RL78/G22, RL78/G23, RL78/G24
ファームウェア アップデート モジュール

要旨

本アプリケーションノートでは、RL78/G22 および RL78/G23、RL78/G24 向けのファームウェアアップデートモジュールについて説明します。以降、本モジュールをファームウェアアップデートモジュールと称します。

本モジュールを使用することで、セキュアブート機能とファームウェアアップデート機能をユーザアプリケーションに容易に組み込むことができます。本アプリケーションノートでは、ファームウェアアップデートモジュールの使用方法、およびユーザアプリケーションへの組み込み方法について説明します。

また、本アプリケーションノートのリリースパッケージにはデモプロジェクトが含まれています。「4 デモプロジェクト」に記載する手順に沿ってデモの実行環境を構築することで、ファームウェアアップデートの基本的な動作を確認することができます。

動作確認デバイス

RL78/G22（R7F102GGE）

RL78/G23（R7F100GSN）

RL78/G24（R7F101GLG）

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。あわせて参照してください。

- RL78 ファミリ ボードサポートパッケージモジュール Software Integration System(R01AN5522)
- RL78 ファミリ Renesas Flash Driver RL78 Type01 ユーザーズマニュアル(R20UT4830)
- RL78 スマート・コンフィグレータ ユーザーガイド: e² studio 編(R20AN0579)
- スマート・コンフィグレータ ユーザーズマニュアル RL78 API リファレンス編(R20UT4852)

ターゲットコンパイラ

- ルネサスエレクトロニクス製 CC-RL V1.11.00
- IAR Systems 製 IAR C/C++ Compiler for Renesas RL78 version 5.10.1
  IAR Assembler for Renesas RL78 version 5.10.1

各コンパイラの動作確認環境に関する詳細な内容はセクション「6.1 動作確認環境」を参照してください。

目次

1. 概要

1.1 ファームウェアアップデートモジュールとは

ファームウェアアップデートとは、機器自身が、機器の制御を行うファームウェアを何らかの手段で入手した新しいファームウェア（本書では更新イメージと呼ぶ）に書き換えることです。ファームウェアアップデートは不具合の修正や新機能の追加、性能向上などのために行われます。

ファームウェアアップデートモジュールは、お客様のシステムにファームウェアアップデート機能を組み込む際に、その部品として利用することが可能なミドルウェアであり、次の機能を備えています。

・通信インタフェースを介して更新イメージを MCU に取り込む機能

・更新イメージの正当性を検証する機能（検証方式は ECDSA NIST P-256 及び SHA256）

・更新イメージを内蔵フラッシュに書き込む（セルフプログラミングする）機能

・更新イメージを有効化する機能

ファームウェアアップデートシステムは、ファームウェアアップデート機能を持つアプリケーションプログラムと、そのプログラムの正当性を検証して起動するセキュアブート機能を持つブートローダの、二つのプログラムで構成されます。

ブートローダは、ファームウェアアップデートが正しく機能するためには必須の機能で、更新イメージの正当性検証を含むファームウェアアップデートの一連の処理が正当なものであることを保証します。

RL78 ファミリのファームウェアアップデートモジュールは、以下の３つのファームウェアアップデート方式を提供します。

・半面更新方式（バッファ面は内蔵フラッシュ）
・全面更新方式（バッファ無し）
・全面更新方式（バッファ面は外部フラッシュ）

ファームウェアアップデートモジュールのユーティリティツールとして、ファームウェアイメージを生成するツール（Renesas Image Generator）を提供します。Renesas Image Generator はファームウェアアップデートモジュールが使用する以下のイメージを生成することができます。

・初期イメージ：初期設定時にフラッシュライタで書き込むイメージファイル (拡張子 mot)です 。イメージファイルはブートローダとアプリケーションプログラムで構成されます。

・更新イメージ：ファームウェアアップデート対象のイメージファイル(拡張子 rsu)

1.2　ファームウェアアップデートモジュールの構成

ファームウェアアップデートモジュールを組み込んだファームウェアアップデートプログラム及びブートローダのモジュール構成を図 1-1 に、使用するモジュールの一覧を表 1-1 に示します。

通信インタフェースを介して受信した更新イメージは、ファームウェアアップデートモジュールと Flash driver を介してターゲットデバイス上の内蔵フラッシュメモリにセルフプログラミングされます。

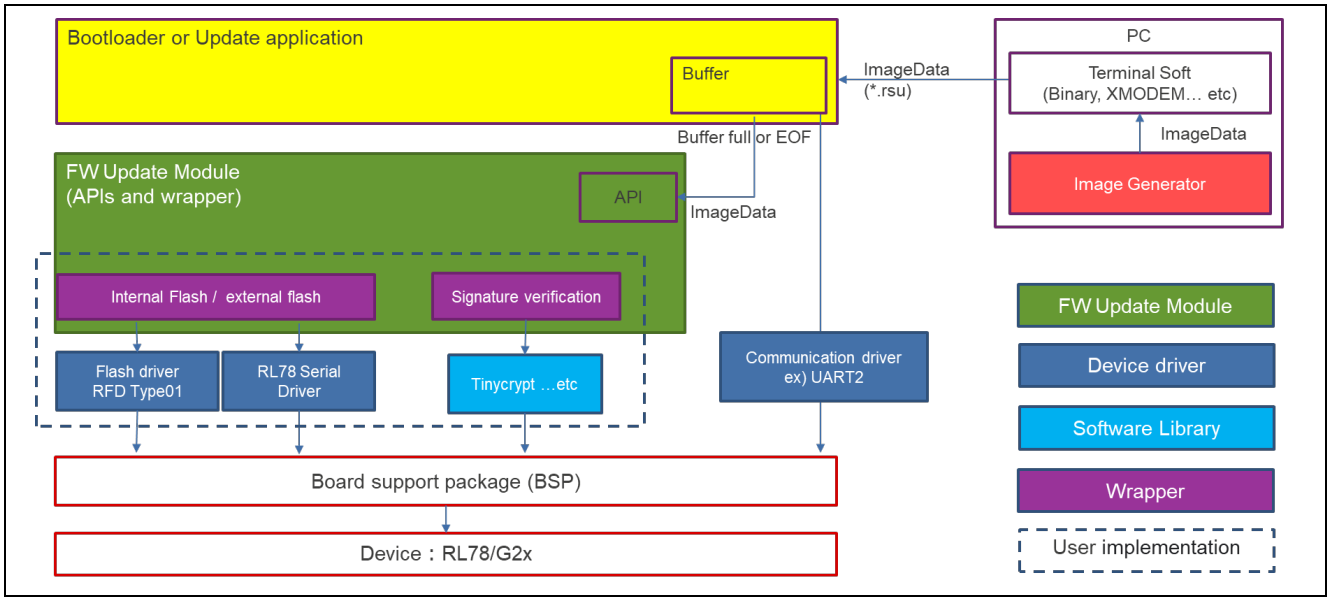


図 1-1 ブートローダ及びファームウェアアップデートプログラムのモジュール構成

表 1-1 ブートローダ及びファームウェアアップデートプログラムで使用する外部モジュール一覧

機能	モジュール名	備考
BSP	r_bsp	スマート・コンフィグレータによる自動生成
UART	r_Config_UART1：RL78/G22、RL78/G24 r_Config_UART2：RL78/G23	スマート・コンフィグレータによる自動生成
PORT	r_Conifg_PORT	スマート・コンフィグレータによる自動生成
FLASH	RFD RL78 Type01	ラッパー関数に実装
CSI	RL78_Serial	ラッパー関数に実装
Serial Flash	r_qspi_flash_mx25l	ラッパー関数に実装
Crypt library	Tinycrypt	ラッパー関数に実装

1.3　各ファームウェアアップデート方式について

RL78 ファミリのファームウェアアップデートモジュールでは、更新対象のファームウェア（更新イメージ）をバッファ面に一旦格納する方式とメイン面に直接書き込む方式を提供します。バッファ面は内蔵フラッシュメモリもしくは外部フラッシュメモリに設けることが可能です。

・メイン面：起動対象のイメージを格納するエリア

・バッファ面：更新対象のイメージを格納するエリア

更新イメージを直接メイン面に書き込む方式は、内蔵フラッシュメモリ全体をメイン面にすることができますが、バッファ面がないため、アップデートが失敗したときに更新前のファームウェアに戻すことはできません。

1.3.1　半面更新方式（バッファ面は内部フラッシュ）

更新イメージを内蔵フラッシュメモリのバッファ面に一旦格納し、その正当性を検証した後でメイン面にコピーします。

この方式では、ファームウェアアップデート機能をアプリケーションプログラムに持たせることができます。

また、メイン面へのコピーの前にファームウェアアップデートに失敗した場合、メイン面に存在する更新前のイメージを起動してファームウェアアップデートをやり直すことができます。

アプリケーションプログラムを格納できるサイズは、内蔵フラッシュメモリにメイン面、バッファ面を設けるため、内蔵フラッシュメモリからブートローダを引いた残りのサイズの半分になります。

1.3.1.1　半面更新方式（バッファ面は内部フラッシュ）のアップデート動作

更新イメージをバッファ面に一旦格納する方式であり、内蔵フラッシュメモリを分割してメイン面とバッファ面を設定します。更新イメージをバッファ面に格納し、バッファ面からメイン面にコピーする事によりファームウェアアップデートを行います。

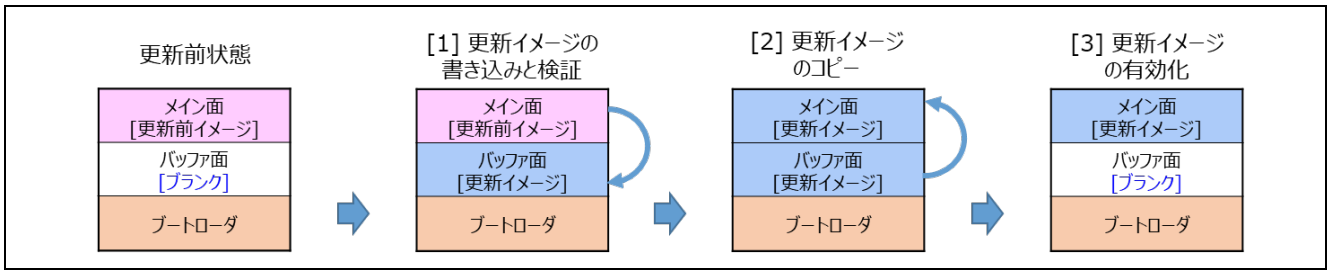


図 1-2 半面更新方式（バッファ面は内部フラッシュ）のアップデートの動作

[1] 更新イメージの書き込みと検証
メイン面の更新前イメージ（アプリケーションプログラム）により、更新イメージをバッファ面に書き込み検証する。

[2] 更新イメージのコピー
検証結果が正常であればリセットし、ブートローダによりメイン面を消去後、バッファ面からメイン面に更新イメージをコピーする。

[3] 更新イメージの有効化
ブートローダにより、バッファ面を消去します。
（デモプログラムでは、バッファ面の消去を行っておりません。ロールバックの対策等で、更新前のイメージを消去する必要が有る場合は、バッファ面のイメージの消去処理を追加してください）

1.3.2　全面更新方式（バッファ無し）

更新イメージをメイン面に書き込み、その後、その正当性検証を行います。

この方式ではファームウェアアップデート機能をブートローダに持たせる必要があります。そのため、ファームウェアアップデートに失敗した場合、ブートローダの機能を使ってファームウェアアップデートをやり直します。ファームウェアアップデートに成功するまで、アプリケーションプログラムの機能を利用することはできません。

アプリケーションプログラムを格納できるサイズは、内蔵フラッシュメモリにメイン面しか設けないため、内蔵フラッシュメモリからブートローダを引いた残りのサイズになります。

1.3.2.1　全面更新方式（バッファ無し）のアップデート動作

更新イメージを直接メイン面に書き込む方式であり、内蔵フラッシュメモリ全体をメイン面にすることができます。バッファ面がないため、アップデートが失敗したときに他の方式の様に更新前のファームウェアに戻すことはできません。

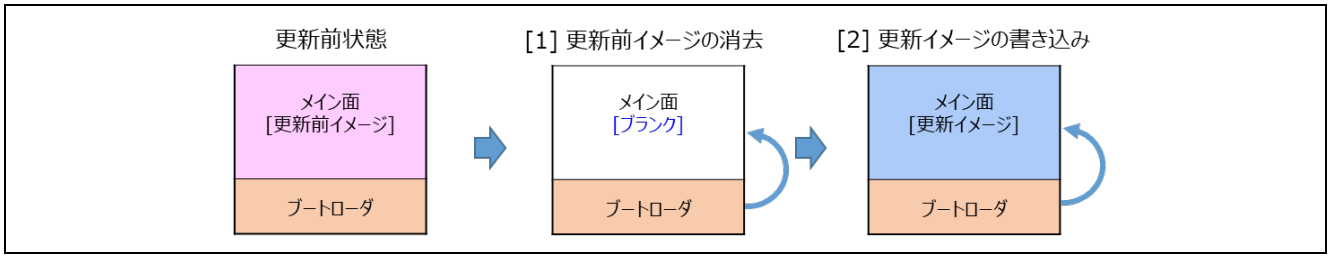


図 1-3 全面更新方式のアップデートの動作

[1] 更新前イメージの消去
メイン面の更新前イメージ（アプリケーションプログラム）により、メイン面の更新を示すデータを設定し、リセットする。その後、ブートローダが起動し、更新を示すデータを確認しメイン面の更新前イメージを消去する。

[2] 更新イメージの書き込み
ブートローダにより、外部から更新イメージをダウンロードし、メイン面に書き込む。書き込んだ更新イメージを検証し、検証結果が正常であればイメージを起動する。

1.3.3　全面更新方式（バッファ面は外部フラッシュ）

更新イメージを内蔵フラッシュメモリのバッファ面に一旦格納し、その正当性を検証した後でメイン面にコピーします。

この方式では、ファームウェアアップデート機能をアプリケーションプログラムに持たせることができます。

また、メイン面へのコピーの前にファームウェアアップデートに失敗した場合、メイン面に存在する更新前のイメージを起動してファームウェアアップデートをやり直すことができます。

アプリケーションプログラムを格納できるサイズは、内蔵フラッシュメモリにメイン面しか設けないため、内蔵フラッシュメモリからブートローダを引いた残りのサイズになります。

1.3.3.1　全面更新方式（バッファ面は外部フラッシュ）のアップデート動作

更新イメージをバッファ面に一旦格納する方式であり、内蔵フラッシュにメイン面、外部フラッシュにバッファ面を設定します。

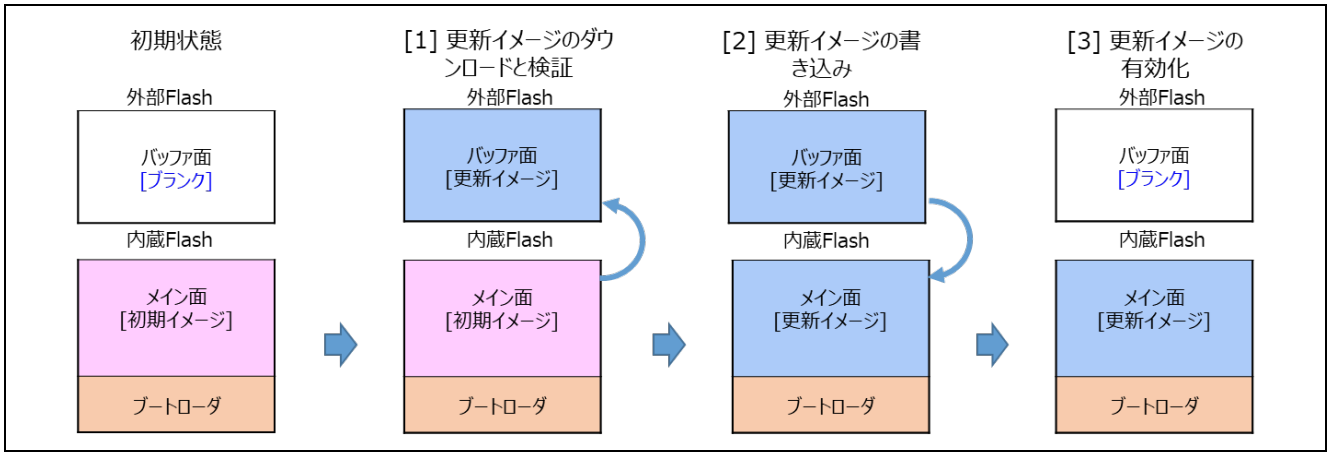


図 1-4 全面更新方式（バッファ面は外部フラッシュ）のファームウェアアップデートの動作

[1] 更新イメージのダウンロードと検証
メイン面のアプリケーションプログラム（初期イメージ）で更新イメージをバッファ面に書き込み検証する。

[2] 更新イメージの書き込み
検証結果が正常であれば、バッファ面からメイン面に更新イメージをコピーする。

[3] 更新イメージの有効化
ブートローダによりバッファ面を消去します。
（デモプログラムでは、バッファ面の消去を行っておりません。ロールバックの対策等で、更新前のイメージを消去する必要が有る場合は、バッファ面のイメージの消去処理を追加してください）

1.4　ファームウェアアップデートの初期状態

ファームウェアアップデートモジュールを使用したファームウェアアップデートシステムを初期状態に設定するには、Renesas Image Generator にて生成した初期イメージをフラッシュライタ等で内蔵フラッシュメモリに書き込むことで構築します。

また、別の方法として、最初にブートローダのみをフラッシュライタ等で書き込み、その後、ブートローダの機能でアプリケーションプログラムの更新イメージを書き込むことでも構築可能です。

1.4.1　Renesas Image Generator を用いた半面更新方式の初期状態の構築方法

Renesas Image Generator を用いた半面更新方式の初期状態の構築の流れを以下に示します。

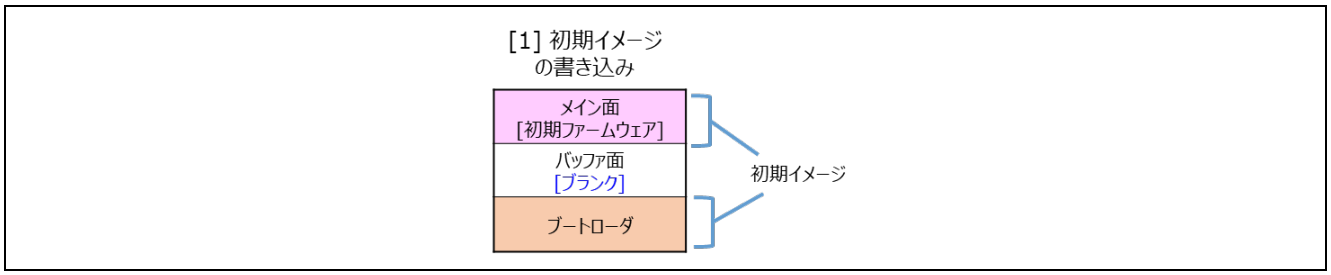


図 1-5 Renesas Image Generator を用いた初期状態の構築（半面更新方式の例）

[1] 初期イメージの書込み

Renesas Image Generator にて生成した初期イメージをフラッシュライタ等で内蔵フラッシュメモリに書き込む。

1.4.2　Renesas Image Generator を用いた全面更新方式の初期状態の構築方法

Renesas Image Generator を用いた全面更新方式の初期状態の構築の流れを以下に示します。

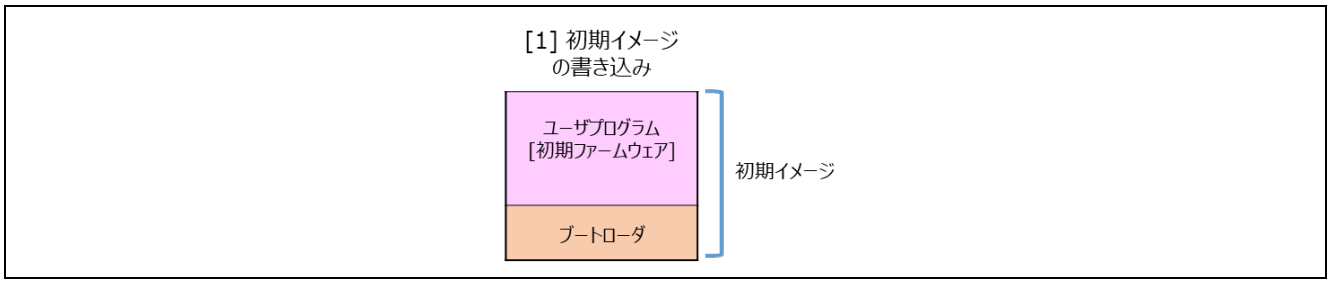


図 1-6 Renesas Image Generator を用いた初期状態の構築（全面更新方式の例）

[1] 初期イメージの書込み

Renesas Image Generator にて生成した初期イメージをフラッシュライタ等で内蔵フラッシュメモリに書き込む。

1.4.3　ブートローダを用いた半面更新方式の初期状態の構築方法

ブートローダを用いた半面更新方式の初期状態の構築の流れを以下に示します。

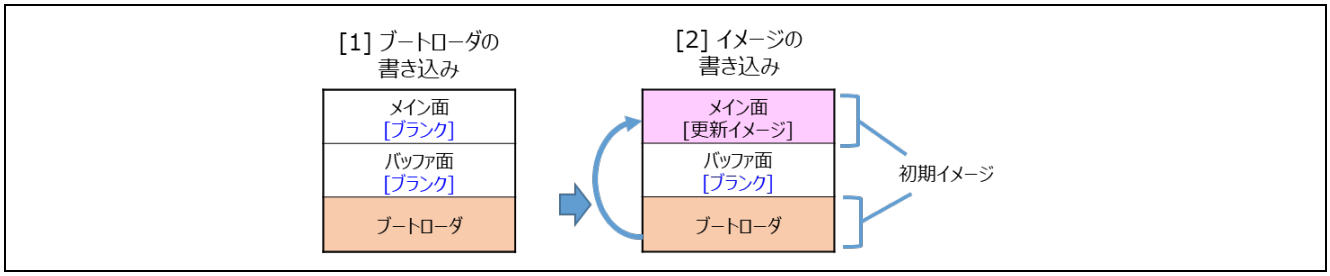


図 1-7 ブートローダを活用した初期状態の構築（半面更新方式）

[1] ブートローダの書き込み
ブートローダ（ブートローダのプロジェクトをビルドし生成された mot ファイル）をフラッシュライタ等で内蔵フラッシュメモリに書き込む。

[2] イメージの書き込み
ブートローダの機能を使って外部から更新イメージ（Renesas Image Generator で生成）をダウンロードし、メイン面に書き込む。書き込んだイメージを検証し、検証結果が正常であれば終了する。

1.4.4　ブートローダを用いた全面更新方式の初期状態の構築方法

ブートローダを用いた半面更新方式の初期状態の構築の流れを以下に示します。

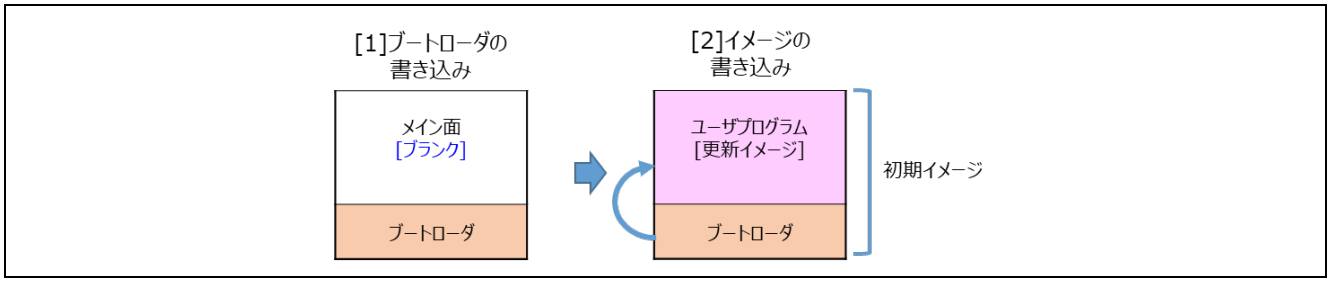


図 1-8 ブートローダを活用した初期状態の構築（全面更新方式）

[1] ブートローダの書き込み
ブートローダ（ブートローダのプロジェクトをビルドし生成された mot ファイル）をフラッシュライタ等で内蔵フラッシュメモリに書き込む。

[2] イメージの書き込み
ブートローダの機能を使って外部から更新イメージ（Renesas Image Generator で生成）をダウンロードし、メイン面に書き込む。書き込んだイメージを検証し、検証結果が正常であれば終了する。

