Renesas Synergy™ プラットフォーム

セキュアブートマネージャ

R12AN0093JU0102
Rev1.02
2019.10.10

(注1) 本資料は英語版を翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は、参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントを参照ください。

要旨(Introduction)

このドキュメントでは、セキュアブートローダ(Secure Bootloader)のインストール、ユーザアプリケーションのダウンロード、展開済みアプリケーションの更新も含め、Synergy Boot Manager (ブートマネージャ) ソリューションを評価する方法を説明します。このインストレーションには、既存の Renesas ファクトリシリアルブートモード(factory serial boot mode)を使用してブートローダをインストールするほか、ファームウェアをマスタリング(Mastering:デジタル署名をすること)するツールや、ブートローダ、証明書、鍵、ユーザアプリケーションをダウンロードするツールも含まれます。プロセスの最後では、セキュアブートローダにアクセスするためのフラッシュアクセスウィンドウ(flash access window: FAW)をロックし、それ以上の変更を不可能にします。その結果、ブートローダは確実に、不変の信頼の基点(root of trust)になり、信頼されるファームウェアのみをブートするようになります。FSPR ビットをセットする (1 に設定する) ことで、デバイスの耐用期間にわたりブートローダをロックすることもできますが、ここではこのデモのために、このビットはセットしません。これにより開発ボードのリセットとフラッシュ再書き込みができるため、このデモを複数回実行したり、開発ボードを他の用途で使用したりすることができます。また、鍵などの機密データや、それらのデータにアクセスするコードなどを独立したメモリセグメントに分離するために、セキュリティ MPU(Security MPU)を設定することもできます。一部の鍵はラッピング(wrapped)され、保存され、MPU により保護されます。

セキュアブートマネージャのアーキテクチャに関する説明は、『Synergy Secure Boot Manager Architecture』(Synergy セキュアブートマネージャアーキテクチャ) ドキュメントを参照してください。

対象デバイス

PK–S5D9

インストールに必要なもの:

1. 2 本 x USB Type-A から Micro-B への変換ケーブル、ブートインタフェースと電源接続用。
2. 2 個 x USB から RS232 シリアル TTL への変換ケーブル。

目次

1. システム要求（System Requirements）

- PC 条件 :Microsoft® Windows® 10、2.0 GHz 以上で動作する Intel® Core™ ファミリプロセッサ (または同等のプロセッサ)、8 GB のメモリ、1TB ハードディスクまたは SSD、4 個の USB 2.0 空きポート。
- Renesas e² studio 統合ソリューション開発環境 (ISDE)。
- Renesas Synergy™ ソフトウェアパッケージ (SSP) v1.4.0。
- VS2008 ランタイムのインストール：Windows Desktop Utilities を実行するために、Microsoft Visual C++ 2008 再頒布可能パッケージ（redistributable）のインストールが必要となることがあります。fa
  これらのユーティリティがない場合、Microsoft 32 ビット (x86) および 64 ビット (x64) Windows パッケージのサイトから、これらのパッケージをダウンロードできます。
  https://www.microsoft.com/en-us/download/details.aspx?id=29
  https://www.microsoft.com/en-us/download/details.aspx?id=15336
- プログラミングインタフェース J5 に接続する前に、https://www.renesas.com/en-eu/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html#downloads から、Renesas Flash Programmer V3 をダウンロードし、インストールします。このツールは、USB デバイスインタフェースを経由してマイクロコントローラ内部のフラッシュメモリにプログラムを直接ロードするために必要なドライバをインストールします。
- Gearmo USB-TTL ケーブル：https://www.gearmo.com/shop/usb-to-3-3v-ttl-pin-header-cable-gm-ttl3vt/。

2. 評価ソフトウェアの解凍（Extract Evaluation Software）

評価ソフトウェアは、SecureBootManager v0.9.zip で提供されています。
zip ファイルの内容を **C:\Renesas\Synergy** に展開します。ここで、「SecureBootManager」フォルダのパスを、**C:\Renesas\Synergy\SecureBootManager** にする必要があります。

注記：

- この説明で使用する各バイナリは、以下のオプションを有効にした状態でコンパイル済みです。
  - Generate Device Identity Certificate (デバイス ID 証明書を生成する)
  - Encrypt the update before sending to MCU (MCU に送信する前に更新を暗号化する)
  - Support programming an app that uses QSPI as part of its application (アプリケーション自体の一部として QSPI を使用するアプリケーションのプログラミングをサポートする)
  - Use QSPI as the Update Area for new images (新しいイメージの更新領域として QSPI を使用する)
- セキュアブートマネージャは、セキュアブートローダ、1st Receiver（「1 次受信部」アプリケーション）と複数のサポートモジュール、および PC ツール（PC Tools）で構成されています。
- セキュアブートローダを、単に「ブートローダ」と呼ぶこともあります。
- 以前の用語では、セキュアブートローダのことを「SecurityKernel」 (セキュリティカーネル) と呼んでいました。この古い呼び名はコードやファイルの数か所に残っている可能性もありますが、現在はこの呼び方は推奨されていません。
- 本書では、ジャンパ J1 の位置を 2-3 に(MD 端子を L に)設定することにより起動される シリアルプログラミングモード（SCI/USB ブートモード） をファクトリブートモード と呼ぶことがあります。

3. PK-S5D9 ボードのセットアップ（PK-S5D9 Board Setup）

この章では、ボードに接続する必要のある複数の端子（pin）と、それらの機能について説明します。

セキュアブートマネージャを使用するには、2 種類の個別セットアップが必要です。最初のセットアップは、セキュアブートローダ (SBL) と 1st Receiver のプログラミング (書き込み) を行うときに使用します。2 番目のセットアップは、ユーザアプリケーションのプログラミングを行うときに必要です。以下の表で、接続をまとめます。必要な接続の図示については、その後の図を参照してください。

PK-S5D9			接続先	用途
PK-S5D9 ボードの ラベル表示	**S5D9 の端子**	**機能名**		
J5		ファクトリブートローダの USB-CDC	PC の USB ポート	SBM/1st Receiver のプログラミング
P32	P302	SCI2-TX	USB-TTL コンバータ端子 – RX	ユーザアプリケーションプログラムのプログラミング
P31	P301	SCI2-RX	USB-TTL コンバータ端子 – TX	
P15	P105	SCI8-TX	USB-TTL コンバータ端子 – RX	コンソールのデバッグ出力
P14	P104	SCI8-RX	USB-TTL コンバータ端子 – TX	
J19		ボードの電源	PC の USB ポート	ボードへの電力供給

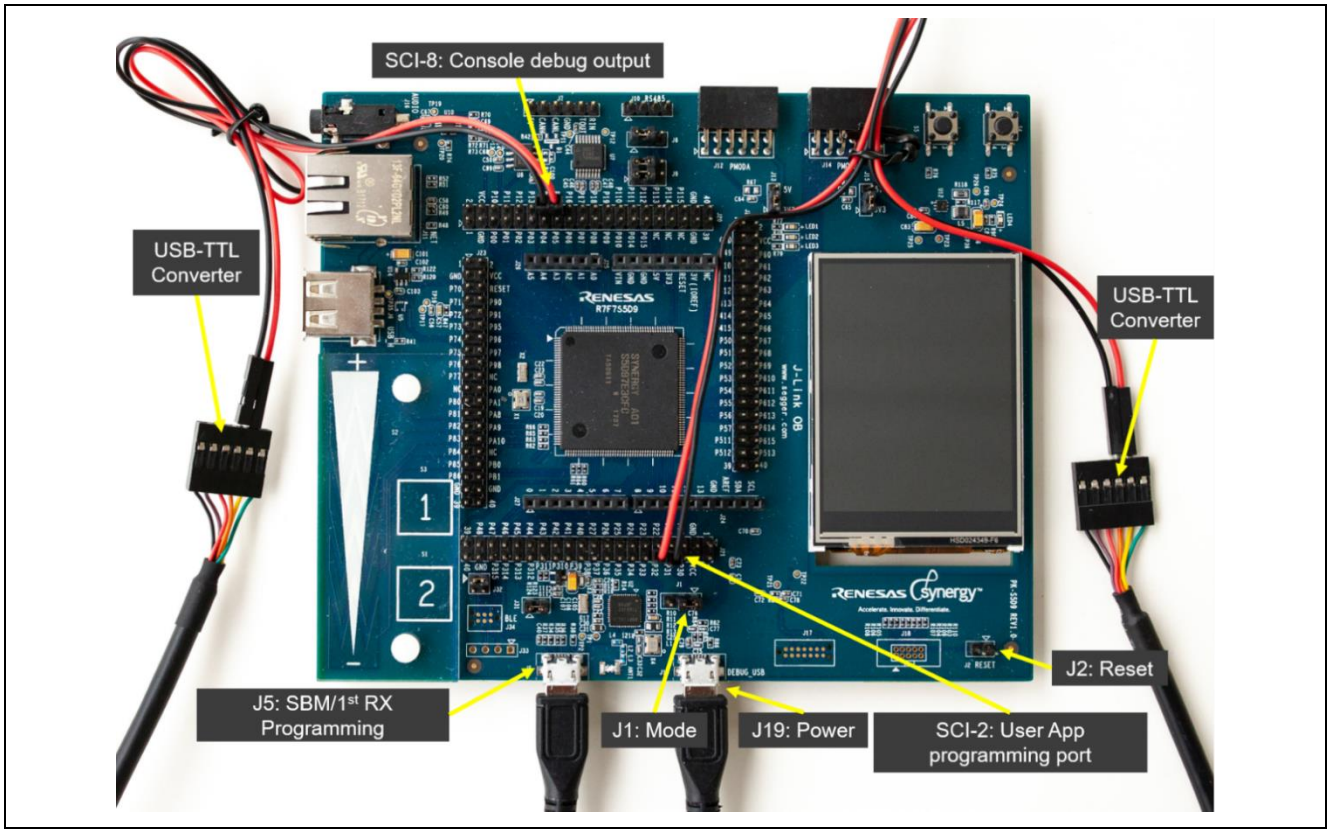


図 1. 配線図

ここでは Gearmo USB-TTL 変換ケーブルを使用していますが、代わりに他の USB-TTL 変換ケーブルも使用できます。

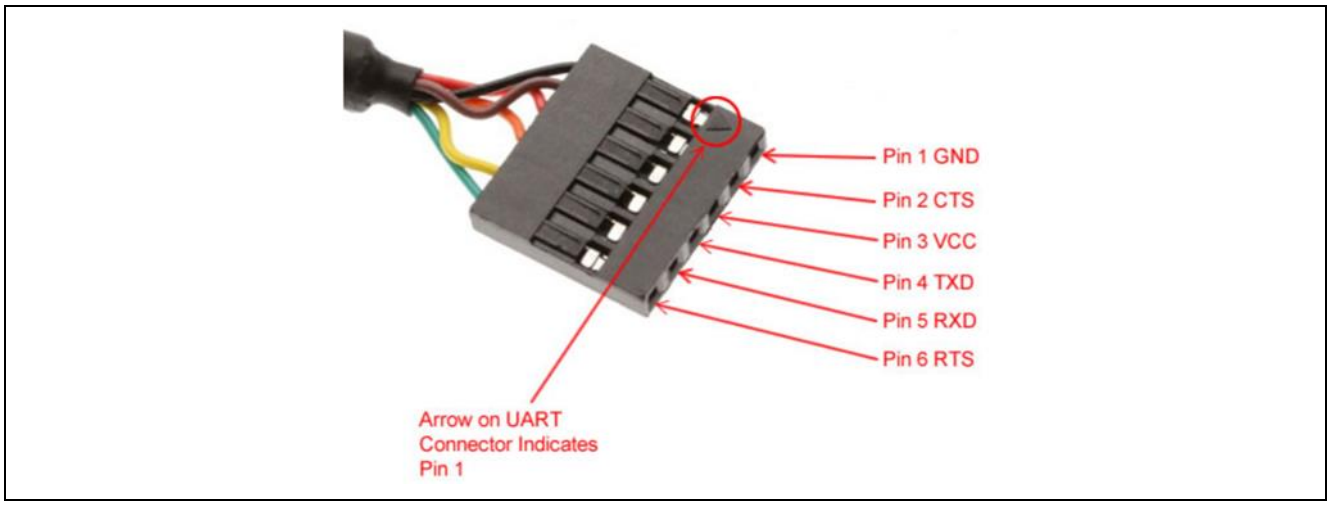


図 2. TTL コネクタのピン配置

4. セキュアブートローダと 1st Receiver のプログラミング（Programming the Secure Bootloader and 1st Receiver）

4.1 PK-S5D9 開発ボードの設定（Configuring the PK-S5D9 Development Board）

PK-S5D9 開発ボードを設定するには、以下の手順に従います。

- USB - TTL 変換ケーブルを、図 1 で示すようにボードの SCI-8 に接続します。USB コネクタを PC に接続します。このコネクタは、コンソールログに情報を表示する目的で使用します。**この時点では、2 番目の USB-TTL 変換ケーブルはまだ PC やボードに接続しないでください。**
- 電力供給の目的で、J19 コネクタと PC の間に Micro USB ケーブルを接続します。
- プログラミングインタフェースとして、J5 コネクタと PC の間に Micro USB ケーブルを接続します。
- ジャンパ J1 の位置を、1-2 から、**ファクトリブートローダモード**（FACTORY BOOT LOADER MODE）を意味する 2-3 に変更します。

