RL78/G22, RL78/G23, RL78/G24, RL78/L23
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、RL78/L23 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）

RL78/L23（R7F100LPL）

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 ファミリ Renesas Flash Driver RL78 Type11 ユーザーズマニュアル(R20UT5539)
- 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 C Compiler Package for RL78 Family
- IAR Systems 製 IAR C/C++ Compiler for Renesas RL78
  IAR Assembler for Renesas RL78
- LLVM for Renesas RL78

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ためのミドルウェアです。

本モジュールは、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 および 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

また、本モジュールは、次の 2 つのプログラムで構成されています。

・アプリケーションプログラム：ユーザプログラム＋ファームウェアアップデートプログラム

・ブートローダ：アプリケーション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プログラム

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。

本モジュールは、次の 4 つのアップデート方式を提供します。各方式についての詳細は 1.3 をご確認ください。

・デュアルバンク（２バンク）方式
・半面更新方式
・全面更新方式（バッファ無し）
・全面更新方式（バッファ有り注）

注）外部フラッシュメモリを使用します。詳細は 1.3.4 をご確認ください。

さらに、ユーティリティツールとして Renesas Image Generator を提供します。本ツール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アプリケーションプログラムおよ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。

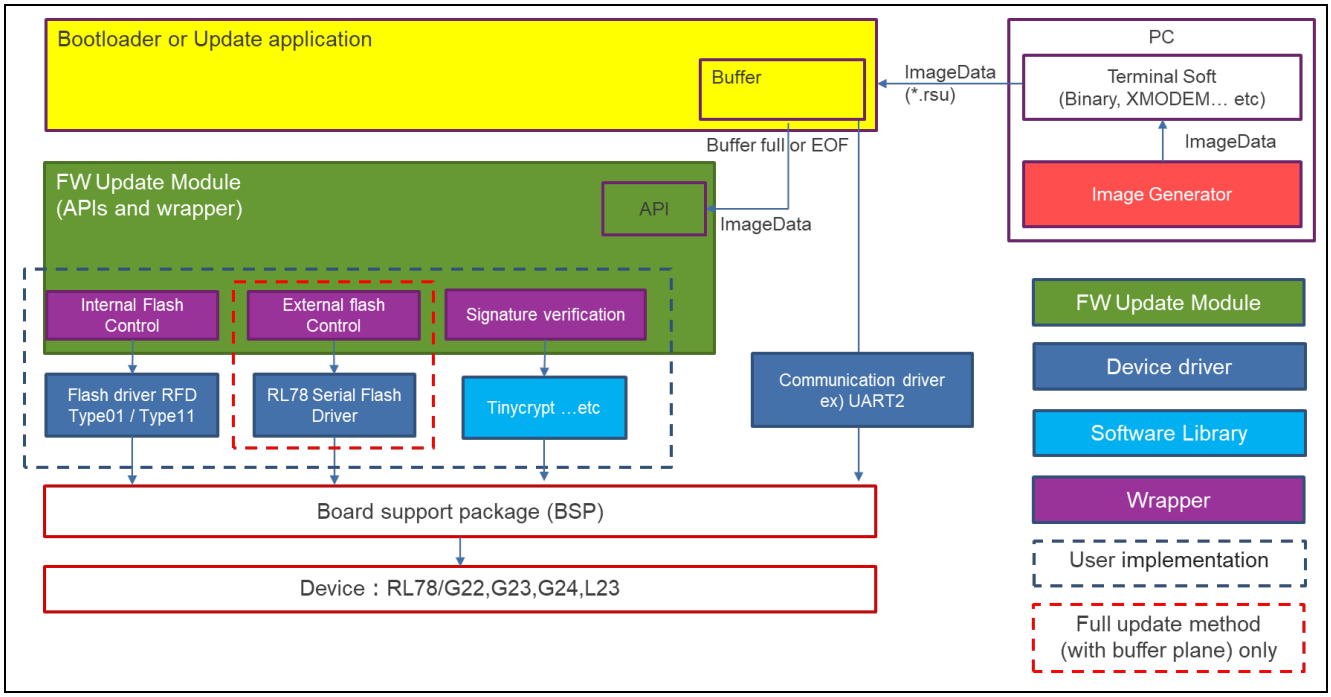


図 1-1 ブートローダおよびアプリケーションプログラムのモジュール構成

表 1-1 ブートローダおよびアプリケーション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、RL78/L23 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：RL78/G22、RL78/G23、RL78/G24 RFD RL78 Type11：RL78/L23	スマート・コンフィグレータによる自動生成
CSI	Config_CSI20 注：RL78/G24 Config_CSI31 注：RL78/G23、RL78/L23	スマート・コンフィグレータによる自動生成
Serial Flash	r_nor_flash 注	スマート・コンフィグレータによる自動生成
Crypt library	Tinycrypt	ラッパー関数に実装

注）全面更新方式（バッファ有り）でのみ必要なモジュールです。それ以外の方式では不要です。

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全面更新方式（バッファ無し）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。ただ、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イメージに戻すことはできません。

デバイスおよびフラッシュメモリの容量毎にアップデート方式のサポート状況が異なりますので以下に詳細を示します。

表 1-2 各製品のアップデート方式のサポート状況

Flash 容量 / 製品名	768KB	512KB	384KB	256KB	192KB	128KB	96KB	64KB
RL78/G22	－	－	－	－	－	－	－	L
RL78/G23	L	L	L	L	L	L	L	－
RL78/G24	－	－	－	－	－	L	－	L
RL78/L23	－	DB/L	－	DB/L	－	L	－	L

DB：デュアルバンク（２バンク）方式のファームウェアアップデート対応製品
L：半面／全面更新方式のファームウェアアップデート対応製品
－：対象外
赤字：デモプロジェクト提供製品

1.3.1　デュアルバンク（2 バンク）方式

更新イメージを内蔵フラッシュのバッファ面に格納し、その正当性を検証した後で、バンクスワップを行いメイン面とバッファ面を入れ替えます。

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、バンクスワップ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内蔵フラッシュのメモリサイズは、内蔵フラッシュメモリをデュアルバンク（2 バンク）機能により 2 分割するため、バンク 1 面のサイズからブートローダを引いたサイズになります。

1.3.1.1　デュアルバンク（2 バンク）方式のアップデート動作

内蔵フラッシュメモリのデュアルバンク（2 バンク）機能を使用し更新イメージをバッファ面に格納し、バンクスワップ機能でバンクを入れ替えることによりファームウェアアップデートを行います。

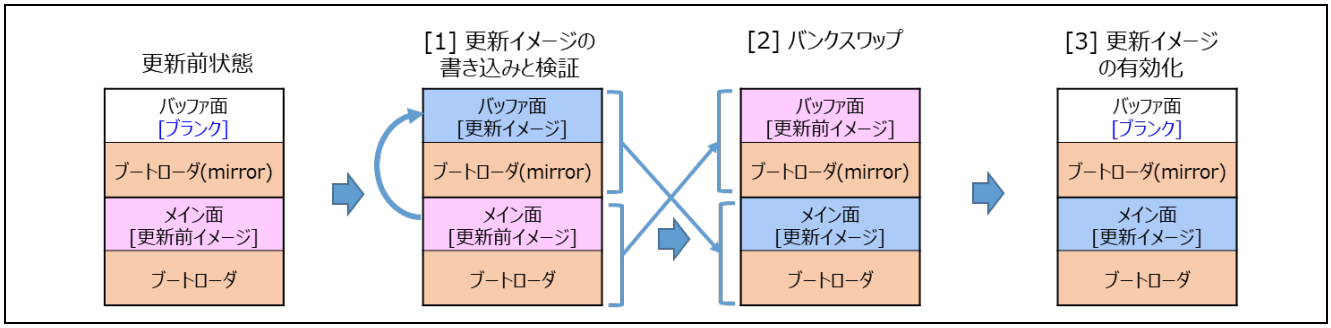


図 1-2 デュアルバンク（2 バンク）方式のアップデートの動作

[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] バンクスワップ
検証結果が正常であれば、バンクスワップを実施する。

[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。
（デモプロジェクトでは、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　半面更新方式

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。

1.3.2.1　半面更新方式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。

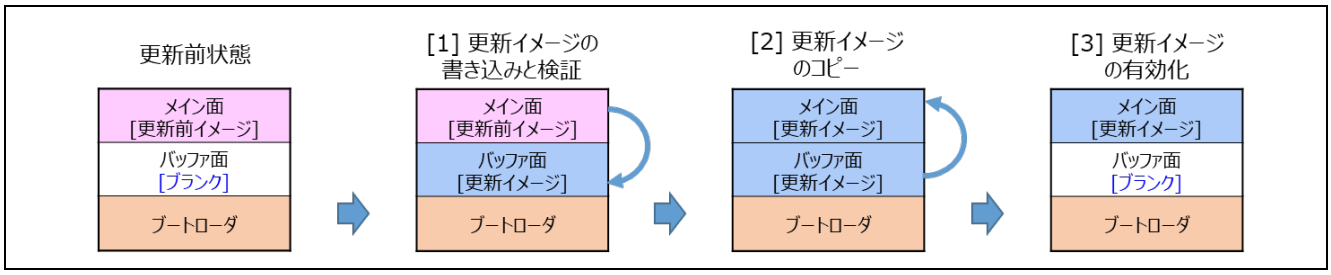


図 1-3 半面更新方式のアップデートの動作

[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。

1.3.3　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。

1.3.3.1　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。

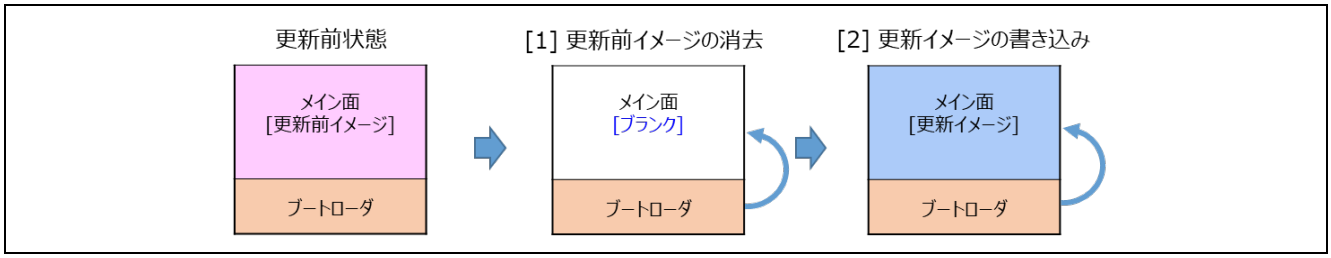


図 1-4 全面更新方式（バッファ無し）のアップデートの動作

[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。

1.3.4　全面更新方式（バッファ有り）

更新イメージを外部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。

1.3.4.1　全面更新方式（バッファ有り）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。

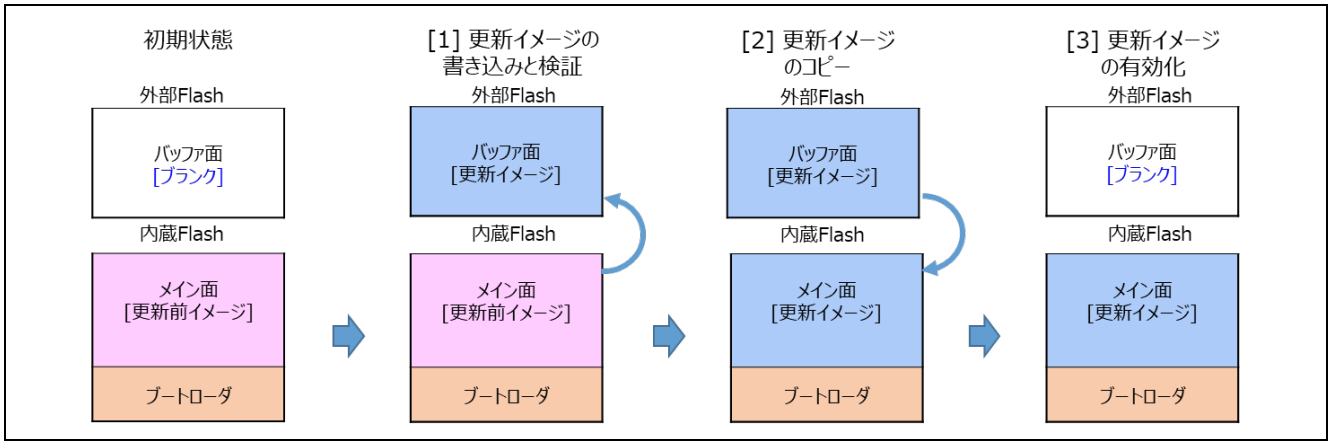


図 1-5 全面更新方式（バッファ有り）のファームウェアアップデートの動作

[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。

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 を用いたデュアルバンク（2 バンク）方式の初期状態の構築方法

Renesas Image Generator を用いたデュアルバンク（2 バンク）方式の初期状態の構築の流れを以下に示します。

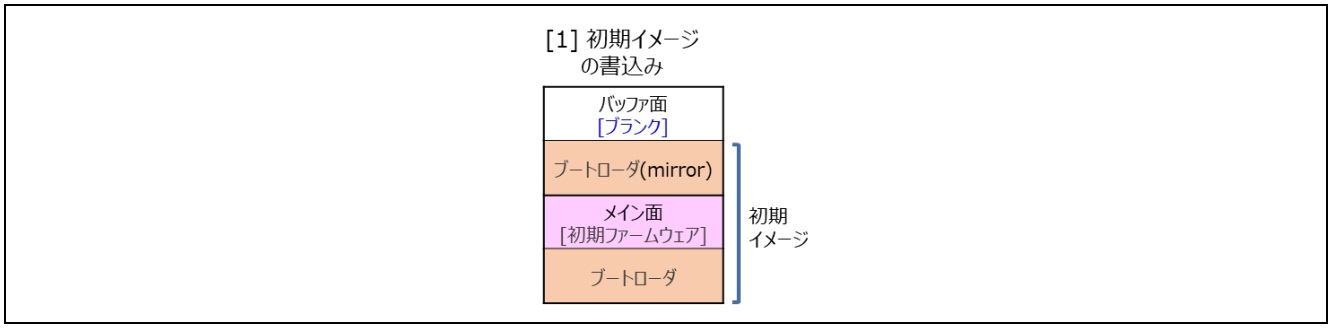


図 1-6 Renesas Image Generator を用いた初期状態の構築（デュアルバンク（2 バンク）方式の例）

[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。

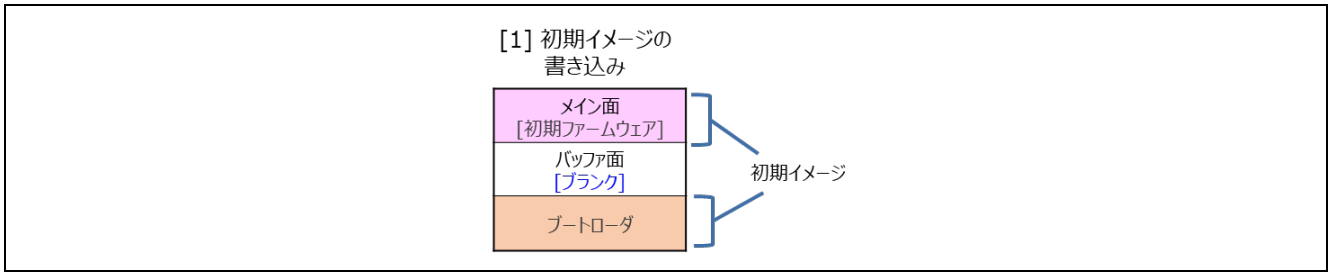


図 1-7 Renesas Image Generator を用いた初期状態の構築（半面更新方式の例）

[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。

1.4.3　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。

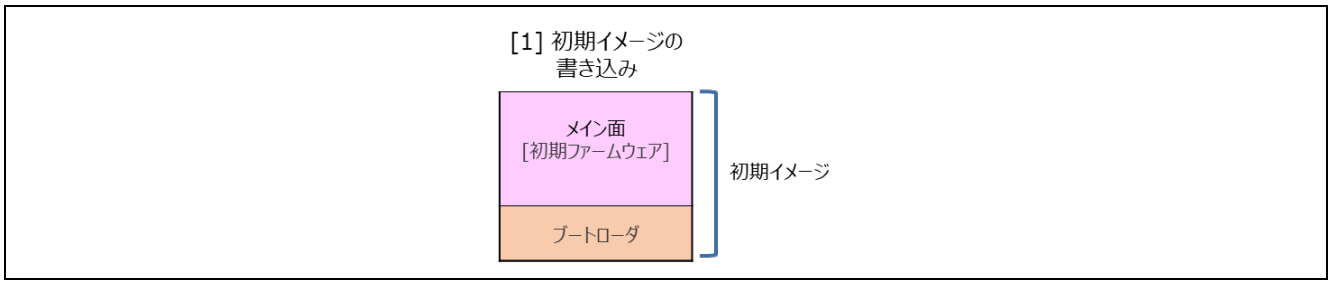


図 1-8 Renesas Image Generator を用いた初期状態の構築（全面更新方式の例）

[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。

1.4.4　ブートローダを用いたデュアルバンク（2 バンク）方式の初期状態の構築方法

ブートローダを用いたデュアルバンク（2 バンク）方式の初期状態の構築の流れを以下に示します。

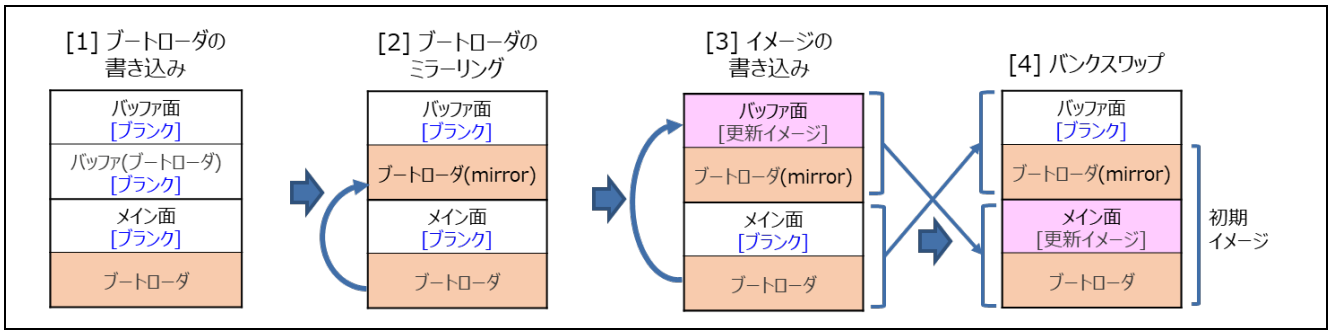


図 1-9 ブートローダを用いた初期状態の構築（デュアルバンク（2 バンク）方式の例）

[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] ブートローダのミラーリング

ブートローダが起動しバッファ面にブートローダをミラーリングする。

[3] 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、バッファ面に書き込み、書き込んだイメージを検証する。