1.5　パッケージ構成

ファームウェアアップデートモジュールのパッケージには、ソフトウェアやツールを含むいくつかのファイルが含まれています。次の表は、それらの内容を示しています。

表 1-2 ファームウェアアップデートモジュールパッケージのフォルダ構成

フォルダ名	説明
r01an6374xx0202-rl78g23-fwupdate.zip¥	
├──Demos	サンプルプロジェクト
│　└──rl	
│　　├──modules	ドライバとライブラリ
│　　│　├──3rd_party	
│　　│　│　└──tinycrypt	Crypt ライブラリ
│　　│　├──etc	
│　　│　│　└──base64	Base64 decode
│　　│　├──flash	Flash ドライバ
│　　│　├──rl78_serial	シリアルドライバ
│　　│　└──r_qspi_flash_mx25l	MX23L ドライバ
│　　├──rl78g22-fpb	RL78/G22-48p FPB 向け
│　　│　└──linear	
│　　│　　├──e2_ccrl	CC-RL 版
│　　│　　│　├──boot_loader	ブートローダ
│　　│　　│　└──fwup_leddemo	LED 点滅アプリケーション
│　　│　　└──iar	IAR 版
│　　│　　　├──boot_loader	ブートローダ
│　　│　　　└──fwup_leddemo	LED 点滅アプリケーション
│　　├──rl78g23-fpb	RL78/G23-128p FPB 向け
│　　│　└──linear	
│　　│　　├──e2_ccrl	CC-RL 版
│　　│　　│　├──boot_loader	ブートローダ
│　　│　　│　├──fwup_leddemo	LED 点滅アプリケーション
│　　│　　│　└──fwup_main	FW アップデートを含むユーザアプリ
│　　│　　└──iar	IAR 版
│　　│　　　├──boot_loader	ブートローダ
│　　│　　　├──fwup_leddemo	LED 点滅アプリケーション
│　　│　　　└──fwup_main	FW アップデートを含むユーザアプリ
│　　└──rl78g24-fpb	RL78/G24-64p FPB 向け
│　　　└──linear	
│　　　　├──e2_ccrl	CC-RL 版
│　　　　│　├──boot_loader	ブートローダ
│　　　　│　├──fwup_leddemo	LED 点滅アプリケーション
│　　　　│　└──fwup_main	FW アップデートを含むユーザアプリ
│　　　　└──iar	IAR 版
│　　　　　├──boot_loader	ブートローダ
│　　　　　├──fwup_leddemo	LED 点滅アプリケーション
│　　　　　└──fwup_main	FW アップデートを含めユーザアプリ
└──Modules	ファームウェアアップデートモジュール
│　├──r_config	コンフィグレーションファイル
│　└──r_fwup	ソースコード

フォルダ名	説明
└──RenesasImageGenerator	Renesas Image Generator （Python プログラムとパラメータファイル）
├──image-gen.py	Renesas Image Generator の Python プログラム
└──RL*78_xxxx*_ImageGenerator_PRM.csv	デモプロジェクト用パラメータファイル

1.6　API の概要

本モジュールに含まれる API 関数を表 1-3 に示します。

表 1-3 API 関数一覧

関数	関数説明
R_FWUP_Open	本モジュールをオープンします。
R_FWUP_Close	本モジュールのクローズ処理を行います。
R_FWUP_IsExistImage	指定エリアのイメージの存在を確認します。
R_FWUP_EraseArea	指定エリアを消去します。
R_FWUP_GetImageSize	イメージのサイズを取得します。
R_FWUP_WriteImage	イメージ（ヘッダ部＋プログラム部）を書き込みます。
R_FWUP_VerifyImage	イメージを検証します。
R_FWUP_ActivateImage	新しいイメージを有効にします。
R_FWUP_ExecImage	新しいイメージを起動します。
R_FWUP_SoftwareReset	ソフトウェアリセットを行います。
R_FWUP_SoftwareDelay	ソフトウェアディレイを行います。
R_FWUP_GetVersion	本モジュールのバージョン番号を返します。
R_FWUP_WriteImageHeader	イメージのヘッダ部を書き込みます。（特殊用途向け）※
R_FWUP_WriteImageProgram	イメージのプログラム部を書き込みます。（特殊用途向け）※

※ 特殊用途向けとは、FreeRTOS を使って OTA を行うお客様向けの仕様となります。FreeRTOS を使わず、ベアメタルで新規でファームウェアアップデートモジュール Rev2.0x を検討されているお客様は、この記載の項目は読み飛ばして頂いて問題ありません。

2. API 情報

本モジュールは下記の条件で動作を確認しています。

2.1 ハードウェアの要求

ご使用の MCU が以下の機能をサポートしている必要があります。

- フラッシュメモリ

2.2 ソフトウェアの要求

本モジュールは以下のドライバに依存しています。

- ボードサポートパッケージ(r_bsp)
- UART ドライバ (r_Config_UART)
- PORT ドライバ(r_Config_PORT)
- Renesas Flash Driver RL78 Type01 (RFD)
- Macronix International 社製 MX25/66L family serial NOR Flash Memory 制御ソフトウェア (r_qspi_flash_mx25l)
- シリアル・アレイ・ユニットの CSI モードを使ったクロック同期式シングルマスタ制御ソフトウェア(rl78_serial)

2.3 サポートされているツールチェーン

本モジュールは「6.1 動作確認環境」に示すツールチェーンで動作確認しています。

2.4 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_fwup_if.h に記載しています。

2.5 整数型

このドライバは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.6　コンパイル時の設定

本モジュールのコンフィグレーションオプションの設定は、r_fwup_config.h で行います。

オプション名および設定値に関する説明を表 2-1 コンフィグレーション設定に示します。

表 2-1 コンフィグレーション設定

Configuration options in r_fwup _config.h	
FWUP_CFG_UPDATE_MODE	アップデート方式を指定します。 0：RL78 では対象外のため設定不可 1：半面更新方式、バッファ面は内部フラッシュ 2：全面更新方式、バッファ無し 3：全面更新方式、バッファ面は外部フラッシュ
FWUP_CFG_FUNCTION_MODE	本モジュールの使用方法を指定します。 0：ブートローダ 1：アプリケーションプログラム
FWUP_CFG_MAIN_AREA_ADDR_L	メイン面の開始アドレスを設定します。
FWUP_CFG_BUF_AREA_ADDR_L	バッファ面（内蔵フラッシュ）の開始アドレスを設定します。
FWUP_CFG_AREA_SIZE	メイン面とバッファ面のサイズを設定します。
FWUP_CFG_CF_BLK_SIZE	内蔵コードフラッシュのブロックサイズを設定します。
FWUP_CFG_CF_W_UNIT_SIZE	内蔵コードフラッシュの書き込み単位を設定します。
FWUP_CFG_EXT_BUF_AREA_ADDR_L	バッファ面（外部フラッシュ）の開始アドレスを設定します。
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	外部フラッシュのブロックサイズもしくはセクタサイズを設定します。
FWUP_CFG_DF_ADDR_L	データフラッシュの開始アドレスを設定します。
FWUP_CFG_DF_BLK_SIZE	データフラッシュのブロックサイズを設定します。
FWUP_CFG_DF_NUM_BLKS	データフラッシュのブロック数を設定します。 データフラッシュがない場合は 0 を設定してください。
FWUP_CFG_FWUPV1_COMPATIBLE	FWUP V1 互換設定を指定します。（特殊用途向け） 0：Disable（通常向けの設定） 1：Enable（特殊用途向けの設定）
FWUP_CFG_SIGNATURE_VERIFICATION	検証方式を指定します。 0：ECDSA＋SHA256 1：SHA256
FWUP_CFG_PRINTF_DISABLE	ログ表示設定を指定します。 0：Enable 1：Disable

2.7　サンプルプロジェクトのコードサイズ

本アプリケーションノートのパッケージに含まれるサンプルプロジェクトの ROM、RAM、最大使用スタックサイズを下表に示します。下表の値は以下の条件で確認しています。

モジュールリビジョン：ファームウェア アップデート モジュール for RL78 v2.0.1

コンパイラバージョン：Renesas Electronics C Compiler Package for RL78 Family V1.11

IAR C/C++ Compiler for Renesas RL78 version 5.10.1

コンフィグレーションオプション：コンフィグレーションオプション設定は FPB ごとに記載

CC-RL

・最適化レベル：サイズ＆実行速度(-Odefault)

・一度も参照のない変数／関数を削除する(-optimize=symbol_delete)

IAR

・最適化レベル：高(バランス)

2.7.1　RL78/G23-128p FPB 用のサンプルプロジェクト

RL78/G23-128p FPB のサンプルプロジェクトのコンフィグレーション設定:

FWUP_CFG_UPDATE_MODE　　1：Single bank with buffer. (default)

FWUP_CFG_SIGNATURE_VERIFICATION 0：ECDSA. (default)

表 2-2 サンプルプロジェクト(boot_loader)の ROM、RAM、スタックサイズ

項目	分類	使用メモリ（単位：byte）	
		CC-RL	IAR Compiler
boot_loader	ROM	21230	30358
	RAM	1343	3660
	スタック	516	3152

表 2-3 サンプルプロジェクト(fwup_main)の ROM、RAM、スタックサイズ

項目	分類	使用メモリ（単位：byte）	
		CC-RL	IAR Compiler
fwup_main	ROM	18142	28095
	RAM	837	3658
	スタック	516	2198

2.7.2　RL78/G24-64p FPB 用のサンプルプロジェクト

RL78/G23-128p FPB のサンプルプロジェクトのコンフィグレーション設定:

FWUP_CFG_UPDATE_MODE　　1：Single bank with buffer. (default)

FWUP_CFG_SIGNATURE_VERIFICATION 0：ECDSA. (default)

表 2-4 サンプルプロジェクト(boot_loader)の ROM、RAM、スタックサイズ

項目	分類	使用メモリ（単位：byte）	
		CC-RL	IAR Compiler
boot_loader	ROM	21541	30648
	RAM	1343	3669
	スタック	516	3152

表 2-5 サンプルプロジェクト(fwup_main)の ROM、RAM、スタックサイズ

項目	分類	使用メモリ（単位：byte）	
		CC-RL	IAR Compiler
fwup_main	ROM	18920	28392
	RAM	837	3667
	スタック	516	2198

2.7.3　RL78/G22-48p FPB 用のサンプルプロジェクト

RL78/G22-48p FPB のサンプルプロジェクトのコンフィグレーション設定:

FWUP_CFG_UPDATE_MODE　　2：Single bank without buffer.

FWUP_CFG_SIGNATURE_VERIFICATION 1：SHA256

表 2-6 サンプルプロジェクト(boot_loader)の ROM、RAM、スタックサイズ

項目	分類	使用メモリ（単位：byte）	
		CC-RL	IAR Compiler
boot_loader	ROM	11807	15915
	RAM	767	2054
	スタック	402	1956

2.8 引数

API 関数の引数を示します。この列挙型は API 関数のプロトタイプ宣言とともに r_fwup_if.h で記載されています。

```
typedef enum fwup_area
{
    FWUP_AREA_MAIN = 0,
    FWUP_AREA_BUFFER,
    FWUP_AREA_DATA_FLASH
} e_fwup_area_t;

typedef enum e_fwup_delay_units
{
    FWUP_DELAY_MICROSECS = 0,
    FWUP_DELAY_MILLISECS,
    FWUP_DELAY_SECS
} e_fwup_delay_units_t;
```

2.9 戻り値

API 関数の戻り値を示します。この列挙型は API 関数のプロトタイプ宣言とともに r_fwup_if.h で記載されています。

```
typedef enum fwup_err
{
    FWUP_SUCCESS = 0,              // Normally terminated.
    FWUP_PROGRESS,                 // Firmware update is in progress.
    FWUP_ERR_FLASH,                // Detect error of flash module.
    FWUP_ERR_VERIFY,               // Verify error.
    FWUP_ERR_FAILURE,              // General error.
} e_fwup_err_t;
```

2.10 API の実装例について

各ファームウェアアップデート方式に対応したブートローダ及びアプリケーションプログラムの実装例を示します。

詳細は本アプリケーションノートのパッケージに含まれるデモプロジェクトのソースコードをご確認ください。図 2-1 半面/全面更新方式（バッファ面あり）のブートローダ実装例

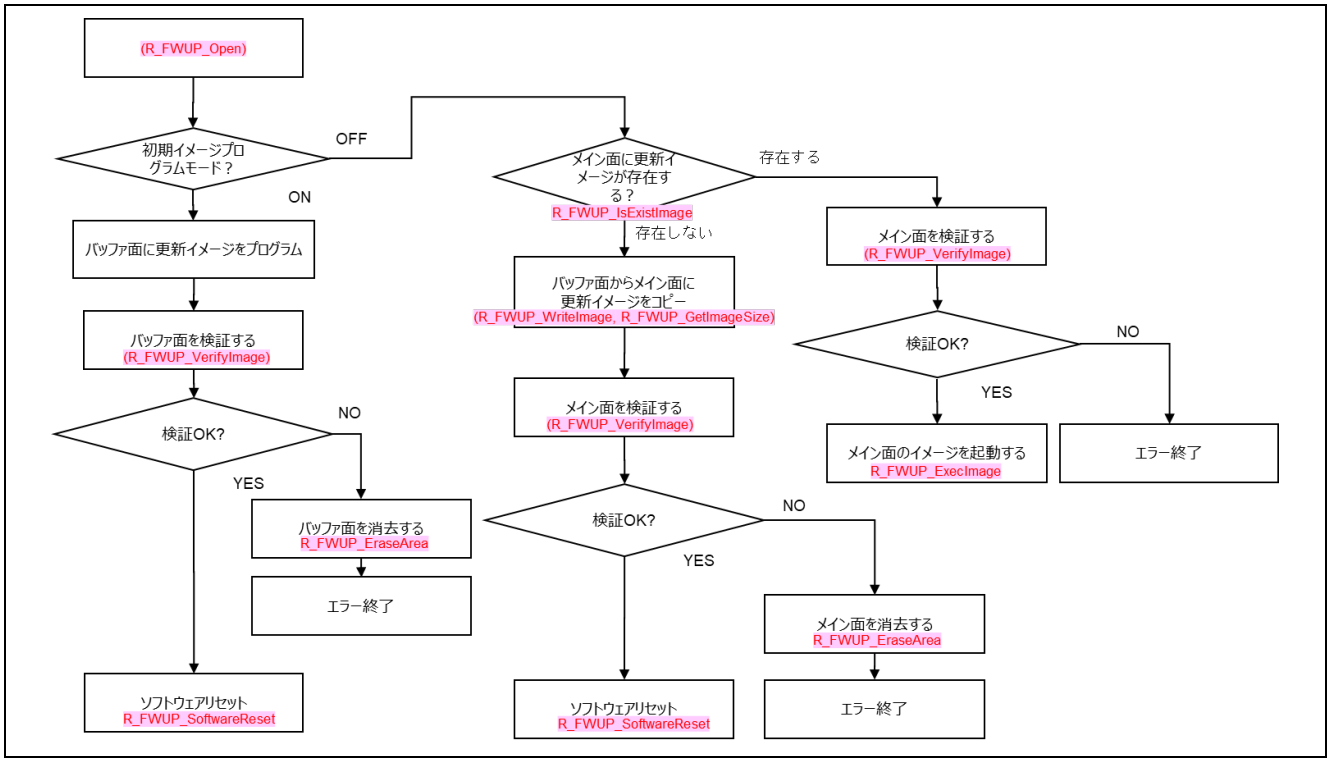


図 2-1 半面/全面更新方式（バッファ面あり）のブートローダ実装例

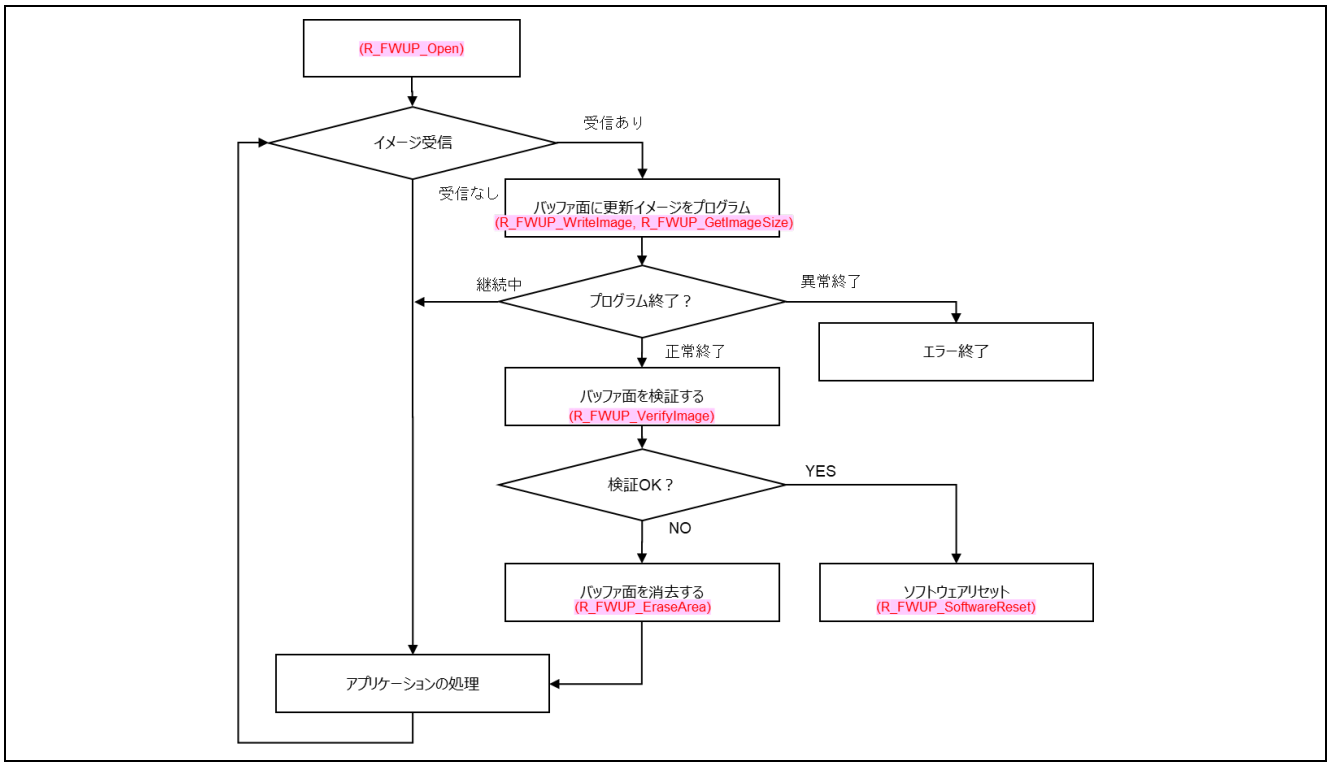


図 2-2 半面/全面更新方式（バッファ面あり）のアプリケーションプログラム実装例

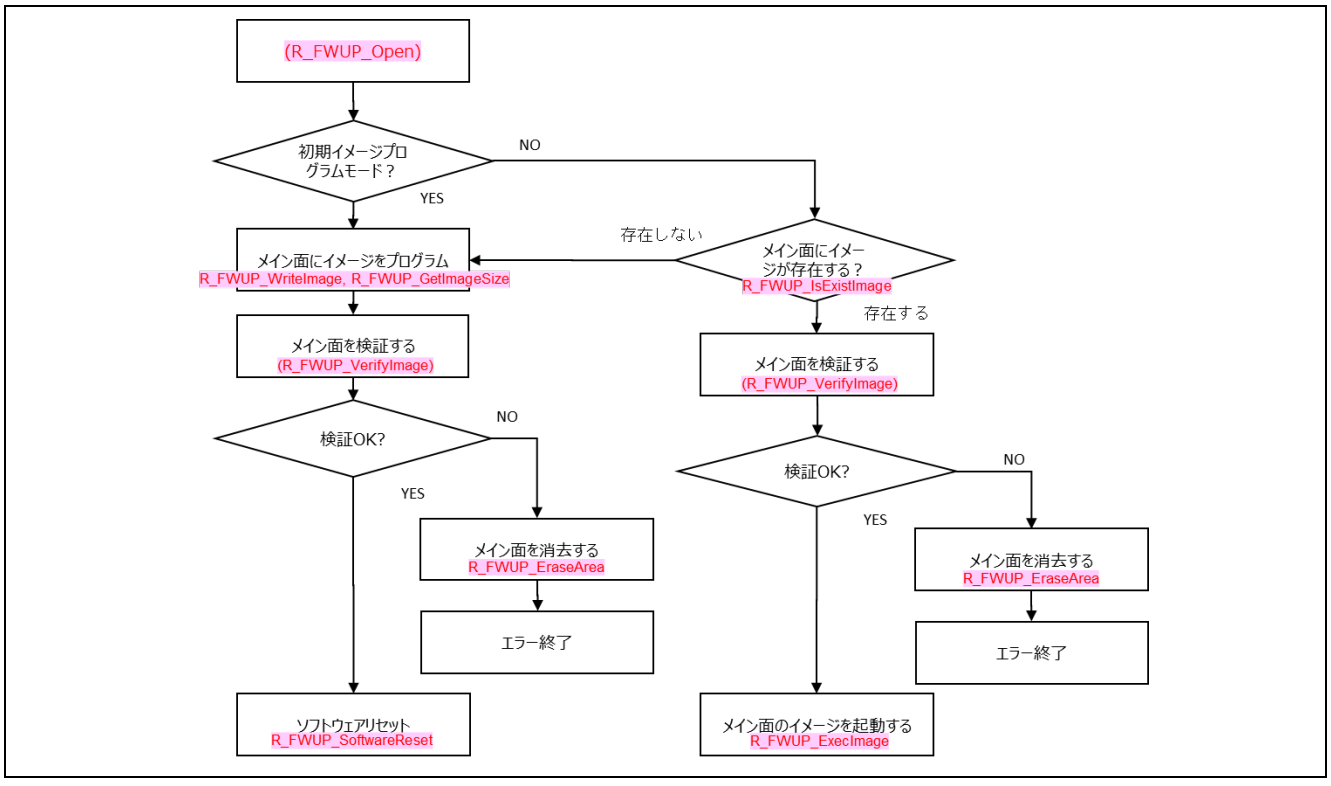


図 2-3 全面更新方式（バッファ無し）のブートローダ実装例

3. API 関数

3.1 R_FWUP_Open 関数

表 3-1 R_FWUP_Open 関数仕様

Format	e_fwup_err_t R_FWUP_Open (void)	
Description	本モジュールのオープン処理を行います。 フラッシュモジュールのオープン処理を実施します。	
Parameters	なし	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.2 R_FWUP_Close 関数

表 3-2 R_FWUP_Close 関数仕様

Format	void R_FWUP_Close (void)
Description	本モジュールのクローズ処理を行います。 フラッシュモジュールのクローズ処理を実施します。
Parameters	なし
Return Values	なし
Special Notes	ー

3.3 R_FWUP_IsExistImage 関数

表 3-3 R_FWUP_IsExistImage 関数仕様

Format	bool R_FWUP_IsExistImage(e_fwup_area_t area)	
Description	指定エリアのイメージの存在を確認します。	
Parameters	area：メイン面（FWUP_AREA_MAIN）、バッファ面（FWUP_AREA_BUFFER）	
Return Values	true	イメージが存在する
	false	イメージが存在しない
Special Notes	処理上で使用する RSU ヘッダ領域のマジックコードが正しく書き込まれていることを確認します。	

3.4　R_FWUP_EraseArea 関数

表 3-4 R_FWUP_EraseArea 関数仕様

Format	e_fwup_err_t R_FWUP_EraseArea(e_fwup_area_t area)	
Description	指定エリアを消去します。	
Parameters	area：メイン面（FWUP_AREA_MAIN）、バッファ面（FWUP_AREA_BUFFER）、データフラッシュ（FWUP_AREA_DATA_FLASH）	
Return Values	FWUP_SUCCES	正常終了
	FWUP_ERR_FLASH	FLASH モジュールエラー
Special Notes	メイン面の消去はブートローダのみ実行可能です。	

3.5　R_FWUP_GetImageSize 関数

表 3-5 R_FWUP_GetImageSize 関数仕様

Format	uint32_t R_FWUP_GetImageSize(void)	
Description	イメージのバイトサイズを返します。 本関数はの RSU ヘッダのアドレス情報を元にイメージのバイトサイズを取得します。そのため先に R_FWUP_WriteImage 関数や R_FWUP_WriteImageProgram 関数で RSU ヘッダアドレス情報をコードフラッシュに書き込んでください。	
Parameters	なし	
Return Values	0	取得中
	1 以上	イメージのサイズ
Special Notes	ー	

3.6　R_FWUP_WriteImage 関数

表 3-6 R_FWUP_WriteImage 関数仕様

Format	e_fwup_err_t R_FWUP_WriteImage(e_fwup_area_t area, uint8_t *p_buf, uint32_t buf_size)	
Description	指定エリアにイメージ（ヘッダ部＋プログラム部）を書き込みます。 イメージのサイズに達するまで本関数をコールします。 イメージのサイズは R_FWUP_GetImageSize()で取得します。	
Parameters	area：メイン面（FWUP_AREA_MAIN）、バッファ面（FWUP_AREA_BUFFER） p_buf：イメージ（ヘッダ＋プログラム）のバッファ buf_size：バッファサイズ※1	
Return Values	FWUP_SUCCES	全イメージの書込み完了
	FWUP_PROGRESS	全イメージの書き込み未完了（今回指定分の書込み完了）
	FWUP_ERR_FLASH	FLASH モジュールエラー※2
	FWUP_ERR_FAILURE	不正なパラメータ
Special Notes	※１：コードフラッシュ書き込み単位の倍数を設定してください。（例）64、128、256 　　また、このサイズはデータフラッシュにも適用されます。 ※２：本エラーが発生した場合、Flash メモリにて書き込みエラーが発生している可能性が高いため、一連の処理をイレースからやり直してください。 FWUP_CFG_FWUPV1_COMPATIBLE が有効の場合、処理上で使用する RSU ヘッダ領域のマジックコードには、FWUP V1 向けの"Renesas"を使用します。	

3.7　R_FWUP_VerifyImage 関数

表 3-7 R_FWUP_VerifyImage 関数仕様

Format	e_fwup_err_t R_FWUP_VerifyImage(e_fwup_area_t area)	
Description	本モジュールに組み込んだ暗号ライブラリを用いてイメージを検証します。	
Parameters	area：メイン面（FWUP_AREA_MAIN）、バッファ面（FWUP_AREA_BUFFER）	
Return Values	FWUP_SUCCES	検証成功
	FWUP_ERR_VERIFY	検証失敗
	FWUP_ERR_FAILURE	不正なパラメータ
Special Notes	ー	

3.8　R_FWUP_ActivateImage 関数

表 3-8 R_FWUP_ActivateImage 関数仕様

Format	e_fwup_err_t R_FWUP_ActivateImage(void)	
Description	新しいイメージを有効化します。 ・半面更新方式 -ブートローダ：バッファ面のイメージをメイン面にコピー -アプリケーションプログラム：何もせずに FWUP_SUCCESS を返す ・全面更新方式 -何もせず FWUP_SUCCESS を返す	
Parameters	なし	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールエラー
Special Notes	ー	