  注記：
  - **通常ブート**（NORMAL BOOT）：J1 が **1-2 の位置** にある場合、S5D9 は内部フラッシュ (ROM) から実行を開始します。これは、初期のフラッシュ書き込みを行った後の通常動作モードです。セキュアブートマネージャのフラッシュ書き込みを行った後は、元の位置に切り替えます。
  - **ファクトリブートローダモード**（FACTORY BOOT LOADER MODE）：J1 が **2-3 の位置** にある場合、S5D9 デバイスは USB メモリプログラムモード内のファクトリブートローダを使用してブートします。その結果、USB デバイスインタフェース (CDC クラス) を経由して、プログラム (この場合はセキュアブートローダ) をマイクロコントローラの内部フラッシュに直接ロードすることができます。このモードは、デバイスの最初のプログラミング（書き込み）のみに使用します。また、これはシリアルプログラミングモード（SCI/USB ブートモード）の事です。

4.2 セキュアブートローダのフラッシュ書き込み（Flashing the Secure Bootloader）

セキュアブートローダと 1st Receiver は、ファクトリブートローダを使用して MCU にフラッシュ書き込みします。1st Receiver は、製造時の早い時点で、アプリケーションのファームウェアを MCU にダウンロードするために使用するコンパニオンアプリケーションです (この時点以降、このアプリケーションを削除することもできます)。1st Receiver は、UART を使用してダウンロードを実行するように設定してあります。

以下の手順を実行します。

1. 新しい一意の鍵（unique key）を複数使用して、鍵データベースを初期化します。これらの鍵は、以下の目的で使用します。
   A. ブートローダ署名鍵（Bootloader Signing key）は、ブートローダに自己署名（self-sign）する目的で使用します。公開鍵（public key）はセキュアブートローダが利用する管理テーブル内に収録されています。この鍵は現在、CRC と同様、ブートローダが破損していないことを保証するために使用しています。この設計を拡張し、公開鍵を個別に提供することもできます。その場合、セキュアブートローダの同一性（identity）も検証でき

ます。したがって、そのブートローダが OEM から出荷されたものであり、改ざんされていないことも保証されます。

B. マスタリング鍵（Mastering key）が生成され、マスタリングツール（Mastering tool）がファームウェアに署名する目的で使用されます。この公開鍵を使用して、アップロードされたファームウェアを検証します。この公開鍵は、ブートローダ内に収録されています。

2. セキュアブートローダと 1st Receiver は、ファクトリブートローダの「フレームプロトコル」(Frame Protocol) を使用して MCU にフラッシュ書き込みします。

セキュアブートローダと 1st Receiver をボードにインストール/フラッシュ書き込みするには、以下の手順を使用します。

- SCI-8 を PC に接続し、MCU をリセットするため、ジャンパを使用してボード上の J2 をショートし、その後解放します。
  - Windows の [Device Manager] を開き、**[Ports]** を開きます。**[Synergy USB Boot]** というデバイスが表示されています。このデバイスを参照し、どの番号のシリアルポート (COM) をエミュレーションに使用しているかを判定します。これは、J5 の USB コネクタに接続されている USB ケーブルを経由してセキュアブートローダをプログラムする (書き込む) ために使用するポートです。この例の場合、COM20 になっています。実際のポート番号は各ユーザの PC によって異なる可能性があります。

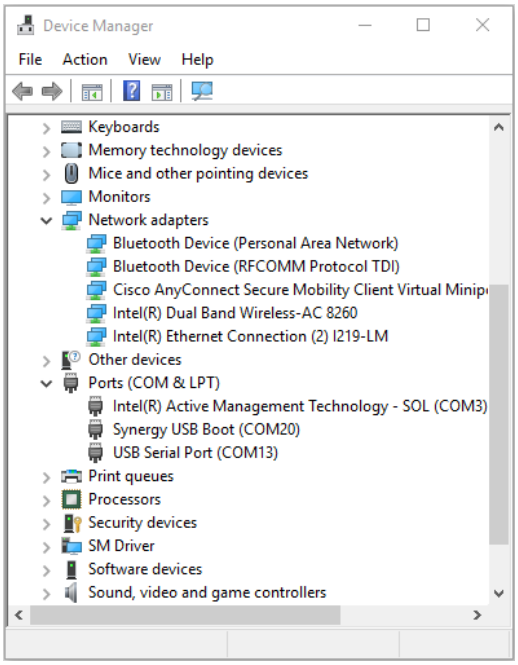


図 3. デバイスマネージャ

ほかにも USB シリアルポートデバイスが表示されます。この例の場合、COM13 です。これは、コンソールログ（console log）の目的で使用するポートです。このケーブルは青とオレンジの配線で識別できます。セキュリティブートローダの開発バージョンでは、ロギング機能（logging）が有効になっています。プログラムサイズ（code footprint）を縮小するために、ロギング機能をコード内で無効にすることもできます (10.2.2 章を参照)。

先へ進み、説明は「COM20」(Synergy USB ブート) および「COM13」(コンソールログ) という名前でポートを参照します。ただし、実際のポート番号はユーザの PC によって変わる可能性があり、説明を適宜読む必要があります。

- コマンドプロンプト（Command Prompt）を開き、**C:\Renesas\Synergy\SecureBootManager\evaluation\bin** に移動します。セキュアブートローダと 1st Receiver をボードに展開するために、「deploy_bootloader.bat <COMportNumber>」コマンドを使用します (例：deploy_bootloader.bat 20)。
- ウィンドウに、カーネルのインストール進行状況が表示されます。鍵の生成、ブートローダと 1st Receiver のパッケージ化、ブートローダと 1st Receiver の MCU へのプログラミング (書き込み) などです。

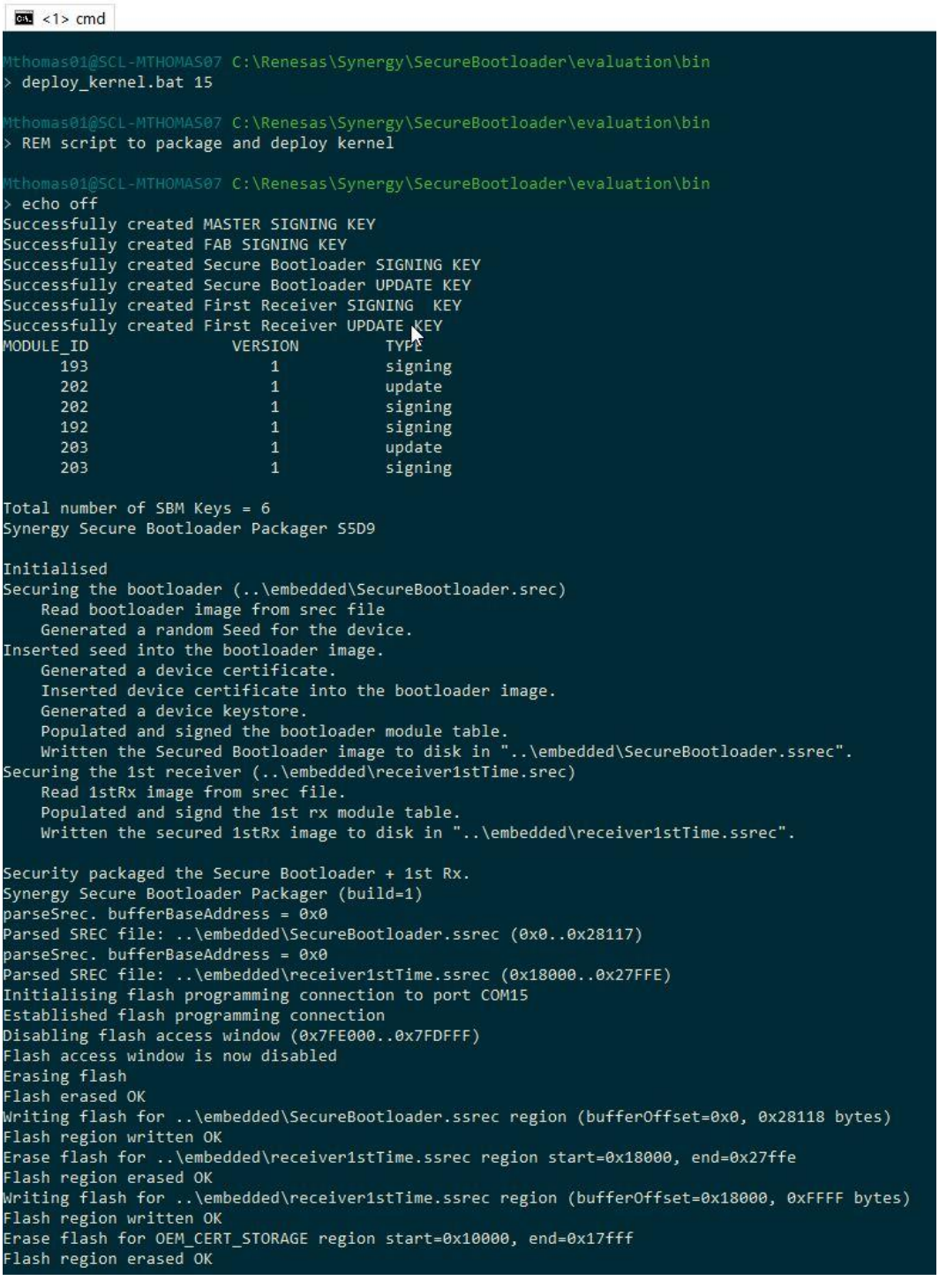


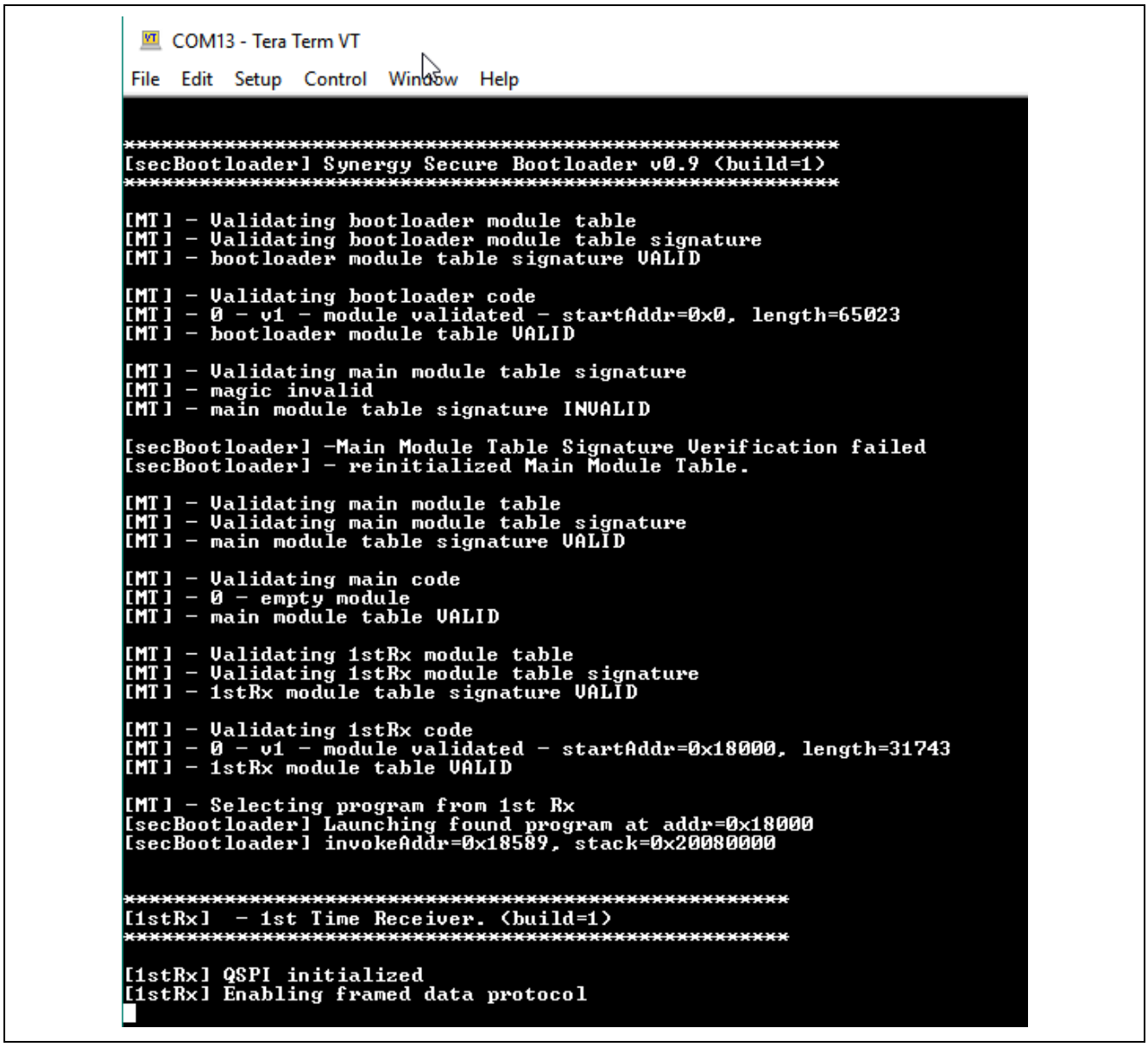
図 4. ブートローダ と 1st Receiver のプログラミング

- コマンドウィンドウに「**Flash region written OK**」(フラッシュ領域への書き込み正常に完了) と表示された時点で、セキュアブートローダと 1st Receiver のプログラミングが終了しています。上のスクリーンショットは OEM 証明書（OEM certificate）を使用している場合であり、その場合は証明書で使用する領域も消去されたことを表す行が表示されます。