[4] バンクスワップ

検証に問題が無ければ、バンクスワップを行い終了する。

1.4.5　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。

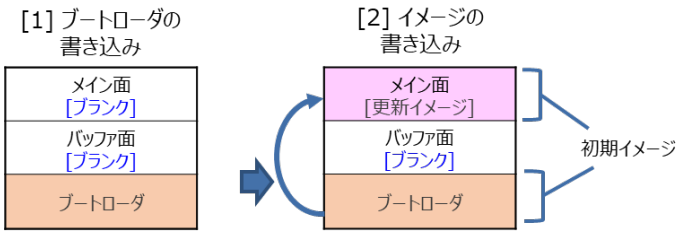


図 1-10 ブートローダを用いた初期状態の構築（半面更新方式）

[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。

1.4.6　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。

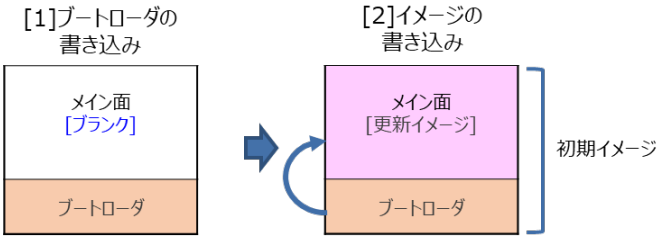


図 1-11 ブートローダを用いた初期状態の構築（全面更新方式）

[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 1-3 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
r01an6374xx0205-rl78g23-fwupdate.zip¥	
├demos	デモプロジェクト
│　├──r_fwup	ファームウェアアップデートモジュール
│　└──rl	
│　　├──modules	ドライバとライブラリ、ラッパー
│　　│　├──3rd_party	
│　　│　│　└──tinycrypt	暗号ライブラリ
│　　│　├──etc	
│　　│　│　└──base64	Base64 decode
│　　│　└──flash	Flash ラッパー
│　　├──rl78g22-fpb	FPB-RL78G22 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（CC-RL と同じ構成）
│　　│　　└──llvm	LLVM 版（CC-RL と同じ構成）
│　　├──rl78g23-fpb	FPB-RL78G23-128p 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（CC-RL と同じ構成）
│　　│　　└──llvm	LLVM 版（CC-RL と同じ構成）
│　　├──rl78g24-fpb	FPB-RL78G24 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（CC-RL と同じ構成）
│　　│　　└──llvm	LLVM 版（CC-RL と同じ構成）
│　　└──rl78l23-fpb	FPB-RL78L23 向け
│　　　├──dualbank(2bank)	デュアルバンク（2 バンク）方式
│　　　│　├──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（CC-RL と同じ構成）
│　　　│　└──llvm	LLVM 版（CC-RL と同じ構成）
│　　　└──linear	デュアルバンク以外の方式

フォルダ名	説明
│　　├──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（CC-RL と同じ構成）
│　　└──llvm	LLVM 版（CC-RL と同じ構成）
└RenesasImageGenerator	Renesas Image Generator
├──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 (CC-RL、IAR 向け)
└──RL*78_xxxx*_ImageGenerator_PRM_llvm.csv	デモプロジェクト用パラメータファイル (LLVM 向け)

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 1-4 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	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、AWS S3 (OTA)を行うお客様向けの仕様となります。
AWS S3 (OTA)を使わず、ファームウェアアップデートモジュール Rev.2.0x 以降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)
- Renesas Flash Driver RL78 Type11 (RFD)
- Serial NOR Flash Memory 制御モジュール Software Integration System (r_nor_flash)

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：デュアルバンク（2 バンク）方式（RL78/L23 のみ） 1：半面更新方式 2：全面更新方式（バッファ無し） 3：全面更新方式（バッファ有り）
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 (default)　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 (default) 1：SHA256
FWUP_CFG_PRINTF_DISABLE	ログ表示設定を指定します。 0：Enable 1：Disable (default)

2.7　デモ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デモプロジェクトの 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 v2.0.5

コンパイラバージョン：Renesas Electronics C Compiler Package for RL78 Family V1.16.0

IAR C/C++ Compiler for Renesas RL78 version 5.20.2

LLVM for Renesas RL78 17.0.1.202512

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)

LLVM

・最適化レベル：-Os

・ガーベジコレクション(--gc-sections)

2.7.1　FPB-RL78G23-128p 用のデモプロジェクト

半面更新方式のコンフィグレーション設定とメモリサイズ：

FWUP_CFG_UPDATE_MODE	1：Single bank with buffer.
FWUP_CFG_SIGNATURE_VERIFICATION	0：ECDSA+SHA256. (default)
FWUP_CFG_PRINTF_DISABLE	1：Disable. (default)

表 2-2 デモプロジェクト(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	LLVM
boot_loader	ROM	20674	26399	23664
	RAM	865	3192	1088
	スタック	522	1504	508

表 2-3 デモプロジェクト(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	LLVM
fwup_main	ROM	16790	24449	21079
	RAM	871	3701	958
	スタック	514	1510	508

2.7.2　FPB-RL78G24 用のデモプロジェクト

半面更新方式のコンフィグレーション設定とメモリサイズ：

FWUP_CFG_UPDATE_MODE	1：Single bank with buffer.
FWUP_CFG_SIGNATURE_VERIFICATION	0：ECDSA+SHA256. (default)
FWUP_CFG_PRINTF_DISABLE	1：Disable. (default)

表 2-4 デモプロジェクト(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	LLVM
boot_loader	ROM	20503	26941	23989
	RAM	879	3224	1074
	スタック	558	1504	508

表 2-5 デモプロジェクト(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	LLVM
fwup_main	ROM	17106	25737	21385
	RAM	877	3734	944
	スタック	514	1510	508

2.7.3　FPB-RL78G22 用のデモプロジェクト

全面更新方式（バッファ無し）のコンフィグレーション設定とメモリサイズ：

FWUP_CFG_UPDATE_MODE	2：Single bank without buffer.
FWUP_CFG_SIGNATURE_VERIFICATION	1：SHA256
FWUP_CFG_PRINTF_DISABLE	1：Disable. (default)

表 2-6 デモプロジェクト(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	LLVM
boot_loader	ROM	10318	11406	11197
	RAM	789	2057	960
	スタック	398	824	284

2.7.4　FPB-RL78L23 用のデモプロジェクト

半面更新方式のコンフィグレーション設定とメモリサイズ：

FWUP_CFG_UPDATE_MODE	1：Single bank with buffer.
FWUP_CFG_SIGNATURE_VERIFICATION	0：ECDSA+SHA256. (default)
FWUP_CFG_PRINTF_DISABLE	1：Disable. (default)

表 2-7 デモプロジェクト(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	LLVM
boot_loader	ROM	21045	26817	24060
	RAM	861	3204	1030
	スタック	522	1504	508

表 2-8 デモプロジェクト(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	LLVM
fwup_main	ROM	17123	24683	21448
	RAM	859	3201	900
	スタック	514	1508	508

デュアルバンク（2 バンク）方式のコンフィグレーション設定とメモリサイズ：

FWUP_CFG_UPDATE_MODE	0：Dual-bank
FWUP_CFG_SIGNATURE_VERIFICATION	0：ECDSA+SHA256. (default)
FWUP_CFG_PRINTF_DISABLE	1：Disable. (default)

表 2-9 デモプロジェクト(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	LLVM
boot_loader	ROM	20803	26938	24175
	RAM	725	3622	1028
	スタック	522	1512	508

表 2-10 デモプロジェクト(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	LLVM
fwup_main	ROM	17257	24723	21717
	RAM	730	4290	898
	スタック	514	1508	508

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およ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。

