ルネサス RA ファミリ

RA8 ファーストステージブートローダを使ったアプリケーションの開発

はじめに

以下に示す Renesas RA8 シリーズ MCU は、「First Stage Bootloader(FSBL)」を ROM にプログラムしており、マスクされています。リセット後に実行され、チップ内のフラッシュメモリにプログラムされた OEM アプリケーションプログラムをシングルチップモードで検証できます。フラッシュメモリの不揮発性により、FSBL はシリコンベースで Root of Trust(RoT)を提供します。

FSBL を使用するには、OEM アプリケーションプログラムが製造時のプログラミング中に検証される必要があります。MCU のファクトリブートアプリケーションプログラムがこの機能を提供します。

このアプリケーションプロジェクトは、ファクトリブートアプリケーションプログラムおよび FSBL が OEM アプリケーションプログラムを検証するために使用する資格情報の設定方法を説明します。その後、アプリケーションプロジェクトは、Cyclic Redundancy Check 32(CRC32)整合性検証または Elliptic Curve Cryptography(ECC)NIST P256 署名認証と HMAC-SHA256 認証のいずれかの技術を使用してアプリケーションプログラムの検証を実演します。

必要なリソース

対象デバイス

以下は、本資料の情報が適用されるルネサス MCU 製品です。 :

- RA8M1
- RA8D1
- RA8T1

ソフトウェアと開発ツール

- e² studio IDE v2023-10
- ルネサスフレキシブルソフトウェアパッケージ(FSP)v5.1.0
  ダウンロードはこちらから https://github.com/renesas/fsp.
-
- SEGGER J-Link® USB driver v7.92o 以降(SEGGER J-Link)
- Renesas Secure Key Management Tool v1.05 以降
- Renesas Flash Programmer v3.13 以降
- GNU Privacy Guard for Windows: Gpg4win
- Renesas Device Lifecycle Management Server
-

ハードウェア

- EK-RA8M1、RA8M1 MCU グループ用評価キット(renesas.com/ra/ek-ra8m1)
- Windows® 10 と Tera Term コンソールまたは同様のアプリケーションを実行可能な PC
- USB デバイスケーブル(タイプ A オス-マイクロ B オス)1 本

前提条件と対象読者

このアプリケーションプロジェクトをお使いいただくには、ルネサス e² studio の使用経験があることを前提としています。また、アプリケーションブートのローディングや暗号化アルゴリズムに関する知識があることが望ましいです。このアプリケーションプロジェクトを実行する前に、ユーザは[ソフトウェアと開発ツール]に記載されているすべてのツールをインストールし、RA8M1 ユーザーズマニュアル ハードウェア編の以下セクションをお読みいただく必要があります。これにより、このアプリケーションプロジェクトのいくつかの説明の背景にある情報が得られ、ユーザがより深く掘り下げたり、学習を拡張するための便利さも提供されます。

- セクション:セキュリティ機能
- セクション:オプション設定メモリ
- セクション:フラッシュメモリ

さらに、本デモンストレーションには非 TrustZone®アプリケーションが使用されています。もし、FSBL を TrustZone®プロジェクトで使用したい場合は、アプリケーションノート Injecting and Updating Secure User Keys (R11AN0496)のセクション 4.1 を参照してください。

目次

1. ファーストステージブートローダの紹介

1.1 概要

セキュアシステムでは、アプリケーションプログラムが実行される前に、プログラムが意図的、偶発的に変更されていないことを確認する必要があります。この確認は通常、ブートローダによって行なわれ、簡単な整合性チェックや、信頼性を保証するための署名検証を含む場合があります。ただし、ブートローダ自体の正当性も保証されなければならないことに注意が必要です。

FSBL が有効になっている場合、ユーザの製品アプリケーションプログラムを検証する 2 つのオプションがあります。

- CRC32 を使用した整合性チェックをします。MCU の初期プログラミング時と実行前に行われます。
- 信頼性のチェックをします。MCU へのプログラミングの初期時、アプリケーションは secp256r1(NIST P256)ECC 曲線を使用して ECDSA で認証されます。また、実行前には MCU のハードウェアユニークキー(以下 HUK)で HMAC-SHA256 を使用して認証されます。

セキュアブートシーケンスは強力な Root of Trust を確保するために、不変のエンティティから始めて段階的に実装されます。RA8 MCU には不変の FSBL が書き込まれていて、セキュアブートシーケンスのスタートとして Root of Trust を提供します。最初に FSBL は OEM のブートローダを検証します。その後 OEM アプリケーションプログラムを検証して、システムを起動するため Chain of Trust(認証パスについては図 1 case 2 を参照)を形成します。したがって、FSBL を使用すると、OEM ブートローダは安全に更新できます。

以下は、MCU シングルチップモードでの FSBL を使用した概要を示したものです。

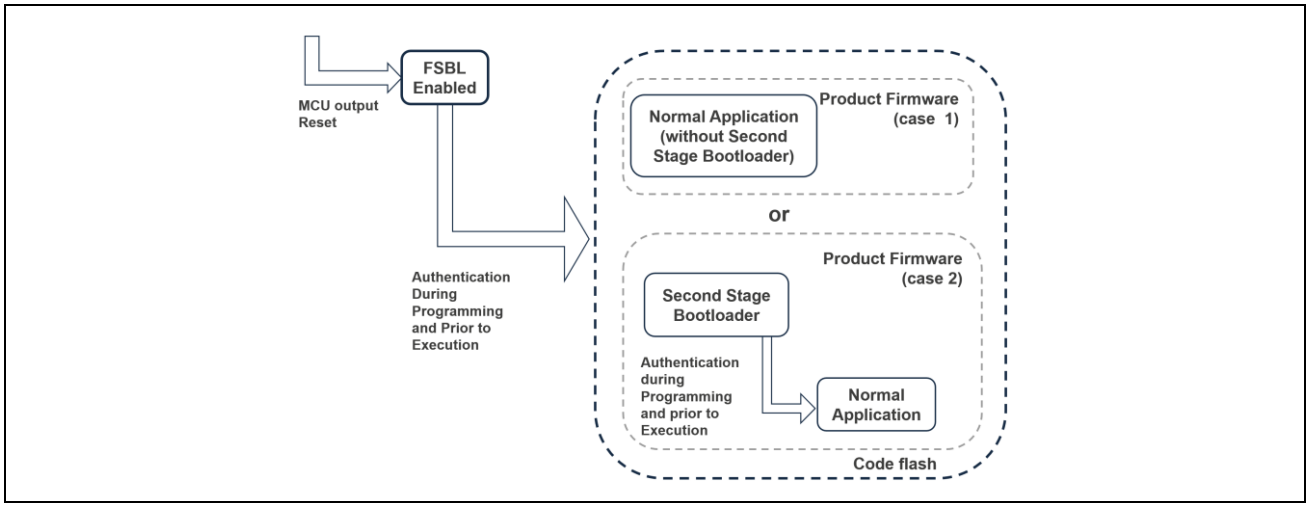


図 1. シングルチップモードでの FSBL の使用方法

RA8 MCU ユーザーズマニュアル ハードウェア編では、セカンドステージブートローダーが存在する case2 に関する説明に焦点を当てています。また、FSBL(ファーストステージブートローダ)に関連する操作についてファーストステージ、セカンダリステージ両方で「OEM_BL」と呼んでいますが、アプリケーションプロジェクトのリリースでは、最初のケース(FSBL が存在しない場合)のみが対象です。ユーザーズマニュアル ハードウェア編と一致させるため、「OEM_BL」は OEM 製品のアプリケーションプログラムを指す用語として使用します。

以下の図のように、FSBL のレジスタは**データフラッシュオプション設定メモリ領域**にマッピングされます。レジスタのプログラミングは、シリアルプログラミング(JTAG または SWD)を使用して IDE(e^2 studio など)で可能です。RFP(ルネサスフラッシュプログラマ)を使用してアプリケーションをプログラムする場合、ブートアクセスモードを介して行われます。これにより、FBSL レジスタで Setting された内容によって、リセット後の MCU の動作状態と、OEM_BL に直接ジャンプするかが決まります。

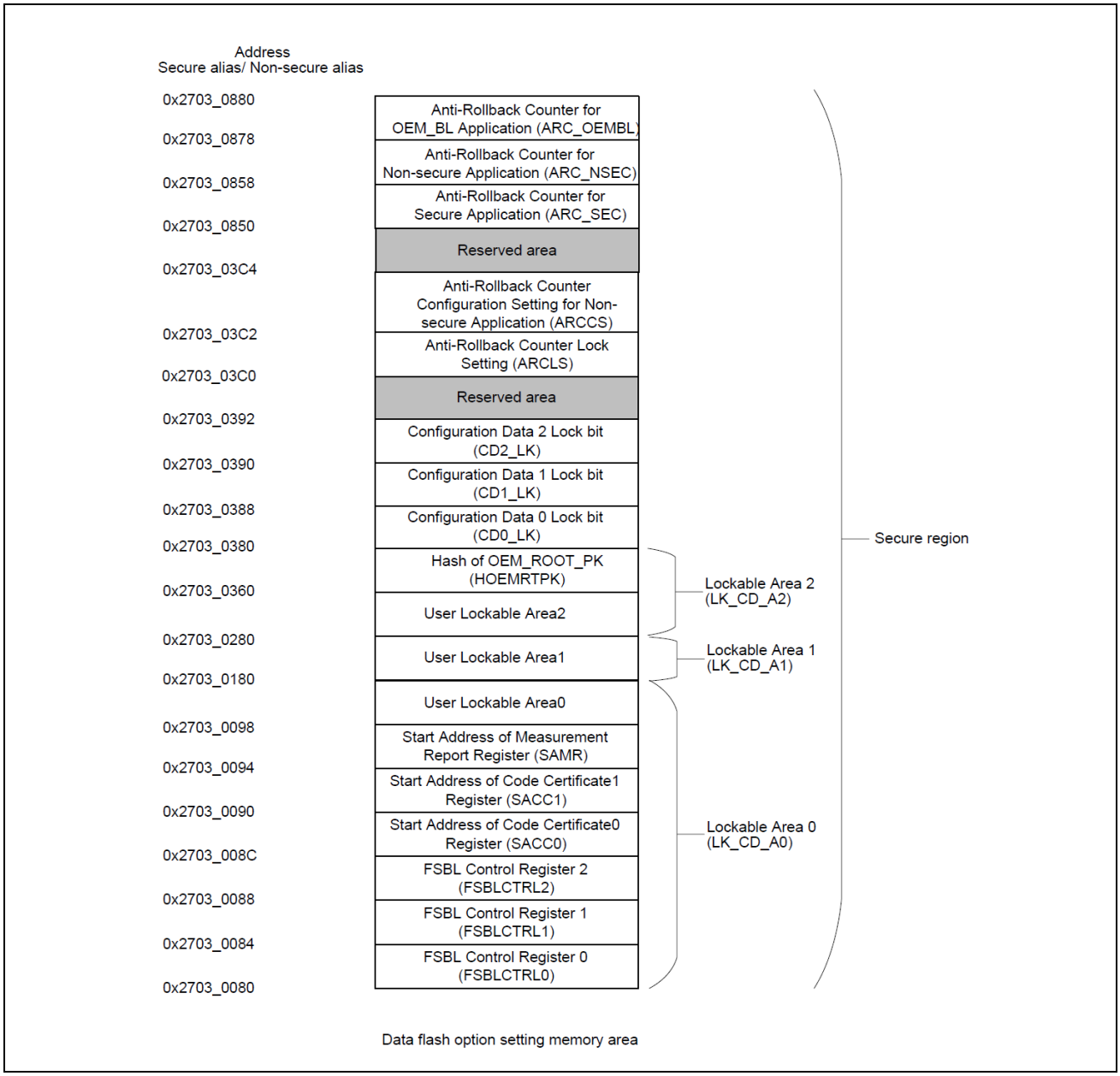


図 2. データフラッシュオプション設定メモリ領域　RA8x1 の場合

「ユーザーロック可能領域は、製造時やその他の製造プログラム中に Renesas Flash Programming ツールを使用してロックすることができます。
但し、誤ってこれらの領域をロックしたら、ロックを解除することはできません。このため、アプリケーション開発中にこれらのレジスタをロックすることは推奨しません。

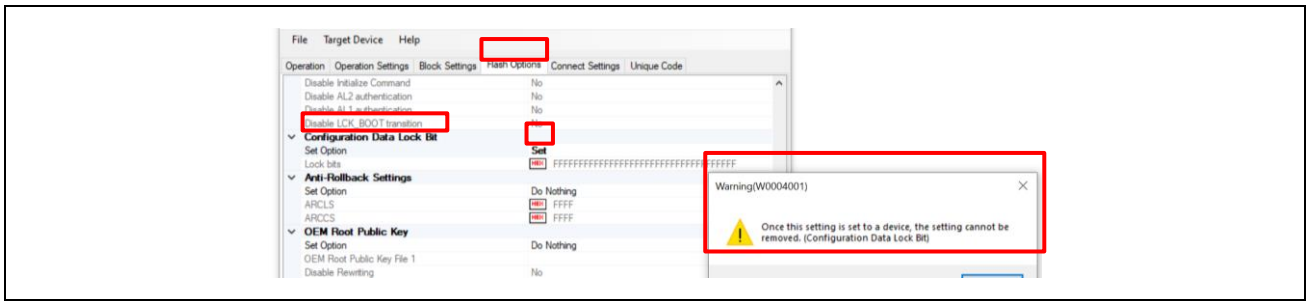


図 3. RFP を使用したデータフラッシュオプション設定レジスタのロック方法

1.2 FSBL ベースの起動オプション

FSBL が有効になっている場合、OEM_BL の検証には 2 つのスキームがサポートされています。

- CRC32 **ブート**:アプリケーションの整合性は CRC32 を使用して検証されます。
- **セキュアブート**:アプリケーションの信頼性は secp256r1 ECC 曲線(NIST P256)を使用してプログラミング時に検証されます。MCU の HUK で HMAC-SHA256 を使用して実行時に検証されます。

FSBL の Setting オプションは、スマートコンフィギュレータの **BSP** タブから選択できます。以下のように FSBL の Setting は、**Secure boot with report measurement** を選択した例を示しています。開発するアプリケーションのセキュリティの目的に基づいて、これらの Setting を調整してください。さまざまなスタックプロパティの詳細については、GitHub の RA Flexible Software Package Documentation > API Reference > BSP > MCU Board Support Package の Build Time Configurations を参照してください。

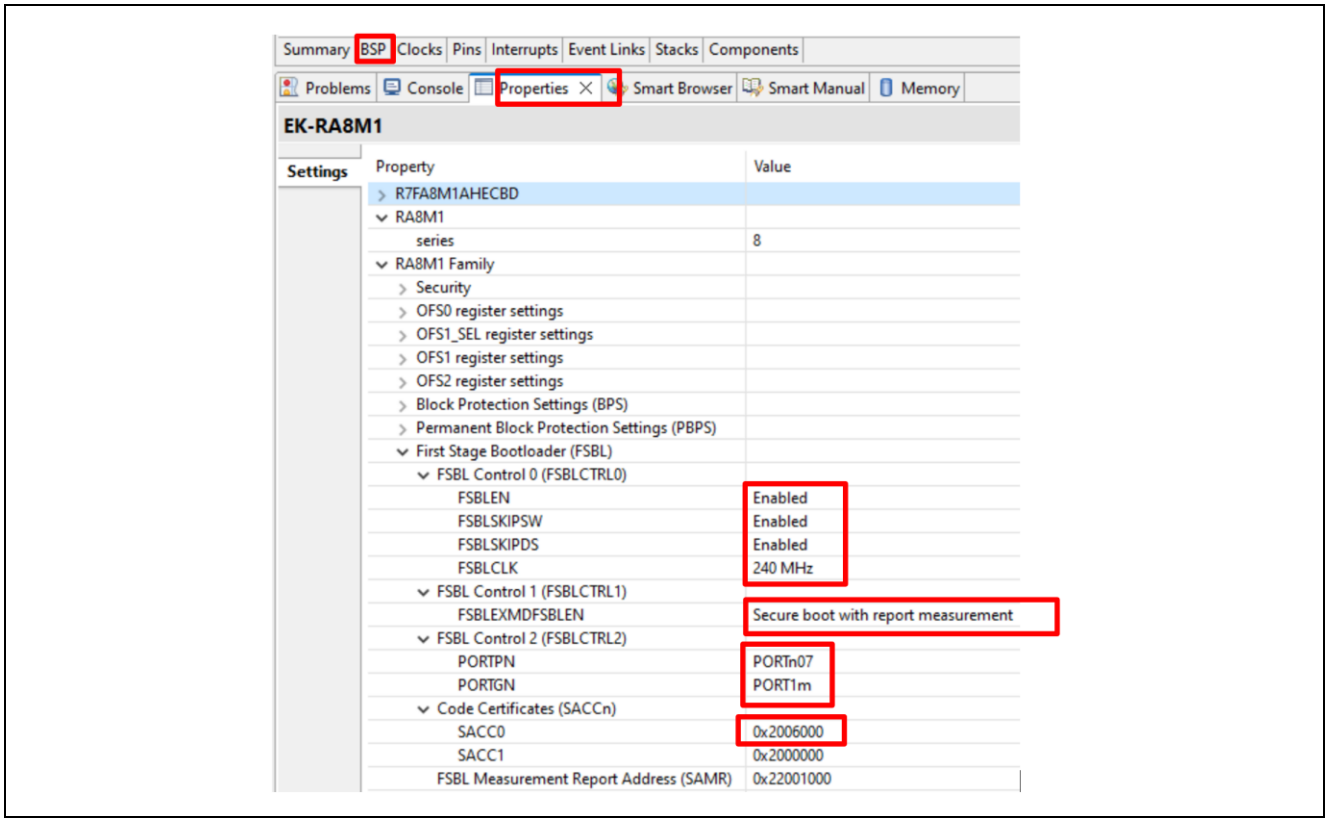


図 4. FSBL の Setting

RA8 MCU を対象とするアプリケーションでは、FSBL を必ず使用する必要はなく、**FSBLEN** プロパティを **Disabled** に Setting することで FSBL を未使用にできます。この場合、プログラミング中または実行前のアプリケーションの検証は行われません。MCU は、FSBL を含まない他の RA MCU と同じように使用できます。

FSBL の検証の実行には時間がかかるため、特定のタイプのリセット後に検証プロセスをスキップしても問題ない場合があります。ソフトウェアリセット時に FSBL をスキップするには、**FSBLSKIPSW** プロパティを **Enabled** に Setting します。同様に、ディープソフトウェアスタンバイリセットからのリセット時に FSBL をスキップするには、**FSBLSKIPDS** プロパティを **Enabled** に Setting します。

CRC とセキュアブートの両方で、図 5 の FSBL を使用した検証オプションに示されているように、**レポート測定**オプションの有無を選択することができます。

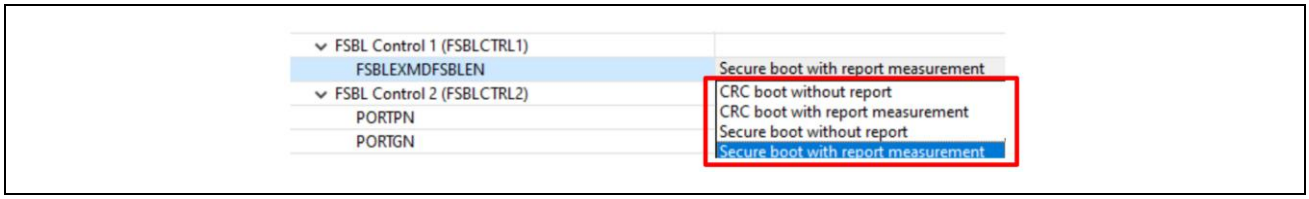


図 5. FSBL を使用した検証オプション

レポート測定を選択した場合、図 6 の情報が SAMR レジスタに格納されているメモリアドレスに出力されます。BSP タブの **FSBL Measurement Report Address** プロパティを使用して Setting されます(参照)。測定レポートの内容は、ARM Platform Security Architecture Attestation API の要件を満たすように生成されます。使用方法については、このリンクの最新バージョン ARM PSA Certified Attestation API マニュアルを参照してください。

