力レジスタ、AES 暗号化 / 復号化コアと制御ロジック、256 ビットの AES キー メモリと関連の入力レジスタで構成されています。AES アクセラレータは、128 ビット ブロックの高速暗号化と復号化を実現します。ブロック暗号化およびブロック復号 (事前生成された復号キーを使用) における AES アクセラレータの性能は、サイクル数および実行時間の両面から表 5-2 に示されています。

表 5-2. AES ハードウェア アクセラレータの主な性能指標

AES キー長	暗号化			復号化		
	サイクル	時間 (32MHz)	時間 (80MHz)	サイクル	時間 (32MHz)	時間 (80MHz)
128 ビット	168	5.25us	2.10us	168	5.25us	2.10us
256 ビット	234	7.31us	2.93us	234	7.31us	2.93us

5.1.2 AESADV

AESADV アクセラレータ モジュールは、FIPS PUB 197 の高度暗号化標準 (AES) に基づき、暗号化および復号処理をハードウェアで高速化します。

AESADV アクセラレータ モジュールは、AES (Advanced Encryption Standard) に従って 128 ビットまたは 256 ビットのキーをハードウェアに配置し、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。AESADV 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
- AESADV 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)

AESADV 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。

AESADV アクセラレータは、128 ビット ブロックの高速暗号化と復号化を実現します。ブロック暗号化およびブロック復号 (事前生成された復号キーを使用) における AESADV アクセラレータの性能は、サイクル数および実行時間の両面から表 5-3 に示されています。この表は、次の入力を供給する際や利用可能な出力を読み出す際の遅延といった、エンジンを停止させるシステム オーバーヘッドが存在しないことを前提としています。

表 5-3. AESADV ハードウェア アクセラレータの主な性能指標

AES キー長	暗号化			復号化		
	サイクル	時間 (32MHz)	時間 (80MHz)	サイクル	時間 (32MHz)	時間 (80MHz)
128 ビット	76	2.38us	0.95us	1.01us	2.38us	0.95us
256 ビット	81	2.53us	1.01us	81	2.53us	1.01us

5.2 ハードウェア真性乱数生成器 (TRNG)

一部の MSPM0 デバイスには、ハードウェア真性乱数生成器 (TRNG) ブロックが搭載されています。TRNG を使用すると、真の乱数シード値を簡単に生成でき、生成された値を決定論的乱数生成器 (DRBG) のシードとして使用できます。

TRNG モジュールは、デバイス内のデルタ-シグマ変調ベースのアナログ エントロピー ソースに基づいて、32 ビットの真性乱数出力を生成します。TRNG には、電力操作攻撃から保護するため、専用レギュレータが付属しています。

内蔵された診断テストにより TRNG のアナログおよびデジタル コンポーネントのパワーオン セルフ テストを実行でき、統計的なセルフ テストにより連続的な監視が可能になります。

TRNG は、暗号化乱数生成器用の NIST SP800-22 統計テスト スイートに合格できる TRNG + DRBG システムの作成に適しています。図 5-1 に、TRNG のブロック図を示します。

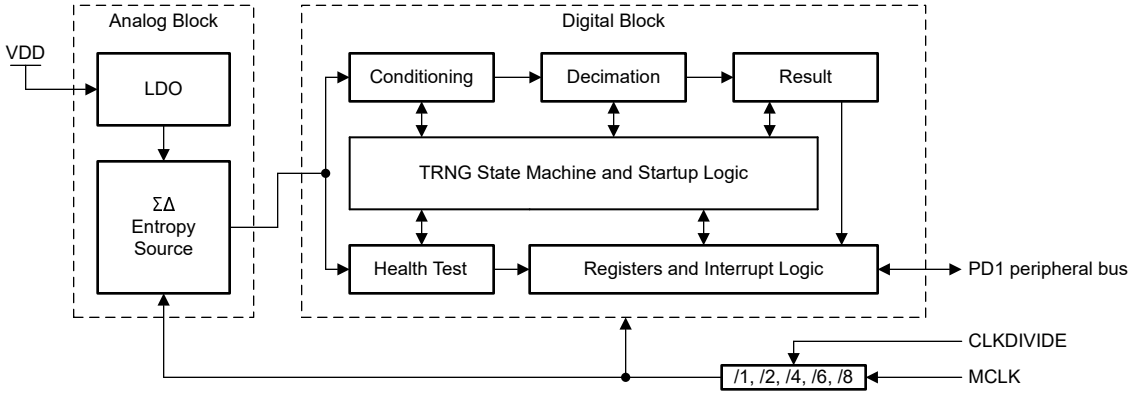

図 5-1. TRNG のブロック図

TRNG の動作の詳細については、デバイス ファミリのテクニカル リファレンス マニュアルを参照してください。

6 FAQ (よくある質問)

1. **MSPM0 CSC ソリューションにおける信頼の根幹とは何ですか？**

   A: 信頼の根幹には，不変の TI ROM ブート コードと静的に書き込み保護された CSC 領域が含まれます。正しく NONMAIN 構成後は、変更できません。詳細については、CSC NONMAIN の構成を参照してください。
2. **CSC はファームウェア更新プロセスを扱いますか？**

   A: いいえ。CSC は，あらかじめ特定のフラッシュ アドレスに配置されたアプリケーション ファームウェアを検証するだけであり，ファームウェア更新プロセス (ブートローダー) を扱ったり，ファームウェアがどのようにフラッシュにロードされるかには関与しません。
3. **CSC ソリューションのさまざまなアルゴリズムのタイミング機能は何ですか？**