2.10.1 デュアルバンク（2 バンク）方式の実装例

デュアルバンク（2 バンク）方式のブートローダの実装例を図 2-1 に示します。

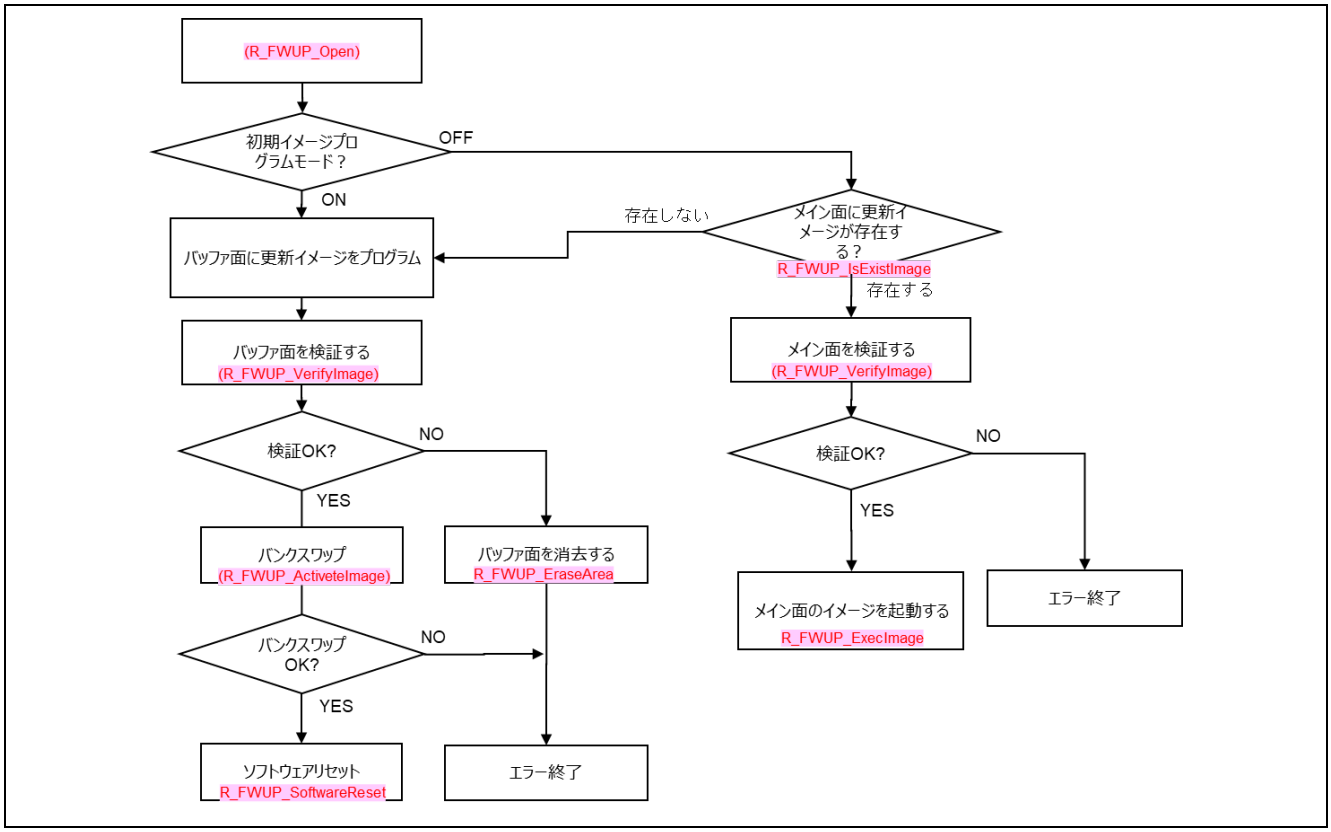


図 2-1 デュアルバンク（2 バンク）方式のブートローダの実装例

デュアルバンク（2 バンク）方式のアプリケーションプログラムの実装例を図 2-2 に示します。

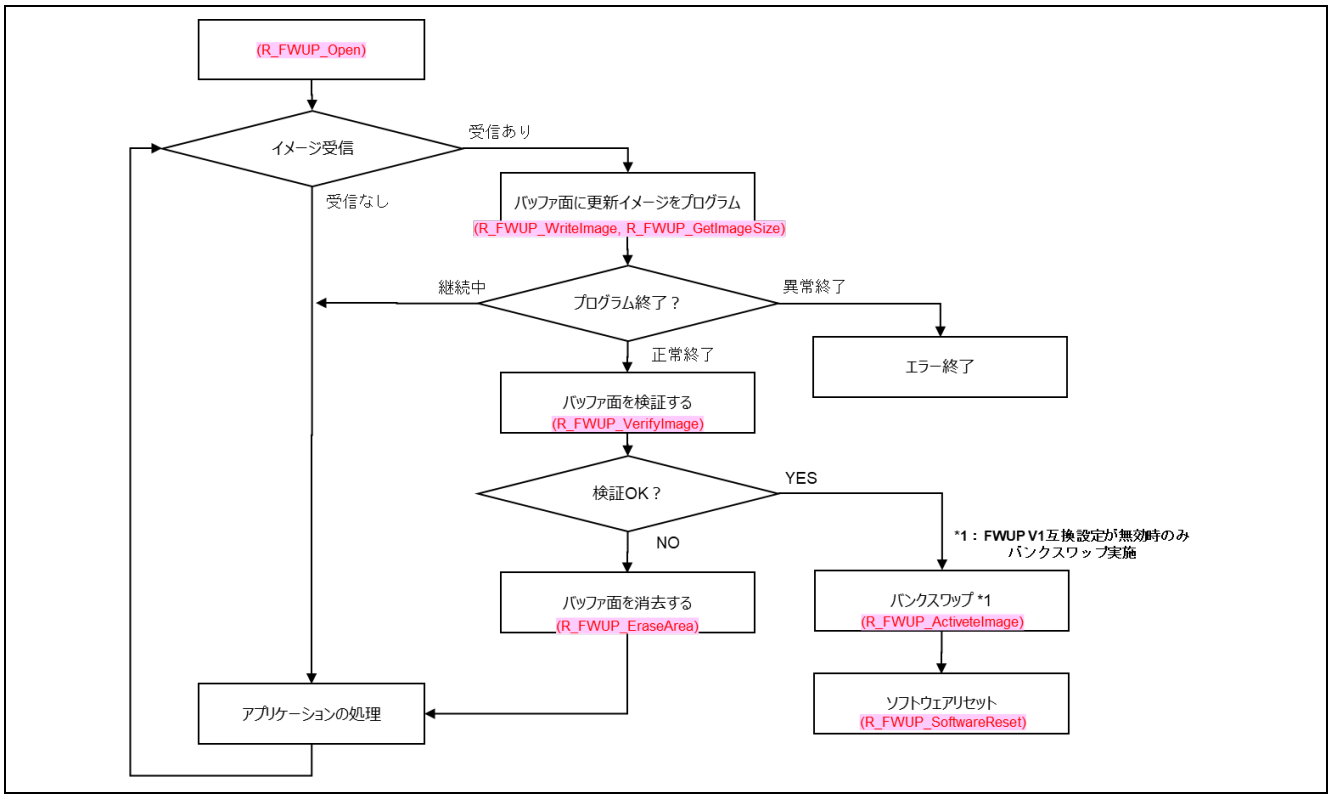


図 2-2 デュアルバンク（2 バンク）方式のアプリケーションプログラムの実装例

2.10.2 半面更新方式/全面更新方式（バッファ有り）の実装例

半面更新方式/全面更新方式（バッファ有り）のブートローダの実装例を図 2-3 に示します。

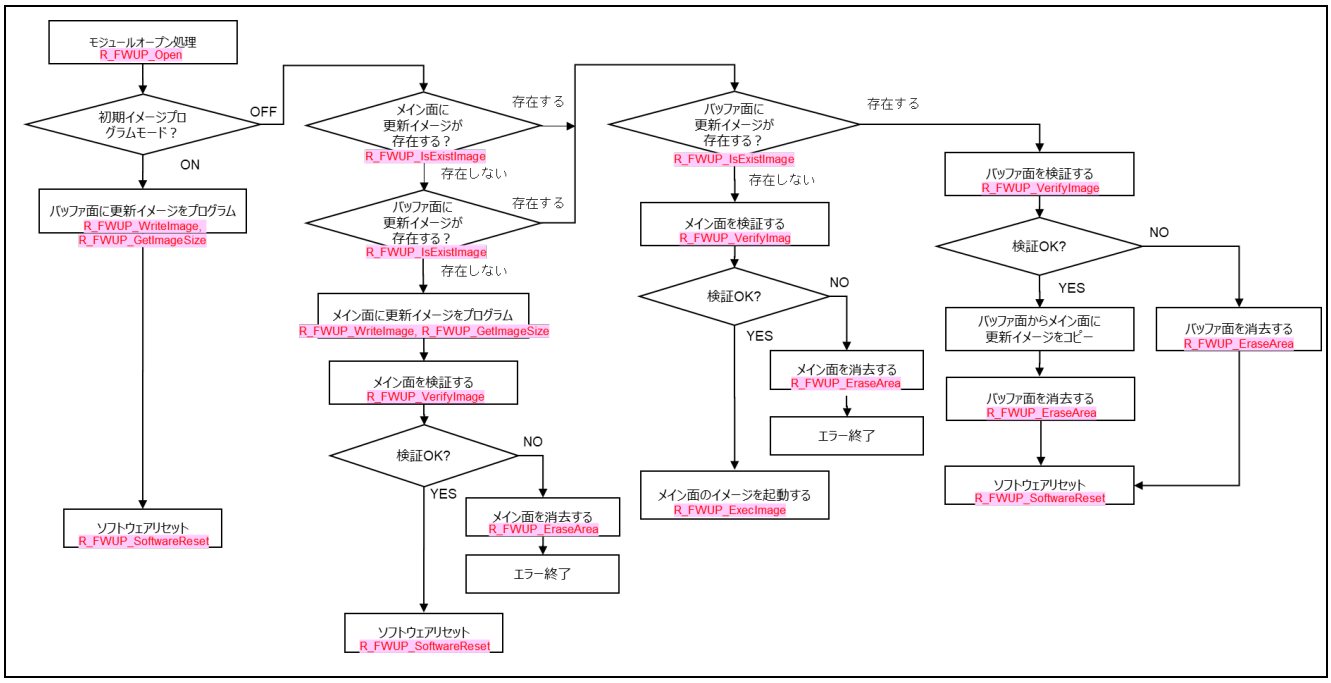


図 2-3 半面更新方式/全面更新方式（バッファ有り）のブートローダ実装例

半面更新方式/全面更新方式（バッファ有り）のアプリケーションプログラムの実装例を図 2-4 に示します。

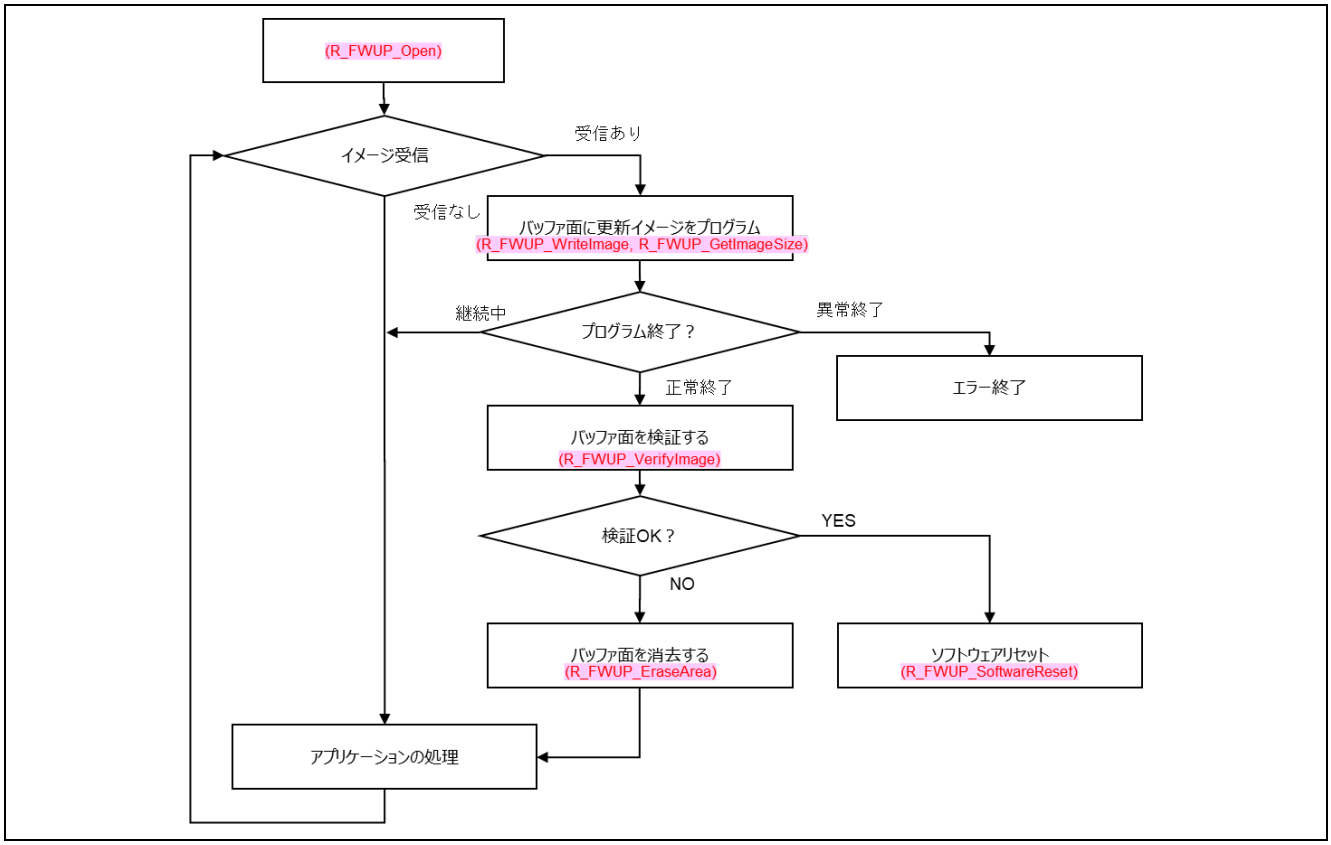


図 2-4 半面更新方式/全面更新方式（バッファ有り）のアプリケーションプログラム実装例

2.10.3 全面更新方式（バッファ無し）の実装例

全面更新方式（バッファ無し）のブートローダの実装例を図 2-5 に示します。

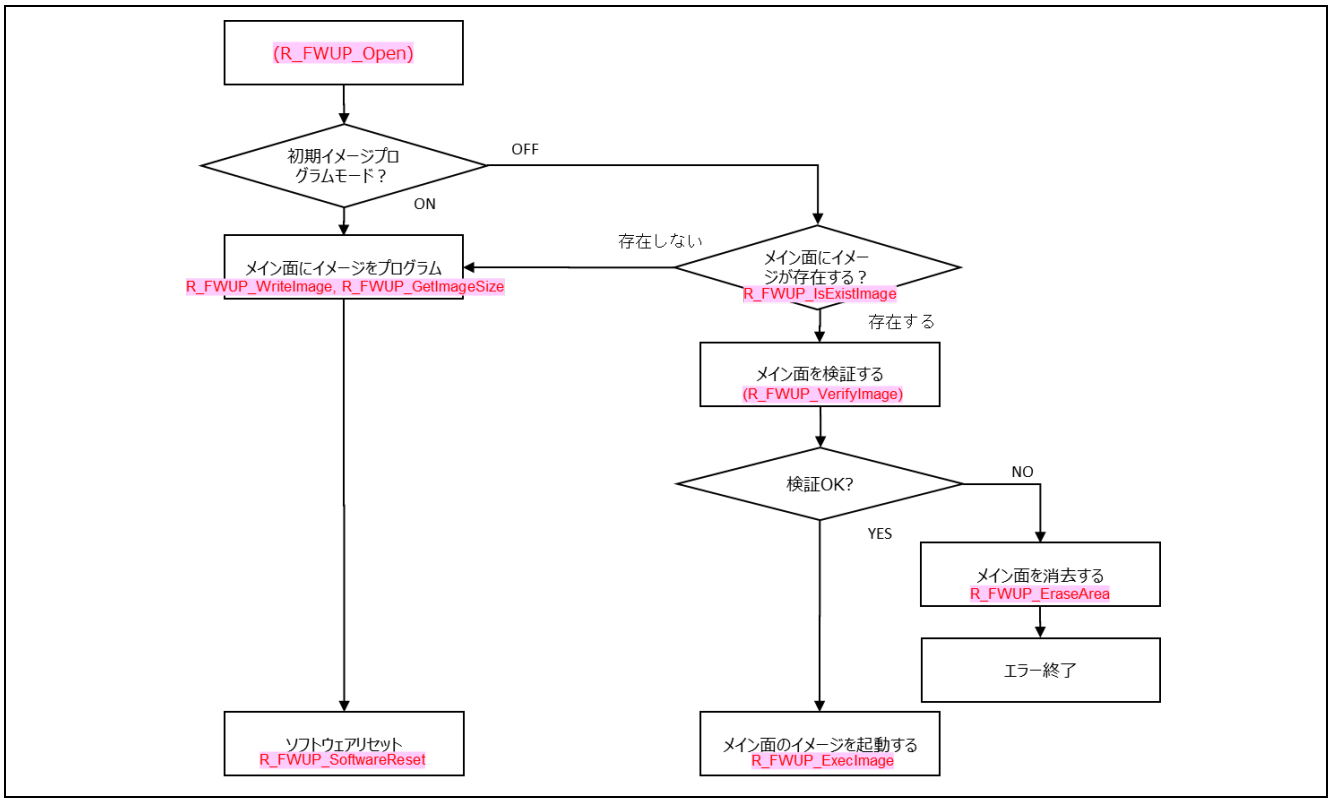


図 2-5 全面更新方式（バッファ無し）のブートローダ実装例

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 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。 ・デュアルバンク（2 バンク）方式 - FWUP_CFG_FWUPV1_COMPATIBLE が無効：バンクスワップ - FWUP_CFG_FWUPV1_COMPATIBLE が有効：何もせず FWUP_SUCCESS を返す ・半面更新方式、全面更新方式（バッファ有り） -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（バッファ無し） -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 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。 本関数では R_RFD_SetExtraBootAreaReq による起動バンク切り換え設定を行い、ソフトウェアリセット後にバンクスワップを実行します。 起動バンク切り換え設定はコード・フラッシュ・プログラミング・モードでなければ使用できないため、直前に R_RFD_SetExtraProgrammingMode を実行してください。 起動バンク切り換え設定の完了後は、エクストラ領域シーケンサの動作終了を確認するために R_RFD_CheckExtraSeqEndStep1 と R_RFD_CheckExtraSeqEndStep2 を実行し、終了の確認後は R_RFD_GetExtraSeqErrorStatus と R_RFD_ClearExtraSeqRegister によるエラー情報の取得と、エクストラ領域シーケンサ制御を行うレジスタのクリアを実行してください。 ソフトウェアリセットを実行する前に、R_RFD_SetExtraNonProgrammingMode を実行してください。	
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	FWUP_CFG_UPDATE_MODE が 0（デュアルバンク（2 バンク）方式）の場合のみ呼び出し可能な関数です。（RL78/L23 のみ）	

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	ー	

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	ー	

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デモプロジェクトです。

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。

デュアルバンク（2 バンク）方式のフォルダ構成：□□¥dualbank(2bank)¥△△¥ の下

半面/全面更新方式のフォルダ構成：□□¥linear¥△△¥ の下

□□：ボード名
△△：コンパイラ（e2_ccrl/iar/llvm）

・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 5.4.0 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 の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 3.11.14 で環境動作確認を行っています。

Python 3.11.14 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ください。

```
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 v3.22.00 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（6.2）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 4.2.3 OpenSSL の実行環境のインストール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。事前に 4.2.2 Python の実行環境のインストールを実施してください。

4.4　デュアルバンク（2 バンク）方式のデモプロジェクト実行手順

この章では FPB-RL78L23 を用いたデュアルバンク（2 バンク）方式のデモプロジェクトについて説明します。

FPB-RL78L23 のデュアルバンク（2 バンク）方式には、boot_loader、fwup_leddemo、fwup_main の 3 つのデモプロジェクトを用意しています。

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、4.8 を参照してください。

4.4.1　実行環境

6.2.4 RL78/L23 の動作確認環境を参照し実行環境を準備します。

4.4.2　デモプロジェクトの構築

以下の手順で、デュアルバンク（2 バンク）方式用の 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ください。

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。

注）各製品のデモプロジェクトは¥modules や¥r_fwup 以下のファイルを相対パスでアクセスします。そのため、インポートする際は、これらのパスが相対的な位置に配置されますようご注意ください。

最も簡単な解決策としましては、「demos」以下を適当なパスに配置し、e2studio 起動時に「demos」をワークスペースとして指定いただくことで相対パスを維持できます。その後、そのワークスペースでデモプロジェクトをインポートするときに「プロジェクトをワークスペースにコピー」のチェックを外した状態でインポートしてください。

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。

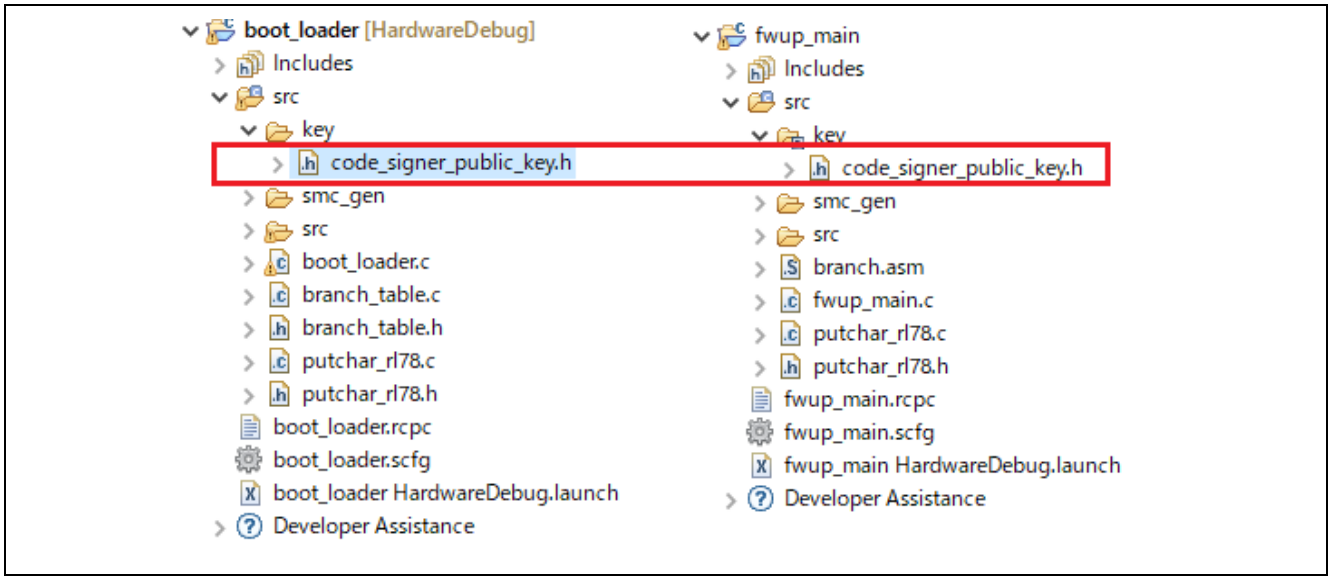


図 4-1 デモプロジェクトの code_signer_public_key.h ファイルの格納場所

```
/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"¥
 * "...base64 data..."¥
 * "-----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、6.2.4.1 のように設定します。

ログ出力に関してはデフォルトでオフに成っているので、boot_loader と fwup_main に関しては、コンフィグ設定の FWUP_CFG_PRINTF_DISABLE を 0 に設定し、fwup_leddemo に関しては、fwup_leddemo.c 内の FWUP_DEMO_PRINTF_DISABLE を 0 に設定してください。