インストールを検証するには、以下の手順を行ってください。

- ターミナルプログラム (TeraTerm、Putty、Hyperterminal など) をコンソールログポート (例：COM13) に接続し、**115200 baud 8N1** (115200 ボー、8 ビット、パリティなし、ストップビット 1 ビット) に設定します。このターミナルを使用して、セキュアブートローダからのデバッグ出力を表示します。
- ボード上で J1 の位置を 1-2 に変更し、**通常ブートモード（NORMAL BOOT MODE）**に切り替えます。

- ジャンパを使用し、ボード上の J2 をシュートし、MCU をリセットします。
- ターミナルプログラムで、以下のようなログオン情報が表示されます。
  - セキュアブートローダが有効（valid）。すなわち、有効な自己署名がなされ、破損や改ざんが発生していないことを示しています。
  - メインのモジュールテーブル（Main module table）が無効。これは、メインプログラム/アプリケーションがまだプログラミングされていないためです。
  - 1st Receiver が有効。
  - ブートローダは、唯一の有効なモジュールである 1st Receiver を実行します。このモジュールは、アプリケーションイメージが自分宛に送信されてくるのを待機している間に、「フレームデータプロトコル」(Framed Data Protocol) を開始します。フレームデータプロトコルは、ファクトリブートローダが使用するプロトコルの拡張バージョンです。



LED の点滅のしかたで、成功または失敗を示します。

- **成功（SUCCESS）**：緑色と黄色が点滅し、ブートローダのインストールに成功したことを示します。1st Receiver はユーザアプリケーションの更新を待っています。
- **失敗（FAILURE）**：赤は、ブートローダが起動対象を何も見つけられなかったこと、またはバス/アドレス/メモリいずれかの例外（exception）が発生したこと、またはアサート（assert）が発生したことを示します。

複数の LED が成功を示していても、上記のようなログが何も表示されない場合、USB-TTL 端子が正しく接続されていること、およびターミナルが正しく設定されていることを確認してください。

5. ユーザアプリケーションのマスタリングとプログラミング (セキュアな製造)（Mastering and Programming the User Application (Secure Manufacturing)）

この章では、署名済みバージョンのユーザアプリケーションを作成し、その後、ブートローダが署名済みユーザアプリケーションの署名を検証して MCU にセキュアにフラッシュ書き込みするために必要な一連のステップを説明します。

製品の製造時に、以下のプロセスを適用します。

1. セキュアマスタリングツール（Secure Mastering tool）を使用し、ユーザアプリケーションのバイナリ（binary）をパッケージ化して、署名済みの書式にします。事前生成済みのユーザアプリケーションを使用します。
2. セキュア製造ツール（Secure Manufacturing tool）は、1st Receiver を使用して署名済みバイナリを MCU の更新領域にフラッシュ書き込みます。1st Receiver はファクトリブートローダと同じプロトコル (フレームプロトコル) を使用しますが、拡張コマンドが存在しています。
3. このデバイスをリセットすると、セキュアブートローダが、更新領域内に新しいイメージが存在していることを検出します。セキュアブートローダは署名を検証し、その署名が有効な場合、アプリケーションをフラッシュ書き込みします。システムの再起動後に、そのアプリケーションが実行されます。
4. システムをブートするたびに、セキュアブートローダによってユーザアプリケーションの検証が行われます。

この一連の過程で、PC はマスタリングツールと製造ツールの両方のホストとして使用されます。以下の手順で起動する GUI は、単純にデモンストレーションの理解を容易にする目的で作成したものです。これらの GUI は、製造環境で使用することを目的として、コマンドラインから操作する複数のプログラムを起動します。実際の製造環境に合わせて、これらのプログラムの変更やカスタマイズを行うことも可能です。

5.1 PK-S5D9 開発ボードの設定（Configuring the PK-S5D9 Development Board）

5.1.1 接続（Connectivity）

本章では、3 章の「PK-S5D9 ボードのセットアップ（PK-S5D9 Board Setup）」で説明した接続の設定が、既に完了していることを想定します。

以下のことを確認してください。

- 2 番目の USB - RS232 シリアル TTL 変換ケーブルを、SCI-2 に接続します (図 1 を参照)。USB コネクタを PC に接続します。このコネクタを使用して、シリアル経由でアプリケーションをフラッシュ書き込みします。
- J5 コネクタから micro-USB ケーブルを取り外します。これは、ユーザアプリケーションをプログラムする目的でファクトリブートローダを使用していないためです。
- ジャンパ J1 が通常ブートモード (1-2 の位置) になっていることを確認します。ジャンパがこの状態になっているときに、前の章でプログラムしたセキュアブートローダを実行できます。
- ジャンパを使用し、ボード上の J2 をショートし、その後解放する方法で、MCU をリセットします。

Windows の [Device Manager] を開き、接続済みの USB -シリアル変換ケーブルが使用しているシリアルポートを確認します。

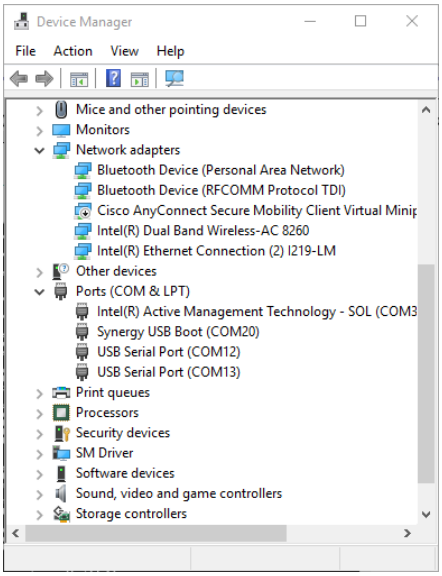


図 5. デバイスマネージャ内でのシリアルポートの特定

この例で、前述のコンソールログに使用しているシリアルポート COM13 に隣接して、さらにシリアルポート COM12 が表示されています。これは、アプリケーションをプログラムするのに使用するポートです。先へ進むと、説明は「COM13」および「COM12」という名前でポートを参照します。ただし、実際のポート番号は各ユーザの PC によって変わる可能性があるので、説明を適宜読み替える必要があります。

ユーザアプリケーションのフラッシュ書き込みにどのポートを使用するのか不明な場合、接続済みの USB ケーブルを取り外し、どちらの COM ポートがデバイスマネージャから消えるかを確認してください。そのポートが、アプリケーションのフラッシュ書き込みに使用するポートになります。

5.2 セキュアマスタリング：署名済みアプリケーションの準備（Preparation of Signed Application）

注記： 解像度の低いディスプレイを使用している場合、GUI アプリケーションの UI 全体が表示されない可能性があります。その場合、このドキュメントで説明している GUI 要素の一部が表示されません。その場合、Windows の [Control Pannel] で、[Display] などの設定を確認してください。**[Smaller - 100% (default)]** が選択されていることを確認してください。

C:\Renesas\Synergy\SecureBootManager\evaluation\gui\masteringManufacturingGUI.exe を実行します。以下のウィンドウが開きます。

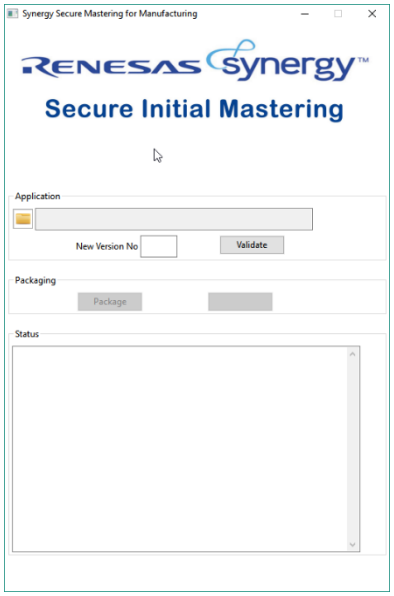


図 6. アプリケーションの初期のセキュアマスタリング用 GUI

1. アプリケーションフォルダをクリックし、以下の場所を参照します。**C:\Renesas\Synergy\SecureBootManager\evaluation\embedded**。次に、[app1.srec] を選択します。このファイルは、パッケージ化しようとするサンプルアプリケーションです。
2. [New version no] (新しいバージョン番号) を「1」に設定します。これは、インストールしようとするアプリケーションの初期バージョンを意味します。
3. **[Validate]** (検証) をクリックし、srec ファイルの解析が正しく行われること、および Key Database (鍵データベース) が新しいバージョン番号である 1 に整合していることを確認します。鍵データベースはファームウェアのバージョンごとに 1 つの鍵を維持しており、順に増加するバージョン番号のみが使用されていることを保証します。セキュアブートローダをインストールする時点で、鍵データベースはリセットされます。検証が成功した場合、**[Package]** (パッケージ化) ボタンが有効になります。
4. **[Package]** (パッケージ化) をクリックします。この操作により、アプリケーションのバイナリに署名し、そのバイナリを **app1_YYYY_MM_DD_HH_mm_ss.ssrec** (独自のバイナリ形式) という名前で、入力 srec ファイルが存在していたのと同じフォルダに書き込まれます。
5. **[Complete]** (完了) ボタンが緑色に変化した後、このボタンをクリックします。個別のウィンドウ内にログファイルが表示されます。
6. マスタリング GUI（mastering GUI）を閉じます。

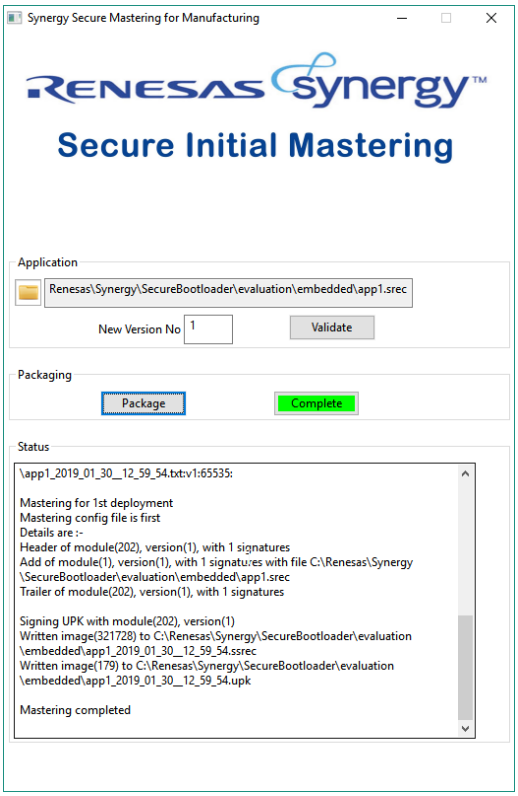


図 7. アプリケーションの初期のマスタリングが成功

5.3　署名済みアプリケーションのフラッシュ書き込み（Flashing Signed Application）

以下の画像で図示している
C:\Renesas\SynergySecureBootloader\evaluation\gui\deployManufacturingGUI.exe を実行します。以下のステップを実施します。

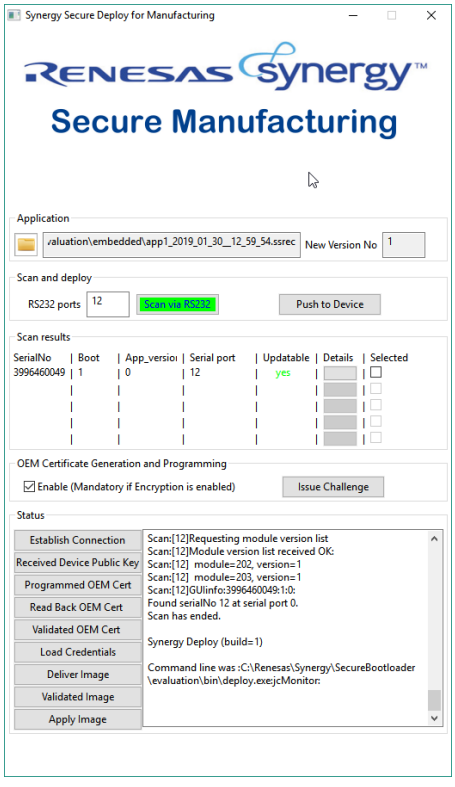


図 8. セキュア製造 GUI

1. アプリケーションフォルダをクリックし、直前のステップで **C:\Renesas\Synergy\SecureBootManager\evaluation\embedded** 内に生成された ssrec ファイルを参照します。
2. アプリケーションのフラッシュ書き込みに使用する COM ポートの番号 (ステップ 5.1.1 で接続したもの) を、[RS232 ports] (RS232 ポート) ボックスに入力します (ここで入力している 12 は、前のスクリーンショットで表示されていた COM12 を意味します)。
3. **[Scan via RS232]** (RS232 経由でスキャン) ボタンをクリックします。上記の例では、デバイスのシリアル番号が 3996460049 であるボードがシリアルポート 12 で見つかりました。ブートローダのバージョン 1 がインストール済みです。[App_version] (アプリケーションのバージョン) が「0」になっているのは、アプリケーションが何もインストールされていないことを示します。[Updateable] (更新可能) フィールドは緑色の [yes] (はい) を表示しており、アプリケーションがインストール可能であることを意味します。
4. [OEM Certificate Generation and Programming] (OEM 証明書生成とプログラミング):
   A. **[Issue Challenge]** (チャレンジの発行) ボタンをクリックし、デバイスに対して乱数のチャレンジを生成します。まだ OEM 証明書がデバイスにプログラムされていないので、チャレンジは失敗します。
   B. OEM 証明書を生成してプログラムすることを意図している場合、[OEM Certificate Generation and Programming] (OEM 証明書の生成とプログラミング) の下にある [Enable] (有効化) ボックスをクリックします。
   C. 暗号化が有効な場合、この操作が必須です。これはイメージの暗号化鍵を暗号化する目的で、デバイス ID を使用するからです。