3.9　R_FWUP_ExecImage 関数

表 3-9 R_FWUP_ExecImage 関数仕様

Format	void R_FWUP_ExecImage(void)
Description	有効なイメージのプログラムを実行します。
Parameters	なし
Return Values	なし
Special Notes	ブートローダからアプリケーションへ遷移する際に割り込みを禁止します。

3.10 R_FWUP_SoftwareReset 関数

表 3-10 R_FWUP_SoftwareReset 関数仕様

Format	void R_FWUP_SoftwareReset(void)
Description	ソフトウェアリセット処理を実行します。
Parameters	なし
Return Values	なし
Special Notes	ー

3.11 R_FWUP_SoftwareDelay 関数

表 3-11 R_FWUP_SoftwareDelay 関数仕様

Format	uint32_t R_FWUP_SoftwareDelay(uint32_t delay, e_fwup_delay_units_t units)	
Description	ソフトウェアディレイ処理を実行します。	
Parameters	delay：ディレイ時間 units：単位（us、ms、sec）	
Return Values	0	正常終了
	Other	異常終了
Special Notes	ー	

3.12 R_FWUP_GetVersion 関数

表 3-12 R_FWUP_GetVersion 関数仕様

Format	uint32_t R_FWUP_GetVersion(void)
Description	本モジュールのバージョンを返します。
Parameters	なし
Return Values	バージョン番号
Special Notes	ー

3.13 R_FWUP_WriteImageHeader 関数

本関数は、ヘッダ情報とプログラム情報を分けて書き込む必要がある特殊用途向けの API となります。通常は、R_FWUP_WriteImage 関数を使用してください。

表 3-13 R_FWUP_WriteImageHeader 関数仕様

Format	e_fwup_err_t R_FWUP_WriteImageHeader (e_fwup_area_t area, uint8_t FWUP_FAR *p_sig_type, uint8_t FWUP_FAR *p_sig, uint32_t sig_size)	
Description	指定エリアのイメージのヘッダ部にブートローダが検証に使用するシグネチャを書き込みます。	
Parameters	area：メイン面（FWUP_AREA_MAIN）、バッファ面（FWUP_AREA_BUFFER） p_sig_type：署名タイプ　文字列“hash-sha256”もしくは“sig-sha256-ecdsa” p_sig：署名 sig_size：署名の長さ（64 を設定してください）	
Return Values	FWUP_SUCCES	書き込み終了
	FWUP_ERR_FLASH	FLASH モジュールエラー
	FWUP_ERR_FAILURE	不正なパラメータ
Special Notes	FWUP_CFG_FWUPV1_COMPATIBLE が有効の場合、処理上で使用する RSU ヘッダ領域のマジックコードには、FWUP V1 向けの"Renesas"を使用します。	

3.14 R_FWUP_WriteImageProgram 関数

本関数は、ヘッダ情報とプログラム情報を分けて書き込む必要がある特殊用途向けの API となります。通常は、R_FWUP_WriteImage 関数を使用してください。

表 3-14 R_FWUP_WriteImageProgram 関数仕様

Format	e_fwup_err_t R_FWUP_WriteImageProgram (e_fwup_area_t area, uint8_t *p_buf, uint32_t offset, uint32_t buf_size)	
Description	指定エリアにイメージのプログラム部を書き込みます。 イメージのプログラム部のサイズに達するまで本関数をコールします。 イメージのサイズは R_FWUP_GetImageSize()で取得します。 本関数はの RSU ヘッダのアドレス情報を元にオフセットによるプログラム書き込みを行います。そのため、本関数の最初の呼び出しでのオフセット（0x200）から 0x100 バイトのデータを必ず設定してください。 （引数としては、offset 引数に 0x200 を指定し、buf_size 引数には 0x100 以上を指定してください）	
Parameters	area：メイン面（FWUP_AREA_MAIN）、バッファ面（FWUP_AREA_BUFFER） p_buf：イメージのプログラム部のバッファ offset：オフセット※1 buf_size：バッファサイズ※2	
Return Values	FWUP_SUCCES	全イメージの書込み完了
	FWUP_PROGRESS	全イメージの書き込み未完了（今回指定分の書込み完了）
	FWUP_ERR_FLASH	FLASH モジュールエラー
	FWUP_ERR_FAILURE	不正なパラメータ
Special Notes	※１：オフセットは 0x200 以降を設定してください。 ※２：コードフラッシュ書き込み単位の倍数を設定してください。（例）64、128、256	

3.15 ラッパー関数

本ラッパー関数にフラッシュドライバと暗号処理を実装します。ソースファイルの下記コメント部分に処理を実装してください。実装方法についてはデモプロジェクトをご覧ください。

```
/**** Start user code ****/
/**** End user code   ****/
```

3.15.1 r_fwup_wrap_com.c、h

3.15.1.1 r_fwup_wrap_disable_interrupt 関数

表 3-15 r_fwup_wrap_disable_interrupt 関数仕様

Format	void r_fwup_wrap_disable_interrupt (void)
Description	割り込み禁止
Parameters	なし
Return Values	なし
Special Notes	ー

3.15.1.2 r_fwup_wrap_enable_interrupt 関数

表 3-16 r_fwup_wrap_disable_interrupt 関数仕様

Format	void r_fwup_wrap_enable_interrupt (void)
Description	割り込み許可
Parameters	なし
Return Values	なし
Special Notes	ー

3.15.1.3 r_fwup_wrap_software_reset 関数

表 3-17 r_fwup_wrap_software_reset 関数仕様

Format	void r_fwup_wrap_software_reset (void)	
Description	ソフトウェアリセット	
Parameters	なし	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.1.4 r_fwup_wrap_software_delay 関数

表 3-18 r_fwup_wrap_software_delay 関数仕様

Format	uint32_t r_fwup_wrap_software_delay (uint32_t delay, e_fwup_delay_units_t units)	
Description	ソフトウェアディレイ	
Parameters	delay：ディレイ時間 uinits：単位（us,ms,sec） FWUP_DELAY_MICROSECS FWUP_DELAY_MILLISECS FWUP_DELAY_SECS	
Return Values	0	正常終了
	Other	異常終了
Special Notes	ー	

3.15.2 r_fwup_wrap_flash.c、h

3.15.2.1 r_fwup_wrap_flash_open 関数

表 3-19 r_fwup_wrap_flash_open 関数仕様

Format	e_fwup_err_t r_fwup_wrap_flash_open (void)	
Description	内蔵フラッシュをオープンします。	
Parameters	なし	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.2 r_fwup_wrap_flash_close 関数

表 3-20 r_fwup_wrap_flash_close 関数仕様

Format	void r_fwup_wrap_flash_close (void)
Description	内蔵フラッシュをクローズします。
Parameters	なし
Return Values	なし
Special Notes	ー

3.15.2.3 r_fwup_wrap_flash_erase 関数

表 3-21 r_fwup_wrap_flash_erase 関数仕様

Format	e_fwup_err_t r_fwup_wrap_flash_erase (uint32_t addr, uint32_t num_blocks)	
Description	内蔵フラッシュをブロック単位で消去します。	
Parameters	addr num_blocks	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.4 r_fwup_wrap_flash_write 関数

表 3-22 r_fwup_wrap_flash_write 関数仕様

Format	e_fwup_err_t r_fwup_wrap_flash_write (uint32_t src_addr, uint32_t dest_addr, uint32_t num_bytes)	
Description	内蔵フラッシュにデータを書き込みます	
Parameters	src_addr:：書き込むデータのポインタ dest_addr：書き込み先のアドレス num_bytes：書き込みサイズ（バイト）	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.5 r_fwup_wrap_flash_read 関数

表 3-23 r_fwup_wrap_flash_read 関数仕様

Format	e_fwup_err_t r_fwup_wrap_flash_read (uint32_t buf_addr, uint32_t src_add, uint32_t size)	
Description	内蔵フラッシュを読み出します。	
Parameters	buf_addr：読みだしたデータを格納するバッファのアドレス src_addr：読み出すアドレス size：読みだすサイズ	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.6 r_fwup_wrap_bank_swap 関数

表 3-24 r_fwup_wrap_bank_swap 関数仕様

Format	e_fwup_err_t r_fwup_wrap_bank_swap (void)	
Description	バンクスワップ処理を行います。	
Parameters	なし	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	R バンク搭載機種のみ（RL78 には未搭載）	

3.15.2.7 r_fwup_wrap_ext_flash_open 関数

表 3-25 r_fwup_wrap_ext_flash_open 関数仕様

Format	e_fwup_err_t r_fwup_wrap_ext_flash_open (void)	
Description	外部フラッシュをオープンします。	
Parameters	なし	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.8 r_fwup_wrap_ext_flash_close 関数

表 3-26 r_fwup_wrap_ext_flash_close 関数仕様

Format	void r_fwup_wrap_ext_flash_close (void)
Description	外部フラッシュをクローズします。
Parameters	なし
Return Values	なし
Special Notes	ー

3.15.2.9 r_fwup_wrap_ext_flash_erase 関数

表 3-27 r_fwup_wrap_ext_flash_erase 関数仕様

Format	e_fwup_err_t r_fwup_wrap_ext_flash_erase (uint32_t offsetadd, uint32_t num_sectors)	
Description	外部フラッシュをセクタ単位で消去します。	
Parameters	offsetadd：消去するセクタの開始アドレス num_sectors：セクタ数	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.10　r_fwup_wrap_ext_flash_write 関数

表 3-28 r_fwup_wrap_ext_flash_write 関数仕様

Format	e_fwup_err_t r_fwup_wrap_ext_flash_write (uint32_t src_addr, uint32_t dest_addr, uint32_t num_bytes);	
Description	外部フラッシュにデータを書き込みます。	
Parameters	src_addr:：書き込むデータのポインタ dest_addr：書き込み先のアドレス num_bytes：書き込みサイズ（バイト）	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.2.11　r_fwup_wrap_ext_flash_read 関数

表 3-29 r_fwup_wrap_ext_flash_read 関数仕様

Format	e_fwup_err_t r_fwup_wrap_ext_flash_read (uint32_t buf_addr, uint32_t src_addr, uint32_t size);	
Description	外部フラッシュを読み出します。	
Parameters	buf_addr：読みだしたデータを格納するバッファのアドレス src_addr：読み出すアドレス size：読みだすサイズ	
Return Values	FWUP_SUCCESS	正常終了
	FWUP_ERR_FLASH	FLASH モジュールのエラー
Special Notes	ー	

3.15.3 r_fwup_wrap_verify.c、h

3.15.3.1 r_fwup_wrap_sha256_init 関数

表 3-30 r_fwup_wrap_sha256_init 関数仕様

Format	int32_t r_fwup_wrap_sha256_init (void *vp_ctx);	
Description	ハッシュ値の計算を開始します	
Parameters	vp_ctx：暗号ライブラリのコンテキストのポインタ	
Return Values	0	正常終了
	Other	異常終了
Special Notes	―	

3.15.3.2 r_fwup_wrap_sha256_update 関数

表 3-31 r_fwup_wrap_sha256_update 関数仕様

Format	int32_t r_fwup_wrap_sha256_update (void *vp_ctx, const uint8_t *p_data, uint32_t datalen)	
Description	指定範囲のハッシュ値を計算します	
Parameters	vp_ctx：暗号ライブラリのコンテキストのポインタ p_data：開始アドレス datalen：データ長（バイト）	
Return Values	0	正常終了
	Other	異常終了
Special Notes	―	

3.15.3.3 r_fwup_wrap_sha256_final 関数

表 3-32 r_fwup_wrap_sha256_final 関数仕様

Format	int32_t r_fwup_wrap_sha256_final (uint8_t *p_hash, void *vp_ctx)	
Description	ハッシュ値の計算を終了し、ハッシュ値を返します	
Parameters	p_hash：計算したハッシュ値を格納するバッファのポインタ vp_ctx：暗号ライブラリのコンテキストのポインタ	
Return Values	0	正常終了
	Other	異常終了
Special Notes	―	

3.15.3.4 r_fwup_wrap_verify_ecdsa 関数

表 3-33 r_fwup_wrap_verify_ecdsa 関数仕様

Format	int32_t r_fwup_wrap_verify_ecdsa (uint8_t *p_hash, uint8_t *p_sig_type, uint8_t *p_sig, uint32_t sig_size)	
Description	ECDSA で検証を行います。	
Parameters	p_hash：ハッシュ値が格納されているバッファのポインタ p_sig_type：シグネチャタイプ p_sig：シグネチャ sig_size：シグネチャサイズ	
Return Values	0	正常終了
	Other	異常終了
Special Notes	―	

3.15.3.5 r_fwup_wrap_get_crypt_context 関数

表 3-34 r_fwup_wrap_ get_crypt_context 関数仕様

Format	void * r_fwup_wrap_get_crypt_context (void);	
Description	暗号ライブラリのコンテキストのポインタを返します。	
Parameters	なし	
Return Values	void *	暗号ライブラリのコンテキストのポインタ
Special Notes	―	

4. デモプロジェクト

本デモプロジェクトは、シリアル通信インタフェース（SCI）を用いたファームウェアアップデートのデモを実施するためのサンプルプログラムです。

4.1 デモプロジェクトの構成

デモプロジェクトは、本モジュールとその他の依存するモジュール、ファームウェアアップデートデモを実施するための main()関数で構成されます。本パッケージでは、1.5 に示すデバイスとコンパイラに対応したデモプロジェクトを提供しています。

本ファームウェアアップデートのデモは、以下のプロジェクトで構成されています。

半面更新方式のフォルダ構成：□□¥linear¥△△¥ の下
全面更新方式のフォルダ構成：□□¥linear¥△△¥ の下

□□：デバイス名
△△：コンパイラ（e2_ccrl/iar）

・boot_loader：ブートローダ

リセット後に最初に実行され、アプリケーションプログラムが改ざんされていないことを検証し、問題無ければアプリケーションプログラムを起動するプログラムです。

・fwup_main：アプリケーションプログラム

更新ファームウェアをダウンロードし、署名検証を行うアプリケーションプログラム（初期ファームウェア）です。

・fwup_leddemo：アプリケーションプログラム（更新用）

LED を点滅させるアプリケーションプログラム（更新用）です。

4.2　動作環境準備

ファームウェアアップデートのデモプロジェクトを実行するには、WindowsPC にツールをインストール（4.2.1～4.2.4 参照）する必要があります。また、WindowsPC とターゲットボードを接続する USB シリアル変換ボード（4.2.5 参照）を使用します。

4.2.1　TeraTerm のインストール

WindowsPC からターゲットボードへのシリアル通信により、ファームウェアアップデートのイメージを転送するために使用します。デモプロジェクトでは、TeraTerm 4.105 で動作確認を実施しています。

インストール後は、シリアルポートの通信設定を表 **4-1** の様に設定してください。

表 4-1 通信仕様

項目	内容
通信方式	調歩同期式通信
ビットレート	115200bps
データ長	8 ビット
パリティ	なし
ストップビット	1 ビット
フロー制御	CTS/RTS

4.2.2　Python 実行環境のインストール

Renesas Image Generator (image-gen.py) で初期イメージと更新イメージを作成するために使用します。

Renesas Image Generator は、ECDSA により署名データを生成します。デモプロジェクトでは、Python 3.9.0 で環境動作確認を行っています。

Python 3.9.0 以上をインストールしてください。

また、Python の暗号化ライブラリ（pycryptodome）を使用しますので、Python をインストール後、コマンドプロンプトから以下の pip コマンドを実行し、ライブラリのインストールを行って下さい。

```
pip install pycryptodome
```

4.2.3　OpenSSL の実行環境のインストール

初期イメージおよび更新イメージの ECDSA 署名データの生成や検証を行うために必要な鍵の生成に OpenSSL を使用します。以下の URL から OpenSSL インストーラをダウンロードして、インストールします。Light 版で問題ありません。

https://slproweb.com/products/Win32OpenSSL.html

4.2.4　フラッシュライタのインストール

初期イメージを書き込むためのフラッシュライタが必要です。

デモプロジェクトでは Renesas Flash Programmer v3.11.01 を使用しています。

Renesas Flash Programmer (Programming GUI) | Renesas

4.2.5　USB シリアル変換ボード

WindowsPC からターゲットボードへのシリアル通信により、ファームウェアアップデートのイメージを転送するために使用します。

ターゲットボードとの接続方法については、該当するターゲットボードの動作確認環境（0）を参照してください。

Pmod USBUART（DIGILENT 製）を使用します。

https://reference.digilentinc.com/reference/pmod/pmodusbuart/start

4.3　実行環境準備

4.3.1　署名生成/検証用鍵の生成

鍵の生成に OpenSSL を使用します。事前に OpenSSL のインストール（4.2.3）を実施してください。

以下の OpenSSL コマンドを実行することで、イメージの署名生成と署名検証に使用する楕円曲線暗号(secp256r1)の鍵ペアを生成し、秘密鍵と公開鍵を抽出します。

```
>openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1

>openssl ec -in secp256r1.keypair -outform PEM -out
secp256r1.privatekey
read EC key
writing EC key

> openssl ec -in secp256r1.keypair -outform PEM -pubout -out
secp256r1.publickey
read EC key
writing EC key
```

4.3.2　Renesas Image Generator の実行環境準備

パッケージに同梱されている ImageGenerator.zip を WindowsPC の任意のフォルダに解凍します。フォルダ名は、カタカナ、全角を含まないようにしてください。

Renesas Image Generator は、Python の実行環境が必要ですので、事前に Python のインストール（4.2.2）を実施してください。

4.4　RL78/G23-128p FPB 用のサンプルプロジェクト

この節では RL78/G23-128p FPB を用いたデモプロジェクトについて説明します。

RL78/G24-64p FPB を用いたデモは RL78/G23-128p FPB と同様のため、この章を参照してください。

RL78/G23-128p FPB 用のデモプロジェクトには、boot_loader、fwup_leddemo、fwup_main の 3 つのデモプロジェクトを CC-RL コンパイラ向けと IAR コンパイラ向けに用意しています。これらのデモプロジェクトは、コンフィグレーション設定の変更で、半面更新方式（バッファ面は内蔵フラッシュ）と全面更新方式（バッファ面は外部フラッシュ）の 2 つのファームウェアアップデート方式に対応しています。

なお、本デモプロジェクトの実行手順は、デバッガ接続を前提としていません。デバッガ接続によりアプリケーションのデバッグの実施方法については、4.6 を参照してください。

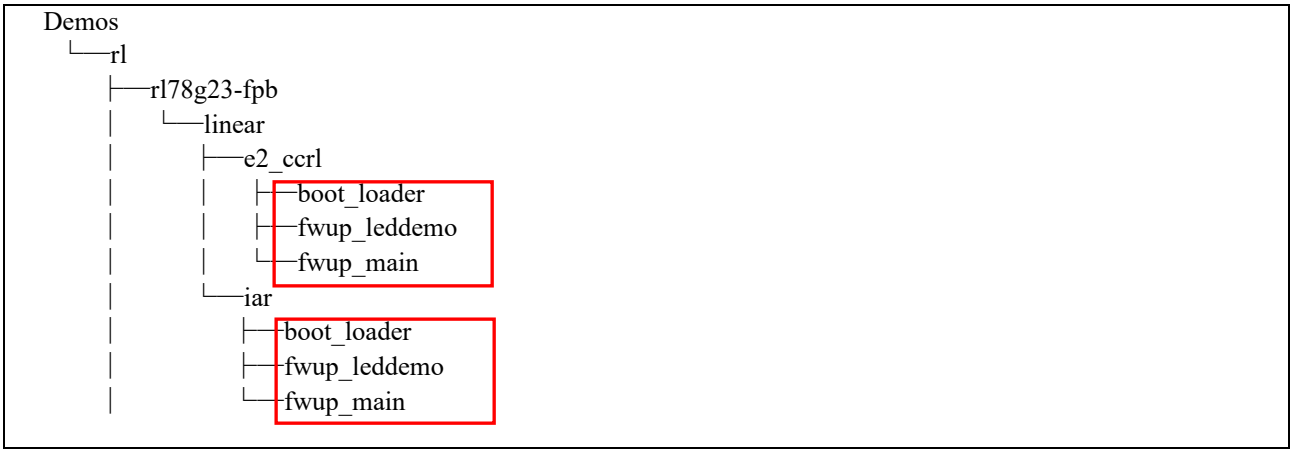


図 4-1 RL78/G23-128p FPB 用デモプロジェクトのフォルダ構成

表 4-2 デモプロジェクトで使用する機器

No.	機器	補足
1	開発 PC	開発を行う PC
2	評価ボード	RL78/G23-128p Fast Prototyping Board
3	ホスト PC	TeraTerm 等のターミナルソフトウェア
4	USB シリアル変換ボード	Pmod USBUART（DIGILENT 製） https://reference.digilentinc.com/reference/pmod/pmodusbuart/start
5	USB ケーブル	USB シリアル変換ボードとホスト PC を USB 接続します。
6	E2Lite	デバッガ
7	外部フラッシュメモリ	Macronix International 社製 MX25/66L family serial NOR Flash Memory 全面更新方式（バッファ面は外部フラッシュ）の動作を確認する際に使用します。

4.4.1　半面更新方式（バッファ面は内蔵フラッシュ）

4.4.1.1　デモプロジェクトの構築

以下の手順で、半面更新方式（バッファ面は内蔵フラッシュ）用の 3 つのデモプロジェクトを構築します。

以下は、e² studio 環境での手順を記載しています。IAR 環境でご使用の際は、IAR の統合開発環境の手順に読み替えて実施して下さい。

① 統合開発環境に boot_loader、fwup_main 、fwup_leddemo のデモプロジェクトをインポートします。
② イメージの検証に使用する公開鍵をデモプロジェクトに追加します。
　プロジェクト boot_loader、fwup_main の code_signer_public_key.h に 4.3.1 で生成した secp256r1.publickey の内容を貼り付けます。

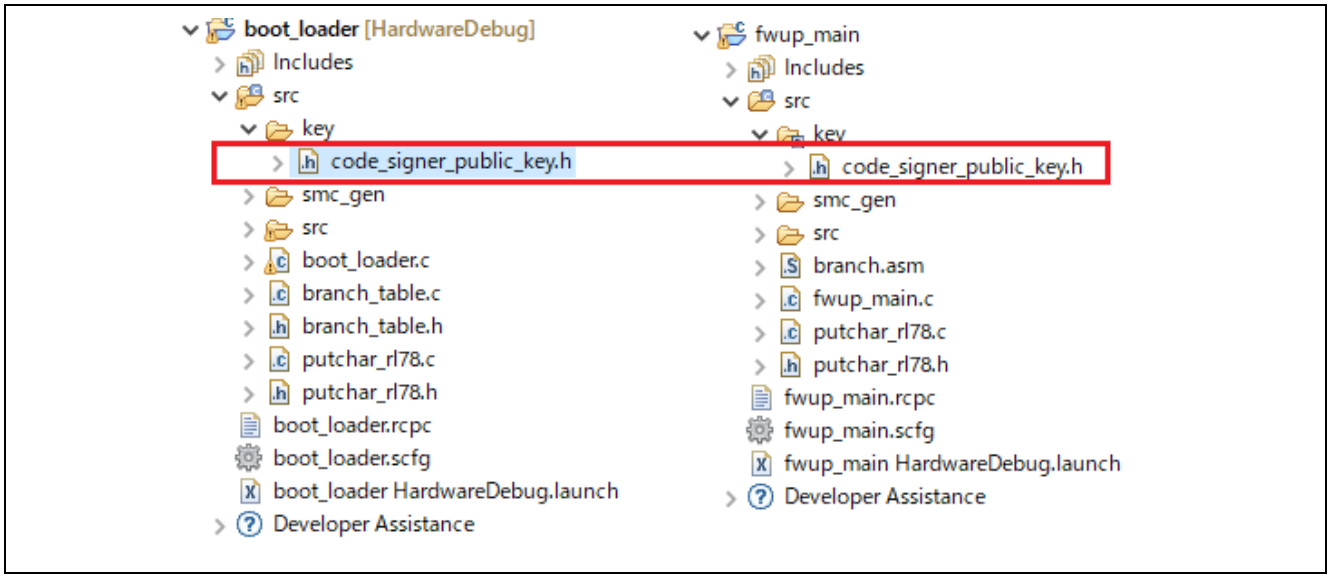


図 4-2 デモプロジェクトの code_signer_public_key.h ファイルの格納場所

```
/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----¥n"¥
 * "...base64 data...¥n"¥
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM ¥
"-----BEGIN PUBLIC KEY-----"¥
ここに secp256r1.publickey の中身を貼り付けます。
"-----END PUBLIC KEY-----"¥
#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

③ ファームウェアアップデートモジュールのコンフィグレーション設定を行います。
　プロジェクトの r_fwup_config.h を開き、表 6-3 RL78/G23 の半面更新方式（バッファ面は内部フラッシュ）のコンフィグ設定のように設定します。

④ デモプロジェクトをビルドします。
　3 つのデモプロジェクトをビルドし、以下の mot ファイルが生成されていることを確認します。

　boot_loader.mot、fwup_main.mot 、fwup_leddemo.mot

4.4.1.2　初期イメージと更新イメージを作成

初期イメージ名を initial_firm.mot、更新イメージ名を fwup_leddemo.rsu として、初期イメージと更新イメージの作成手順を説明します。

① Renesas Image Generator と同じフォルダにビルドしたデモプロジェクトの mot ファイルと 4.3.1 で生成した秘密鍵を格納します。

```
image-gen.py
RL78_G23_ImageGenerator_PRM.csv
RL78_G24_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
```

② 以下のコマンドを実行し、初期イメージを作成します。

```
> python image-gen.py -iup fwup_main.mot -ip
RL78_G23_ImageGenerator_PRM.csv -o initial_firm -ibp boot_loader.mot
-vt ecdsa -key secp256r1.privatekey

Successfully generated the initial_firm.mot file.
```

③ 以下のコマンドを実行し、更新イメージを作成します。