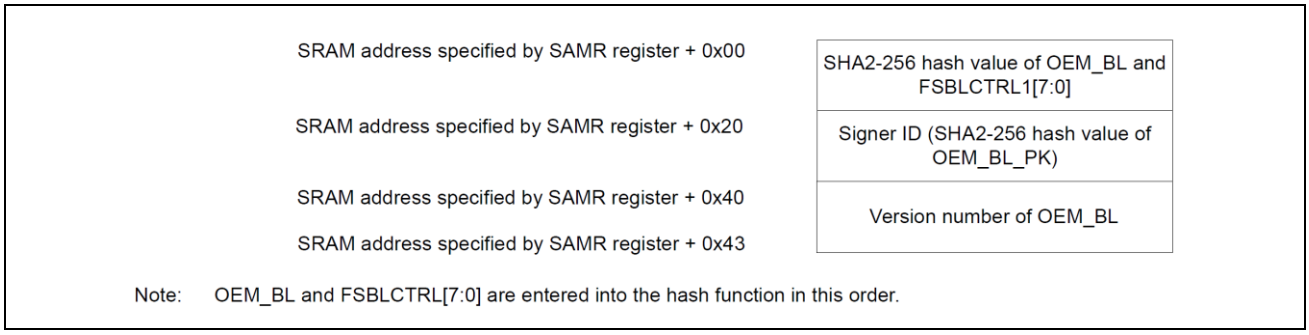


図 6. 測定レポートの内容と場所

CRC ブートの場合、Signer ID は計算された CRC を 8 回出力します。各 CRC32 は 4 バイトで、Signer ID は 32 バイトです。
ブートに失敗すると、FSBLCTRL2 レジスタで Setting されたポートに“High”出力され(図 4 を参照)、MCU は CPU スリープモードに移行します。例えば、図 4 では P107 ピンがセキュアブート失敗時に”High“を出力するピンとして選択されています。このピンはボード上の赤色の LED を制御し、ブートシーケンスが失敗すると LED を点灯します。

1.2.1 製品へのプログラミング中の OEM_BL 検証の概要

製品へのプログラミング中、OEM_BL は ECDSA を使用して MCU ブートアプリケーションプログラムによって検証されます。RA8 MCU は、JTAG、SCI、および USB インタフェースを介して MCU ブート モードへのアクセスを提供します。ブートモードの詳細については、ユーザーズマニュアル ハードウェア編の動作モードを参照してください。

ブートアプリケーションプログラムは、例えば RFP を通して通信が確立した後、OEM_ROOT_PK のハッシュをシステムの Root of Trust としてマッピングされていないフラッシュ領域にプログラムします。また OEM_BL が MCU にプログラムされると、コード証明書のイメージバージョンを読み取ります。

イメージバージョンが以前にプログラムされたバージョンよりも新しいバージョンである場合、ブートアプリケーションプログラムは、OEM_ROOT_PK(キー証明書に保存されている)のハッシュを計算し、MCU にプログラムされた OEM_ROOT_PK のハッシュと比較します。

これにより、コード証明書の OEM_BL_PK が検証されます。OEM_BL_PK が検証されると、ブートアプリケーションプログラムがコード証明書の署名を使用して OEM_BL を検証します。

OEM_BL が検証されると、ブートアプリケーションプログラムはアンチロールバックカウンタをコード証明書にプログラムされたイメージバージョンまでインクリメントし、OEM_BL_digest を計算します。その後、コード証明書と OEM_BL_digest が MCU にプログラムされます。

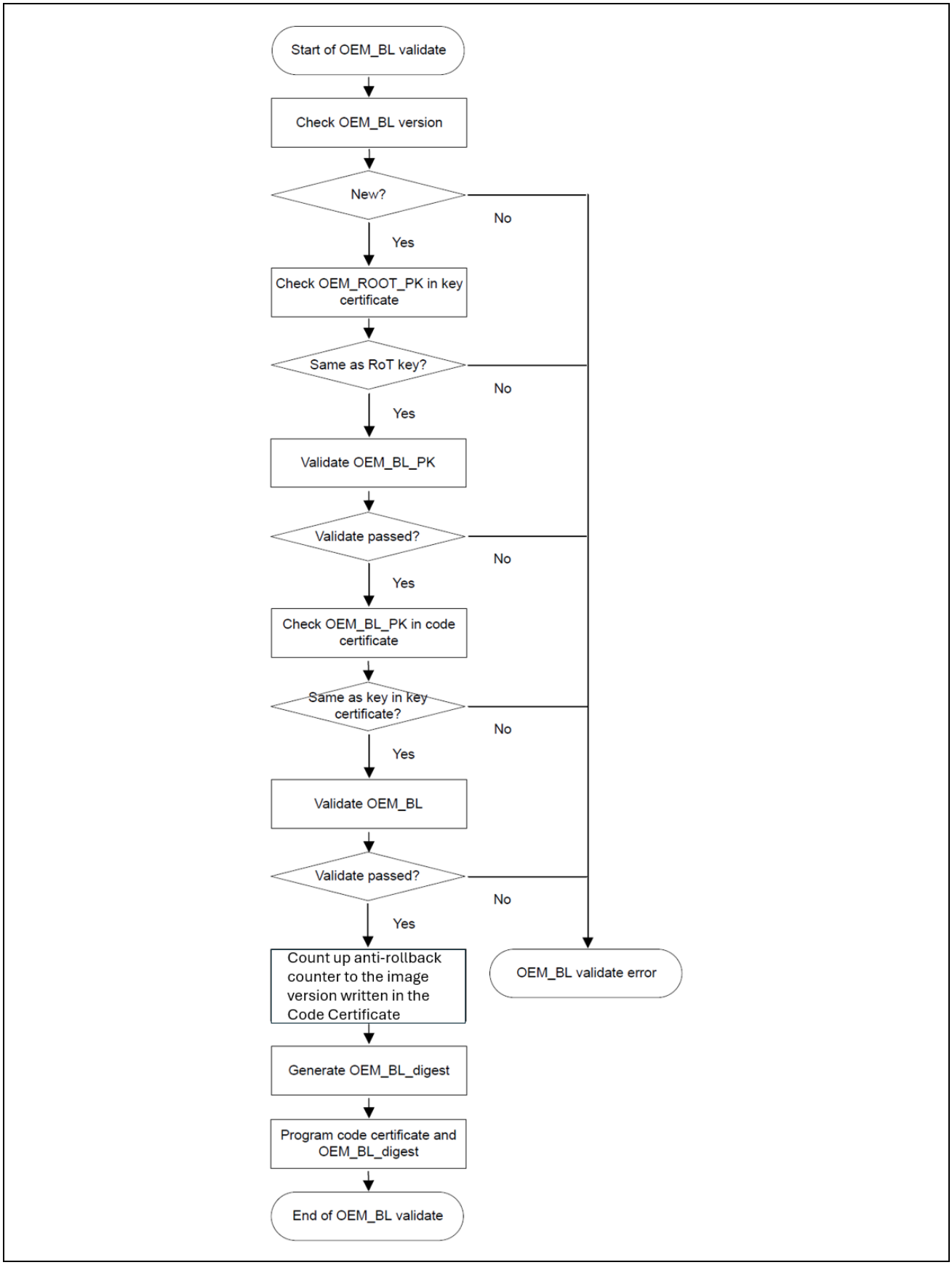


図 7. 製品プログラミングモードでの OEM_BL 検証フロー

1.2.2 シングルチップ動作時の OEM_BL 検証

シングルチップモードでは、OEM_BL は HMAC-SHA256 を使用して FSBL で検証されます。OEM_BL 製品へプログラミングが成功したと仮定します。セキュアブートが有効になっている場合、MCU リセット後に ROM 内の FSBL が実行されます。FSBL は、OEM_BL とコード証明書の HMAC 値(OEM_BL_digest)を計算し、製品へプログラミング中、コードフラッシュに格納される OEM_BL_digest と比較します。
OEM_BL_digest が一致する場合、FSBL は OEM_BL にジャンプします。一致しない場合、FSBL は CPU をスリープモードに移行させ、オプションでユーザが選択したポートピンに”High”を出力します。

CRC ブートが選択されると、FSBL は OEM_BL の CRC を計算し、コード証明書にあるとされる CRC 値と比較します。CRC 値が一致した場合、FSBL は OEM_BL にジャンプします。

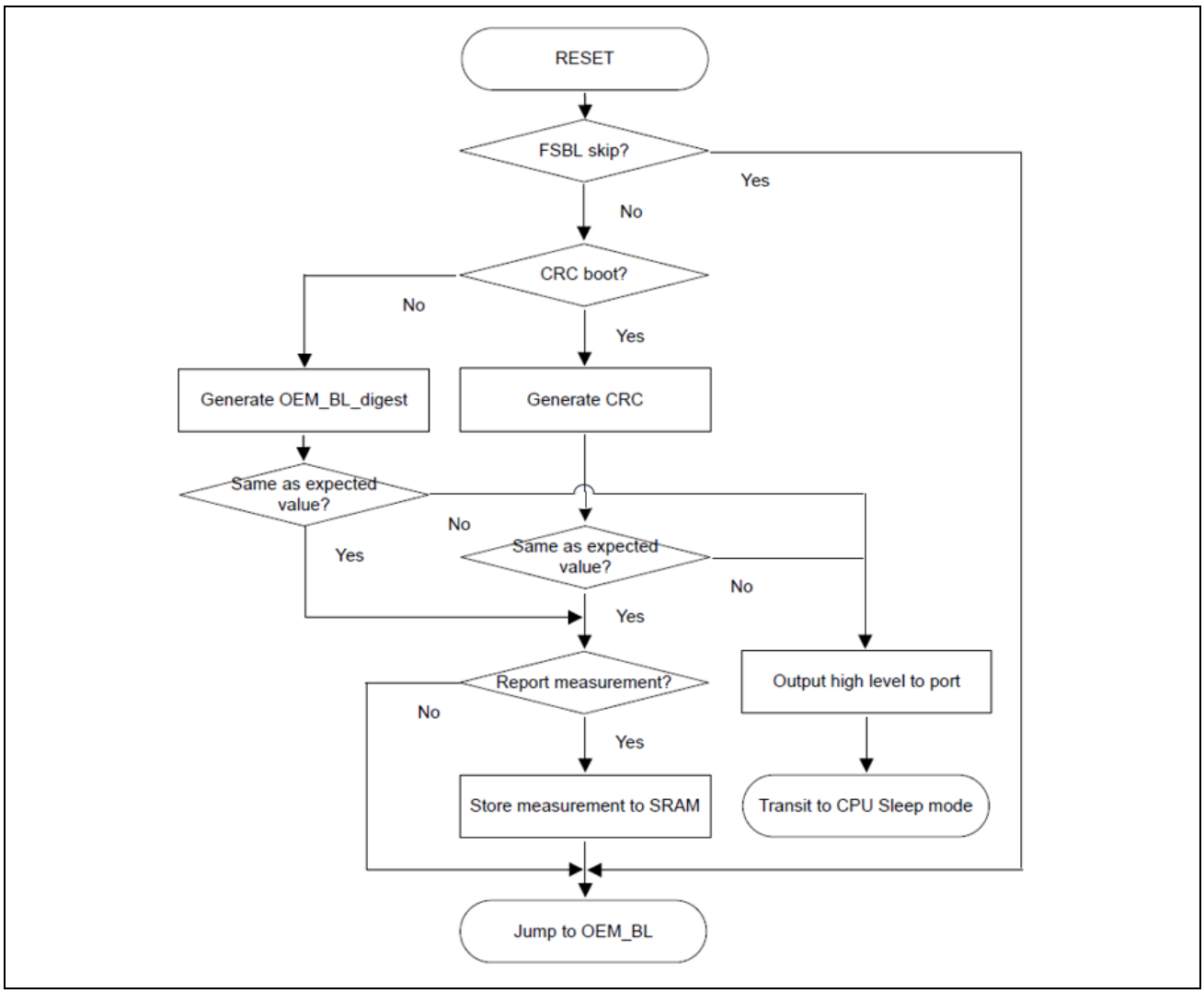


図 8. シングルチップモードでの OEM_BL の検証

1.2.3 シングルチップモードで HMAC-SHA256 を使用する利点

HMAC-SHA256 は、RA8 MCU のハードウェア機能に基づいて以下の利点を提供します。:

- HMAC の共有シークレット(MCU の HUK)が MCU の外部に公開されないため、セキュリティが維持されます。
- HUK は MCU 固有であり、複製の防止、保護を提供します。
- デジタル署名の検証よりも動作が大幅に高速です。
- HMAC-SHA256 のオペレーションには量子耐性があります。

1.3 認証チェックに使用される認証情報の詳細

このセクションでは、ECDSA および HMAC-SHA256 操作に基づくシステムの Chain of Trust について説明します。

システムの Chain of Trust は、以下のコンポーネントによって確立されます。

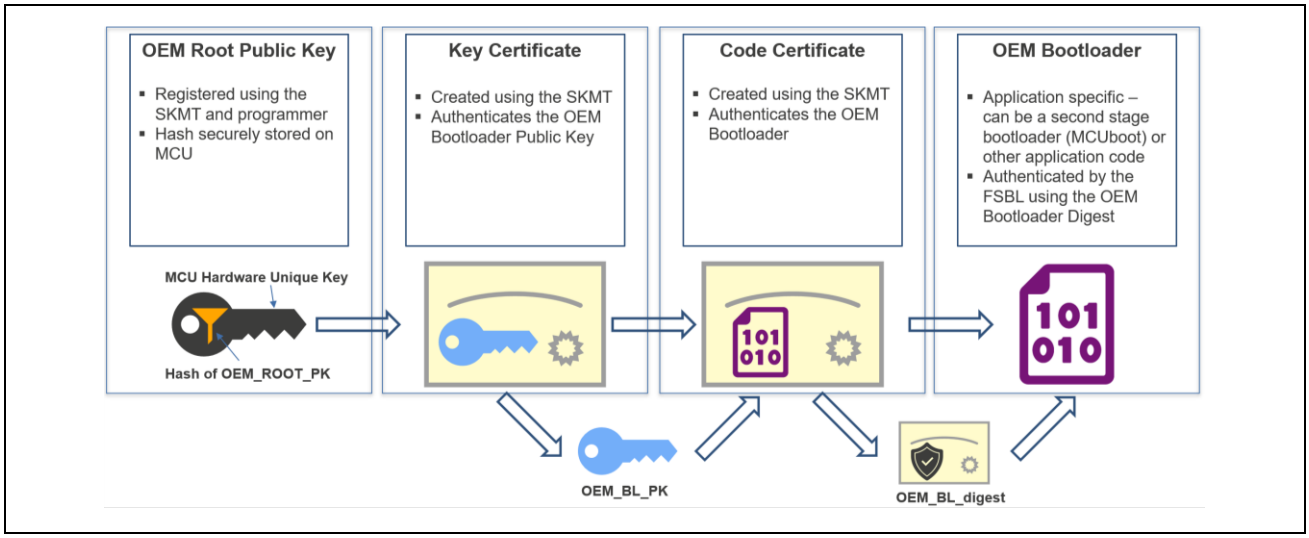


図 9. Chain of Trust

以下のセクションでは、図 9 に示されたさまざまな認証情報の形式と内容、製品のプログラミング中や製品アプリケーションプログラムの実行前におけるその使用方法について説明します。

1.3.1 Root of Trust

セキュアブートを使用する場合は、2 セットの secp256r1 ECC キーペアを生成する必要があります。:

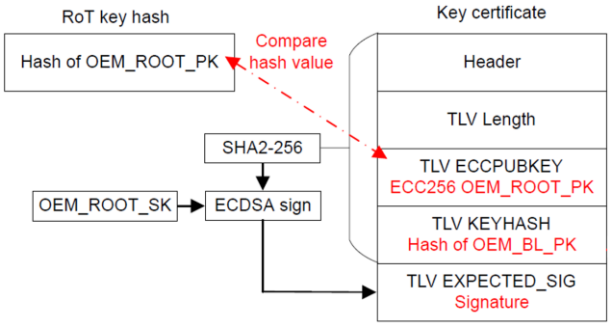
- OEM_ROOT_PK: OEM Root Key pair の公開部分
- OEM_ROOT_SK: OEM Root Key pair のプライベート(シークレット)部分
- OEM_BL_PK: OEM_BL Key pair の公開部分
- OEM_BL_SK: OEM_BL Key pair のプライベート(シークレット)部分

OEM_ROOT_PK の SHA256 ハッシュは、RA8 MCU のユーザーズマニュアル ハードウェア編のセキュアキーインジェクションで説明されているプロセスを使用して、Root of Trust としてロック可能なデータフラッシュ領域(HOEMRTPK)に格納されます。Root of Trust のインジェクションは、アプリケーション製品へプログラミング中に行われます。

1.3.2 キー証明書

キー証明書は、HMAC ブート対応アプリケーション製品へのプログラミング時に重要な役割を果たします。以下は、キー証明書の作成と使用に関する重要な情報です。

- キー証明書は OEM_ROOT_SK によって署名されます。
- OEM_ROOT_PK は署名されたキー証明書に含まれ、HOEMRTPK によって検証されます。
- OEM_BL_PK のハッシュはキー証明書に含まれており、ブートアプリケーションプログラムによって OEM_BL_PK を検証するために使用されます。
- キー証明書は OEM_BL_PK が検証された後に破棄されます。



Field		Size [byte]	Description
Header	Magic	4	Set to 0x6B657963
	Manifest version	4	Set to 0x00010000
	Flags	4	Reserved for future use
	Reserved	20	Reserved for future use
TLV Length		4	Length in bytes of TLV field
TLV ECCPUBKEY	Type & Length	4	Set to 0x00088010
	Value	64	ECC P-256 OEM_ROOT_PK
TLV KEYHASH	Type & Length	4	Set to 0x10144008
	Value	32	SHA2-256 hash value of OEM_BL_PK
TLV EXPECTED_SIG	Type & Length	4	Set to 0x20088410
	Value	64	Signature