注記: OEM 証明書をプログラムした後、SBM (セキュアブートマネージャ) を含めてすべてを消去しない限り、この証明書の消去や再プログラムを行うことは不可能です。

5. この時点で、コンソールログをクリア (または再起動) することを推奨します。
6. このボードに対応する [Selected] (選択済み) ボックスをオンにした後、**[Push to Device]** (デバイスへの書き込み) をクリックします。ソフトウェア (および、[有効化] を選択した場合は OEM 証明書も) の展開プロセスが開始されます。
   — [Status] (ステータス) の [Establish Connection] (接続の確立) ボックスが緑色になります。
   — [OEM Certificate Generation and Programming] (OEM 証明書の生成とプログラミング) を有効にした場合。
     - デバイスが鍵ペアを生成して公開鍵を展開ツールに送信した時点で、**[Received Device Public Key]** (デバイスの公開鍵を受信済み) ボタンが緑色になります。
     - 展開ツール（deploy tool）が X.509 証明書を作成し、デバイスにその証明書をプログラムした時点で、**[Programmed OEM Cert]** (OEM 証明書の書き込み完了) ボタンが緑色になります。
     - 展開ツールが、書き込んだ証明書をデバイスから読み取った時点で、**[Read Back OEM Cert]** (OEM 証明書の読み取り完了) ボタンが緑色になります。
     - 展開ツールが CA (認証局) を基準として、デバイスから受け取った OEM 証明書を検証した時点で、**[Validated OEM Cert]** (OEM 証明書の検証完了) ボタンが緑色になります。
   — 更新イメージをロードする前に、QSPI メモリを消去するために 10 秒の遅延が発生します。
   — 署名済みかつ暗号化済みの srecord ファイルを GUI にロードした時点で、**[Deliver Image]** (イメージの配信) ボタンが緑色になります。
   — ボード宛にアプリケーションを送信し、ボードでそのアプリケーションを検証した後、実行領域に適用した時点で、**[Validated Image]** (検証済みイメージ) と **[Apply Image]** (イメージの適用) の各ボタン全体に重ね合わせる形で緑色の進行状況が表示されます。
   — デバイスに対応する [Scan Results] (スキャン結果) グループ内の [App_Version] (アプリケーションのバージョン) エントリが更新されます。アプリケーションがインストールされたことを反映する形で、(元の値である「0」から)「1」に更新されます。
7. デバイスがリブート (再起動) します。現在、アプリケーションが実行されていることに注意してください。橙の LED は点灯したまま、緑の LED と赤の LED が交互に点灯します。LCD ディスプレイは、「Synergy Secure Bootloader demo app 1」(デフォルトは version0、バージョン 0) がインストールされたこと、および他のいくつかのステータス情報を表示します。「module=1, version = 1」(モジュール = 1、バージョン = 1) という行に注意してください。これは、アプリケーションのバージョン番号 (「1」は、1 番目のモジュールであること) を意味します。
8. コンソールログをスクロールして全体を表示し、一連のイベントを確認します。下図はスクリーンショットの一部で、コンソールログを示しています。

```
*********************************************************
[secBootloader] Synergy Secure Bootloader v0.9 (build=1)
*********************************************************

[MT] - Validating bootloader module table
[MT] - Validating bootloader module table signature
[MT] - bootloader module table signature VALID

[MT] - Validating bootloader code
[MT] - 0 - v1 - module validated - startAddr=0x0, length=65023
[MT] - bootloader module table VALID

[MT] - Validating main module table signature
[MT] - main module table signature VALID

[secBootloader] - found a pending update. Applying it
[secBootloader] - Pending update location is 0x60020000, 322048.
UPK - valid
decryptInitialise - decrypting first 32784 bytes at 0x60020128
count=0
decryptNextSegment - decrypting 176 bytes at 0x60028138
decodeHeader PASSED
decodeAddModify - module(1) is valid. Adding it to the module table
decryptNextSegment - decrypting 368 bytes at 0x600281e8
decodeAddModify - programming segment of size 32768 at 0x30000
decryptNextSegment - decrypting 32768 bytes at 0x60028358
decodeAddModify - programming segment of size 32768 at 0x38000
decryptNextSegment - decrypting 32768 bytes at 0x60030358
decodeAddModify - programming segment of size 32768 at 0x40000
decryptNextSegment - decrypting 32768 bytes at 0x60038358
decodeAddModify - programming segment of size 32768 at 0x48000
decryptNextSegment - decrypting 32768 bytes at 0x60040358
decodeAddModify - programming segment of size 32768 at 0x50000
decryptNextSegment - decrypting 32768 bytes at 0x60048358
decodeAddModify - programming segment of size 32768 at 0x58000
decryptNextSegment - decrypting 32768 bytes at 0x60050358
decodeAddModify - programming segment of size 32768 at 0x60000
decryptNextSegment - decrypting 32768 bytes at 0x60058358
decodeAddModify - programming segment of size 32768 at 0x68000
decryptNextSegment - decrypting 32768 bytes at 0x60060358
decodeAddModify - programming segment of size 32768 at 0x70000
decryptNextSegment - decrypting 26256 bytes at 0x60068358
decodeAddModify - programming segment of size 26112 at 0x78000
decodeAddModify - PASSED
decodeTrailer - PASSED
Decoded update.
[secBootloader] - Update has been applied. Rebooting.


*********************************************************
[secBootloader] Synergy Secure Bootloader v0.9 (build=1)
*********************************************************

[MT] - Validating bootloader module table
[MT] - Validating bootloader module table signature
[MT] - bootloader module table signature VALID

[MT] - Validating bootloader code
[MT] - 0 - v1 - module validated - startAddr=0x0, length=65023
[MT] - bootloader module table VALID

[MT] - Validating main module table signature
[MT] - main module table signature VALID

[MT] - Validating main module table
[MT] - Validating main module table signature
[MT] - main module table signature VALID

[MT] - Validating main code
[MT] - 0 - v1 - module validated - startAddr=0x30000, length=321024
[MT] - main module table VALID

[MT] - Validating 1stRx module table
[MT] - Validating 1stRx module table signature
[MT] - 1stRx module table signature VALID

[MT] - Validating 1stRx code
[MT] - 0 - v1 - module validated - startAddr=0x18000, length=31743
[MT] - 1stRx module table VALID

[MT] - Selecting program from main module table.
[secBootloader] Launching found program at addr=0x30000
[secBootloader] invokeAddr=0x36c25, stack=0x2000eb08
```

— 1st Receiver は、「鍵ペアを作成し、その後、公開鍵をセキュア製造ツール（Secure Manufacturing tool）に送信する」ことを求めるリクエストをセキュア製造ツールから受け取ります。その後、セキュア製造ツールは証明書を作成し、その証明書を 1st Receiver に送り返します。1st Receiver は OEM 証明書領域にその証明書をプログラムします。次に、チャレンジ/応答シーケンスが実行され、証明書が正しく MCU にプログラムされたことを検証します。

— 1st Receiver はセキュア製造ツールから、デバイス証明書（Device Certificate）とモジュールリスト（module list）のリクエストを受け取ります。その後、これらの情報をセキュア製造ツールに返します。

— 1st Receiver は、このツールから消去コマンドを受け取った後、QSPI 内のアプリケーション更新領域を消去します (図 14. メモリのレイアウト を参照)。

— 1st Receiver は、write (書き込み) コマンドを受け取った後、アプリケーション更新領域にユーザアプリケーションをプログラムします。

— 1st Receiver は、アプリケーションを適用する、すなわち、アプリケーションを実行領域にプログラムするためのコマンドを受け取り、この機能を実行するためのブートローダ API を起動します。このブートローダ API は、データフラッシュの更新にかかわる情報を保存し、MCU をリセットします。

— デバイスがリブートします。ブートローダが起動し、更新領域内で更新が使用可能であることを確認すると、イメージの検証と暗号化解除（decrypt）を行い、その後、実行領域にイメージをプログラムします。その後、MCU がリセットされます。

- デバイスが再びリブートします。ブートローダが起動し、ユーザアプリケーションの署名を検証して、アプリケーションの正当性（authenticity）と、正常にプログラムされたかどうかを判定します。成功した場合、ブートマネージャがそのアプリケーションを実行します。**注記：デバイスがブートするたびに、このステップが実施されます。**
- この時点で、署名済みユーザアプリケーションのデバイスへのプログラムが完了しています。セキュアブートローダはそのアプリケーションの検証を完了し、既に実行しました。

9. OEM 証明書が生成され、プログラムされた場合は、**[Issue Challenge]** (チャレンジの発行) ボタンをクリックして、デバイスに対してチャレンジを発行し、デバイスが証明書の所有者であることを検証します。この検証を実施するために、ツールはランダムデータバッファ（randsam buffer of data）をデバイスに送信します。デバイスは自らの公開鍵を使用してそのバッファに署名し、ツールはデバイスから受け取った OEM 証明書を使用してその署名を検証します。この検証を機能させるために、ユーザアプリケーション (この例では app1) に、チャレンジに応答する能力を付与する必要があります。app1 には、チャレンジに対する応答を実装するために、crypto HAL calls を使用する方法を示すサンプルが付属しています。
10. もう一度 **[Scan via RS232]** (RS232 経由でスキャン) をクリックすると、[Updateable] (更新可能) フィールドが赤い [no] (いいえ) に設定されます。(バージョンが「1」であるアプリケーションを、自らを使用して「更新」することはできません。)
11. GUI を終了します。

6. 展開済みアプリケーションの更新 (セキュア更新)（Updating the Deployed Application (Secure Update)）

セキュア更新は、2 段階で構成されています。

1. 更新マスタリングツール（Update Mastering tool）を使用してアプリケーションのバイナリ（application binary）をパッケージ化します。このバイナリは、デバイスに既にインストールしたアプリケーションに対する更新です。更新が許されるのは、既にインストール済みのバージョンより大きいバージョン番号をもつアプリケーションのみです。(たとえば、バージョン 2 をインストール済みの場合、インストール可能なのはバージョン 3、4 などに限定されます。)
2. 更新を開発ボードにインストールするには、更新展開ツール（Update Deployment tool）を使用します。

注記：更新をデバイスに展開するメカニズムは、アプリケーション固有（application specific）です。更新をダウンロードできるように、アプリケーションレベルでの接続を確保するのはアプリケーション側の責任です。ただし、このリファレンスデザインの目的として、アプリケーションは 1st Receiver が使用するのと同じメカニズム、すなわち UART 上で動作するフレームプロトコル（Frame Protcol）を使用します。

6.1　署名済み更新の準備（Preparation of Signed Update）

以下の画像で図示している **C:\Renesas\Synergy\SecureBootManager\evaluation\gui\masteringUpdateGUI.exe** を実行します。以下のステップを実施します。