```
> python image-gen.py -iup fwup_leddemo.mot -ip
RL78_G23_ImageGenerator_PRM.csv -o fwup_leddemo -vt ecdsa -key
secp256r1.privatekey

Successfully generated the fwup_leddemo.rsu file.
```

Renesas Image Generator と同じフォルダに初期イメージと更新イメージが生成されていることを確認してください。

```
image-gen.py
RL78_G23_ImageGenerator_PRM.csv
RL78_G24_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.4.1.3　初期イメージの書き込み

初期イメージ(initial_firm.mot)をフラッシュライタでボードに書き込みます。書き込み後はボードの電源を OFF し、デバッガ（E2 Lite）の接続を外しておいてください。

4.4.1.4　ファームウェアアップデートの実行

初期イメージが起動するとターミナル経由で更新イメージの転送を待ちます。受信した更新イメージをフラッシュに書き込み、転送完了後に署名検証を経て更新イメージのファームウェアを起動します。

以下の手順により、ファームウェアアップデートを実施してください。

① 「図 **6-1** RL78/G23-128p FPB 半面更新方式（バッファ面は内部フラッシュ）の機器接続図」を参考に機器を接続してください。

② PC のターミナルソフトを起動しシリアル COM ポートを選択し接続設定を行います。

③ ボードの電源を投入します。以下のメッセージが出力されます。

```
==== RL78G23 : BootLoader [with buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute new image ...

==== RL78G23 : Update from User [with buffer] ver 1.0.0 ====
send image(*.rsu) via UART.
```

④ ターミナルから更新イメージを送信します。

ファイル送信>バイナリをチェック>fwup_leddemo.rsu

更新イメージの転送中は以下のメッセージが出力され、インストールと署名検証が終了するとソフトウェアリセットします。

```
W 0x59000, 128 ... OK
W 0x59080, 128 ... OK
...
W 0x5BA00, 128 ... OK
W 0x5BA80, 128 ... OK
verify install area buffer [sig-sha256-ecdsa]...OK
software reset...
```

⑤ ブートローダでアクティベート処理を実行し再度ソフトウェアリセットします。

```
==== RL78G23 : BootLoader [with buffer] ====
verify install area buffer [sig-sha256-ecdsa]...OK
copy to main area ... OK
software reset...
```

⑥ ブートローダで署名検証が終了すると更新イメージのファームウェアが起動します。

以下のメッセージが出力されて LED が点滅していれば正常です。

```
==== RL78G23 : BootLoader [with buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute new image ...

----------------------------------------
FWUP demo (ver 0.1.1)
----------------------------------------
Check the LEDs on the board.
```

※デモプログラムでは、バッファ面の消去を行っておりません。ロールバックの対策等で、更新前のイメージを消去する必要がある場合は、ブートローダでの検証完了後、バッファ面のイメージの消去処理を追加してください。

4.4.2　全面更新方式（バッファ面は外部フラッシュ）

4.4.2.1　デモプロジェクトの構築

以下の手順で、全面更新方式（バッファ面は外部フラッシュ）用の 3 つのデモプロジェクトを構築します。

以下は、e² studio 環境での手順を記載しています。IAR 環境でご使用の際は、IAR の統合開発環境の手順に読み替えて実施して下さい。

① 統合開発環境に boot_loader、fwup_leddemo のデモプロジェクトをインポートします。
② イメージの検証に使用する公開鍵をデモプロジェクトに追加します。
プロジェクト boot_loader の code_signer_public_key.h に 4.3.1 で生成した secp256r1.publickey の内容を貼り付けます。

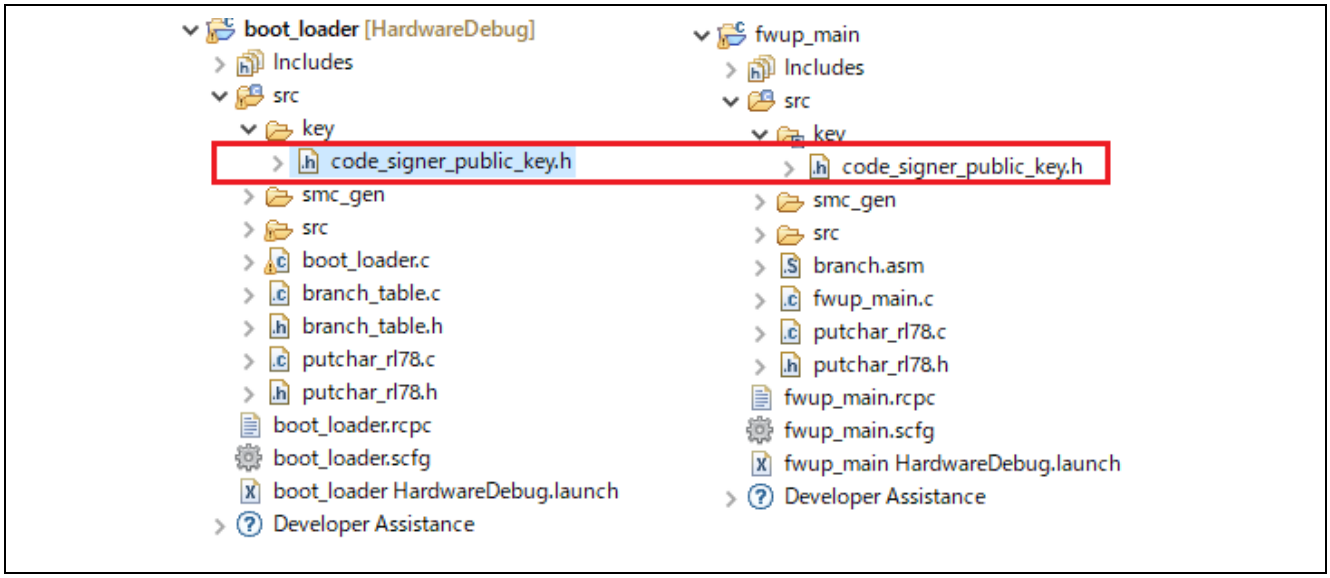


図 4-3 デモプロジェクトの code_signer_public_key.h ファイルの格納場所

```
/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----¥n"¥
 * "...base64 data...¥n"¥
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM ¥
"-----BEGIN PUBLIC KEY-----"¥
ここに secp256r1.publickey の中身を貼り付けます。
"-----END PUBLIC KEY-----"¥
#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

③ ファームウェアアップデートモジュールのコンフィグレーション設定を行います。
プロジェクトの r_fwup_config.h を開き、表 6-4 RL78/G23 の全面更新方式（バッファ面は外部フラッシュ）のコンフィグ設定のように設定します。イメージの検証に使用する公開鍵をデモプロジェクトに追加します。

④ デモプロジェクトをビルドします。
3 つのデモプロジェクトをビルドし、以下の mot ファイルが生成されていることを確認します。

boot_loader.mot、fwup_main.mot 、fwup_leddemo.mot

4.4.2.2　初期イメージと更新イメージを作成

初期イメージ名を initial_firm.mot、更新イメージ名を fwup_leddemo.rsu として、初期イメージと更新イメージの作成手順を説明します。

① Renesas Image Generator と同じフォルダにビルドしたデモプロジェクトの mot ファイルと 4.3.1 で生成した秘密鍵を格納します。

```
image-gen.py
RL78_G23_ImageGenerator_PRM.csv
RL78_G24_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
```

② 以下のコマンドを実行し、初期イメージを作成します。

```
> python image-gen.py -iup fwup_main.mot -ip
RL78_G23_ImageGenerator_PRM.csv -o initial_firm -ibp boot_loader.mot
-vt ecdsa -key secp256r1.privatekey

Successfully generated the initial_firm.mot file.
```

③ 以下のコマンドを実行し、更新イメージを作成します。

```
> python image-gen.py -iup fwup_leddemo.mot -ip
RL78_G23_ImageGenerator_PRM.csv -o fwup_leddemo -vt ecdsa -key
secp256r1.privatekey

Successfully generated the fwup_leddemo.rsu file.
```

Renesas Image Generator と同じフォルダに初期イメージと更新イメージが生成されていることを確認してください。

```
image-gen.py
RL78_G23_ImageGenerator_PRM.csv
RL78_G24_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.4.2.3　初期イメージの書き込み

初期イメージ(initial_firm.mot)をフラッシュライタでボードに書き込みます。書き込み後はボードの電源をOFF し、デバッガ（E2 Lite）の接続を外しておいてください。

4.4.2.4　ファームウェアアップデートの実行

初期イメージが起動するとターミナル経由で更新イメージの転送を待ちます。受信した更新イメージをフラッシュに書き込み、転送完了後に署名検証を経て更新イメージのファームウェアを起動します。

以下の手順により、ファームウェアアップデートを実施してください。

① 「図 **6-2** RL78/G23-128p FPB 全面更新方式（バッファ面は外部フラッシュ）の機器接続図」を参考に機器を接続してください。

② PC のターミナルソフトを起動しシリアル COM ポートを選択し接続設定を行います。

③ ボードの電源を投入します。以下のメッセージが出力されます。

```
==== RL78G23 : BootLoader [with ext-buffer] ====
verify install area main[sig-sha256-ecdsa]...OK
execute new image ...

==== RL78G23 : Update from User [with ext-buffer] ver 1.0.0 ====
send image(*.rsu) via UART.
```

④ ターミナルから更新イメージを送信します。

ファイル送信>バイナリをチェック>fwup_leddemo.rsu

更新イメージの転送中は以下のメッセージが出力され、インストールと署名検証が終了するとソフトウェアリセットします。

```
W 0x0000, 128 ... OK
W 0x0080, 128 ... OK
...
W 0x1A00, 128 ... OK
W 0x1A80, 128 ... OK
verify install area buffer [sig-sha256-ecdsa]...OK
software reset...
```

⑤ ブートローダでアクティベート処理を実行し再度ソフトウェアリセットします。

```
==== RL78G23 : BootLoader [with ext-buffer] ====
verify install area buffer [sig-sha256-ecdsa]...OK
copy to main area ... OK
software reset...
```

⑥ ブートローダで署名検証が終了すると更新イメージのファームウェアが起動します。以下のメッセージが出力されて LED が点滅していれば正常です。

```
==== RL78G23 : BootLoader [with ext-buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute new image ...

----------------------------------------
FWUP demo (ver 0.1.1)
----------------------------------------
Check the LEDs on the board.
```

※デモプログラムでは、バッファ面の消去を行っておりません。ロールバックの対策等で、更新前のイメージを消去する必要がある場合は、ブートローダでの検証完了後、バッファ面のイメージの消去処理を追加してください。

4.5　RL78/G22-48p FPB 用のサンプルプロジェクト

この節では RL78/G22-48p FPB を用いたデモプロジェクトについて説明します。

RL78/ G22-48p FPB 用のデモプロジェクトは、に示すように、boot_loader、fwup_leddemo の 2 つのデモプロジェクトを CC-RL コンパイラ向けと IAR コンパイラ向けに用意しています。これらのデモプロジェクトは、全面更新方式（バッファ無し）のファームウェアアップデート方式に対応しています。

なお、本デモプロジェクトの実行手順は、デバッガ接続を前提としていません。デバッガ接続によりアプリケーションのデバッグの実施方法については、4.6 を参照してください。

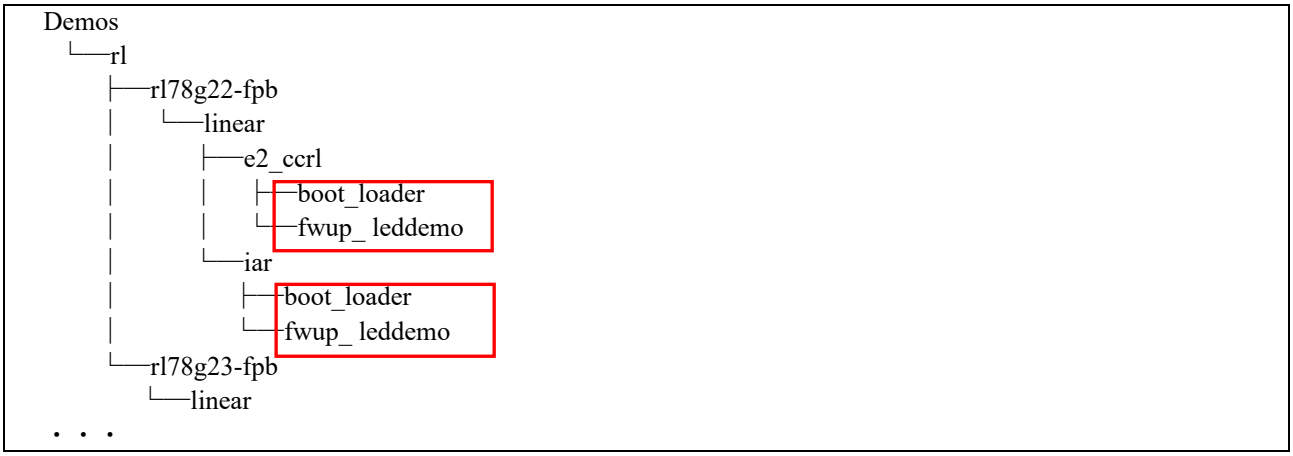


図 4-4 RL78/G22-48p FPB 用デモプロジェクトのフォルダ構成

表 4-3 デモプロジェクトで使用する機器

No.	機器	補足
1	開発 PC	開発を行う PC
2	評価ボード	RL78/G22-48p Fast Prototyping Board
3	ホスト PC	TeraTerm 等のターミナルソフトウェア
4	USB シリアル変換ボード	Pmod USBUART（DIGILENT 製） https://reference.digilentinc.com/reference/pmod/pmodusbuart/start
5	USB ケーブル	USB シリアル変換ボードとホスト PC を USB 接続します。
6	E2Lite	デバッガ

4.5.1　全面更新方式（バッファ無し）

4.5.1.1　デモプロジェクトの構築

以下の手順で、全面更新方式（バッファ無し）用の 2 つのデモプロジェクトを構築します。

以下は、e² studio 環境での手順を記載しています。IAR 環境でご使用の際は、IAR の統合開発環境の手順に読み替えて実施して下さい。

① 統合開発環境に boot_loader、fwup_leddemo のデモプロジェクトをインポートします。
② ファームウェアアップデートモジュールのコンフィグレーション設定を行います。
プロジェクトの r_fwup_config.h を開き、表 6-7 RL78/G22 の全面更新方式（バッファ無し）のコンフィグ設定のように設定します。
③ ファームウェアアップデートモジュールのコンフィグレーション設定を行います。
a. プロジェクト（boot_loader）をビルドし boot_loader.mot を生成します。
b. プロジェクト（fwup_leddemo）をビルドし fwup_leddemo.mot を生成します。
c. fwup_leddemo.mot を fwup_leddemo_011.mot にリネームします。
d. プロジェクト（fwup_leddemo）のバージョンを以下のように変更してビルドし、fwup_leddemo.mot を生成します。

```
fwup_leddemo.c
---
#define FWUP_DEMO_VER_MAJOR      (0)
#define FWUP_DEMO_VER_MINOR      (1)
#define FWUP_DEMO_VER_BUILD      (1)★1->2
```

e. fwup_leddemo.mot を fwup_leddemo_012.mot にリネームします。

4.5.1.2　初期イメージと更新イメージを作成

初期イメージ名を initial_firm.mot、更新イメージ名を fwup_leddemo_012.rsu として、初期イメージと更新イメージの作成手順を説明します。

① Renesas Image Generator と同じフォルダにビルドしたデモプロジェクトの mot ファイルを格納します。

```
image-gen.py
RL78_G22_ImageGenerator_PRM.csv
boot_loader.mot
fwup_leddemo_011.mot
fwup_leddemo_012.mot
```

② 以下のコマンドを実行し、初期イメージを作成します。

```
> python image-gen.py -iup fwup_leddemo_011.mot -ip
RL78_G22_ImageGenerator_PRM.csv -o initial_firm -ibp boot_loader.mot

Successfully generated the initial_firm.mot file.
```

③ 以下のコマンドを実行し、更新イメージを作成します。

```
> python image-gen.py -iup fwup_leddemo_012.mot -ip
RL78_G22_ImageGenerator_PRM.csv -o fwup_leddemo_012

Successfully generated the fwup_leddemo_012.rsu file.
```

Renesas Image Generator と同じフォルダに初期イメージと更新イメージが生成されていることを確認してください。

```
image-gen.py
RL78_G22_ImageGenerator_PRM.csv
boot_loader.mot
fwup_leddemo_011.mot
fwup_leddemo_012.mot
fwup_leddemo_012.rsu
initial_firm.mot
```

4.5.1.3　初期イメージの書き込み

初期イメージ(initial_firm.mot)をフラッシュライタでボードに書き込みます。書き込み後はボードの電源を OFF し、デバッガ（E2 Lite）の接続を外しておいてください。

4.5.1.4　ファームウェアアップデートの実行

初期イメージが起動すると LED が点滅します。ボードの USER_SW を押しながら RESET_SW を ON→OFF してアップデートモードに入り、ターミナル経由で更新イメージの転送を待ちます。受信した更新イメージをフラッシュに書き込み、転送完了後に検証を経て更新イメージのファームウェアを起動します。

以下の手順により、ファームウェアアップデートを実施してください。

① 「図 **6-11** RL78/G22-48p FPB 全面更新方式（バッファなし）の機器接続図」を参考に機器を接続してください。

② PC のターミナルソフトを起動しシリアル COM ポートを選択し接続設定を行います。

③ ボードの電源を投入します。以下のメッセージが出力されます。

```
==== RL78G22 : BootLoader [without buffer] ====
verify install area main [hash-sha256]...OK
execute new image ...

----------------------------------------
FWUP demo (ver 0.1.1)
----------------------------------------
```

④ USER_SW を押しながら RESET_SW を ON→OFF します。

```
==== RL78G22 : Image updater [without buffer] ====
send image(*.rsu) via UART.
```

⑤ ターミナルから更新イメージを送信します。

ファイル送信>バイナリをチェック>fwup_leddemo_012.rsu

更新イメージの転送中は以下のメッセージが出力され、インストールと署名検証が終了するとソフトウェアリセットします。

```
W 0x2000, 64 ... OK
W 0x2040, 64 ... OK
･･･
W 0x4D00, 64 ... OK
W 0x4D40, 64 ... OK
verify install area 0 [hash-sha256]...OK
software reset...
```

⑥ ブートローダで署名検証が終了すると更新イメージのファームウェアが起動します。以下のメッセージが出力されて LED が点滅していれば正常です。

```
==== RL78G22 : BootLoader [without buffer] ====
verify install area main [hash-sha256]...OK
execute new image ...

----------------------------------------
FWUP demo (ver 0.1.2)
----------------------------------------
Check the LEDs on the board.
```

4.6　デモプロジェクトのデバック方法

e² studio の環境で本プロジェクト（ブートローダ＋アプリケーションプログラム）のデバッグを実施したい場合、以下の手順でデバッグを行うことが可能です。

なお、本デモプロジェクトは、デバッガ（E2 Lite）でエミュレータから電源を供給する設定となっています。他のデバッガによる接続やターゲットボードから電源を供給したい場合には、デバッガの設定を変更してください。

(1) ブートローダとアプリケーションプログラムをビルドします。

ブートローダ（boot_loader）とアプリケーションプログラム（fwup_main）のビルドを行います。

詳細は、「4.x.x.1 デモプロジェクトの構築」を参照して下さい。

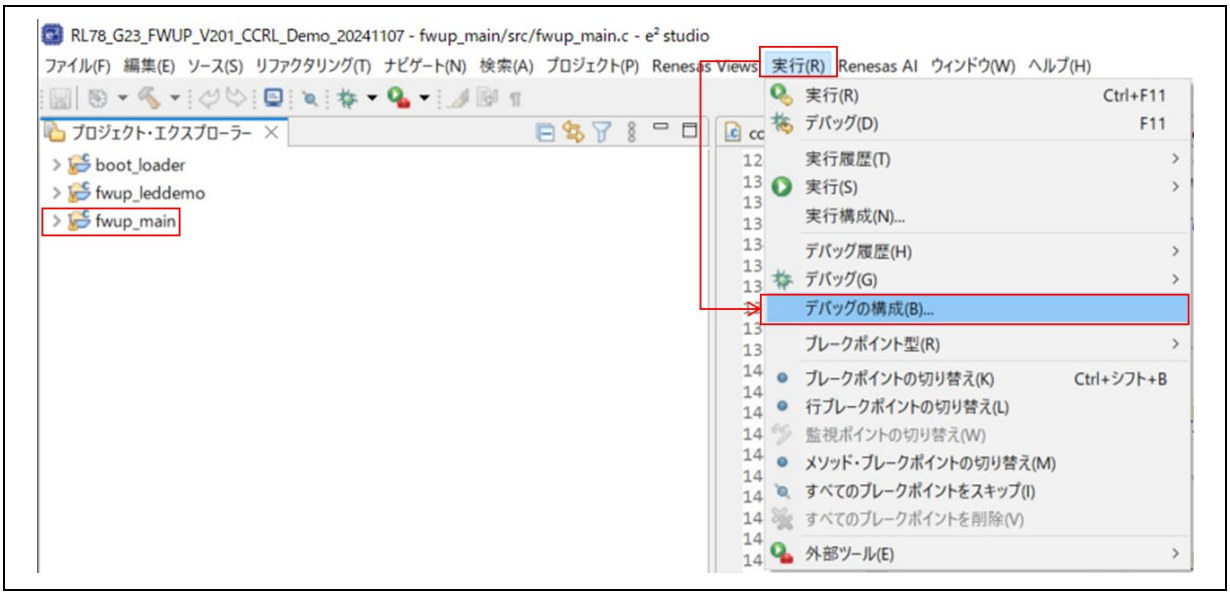
(2) 初期イメージを生成します。

Renesas Image Generator により、ブートローダ（boot_loader）とアプリケーションプログラム（fwup_main）で構成される初期イメージファイル(.mot)を生成します。詳細は、「4.x.x.2 初期イメージと更新イメージを作成」を参照してください。

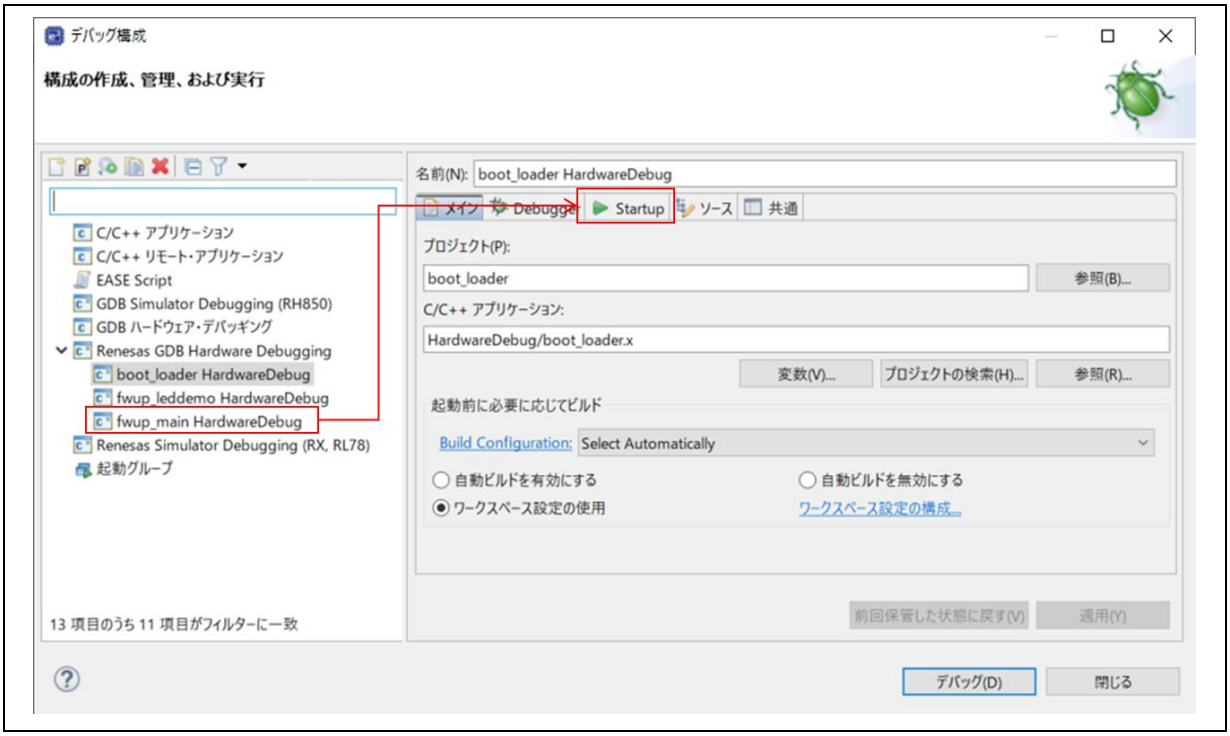
(3) アプリケーションプログラム（fwup_main）のデバッグ設定を行います。