図 10. OEM_BL_PK を検証するためのキー証明書

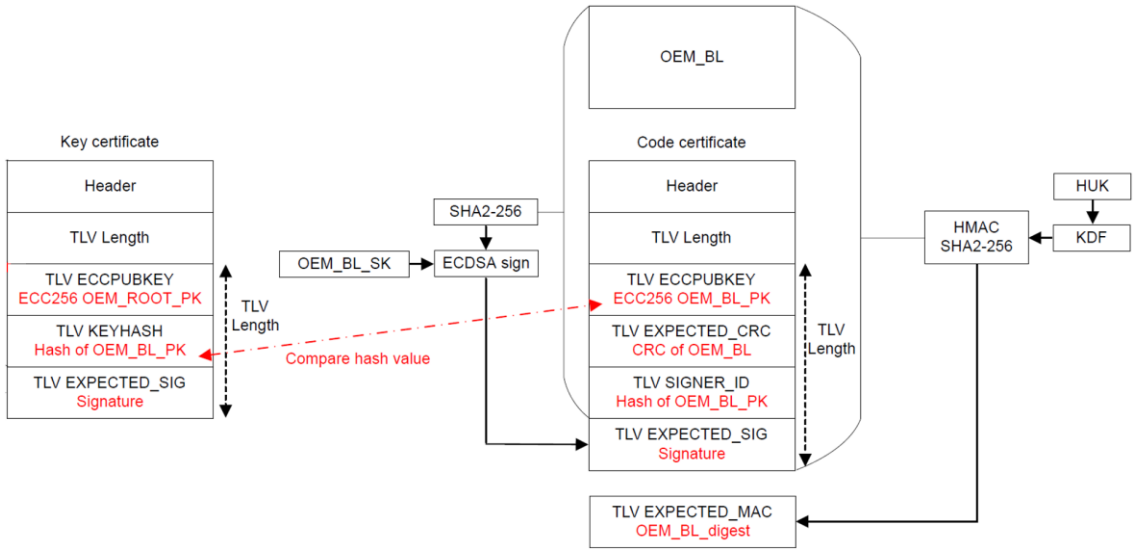
1.3.3 コード証明書

コード証明書は、MCU の製品へプログラミング時、MCU のシングルチップ動作モード時において重要な役割を果たします。

- コード証明書は OEM_BL_S によって署名されています。
- OEM_BL_PK はコード証明書に含まれています。キー証明書に保存されたハッシュによって検証されます。その後、OEM_BL_PK はコード証明書に含まれている OEM_BL の署名検証に使用します。
- コード証明書には、シングルチップモード中の FSBL 検証プロセスに使用される認証情報が含まれています。OEM_BL がプログラムされた後、フラッシュメモリ(コードフラッシュまたはデータフラッシュ)にプログラムする必要があります。
- セキュアブートを使用する際、アプリケーションのイメージバージョンはコード証明書の一部となります。有効なイメージバージョンは 1～64 であり、セキュアブート付き FSBL が有効な場合、アプリケーションは 64 回更新できることになります。FSBL はアンチロールバックスキームを実装しています。したがって、新しいアプリケーションイメージに更新するには、次の新しいイメージのバージョンが前のアプリケーションイメージバージョンより新しくなければなりません。イメージバージョンの操作や使用方法についての詳細は、ユーザーズマニュアル ハードウェア編のアンチロールバックカウンタを参照してください。
- フラッシュ内のコード証明書の場所は、SACC0 レジスタ、SACC1 レジスタで指定する必要があります。SACC0 レジスタは、デュアルバンクモードで下位バンクがバンク 0 の場合、またはリニアモードで起動領域がデフォルトブロック(ブロック 0)の場合に、コード証明書の開始アドレスを指定します。SACC1 レジスタは、デュアルバンクモードで下位バンクがバンク 1 の場合、またはリニアモードで起動領域が代替ブロック(ブロック 1)の場合に、開始アドレスを指定します。SACCx レジスタ Setting は、RA スマートコンフィギュレータと FSP ボードサポートパッケージ(BSP)によって処理されます。図 4 に Code Certificate プログラミングの位置を示しています。特定のアプリケーションでは必要に応じて調整してください。
- FSBL 署名の認証では、TLV EXPECTED_CRC フィールドは使用されません。コード証明書の構成は変更されません。
- 製品へプログラミング中に OEM_BL の検証に成功すると、ブートアプリケーションプログラムは MCU の HUK を使用して、OEM_BL とコード証明書の HMAC-SHA256 ダイジェスト(OEM_BL_digest)を計算します。ブートアプリケーションプログラムは、OEM_BL_digest をコード証明書の直後のメモリ領域にプログラムします。

Field		Size [byte]	Description
TLV EXPECTED_MAC	Type & Length	4	Fixed value 0x30184008
	Value	32	Unique OEM_BL_digest in each MCU

図 11. OEM_BL_digest の詳細



Field		Size [byte]	Description
Header	Magic	4	Set to 0x636F6463
	Manifest version	4	Set to 0x00010000
	Flags	4	Set to 0x00000000
	Load Addr	4	Set to 0x02000000
	Dest Addr	4	Set to 0x02000000
	Image size	4	Size in bytes of OEM_BL set in multiples of 16, minimum size is 64 bytes
	Image version	4	Version number of OEM_BL. Can be specified from 1 to 64
	Build number	4	Set to 0x00000000
TLV length		4	Length in bytes of TLV field
TLV ECCPUBKEY	Type & Length	4	Set to 0x01088010
	Value	64	ECC P-256 OEM_BL_PK
TLV EXPECTED_CRC	Type & Length	4	Set to 0x40000001
	Value	4	CRC32 of OEM_BL
TLV SIGNER_ID	Type & Length	4	Set to 0x10144008
	Value	32	SHA2-256 hash value of OEM_BL_PK
TLV EXPECTED_SIG	Type & Length	4	Set to 0x25088410
	Value	64	Signature of [Code Certificate | OEM_BL], signed by the OEM_BL_SK When generating the signature, enter the code certificate and OEM_BL in the hash function in this order.

図 12. HMAC ブートを使用して OEM_BL を検証するためのコード証明書

1.3.4 製品へのプログラミングとセキュアブートにおける認証情報の概要

アプリケーションのプログラミング段階では、以下の項目が MCU にプログラムされます。:

- OEM_BOOT_PK のハッシュ
- OEM_BL(アプリケーションイメージ)
- コード証明書

以下の図は、セキュアブートを使用した製品のプログラミングに SKMT と RFP を使用した全体的なワークフローをまとめたものです。SKMT と RFP の使用法についての詳しい手順は、セクション 2 で説明します。

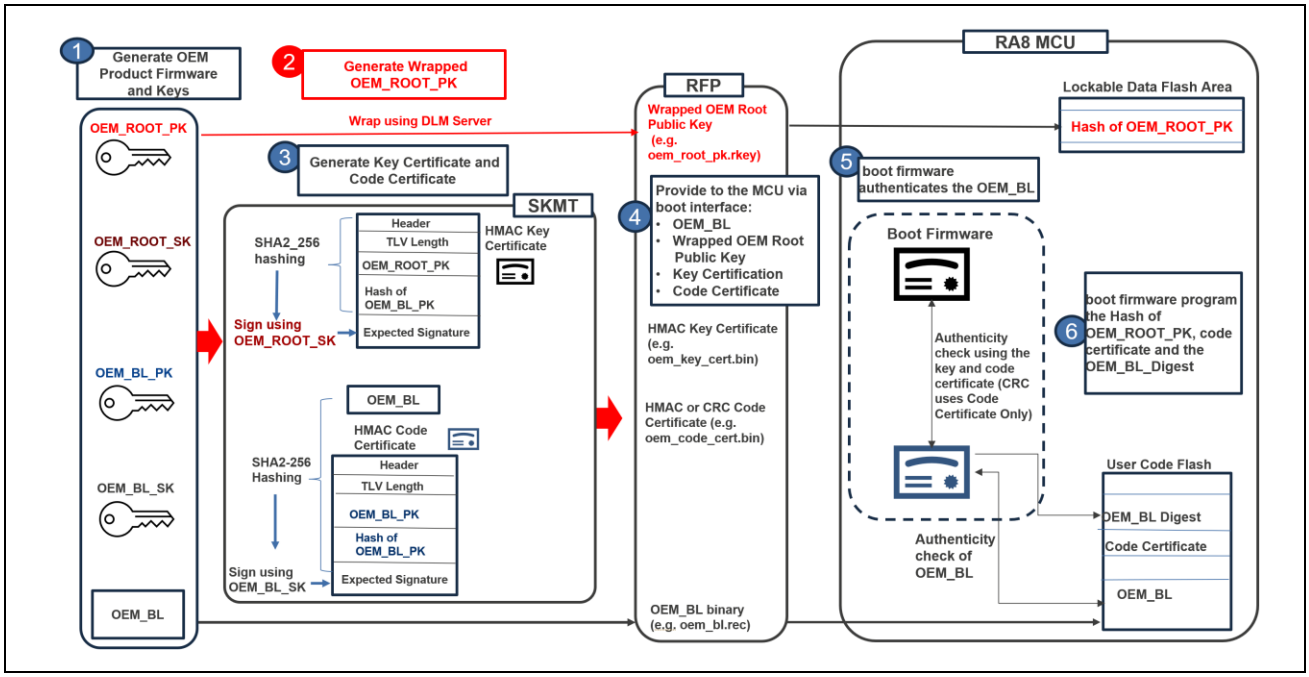


図 13. セキュアブート製品のプログラミング認証情報の概要

1.4 整合性チェックで使用される認証情報の詳細

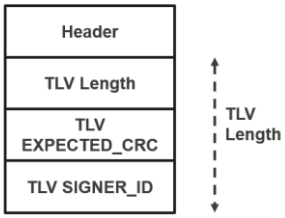
CRC を使用した検証は比較的簡単です。上記のセキュアブートプロセスとの違いは以下のとおりです。:

- OEM Root Public Key(OEM_ROOT_PK)は使用されないので、OEM_ROOT キーペアを作成する必要も、OEM_ROOT_PK をインジェクションして OEM_ROOT_PK のハッシュ(HOEMRTPK)を作成する必要もありません。
- キー証明書は使用されず、OEM_BL は署名されないので OEM_BL キーペアを作成する必要はありません。
- アンチロールバックポリシーは適用されません。アンチロールバックカウンタ(ARC_OEMBLn)とコード証明書のイメージバージョンフィールドは使用されません。
- 製品へプログラミング中に、ブートアプリケーションプログラムは OEM_BL の CRC32 を計算します。計算された CRC32 の値がコード証明書の値と一致する場合、ブートアプリケーションプログラムは SACC0/1 レジスタで指定された場所にコード証明書をプログラムします。
- シングルチップモードでは、FSBL は OEM_BL の CRC32 を計算し、コード証明書の CRC と比較します。CRC の値が一致すると、FSBL は OEM_BL にジャンプします。

1.4.1 コード証明書

CRC ブートのコード証明書には、以下のコンポーネントが含まれています。TLV ECCPUBKEY と TLV_EXPECTED_SIG フィールドは、コード証明書には存在しません。イメージバージョンフィールドは、ブート中には使用されません。

Code certificate



Field		Size [byte]	Description
Header	Magic	4	Set to 0x636F6463
	Manifest version	4	Set to 0x00010000
	Flags	4	Set to 0x00000000
	Load Addr	4	Set to 0x02000000
	Dest Addr	4	Set to 0x02000000
	Image size	4	Size in bytes of OEM_BL set in multiples of 16, minimum size is 64 bytes
	Image version	4	Version number of OEM_BL. Can be specified from 1 to 64
	Build number	4	Set to 0x00000000
TLV length		4	Length in bytes of TLV field
TLV EXPECTED_CRC	Type & Length	4	Set to 0x40000001
	Value	4	CRC32 of OEM_BL
TLV SIGNER_ID	Type & Length	4	Set to 0x10144008
	Value	32	SHA2-256 hash value of OEM_BL_PK

図 14. CRC ブートを使用して OEM_BL を検証するためのコード証明書

1.4.2 製品へのプログラミング、CRC ブートにおける認証情報の概要

RFP によるアプリケーションのプログラミング段階では、以下の項目が MCU にプログラムされます。:

- OEM_BL(アプリケーションイメージ)
- コード証明書

以下の図は、CRC ブートを使用した製品へのプログラミング中に SKMT(ルネサスセキュアキーマネージメントツール)と RFP を使用した全体的なワークフローをまとめたものです。SKMT と RFP の使い方についての詳しい手順は、セクション 3 に説明します。

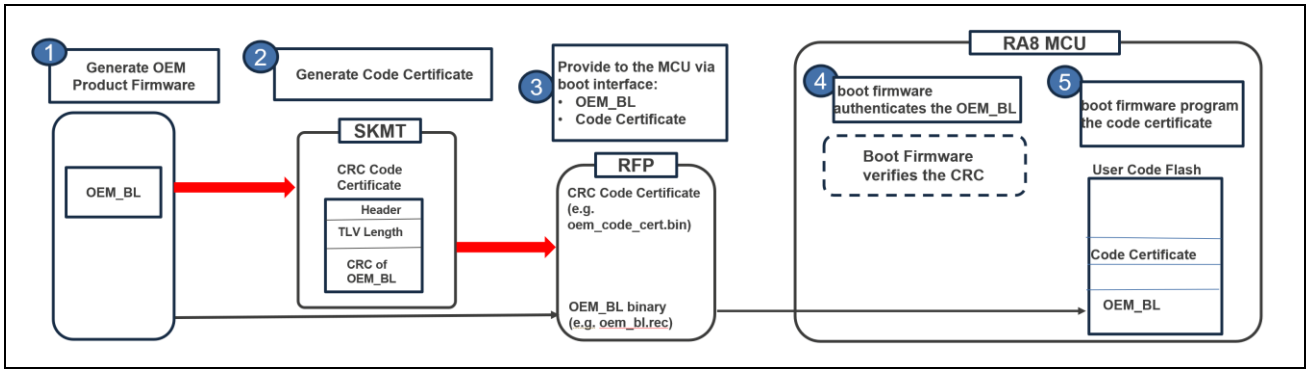


図 15. CRC ブート製品へのプログラミング認証情報の概要

1.5 ルネサスセキュアファクトリプログラミングでの FSBL 使用

RA8 MCU は、セキュアファクトリプログラミングにより、暗号化されたアプリケーションプログラムイメージのプログラミングをサポートし、非セキュアな環境でもセキュアなアプリケーションプログラムのプログラミングを可能にします。

ルネサスセキュアファクトリプログラミングは、FSBL の有無にかかわらずアプリケーションプログラムを実装できます。FSBL をセキュアファクトリプログラミングで使用する場合、アプリケーションは FSBL を Setting する必要があります。暗号化されたイメージが MCU にプログラムされた後、リセットを実行する前に FSBL を起動します。セキュアファクトリプログラミング後にデバイスを消去して FSBL をオフにするには、MCU Initialize Device コマンドを実行します。(図 23 のように RFP を使用します)。詳細については、ユーザーズマニュアル ハードウェア編のセキュアファクトリプログラミングを参照してください。

2. 認証された製品プログラミングと HMAC-SHA256 ブートデモンストレーション

このセクションでは、認証された製品プログラミングと HMAC ブートの認証の手順を説明します。なお、本 APN には以下も同梱しており、これらを使用したロセスを示しています。

blinky_hmac_boot_ra8m1.zip
blinky_crc_boot_ra8m1.zip

Root of Trust を導入する手順は、アプリケーションノート R11AN0496 で説明されているセキュアキーインジェクションプロセスと多くの共通点があるので、必要に応じて R11AN0496 のセクションを参照してください。

2.1 ECC secp256r1 キーペア 2 セットの準備

前のセクションで説明したように、システムの Chain of Trust を構築するには 2 セットの ECC secp256r1 キーペアが必要です。このアプリケーションプロジェクトでは、OpenSSL を使用してキーペアを生成する方法が示されています。さらに、NIST CAVP ECDSA テストベクタを使って Chain of Trust を構築する方法も説明されています。

2.1.1 OpenSSL を使用した ECC キーペアの生成

評価用 PC の OS が Linux の場合は、次のリンクから OpenSSL を取得できます:
https://www.openssl.org/source/

Windows の場合は、次のリンクから OpenSSL を取得できます:
https://slproweb.com/products/Win32OpenSSL.html

OpenSSL のダウンロード後、インストールし、コマンドプロンプトから\bin フォルダに移動します。

以下のコマンドで OEM Root Key Pair を生成します。:

C:\Program Files\OpenSSL-Win64\bin>openssl ecparam -name prime256v1 -genkey -noout -out c:\RA8_FSBL\openssl_keys\oem_root_private_key.pem

以下のコマンドで OEM Root Key Pair の Public Key を生成します。

C:\Program Files\OpenSSL-Win64\bin>openssl ec -in c:\RA8_FSBL\openssl_keys\oem_root_private_key.pem -pubout -out c:\RA8_FSBL\openssl_keys\oem_root_public_key.pem

```
read EC key
writing EC key
```

同じコマンドを使用して OEM_BL Key Pair を生成することができます。:

C:\Program Files\OpenSSL-Win64\bin>openssl ecparam -name prime256v1 -genkey -noout -out c:\RA8_FSBL\openssl_keys\oem_bl_private_key.pem

以下のコマンドで、OEM_BL Key Pair の Public Key を生成します。

C:\Program Files\OpenSSL-Win64\bin>openssl ec -in c:\RA8_FSBL\openssl_keys\oem_bl_private_key.pem -pubout -out c:\RA8_FSBL\openssl_keys\oem_bl_public_key.pem

```
read EC key
writing EC key
```

2.1.2 ECC キーペアとした NIST CAVP テストベクタのダウンロード

このアプリケーションのプロジェクトでは、FSBL を使用した HMAC によるブートをデモンストレートするために、以下の NIST Cryptographic Algorithm Validation Program(CAVP)テストベクタが使用されます。

NIST CAVP テストベクタは、以下のリンクからダウンロードできます。使用するのは ECDSA ベクタです。

Cryptographic Algorithm Validation Program | CSRC (nist.gov)

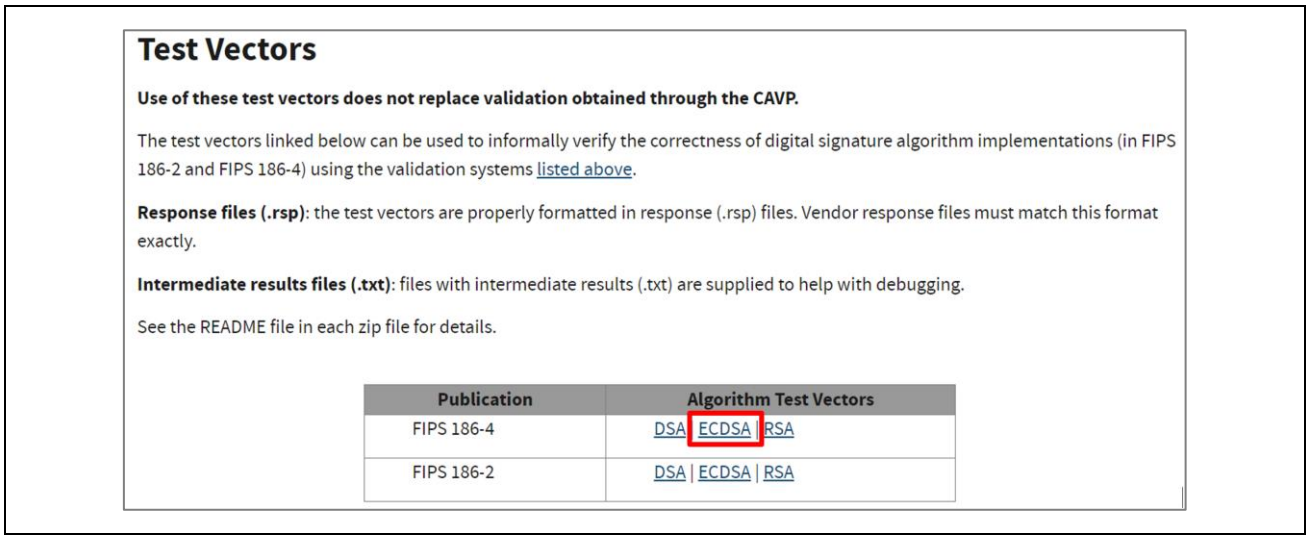


図 16. ECDSA テストベクタ

zip ファイル `186-4ecdsatestvectors.zip` をダウンロード後、解凍してプレーンテキストファイル `KeyPair.rsp` で以下のベクタを見つけてください。