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
（LLVM に関しては、mot ファイルは srec ファイルとなります）

4.4.3　初期イメージと更新イメージを生成

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_L23_ImageGenerator_PRM_dual.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_L23_ImageGenerator_PRM_dual.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_L23_ImageGenerator_PRM_dual.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_L23_ImageGenerator_PRM_dual.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.4.4　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。

注）デュアルバンク（2 バンク）方式のみ Renesas Flash Programmer の設定を変更してください。

・操作設定 > コマンド
　→「フラッシュオプション書き込み」 にチェック
　→「フラッシュオプションベリファイ」 にチェック

・フラッシュオプション > ブートクラスタサイズ/バンクスワップ
　→ 設定オプション： 設定する
　→ BTBLS：07

4.4.5　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 6.2.4 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。

```
==== RL78L23 : BootLoader [dual bank, bank0] ====
verify install area main [sig-sha256-ecdsa]...OK
execute new image ...

==== RL78L23 : Update from User [dual bank] 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ます。

```
W 0x41000, 128 ... OK
W 0x41080, 128 ... OK
...
W 0x43980, 128 ... OK
W 0x43A00, 128 ... OK
```

⑤ 更新ファームウェアのインストールと署名検証が終了すると、バンクスワップ等の処理を経て、更新ファームウェアにジャンプしプログラムが実行されます。

```
verify install area buffer [sig-sha256-ecdsa]...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。

```
==== RL78L23 : BootLoader [dual bank, bank1] ====
verify install area main [sig-sha256-ecdsa]...OK
execute new image ...

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

※デモプロジェクトでは、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、バンクスワップ後のブートローダによるメイン面の検証終了後に、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　半面更新方式のデモプロジェクト実行手順

この章では FPB-RL78G23-128p を用いた半面更新方式のデモプロジェクトについて説明します。

FPB-RL78G24 および FPB-RL78L23 を用いたデモプロジェクトも FPB-RL78G23-128p と同様のため、この章を参照してください。

FPB-RL78G23-128p には、boot_loader、fwup_lddemo、fwup_main の 3 つのデモプロジェクトを用意しています。

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、半面更新方式と全面更新方式（バッファ有り）、全面更新方式（バッファ無し）の 3 つのファームウェアアップデート方式に対応します。

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。デバッガ接続によるアプリケーションのデバッグの実施方法については、4.8 を参照してください。

4.5.1　実行環境

6.2.1 RL78/G23 の動作確認環境を参照し実行環境を準備します。

4.5.2　デモプロジェクトの構築

以下の手順で、半面更新方式用の 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ください。

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。

注）各製品のデモプロジェクトは¥modules や¥r_fwup 以下のファイルを相対パスでアクセスします。そのため、インポートする際は、これらのパスが相対的な位置に配置されますようご注意ください。

最も簡単な解決策としましては、「demos」以下を適当なパスに配置し、e2studio 起動時に「demos」をワークスペースとして指定いただくことで相対パスを維持できます。その後、そのワークスペースでデモプロジェクトをインポートするときに「プロジェクトをワークスペースにコピー」のチェックを外した状態でインポートしてください。

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。

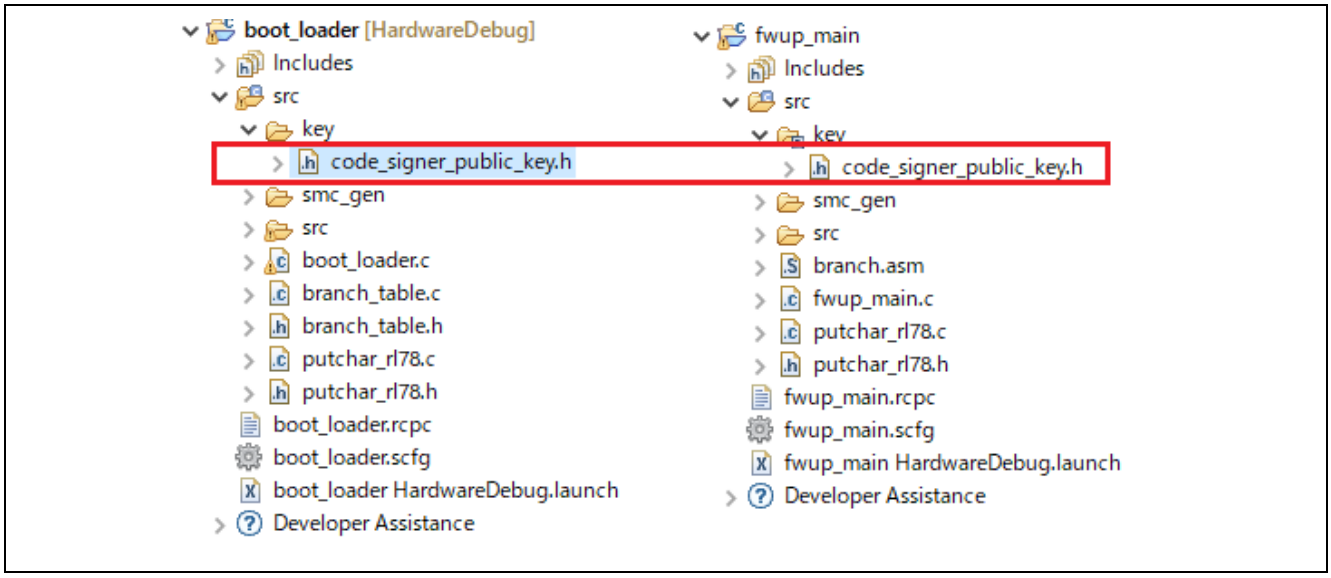


図 4-2 デモプロジェクトの code_signer_public_key.h ファイルの格納場所

```
/*
 * 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、6.2.1.1 のように設定します。
デモプロジェクトでは、ログ出力はデフォルトでオフに設定されています。
ログ出力を有効にするために、boot_loader と fwup_main に関しては、コンフィグレーション設定の FWUP_CFG_PRINTF_DISABLE を 0 に、fwup_leddemo に関しては、fwup_leddemo.c 内の FWUP_DEMO_PRINTF_DISABLE を 0 にしてください。

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
（LLVM に関しては、mot ファイルは srec ファイルとなります）

4.5.3　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
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
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.5.4　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。

4.5.5　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 6.2.1 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.
```

4.6　全面更新方式（バッファ無し）のデモプロジェクト実行手順

この章では FPB-RL78G22 を用いた全面更新方式（バッファ無し）のデモプロジェクトについて説明します。

FPB-RL78G23-128p および FPB-RL78G24、FPB-RL78L23 を用いたデモプロジェクトも FPB-RL78G22 と同様のため、この章を参照してください。

FPB-RL78G22 には、boot_loader、fwup_leddemo の 2 つのデモプロジェクトを用意しています。

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。デバッガ接続によるアプリケーションのデバッグの実施方法については、4.8 を参照してください。

4.6.1　実行環境

6.2.3 RL78/G22 の動作確認環境を参照し実行環境を準備します。

4.6.2　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ください。

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。

注）各製品のデモプロジェクトは¥modules や¥r_fwup 以下のファイルを相対パスでアクセスします。そのため、インポートする際は、これらのパスが相対的な位置に配置されますようご注意ください。

最も簡単な解決策としましては、「demos」以下を適当なパスに配置し、e2studio 起動時に「demos」をワークスペースとして指定いただくことで相対パスを維持できます。その後、そのワークスペースでデモプロジェクトをインポートするときに「プロジェクトをワークスペースにコピー」のチェックを外した状態でインポートしてください。

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、6.2.3.1 のように設定します
デモプロジェクトでは、ログ出力はデフォルトでオフに設定されています。
ログ出力を有効にするために、boot_loader に関しては、コンフィグレーション設定の FWUP_CFG_PRINTF_DISABLE を 0 に、fwup_leddemo に関しては、fwup_leddemo.c 内の FWUP_DEMO_PRINTF_DISABLE を 0 にしてください。

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。

（LLVM に関しては、mot ファイルは srec ファイルとなります）

4.6.3　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_full.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_full.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_full.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_full.csv
boot_loader.mot
fwup_leddemo_011.mot
fwup_leddemo_012.mot
fwup_leddemo_012.rsu
initial_firm.mot
```

4.6.4　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。

4.6.5　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 6.2.3 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 し初期イメージプログラムモードを ON にします。

```
==== 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.
```

4.7　全面更新方式（バッファ有り）のデモプロジェクト実行手順

この章では FPB-RL78G23-128p を用いた全面更新方式（バッファ有り）のデモプロジェクトについて説明します。

FPB-RL78G24 および FPB-RL78L23 を用いたデモプロジェクトも FPB-RL78G23-128p と同様のため、この章を参照してください。

FPB-RL78G23-128p には、boot_loader、fwup_lddemo、fwup_main の 3 つのデモプロジェクトを用意しています。

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、半面更新方式と全面更新方式（バッファ有り）、全面更新方式（バッファ無し）の 3 つのファームウェアアップデート方式に対応します。

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。デバッガ接続によるアプリケーションのデバッグの実施方法については、4.8 を参照してください。

4.7.1　実行環境

6.2.1 RL78/G23 の動作確認環境を参照し実行環境を準備します。

4.7.2　デモプロジェクトの構築

以下の手順で、全面更新方式（バッファ有り）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ください。

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。

注）各製品のデモプロジェクトは¥modules や¥r_fwup 以下のファイルを相対パスでアクセスします。そのため、インポートする際は、これらのパスが相対的な位置に配置されますようご注意ください。

最も簡単な解決策としましては、「demos」以下を適当なパスに配置し、e2studio 起動時に「demos」をワークスペースとして指定いただくことで相対パスを維持できます。その後、そのワークスペースでデモプロジェクトをインポートするときに「プロジェクトをワークスペースにコピー」のチェックを外した状態でインポートしてください。

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。

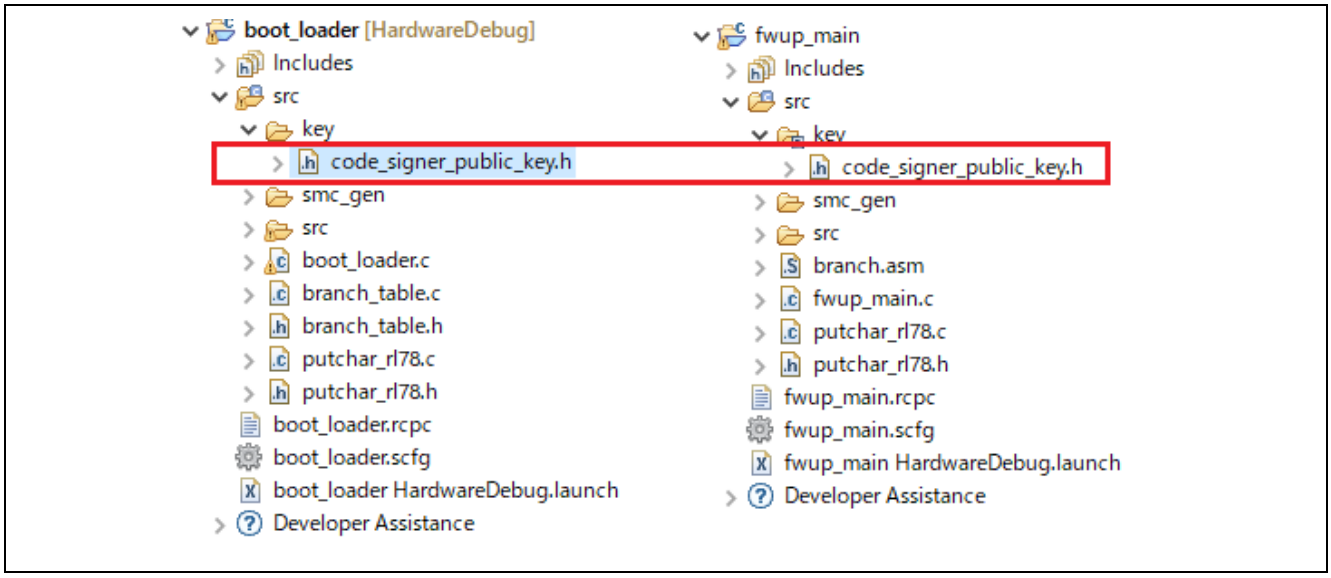


図 4-3 デモプロジェクトの code_signer_public_key.h ファイルの格納場所

```
/*
 * 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、6.2.1.2 のように設定します。
デモプロジェクトでは、ログ出力はデフォルトでオフに設定されています。
ログ出力を有効にするために、boot_loader と fwup_main に関しては、コンフィグレーション設定の FWUP_CFG_PRINTF_DISABLE を 0 に、fwup_leddemo に関しては、fwup_leddemo.c 内の FWUP_DEMO_PRINTF_DISABLE を 0 にしてください。

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
（LLVM に関しては、mot ファイルは srec ファイルとなります）

4.7.3　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_full.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_full.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_full.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_full.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.7.4　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。

4.7.5　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 6.2.1 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.
```

4.8　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）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) 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.2 デモプロジェクトの構築」を参照してください。

(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.3 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。

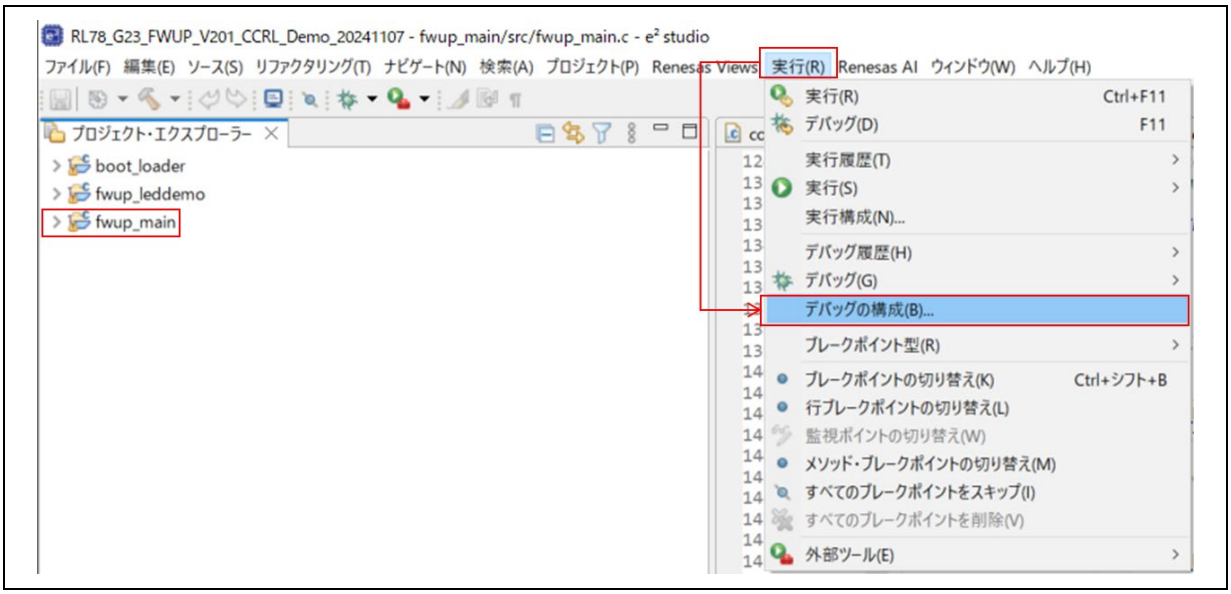
(3) フラッシュメモリを消去します。（デュアルバンク（2 バンク）方式のみ）

フラッシュライタにてチップイレースを行い、フラッシュメモリを消去します。
デュアルバンク（2 バンク）方式以外の場合は、この工程は不要です。

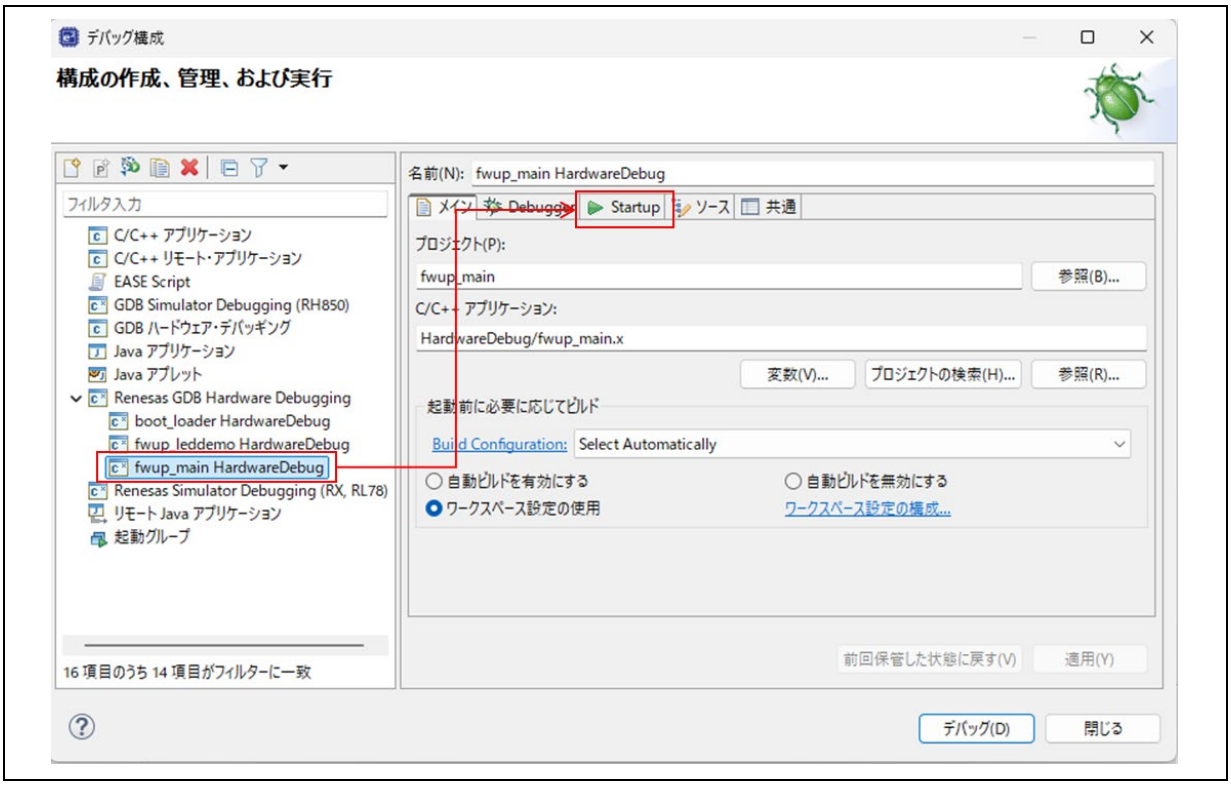
(4) アプリケーションプログラム（fwup_main）のデバッグ設定を行います。