以下の手順により、アプリケーションプログラム（fwup_main）のデバッグ設定を行います。

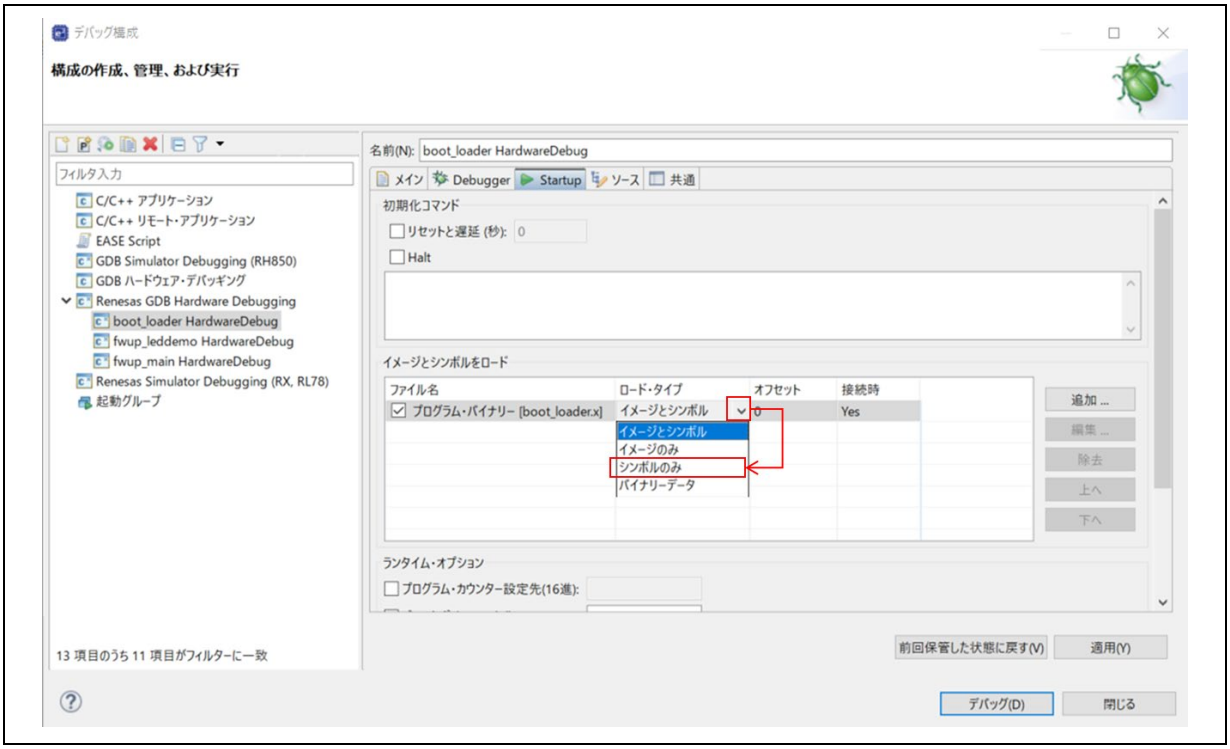
① 実行(R)->デバッグ構成(B)を開き、fwup_main_HardwareDebug を選択します。



② fwup_main_HardwareDebug を選択し、Startup をクリックします。

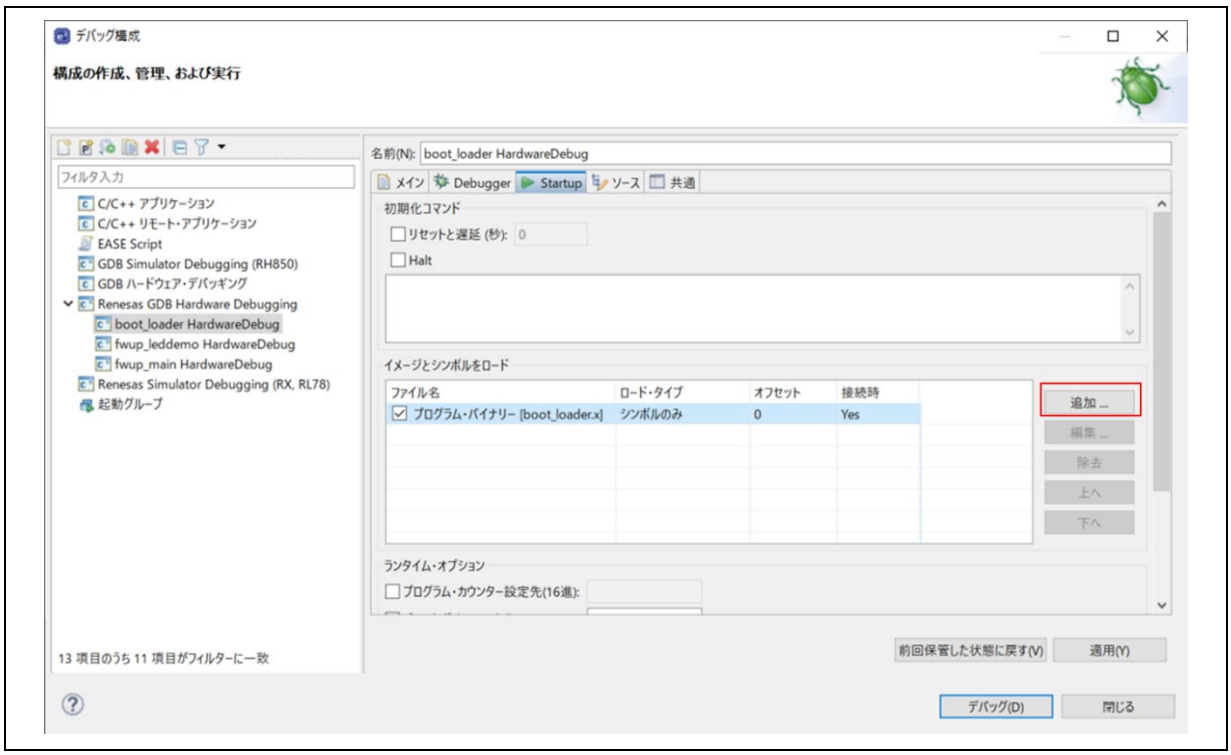


③ プログラム・バイナリ[fwup_main.x]のロード・タイプを「イメージとシンボル」から「シンボルのみ」に変更します。

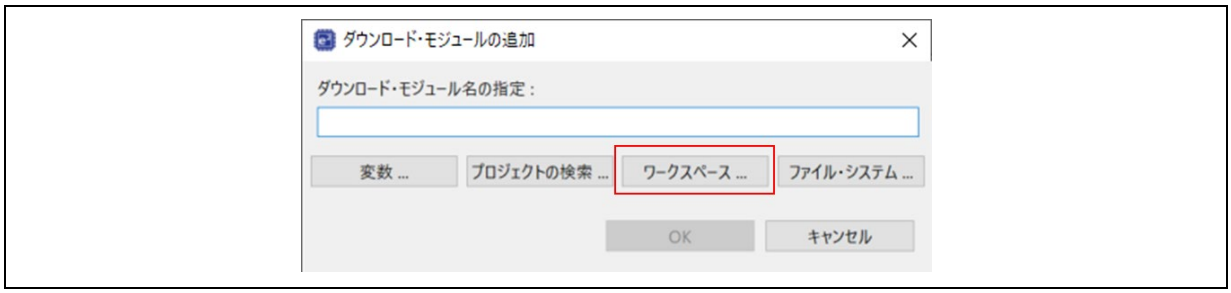


(4) ブートローダ（boot_loader）のシンボルを追加します。
以下の手順により、手順(1)でビルドしたブートローダ（boot_loader）のシンボルを追加します。

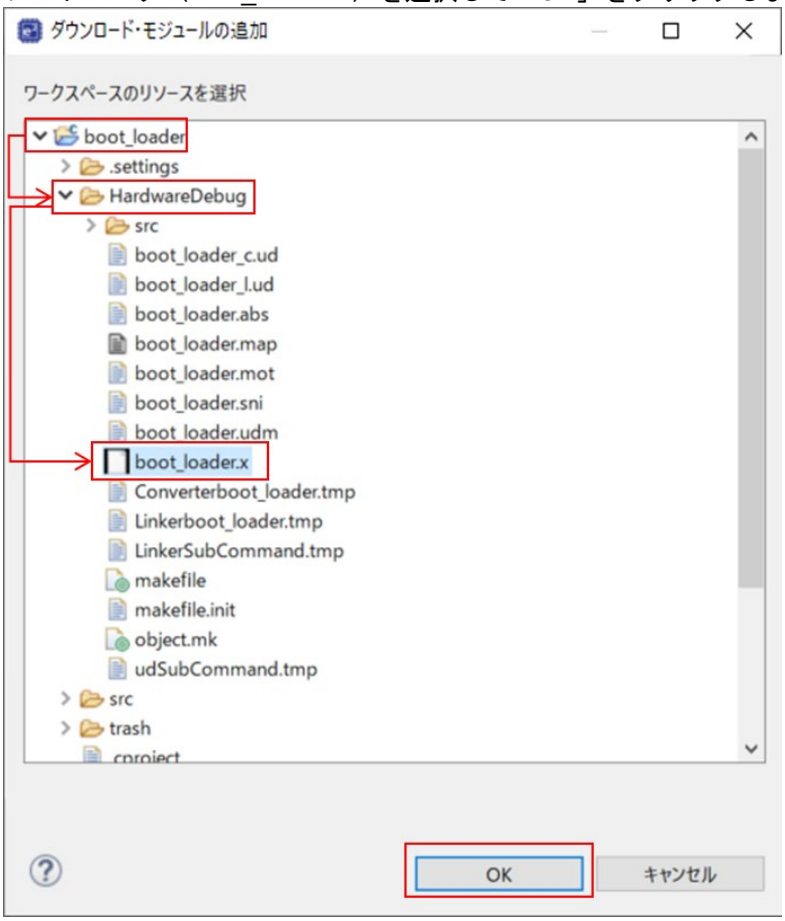
① 追加をクリックします。



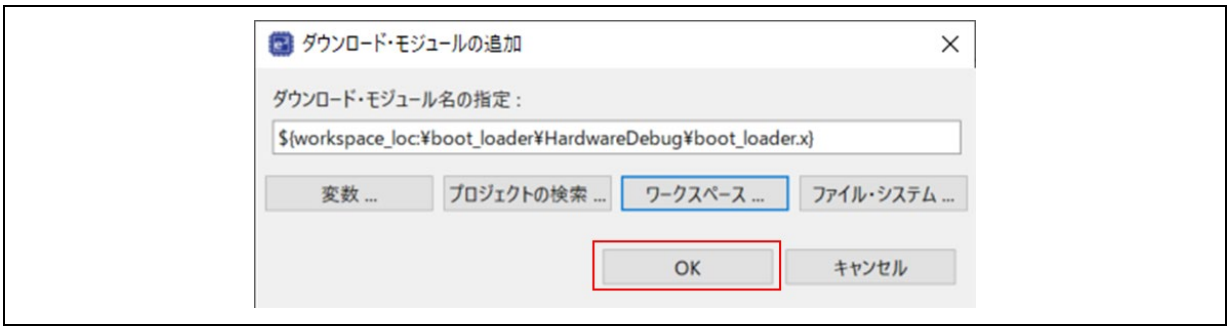
② ワークスペースをクリックします。



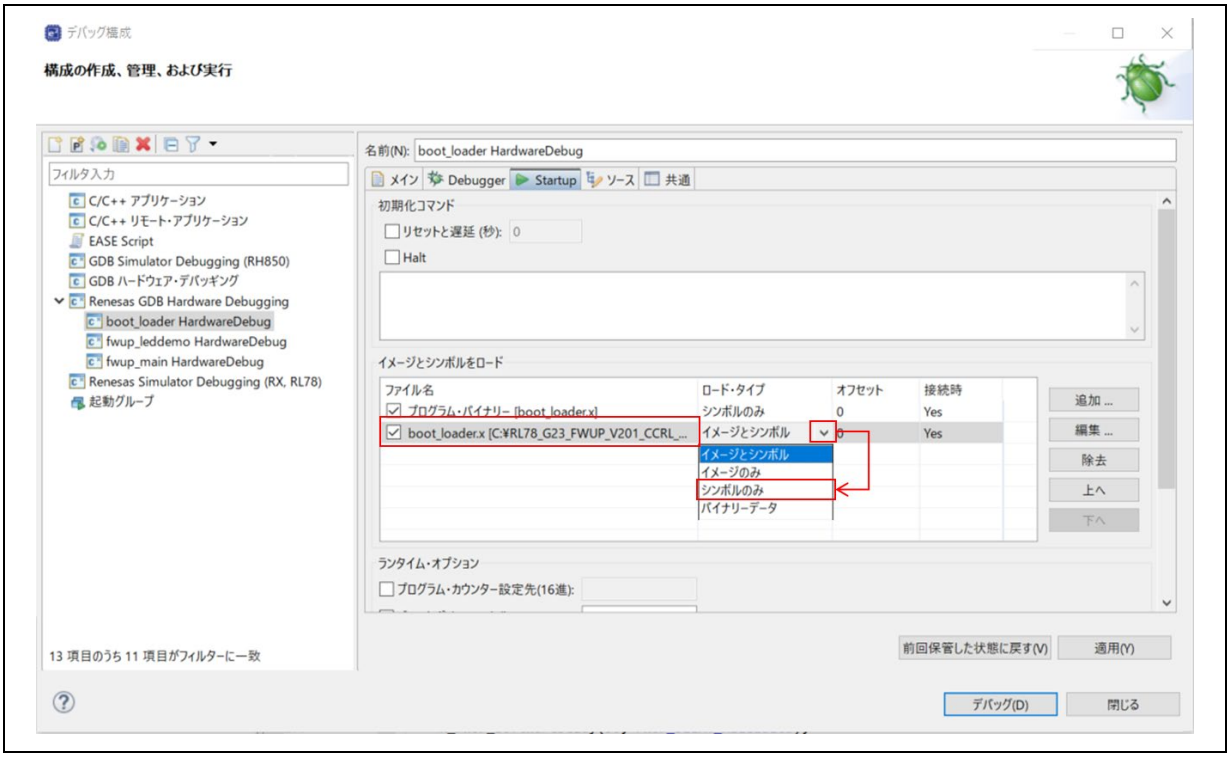
③ ブートローダ（boot_loader.x）を選択して「OK」をクリックします。



④ ダウンロード・モジュール名の設定がブートローダ（boot_loader.x）になっていることを確認して「OK」をクリックします。

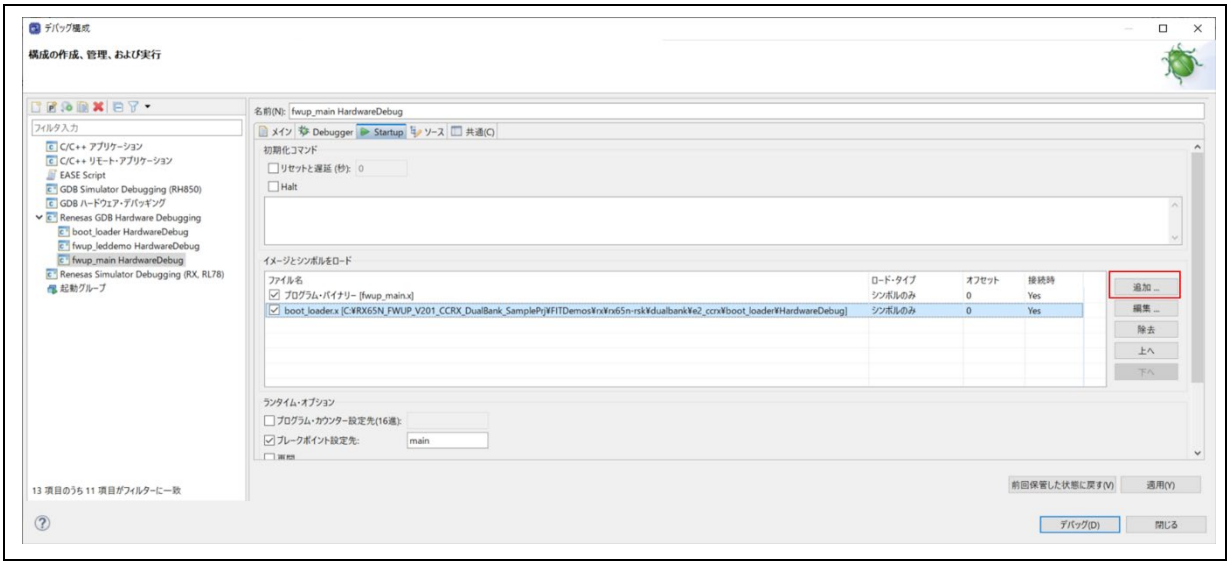


⑤ ブートローダ（boot_loader.x）のロード・タイプを「イメージとシンボル」から「シンボルのみ」に変更します。

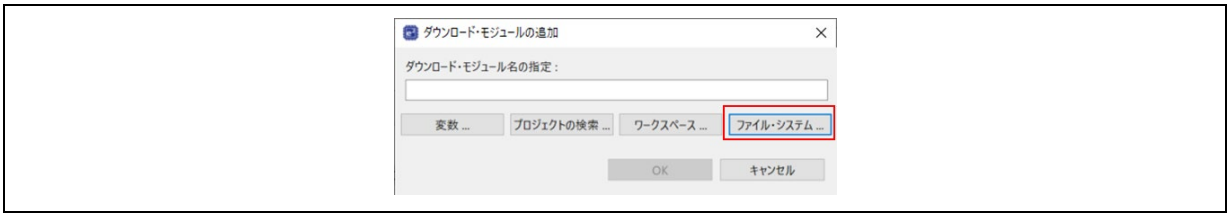


(5) 初期イメージ（initial_firm.mot）のイメージを追加します。
以下の手順により、手順(2)で生成した初期イメージ（initial_firm.mot）のイメージを追加します。

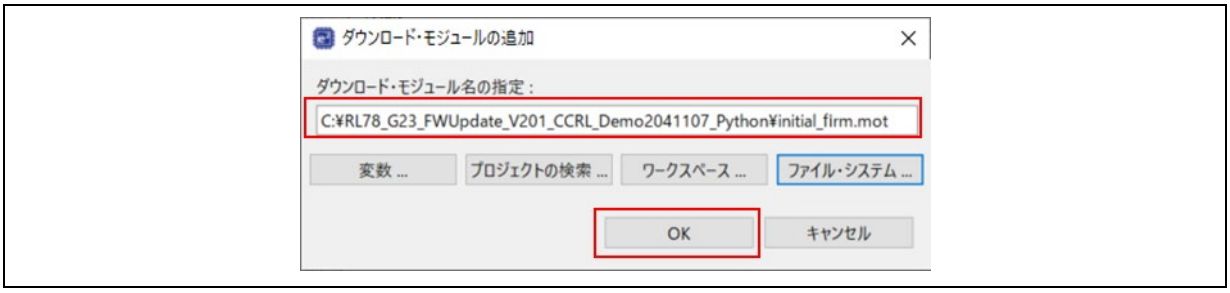
① 追加をクリックします。



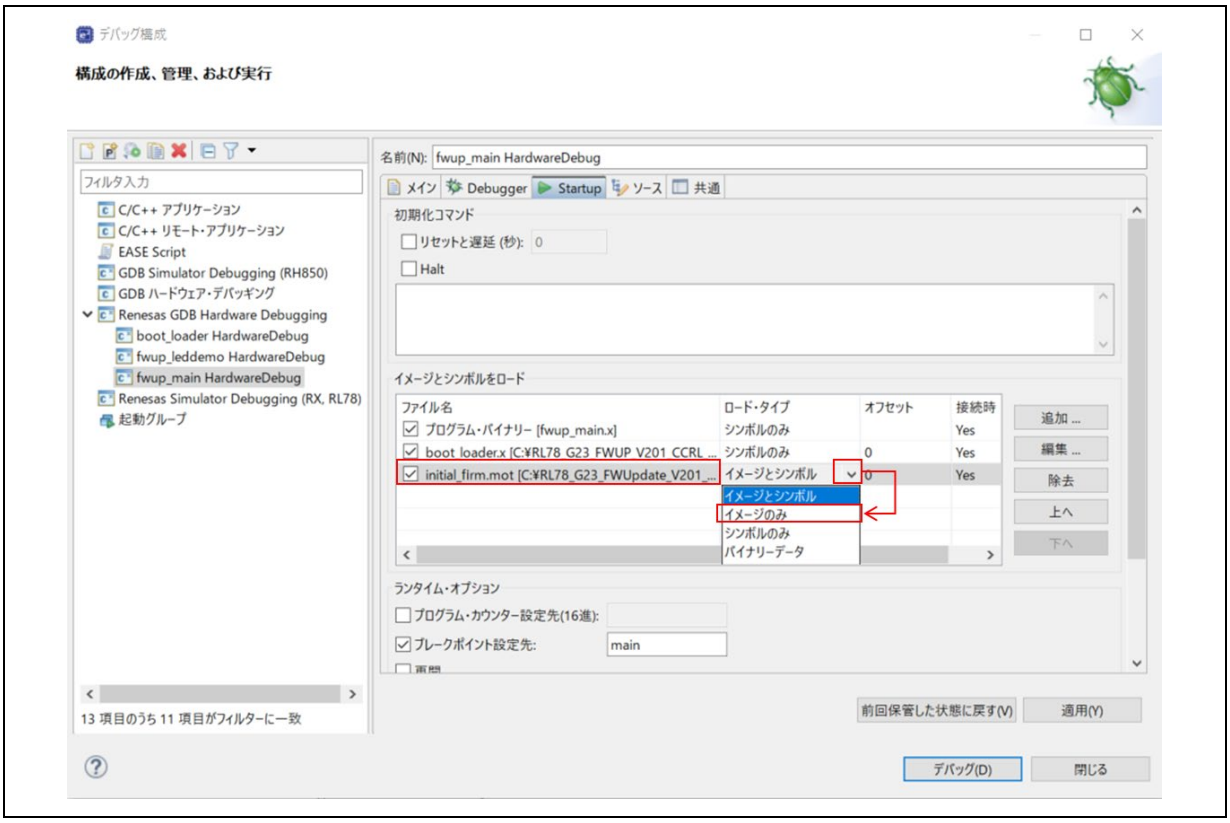
② 「ファイル・システム」をクリックします。



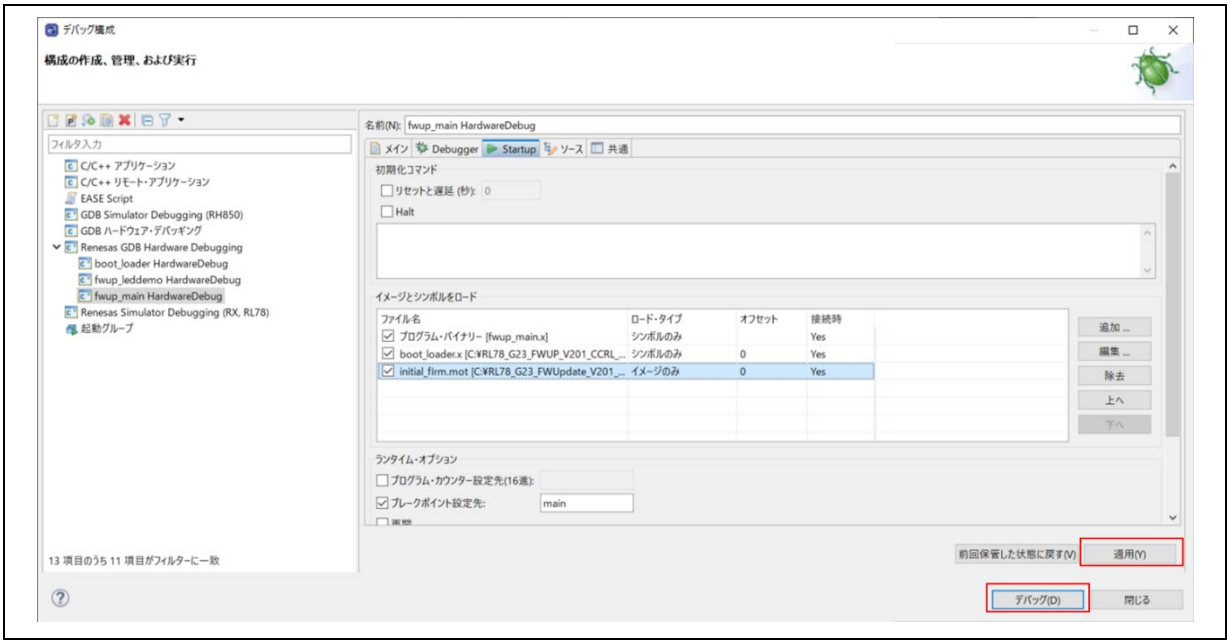
③ 初期イメージ（initial_firm.mot）を選択して「OK」をクリックします。