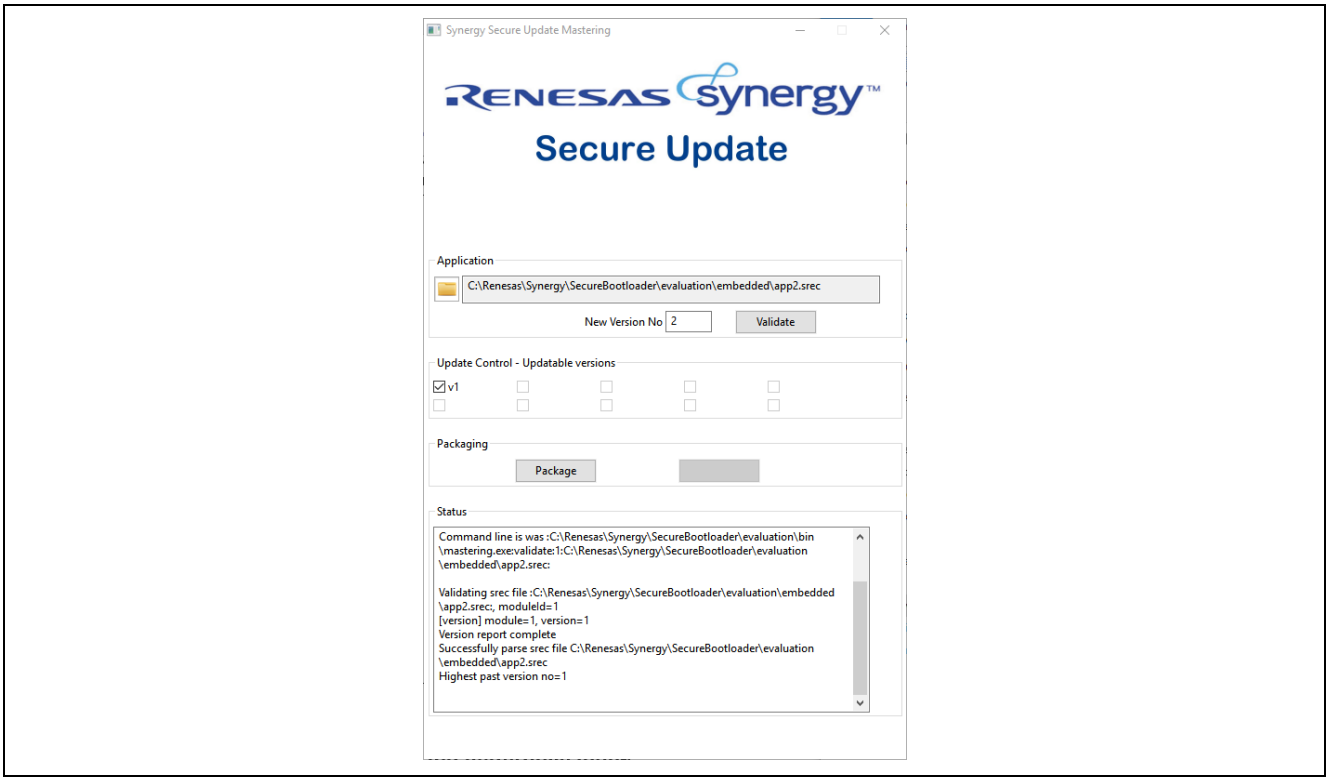


図 9. 署名済み更新のマスタリング GUI

1. アプリケーションフォルダをクリックし、**C:\Renesas\Synergy\SecureBootManager\evaluation\embedded** を参照して、**app2.srec** を選択します。これは、パッケージ化しようとするサンプルアプリケーションの更新です。App1 に比べてわずかな違いがあります。
2. [New version no] (新しいバージョン番号) を「2」に設定し、**[Validate]** (検証) をクリックします。これにより、.srec ファイルが解析可能（parsed）であるかどうか、バージョン 2 がまだ使用されていないかがチェックされます。すなわち、データベース内に、このバージョン番号に対応するバージョン 2 がまだ使用されていないか、（すなわち、データベース内にこのバージョン番号に対応するエントリがまだ存在していないかどうか）がチェックされます。
3. [Update Control box] (更新の制御ボックス) で、既にフィールドに展開済みであるバージョンのうち、この更新の適用先として使用するバージョンにチェックマークを付けます。この例では、アプリケーションコードの v1 が既に展開済みなので、[v1] ボックスにチェックマークを付けます。その結果、バージョン v1 を保持しているデバイスを、マスタリングしようとしている新しいバージョンで更新できるようになります。
4. **[Package]**（パッケージ化）をクリックすると、アプリケーションのバイナリに署名され、そのバイナリは app2_YYYY_MM_DD_HH_mm_ss.ssrec (独自のバイナリ形式) という名前で、入力 srec ファイルと同じフォルダに書き込まれます。
5. GUI を終了します。

6.2　更新済みアプリケーションのフラッシュ書き込み（Flashing the Updated Application）

C:\Renesas\Synergy\SecureBootManager\evaluation\gui\deployUpdateGUI.exe を実行します。

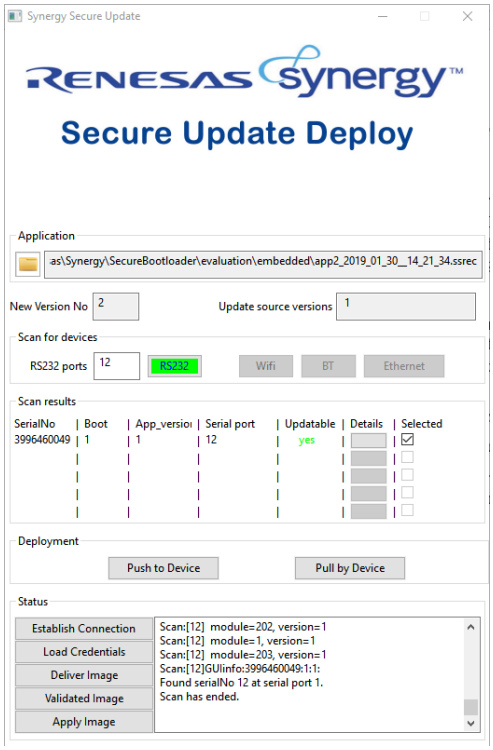


図 10. 署名済み更新の展開 GUI

1. フォルダアイコンをクリックし、作成したばかりの以下のファイルを選択します。**C:\Renesas\Synergy\SecureBootManager\evaluation\embedded\ app2_YYYY_MM_DD_HH_mm_ss.ssrec**。
2. この .ssrec ファイルから新しいバージョン番号が抽出され、「2」に設定されます。
3. [Update Source versions]（更新ソースのバージョン）は、このファイルから抽出されます。この例の場合、「1」に設定されています。これは、バージョン 2 に更新できる、このアプリケーションの唯一のバージョンです。評価ボード上のデバイスは、列挙されたバージョンのいずれかが更新に一致する場合のみ、その更新をインストールします。該当しないバージョンを保持しているボードに対して更新を適用しようとした場合、デバイスのセキュアブートローダは更新の署名に基づいて、その更新のインストールを拒否します。
4. 5.1.1 章で説明されているとおりにボードが接続されていることを確認します。(最初からこのドキュメントの手順に従っている場合、ここで変更を加える必要はありません)。
5. [RS232 Ports] (RS232 ポート) を、プログラミングポートに割り当てられている COM ポートの番号に設定します。
6. **[RS232]** ボタンをクリックします。更新ツールは有効なデバイスが PC に接続されていることを確認し、[Scan results] (スキャン結果) テーブルにそれらのデバイスを表示します。この GUI を使用して、一度に最大 5 台のデバイスを更新することができます。
7. この例では、[App_version] (アプリケーションのバージョン) が「1」であるボードが 1 枚見つかり、[Updateable] (更新可能) フィールドが緑色の [yes] (はい) に設定されます。**[Selected] (選択) ボックスをオンにして、このデバイスを選択します**。
8. **[Push to Device]** (デバイスへの書き込み) ボタンをクリックします。
9. 更新がデバイスに展開されます。
10. デバッグターミナルにもシーケンスのログが出力されます。この例では、デバッグログでメインモジュールのバージョンが「2」と表示されることに注意してください。
11. GUI を終了します。

6.3　デバイスのリブート（Device Reboot）

更新のインストールが完了した時点で、デバイスがリブートします。セキュアブートコードは、アプリケーションの整合性（integrity）と正当性（authenticity）をチェックします。アプリケーションが有効である場合、セキュアブートコードはデバイスをリブートし、緑の LED と赤の LED が交互にそれぞれ 3 回点滅し、アプリケーションの更新が完了したことを示します。

注記：LCD は、「module=1, version = 2」(モジュール = 1、バージョン = 2) という行を表示し、バージョン番号が「1」から「2」に増加したことを示します。

7.　次の手順（Next Steps）

ここまでのステップで、以下のことを実施しました。

1. セキュアブートローダと 1st Receiver のパッケージ化。
2. セキュアブートローダと 1st Receiver の展開。
3. アプリケーションのバージョン 1 として開発ボードにインストールする目的で、App1 のパッケージ化とマスタリング。
4. セキュア製造プロセス（Secure Manufacturing Process）を使用し、App1 を開発ボードに展開。
5. アプリケーションの v1 を既にプログラムしたボードを更新する目的で、App2 のパッケージ化とマスタリング。
6. セキュア更新プロセスを使用し、開発ボードを App2 に更新。

7.1　鍵データベース（Key Database）

コマンドウィンドウを開き、**C:\Renesas\Synergy\SecureBootManager\evaluation\bin** フォルダを選択し、KeystoreManager.exe print コマンドを実行してデータベース内にある複数の鍵を表示します。Module 1 (この例のアプリケーション) に対応する複数の鍵を書きとめます。アプリケーションのバージョンごとに、署名と更新に使用する鍵が存在します。このユーティリティ (コマンド) を動作させ、このデモの全ステップを通して、保存されている鍵がどのように変化するかを確認することもできます。

また、マスタリングツールも、鍵を生成して保存する目的でこのツールを起動します。

KeyStore は、鍵の生成と保存を行う目的で、Microsoft CAPI Next Generation (Microsoft 暗号化 API 次世代バージョン) を使用します。

注記：この実行可能ファイルは、print と printall の各コマンドを使用してデータベースの情報を表示することを目的としています。それ以外の用途に使用しないでください。鍵データベースに変更を加えた場合、将来の更新が失敗する可能性があります。

7.2　評価のリセット（Reset the Evaluation）

セキュア製造プロセスとセキュア更新プロセスのため、アプリケーションが既にインストールされているボードに対してセキュア製造プロセスを再度実行することはできません。この問題を避け、評価に使用できるように、スクリプトが提供されています。このスクリプトは更新プロセスを使用して、ボードにインストール済みのアプリケーションを削除し、PC が保持している鍵データベースをリセットします。この方法により、再実行可能になります。

7.2.1　アプリケーションのリセット（Resetting Application）

セキュアブートマネージャをインストールしたが、アプリケーションのマスタリングやインストールをまだ実施していないポイント（5 章、「ユーザアプリケーションのマスタリングとプログラミング (セキュアな製造)（Mastering and Programming the User Application (Secure Manufacturing)）」を参照してください）までフローをリセットしてさかのぼるには、以下のステップを実行します。

1. J1 が 1-2 の位置にあることを確認します。
2. ジャンパを使用し、ボード上の J2 を短絡し、その後解放する方法で、MCU をリセットします。
3. コマンドウィンドウを開き、**C:\Renesas\Synergy\SecureBootManager\evaluation\resetEvaluation** に移動します。
4. 「resetEvaluation.bat <COMportNo>」を実行します。ここで、<COMportNo > はプログラミング用ポートの番号です (たとえば、resetEvaluation.bat 12)。

注記: リセットにより、部分的にポートが変わると、ユーザアプリケーションが USB (たとえば、app2 usb.srec) を使用している場合、以下のステップは動作しません。その場合、次の章へ進んでシステムをリセットし、最初の状態に戻してください。

このスクリプトを実行した時点で、以下の状態になります。

- 鍵データベースが更新され、アプリケーションのすべてのバージョンに対応する鍵を削除済み。
- アプリケーションをデバイスから削除済み。
- 既にマスタリングしたすべてのファイルが、evaluation/embedded (評価/組み込み) フォルダから削除済み。
- 鍵データベースは、セキュアブートマネージャと 1st receiver のマスタリングに使用した複数の鍵を引き続き保持。
- セキュアブートマネージャはデバイス上に引き続き存在し、1st receiver は再インストール済み。
- デバイスはリブートされ、1st receiver は新しいアプリケーションイメージが送られてくるまで待機しています。デバッグログを参照して、このことを検証できます。ログは、1st receiver がブートされ、フレームプロトコル（frame protocol）を有効にしたことを示しています。緑と黄色の LED が点滅します。
- この段階で、5 章、「ユーザアプリケーションのマスタリングとプログラミング (セキュアな製造)（Mastering and Programming the User Application (Secure Manufacturing)）」からステップを再実行することができます。

注記: OEM 証明書を最初の場所に既にプログラムした場合、このステップを実行しても OEM 証明書はクリアされません。OEM 証明書を再プログラムできるようにするには、以下の章に従って SBM (セキュアブートマネージャ) をリセットします。

5. このステップが機能しない場合、以下の章に従って、システム全体をリセットします。

7.2.2 セキュアブートマネージャのリセット（Resetting Secure Boot Manager）

デモをリセットして、セキュアブートマネージャをまだインストールしていない最初のポイント (3 章を参照) までさかのぼるには、以下のステップを実行します。

1. コマンドウィンドウを開き、**C:\Renesas\Synergy\SecureBootManager\evaluation\bin** に移動します。
2. 「**keystoreManager.exe** clear」を実行します。

この時点で、以下の状態になっています。

- 鍵データベースのすべての鍵を削除済み。
- この段階で、3 章からステップを再実行できます。

SMPU と FAW の各ビットを有効にしてある場合 (このデモのデフォルト設定)、以下の章に従ってこれらのビットをクリア (0 に設定) します。

7.2.3 SMPU/FAW ビットのリセット（Resetting the SMPU/FAW bits）

SBM の設定でセキュア MPU を有効にしていた場合、それ以降、JTAG (J5) 経由でデバッガに接続することはできません。同様に、FAW (フラッシュアクセスウィンドウの設定) のビットをセット (1 に設定) した場合、通常のプログラミング (書き込み) でこれらのビットを消去することはできません。

この状態から復元するには、以下の手順に従ってください。

1. MCU をブートモードに設定しますが、J1 を **2-3 の位置**に変更します。
2. J2 を使用してボードをリセットします。
3. 「S5D9_ERASE_SMPU_OSIS_AWS.bat」ファイルを実行します。このファイルは、**\SecureBootManager\evaluation\resetEvaluation\Erase_SMPU_OSIS_AWS** の下にあります。このバッチファイル（batch file）内で、J-Link のインストールパスに合わせて J-Link のパスを編集する必要が生じることがあります。
4. J2 を使用してボードをリセットします。

このスクリプトは最初に JTAG ID と FAW の設定をクリアし、次に最初にフラッシュブロックを (したがって、セキュア MPU のビットも) 消去します。

注記: **FSPR ビットがセット済みの場合、セキュア MPU または FAW のビットをクリアすることはできません。FSPR を一度セットした後、変更することはできません。詳細は、10.1 セキュリティ設定（Security Settings）、および MCU のハードウェアユーザマニュアルを参照してください。**

7.3　パッケージ化と、他の更新の適用（Package and Apply Further Updates）

一連の手順の最後として、app4.srec を使用して他の更新を適用することもできます。この srec ファイルも、**C:\Renesas\Synergy\SecureBootManager\evaluation\embedded\bin** にあります。

- インストール済みのすべてのバージョンのボックスにチェックマークを付けます。

署名済み更新のマスタリング GUI（Signed Update Mastering GUI）で表示されている複数のバージョンのうち、特定のバージョンを選択することで、更新を選択的に行うこともできます。