OEM ルートキーペア用の ECC P256 secp256r1 キーペア

以下のベクタは、OEM Root Key Pair に使用されます。これはセクション 2. 2 で Wrapped OEM Root Public Key の生成に使用されます。また、セクション 2. 4 でキー証明書を生成する際にも使用されます。

```
d = c9806898a0334916c860748880a541f093b579a9b1f32934d86c363c39800357
Qx = d0720dc691aa80096ba32fed1cb97c2b620690d06de0317b8618d5ce65eb728f
Qy = 9681b517b1cda17d0d83d335d9c4a8a9a9b0b1b3c7106d8f3c72bc5093dc275f
```

OEM APP(BL)キーペア用の ECC P256 secp256r1 キーペア

以下のベクタは、OEM APP(BL)キーペアに使用されます。これは、セクション 2. 4 でコード証明書の生成するために使用されます。

```
d = 710735c8388f48c684a97bd66751cc5f5a122d6b9a96a2dbe73662f78217446d
Qx = f6836a8add91cb182d8d258dda6680690eb724a66dc3bb60d2322565c39e4ab9
Qy = 1f837aa32864870cb8e8d0ac2ff31f824e7beddc4bb7ad72c173ad974b289dc2
```

2.2 Root of Trust の導入

前述のように、OEM_ROOT_PK のハッシュはシステムの Root of Trust です。FSBL が有効になっている場合、アプリケーションのプログラミングが成功するとこのハッシュがインジェクションされます。

OEM_ROOT_PK は、RFP を使用して MCU にインジェクションする前に、ルネサス DLM サーバを使ってラップする必要があります。OEM Root Public Key をラップする前に、Renesas Key Wrap Service を使用して User Factory Programming Key をラップする手順を実行してください。UFPK ラップのセキュリティ考慮事項やプロセスについては、R11AN0496 の以下のセクションを参照して、使用背景を理解し、ルネサス DLM サーバとの PGP 暗号化通信を確立してください。

- 顧客の PGP キーペアを生成するには、**Create PGP Key Pair** 章を参照してください。
- DLM サーバに登録するには、**Registration with DLM Server** 章を参照してください。
- DLM サーバと暗号化通信を確立するには、**Exchange User and Renesas PGP Public Keys** 章を参照してください。

2.2.1 Wrapped User Factory Programming Key の生成

R11AN0496 の上記のセクションに基づく操作完了後、次は R11AN0496 の **Wrapping the UFPK** 章を参照して W-UFPK を生成します。ただし、このプロセス中に更新すべき重要な手順がいくつかあります。

まず、**Wrapping the UFPK** 章を読み、その後のセクションを読む前に、OEM Root Public Key インジェクションの目的で Wrapped UFPK を生成するために操作を更新する箇所を特定してください。その後、そのフロー内で次に更新するべき操作を特定し、R11AN0496 に従って W-UFPK を生成してください。

- 特定のプレーンテキストの UPFK を入力すると、DLM サーバ(W-UFPK)の出力は、セキュリティエンジンで選択されたモード(互換モードや保護モード)に応じて変わります。OEM ルートキーをラップする時に DLM サーバで選択されたオプション(MCU 保護モードを使用)と一致させるため、SKMT で UFPK を生成する際には、**Overview** で **RA Family、RSIP-E51A Security Functions、Protected Mode** オプションを選択してください。
  - R11AN0496 は RSIP 互換モードを使用するため、SKMT で RA ファミリ、RSIP-E51A 互換モードを選択する必要があります。FSBL 使用時に必要なプロテクトモードへの変更にご注意ください。

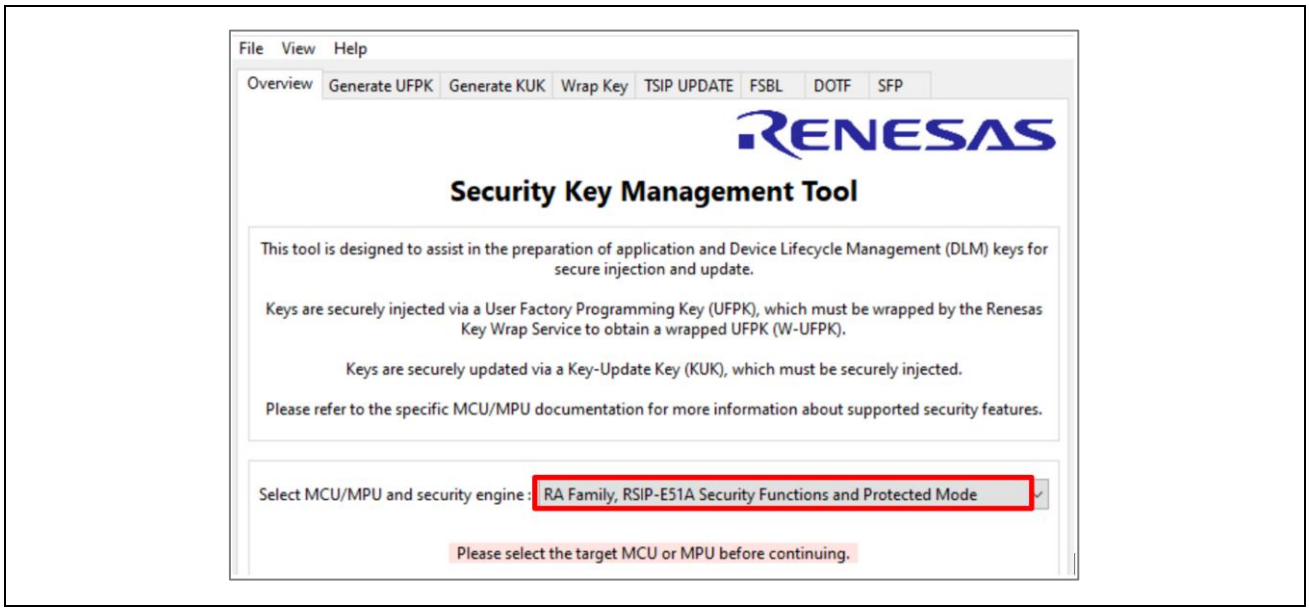


図 17. RSIP-E51A セキュリティ機能とプロテクトモードの選択

- 同じ理由で、DLM サーバを使用して UFPK をラップする場合は、互換モードではなく、**DLM and Protected Mode** で以下の RA8 の項目を選択してください。

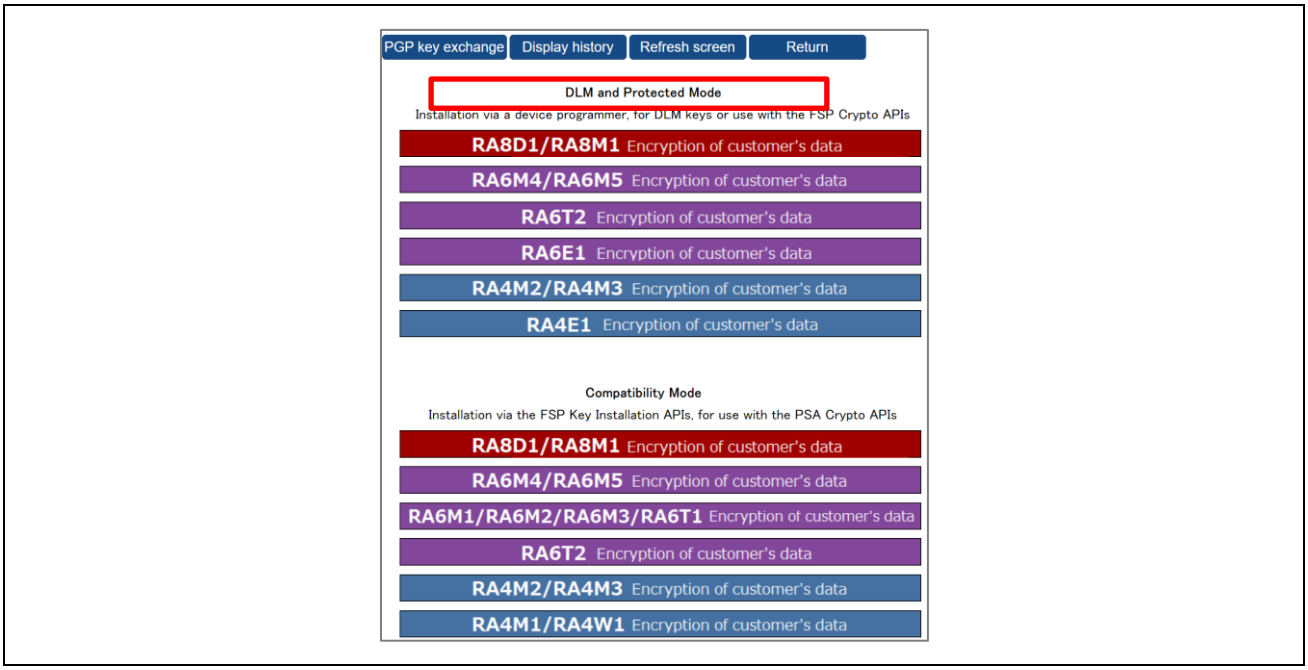


図 18. RA8D1/RA8M1 MCU グループ DLM とプロテクトモードの選択

- Wrapped UFPK を生成するための他の手順は、R11AN0496 で説明されているものと同じです。

このアプリケーションプロジェクトの残りでは、以下の UFPK と W-UFPK がすでに作成されていることを前提としています。:

`ra8x1_ufpk.key`:the UFPK
`ra8x1_ufpk.key_enc.key`: the DLM Wrapped W-UFPK

UFPK と W-UFPK が生成されたら、OEM Root Public Key のラッピングは、R11AN0496 の Wrap an Initial ECC Public Key with the UFPK のパートに記載されている ECC Public Key のラッピングプロセスと同様に行わなければなりません。

2.2.2 OpenSSL と SKMT CLI インタフェースを使用した Wrapped OEM Root Public Key の生成

Secure Key Management Tool(SKMT)がインストールされたら、コマンドラインウィンドウを開き、\cli フォルダに移動し、以下のコマンドを使用して Wrapped OEM Root Public Key を生成します。このキーは、RFP を使用したセキュアキーインジェクションプロセスを使用して MCU に照会されます。

```
C:\Renesas\SecurityKeyMangementTool\cli>skmt.exe /genkey /ufpk
file="C:\RA8_FSBL\ufpk\ra8x1_ufpk.key" /wufpk
file="C:\RA8_FSBL\ufpk\ra8x1_ufpk.key_enc.key" /mcu "RA-RSIP-E51A" /keytype "OEM_ROOT_PK"
/filetype "rfp" /iv "55AA55AA55AA55AA55AA55AA55AA55AA" /key
file="C:\RA8_FSBL\openssl_keys\oem_root_private_key.pem" /output "
C:\RA8_FSBL\openssl_keys\oem_root_pk_cli.rkey"

Output File: C:\RA8_FSBL\openssl_keys\oem_root_pk_cli.rkey

UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F

W-UFPK: 00000000A7BF7EB27054D78E07C504291520678AA7BF7EB27054D78E07C504291520678A

IV: 55AA55AA55AA55AA55AA55AA55AA55AA

Encrypted key:
E71776A79F2BFF879CE3A434C5D0AEFBA934214518114AA89E7CAD10BD45D25623CE6710EC929971BAD200814
B6D633A670447B51347EA24EC7908C9C66AA933F7DD64E2DBDB1B831CED6E3B1347C69B
```

この例の oem_root_pk_cli.rkey は、このアプリケーションプロジェクトのサンプルプロジェクトとは一致しないので、このキーを使用しないでください。

2.2.3 SKMT GUI インタフェースを使用した Wrapped OEM Root Public Key の生成

ここでは、セクション 2. 1. 2 のように OEM Root Public Key pair を使用して、Wrapped OEM Root Public Key を生成します。

SKMT を起動し **Overview** で **RA Family**、**RSIP-E51A Security Functions**、**Protected Mode** を選択します。(図 17 を参照)。次に、SKMT の **Wrap Key** タブに移動し、**Key Type** に **OEM Root public** を選択し、Wrapping Key(UFPK、W-UFPK)を入力します(図 19 参照)。

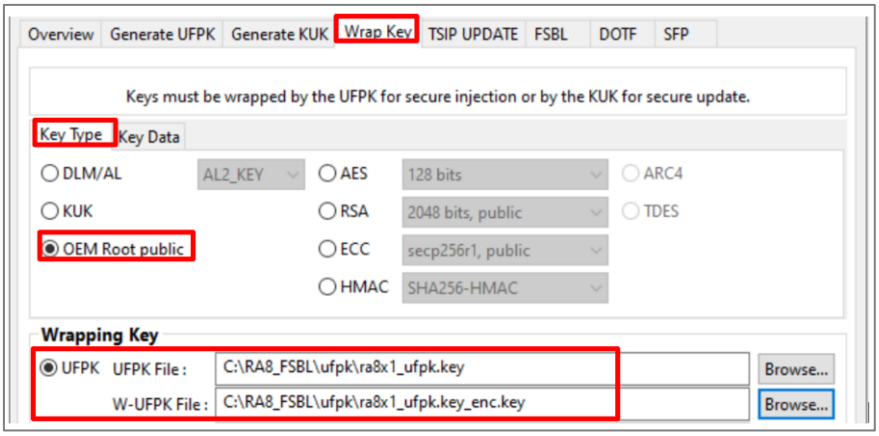


図 19. OEM Root Public Key のラップ

Key Data タブで、OEM Root Public Key を入力します。

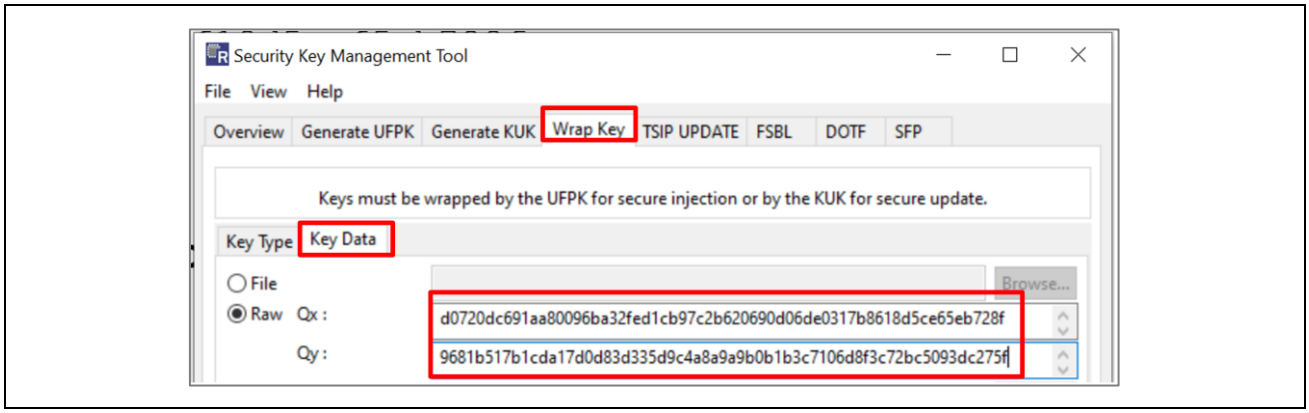


図 20. OEM Root Public Key の提供

次に、出力タイプとして **RFP** を選択し、出力先を指定して、ファイル名を付けてから、**Generate File** をクリックし、Wrapped OEM Root Public Key を生成します。

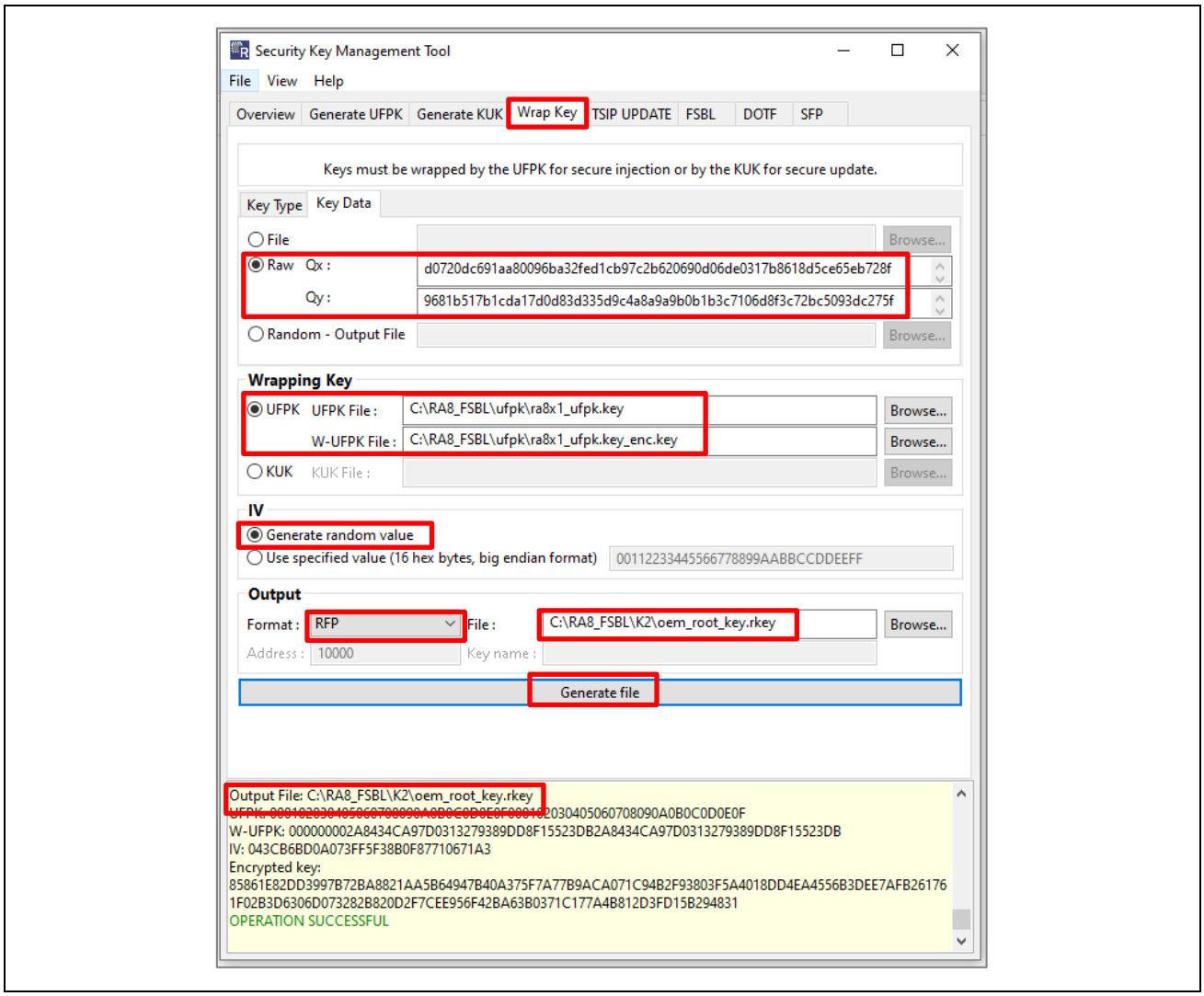


図 21. Wrapped OEM Root Public Key の生成

この Wrapped OEM Root Public Key(`oem_root_key.rkey`)は、システム評価を便利にするために提供されています。この Root Public Key がキーおよびコード証明書の生成に使用されている場合に使用できます。

また、同じ Root Public Key

2.3 HMAC-SHA256 ブートを有効にした Blinky アプリケーションの準備

通常、FSBL（First Stage Boot Loader）を有効にしてブートを行う場合、まず FSBL が無効な OEM_BL（Original Equipment Manufacturer Boot Loader）を開発します。OEM_BL が完全にテストされてから、FSBL を有効にすることができます。