④ 初期イメージ（initial_firm.mot）のロード・タイプを「イメージとシンボル」から「イメージのみ」に変更して「OK」をクリックします。

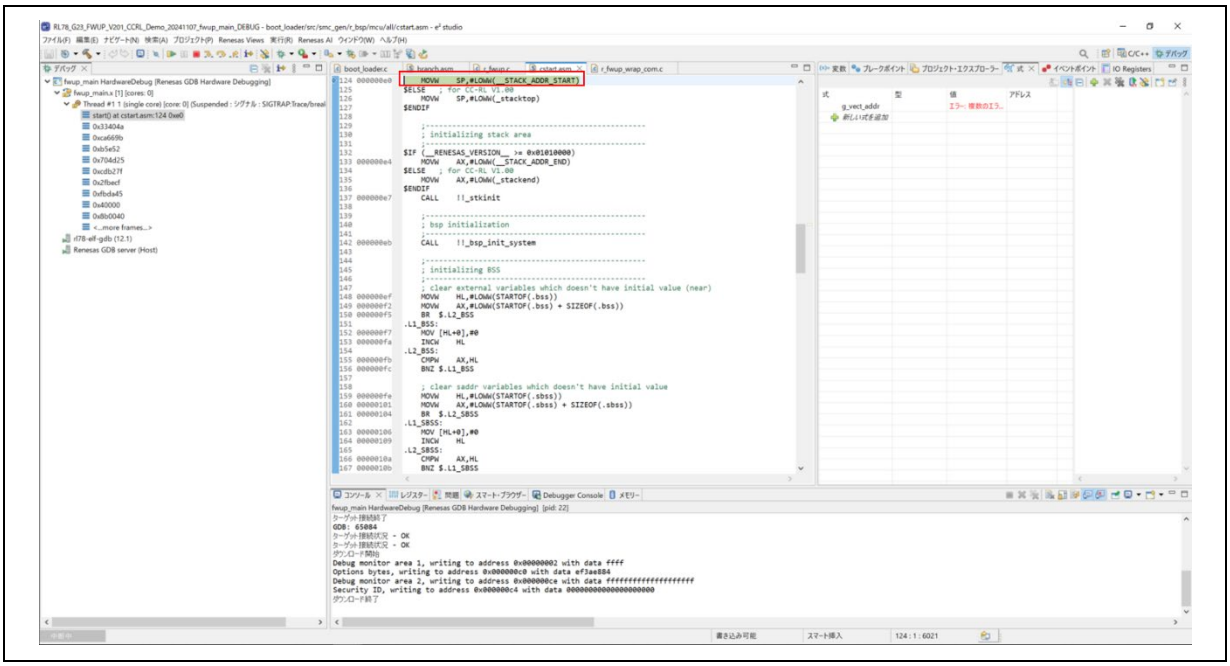


⑤ 「適用(Y)」をクリックし、「デバッグ(D)」をクリックします。

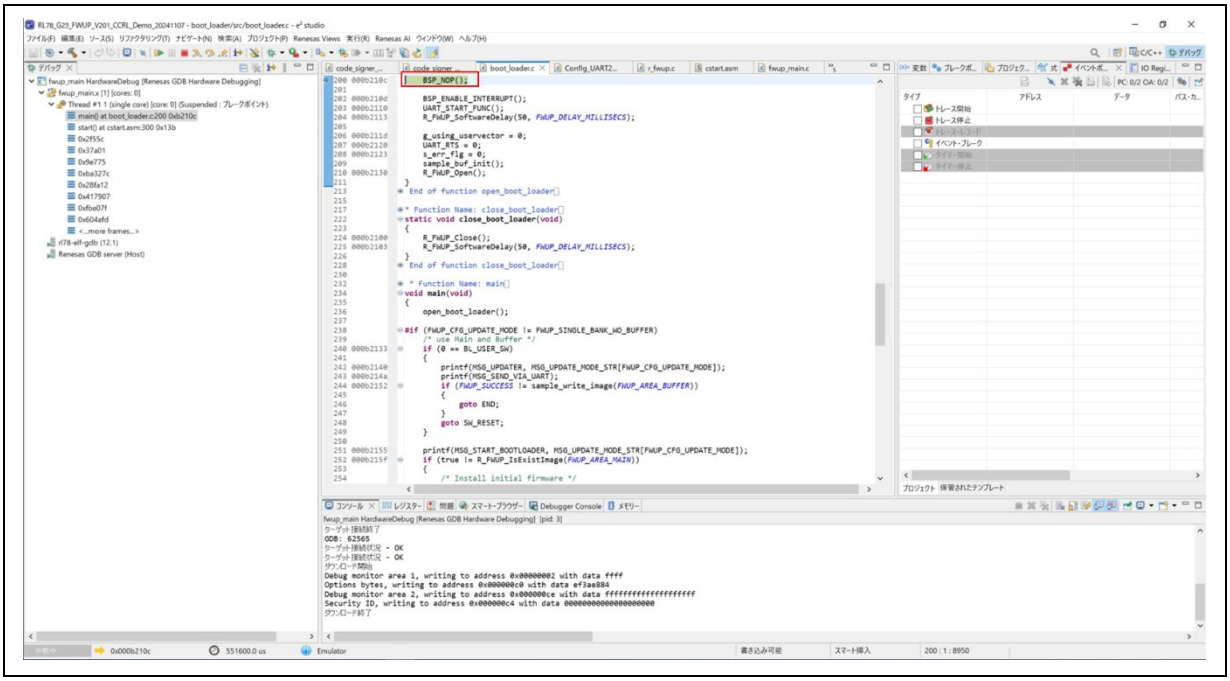


(6)デバッガを起動します。

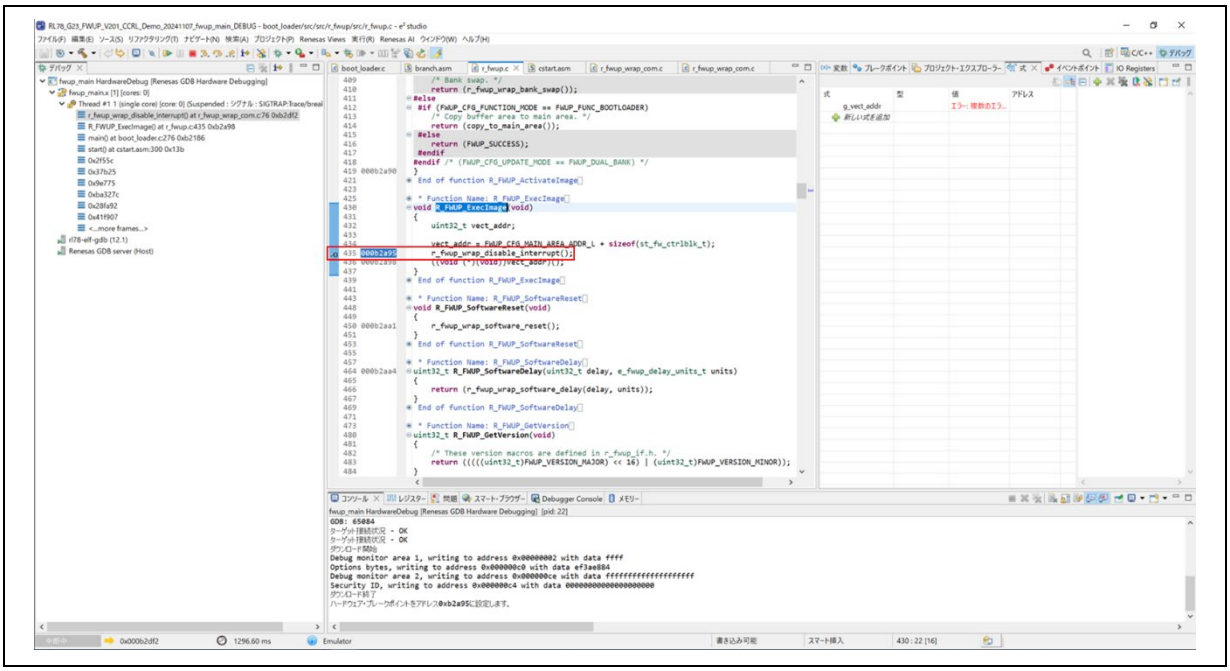
デバッガが起動すると、以下、boot_loader プロジェクトの cstart.asm にジャンプします。



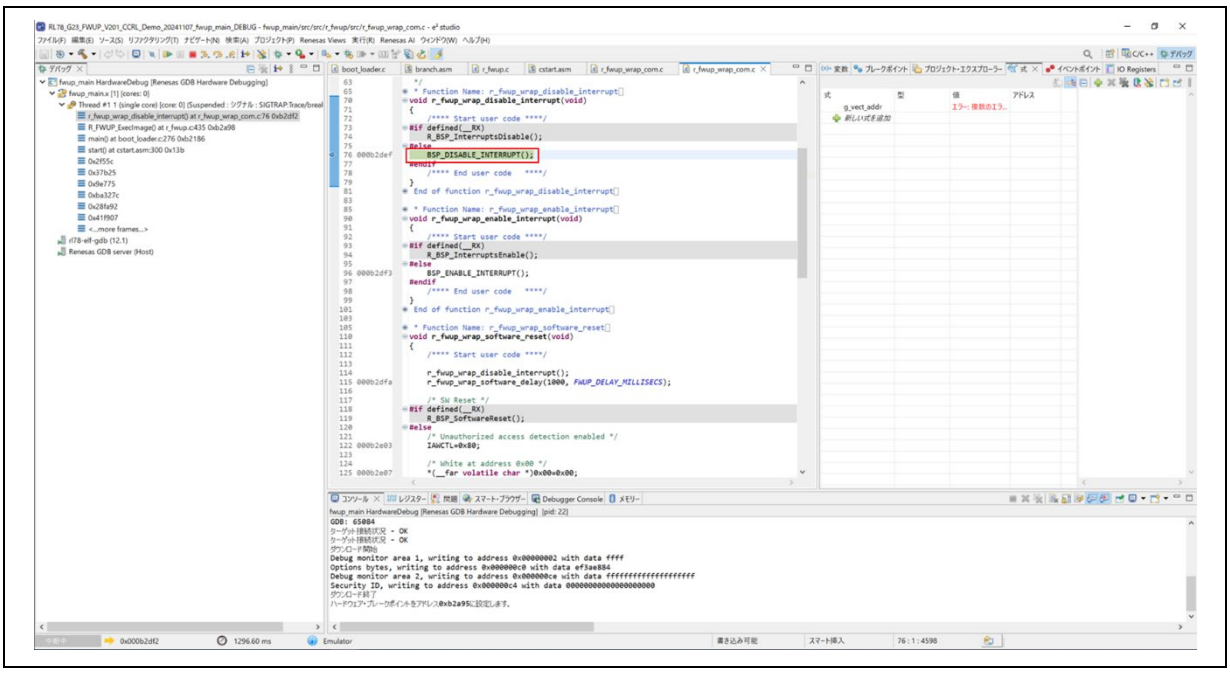
(7)プログラムを再開します。
　再開(M)をクリックすると、boot_loader.c の main()関数の先頭で停止します。



(8) fwup_main プロジェクトの main()にブレイクポイントを設定します。
　fwup_main プロジェクトの main()の以下赤枠にブレイクポイントを設定します。



(9)プログラムを再開します。
　　再開(M)をクリックし、(8)で設定したブレイクポイントで停止することを確認します。



5. Renesas Image Generator

Renesas Image Generator は、ファームウェアアップデートモジュールで使用するファームウェアイメージを生成するユーティリティツールです。Renesas Image Generator はファームウェアアップデートモジュールが使用する以下のイメージを生成することができます。

・初期イメージ：ブートローダとアプリケーションプログラムで構成されるシステムの初期設定時にフラッシュライタで書き込むイメージファイル(拡張子 mot)

・更新イメージ：ファームウェアアップデート対象のイメージファイル(拡張子 rsu)

イメージの生成方法については 5.1、イメージの構成やパラメータファイルの詳細については 5.2～5.3 を参照してください。

Renesas Image Generator は Python 上で動作するプログラムです。

5.1 イメージの生成方法

Renesas Image Generator (image-gen.py) の仕様と、本ツールを使用してイメージファイル（初期イメージまたは更新イメージ）を生成する方法を説明します。

初期イメージの生成方法については 5.1.1、更新イメージの生成方法については 5.1.2 を参照してください。

image-gen.py のコマンド形式は次のとおりです。

```
python image-gen.py < options >
```

image-gen.py のオプションには、必須のものと省略可能なものがあります。表 5-1 に必須のオプション、表 5-2 に省略可能なオプションを示します。

表 5-1 image-gen.py の必須オプション

オプション	説明
-iup <file>	アプリケーションプログラムを指定します。 <file>文字列にはアプリケーションプログラムの mot ファイル名(ファイル名を含めたフルパスまたは相対パス)を指定してください。
-ip <file>	パラメータファイルを指定します。 <file>の文字列には、入力するパラメータファイル名（ファイル名を含めたファイルのフルパスまたは相対パス）を指定してください。
-o <file>	出力するイメージのファイル名を指定します。 出力するイメージファイルのファイル名（ファイル名を含めたフルパスまたは相対パス）を拡張子なし<file>文字列で指定します。 ファイル拡張子は、-ibp <file>のオプションでブートローダを指定した時は出力イメージが初期イメージであると判断して.mot となり、-ibp <file>の指定を省略した時は出力イメージが更新イメージであると判断して.rsu となります。

表 5-2 image-gen.py の省略可能なオプション

オプション	説明
-ibp <file>	ブートローダを指定します。 <file>文字列にはブートローダの mot ファイル名(ファイル名を含めたフルパスまたは相対パス)を指定してください。 mot ファイルを生成する場合に指定してください。
--key <file>	ECDSA でイメージを署名するための鍵ファイル名を指定します。 (-vt オプションに sha256 を指定した場合、このオプションは設定不要です。) コマンド実行フォルダに「secp256r1.privatekey」ファイルを格納してください。 ファイル名を変更する場合は、ファイル名を含めたファイルのフルパスまたは相対パスを指定してください。
-vt <VerificationType>[sha256 / ecdsa]	ファームウェアアップデートモジュールでのイメージ検証方式を指定します。 sha256：イメージのハッシュを付加します。 このオプションを省略した場合、"sha256" が指定されたものとみなされます。 ecdsa： イメージの署名を付加します。 -key で指定する鍵ファイルにより署名データを生成します。 -key で鍵ファイルを指定しないとエラーとなります。
-ff <FileFormat>[BareMetal / RTOS]	RSU フォーマット タイプを指定します。 -ff に指定可能な<FileFormat>は、以下の通りです。 BareMetal：アプリケーションプログラムのデータに RSU ヘッダ署名情報を付加したイメージを生成します。本デモプロジェクトで使用する RSU フォーマットです。 このオプションを省略した場合、"BareMetal"が指定されたとみなされます。 RTOS：FreeRTOS OTA 向けの更新イメージを生成します。 FreeRTOS OTA 向けの更新イメージは、RSU ヘッダ署名情報（更新イメージの先頭から 0x200 バイトのデータ）を付加しません。 BareMetal_FWUP_V2_V1_DATA：特殊用途向け RTOS_FWUP_V2_V1_DATA：特殊用途向け
.-h	コマンドの一覧を出力します。 このツールの使用に関するヘルプを表示するには、このオプションを指定してください。

5.1.1　初期イメージの生成方法

Renesas Image Generator は、ビルドにより生成されたブートローダファイル(.mot)名、アプリケーションプログラム (.mot)、パラメータファイル名(.csv)、出力ファイル名(拡張子なし)、ファームウェアアップデートモジュールでのイメージ検証方式(ecdsa/sha256)をコマンドラインオプションに指定し、初期イメージファイル(.mot)を生成します。

コマンド入力例

```
> python image-gen.py -iup fwup_main.mot -ip
RL78_G23_ImageGenerator_PRM.csv -o initial_firm -ibp boot_loader.mot
-vt ecdsa -key secp256r1.privatekey
```

fwup_main.mot：アプリケーションプログラムの mot ファイル名
RL78_G23_ImageGenerator_PRM.csv：入力するパラメータファイル名
initial_firm：出力する初期イメージファイルのファイル名
boot_loader.mot：ブートローダの mot ファイル名
ecdsa：ECDSA でイメージを署名
secp256r1.privatekey：ECDSA でイメージを署名するための鍵ファイル名

5.1.2　更新イメージの生成方法

Renesas Image Generator は、ビルドにより生成された更新用アプリケーションプログラム (.mot)、パラメータファイル名(.csv)、出力ファイル名(拡張子なし)、ファームウェアアップデートモジュールでのイメージ検証方式(ecdsa/sha256)をコマンドラインオプションに設定し、更新イメージファイル(.rsu)を生成します。

コマンド入力例

```
> python image-gen.py -iup fwup_leddemo.mot -ip
RL78_G23_ImageGenerator_PRM.csv -o fwup_leddemo -vt ecdsa -key
secp256r1.privatekey
```

fwup_leddemo.mot：更新用アプリケーションプログラムの mot ファイル名
RL78_G23_ImageGenerator_PRM.csv：入力するパラメータファイル名
fwup_leddemo：出力する更新イメージファイルのファイル名
ecdsa：ECDSA でイメージを署名
secp256r1.privatekey：ECDSA でイメージを署名するための鍵ファイル名

5.2　イメージファイル

5.2.1　更新イメージファイル

Renesas Image Generator が生成する更新イメージファイルの構成図を図 5-1 に示します。

なお、RSU ヘッダのフォーマットについては、表 5-3 を参照してください。

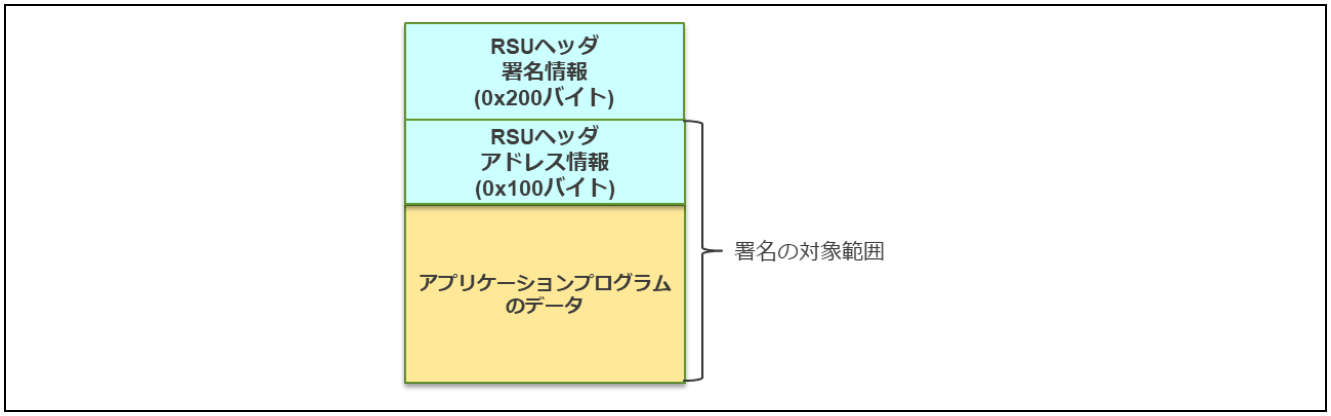


図 5-1 更新イメージファイルの構成

更新イメージファイルは RSU ヘッダとアプリケーションプログラムのデータで構成します。RSU ヘッダには、アプリケーションプログラムの正当性検証に必要なアプリケーションプログラムの配置情報とそれらをもとに計算したアプリケーションプログラムの署名値やハッシュ値を格納します。RSU ヘッダに続き、RSU ヘッダに格納したプログラム配置情報に対応するアプリケーションプログラムのデータを配置します。Renesas Image Generator は、アプリケーションプログラムのデータを、コードフラッシュに配置するデータ、データフラッシュに配置するデータの順に並べます。それぞれ、ユーザが生成したアプリケーションプログラムファイル(.mot)から有効なコードフラッシュデータ及びデータフラッシュのデータを取り出し、バイナリデータに変換してセットします。

更新イメージファイルは、半面更新方式および全面更新方式で同じ構成です。

表 5-3 RSU ヘッダフォーマット (1/2)

オフセット	項目	長さ (Byte)	説明
0x00000000	Magic Code	7	マジックコード（”RELFWV2”）
0x00000007	Reserved	1	予約領域
0x00000008	Firmware Verification Type	32	イメージ検証方式 イメージ検証に ECDSA を使用する場合は sig-sha256-ecdsa、ハッシュを使用する場合は、hash-sha256 を設定します。
0x00000028	Signature size	4	Signature に格納される署名値またはハッシュ値のデータサイズ Firmware Verification Type が sig-sha256-ecdsa の場合 0x40、hash-sha256 の場合 0x20 を設定します。
0x0000002C	Signature	64	イメージ検証に用いる署名値またはハッシュ値 Firmware Verification Type が hash-sha256 の場合、33 から 64 バイト目に 0x00 を設定します。
0x0000006C	RSU File Size	4	更新イメージファイル全体のファイルサイズ
0x00000070	Reserved	400	予約領域

表 5-4 RSU ヘッダフォーマット (2/2)

オフセット	項目	長さ (Byte)	説明
0x00000200	Program Data Num	4	後続する分割されたアプリケーションプログラムまたはデータフラッシュの数（最大 31 件）
0x00000204	Start Address[0]	4	1 件目のアプリケーションプログラムまたはデータフラッシュの先頭アドレス
0x00000208	Data Size[0]	4	1 件目のアプリケーションプログラムまたはデータフラッシュのサイズ
0x0000020C	Start Address[1]	4	2 件目のアプリケーションプログラムまたはデータフラッシュの先頭アドレス
0x00000210	Data Size[1]	4	2 件目のアプリケーションプログラムまたはデータフラッシュのサイズ
・ ・ ・	・ ・ ・		
0x000002F4	Start Address[30]	4	31 件目のアプリケーションプログラムまたはデータフラッシュの先頭アドレス
0x000002F8	Data Size[30]	4	31 件目のアプリケーションプログラムまたはデータフラッシュのサイズ
0x000002FC	Reserved	4	予約領域

更新イメージファイルを生成するメカニズムについては、図 5-2 を参照してください。

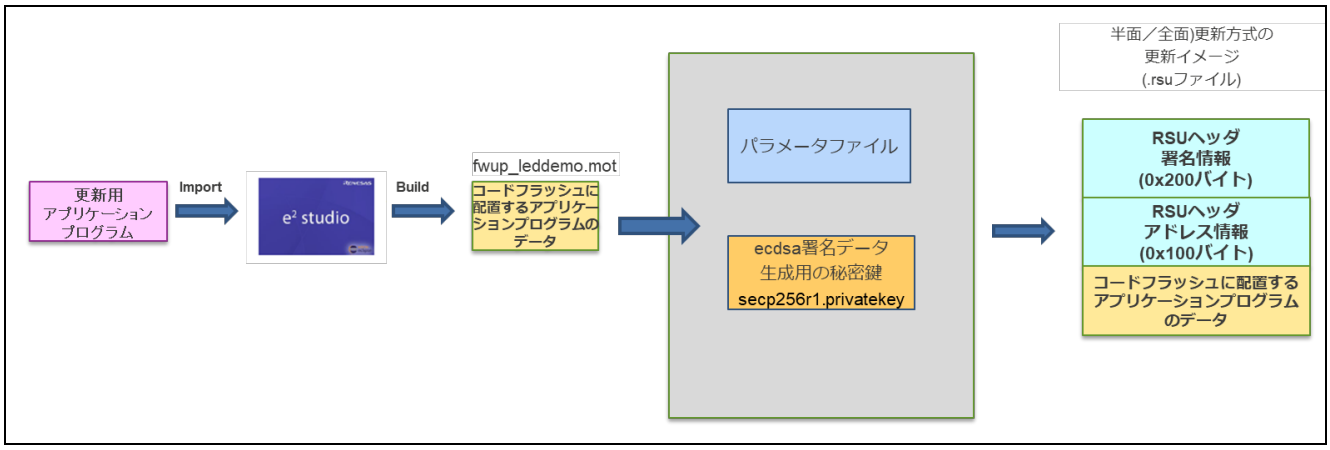


図 5-2 半面更新方式/全面更新方式の更新イメージ

・パラメータファイルは、イメージファイルを生成するために必要なデバイスのアドレス情報などが記載された CSV 形式のファイルです。

・ECDSA 署名値生成用の秘密鍵は、ファームウェアアップデートモジュールでのイメージ検証方式を ecdsa に指定した場合に使用します。

5.2.2　初期イメージファイル

初期イメージファイルは、RSU ヘッダとアプリケーションプログラムのデータにブートローダのプログラムデータを加えたものです。また、図 5-3 に初期イメージファイルの構成図を示します。

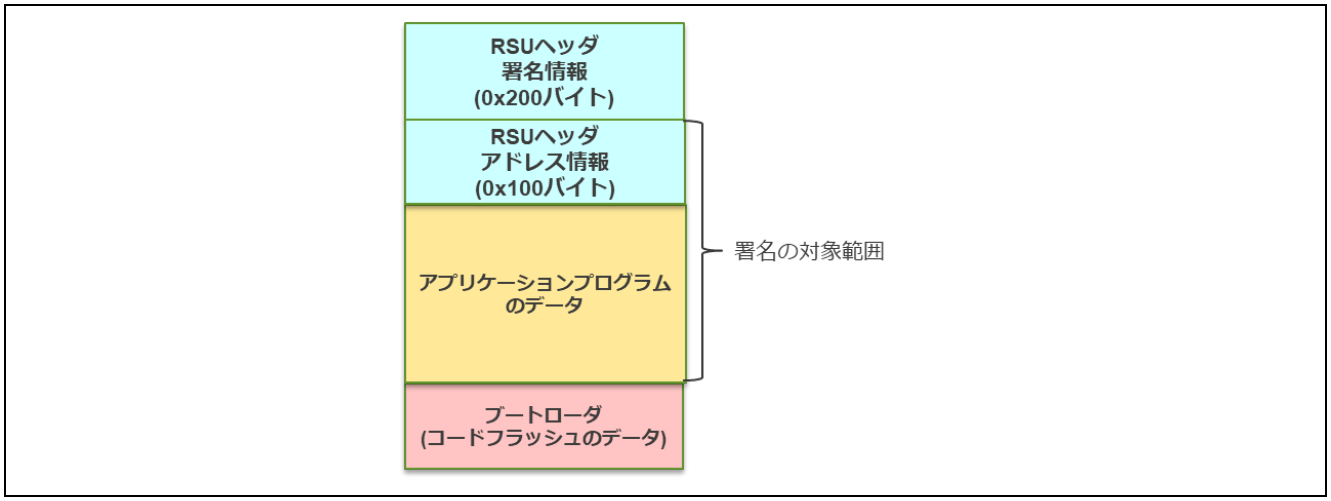


図 5-3 初期イメージファイル（半面／全面更新方式）の構成

半面更新方式および全面更新方式の初期イメージファイルでは、コードフラッシュのメイン面に配置するブートローダのデータは、ユーザが生成したブートローダのファイル(boot_loader.mot)のデータをそのまま使用します。