   A: 詳細については、CSC 性能を参照してください。
4. **CSC またはバンク スワップ機能を搭載してアプリケーションを実行しているデバイスに新しいファームウェアをダウンロードしようとすると、CCS/Uniflash に消去エラーが報告されるのはなぜですか？**

   A: バンク スワップがイネーブルの場合、ロジック low バンクは読み取り実行権限を取得し、書き込み / 消去権限を失います。もう一方のバンク (ロジック high バンク) は読み取り可能で書き込み可能ですが、実行可能ではありません。CCS または Uniflash がアドレス 0x0000 からファームウェアのダウンロードを開始しようとすると、ロジック low バンク アドレスが消去できないため、消去エラーが報告されます。high バンク アドレスからファームウェアを更新するか、プログラムをロードする前に工場出荷時リセットを行うことができます。
5. **CSC サンプルをダウンロードした後に電源サイクルまたは NRST リセットが必要な理由は何ですか？**

   A: CSC のサンプルには、CSC を有効にするための NONMAIN 構成が含まれています。NONMAIN 設定はブート コード実行時に有効となり，MSPM0 が ROM ブートコードに戻るには，BOOTRST (またはそれ以上のレベルのリセット) が必要となります。
6. **CSC でアプリケーション プログラムの開始アドレスを変更するにはどうすればよいですか？**

   A: 詳細については、CSC サンプルでのカスタマイズ変更を参照してください。
7. **CSC、アプリケーション イメージ、および NONMAIN 領域の出力には、どの出力形式を選択すればよいですか？**

   A: .txt/.bin/.hex 形式をファームウェアの更新に使用できます。NONMAIN 構成は CSC と一緒にプログラムする必要があり、アプリケーション ファームウェアと一緒に NONMAIN 領域を更新しないでください。ガイダンスについては、ステップ バイ ステップ ガイダンスを参照してください。
8. **customer_secure_sample_image のビルド時に CCS がビルド後の失敗エラーを報告するのはなぜですか？**

   A: CSC サンプル イメージ サンプルをビルドする前に、Python 環境が正常に設定されているかどうかを確認してください。CSC サンプルが CSC イメージ サンプルと同じワークスペース内にあることを確認します。
9. **アプリケーション プログラムで非対称暗号化 / 復号化のための秘密鍵ストレージ領域はありますか？**

   A: MSPM0 デバイスは、AES エンジンに対して対称型秘密キー ストレージ (KEYSTORE) のみを提供します。非対称暗号化/復号アルゴリズムの鍵 (例えば ECDSA 公開鍵) は SECRET 領域に格納されます。このシークレット リージョンは、特権状態 (INITDONE より前) でのみアクセスでき、アプリケーション実行時にファイアウォールによって読み取り保護および書き込み保護されます。
10. **リンカ ファイルでアプリケーション アドレスを指定するにはどうすればよいですか？**

    A: バンクスワップが有効な構成では，アプリケーション プログラムは常に指定された論理 low バンクのアドレスでビルドされる必要があります。MSPM0G3519 を例にして、リンカ ファイル (CCS では .cmd) で定義されているアドレス範囲は 0x00000 ～ 0x40000 である必要があります。しかし，アプリケーション ファームウェアを更新する際は，プログラムがロジック low バンクで実行中であり，読み書き可能なのはロジック high バンクだけであるため，ファームウェアは論理 high バンクのアドレスにロードする必要があります。
11. **独自のアプリケーション イメージ形式を定義する場合はどうすればよいですか？**

    A: 現在、SDK CSC サンプルは、MCUBOOT が提供する署名ツール imgtool を使用して、ヘッダと署名情報などのアプリケーション イメージを生成しています。ユーザは独自の画像フォーマットを定義することができますが、独自に定義された画像フォーマットのために CSC で解析プログラムを達成する必要があります。
12. **セキュア ブートにのみ対称型のアプローチを使用する場合はどうすればよいですか？**

A: AESADV をサポートしている MSPM0 デバイスでの対称的画像検証に、AES-CMAC を使用できます。詳細については、プラットフォーム セキュリティ イネーブラを参照してください。MCU に保存されている AES キーが、事前に画像ベンダと整合していることを確認してください。

7 まとめ

MSPM0 MCU が提供するセキュリティ・イネーブラは、新しいアプリケーションにサイバー・セキュリティ機能を追加する場合に、機能と価値を提供します。魅力的な価格帯で独自の機能 (パスワード認証を使用したアプリケーションのデバッグ、一括消去、工場出荷時リセットなど) が搭載されているため、構成をシンプルかつ明確なものに維持しながら、さまざまな開発と製造現場で使用できます。

8 参考資料

- 『MSPM0 G シリーズ 80MHz マイクロコントローラ テクニカル リファレンス マニュアル』
- 『MSPM0 L シリーズ 32 MHz マイコン テクニカルリファレンス マニュアル (改訂版 E)』
- 『MSPM0 C シリーズ 24 MHz マイコン テクニカルリファレンス マニュアル (改訂版 C)』
- 『MSPM0 H シリーズ 32MHz マイクロコントローラ テクニカル リファレンス マニュアル』
- MSPM0 ファミリのフラッシュ マルチ バンク機能
- MSPM0 カスタマ セキュア コードおよびブートローダー (CSC) ユーザー ガイド
- GitHub - mcu-tools/mcuboot: 32 ビット マイコン向けのセキュア ブート

9 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。

Changes from JANUARY 31, 2023 to DECEMBER 31, 2025 (from Revision * (January 2023) to Revision A (December 2025))	Page
• MSPM0 マイコンプラットフォームのセキュリティ イネーブラ表を更新。	3
• MSPM0 のセキュア ブートフローと実装を更新。	20
• MSPM0 のセキュア ストレージ機能を更新。	36

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。

最終更新日：2025 年 10 月