注記：app_usb.srec を使用することもできます。このプロジェクトは、ユーザアプリケーションで UART の代わりに USB-CDC を使用します。J5 が接続されていること、そしてアプリケーションが USB 経由で通信できることを確認してください。展開ツールは、ダウンロードの送信に使用したのと同じポートを経由して、更新済みのコードが応答することを予期しています。USB は他のポートへ切り替えを行うので、展開コードは自らが予期していたことを確認できません。この確認が出来なくても更新プロセスで何も問題は起きませんが、GUI は無限に応答を待つことになります。更新済みのデバイスが MCU の USB-CDC ポートとして列挙された時点で、開発者は GUI を終了することができます。

7.4　トラブルシューティング（Troubleshooting）

一般的に発生する問題とその解決策は、以下のとおりです。

- 通信に失敗する（Communication fails）
  - 配線が説明どおりに接続されていることを確認します。
  - シリアル接続が規定のボーレート（baud rate）に設定されていることを確認します。
- プログラミング (書き込み) に失敗する（Programing fails）
  - 1st Receiver を実行して、ボードをリセットします。
  - ステータスを表示できるように、ログポート（log port）が接続されていることを確認します。
- ボードにインストール済みのアプリケーションと、PC に保存されている鍵データベース（key database）の両方をリセットしない限り、一連の説明を再実行（re-run）することはできません。2 回目のパッケージ化を試みた場合、赤の **FAILED** (失敗) ボタンが表示されます。ステータスウィンドウに、「failed to make new keys for add of module.」(モジュールを追加するための新しい鍵の作成に失敗しました。) と表示されます。

  1. コマンドウィンドウを開き、**C:\Renesas\Synergy\SecureBootManager\evaluation\resetEvaluation** に移動します。
  2. 「resetEvaluation.bat <COMportNo>」を実行します。
  3. ここで、<COMportNo > は、USB からシリアルへのコンバートアダプタ（converter adapter）に対応するポート番号です。
  4. マスタリングプロセス（masteringprocess）を再実行します。

8.　バイナリのビルド（Building the Binaries）

組み込みプロジェクトは、**C:\Renesas\Synergy\SecureBootManager\renesas\src\embedded** フォルダに配置されています。コードを再コンパイルしようとする場合のみ、以下の手順を実行する必要があります。

8.1　組み込みプログラム（Embedded Programs）

1. e2 studio で、**[File]** -> **[Import…]** -> **[Existing Projects into Workspace]**を選択し、**[Select Root Directory]** セクションで上記のフォルダを参照します。

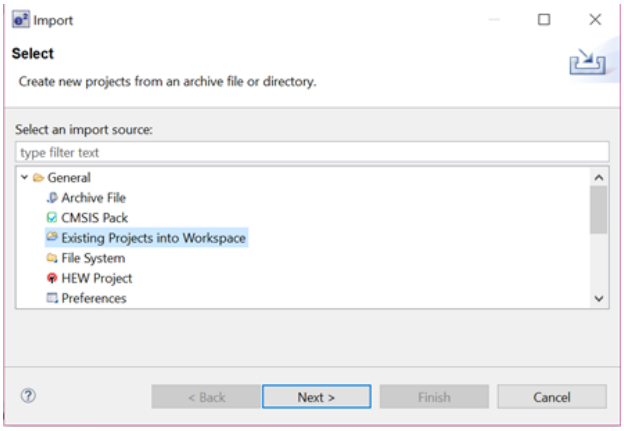


図 11. プロジェクトのインポート

2. 以下の図のように、すべてのプロジェクトをインポートするように選択を行います。**[Copy projects into workspace] (ワークスペースにプロジェクトをコピーする) ボックスをオンにしないでください。**

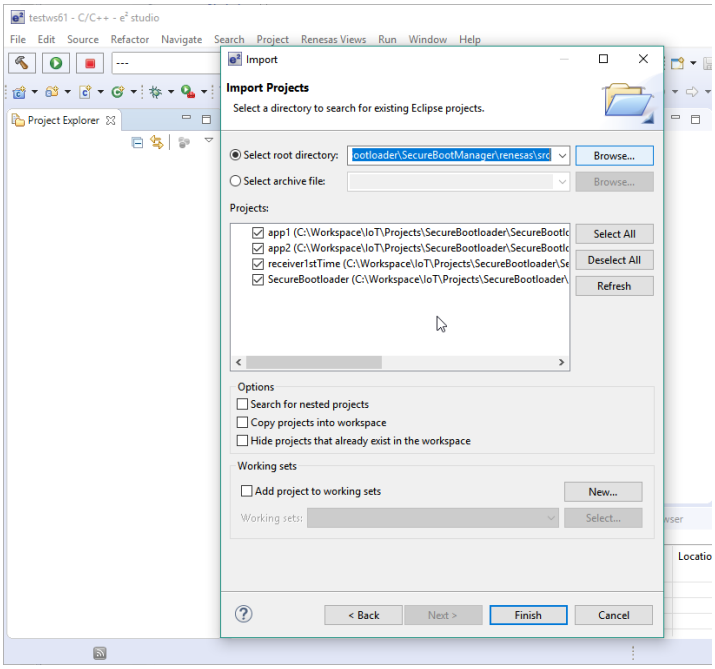


図 12. プロジェクトのリスト

3. e² studio で、各プロジェクトをビルドします。プロジェクトのビルドは、エラーが 0 で、戻り値 (リターンコード) が 0 であることが必要です。
   セキュアブートマネージャのサイズは、O2 最適化を指定し、デバッグ出力（debug prints）機能を有効にした場合は約 52 K バイトの ROM サイズ、デバッグ出力機能を無効にした場合は約 32 K バイトの ROM サイズです (10.2.2 章を参照)。

8.2　PC のプログラム（PC Programs）

- PC のコードはいずれも、Visual Studio® 2017 を使用してコンパイルします。
- Windows Explorer で、**C:\Renesas\Synergy\SecureBootManager\renesas\src\pc\apps** のパスを選択し、***.sln** ファイルを見つけます。
- 各 **.sln** ファイルを順にダブルクリックし、VS2017 にロードします。
- VS2017 で各プロジェクトをビルドします。プロジェクトは、エラーが 0 でビルドされることが必要です。deploy (展開) プロジェクトはデバッグに関連するいくつかの警告が発生しますが、これらは openSSL ライブラリのデバッグシンボル（debug symbol）に関連しており、コードの機能には影響ありません。この例は、deploy プロジェクトをコンパイルした後です。Microsoft のライブラリ依存（dependency）を最小にするには、リリースモード（Release mode）でプログラムをビルドします。

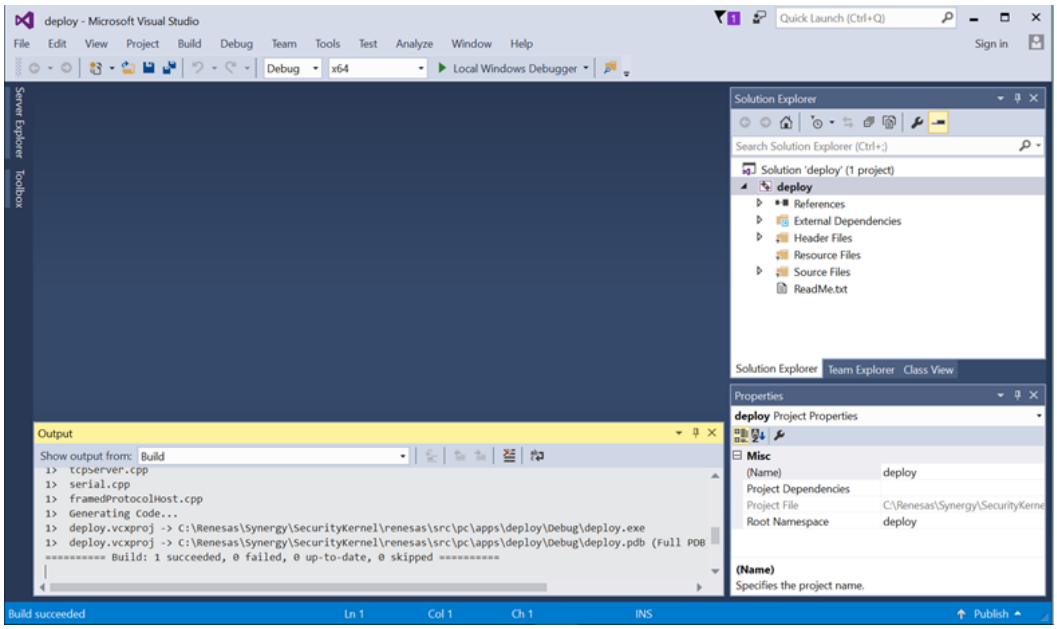


図 13. プロジェクトのコンパイル

9.　SSP アプリケーションの構成（Configuring an SSP Application）

セキュアブートローダアプリケーションで SSP アプリケーションを使用するには、共有システム要素（shared system element）について明確に理解することが必要です。以下の事項を検討する必要があります。

- SSP ユーザアプリケーションのリンカ（linker）セクションを更新し、コードの他の部分と重複しないようにしてください。
- SSP configurator は、アプリケーション内で ROM レジスタ (FAW、OFS、S-MPU) を構成するためのオプションを表示しますが、これらのレジスタを構成できるのはブートローダ内のみです。
- アプリケーションの更新を実行するためには、アプリケーションの側で、更新をダウンロードするために外部と通信できる能力、およびフラッシュ API を使用して更新を内部の更新領域にプログラムできる能力が必要です。

以下のステップは、新しい SSP アプリケーションが必要とするリンカの変更と、更新をダウンロードできるように SSP アプリケーションに対するフレームプロトコルの実装の追加を示しています。

9.1.1　リンカスクリプトの更新（Updating the linker script）

- 図 14 を参照して、メモリレイアウトを理解してください。
- この章では、付属の app1 プロジェクトに対応するリンカスクリプトを参考として使用します。このファイルは、以下の実行内容に似た構造を採用しています。このファイルは、**script** フォルダ内にある S5D9.ld ファイルです。e² studio 内で開く場合、このファイルを開いた後、下部にある [s5d9.ld] タブをクリックすると、このファイルがテキストモードで表示されます。
- 新しい SSP プロジェクトを作成し、希望する任意の機能（any functionality）を追加します (たとえば、特定のシーケンスを指定した Blinky)。
- 開発する SSP プロジェクトで、そのプロジェクトの scripts フォルダ内にある s5d9.ld リンカスクリプトファイルを開きます。e² studio 内で開く場合、このファイルを開いた後、下部にある [s5d9.ld] タブをクリックすると、このファイ

ルがテキストモードで表示されます。このファイルの先頭で、MEMORY 領域は SSP プロジェクト向けのレイアウトになっています。

- Bootloader (ブートローダ) プロジェクトで使用するメモリレイアウトは、**C:\Renesas\Synergy\SecureBootManager\evaluation\src\embedded\common\loaderScript\memoryMap.ld** 内で定義されています。
- 開発するアプリケーションのリンカスクリプトファイル内の MEMORY 定義（definition）を置き換えるため、S5D9.ld ファイルの先頭で **memoryMap.ld** ファイルをインクルード（include）します。ただし、ID Code (ID コード) セクションを除きます。include で、memoryMap.ld ファイルのパス全体を指定する必要が生じることがあります。

```
INCLUDE
C:\Renesas\Synergy\SecureBootManager\evaluation\src\embedded\common\loaderScript\memoryMap.ld

MEMORY

{

  ID_CODES (rx)      :ORIGIN = 0x0100A150, LENGTH = 0x10    /* 16 bytes (16 バイト) */

}
```

- メモリレイアウトを理解するために、memoryMap.ld ファイル全体を参照してください。特に、ユーザアプリケーション向けの予約領域（area reserved）と、更新向けの一時的な保持領域 (Update Area、更新領域) に注意を払います。以下の図を参照してください。