初期イメージファイルを生成するメカニズムについては、図 5-4 を参照してください。

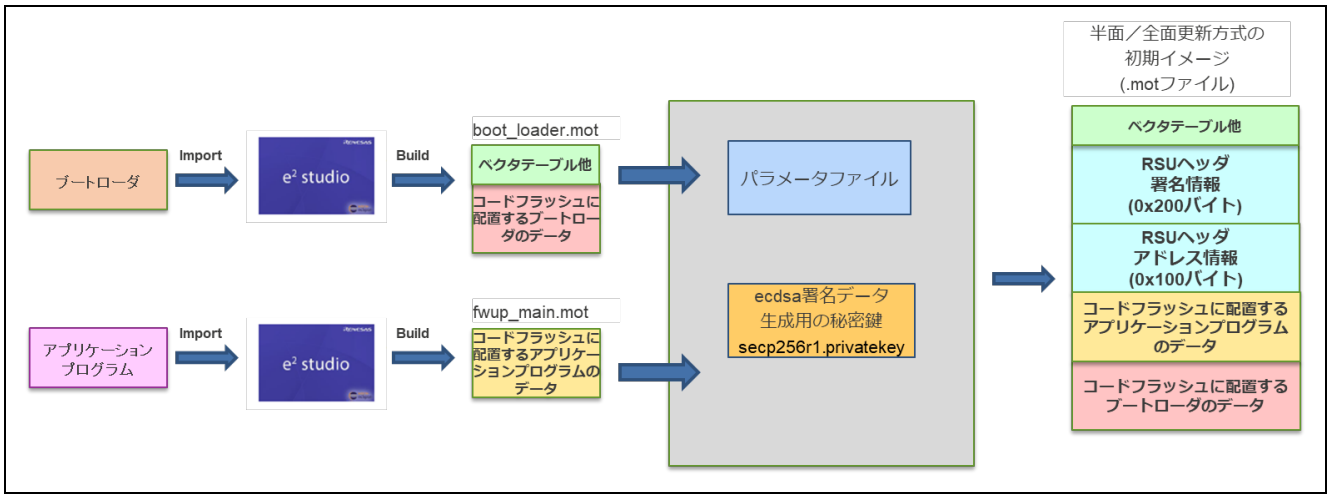


図 5-4 半面更新方式および全面更新方式の初期イメージ

- パラメータファイルは、イメージファイルを生成するために必要なデバイスのアドレス情報などが記載された CSV 形式のファイルです。
- ECDSA 署名値生成用の秘密鍵は、ファームウェアアップデートモジュールでのイメージ検証方式を ecdsa に指定した場合に使用します。

5.3　パラメータファイル

パラメータファイルとは、Renesas Image Generator がサンプルプログラムの初期イメージファイルや更新イメージファイルを生成するために必要な情報が記載されたものであり、リリースパッケージに Renesas Image Generator の Python プログラム一式として同梱されています。 お客様がデモプロジェクトを対象に初期イメージや更新イメージを生成する場合、パラメータファイルの内容を変更する必要はありません。

5.3.1　パラメータファイルの内容

パラメータファイルに記載している項目は、全てのデバイスで共通ですが、デバイス毎に設定内容が異なります。 表 5-5 に、RL78/G23 のデモプロジェクトのパラメータファイルの内容を示します。
また、図 5-5 に RL78/G23 半面更新方式のイメージ生成を行う際に参照するパラメータ、図 5-6 に RL78/G23 半面更新方式の初期イメージ生成を行う際に参照するパラメータ例を示します。

表 5-5 パラメータファイルの内容

パラメータ名	説明	設定内容例 RL78/G23
device Type	Linear Mode：半面更新方式または全面更新方式向け mot ファイル生成	Linear Mode
Code Flash Size(Dual Mode Only)	コードフラッシュのサイズ （RL78 では設定対象外）	No Used.
Bootloader Start Address	ブートローダの開始アドレス	0x000B1000
Bootloader End Address	ブートローダの終了アドレス	0x000BFFFF
User Program Start Address	メイン面のアプリケーションプログラムの開始アドレス	0x00001000
User Program End Address	メイン面のアプリケーションプログラムの終了アドレス	0x00058FFF
OFS Data Start Address	OFSM データ開始アドレス （RL78 では設定対象外）	No Used.
OFS Data End Address	OFSM データ終了アドレス （RL78 では設定対象外）	No Used.
Data Flash Start Address	データフラッシュ開始アドレス (データフラッシュのデータを生成しない場合は'No Used.'を設定します)	0x000F1000
Data Flash End Address	データフラッシュ終了アドレス (データフラッシュのデータを生成しない場合は'No Used.'を設定します)	0x000F2FFF
Near Data Start Address(RL78 Only)	RL78 用 Near ブートローダ開始アドレス	0x00000000
Near Data End Address(RL78 Only)	RL78 用 Near ブートローダ終了アドレス	0x00000FFF
Flash Write Size	フラッシュ書き込みサイズ(フラッシュへの 1 回の書き込みに必要なバイト数を 10 進数で指定します)	128

各パラメータに指定する数値は、Flash Write Size は 10 進数、その他パラメータは、16 進数（先頭に 0x を付加）で指定します。

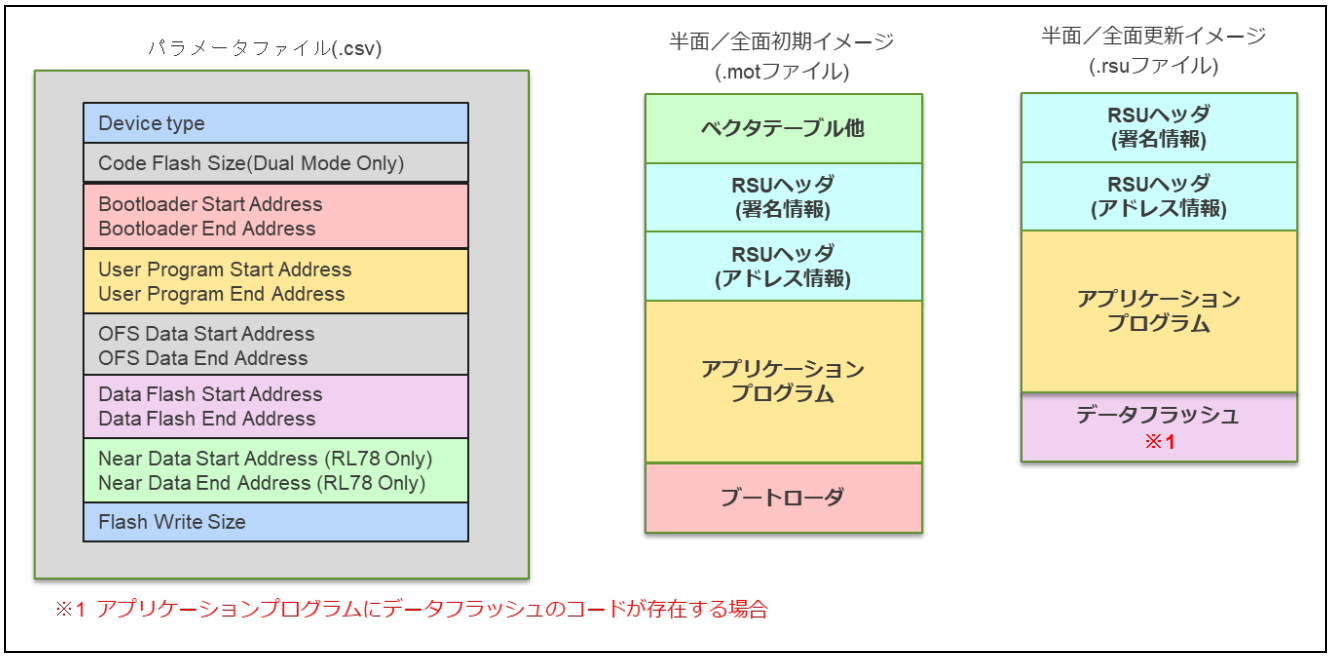


図 5-5 イメージファイルを生成する際に参照するパラメータ

・Device type は、初期イメージの生成方法を判別するために使用します。”Linear Mode”が設定されている場合、半面更新方式または全面更新方式向けの初期イメージを生成します。

・ブートローダのファイル(boot_loader.mot)を入力データとして、Bootloader Start Address から Bootloader End Address までの範囲をブートローダのコードフラッシュとして生成します。

・アプリケーションプログラムファイル(.mot) を入力データとして、User Program Start Address から User Program End Address までの範囲をアプリケーションプログラムのコードフラッシュとして生成します。

・アプリケーションプログラムファイル(.mot) を入力データとして、Data Flash Start Address から Data Flash End Address までの範囲をデータフラッシュとして生成します。（本デモプロジェクトでは、データフラッシュは使用していません）

・ブートローダのファイル(boot_loader.mot)を入力データとして、Near Data Flash Start Address (RL78 Only)から Near Data Flash End Address (RL78 Only)までの範囲をブートローダのベクタテーブル他をコードとして生成します。

・Flash Write Size は、フラッシュに書き込む際の最小単位として RSU ヘッダ（アドレス情報）のデータサイズの設定に使用します。

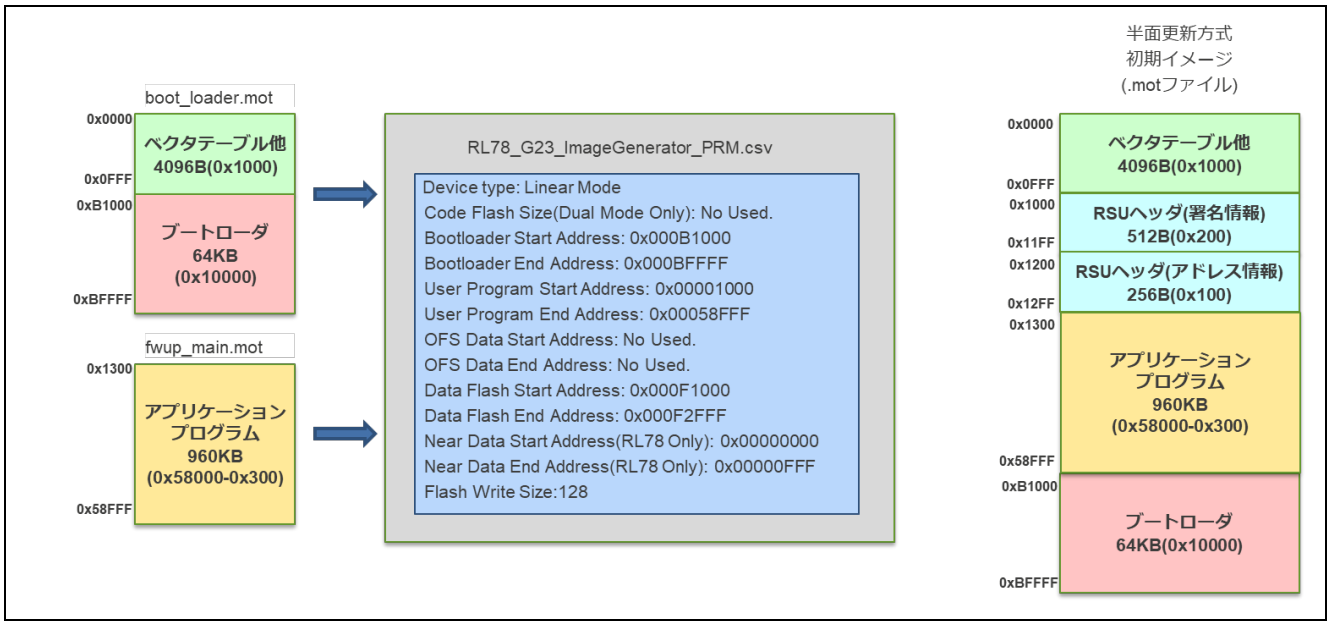


図 5-6 RL78/G23 の半面更新方式の初期イメージ生成を行う際に参照するパラメータ例

6. 付録

6.1 動作確認環境

本モジュールの動作確認環境を以下に示します。

表 6-1 動作確認環境 (CC-RL)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2023-07
C コンパイラ	ルネサスエレクトロニクス製 CC-RL V1.11.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	リトルエンディアン
モジュールのリビジョン	Rev.2.01
使用ボード	RL78/G22-48p Fast Prototyping Board 48 ピン版（RTK7RLG220C00000BJ） RL78/G23-128p Fast Prototyping Board 128 ピン版（RTK7RLG230CSN000BJ） RL78/G24-64p Fast Prototyping Board 64 ピン版（RTK7RLG240C00000BJ）

表 6-2 動作確認環境 (IAR)

項目	内容
統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 5.10.1
C コンパイラ	IAR Systems 製 IAR C/C++ Compiler for Renesas RL78 version 5.10.1 IAR Assembler for Renesas RL78 version 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	リトルエンディアン
モジュールのリビジョン	Rev.2.01
使用ボード	RL78/G22-48p Fast Prototyping Board 48 ピン版（RTK7RLG220C00000BJ） RL78/G23-128p Fast Prototyping Board 128 ピン版（RTK7RLG230CSN000BJ） RL78/G24-64p Fast Prototyping Board 64 ピン版（RTK7RLG240C00000BJ）

6.2　デモプロジェクトの動作環境

本モジュールは複数のコンパイラに対応しています。本モジュールを使用するにあたり、コンパイラ毎に異なる設定を以下に示します。

6.2.1　RL78/G23 の動作確認環境

実行環境と接続図を以下に示します。

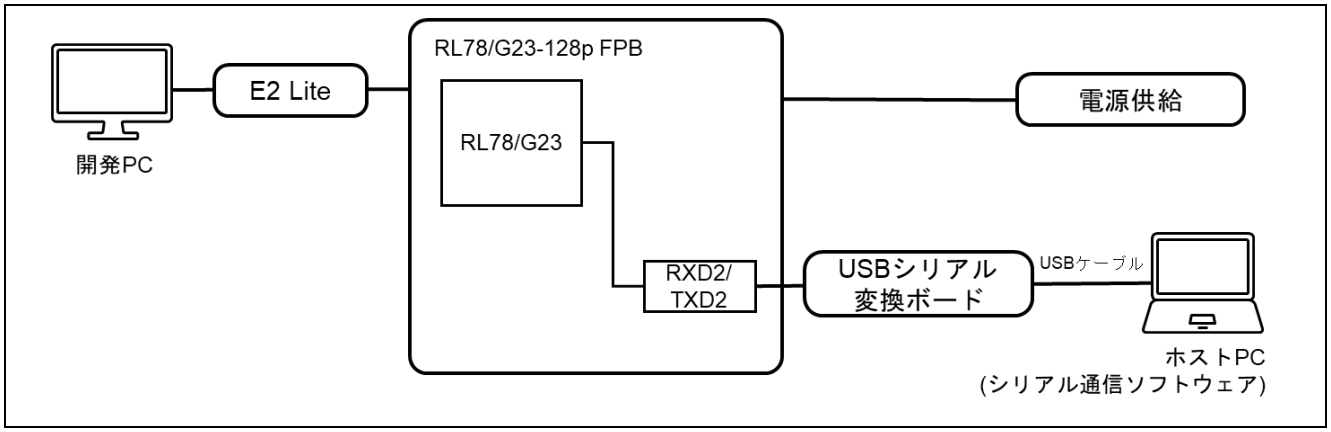


図 6-1 RL78/G23-128p FPB 半面更新方式（バッファ面は内部フラッシュ）の機器接続図

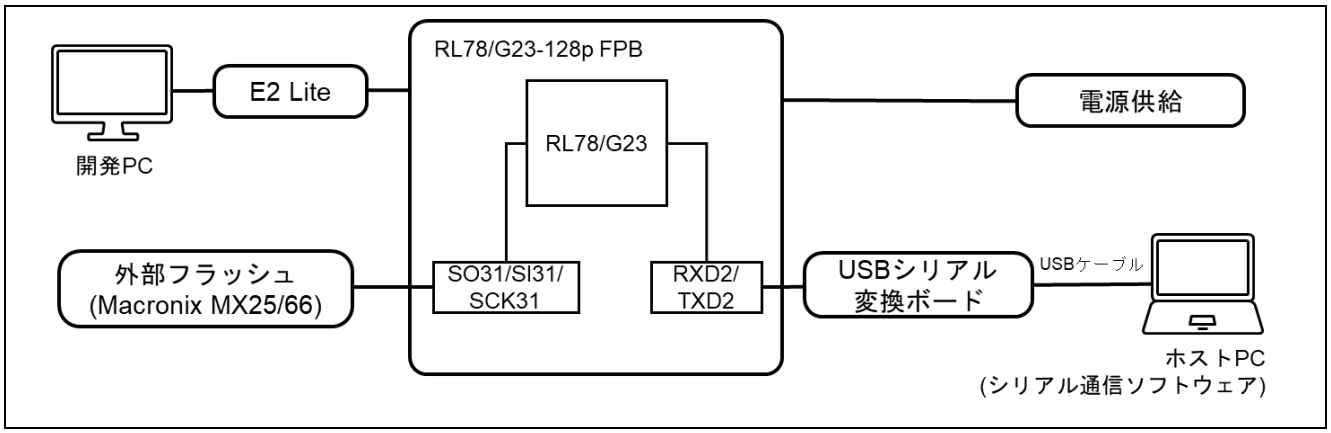


図 6-2 RL78/G23-128p FPB 全面更新方式（バッファ面は外部フラッシュ）の機器接続図

ピンアサインについて、以下図に示します。

■UART(Red)

Arduino J8		USB-UART
2	P61(RTS)	CTS
3	RXD2	TX
4	TXD2	RX

■External Flash(Green)

MCU Header J1		MX25/66L	Note
14	SO31	SI	1Kohm pull up
15	SI31	SO	1Kohm pull up
16	SCK31	SCLK	1Kohm pull up
18	P56(CS)	CE#	1Kohm pull up

Arduino J5		MX25/66L	Note
4	3V3	VCC	
6	GND	GND	

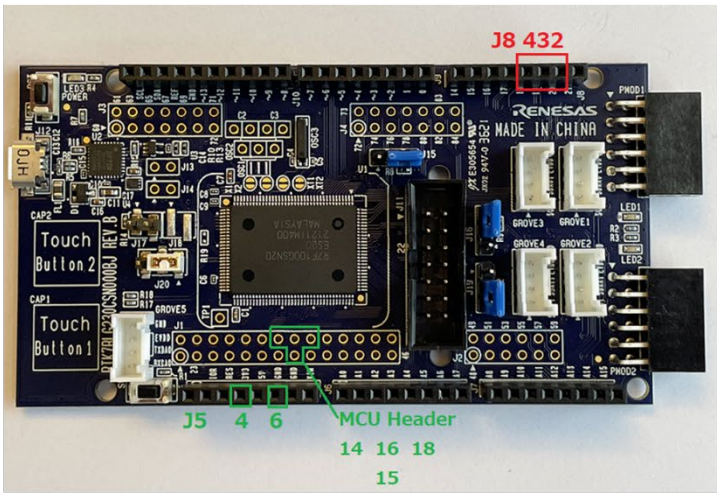


図 6-3 RL78/G23-128p FPB ピン情報

6.2.1.1　半面更新方式（バッファ面は内部フラッシュ）のデモプロジェクトのメモリマップ

RL78/G23 の半面更新方式のデモプロジェクトのメモリマップおよびコンフィグレーション設定のメモリマップについて、以下に示します。

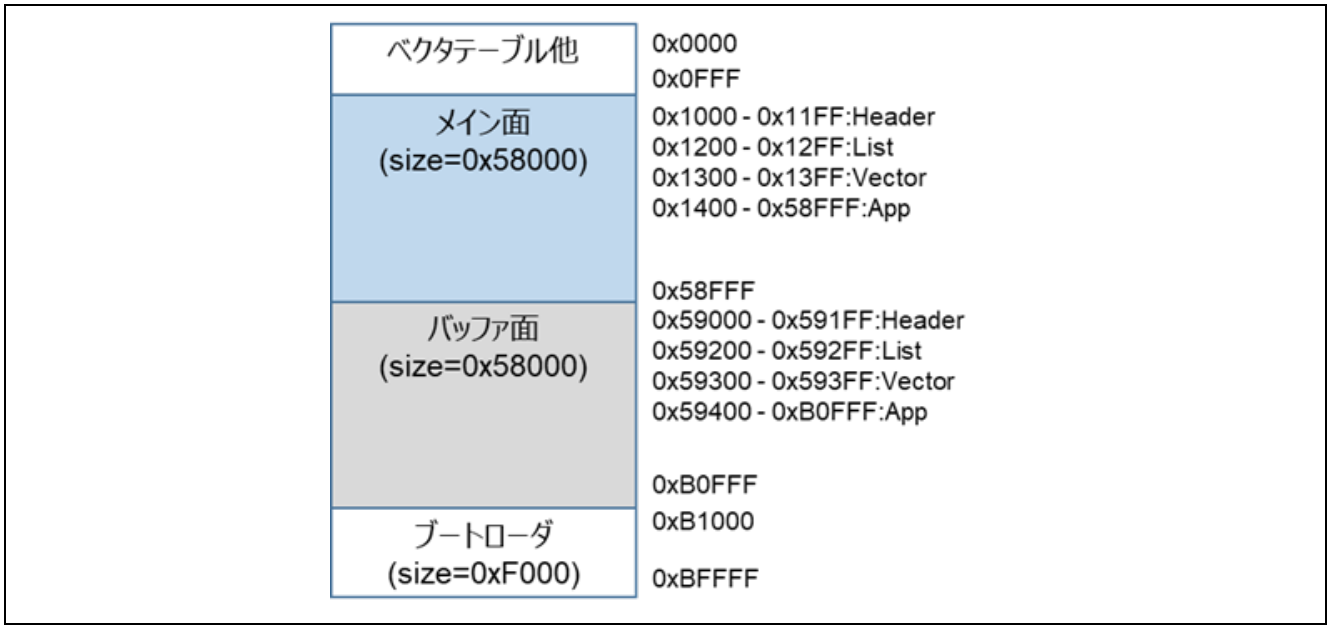


図 6-4 RL78/G23 の半面更新方式（バッファ面は内部フラッシュ）のデモプロジェクトのメモリマップ

表 6-3 RL78/G23 の半面更新方式（バッファ面は内部フラッシュ）のコンフィグ設定

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	1	1
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x59000	0x59000
FWUP_CFG_AREA_SIZE	0x58000	0x58000
FWUP_CFG_CF_BLK_SIZE	2048	2048
FWUP_CFG_CF_W_UNIT_SIZE	128	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L	0x0000	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	4096	4096
FWUP_CFG_DF_ADDR_L	0xF1000	0xF1000
FWUP_CFG_DF_BLK_SIZE	256	256
FWUP_CFG_DF_NUM_BLKS	32	32
FWUP_CFG_FWUPV1_COMPATIBLE	0	0
FWUP_CFG_SIGNATURE_VERIFICATION	0	0
FWUP_CFG_PRINTF_DISABLE	0	0

6.2.1.2　全面更新方式（バッファ面は外部フラッシュ）のデモプロジェクトのメモリマップ

RL78/G23 の全面更新方式のデモプロジェクトのメモリマップおよびコンフィグレーション設定のメモリマップについて、以下に示します。

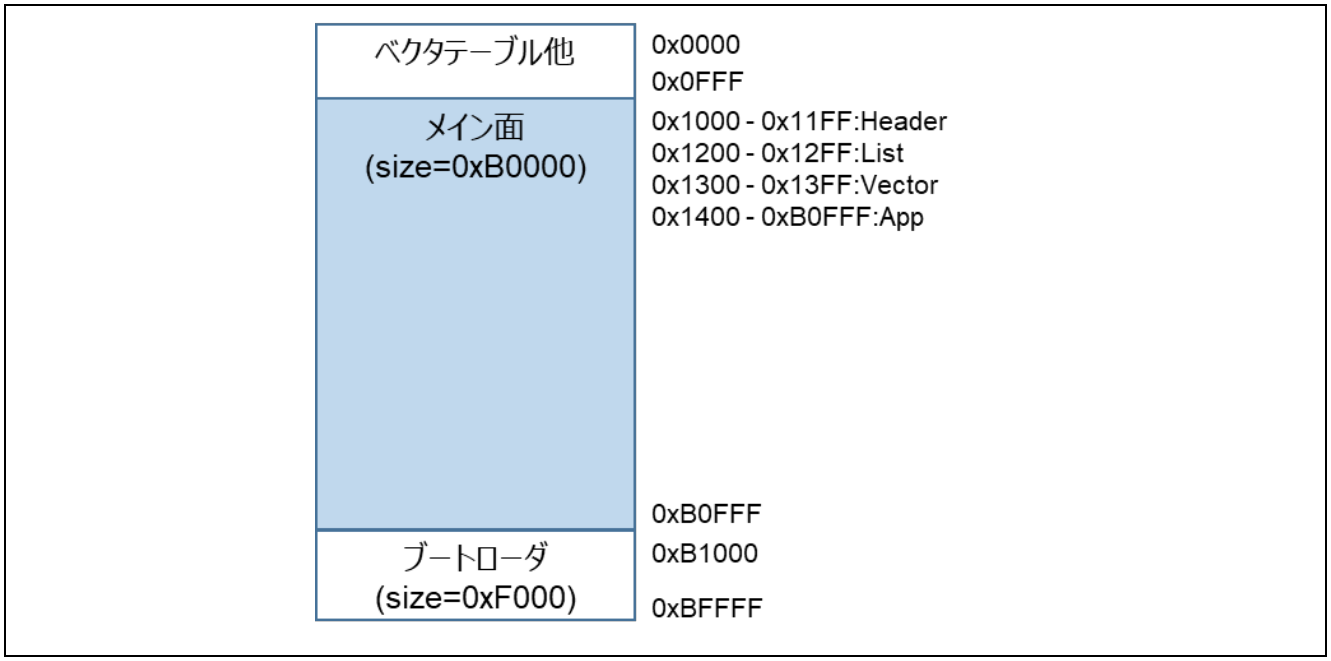


図 6-5　RL78/G23 の全面更新方式（バッファ面は外部フラッシュ）のデモプロジェクトのメモリマップ

表 6-4 RL78/G23 の全面更新方式（バッファ面は外部フラッシュ）のコンフィグ設定

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x59000	0x59000
FWUP_CFG_AREA_SIZE	0xB0000	0xB0000
FWUP_CFG_CF_BLK_SIZE	2048	2048
FWUP_CFG_CF_W_UNIT_SIZE	128	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L	0x0000	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	4096	4096
FWUP_CFG_DF_ADDR_L	0xF1000	0xF1000
FWUP_CFG_DF_BLK_SIZE	256	256
FWUP_CFG_DF_NUM_BLKS	32	32
FWUP_CFG_FWUPV1_COMPATIBLE	0	0
FWUP_CFG_SIGNATURE_VERIFICATION	0	0
FWUP_CFG_PRINTF_DISABLE	0	0