「Create a New Project for Blinky」に記載されているように、blinky_hmac_boot_ra8m1 というプロジェクト例は、RA8M1 用の RTOS サポートなしで作成されています。デフォルトでは、Trust Zone の境界設定がデバッグ設定で有効になっています。

FSBL のプロパティは、Setting に基づいて構成されています。アプリケーションのデバッグを便利にするために、ソフトウェアリセットやディープソフトウェアスタンバイモードからの復帰時に FSBL スキップが有効になっています。さらに、プロジェクトはエラーレポート用端子としてポート P107 ピンを使用しており初期状態で Low に Setting していて、FSBL がブート失敗時にポート P107 ピンを High にすることができます。e² studio のサンプルプロジェクトでは FSBL が有効になっていますが、e² studio でアプリケーションイメージがダウンロードされた後には FSBL は有効になりません。FSBL は次回の電源サイクル後にのみ有効になります。

初期動作で既に FSBL が有効になっている場合 MCU は、**MCU Initialize** コマンドを実行し、ブートアプリケーションプログラムを使用して、オプション設定メモリにある既存の FSBL とトラストゾーンの Setting を消去します。これは、**ルネサスデバイスパーティションマネージャ**(RDPM)(e² studio にネイティブでインストールされています。)または RFP を使用して実現できます。

e² studio を起動したら、**Run -> Renesas Debug Tools -> Renesas Device Partition Manager** を選択して、RDPM を起動できます。

EK-RA8M1 では、デフォルトのジャンパ Setting で SCI ブートモードが有効になっており、SCI 信号がデバッグインターフェイスに指定されています。RDPM を使用する際は、SCI インタフェース、SWD インタフェースのいずれかを選択できます。

以下の画像では、SCI インタフェースが選択されています。

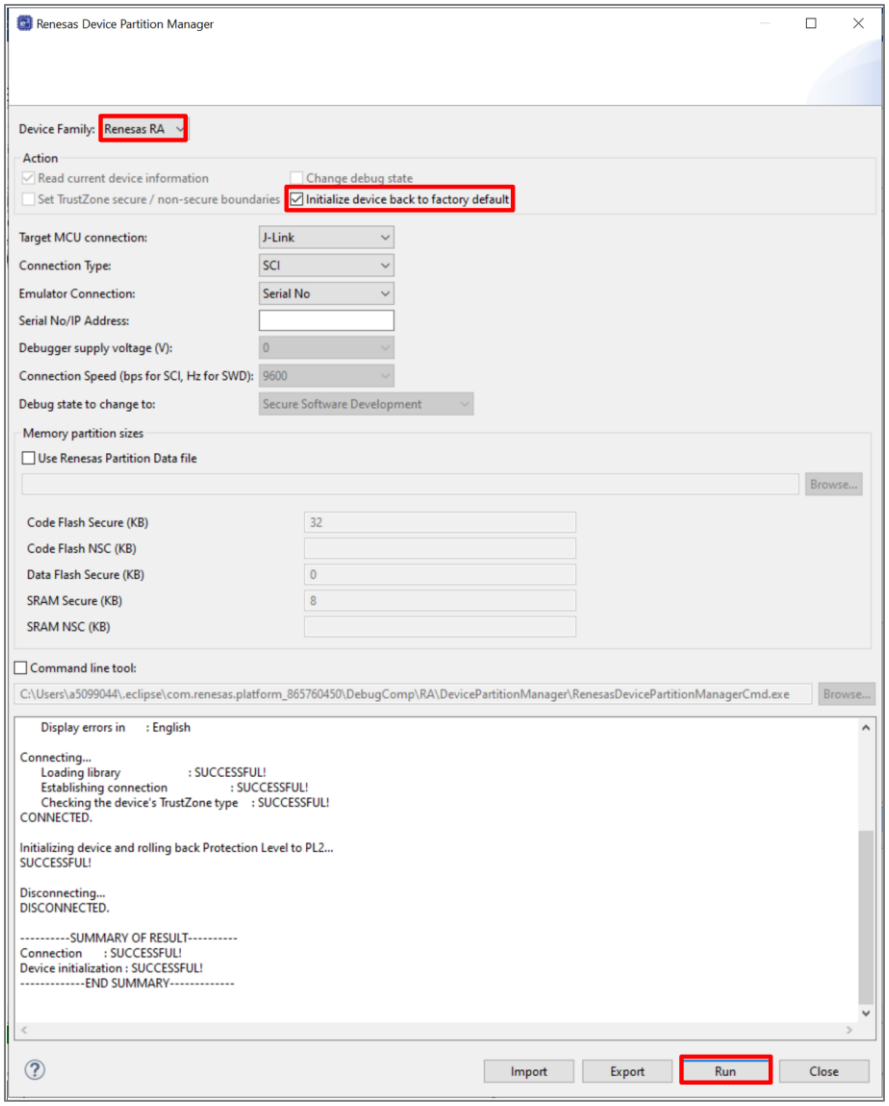


図 22. RDPM を使用した MCU の初期化

図 23 は、RFP を使用してデバイスを初期化することを選択するコマンドを示します。

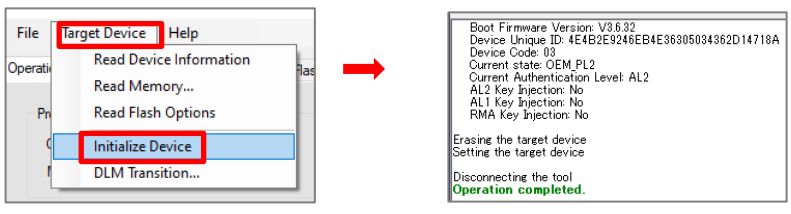


図 23. RFP を使用した MCU の初期化

デバイスを初期化した後、以下の手順に従って付属のサンプルプロジェクトをビルドし、実行してください。

blinky プロジェクト blinky_hmac_boot_ra8m1 をワークスペースにインポートし、configuration.xml ファイルを開きます。そして Generate Project Content」をクリックします。次に、Micro USB ケーブルを使用して、J10 にある J-Link OB USB を評価ボードの USB に接続します。blinky プロジェクト（blinky_hmac_boot_ra8m1）をビルドして実行し、その動作を確認します。

- コンパイル結果は、セクション 2.4 でコード証明書の作成に使用されます。
- 3 つの LED が点滅します。

```
arm-none-eabi-size --format=berkeley "blinky_hmac_boot_ra8m1.elf"
   text    data     bss     dec     hex filename
  11576      92    2220   13888    3640 blinky_hmac_boot_ra8m1.elf

16:53:29 Build Finished. 0 errors, 0 warnings. (took 396ms)
```

図 24. FSBL HMAC ブート対応の Blinky プロジェクト

Tera Term を開き、ボーレートを 115200 に Setting します。

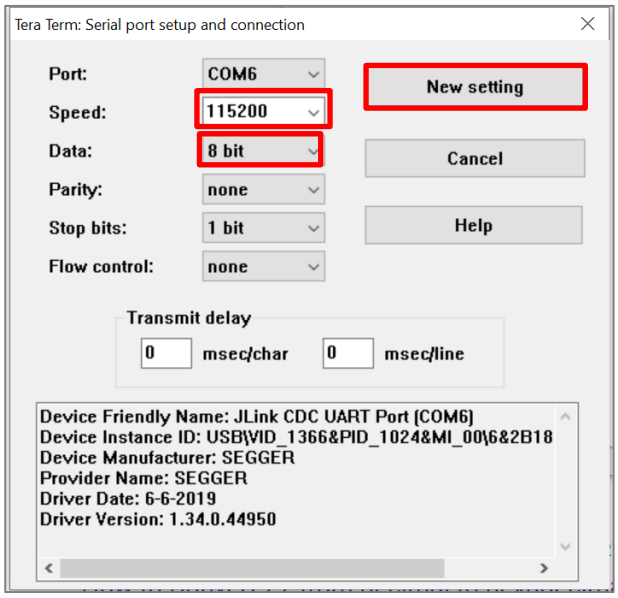


図 25. Tera Term の Setting

New setting をクリックし、コンソールで以下の出力を確認します。FSBL のレジスタ Setting、ブート測定レポート、HMAC ダイジェストの出力に注目してください。出力の意味を理解するには、ユーザーズマニュアル ハードウェア編のレジスタ定義、コード証明書の内容(図 12)を参照してください。FSBL はまだ有効になっていないので、version_num_oem_bl フィールドは 0、ダイジェスト領域は 0xFF になります。

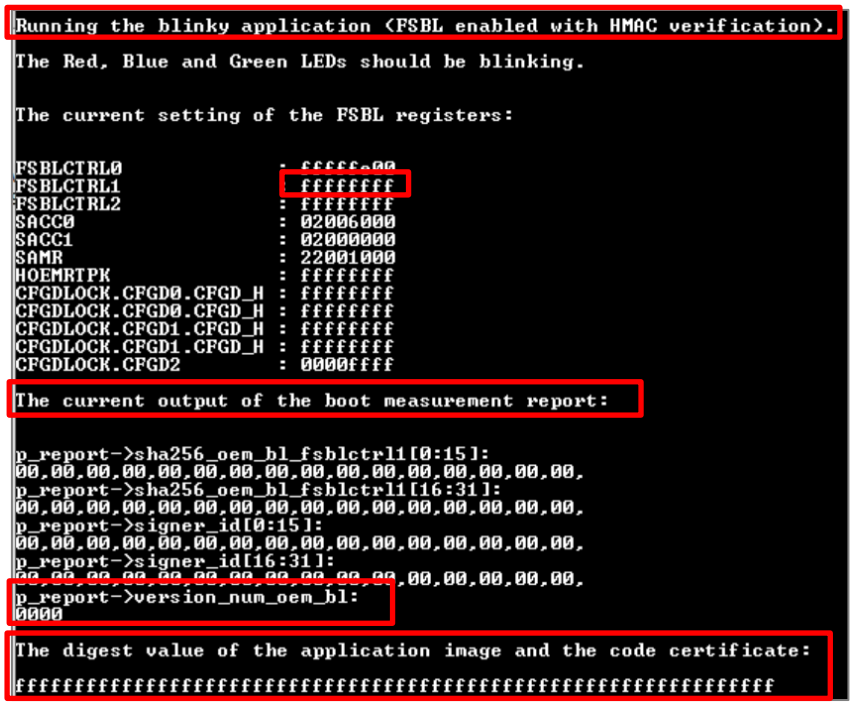
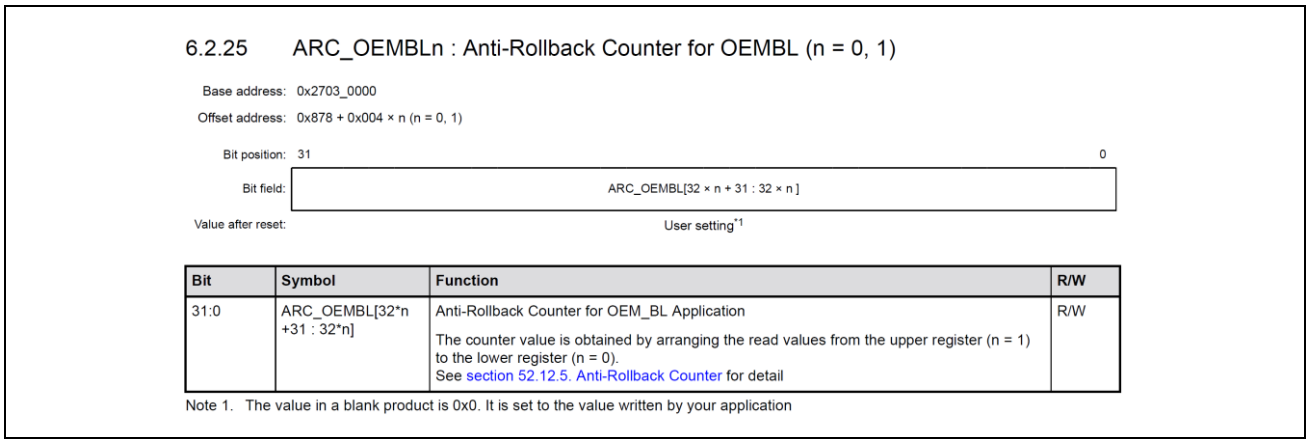


図 26. HMAC ブートの J-Link コンソール出力

MCU に Power-ON リセットが発行されると、FSBL が起動しますが、Root of Trust とコード証明書が MCU にプログラムされていないため、blinky は起動しなくなります。アプリケーションを正常に起動するためには、POR が発行される前に OEM_ROOT_PK をインジェクションし、キー証明書とコード証明書を提示します。このプロセスについては、以降のセクションで説明します。セクション 2.6 では、FSBL を有効にしてアプリケーションを起動する方法について詳しく説明します。

2.4 MCU OEM_BL アンチロールバックカウンタ値の取得

セクション 1. 3. 3 で説明したように、HMAC ブート用のコード証明書には、1～64 までの範囲のイメージバージョンのフィールドを含める必要があります。このイメージバージョン番号は、オプション設定メモリ内の OEM_BL のアンチロールバックカウンタ n(n=0,1)と呼ばれる 2 つのレジスタに永続的に保存されます。(このレジスタの位置については図 2 を参照)



6.2.25 ARC_OEMBLn : Anti-Rollback Counter for OEMBL (n = 0, 1)

Base address: 0x2703_0000

Offset address: 0x878 + 0x004 × n (n = 0, 1)

Bit position:	31 … 0
Bit field:	ARC_OEMBL[32 × n + 31 : 32 × n]
Value after reset:	User setting*1

Bit	Symbol	Function	R/W
31:0	ARC_OEMBL[32*n +31 : 32*n]	Anti-Rollback Counter for OEM_BL Application The counter value is obtained by arranging the read values from the upper register (n = 1) to the lower register (n = 0). See section 52.12.5. Anti-Rollback Counter for detail	R/W

Note 1. The value in a blank product is 0x0. It is set to the value written by your application

図 27. OEMBL のアンチロールバック カウンタ

イメージ バージョン番号はこれら 2 つのレジスタに格納された値ではなく、解釈された値であることに注意してください。このカウンタの動作の詳細については、ユーザーズマニュアル ハードウェア編 セクション **52.12.5 アンチロールバックカウンタ** を参照してください。製品へのプログラミング中は、ブートアプリケーションプログラムのみがこのカウンタにプログラムすることができます。基本的に、アプリケーションの更新が成功するたびに、カウンタは 1 ビットに 1 を Setting していきます。例として、デバイスが 3 回更新された場合、h’7(b 0111)が 2 つのカウンタに格納されます。これらの 32 ビットカウンタは、64 回のアプリケーション更新に使用できます。

アプリケーションプロジェクトには、このレジスタ値を読み取る J-Link スクリプトが含まれています。`read_arc_oembl_jlink_script.zip` を解凍し、`ra8m1.bat` をダブルクリックして、これら 2 つのレジスタの値を読み取ります。図 28 に、J-Link スクリプトを実行した際の出力例を示しています。この例では、レジスタ 0x27030878(ARC_OEMBL0)は 0xFFFFFFFF の値を保持し、レジスタ 0x2703087C(ARC_OEMBL1)は 0x00000001 の値が保持されています。ARC_OEMBL0 と ARC_OEMBL1 レジスタには合計 33 ビットに Setting されています。従って、ユーザが次に選択できるイメージバージョンは 34 以上 64 以下である必要があります。

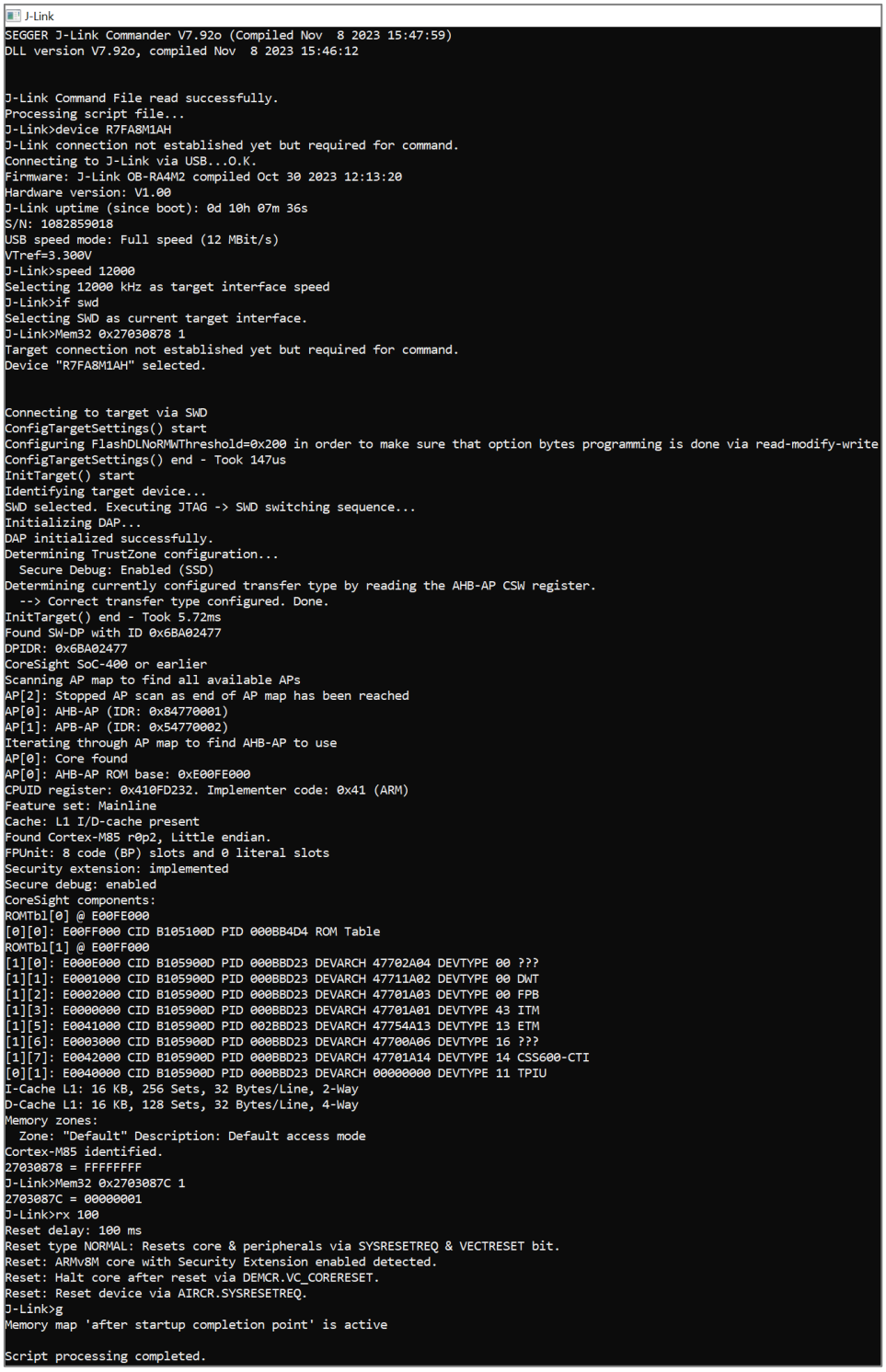


図 28. J-Link スクリプトによる ARC_OEMBL レジスタのリード

2.5 鍵証明書とコード証明書の生成

前述のように、HMAC-SHA256 ベースのブートオプションでは、鍵とコード証明書を生成するには、2 セットの ECC-P256 secp256r1 の公開鍵と秘密鍵が必要です。このセクションでは、2 つのデモンストレーションを提供しています。セキュリティの観点から、例示された鍵は製品サポートには使用しないでください。