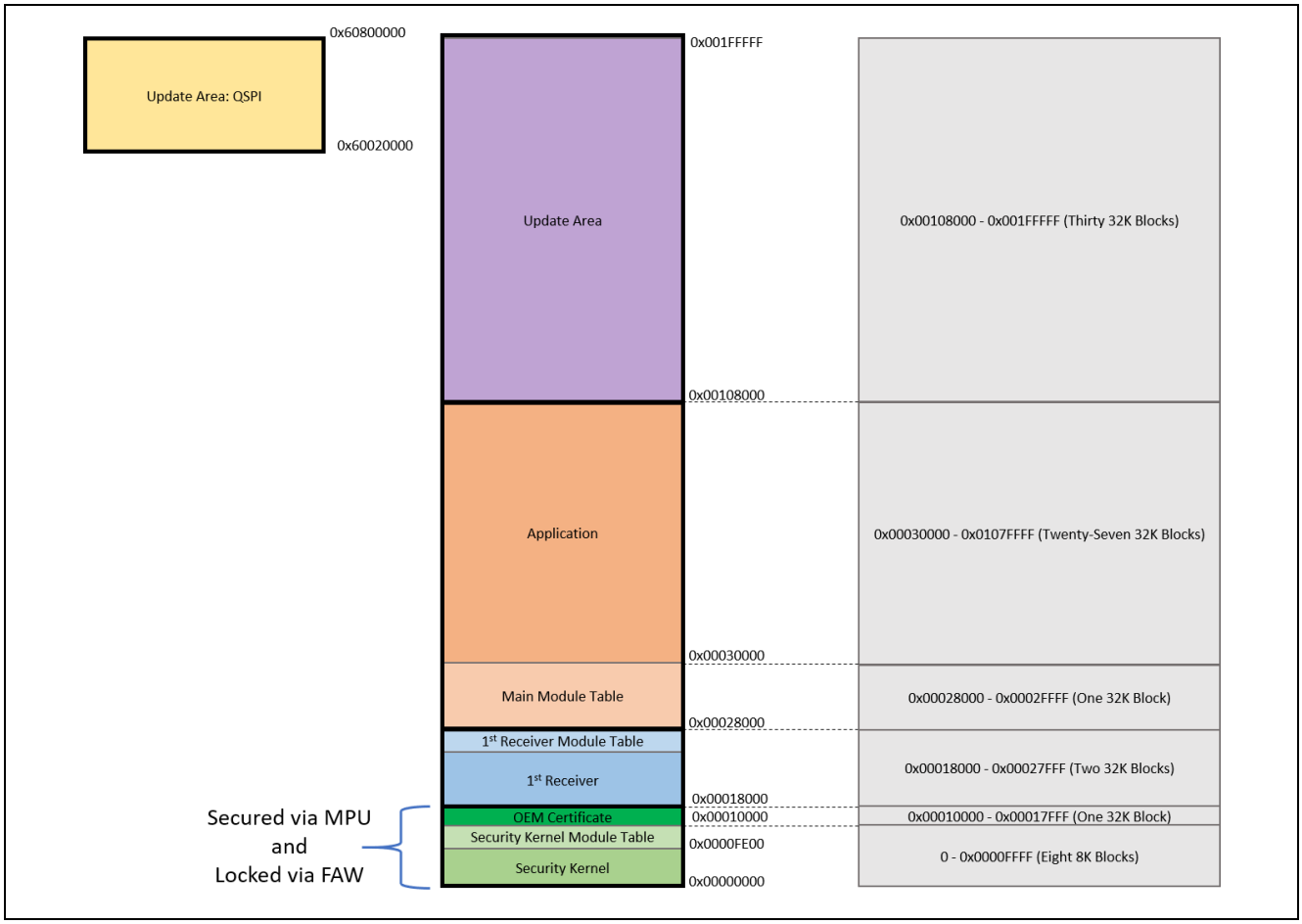


図 14. メモリのレイアウト

- 「KEEP(*(.vectors))」という行の前に、「__Vectors_Start = .;」という行を追加します。SSP が使用する「__Vectors_Start」というシンボルに対応する定義を作成するうえで、この行が必須です。
- DTC ベクタテーブル（vector table）のエントリ（entry）をコメントアウト（Comment out）します。この結果、DTC ベクタテーブルが強制的に固定の RAM 位置に配置されます。DTC ベクタテーブルを SSP プロジェクト内で使用する場合、RAM の中にこのテーブルが自動的に配置されます。

```
/*
.ssp_dtc_vector_table :
{
        .= ORIGIN(RAM);
        *(.ssp_dtc_vector_table)
} > RAM
*/
```

- プロジェクトをビルドします。app1.srec の代わりに、ここで生成された srec ファイルをセキュア製造プロセスまたはセキュア更新プロセスに使用できるようになります。
- ID Code と ROM register の各セクションは SSP アプリケーションの中で維持されてますが、これらはメモリに書き込まれていないことに注意してください。これらの位置は、ブートローダのコード自体の中で設定されるだけです。SSP がこれらのセクションが存在することを予期しているという理由で、これらが維持されているにすぎません。この点は将来修正する予定です。

9.1.2 SSP アプリケーションへのダウンロード機能の追加（Adding Download Capability to the SSP Application）

展開型のアプリケーションが、フィールド展開後（after field deployment）に更新（uodated）されることを予期している場合、そのアプリケーションに H/W (ハードウェア) 通信機能と、ダウンロードプロトコルソフトウェアが実装されている必要があります。現在の PC 側の更新展開ツール（update deployment tool）は、シリアルチャネル（serial channel）を経由して更新を送信するように実装したフレームプロトコル（Frame Protocol）を使用しています。付属の app1 プロジェクトは、シリアルポート経由でフレームプロトコルを使用して送信されたデータを受信する方法を示しています。ユーザが開発する SSP アプリケーションにダウンロード機能を追加する方法の詳細は、このプロジェクトを参照してください。付属の app2 プロジェクトは、USB 経由のダウンロード機能をユーザアプリケーションに追加する方法例を示しています。

開発するプロジェクトに、UART 経由のダウンロード機能を追加する主なステップは、以下のとおりです。

- フラッシュへの書き込みをサポートするために、r_flash_hp モジュールを追加します。
- データ転送（data transfer）をサポートするために、r_sci_uart モジュールを追加します。同じメディアを使用するように送信側と受信側の両方を更新する限り、UART 機能を他の任意の通信チャネルで置き換えることも可能です。
- 更新のダウンロード先となるアプリケーションスレッド（appliation thread）に、セマフォ（semaphore）とキュー（queue）を追加します。
- SSP プロジェクトにフレームプロトコルファイルを追加します。
  - framedProtocolCommon.c
  - framedProtocolCommon.h
  - framedProtocolTarget.c
  - framedProtocolTarget.h

- これらのモジュールを初期化し、受信したデータを処理する方法の詳細は、**app1/src** フォルダ内の **PMOD_thread_entry.c** ファイルを参照してください。

9.1.3 SSP アプリケーションへの OEM 証明書のチャレンジ/応答機能の追加（Adding OEM Certificate Challenge/Response Capability to the SSP Application）

1st Receiver アプリケーションと PC 展開ツールの組み合わせは、ホストがデバイス証明書（device certificate）の識別情報（identity）を検証する方法を示しています。

一連の流れは以下の通りです。

1. PC 側展開ツール（deploy tool）は、MCU 上の 1st Receiver に対して OEM 証明書（certificate）を要求します。
2. 1st Receiver は、この証明書を受け取る目的で SBM API を呼び出した後、その証明書を展開ツールに送信します。
3. 展開ツールはローカル CA を基準としてその証明書を検証し、その証明書が有効かどうかを判定します。このステップは、証明書のみを検証し、送信側デバイスを検証しません。
4. 次に、展開ツールはランダムデータバッファ（randam data buffer）を生成し、そのバッファを 1st Receiver に送信します。
5. 1st Receiver は、SCE ハードウェアを呼び出す形で、ラッピングされた秘密鍵（wrapped private key）を使用してそのデータに署名し、その署名を展開ツールに送信します。
6. 展開ツールは、X.509 証明書（certificate）から取得した公開鍵（public key）を使用してその署名を検証します。この操作により、デバイスの識別情報（device identity）を検証します。

1st Receiver と展開ツール内のコードは、サンプルとして公開済みです。

チャレンジ/応答（challenge/response）のメカニズムは、暗号化 HAL モジュール（crypt HAL module）を使用する必要があるため、SSP ユーザアプリケーションがこのアクションを実行する場合は、そのアプリケーションは SSP が提供している暗号化 HAL モジュールを使用する必要があります。frameProtocolTarget.c ファイル内で、フレームプロトコルを経由してチャレンジ要求を受け取ります。この際に使用する handleHrkCertChallengeResp() 関数は、hwSignData() を呼び出します。この関数は、ユーザアプリケーション内で実装する必要があります。この関数は、サンプルとして 1st Receiver 内に実装済みですが、この関数は、1st Receiver プロジェクトの一部である暗号化モジュールを使用しています。ユーザ側を想定した app1 プロジェクトでも、challengeResponse.c ファイル内でサンプルを実装済みです。このサンプルは、SSP HAL 呼び出し（SSP HAL call）を使用して同じ機能を実装しています。

9.1.4 デバッグ（Debugging）

ユーザアプリケーションは、デフォルトのリンカ設定を使用して、通常のスタンドアロンプロジェクトと同じ方法で開発とデバッグを行うことができます。ただし、アプリケーションの機能が完成し、セキュアブートローダと組み合わせてテストする段階でこのようなテストを可能にするためには、デバッガにいくつかの変更を加える必要があります。

前の章の説明に従ってプロジェクトのリンカ設定を変更した後、すべてのデバッグシンボルが生成されるように、デバッグモードでそのプロジェクトをコンパイルします。そして、5.2 章の説明に従って srec ファイルをマスタリング（mastering）し、このマスタリング済み srec ファイルを、インストール済みのセキュアブートローダを使用して、デバイスにダウンロードする必要があります。

マスタリング済みのユーザアプリケーションをデバイスにダウンロードした時点で、デバッグセッション設定（debug session setting）を更新します。これは、デバイスにデバッガを接続している状態でコードフラッシュを更新しないようにするためです。実施この更新は、EnableFlashDL フラグを 0 に設定するなど、プロジェクト内の <debug_session_name>.jlink ファイルに変更を加えることで行います。

付属の Jlink デバッグセッションは、ブートローダと 1st Receiver に対して、フラッシュのダウンロード機能を無効にしてあります。

現在、デバイスにデバッガを接続している状態では、このセッションはデバッグシンボルをロードしますが、デバイスに何もコードをダウンロードしません。

この時点でコードを更新しようとする場合、更新マスタリングプロセス（update mastering process）全体をもう一度実行する必要があります。このため、システム統合ステージに進む前には、セキュアブートマネージャは使用せず、アプリケーションのすべての開発とテストを完了させておくことを推奨します。

デバッグの実行中にコードをデバイスにダウンロードしようとする場合、必ず上記のフラグを 2 に変更してください。

10. プロジェクトの設定（Project Configurations）

10.1 セキュリティ設定（Security Settings）

この章では、セキュリティ戦略（security strateg）の一環としてデータへのアクセスを制御する目的で使用する、MCU 内のさまざまなレジスタについて説明します。

10.1.1 セキュリティ MPU 領域の設定（Configuring the Security MPU area）

セキュリティ MPU の各レジスタを使用して、複数のメモリ領域への読み取りアクセスを制御します。ユーザはこのハードウェアを使用して、3 つのセキュア領域と、それらに直接対応しているわけではない 3 つの PC (プログラムカウンタ) 範囲を設定することができます。これらのセキュア領域を、ROM、RAM、またはデータフラッシュの中に配置可能です。

PC (プログラムカウンタ) が、定義済み PC 範囲（defined PC ranges）のいずれかの中にある場合のみ、すなわち、同じ範囲内に存在するコードを使用する場合のみ、セキュア領域を読み取ることができます。

セキュリティブートマネージャ（Secure Boot Manager）ファームウェアとインストーラの PC ツール (パーソナルコンピュータ上で動作するツール) は、セキュリティ MPU の各レジスタが、セキュリティブートマネージャ（Security Boot Manager）、セキュアブートマネージャのモジュールテーブル（Secure Boot Manager Module Table）、OEM 証明書のコード領域（OEM Certificate area of code）をカバーするように設定してあります。この方法で、セキュリティブートマネージャのコードベース（code base）を読み取ることを防止しています。

これらのレジスタは、SecureBootloader\sbmCfg\mcu_rom_cfg.h 内で構成します。これらのレジスタを変更する方法は、ハードウェアマニュアルを参照してください。

現在の設定は、以下のとおりです。

```
#define SECMPUAC_DISPC0_VAL (0U) //SECMPU Program Counter region 0 Enabled
#define MCU_CFG_SECMPU_PC0_START 0x00000000U // Security MPU Program counter region 0 Start address
#define MCU_CFG_SECMPU_PC0_END ((ADDR_BOOTLOADER_MT -1) | 0x3) // * Security MPU Program counter region 0 End address

#define SECMPUAC_DISPC1_VAL (1U) // * SECMPU Program Counter region 1 Disabled
#define MCU_CFG_SECMPU_PC1_START 0x00000000U // Security MPU Program counter region 1 Start address
#define MCU_CFG_SECMPU_PC1_END 0xFFFFFFFFU // Security MPU Program counter region 1 End address

#define SECMPUAC_DIS0_VAL (0U) // SECMPU Region 0 Enabled
#define MCU_CFG_SECMPU_S0_START 0x00000000U // Security MPU Region 0 (code flash) Start address
#define MCU_CFG_SECMPU_S0_END ((ADDR_1ST_RX -1) | 0x3) // Security MPU Region 0 (code flash) End address

#define SECMPUAC_DIS1_VAL (1U) // SECMPU Region 1 Disabled
#define MCU_CFG_SECMPU_S1_START 0x1FF00000U // Security MPU Region 1 (SRAM) start address
#define MCU_CFG_SECMPU_S1_END 0x200FFFFFU // Security MPU Region 1 (SRAM) end address

#define SECMPUAC_DIS2_VAL (1U) // SECMPU Region 2 Disabled
#define MCU_CFG_SECMPU_S2_START 0x400C0000U // Security MPU Region 2 (security function 1) start address
#define MCU_CFG_SECMPU_S2_END 0x400DFFFFU //Security MPU Region 2 (security function 1) end address

#define SECMPUAC_DIS3_VAL (1U) // SECMPU Region 3 Disabled
#define MCU_CFG_SECMPU_S3_START 0x400C0000U // Security MPU Region 3 (security function 2) start address
#define MCU_CFG_SECMPU_S3_END 0x400DFFFFU //Security MPU Region 3 (security function 1) end address
```

セキュリティ MPU を有効にした状態でセキュアブートマネージャのインストールが成功した後、**プログラミングインタフェース（programming interface）J5 はアクセス不可になります**。すなわち、通常の JTAG インタフェースを経由して、アプリケーションのフラッシュ書き込みを再度実施することは不可能になります。プログラミングインタフェース J5 へのアクセスを再度可能にするには、ブロック 0 全体を消去することで、セキュリティ MPU の設定を無効にする必要があります。この作業を行うには、7.2.3 章 SMPU/FAW ビットのリセット（Resetting the SMPU/FAW bits）の説明に従ってスクリプトを実行します。