6.2.2　RL78/G24 の動作確認環境

実行環境と接続図を以下に示します。

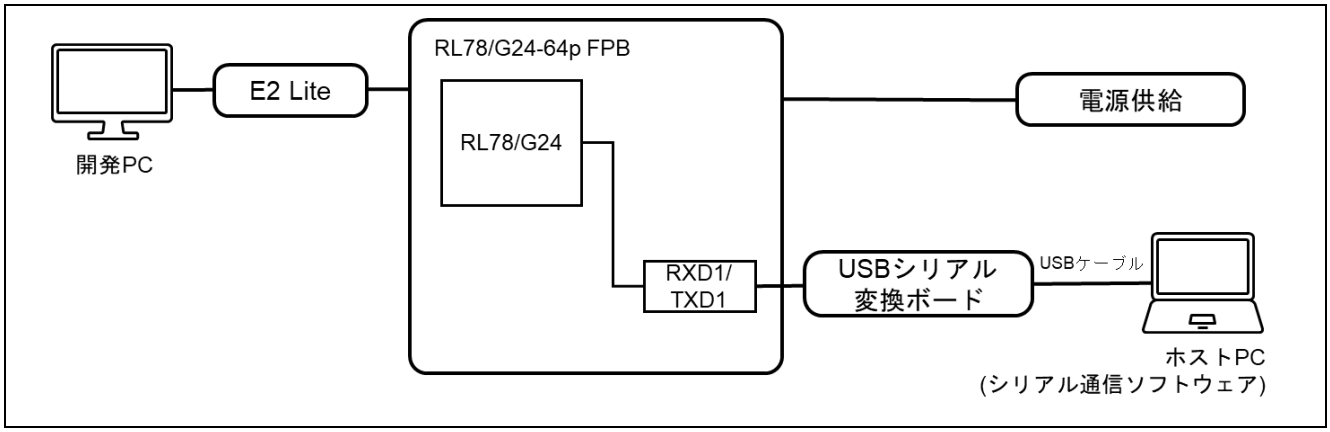


図 6-6 RL78/G24-64p FPB 半面更新方式（バッファ面は内部フラッシュ）の機器接続図

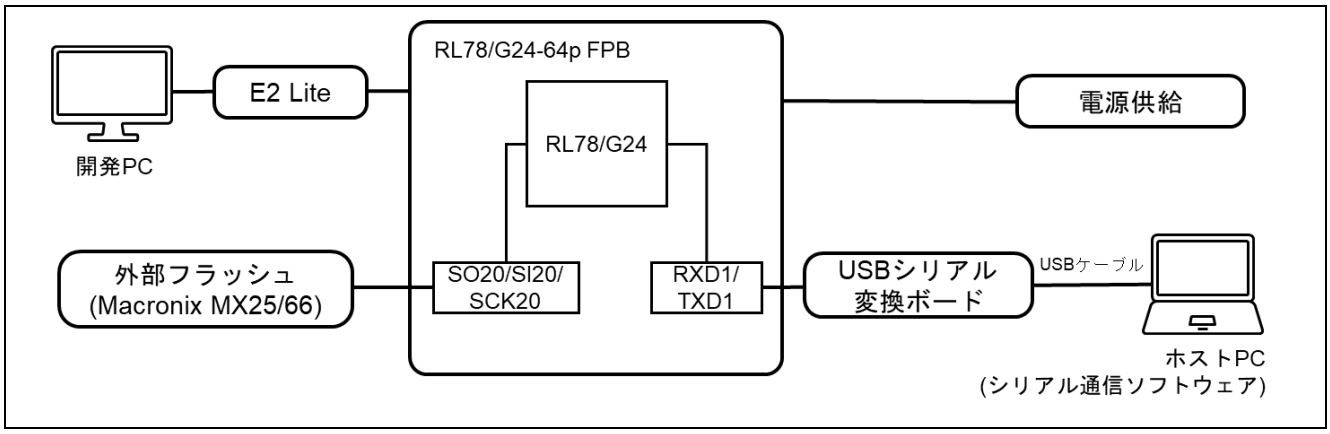


図 6-7 RL78/G24-64p FPB 全面更新方式（バッファ面は外部フラッシュ）の機器接続図

ピンアサインについて、以下図に示します。

■UART(Red)

Arduino J5		USB-UART	Note
1	RXD1	TX	
2	TXD1	RX	
3	P140(RTS)	CTS	

■External Flash(Green)

Arduino J5,J6		MX25/66L	Note
J6 3	SO20	SI	1Kohm pull up
J6 2	SI20	SO	1Kohm pull up
J5 7	SCK20	SCLK	1Kohm pull up
J5 6	P16(CS)	CE#	1Kohm pull up

Arduino J3		MX25/66L	Note
4	3V3	VCC	
6	GND	GND	

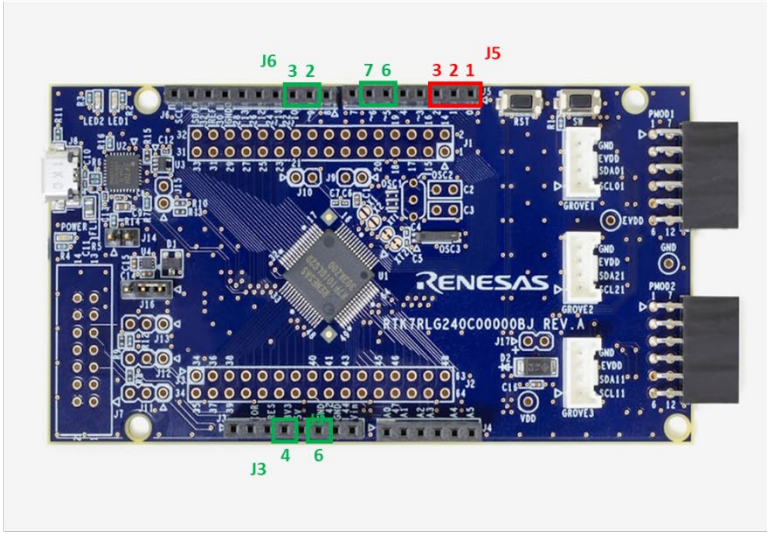


図 6-8 RL78/G24-64p FPB ピン情報

6.2.2.1　半面更新方式（バッファ面は内部フラッシュ）のデモプロジェクトのメモリマップ

RL78/G24 の半面更新方式（バッファ面は内部フラッシュ）のデモプロジェクトのメモリマップおよびコンフィグレーション設定のメモリマップについて、以下に示します。

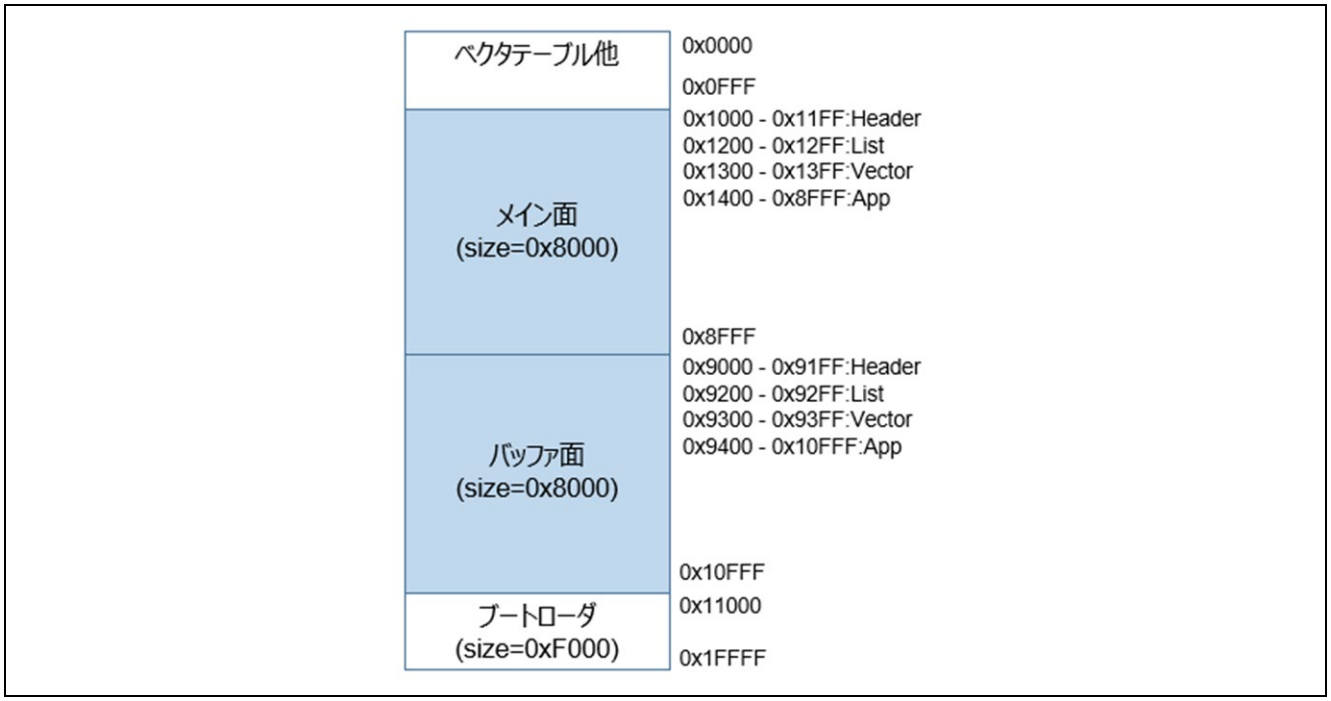


図 6-9 RL78/G24 の半面更新方式（バッファ面は内部フラッシュ）のデモプロジェクトのメモリマップ

表 6-5 RL78/G24 の半面更新方式（バッファ面は内部フラッシュ）のコンフィグ設定

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	1	1
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x9000	0x9000
FWUP_CFG_AREA_SIZE	0x8000	0x8000
FWUP_CFG_CF_BLK_SIZE	2048	2048
FWUP_CFG_CF_W_UNIT_SIZE	128	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L	0x0000	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	4096	4096
FWUP_CFG_DF_ADDR_L	0xF1000	0xF1000
FWUP_CFG_DF_BLK_SIZE	256	256
FWUP_CFG_DF_NUM_BLKS	16	16
FWUP_CFG_FWUPV1_COMPATIBLE	0	0
FWUP_CFG_SIGNATURE_VERIFICATION	0	0
FWUP_CFG_PRINTF_DISABLE	0	0

6.2.2.2　全面更新方式（バッファ面は外部フラッシュ）のデモプロジェクトのメモリマップ

RL78/G24 の全面更新方式（バッファ面は外部フラッシュ）のデモプロジェクトのメモリマップおよびコンフィグレーション設定のメモリマップについて、以下に示します。

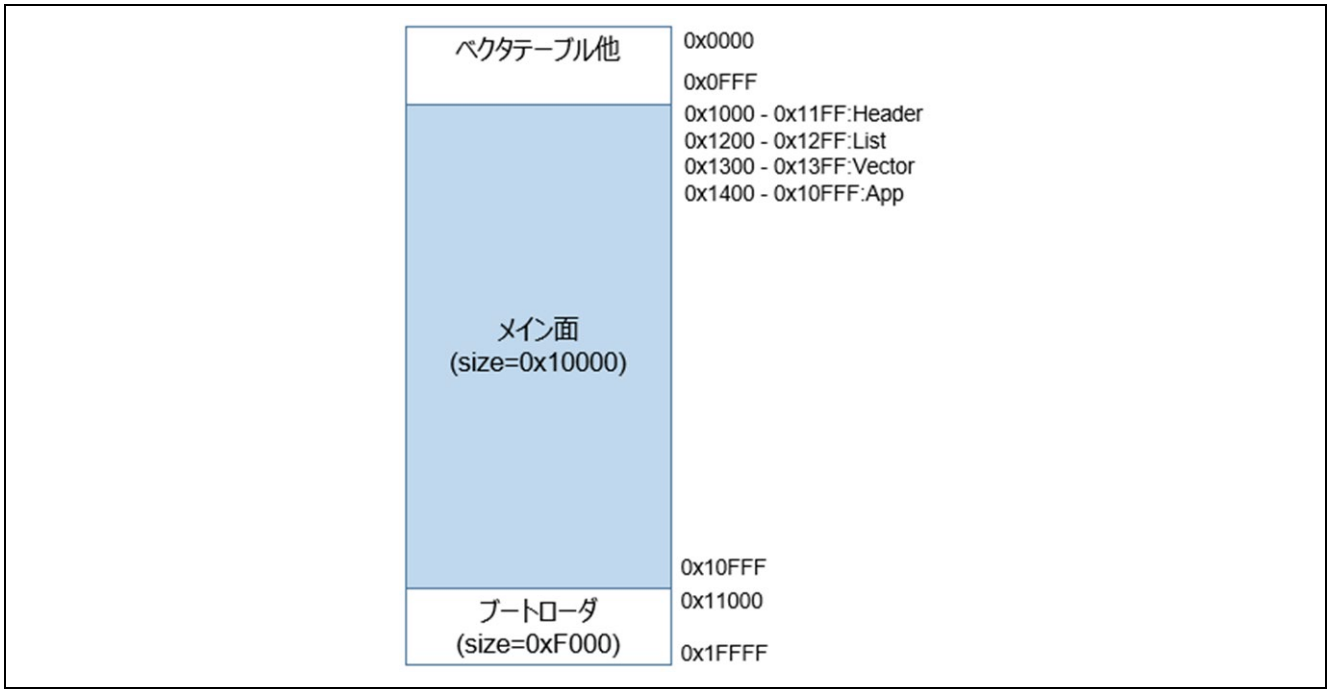


図 6-10　RL78/G24 の全面更新方式（バッファ面は外部フラッシュ）のデモプロジェクトのメモリマップ

表 6-6 RL78/G24 の全面更新方式（バッファ面は外部フラッシュ）のコンフィグ設定

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x9000	0x9000
FWUP_CFG_AREA_SIZE	0x10000	0x10000
FWUP_CFG_CF_BLK_SIZE	2048	2048
FWUP_CFG_CF_W_UNIT_SIZE	128	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L	0x0000	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	4096	4096
FWUP_CFG_DF_ADDR_L	0xF1000	0xF1000
FWUP_CFG_DF_BLK_SIZE	256	256
FWUP_CFG_DF_NUM_BLKS	16	16
FWUP_CFG_FWUPV1_COMPATIBLE	0	0
FWUP_CFG_SIGNATURE_VERIFICATION	0	0
FWUP_CFG_PRINTF_DISABLE	0	0

6.2.3　RL78/G22 の動作確認環境

実行環境と接続図を以下に示します。

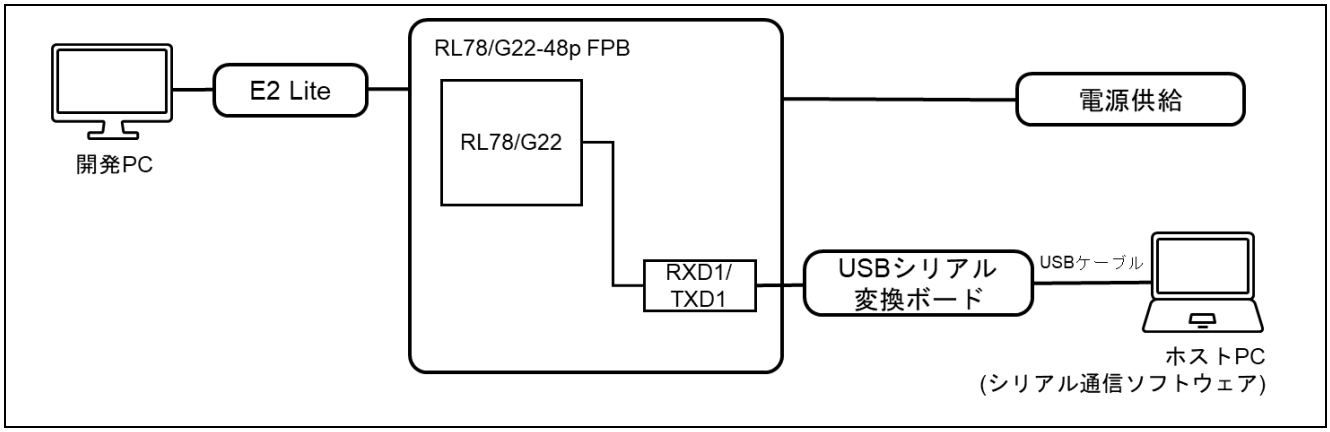


図 6-11 RL78/G22-48p FPB 全面更新方式（バッファなし）の機器接続図

ピンアサインについて、以下図に示します。

Arduino J7		USB-UART
1	RXD1	TX
2	TXD1	RX
3	P140	CTS

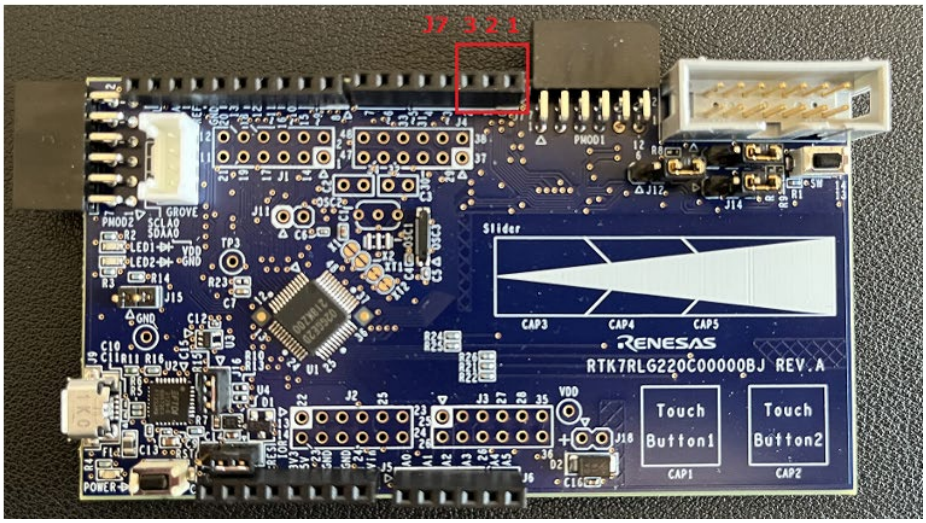


図 6-12 RL78/G22-48p FPB ピン情報

6.2.3.1 全面更新方式（バッファ無し）のデモプロジェクトのメモリマップ

RL78/G22 の全面更新方式（バッファ無し）のデモプロジェクトのメモリマップおよびコンフィグレーション設定のメモリマップについて、以下に示します。

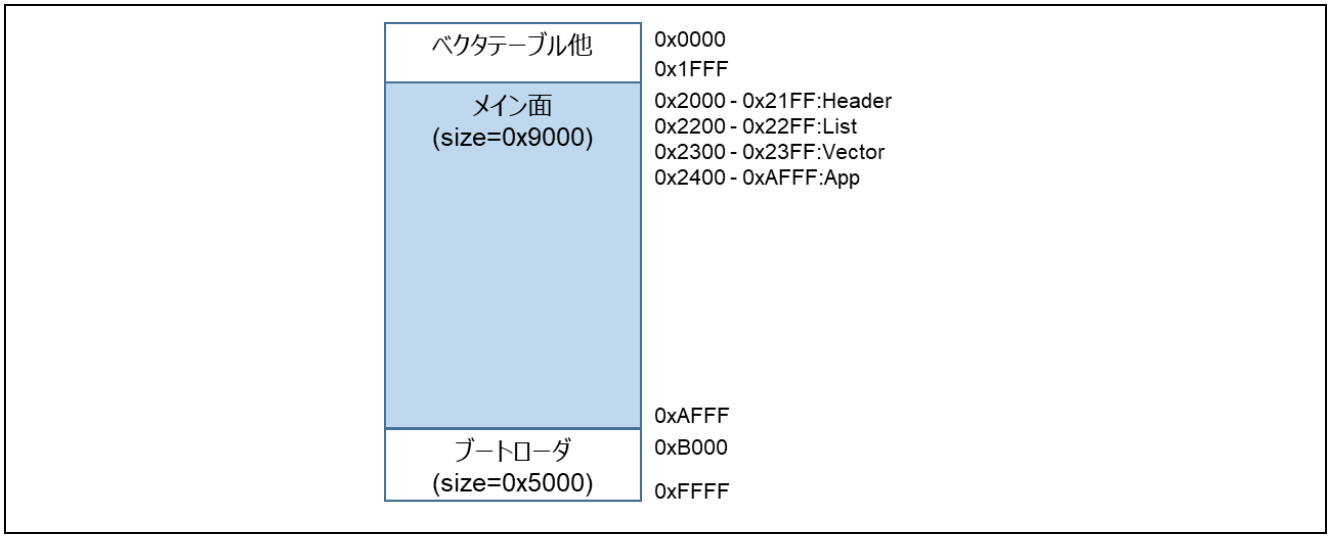


図 6-13　RL78/G22 の全面更新方式（バッファ無し）のデモプロジェクトのメモリマップ

表 6-7 RL78/G22 の全面更新方式（バッファ無し）のコンフィグ設定

Configuration options in r_fwup _config.h	
パラメータ名	boot_loader
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x2000
FWUP_CFG_AREA_SIZE	0x9000
FWUP_CFG_CF_BLK_SIZE	2048
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	4096
FWUP_CFG_DF_ADDR_L	0xF1000
FWUP_CFG_DF_BLK_SIZE	256
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	1
FWUP_CFG_PRINTF_DISABLE	0

6.3　デモプロジェクトで利用するオープンソースのライセンス情報

本製品のデモプロジェクトは、オープンソース TinyCrypt を使用しています。暗号ライブラリに TinyCrypto を使用する場合は、TinyCrypt のライセンス条項が定める使用条件を遵守する必要があります。

TinyCrypt のライセンス条項は以下を確認してください。

URL：https://github.com/intel/tinycrypt

ライセンス：https://github.com/intel/tinycrypt/blob/master/LICENSE

7. 注意事項

7.1 ブートローダからアプリケーションへの遷移時の注意事項

ブートローダのサンプルプログラムからアプリケーションへの遷移時には、ブートローダの周辺機能の設定がアプリケーションに引き継がれることになります。

サンプルのブートローダで使用する周辺機能に関しては、ブートローダ終了時に各モジュールの API 関数を close した状態にします。また、その他の設定に関してはスマート・コンフィグレータを使用した際の初期値となります。

お客様にて、ブートローダのサンプルプログラムを改造して使用される場合は、ブートローダにて設定した周辺機能の設定がアプリケーション側に引き継がれる事になりますので、ブートローダからアプリケーションに遷移する前に周辺機能の設定を初期化するか、アプリケーション側と周辺機能の設定を共通化されることを推奨します。

アプリケーションを作成される際は、ブートローダの実装を考慮して開発頂きますようお願いします。

表 7-1 ブートローダで使用する周辺機能の注意事項

周辺機能	ブートローダでの設定および注意事項
ボードに関する機能	スマート・コンフィグレータにて組み込んだ際の初期値となります。ブートローダでは設定を変更しておりません。 注）RL78/G24 向けのデモプロジェクトについては、デフォルト値と異なる PLL 設定：32MHz に設定されています。
フラッシュメモリに関する機能	フラッシュメモリに関する周辺機能に関しては、Close 処理を行いアプリケーションに遷移します。
シリアル通信に関する機能	シリアル通信に関する周辺機能に関しては、Close 処理を行いアプリケーションに遷移します。 ブートローダで使用する SCI のチャンネルは 6.2 デモプロジェクトの動作環境の製品毎の機器接続図を参照ください。
オプション設定メモリ	オプション設定メモリに関してはブートローダとアプリケーションプログラムで一意の値を設定してください。
その他の機能	その他の機能の設定に関してはスマート・コンフィグレータを使用した際の初期値となります。 また、割り込み許可フラグを割り込み禁止にし、アプリケーションに遷移します。

7.2 ブートローダ領域のセキュリティ対策について

お客様にて、ファームウェアアップデートモジュールを製品化する場合、ブートローダ（boot_loader）を展開しているコードフラッシュの領域にプロテクトをかけることをお勧めします。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
2.00	2023.7.20	-	初版発行
2.01	2023.11.22	1 12-13 17 19 22 22 22 23 23 24 55 56-65 67	・RL78/G24 追加 ・フォルダ構成にデバイス追加 ・コンフィグレーション設定に FWUP_CFG_CF_W_UNIT_SIZE と FWUP_CFG_FWUPV1_COMPATIBLE を追加 ・ROM/RAM/スタックにデバイス追加 ・R_FWUP_EraseArea 関数のパラメータ追加 ・R_FWUP_GetImageSize 関数の説明追加 ・R_FWUP_WriteImageHeader 関数の戻り値追加 ・R_FWUP_WriteImageProgram 関数のパラメータ追加 ・R_FWUP_WriteImage 関数の戻り値追加 ・R_FWUP_VerifyImage 関数の戻り値追加 ・動作確認環境の使用ボード追加 ・動作確認環境にデバイス追加 ・注意事項を追加
2.02	2024.12.13	9-12 21-22 27 27 - - 35-51 52-60 61-69 82	・1.4 章を見直し ・1.6 章の図を 2.10 章に移動 ・3.6 R_FWUP_WriteImageHeader 関数を 3.13 章に移動 ・3.7 R_FWUP_WriteImageProgram 関数を 3.14 章に移動 ・4. Renesas Image Generator と 5. デモプロジェクトを入れ替え ・4. デモプロジェクト の記載内容を見直し ・4.6 デモプロジェクトのデバッグ方法 を追加 ・5. Renesas Image Generator を見直し ・7.2 ブートローダ領域のセキュリティ対策について を追加

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注入により、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　2020.10)

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。