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。

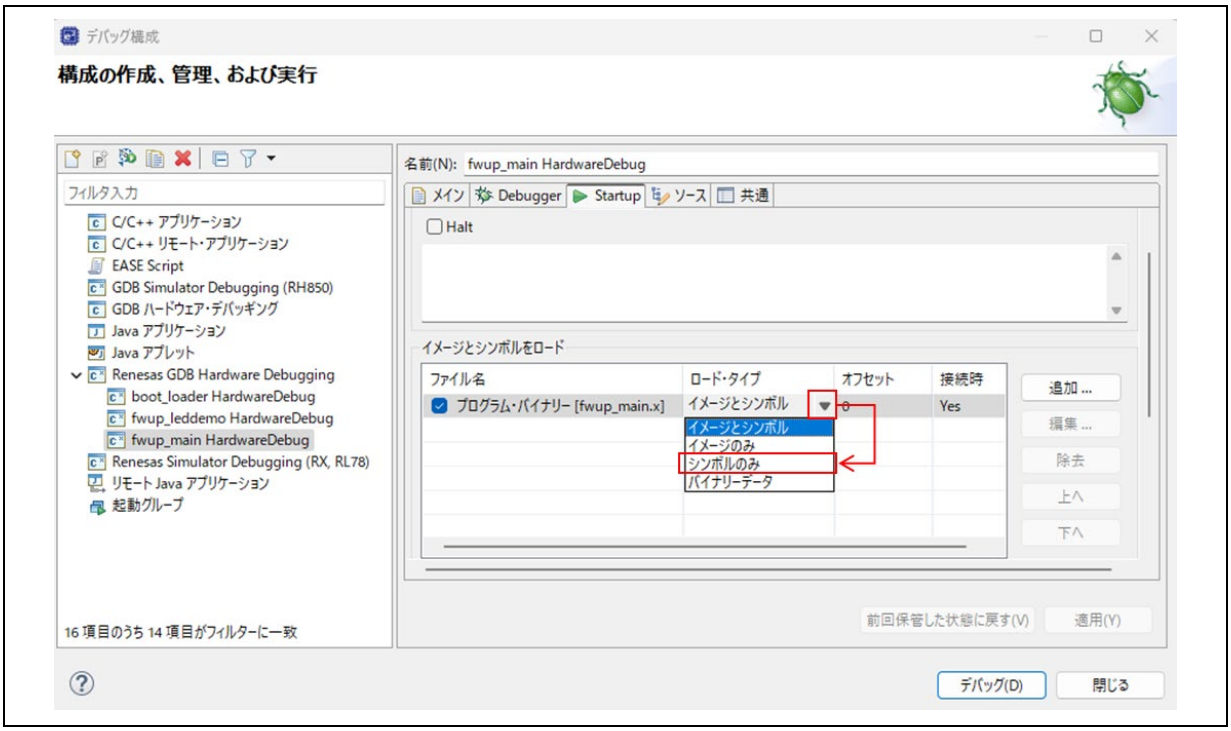
① 「実行(R)」→「デバッグの構成(B)」を開き、fwup_main_HardwareDebug を選択します。



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



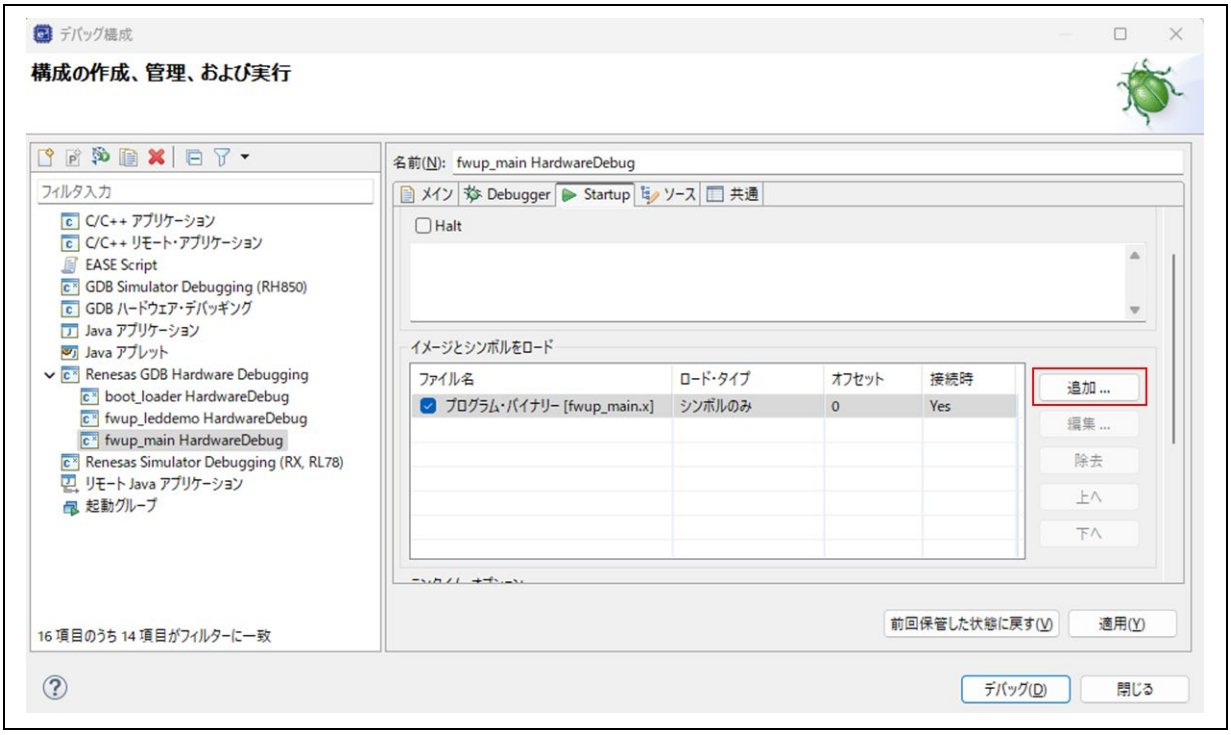
③ プログラム・バイナリ[fwup_main.x]のロード・タイプを「イメージとシンボル」から「シンボルのみ」に変更します。



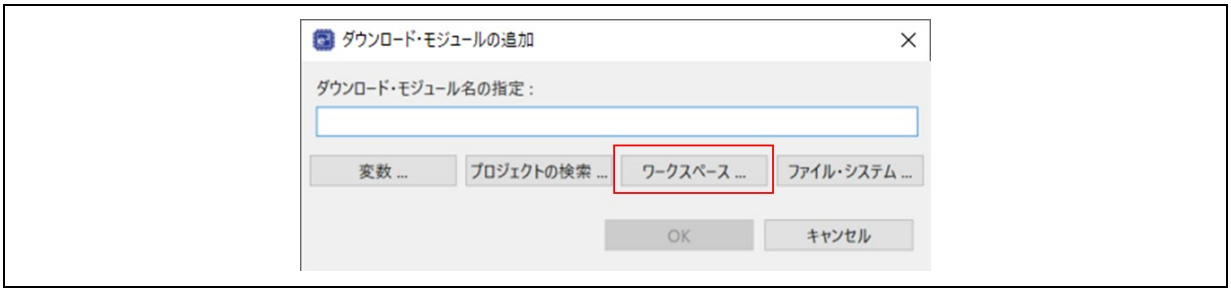
(5) ブートローダ（boot_loader）のシンボルを追加します。

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。

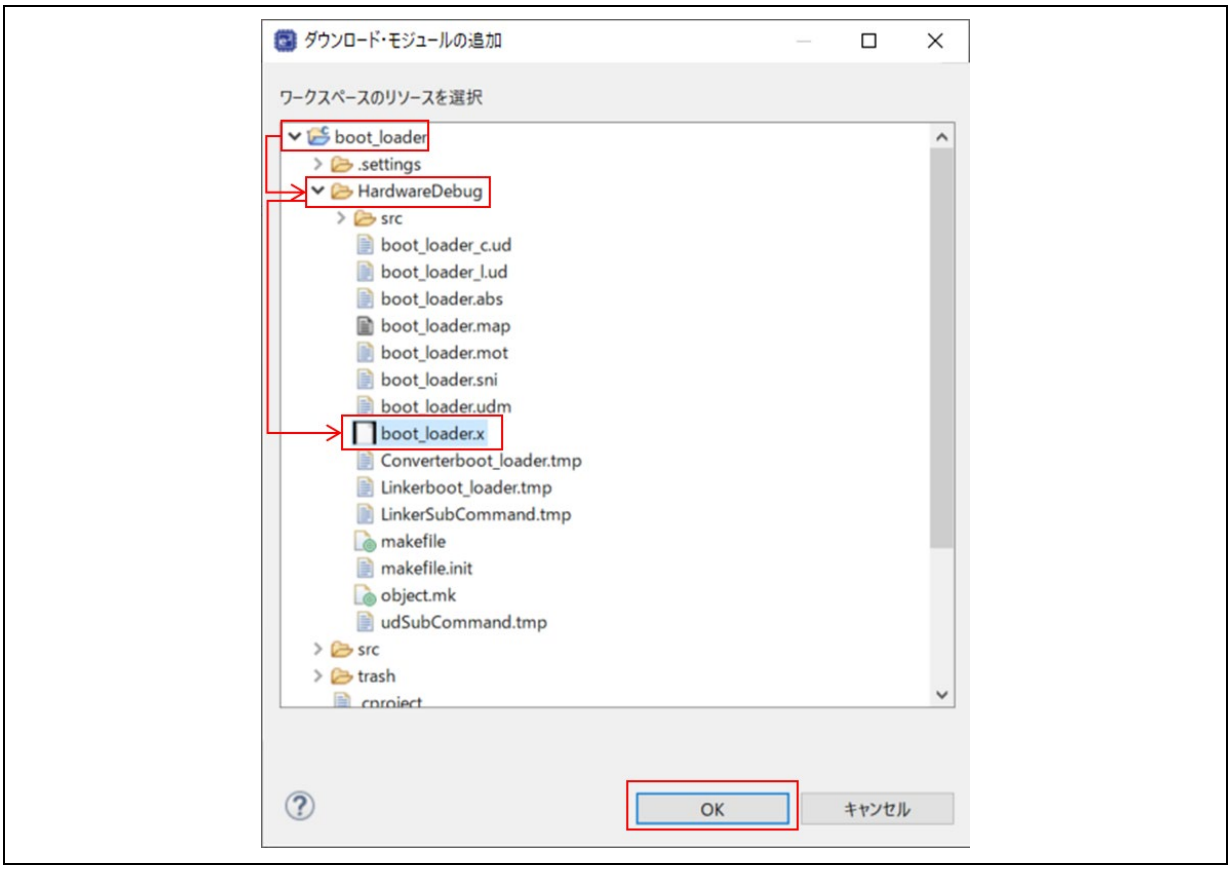
① 「追加」をクリックします。



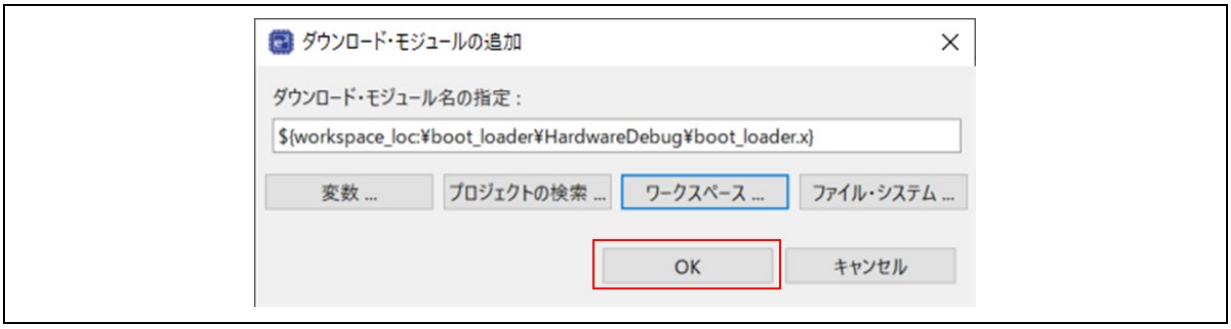
② 「ワークスペース」をクリックします。



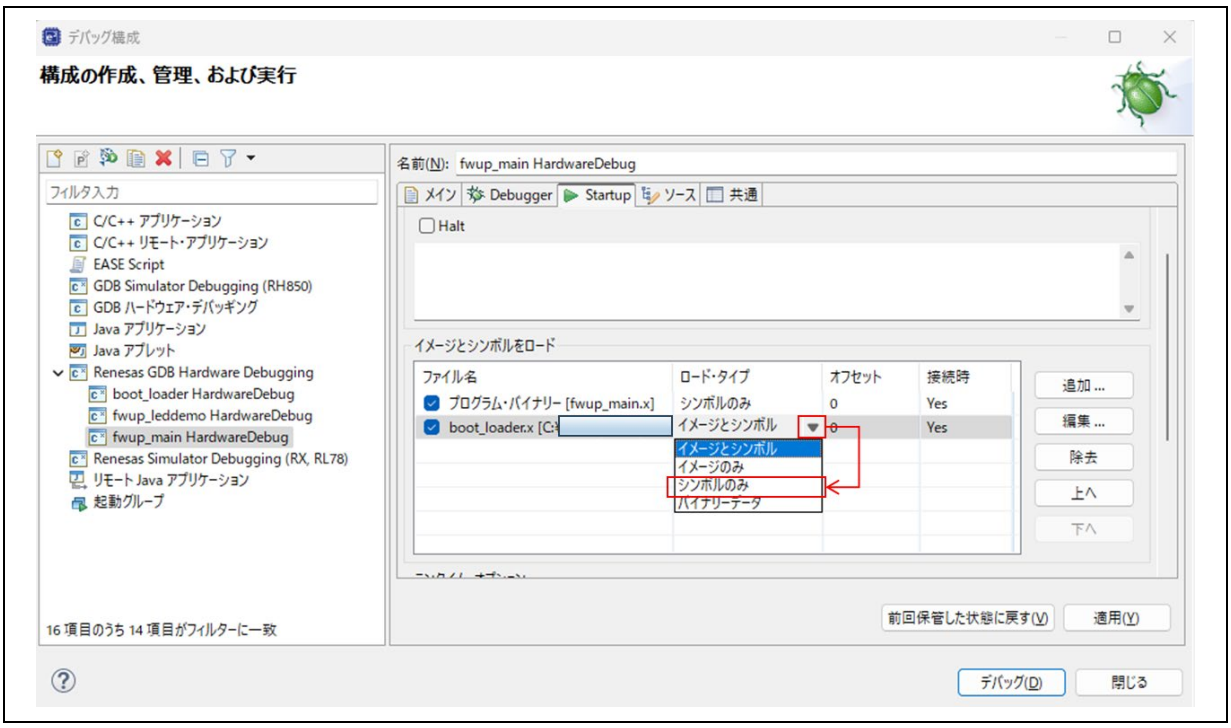
③ ブートローダ（boot_loader.x）を選択して「OK」をクリックします。