- OpenSSL と SKMT コマンドラインインターフェイスの使用
- NIST ベクタと SKMT GUI インタフェースの使用

2.5.1 OpenSSL と SKMT コマンドラインインターフェイスの使用

鍵とコード証明書を生成するには、いくつかのプロパティを Setting する必要があります。:

- 入力:`/loadaddr` アプリケーションのロードアドレス。
- 入力:`/oembl_size` OEM_BL のサイズ。
- 入力：`/ver` アプリケーションのバージョン。以前に FSBL を介して古いバージョンのアプリケーションから起動した MCU の場合、次のバージョンはより高いバージョンである必要があります。バージョン番号は、生成されるコード証明書に含まれます(図 12 参照)。この例では 3 に Setting されており、ARM_OEMBLn の値が 0、1、2 の場合に使用できます。
- 入力:`/cfsize` 使用するデバイスのコードフラッシュサイズ。
- 入力:`/oembl` アプリケーション バイナリ。
- 入力:`/oemroot_private` OEM_ROOT_SK。
- 入力:`/oembl_private` OEM_BL_ROOT_SK。
- 出力:`/output_codecert`、`/output_keycert` キー証明書とコード証明書。

コマンドウィンドウを開き、`\SecurityKeyMangementTool\cli` フォルダに移動し、以下のコマンドを実行して鍵とコード証明書を生成します。ファイルの場所は必要に応じて変更してください。

```
C:\Renesas\SecurityKeyMangementTool\cli>skmt.exe /gencert /mode "signature" /loadaddr "02000000"
/oembl_size "8000" /cfsize "2000000" /ver "3" /oembl
"C:\RA8_FSBL\fsp_v510\blinky_hmac_boot_ra8m1\Debug\blinky_hmac_boot_ra8m1.srec"
/oembl_private file="C:\RA8_FSBL\openssl_keys\oem_bl_private_key.pem" /oemroot_private
file="C:\RA8_FSBL\openssl_keys\oem_root_private_key.pem" /output_codecert
"C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_ccert.bin" /output_keycert
"C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_kcert.bin"

Output File: C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_kcert.bin

Output File: C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_ccert.bin
```

2.5.2 NIST ECC のキーペアと SKMT GUI インタフェースの使用

このセクションでは、証明書の生成を説明するために、セクション 2. 1. 2 で提供された NIST CAVP ECC secp25r1 の 2 セットのキーペアを使用します。キーとコード証明書を生成するために、キーペアのプライベートキー部分と公開キー部分の両方を SKMT に提供する必要があります。

SKMT GUI を起動します。**Overview** で、**RA Family**、**RIP-E51A Security Functions**、**Protected Mode を選択します。**

次に、セクション 2. 1 で生成された.srec ファイルを選択し、アプリケーションのバージョンを定義します。

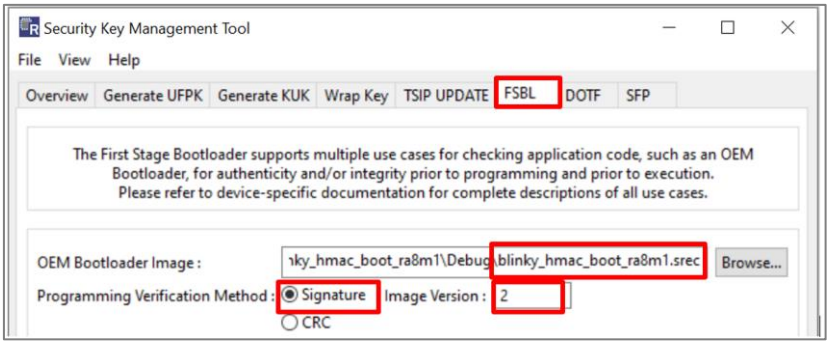


図 29. アプリケーションとバージョン番号の選択

図 29 のようにイメージバージョン定義は、コード証明書に保存されます。セクション 1.3.3 で説明したように、許可されるバージョン番号は 1～64 であり、更新イメージは現在プログラムされているイメージよりも新しいバージョンである必要があります。FSBL HMAC ベースのブートを初めて実行する場合、イメージバージョンは 1 に Setting できます。

FSBL ページで **OEM Root key** を Setting します。

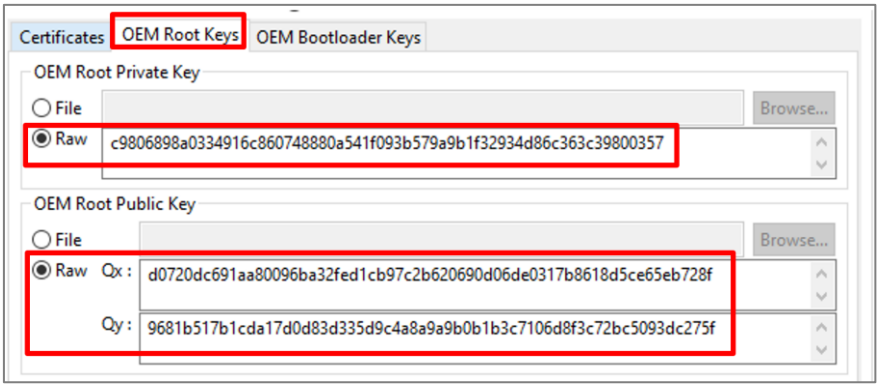


図 30. OEM ルートキーの提供

次に、**OEM ブートローダキー**を提供します。

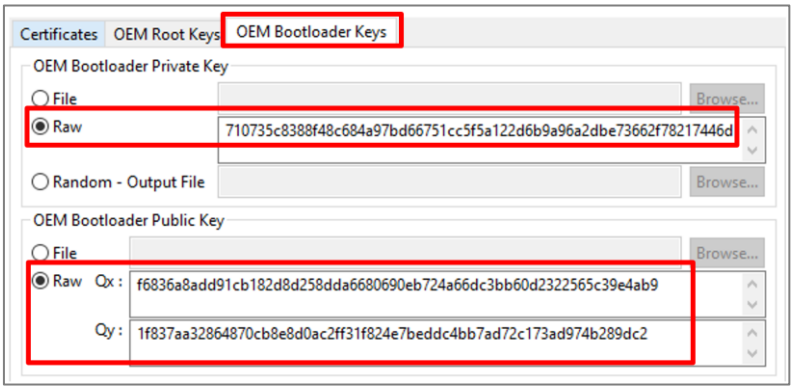


図 31. OEM ブートローダキーの提供

次に、**Certificates** タブに移動し、キー証明書とコード証明書の名前と場所を定義し、**Generate File(s)**をクリックします。キー証明書とコード証明書が生成されます。

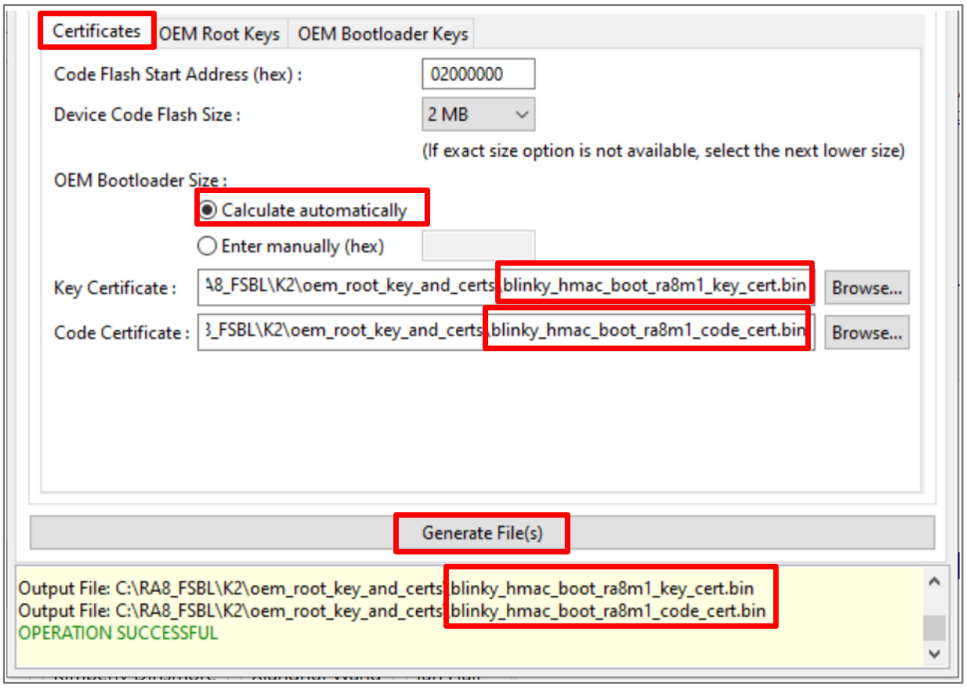


図 32. キーとコード証明書の生成

SKMT は内部的に NIST ベクトルも出力できます。これは、**Random – Output File** を選択することで可能です。

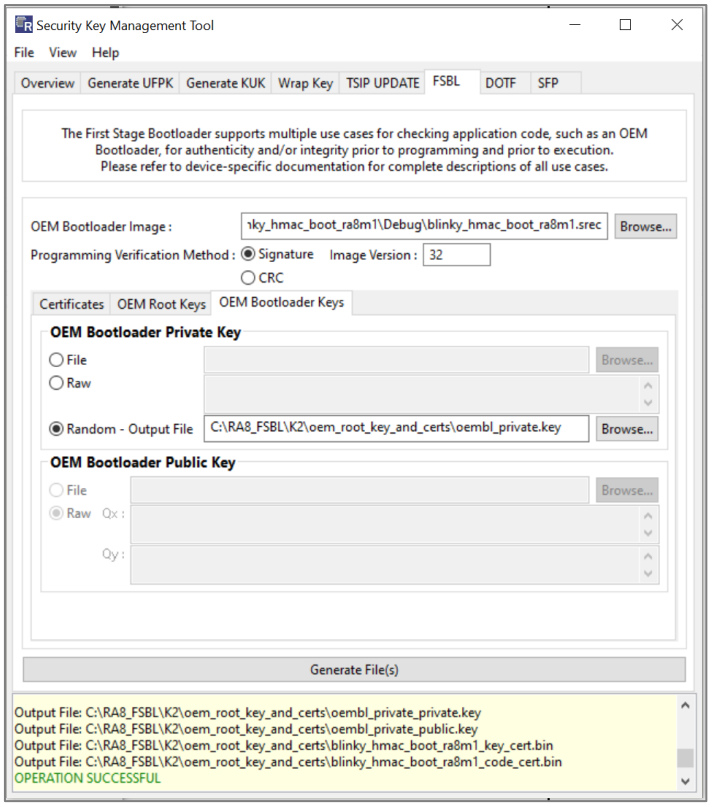


図 33. SKMT に保存された NIST テストベクトルの出力

図 32、図 33 で生成された証明書は、このアプリケーションプロジェクトで説明されている他の操作に使用できます。

SKMT v1.05 には、イメージバージョン 64 でコード証明書を生成できないバグがあることに注意してください。イメージバージョン 64 が要求されると以下のエラーが表示されます。

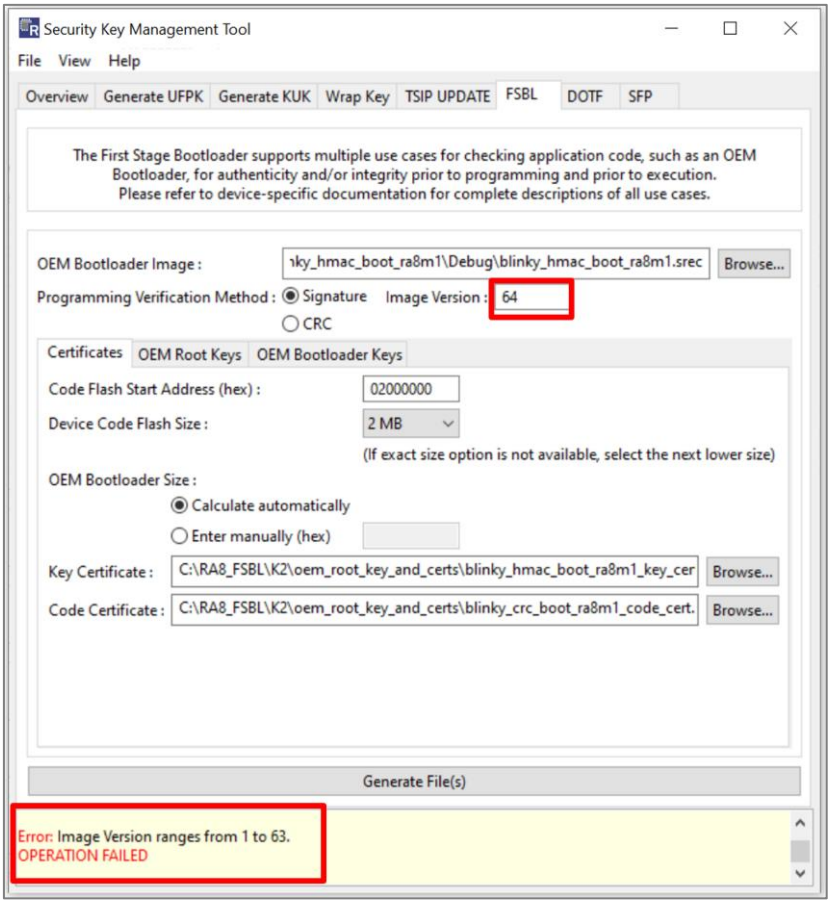


図 34. イメージバージョン 64 でコード証明書生成に関する SKMT v1.05 のバグ

2.6 認証付き製品プログラミングと HMAC ブートのデモンストレーション

rfp_ra8m1.zip をダウンロードし、その内容をローカルフォルダに解凍してください。次に、**RFP** を起動し、rfp_ra8m1.rpj プロジェクトを開きます。このサンプル RFP プロジェクトでは、SWD インタフェースがブートインタフェースとして選択されているため、USB ケーブルを使用してブートモードにアクセスできます。

以前のアプリケーションで FSBL 設定になっている場合や、TrustZone®境界が Setting されている場合は、次のセクションに進む前に**デバイスの初期化**(図 23 を参照)コマンドを実行する必要があります。これにより、前の FSBL 設定や TrustZone®設定が削除します。

次に、RFP を使用して MCU にアプリケーションをダウンロードし、アプリケーションを認証するための認証情報を提供します。フラッシュオプションのプログラミングを通じて、いくつかの認証情報が MCU にインジェクションされます(セクション 1.3.4 を参照)。以下の動作**設定**が Setting されていることを確認し、以前のアプリケーションを消去し、新しいアプリケーションをダウンロードして、フラッシュオプションをプログラムします。

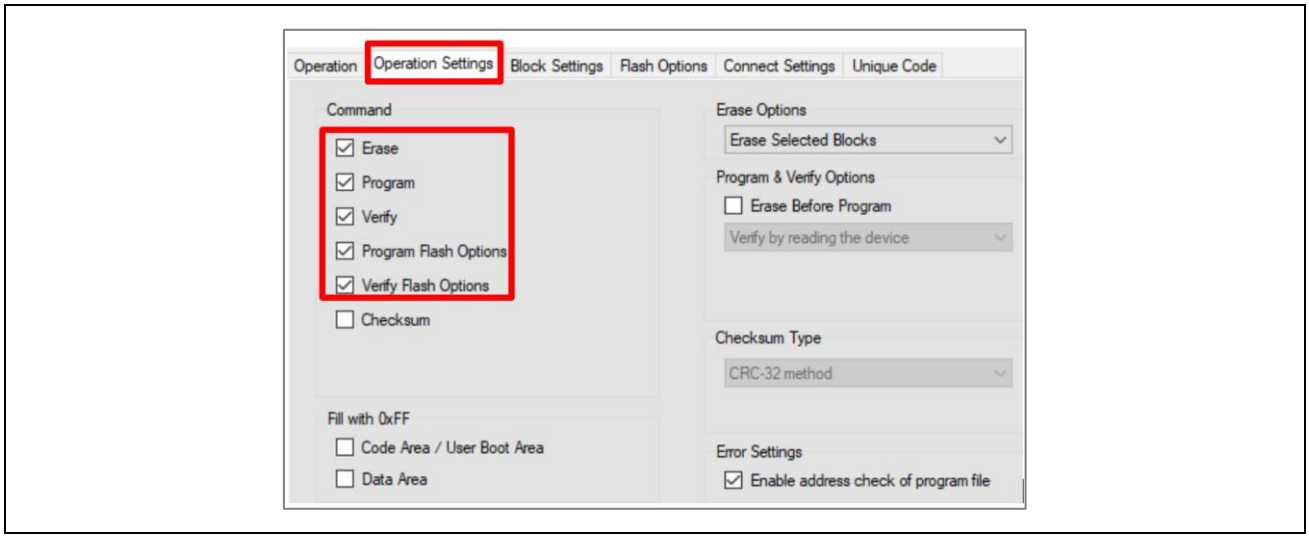


図 35. 動作設定の構成

MCU にプログラムするアプリケーションファイルを選択します。

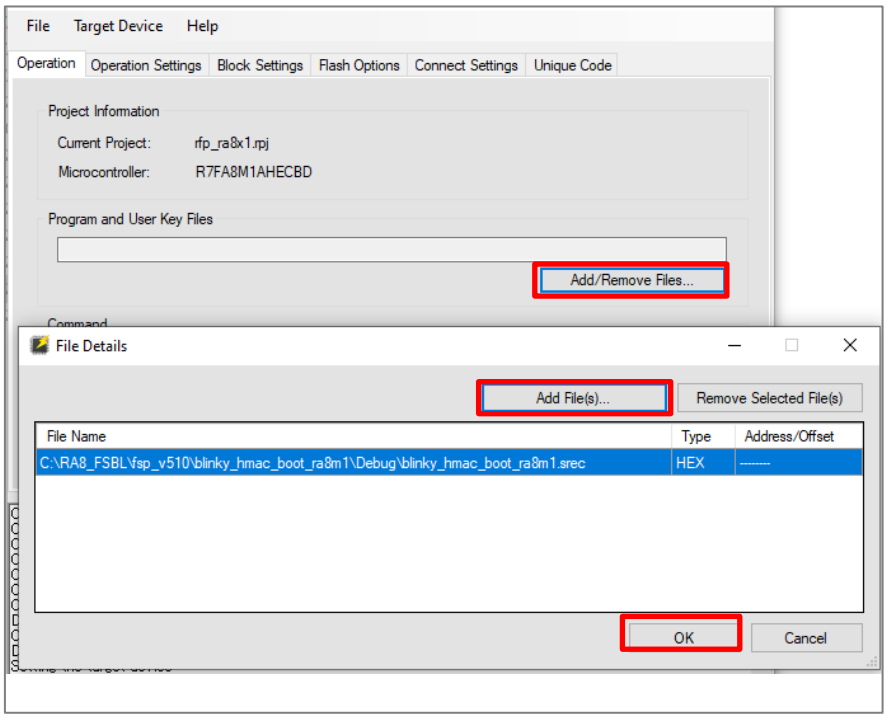


図 36. アプリケーションプロジェクトを選択

Flash Options 設定で **OEM Root Public Key**、**Code Certificate**、**Key Certificate** を Setting します。