10.1.2 フラッシュアクセスウィンドウの設定（Configuring the Flash Access Window）

フラッシュアクセスウィンドウ (FAW) は、セキュリティブートマネージャと 1st Receiver が上書きまたは変更されることを防止する目的で使用します。FAW は、ユーザが消去と再プログラム (再書き込み) を実行できるアドレス範囲を設定することでその機能を果たします。FAW は、Visual Studio の Kernel Installer アプリケーション内で、SBM モ

ジュールテーブルの最後から、プログラムフラッシュの先頭までのアドレス範囲をカバーするように設定してあります。

FAW は、デフォルトで有効になっています。ただし開発中は、セキュアブートマネージャを再プログラムできるように、すなわち、プログラムするたびにボードをロックすることがないように FAW を無効にしておくことを推奨します。

FAW を有効にするには、Visual Studio の Kernel Installer プロジェクト内で ENABLE_FLASH_ACCESS_WINDOW を定義します。この作業を行うには、そのプロジェクトのプロパティを開き、**[C/C++>Preprocessor]**の下で、ENABLE_FLASH_ACCESS_WINDOW という [Preprocessor Definition]を追加します。FAW を無効にするには、プロジェクトからこの定義を削除します。

これらのビットを MCU にプログラムした後、これらを消去するには特別な方法が必要になります。これらのビットを消去するには、7.2.3 章 SMPU/FAW ビットのリセット（Resetting the SMPU/FAW bits）の説明にあるスクリプトを実行します。

10.1.3 FSPR の構成（Configuring the FSPR）

FSPR ビットは、セキュア MPU と FAW の各レジスタの変更を防止する目的で使用します。**このビットを一度セットした後は変更が不可能**なので、製造時にセキュリティブートマネージャをデバイスにプログラムする時点でのみ、このビットは設定されます。

FSPR をプログラムするには、Visual Studio の Kernel Installer プロジェクトで ACTIVATE_THE_FSPR を定義します。この作業を行うには、そのプロジェクトのプロパティを開き、**[C/C++>Preprocessor]** の下で、ACTIVATE_THE_FSPR という [Preprocessor Definition] (プリプロセッサ定義) を追加します。

プロジェクト内で、この定義はデフォルトで無効になっています。

誤った内容をプログラミングした場合、デバイスの回復が不可能なこともあるので、デバイスのカーネルとアプリケーションが正しく構築されていることを確認するように、注意を払うことが必要です。

10.1.4 JTAG アクセスの設定（Configuring JTAG access）

OSIS レジスタ内の ID Code (ID コード) はまだ設定されていません。この結果、JTAG がデバイスにアクセスできます。証明書インストーラツール（Certificate Installer tool）は、ID Code レジスタが設定でき、またセキュアブートローダでプロビジョニングしたデバイスに対して JTAG アクセスを制限することができます。
開発時にセキュアブートマネージャと 1st Receiver が再プログラムできるように、JTAG ID はデフォルトで無効になっています。永続的にボードをロックすることが出来ません。JTAG ID を有効にするには、Visual Studio の Certificate Installer プロジェクト内で「ENABLE_JTAG_LOCKING」を定義します。この作業を行うには、そのプロジェクトのプロパティを開き、[C/C++>Preprocessor]の下で、「ENABLE_JTAG_LOCKING」という [Preprocessor Definition] (プリプロセッサ定義) を追加します。

次にこのプロジェクトを再コンパイルし、希望の JTAG ロック ID を使用して実行可能ファイルを起動します。

10.2 全般的な設定（General Settings）

10.2.1 OEM 証明書の使用の有効化（Enabling usage of OEM Certificate）

システムで OEM 証明書の使用を有効にするには、ENABLE_SBM_OEM_CERTIFICATE シンボルをコンパイラのプリプロセッサに追加します。具体的には、e² studio で **[SecureBootloader]** プロジェクトを右クリックし、**[Settings]>[Cross ARM C Compiler]** の下でシンボルを追加します。この作業を 1st Receiver プロジェクトとユーザアプリケーションでも同様に実行します（新しい更新をダウンロードするために、ユーザアプリケーションで frameProtocolTarget.c ファイルを使用している場合）。

10.2.2 ハードウェア暗号化エンジンの使用の有効化（Enabling use of Hardware Crypto Engine）

ユーザはこのソフトウェアを使用して、暗号化操作（cryptographic operation）にハードウェア要素またはソフトウェア要素を使用するかどうかを決定することができます。S5D9 は必要なハードウェア暗号化要素すべてをサポートしているので、最高の性能を達成するために、すべての要素を有効にすることを推奨します。

crypto_cfg.h ファイルは、SHA256 と ECC (署名と検証) に対するハードウェアサポートを有効にするために定義する必要のあるマクロを記述しています。

10.2.3 デバッグ出力の有効化（Enabling Debug Prints）

SCI8 シリアルポートでデバッグ出力を無効にするには、DISABLE_PRINTF マクロをコンパイラのプリプロセッサに追加します。そのために、e² studio で **[SecureBootloader]**プロジェクトを右クリックします。**[Settings]>[Cross ARM C Compiler]**の下で、このマクロを追加します。この作業を実行すると、ソフトウェアのサイズが非常に小さくなります。

10.2.4 OFS レジスタの設定（Configuring the OFS registers）

OFS register (OFS レジスタ) ビットは、mcu_rom_cfg.h ファイル内で設定できます。これらのレジスタの詳細は、ハードウェアマニュアルを参照ください。

10.2.5 ユーザ定義関数（User-defined Functions）

SecureBootloader (セキュアブートローダ) プロジェクトは、セキュアブートローダのフローを制御する目的でユーザが定義することのできるフック（hook）を記述しています。アプリケーションが必要とする場合、ユーザはこれらの関数を置き換えることができます。

これらの関数は、securityDebug.h と securityUser.h で列挙されており、サンプルの実装は securityDebug.c と securityUser.c で記述されています。

10.3 設定の概要（Configuration Summary）

マクロ	説明	設定方法	PC 向けプロジェクト	e² studio プロジェクト
RX_1ST_BUILD (1st Receiver のビルド)	現在動作しているコードが 1stRX (1st Receiver) であるかどうかを判定するために、フレームプロトコルの組み込み実装の中で使用します。変更しないでください。	プロジェクト側で定義	N/A	1ST_RX
ENABLE_WINDOWS_KEYSTORE (Windows のキーストローク有効化)	Microsoft CAPI (暗号化 API) を使用し、鍵の管理に基づいて Excel のシートを置き換えます。変更しないでください。	プロジェクト側で定義	KernelPackager、KernelInstaller、Mastering、Deploy (カーネルパッケージ化、カーネルインストーラ、マスタリング、展開)	N/A
ENABLE_FLASH-ACCESS_WINDOW (フラッシュアクセスウィンドウ (FAW) の有効化)	フラッシュアクセスウィンドウ (FAW) を設定する目的で使用します。	プロジェクト側で定義	KernelInstaller (カーネルインストーラ) または CertificateInstaller (証明書インストーラ)	N/A
ADDR_FAW_START (FAW 開始アドレス) ADDR_FAW_END (FAW 終了アドレス)	FAW のアドレス範囲を設定します	memoryMap.h	N/A	SecureBootloader (セキュアブートローダ)
ACTIVATE_THE_FSPR (FSPR の有効化)	FSPR ビットをロックします	プロジェクト側で定義	KernelInstaller (カーネルインストーラ) または CertificateInstaller (証明書インストーラ)	N/A

マクロ	説明	設定方法	PC 向けプロジェクト	e² studio プロジェクト
ENABLE_QSPI_FLASH_DOWNLOAD_SUPPORT (QSPI フラッシュダウンロードのサポートの有効化)	QSPI を Update Area (更新領域) として使用することを許可します	プロジェクト側で定義	Mastering、Deploy (マスタリング、展開)	すべて
ENABLE_QSPI_FLASH_USERDATA_SUPPORT (QSPI フラッシュユーザデータのサポートの有効化)	アプリケーション自体の一部として QSPI を使用するアプリケーションのプログラミングをサポートします。	プロジェクト側で定義	N/A	SecureBootloader (セキュアブートローダ)
ENABLE_SBM_OEM_CERTIFICATE (SBM OEM 証明書の有効化)	デバイス ID の作成とプログラミング (書き込み) をサポートします	プロジェクト側で定義	N/A	SecureBootloader、1ST_RX (セキュアブートローダ 1st Receiver)
ENABLE_ENCRYPTION (暗号化の有効化)	更新イメージの暗号化/暗号化解除をサポートします	プロジェクト側で定義	Deploy (展開)	SecureBootloader (セキュアブートローダ)
ENABLE_JTAG_LOCKING (JTAG ロックの有効化)	JTAG ロック ID を設定します	プロジェクト側で定義	CertificateInstaller (証明書インストーラ)	N/A
ADDR_BOOTLOADER (ブートローダのアドレス) BOOTLOADER_FLASH (ブートローダのフラッシュ)	ブートローダの開始アドレス	memoryMap.h memoryMap.ld	すべて	すべて
ADDR_BOOTLOADER_MT (ブートローダ MT のアドレス) MT_BOOTLOADER (ブートローダの MT)	ブートローダモジュールテーブルの開始アドレス	memoryMap.h memoryMap.ld	すべて	すべて
ADDR_1ST_RX (1st Receiver のアドレス) FIRST_RX (1st Receiver)	1st Receiver のコードの開始アドレス	memoryMap.h memoryMap.ld	すべて	すべて
ADDR_1ST_RX_MT (1st Receiver MT のアドレス) FIRST_RX_MT (1st Receiver の MT)	1st Receiver モジュールテーブルの開始アドレス	memoryMap.h memoryMap.ld	すべて	すべて
ADDR_OEM_CERT (OEM 証明書のアドレス) OEM_CERT (OEM 証明書)	OEM 証明書を保持する領域の開始アドレス	memoryMap.h memoryMap.ld	すべて	すべて
SIZE_OEM_CERT (OEM 証明書のサイズ)	OEM 証明書領域のサイズ。フラッシュのブロックサイズ 1 つ分にする必要があります。	memoryMap.h	すべて	すべて
FLASH (フラッシュ)	アプリケーションの開始位置	memoryMap.ld	すべて	すべて
ADDR_MAIN_MT (メイン MT のアドレス) MT_MAIN (メインの MT)	アプリケーションのモジュールテーブルの開始位置	memoryMap.h memoryMap.ld	すべて	すべて
ADDR_PENDING_UPDATE (保留中の更新のアドレス) PENDING_UPDATE (保留中の更新)	更新のダウンロード先となる領域の開始位置	memoryMap.h memoryMap.ld	すべて	すべて
BOOTLOADER_RAM_START_ADDR (ブートローダ RAM の開始アドレス) BOOTLOADER_RAM (ブートローダの RAM)	ブートローダが使用する RAM の開始位置	memoryMap.h memoryMap.ld	すべて	すべて
APP_RAM_START_ADDR (アプリケーションの RAM の開始アドレス) RAM	ユーザアプリケーションが使用できる RAM の開始位置	memoryMap.h memoryMap.ld	すべて	すべて
BOOTLOADER_DF_START_ADDR (ブートローダのデータフラッシュの開始アドレス) BOOTLOADER_DF_END_ADDR (ブートローダのデータフラッシュの終了アドレス) BOOTLOADER_DATA_FLASH (ブートローダのデータフラッシュ)	ブートローダが使用するデータフラッシュ	memoryMap.h memoryMap.ld	すべて	すべて

Licenses

11. ライセンス（License）

Third-party cryptographic code is used in this demonstration and is subject to the following licenses.

11.1 SHA256 + HMAC

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

11.2 Elliptic Curve

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Web サイトおよびサポート

以下のさまざまな URL にアクセスし、Synergy プラットフォームの主な要素に関する詳細を確認し、それらに関連するドキュメントをダウンロードし、サポートを活用してください。

Synergy ソフトウェア	www.renesas.com/synergy/software
Synergy ソフトウェアパッケージ	www.renesas.com/synergy/ssp
ソフトウェアアドオン	www.renesas.com/synergy/addons
ソフトウェア用語集	www.renesas.com/synergy/softwareglossary
開発ツール	www.renesas.com/synergy/tools
Synergy ハードウェア	www.renesas.com/synergy/hardware
マイクロコントローラ	www.renesas.com/synergy/mcus
MCU 用語集	www.renesas.com/synergy/mcuglossary
パラメトリック検索	www.renesas.com/synergy/parametric
キット	www.renesas.com/synergy/kits
Synergy ソリューション Gallery	www.renesas.com/synergy/solutionsgallery
パートナープロジェクト	www.renesas.com/synergy/partnerprojects
アプリケーションプロジェクト	www.renesas.com/synergy/applicationprojects
セルフサービスサポートリソース:	
ドキュメント	www.renesas.com/synergy/docs
ナレッジベース	www.renesas.com/synergy/knowledgebase
フォーラム	www.renesas.com/synergy/forum
トレーニング	www.renesas.com/synergy/training
ビデオ	www.renesas.com/synergy/videos
Web チケット	www.renesas.com/synergy/resourcelibrary

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.02	2019.10.10	—	初版 英文版 R12AN0093EU0102 Rev.1.02 を翻訳

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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、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。
5. 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。
6. 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。
7. 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。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。