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。



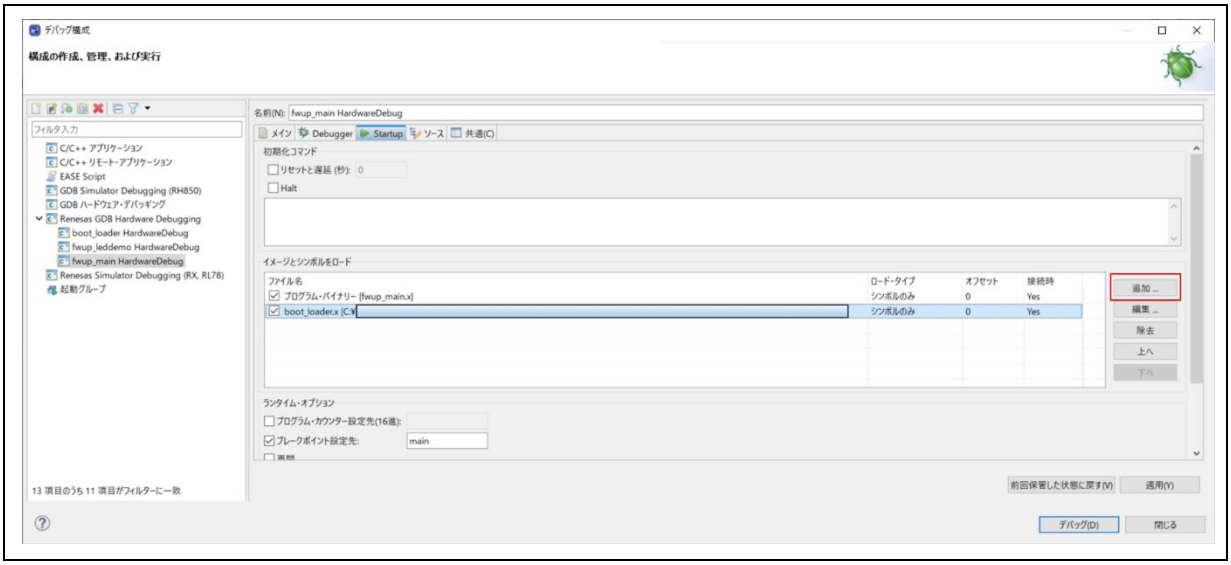
⑤ ブートローダ（boot_loader.x）のロード・タイプを「イメージとシンボル」から「シンボルのみ」に変更します。



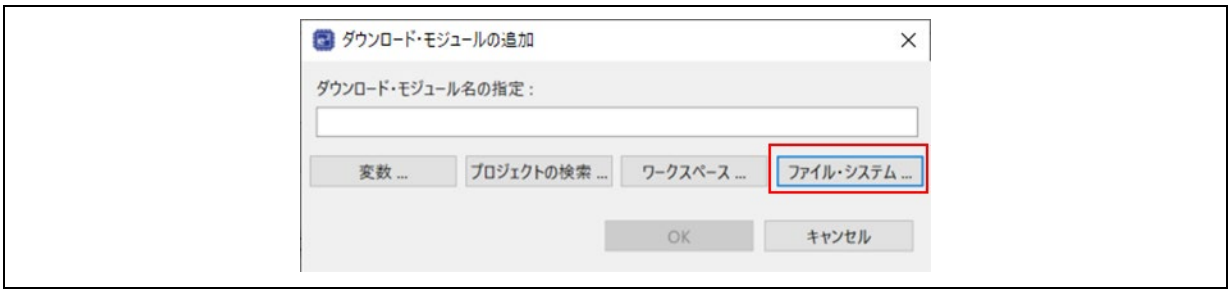
(6) 初期イメージ（initial_firm.mot）のイメージを追加します。

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。

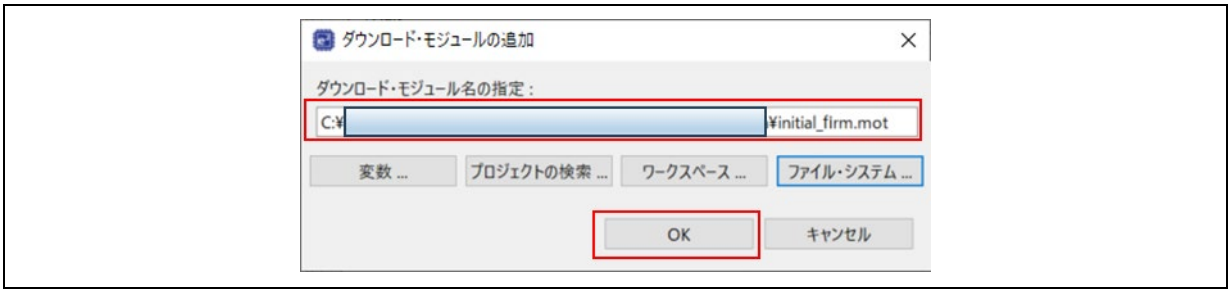
① 「追加」をクリックします。



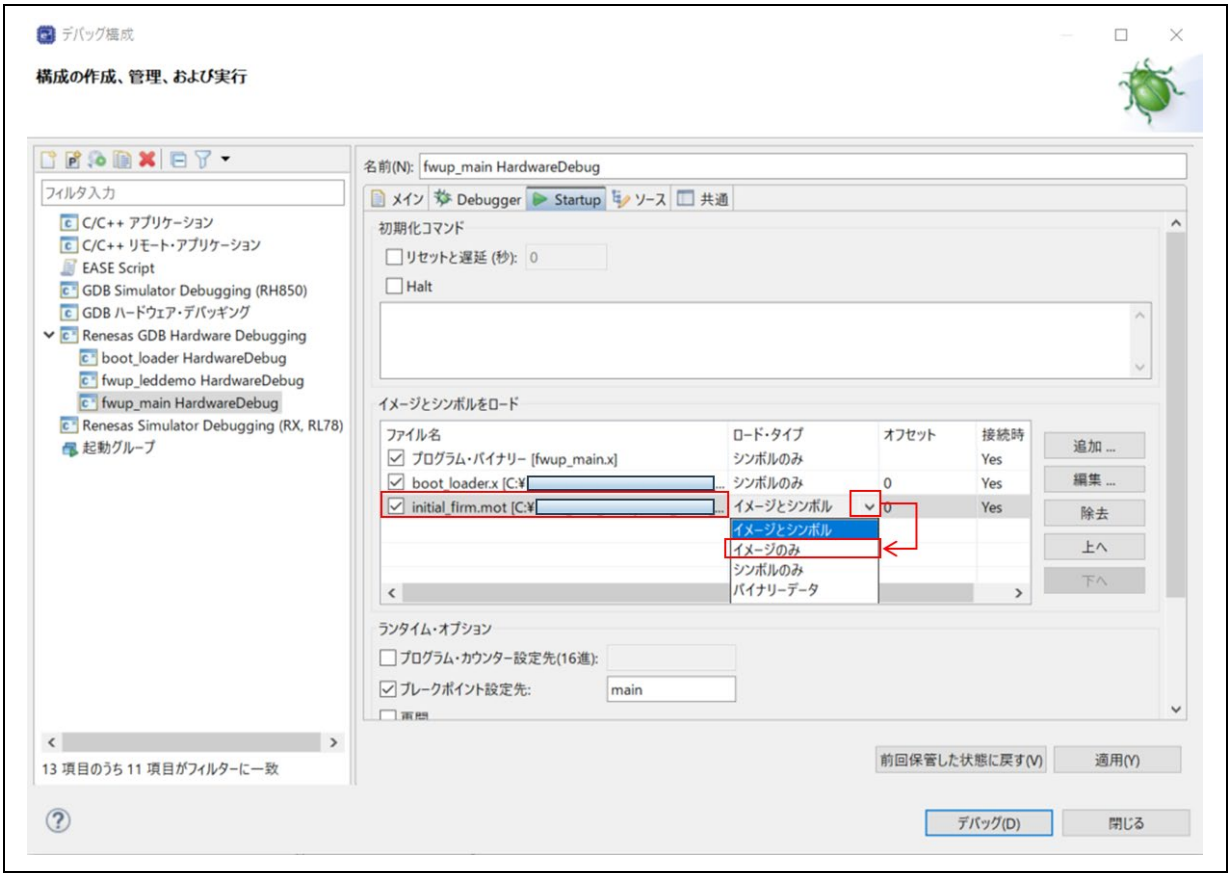
② 「ファイル・システム」をクリックします。



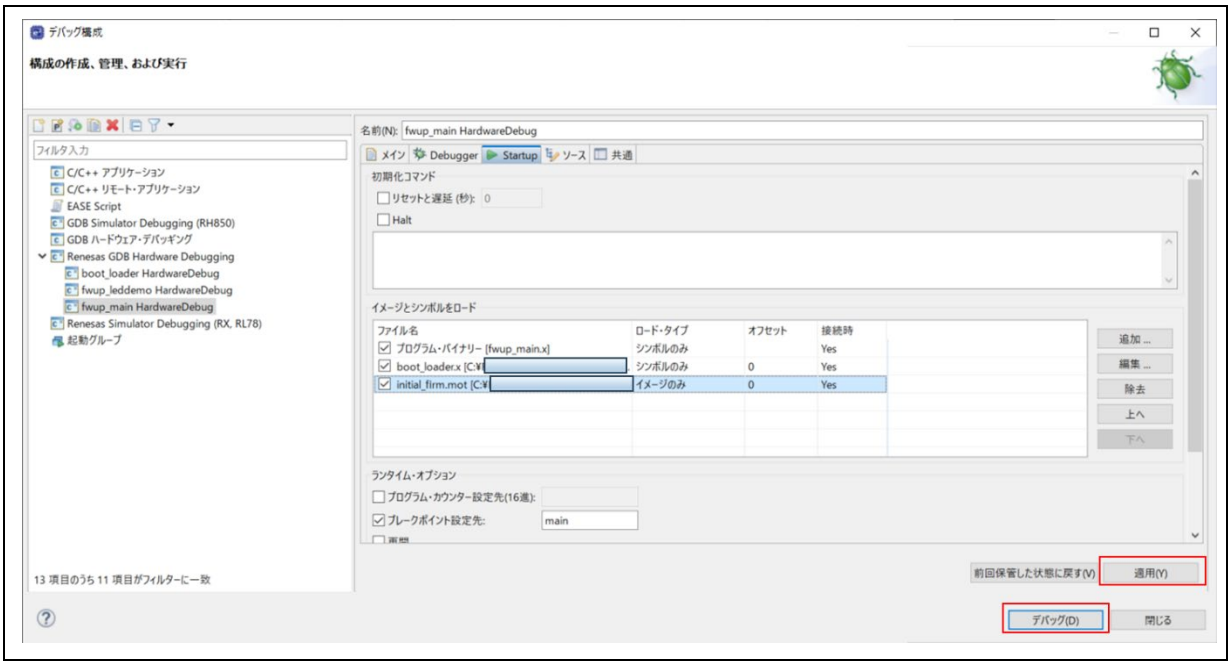
③ 初期イメージ（initial_firm.mot）を選択して「OK」をクリックします。