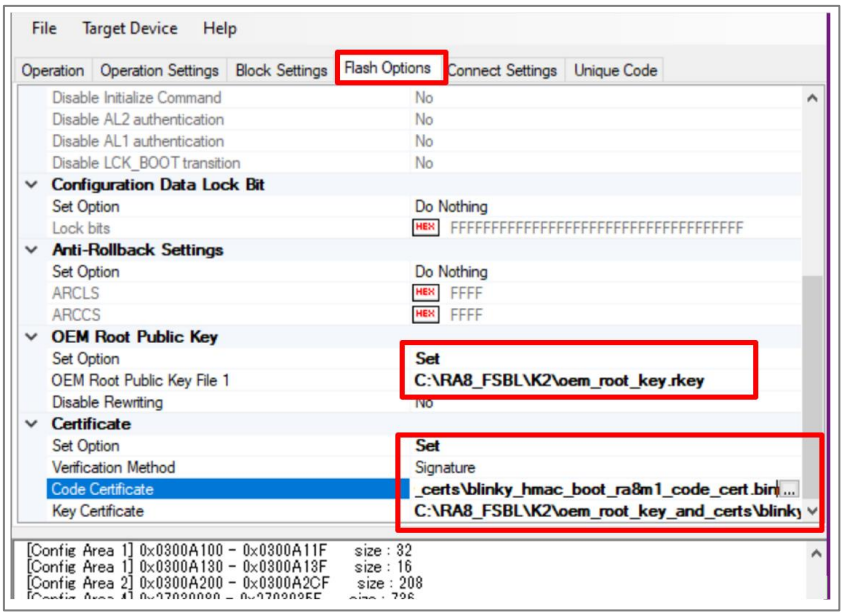


図 37. OEM ルートキーと証明書の Setting

ここで **Operation** タブに移動し、**Start** ボタンをクリックします。RFP は SREC から SACC0 レジスタ値を読み取り、指示されたメモリアドレスにコード証明書をプログラムします。FSBL はコード証明書のバージョン番号を比較し、バージョン番号の方が新しければ OEM_BL の認証を進めます。認証が成功すると、OEM HMAC ダイジェストがコード証明書の後の領域にプログラムされます。次の電源投入サイクルで、FSBL は保存されたダイジェストと FSBL が計算したダイジェストを使用してアプリケーションを認証し、比較が成功すると、新しいアプリケーションが起動し、3 つの LED が点滅するはずです。

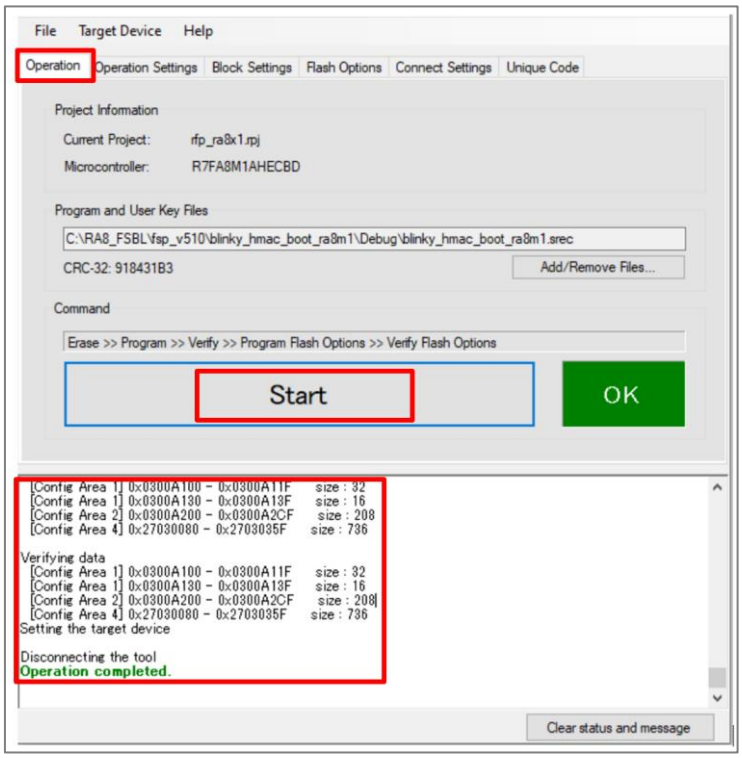


図 38. HMAC SHA2-256 検証に基づくアプリケーションの更新成功

起動に成功すると、J-Link コンソールは以下のような情報を出力します。`version_num_oem_bl` とイメージダイジェストは、キットの状態とプログラムされたアプリケーションによって異なります。

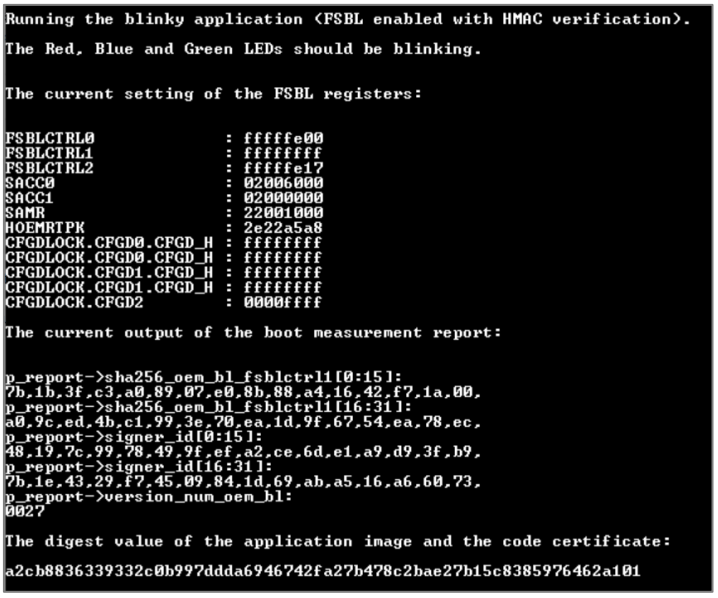


図 39. HMAC ブートの J-Link コンソール出力

HMAC ブートの失敗は容易に把握できます。まず図 23 のようにデバイスを初期化し、セクション 2. 6 の手順を実行します。ただし、間違ったアプリケーションイメージ、例えば CRC ブートコードのバイナリイメージや更新された HMAC ブートコードを指定することを除きます。いずれの場合も、図 38 の RFP プログラミングは失敗し、エラーレポートとして P107 ピンが”High”になり、赤色の LED が点灯します。

3. CRC チェックと CRC ブートデモンストレーション製品へプログラミング

CRC ブートを使用する場合、プログラミング中、MCU リセット後の通常実行中に CRC32 整合性チェックが実行されます。

3.1 CRC ブート FSBL を使用した blinky サンプルプロジェクト準備

blinky サンプルプロジェクト `blinky_crc_boot_ra8m1` は、図 40 のように FSBL Execution Mode を **CRC boot and report measurement** に更新することで作成されます。同じくエラーレポートとしてポート P107 ピンが使用されます。さらに、J-Link コンソールメッセージは CRC ブートモードを示すように更新されます。

`blinky_crc_boot_ra8m1` をワークスペースにインポートし、`configuration.xml` ファイルを実行して、**Generate Project Content** をクリックします。blinky プロジェクト(`blinky_crc_boot_ra8m1`)をビルドして実行し、その機能を確認します。

- コンパイル結果は、セクション 3. 2 でコード証明書の作成に使用されます。
- 3 つの LED が点滅しているはずです。

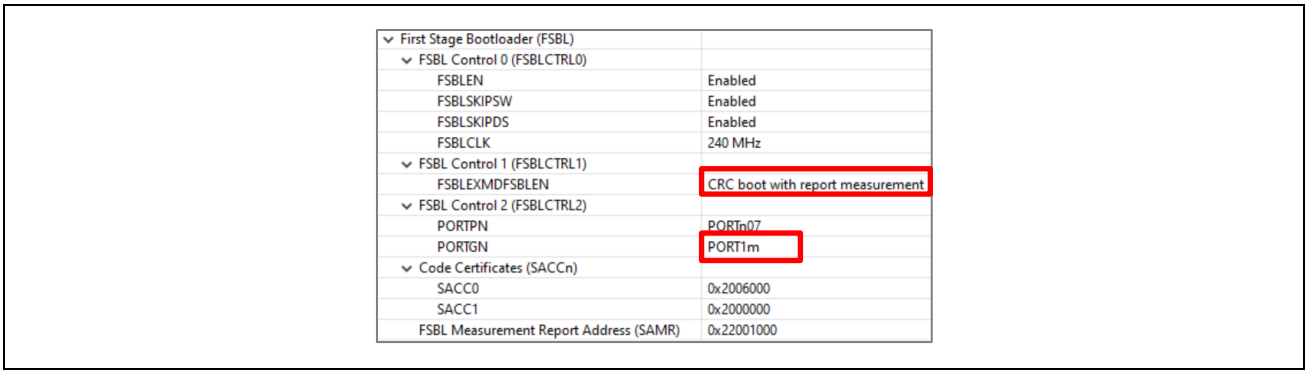


図 40. CRC ブート用の FSBL 設定

図 25 のように Tera Term を Setting し、次の Tera Term の出力を確認してください。コード証明書はまだプログラムされていないので、CRC の出力はすべて 0xFF になります。また、イメージバージョン情報は CRC ブートでは使用されないので、version_num_oem_bl の出力はすべて 0 です。

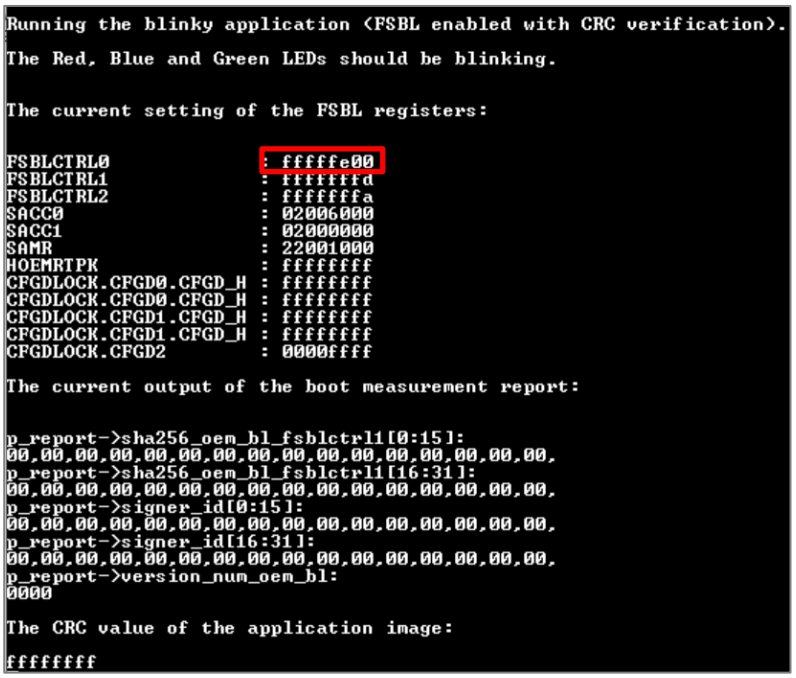


図 41. CRC ブートの RTT ビューア出力

3.2 コード証明書の作成

CRC ブート用のコード証明書には、OEM アプリケーションの期待値が CRC が含まれている必要があります。MCU リセット時、整合性チェックが実行されている場合、FSBL はアプリケーションの CRC を計算し、保存されているコード証明書の値と比較します。

SKMT CLI を使用した CRC ブート用のコード証明書の作成

コマンドラインを開き、SKMT\cli フォルダに移動し、以下のコマンドを使用して、セクション 3. 1 で作成したプロジェク トに基づいてコード証明書を生成します。

```
C:\Renesas\SecurityKeyMangementTool\cli>skmt.exe /gencert /mode "CRC" /loadaddr "02000000"
/oembl_size "8000" /cfsize "200000" /oembl
"C:\RA8_FSBL\fsp_v510\blinky_crc_boot_ra8m1\Debug\blinky_crc_boot_ra8m1.srec"
/output_codecert
"C:\RA8_FSBL\K2\certificates\blinky_crc_boot_ra8m1_code_cert.bin"

Output File: C:\RA8_FSBL\K2\certificates\blinky_crc_boot_ra8m1_code_cert.bin
```

SKMT GUI を使用した CRC ブート用コード証明書の作成

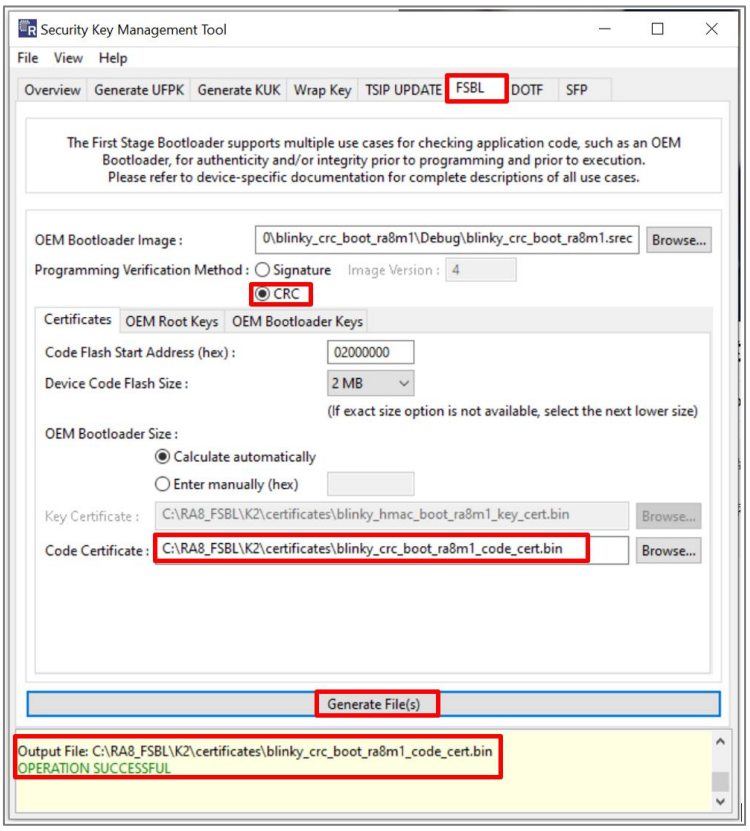


図 42. CRC ブートコード証明書の作成

CRC を指定してコード証明書を生成した場合、CRC 値はコード証明書の署名者 ID フィールドのダミー値として出力されます。CEC 検証のために作成される測定レポートでは Signer ID は CRC 値の 8 倍の長さで使用されます。CRC32 は 4 バイト(32 ビット)で、Signer ID(SHA256)は 32 バイトなので、CRC 値は Signer ID フィールドに 8 回書き込まれます。

3.3 RFP を使用したアプリケーションの CRC ブートの実証

RFP を起動し、付属の `rfp_ra8m1.rpj` を開きます。アプリケーションを MCU にダウンロードし、MCU の SWD ブートインタフェースを介してアプリケーションを認証し起動するための認証情報を提供します。以下の動作設定が構成されていることを確認します。

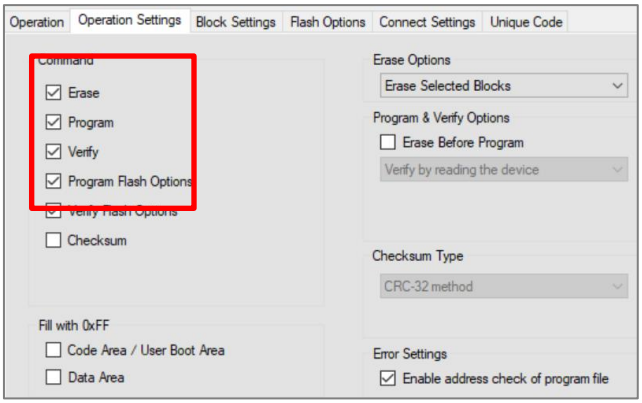


図 43. アプリケーションをダウンロードする前の動作設定と

MCU にプログラムするアプリケーションプロジェクトを選択します。

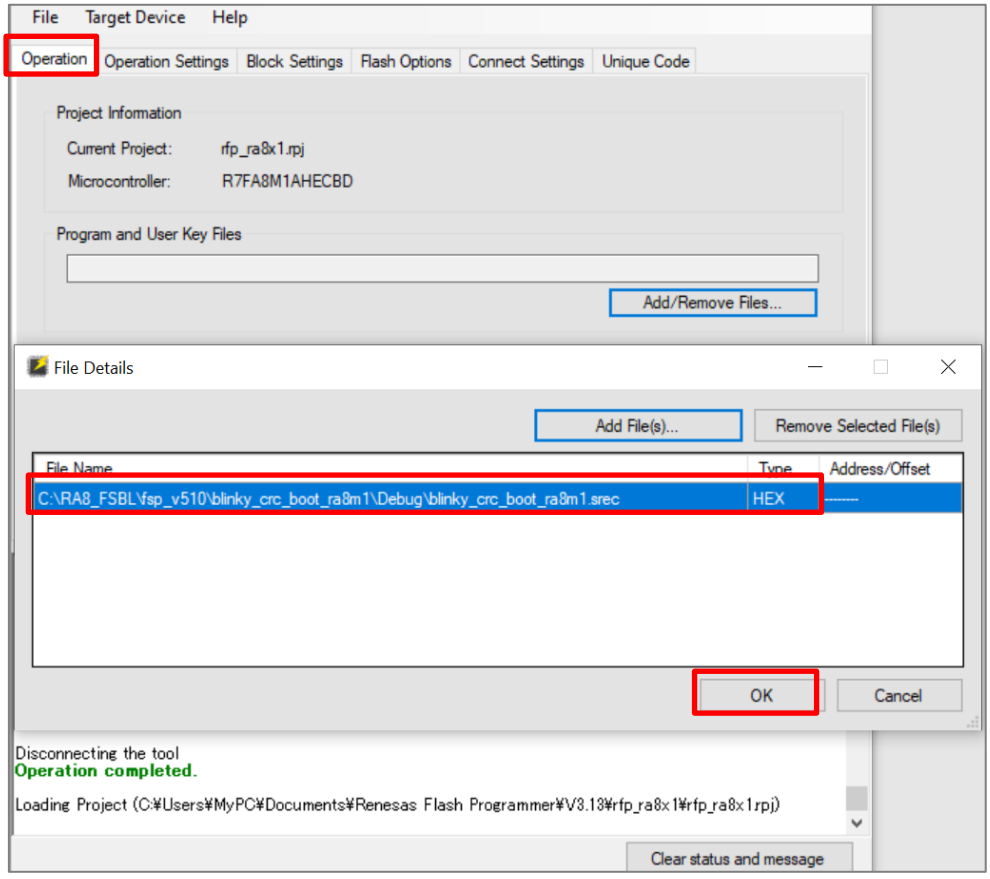


図 44. アプリケーションプロジェクトの選択

OK をクリックし、**Flash Options** ページに移動して **Code Certificate** を Setting します。

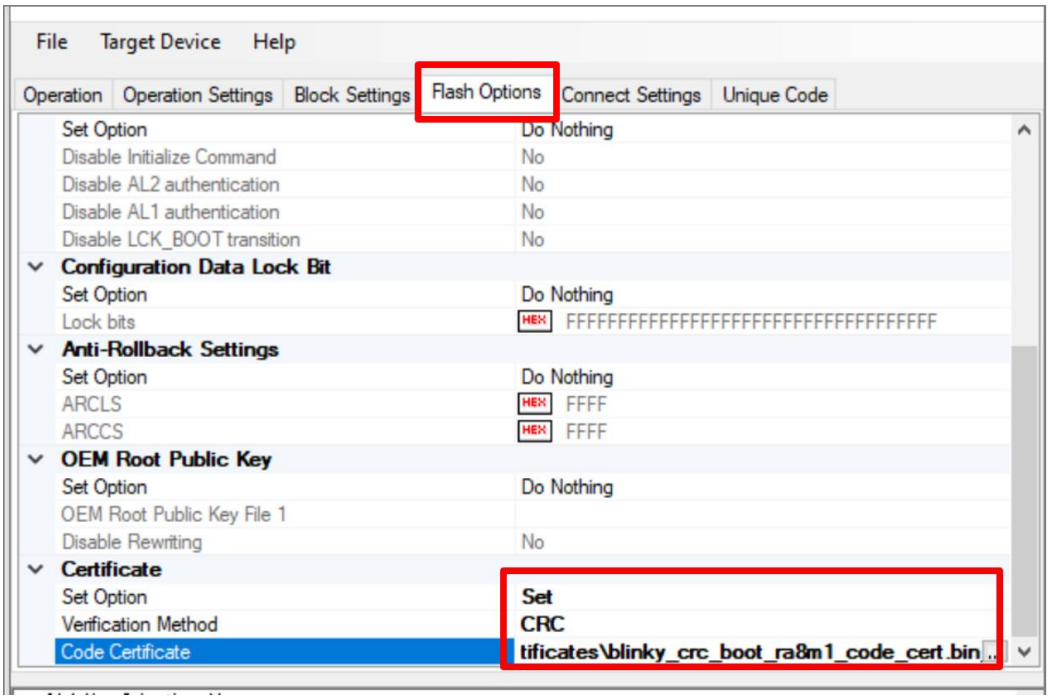


図 45. CRC コード証明書の Setting

次に、**Operation** タブに移動し、**Start** を選択すると、コード証明書と同様にアプリケーションがプログラミングされます。3 つの LED が点滅するはずです。