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。

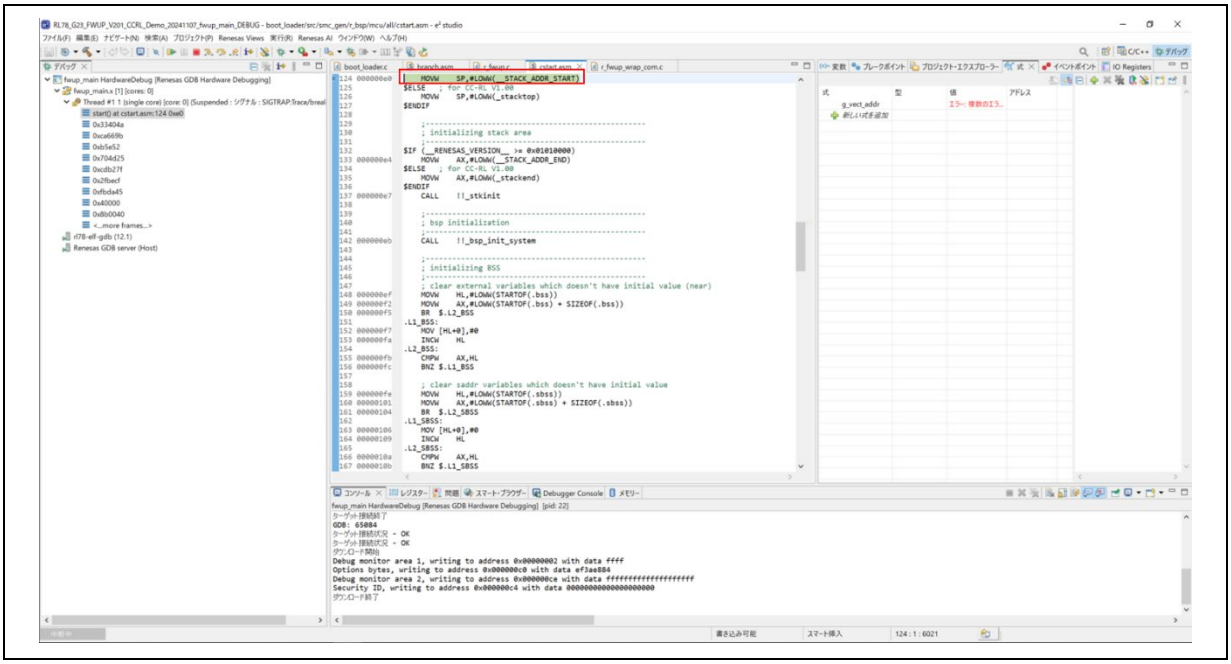


⑤ 「適用(Y)」をクリックし、「デバッグ(D)」をクリックします。



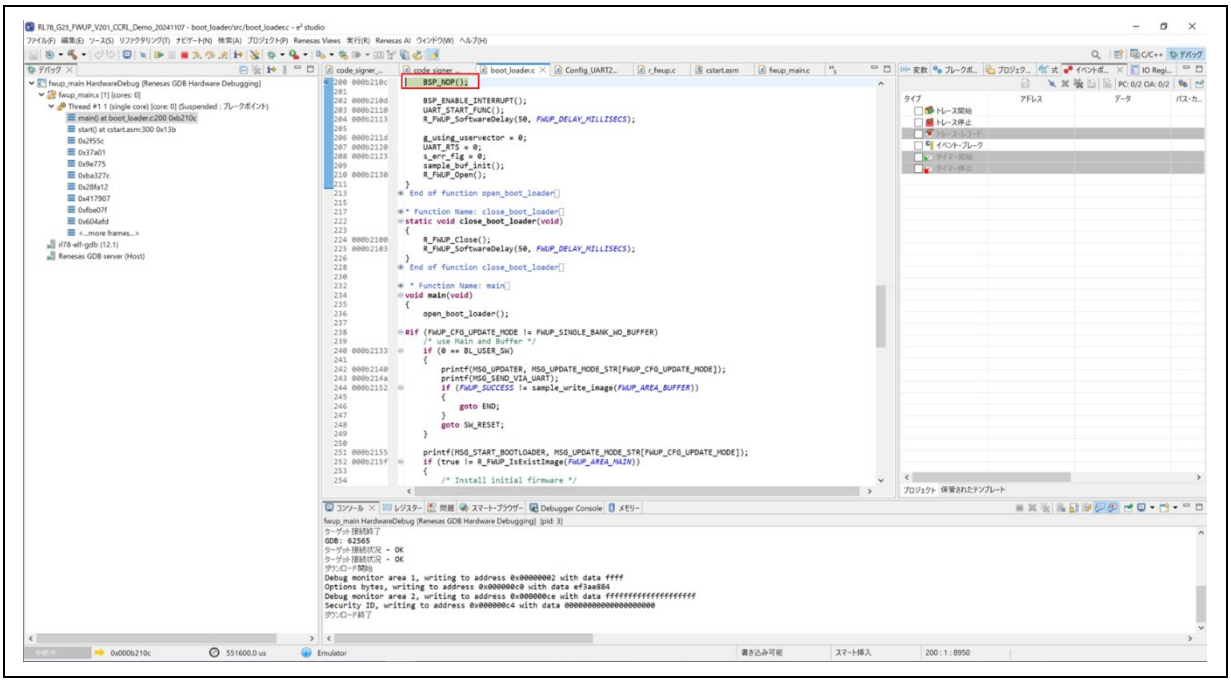
(7) デバッガを起動します。

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。



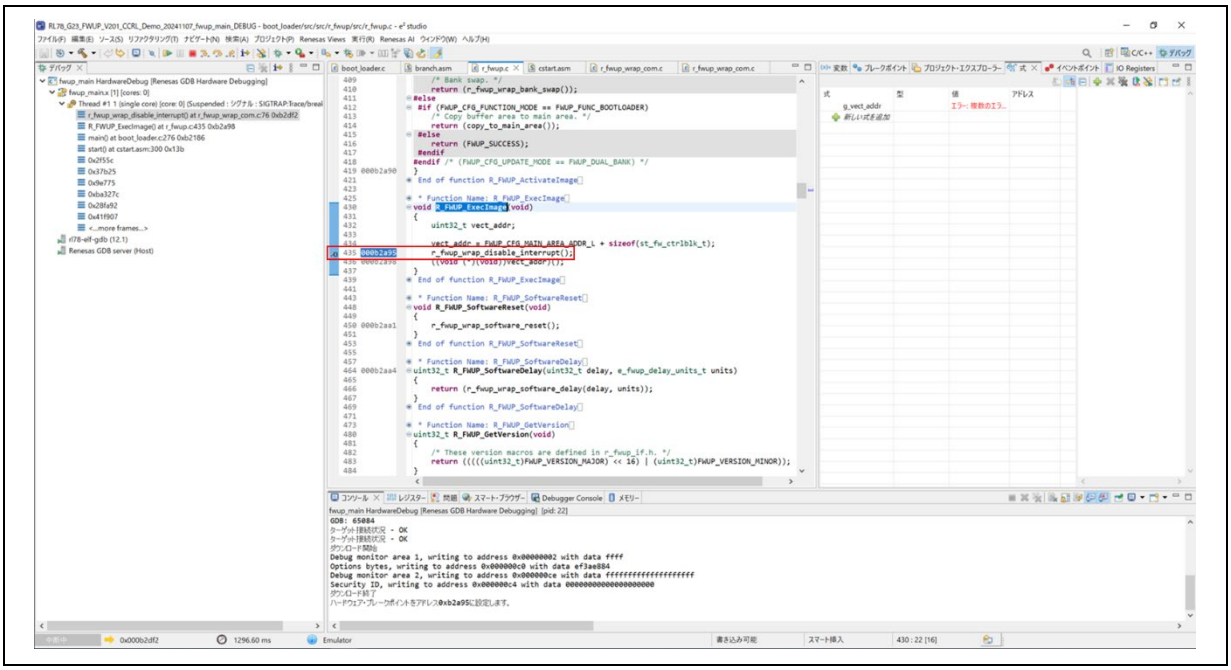
(8) プログラムを再開します。

「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。



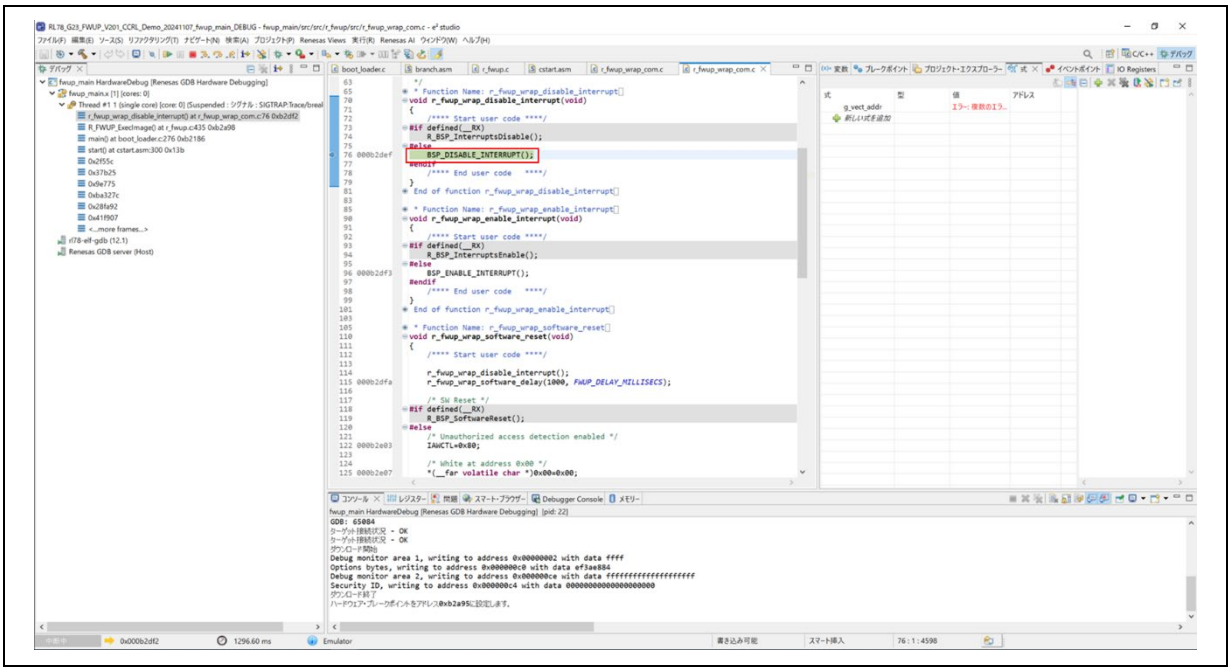
(9) fwup_main プロジェクトの main()にブレイクポイントを設定します。

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。



(10) 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。



4.9　デモプロジェクトの更新イメージ取得経路の変更方法

本デモプロジェクトでは、更新イメージを PC からシリアル通信経由で取得していますが、このイメージの取得経路を変更（SD カード、USB メモリ等）する場合について記載します。

図 4-5 にデモプロジェクト（fwup_main.c の main 関数）のフローチャートを示します。まずは、デモプロジェクトを参考に更新イメージの取得経路の変更を検討ください。

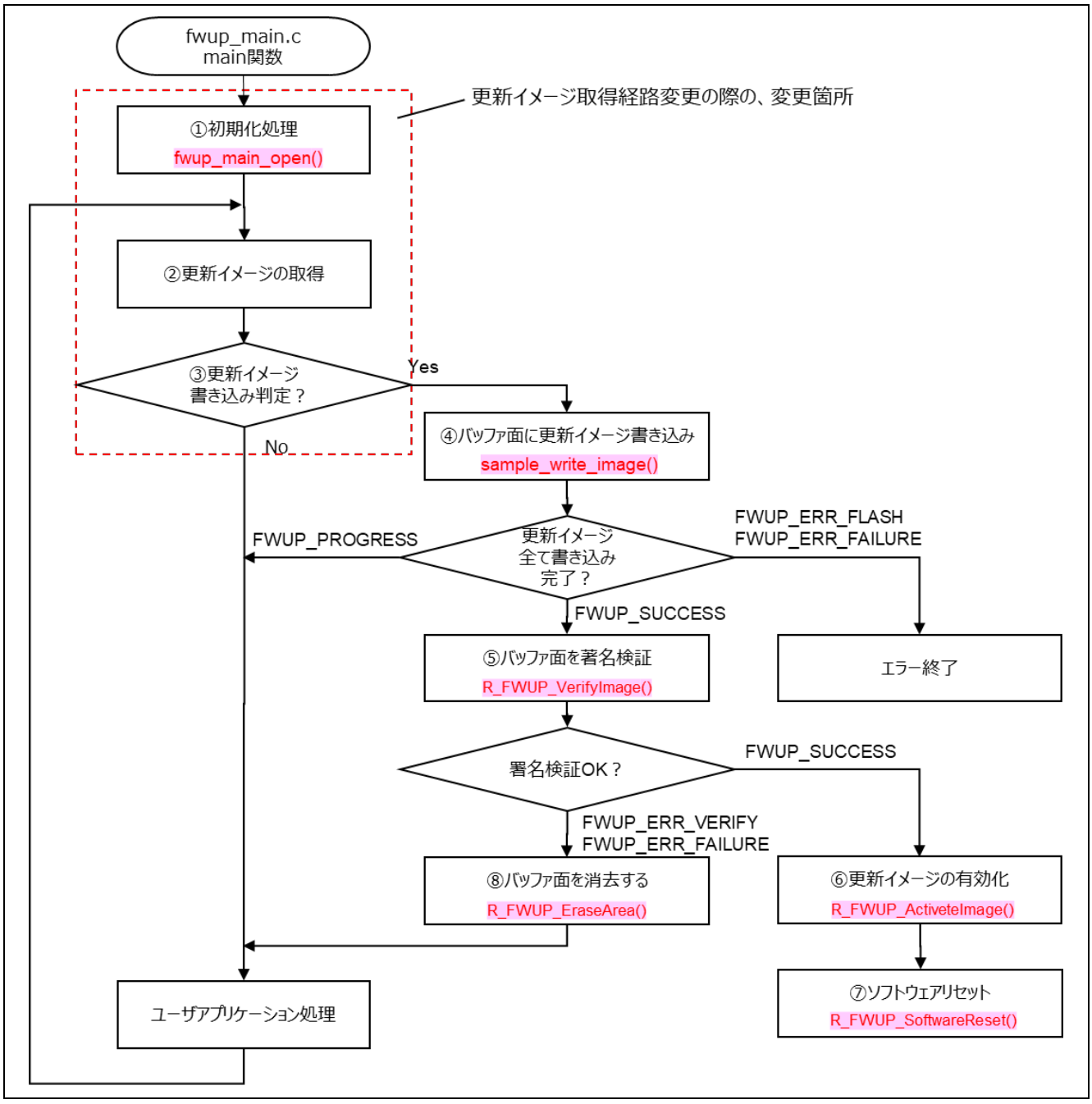


図 4-4 デモプロジェクトの fwup_main の処理フロー

注：全面更新方式（バッファ無し）に関しては、更新イメージの取得をブートローダで実施しています。そのため、この方式でご検討の場合は、説明内容をブートローダに置き換えて検討ください。

更新イメージ取得経路を変更する際の変更箇所としては、①②③の部分となります。④以降に関してはアップデートの動作に成りますので、基本的には変更する必要は有りません。

① 初期化処理に関しては、ファームウェアアップデートモジュールの初期化のみならず、更新イメージを取得する際に使用する各種モジュールの初期化等を行います。

デモプロジェクトでは、シリアル通信モジュール、ファームウェアアップデートモジュールの初期化（Open 処理）および変数の初期化を行っています。

② 外部から取得した更新イメージは、RAM 等にバッファリングしてください。
外部からの更新イメージの取得に関しては、多様な経路（SD カード、USB メモリ等）が考えられますが、イメージ取得に関しては Renesas から提供されているアプリケーションノート等を参考にお客様にて構築をお願いします。

デモプロジェクトでは、シリアル通信モジュールを使用し、通信完了割り込みにより発生するコールバック（sci_callback）にて更新イメージを取得しています。更新イメージのバッファリングが完了すると RTS 端子（FWUP_UART_RTS）をセットします。これにより通信相手の PC からの通信を中断させ、main 処理での書き込み開始の判定通知にもしています。

③ 更新イメージの書き込み判定に関しては、更新イメージのバッファリングが完了しましたら、更新イメージの書き込み処理を行ってください。
更新イメージは一度に全てを揃える必要はありません。それぞれのシステムで都合の良いサイズで更新イメージを取得しバッファリング頂き、順次書き込み処理を行ってください。ただし、それぞれの製品毎にフラッシュメモリへの書き込み単位が有りますので、その書き込み単位の倍数での処理をお願いします。（例：RL78/G23 の場合、コードフラッシュへの書き込み単位は 4 バイトなので、その倍数である 64, 128, 256 バイト等を設定）
更新イメージを書き込む際、フラッシュメモリの処理中は割り込み禁止となります。そのため、割り込み処理にて更新イメージの取得を行う場合は、書き込み中に取りこぼしの可能性があるため、デモプロジェクトで実施しているシリアル通信の RTS 制御のような、更新イメージの取得を中断するような処理をご検討ください。

デモプロジェクトでは、RTS 端子（FWUP_UART_RTS）の状態を確認し、書き込みを開始しています。

その他、デモプロジェクトを理解頂く上での注意点を以下にまとめます。

＜fwup_main.c に関する注意点＞

- デモプロジェクトの更新イメージの取得にはシリアル通信を使用していますが、シリアル通信の特性上、RTS 制御による通信相手からの通信の中断にディレイが発生する可能性があるため、書き込み前にディレイ処理を設けて、更新イメージデータの取りこぼしを対策しています。
- ④更新イメージの書き込みに関しては、更新イメージの先頭から順番に書き込みを行ってください。
通信方式等によりデータの順番が入れ替わる様な場合は、順番を整列した後行ってください。
更新イメージの書き込みはバッファサイズにもよりますが、一度の書き込みでは完了しませんので、複数回に分けて書き込む必要が有ります。その際、更新イメージの全サイズ分の書き込みが完了するまでは、継続状態（FWUP__PROGRESS）の戻り値が返りますので、続けて更新イメージの取得および書き込み処理を繰り返してください。
更新イメージの書き込みが全て完了すると書き込み成功（FWUP_SUCCESS）の戻り値が返ります。
- ④更新イメージの書き込みで、フラッシュエラー（FWUP_ERR_FLASH）の戻り値が返った場合、フラッシュメモリにエラーが発生している可能性が高いため、バッファ面を消去してから一連の処理をやり直してください。
- 更新イメージのサイズ取得は R_FWUP_GetImageSize 関数にて行います。サイズ情報は更新イメージの先頭にある RSU ヘッダ部分にあるため、RSU ヘッダ部分の書き込みが完了するまで正しく取得できません。正しく取得できるまではこの API の戻り値は 0 が返ります。

- ⑥更新イメージの有効化処理に関しては、アップデート方式により処理内容が異なります。
  デュアルバンク（2 バンク）方式：バンクスワップ処理を行います。
  デュアルバンク以外の方式：特に何も行いません。
  そのため、デュアルバンク以外の方式の場合は更新イメージの有効化処理は不要です。デュアルバンク方式以外のアップデートの場合、バッファ面からメイン面へのコピー等の処理はブートローダにて行います。
- ⑦ソフトウェアリセットに関して、デモプロジェクトでは書き込み完了直後に実施していますが、それぞれのシステムで都合の良いタイミングで実施頂いて問題ありません。
- ⑧バッファ面の消去に関しては、バッファ面の署名検証に失敗した場合に更新イメージを消去します。消去後の動作に関しては、お客様のシステムに依存します。
  フィールドにて署名検証に失敗する状況は、不正な更新イメージでアップデートされている可能も考えられますので、リトライせずにアップデート動作を中止することをお勧めします。

＜bootloader.c に関する注意点＞

- ブートローダに関しては、汎用的に使用できるよう作成しておりますので、基本的にはそのままご使用いただけます。
- 全面更新方式（バッファ無し）では、更新イメージの取得をブートローダで実施しています。そのため、この方式でご検討されている場合はブートローダ側を変更してください。
  （ブートローダの更新イメージ取得からアップデートの処理は、fwup_main と同様の処理の流れに成っています）
- デュアルバンク（2 バンク）方式のデモプロジェクトでは、アップデート後に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残したくない場合は、ブートローダでのメイン面の署名検証成功後にバッファ面の消去を追加してください。
  デュアルバンク以外の方式では、ブートローダでのアップデート処理（バッファ面からメイン面のコピー）後にバッファ面の消去を行っています。
- ブートローダを変更して使用する場合は、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も参照してください。

4.10 デモプロジェクトへの割り込み処理追加方法

デモプロジェクトの割り込み処理の流れ、およびユーザ定義割り込み処理を追加する方法について記載します。

4.10.1 デモプロジェクトの割り込み処理の流れ

RL78 では、ベクタテーブルのアドレス（0x00～0x7F）が固定されているため、ブートローダとユーザアプリケーションで共通のベクタテーブルを共有する必要があります。このため、割り込み発生時に、ブートローダまたはユーザアプリケーションのどちらの割り込みかを判別して振り分ける必要があり、それらの処理をブートローダにて行っています。

ここでは、RL78/G23 のデモプロジェクトを例に、割り込み処理の流れを説明します。

① 割り込み発生
② ベクタテーブルに登録されている割り込みハンドラへジャンプ
③ 割り込みハンドラで、実行モード（RAM 上の Flag）を判定し、
- Flag＝0：ブートローダの割り込みコールバックを実行
- Flag＝1：ユーザアプリケーションの割り込みコールバックを実行