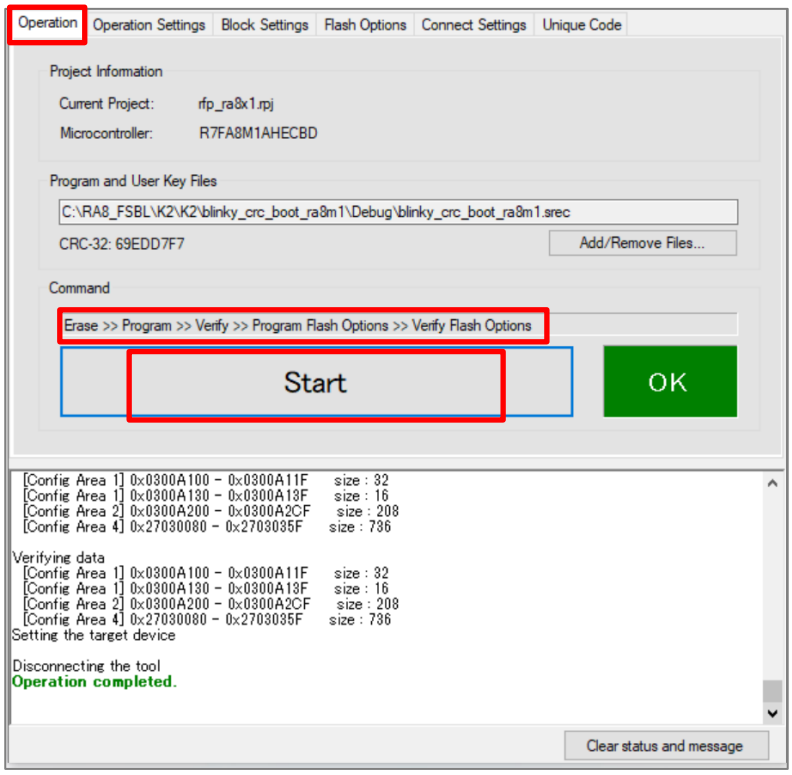


図 46. CRC ブートアプリケーションのダウンロード

J-Link のコンソールには、以下のようなメッセージが表示されます。CRC 値が `signer_id` として 4 回繰り返されていることに注意してください。

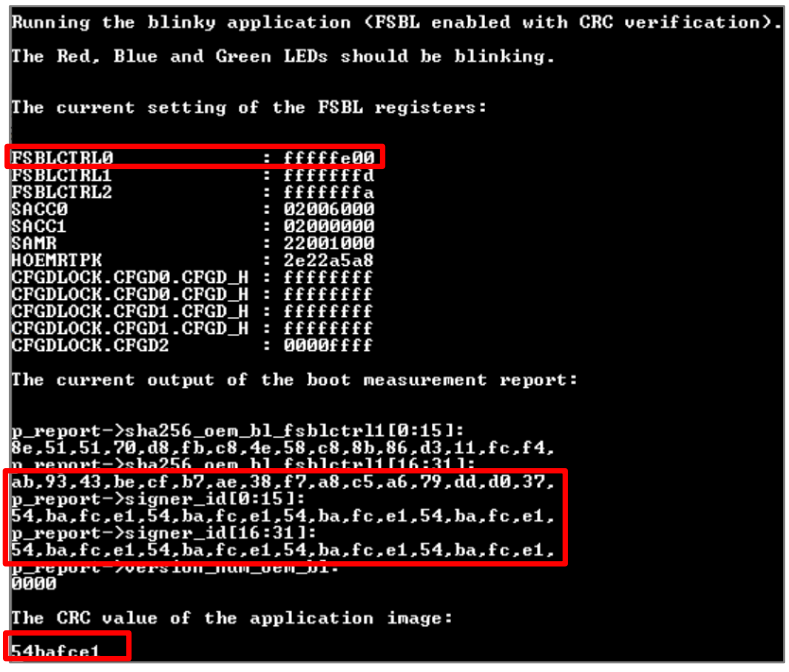


図 47. CRC ブートの成功

RFP から計算された CRC32 は、SKMT から計算された CRC32 とは異なることに注意してください。RFP はファイル全体の CRC32 を計算します。例えば、.srec ファイルの場合、すべてのレコードフィールドが CRC32 の計算に含まれます。しかし、SKMT の場合、CRC32 の計算には OEM_BL のバイナリ値のみが含まれます。

CRC ブートの失敗は容易に把握できます。図 23 のようにデバイスを初期化し、セクション 3.3 の手順を実行します。ただし、間違ったバイナリイメージを提供する場合は除きます。例えば、HMAC ブートコードのバイナリイメージを提供するか、CRC ブートコードのマイナーアップデートを実行します。いずれの場合も、図 46 の RFP プログラミングは失敗し、ポート P107 ピンは”High”になり、赤色の LED が点灯します。

4. 付録

4.1 TrustZone®プロジェクトでの使用上の注意点

Trustzone®プロジェクトのセットが以下のとおりであると仮定します。:

- セキュアアプリケーション用 `blinky_s`
- 一致する非セキュアアプリケーション用 `blinky_ns`

FSBL を HMAC ブートの TrustZone®プロジェクトのセットで使用する場合、ECC キーペアと OEM ルートキーの生成は、セクション 2.1、2.2 と同じ方法で実行できます。セクション 2.4 で説明された同じスクリプトを使用して、ARC_OEMBL レジスタの値を取得することもできます。

セクション 2.3、2.5、2.6 の操作は、TrustZone®プロジェクトで FSBL を使用するように調整する必要があります。IDE から生成された.rdp ファイルには、セキュアプロジェクトのフラッシュと SRAM の使用サイドに基づいた TrustZon®境界が含まれており、TrustZone®境界の Setting に使用されます。

- セクション 2.3 では、FSBL を `blinky_s` プロジェクトで有効にする必要があります。`SACCx` が `blinky_s` と `blinky_ns` が使用するフラッシュ領域の外側にあることを確認してください。e² studio を使用する場合、以下の選択が有効になっていると、IDE は生成された `blinky_s.rdp` ファイルを取得し TrustZone®境界をプログラムします。

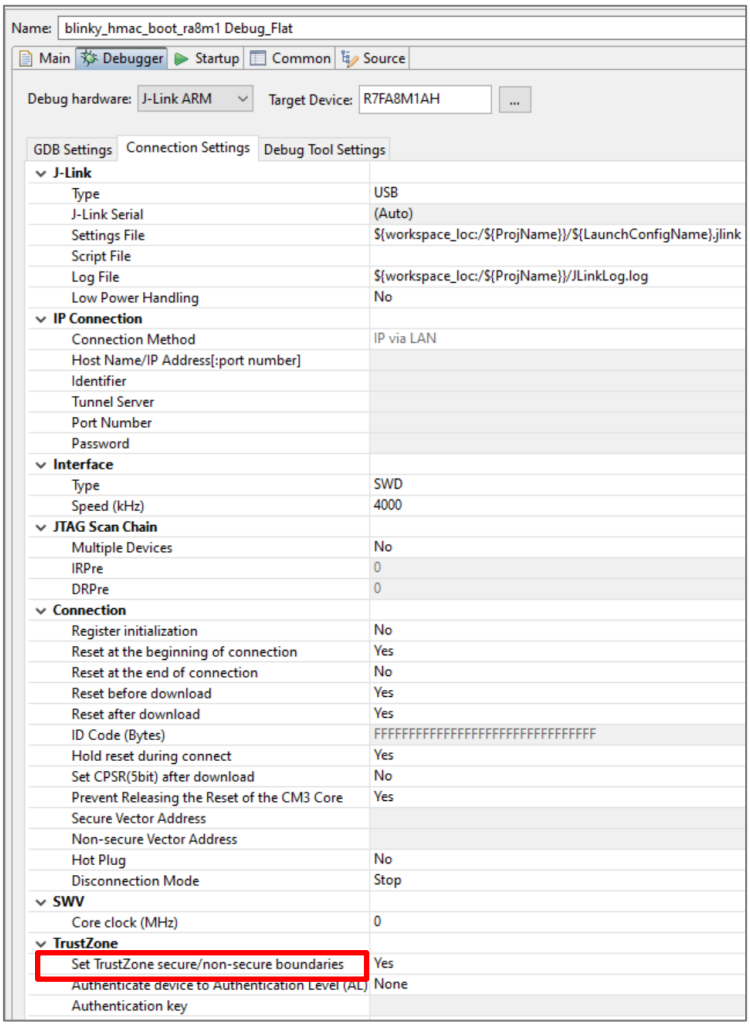


図 48. e² studio で TrustZone®境界プログラミングの有効化

TrustZone®ベースのアプリケーション開発に Keil と IAR を使用する場合は、RDPM を手動で起動する必要があります。図 45 のように、RDPM は生成された.rdp ファイルを使用して TrustZone®境界をプログラムするように Setting します。

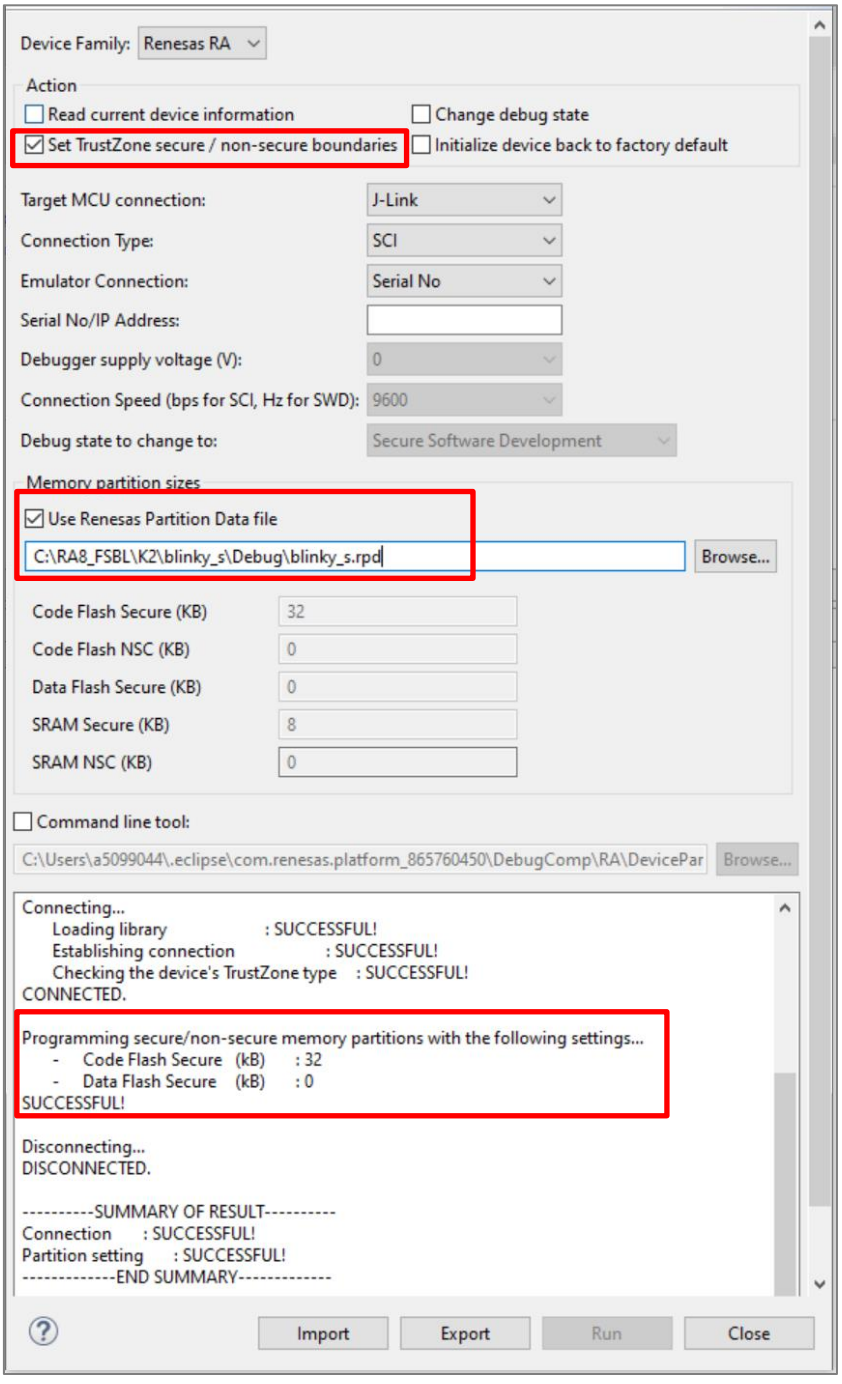


図 49. RDPM を使用した TrustZone®境界の Setting

- セクション 2.5 では、キー証明書とコード証明書の生成に `blinky_s` のコンパイル済みバイナリを使用します。非セキュアアプリケーションは、証明書の生成には使用されません。
- セクション 2.6 で、セキュアアプリケーションと非セキュアアプリケーションをプログラムする必要があります。また TrustZone®境界は、セキュアプロジェクトから生成された `blinky_s.rdp` ファイルを用いて Setting します。従って、図 36 のようにアプリケーションを提供することだけでなく、セクション 2.6 で記載されているように他のすべての Setting に加えて、図 50 のように RFP フラッシュオプションで `blinky_s.rdp` ファイルを選択して TrustZone®境界を Setting する必要があります。

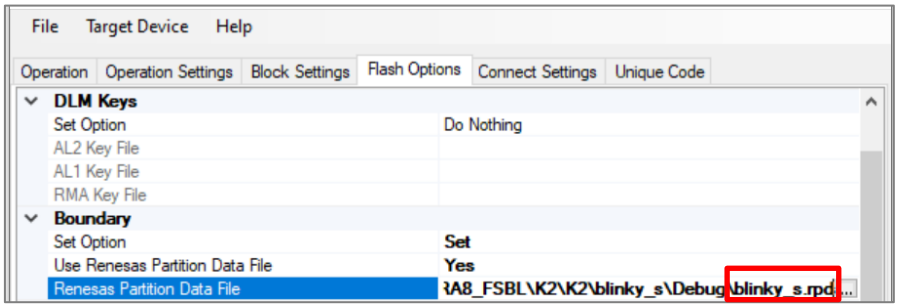


図 50. RFP を使用した TrustZone®境界の Setting

FSBL を CRC ブートの TrustZone®プロジェクトのセットで使用する場合、コード証明書のみが生成されることを除き、HMAC ブートと同様の方法でセクション 3.1、3.2、3.3 の操作を調整する必要があります。

TrustZone®プロジェクトの一般的な使用方法については、アプリケーションプロジェクトの、RA8 MCU クイックデザインガイド R01AN7087、TrustZone®を使用したセキュリティデザイン R11AN01467 を参照してください。

4.2 RFP 使用時のエラーのデバッグ

RFP 操作から出力されるエラーコードを理解しておくには、RFP ユーザーズマニュアルを参照してください。ドキュメント情報は参考資料のセクションに記載されています。参考までに、以下のエラーコードは、イメージバージョン番号が正しく Setting されていないことを示しています。

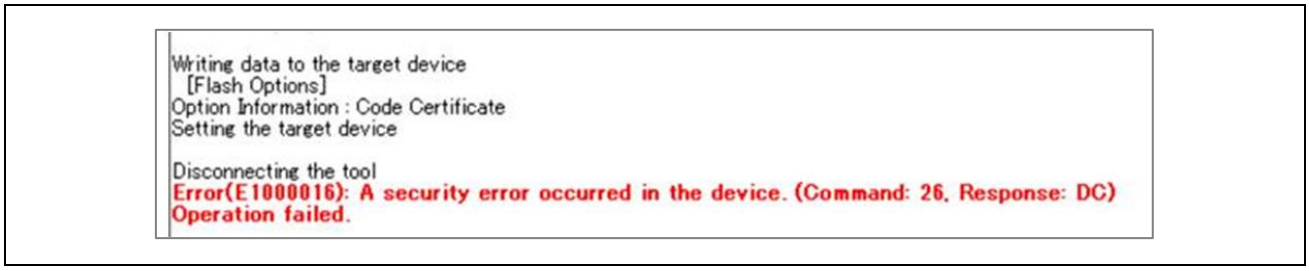


図 51. イメージバージョン番号のエラー

5. References

1. Rebesas RA Family Injecting and Updating Secure User Keys (R11AN0496)
2. RA8M1 グループ　ユーザーズ マニュアル　ハードウェア編(R01UH0994)
3. Renesas Secure Key Management Tool ユーザーズマニュアル(R20UT5349)
4. Renesas Flash Programmer Flash memory programming software ユーザーズマニュアル(R20UT5352)
5. Renesas RA Security Design using TrustZone with IP Protection (R11AN0467)
6. Renesas RA8 MCU Quick Design Guide (R01AN7087)

6. ウェブサイトとサポート

RA ファミリのマイクロコントローラの詳細、ツール、ドキュメントのダウンロード、サポートについては、以下の URL をご覧ください。

EK-RA8M1 リソース	renesas.com/ra/ek-ra8m1
RA 製品情報	renesas.com/ra
フレキシブルソフトウェアパッケージ(FSP)	renesas.com/ra/fsp
RA 製品サポートフォーラム	renesas.com/ra/forum
ルネサスサポート	renesas.com/support

改訂履歴

Rev.	日付	説明	
		ページ	概要
1.00	2024 年 11 月 13 日	–	初版バージョン

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する使用上の注意事項について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流インジェクションにより、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に電源オフ時における入力信号についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、未使用端子の処理に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を標準水準および高品質水準に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（脆弱性問題といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の半導体デバイスの使用上の一般的な注意事項等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、外国為替及び外国貿易法その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている当社とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている当社製品とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2024.10)

本社

豊洲フォレシア、豊洲 3-2-24、
135-0061 東京都江東区豊洲 3-2-24 豊洲フォレシア TOYOSU FORESIA, 3-2-24

www.renesas.com

商標

お問い合わせ先

製品、技術、ドキュメントの最新版、最寄りの営業所などに関する詳しい情報は、こちらをご覧ください。

www.renesas.com/contact/