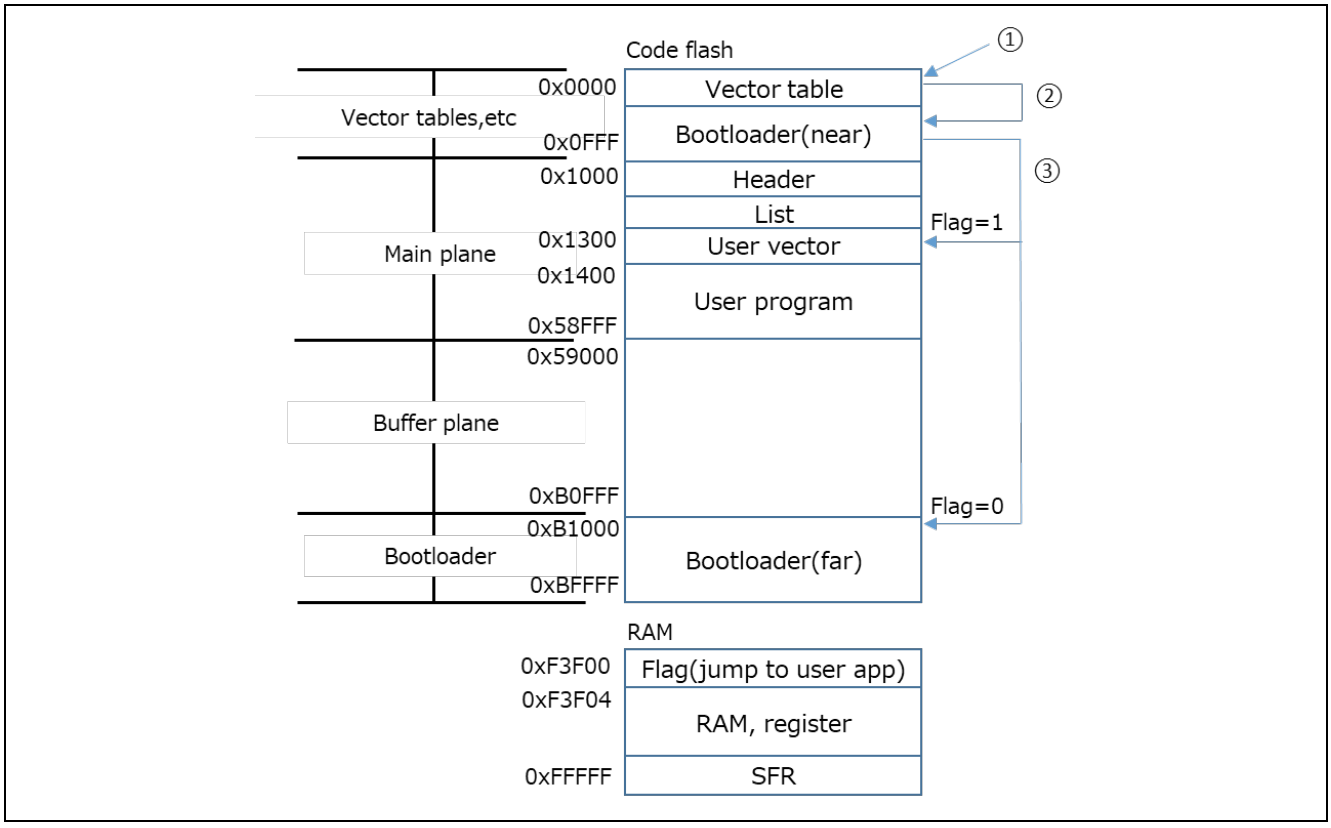


図 4-5 割り込み処理の流れ

4.10.2 デモプロジェクトへの割り込み処理追加手順

本章では、デモプロジェクト（fwup_main.c）に割り込み処理を追加した場合の手順を説明します。

① コード生成機能を使用して、割り込みを利用するコンポーネントを追加します。

② 追加されたソースコードのハードウェア割り込みハンドラの定義部分を削除します。

UART2 を追加した場合は、Config_UART2_user.c のハードウェア割り込みハンドラの定義部分を削除します。

fwup_main¥src¥Config_UART2¥Config_UART2_user.c

```
/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "Config_UART2.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
Pragma directive
***********************************************************************/
#pragma interrupt r_Config_UART2_interrupt_send(vect=INTST2)
#pragma interrupt r_Config_UART2_interrupt_receive(vect=INTSR2)
#pragma interrupt r_Config_UART2_interrupt_error(vect=INTSRE2)
/* Start user code for pragma. Do not edit comment generated here */
```

削除

③　グローバル定義の割り込みハンドラ関数を作成し、既存の関数と置き換えます。

UART2 の場合は、Config_UART2_user.c の下記の関数を置き換えます。

- static void __near r_Config_UART2_interrupt_send(void)　⇒　void r_isr_txi(void)
- static void __near r_Config_UART2_interrupt_receive(void)　⇒　void r_isr_rxi(void)
- static void __near r_Config_UART2_interrupt_error(void)　⇒　void r_isr_eri(void)

fwup_main¥src¥Config_UART2¥Config_UART2_user.c

```
static void r_Config_UART2_callback_error(uint8_t err_type)
{
    /* Start user code for r_Config_UART2_callback_error. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}
static void __near r_Config_UART2_interrupt_send(void)
  ・・・
static void __near r_Config_UART2_interrupt_receive(void)
  ・・・
static void __near r_Config_UART2_interrupt_error(void)
  ・・・
```

削除

```
/* Start user code for adding. Do not edit comment generated here */
void r_isr_txi(void)
{
    if (g_uart2_tx_count > 0U)
    {
        TXD2 = *gp_uart2_tx_address;
        gp_uart2_tx_address++;
        g_uart2_tx_count--;
    }
    else
    {
        r_Config_UART2_callback_sendend();
    }
}

void r_isr_rxi(void)
{
    uint8_t rx_data;
    rx_data = RXD2;
    sample_buffering(rx_data);
}

void r_isr_eri(void)
{
    uint8_t err_type;

    *gp_uart2_rx_address = RXD2;
    err_type = (uint8_t)(SSR11 & 0x0007U);
    SIR11 = (uint16_t)err_type;
    r_Config_UART2_callback_error(err_type);
}
/* End user code. Do not edit comment generated here */
```

追加

④　置き換えた割り込みハンドラ関数を、branch.asm に登録します。

UART2 の場合は、ブランチテーブルの 0x14～0x18 に③で置き換えた関数を登録します。

```
fwup_main¥src¥ branch.asm
usr_vect   .cseg at 0x00001300
          br      !!_start       ;reset
          .db4    0xffffffff
          .db4    0xffffffff     ;vect=0x04
          .db4    0xffffffff     ;vect=0x06
          .db4    0xffffffff     ;vect=0x08
          .db4    0xffffffff     ;vect=0x0A
          .db4    0xffffffff     ;vect=0x0C
          .db4    0xffffffff     ;vect=0x0E
          .db4    0xffffffff     ;vect=0x10
          .db4    0xffffffff     ;vect=0x12
          br      !!_r_isr_txi   ;vect=0x14
          br      !!_r_isr_rxi   ;vect=0x16
          br      !!_r_isr_eri   ;vect=0x18
          .db4    0xffffffff     ;vect=0x1A
          .db4    0xffffffff     ;vect=0x1C
          .db4    0xffffffff     ;vect=0x1E
          .db4    0xffffffff     ;vect=0x20
```

登録

4.11 デモプロジェクトと異なるメモリ構成で使用する場合

デモプロジェクトと異なるメモリ構成でファームウェアアップデートを実施したい場合、パッケージに添付されているパラメータファイルをそのまま使用することはできません。必要な箇所をメモリ構成に沿って変更頂くことで初期イメージや更新イメージを生成することができます。

デモプロジェクトのメモリ構成に関しては、6.2 デモプロジェクトの動作確認環境を、パラメータファイルの詳細に関しては、5.3 パラメータファイルを参照して下さい。

例 1 として、RL78/G23 の半面更新方式向けに提供しているパラメータファイルを、以下の様なメモリ構成に変更した場合の変更箇所を示します。

・製品名：RL78/G23
・アップデート方式：半面更新方式
・コードフラッシュサイズ：768KB → 256KB
・ブートローダサイズ：64KB → 48KB

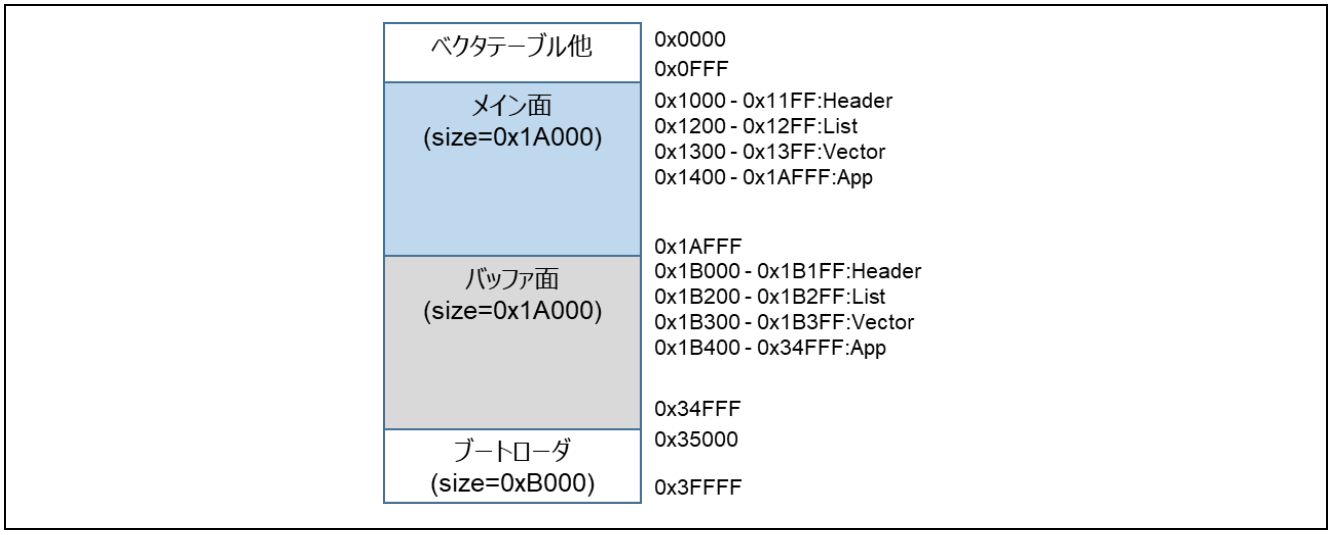


図 4-6 RL78/G23（コードフラッシュ:256KB,ブートローダ:48KB）の半面更新方式のメモリマップ

この例では、ブートローダのサイズをデモプロジェクトで確保している 64KB 注から 48KB 注に削減しています。
ファームウェアアップデートのコンフィグレーションオプション設定等によりブートローダのサイズも変化しますので、それに合わせてブートローダが使用しない領域をメイン面/バッファ面に転用可能です。

注）ブートローダのサイズにはベクタテーブル他も含みます。

パラメータファイルの変更箇所は以下の通り。

・Bootloader Start Address: 0x000B1000 → 0x00035000
・Bootloader End Address: 0x000BFFFF → 0x0003FFFF
・User Program End Address: 0x00058FFF → 0x0001AFFF

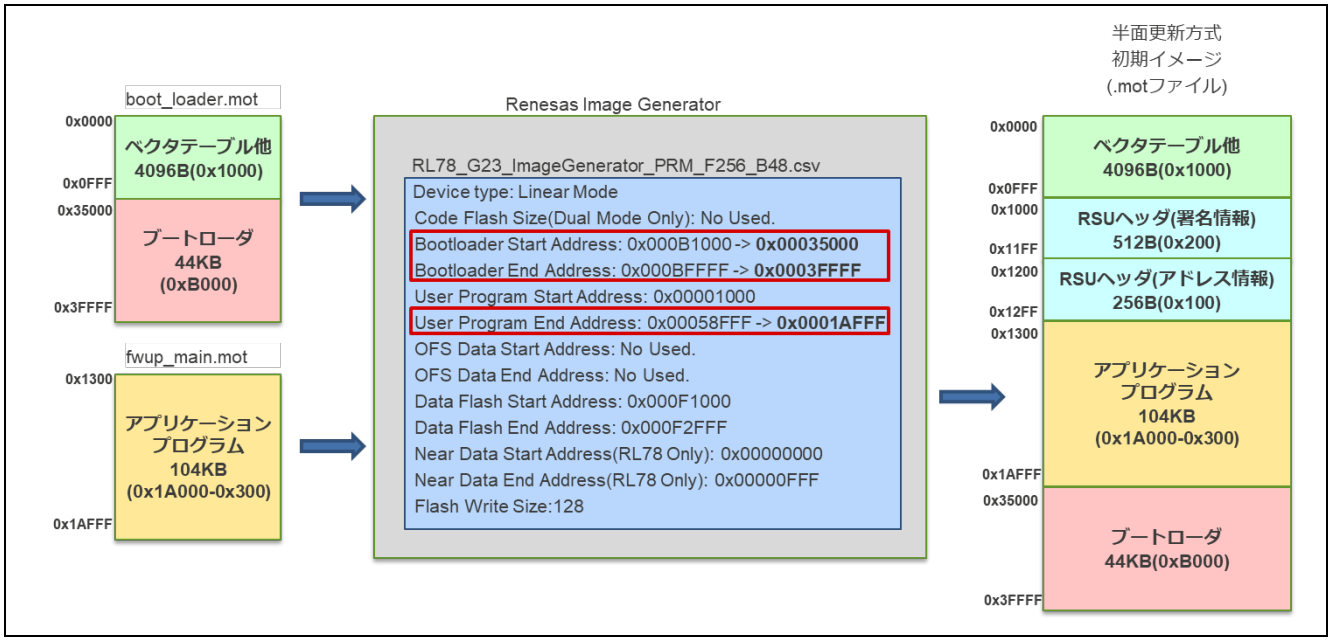


図 4-7 RL78/G23（コードフラッシュ:256KB,ブートローダ:48KB）の
半面更新方式のパラメータファイル変更例

例 2 として、RL78/L23 のデュアルバンク（2 バンク）方式向けに提供しているパラメータファイルを、以下の様なメモリ構成に変更した場合の変更箇所を示します。

・製品名：RL78/L23
・アップデート方式：デュアルバンク（2 バンク）方式
・コードフラッシュサイズ：512KB → 256KB
・ブートローダサイズ：64KB → 48KB

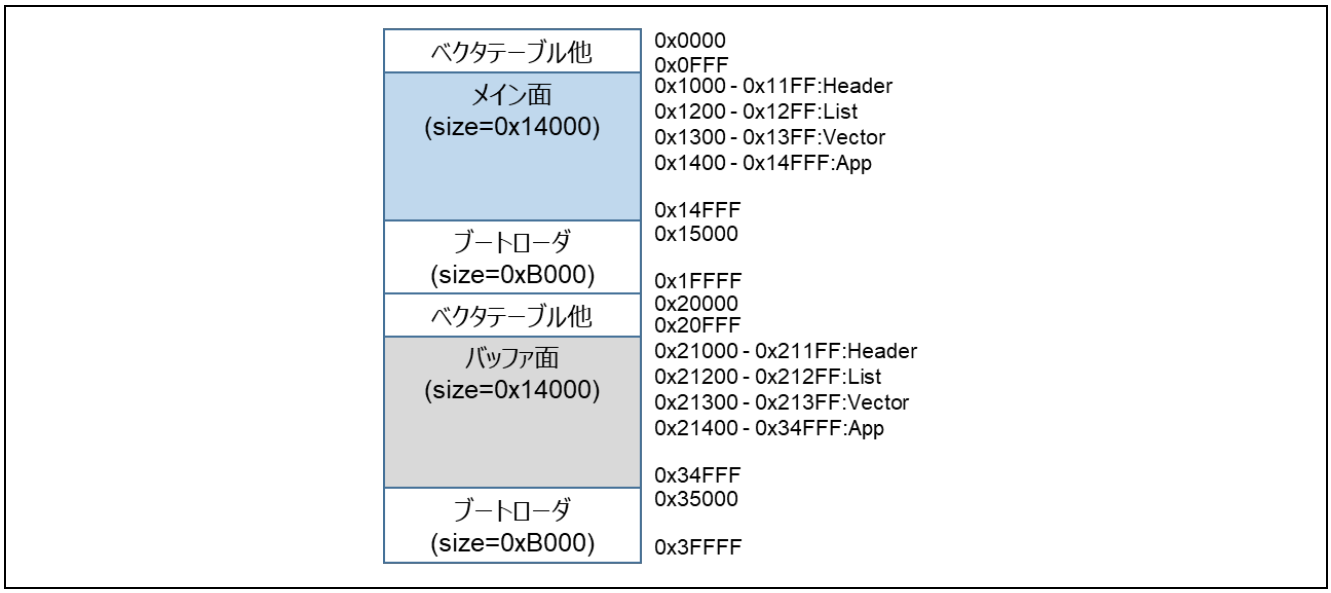


図 4-8 RL78/L23（コードフラッシュ:256KB,ブートローダ:48KB）の
デュアルバンク（2 バンク）方式のメモリマップ

パラメータファイルの変更箇所は以下の通り。

・Code Flash Size(Dual Mode Only): 0x00080000 → 0x00040000
・Bootloader Start Address: 0x00031000 → 0x00015000
・Bootloader End Address: 0x0003FFFF → 0x0001FFFF
・User Program End Address: 0x00030FFF → 0x00014FFF

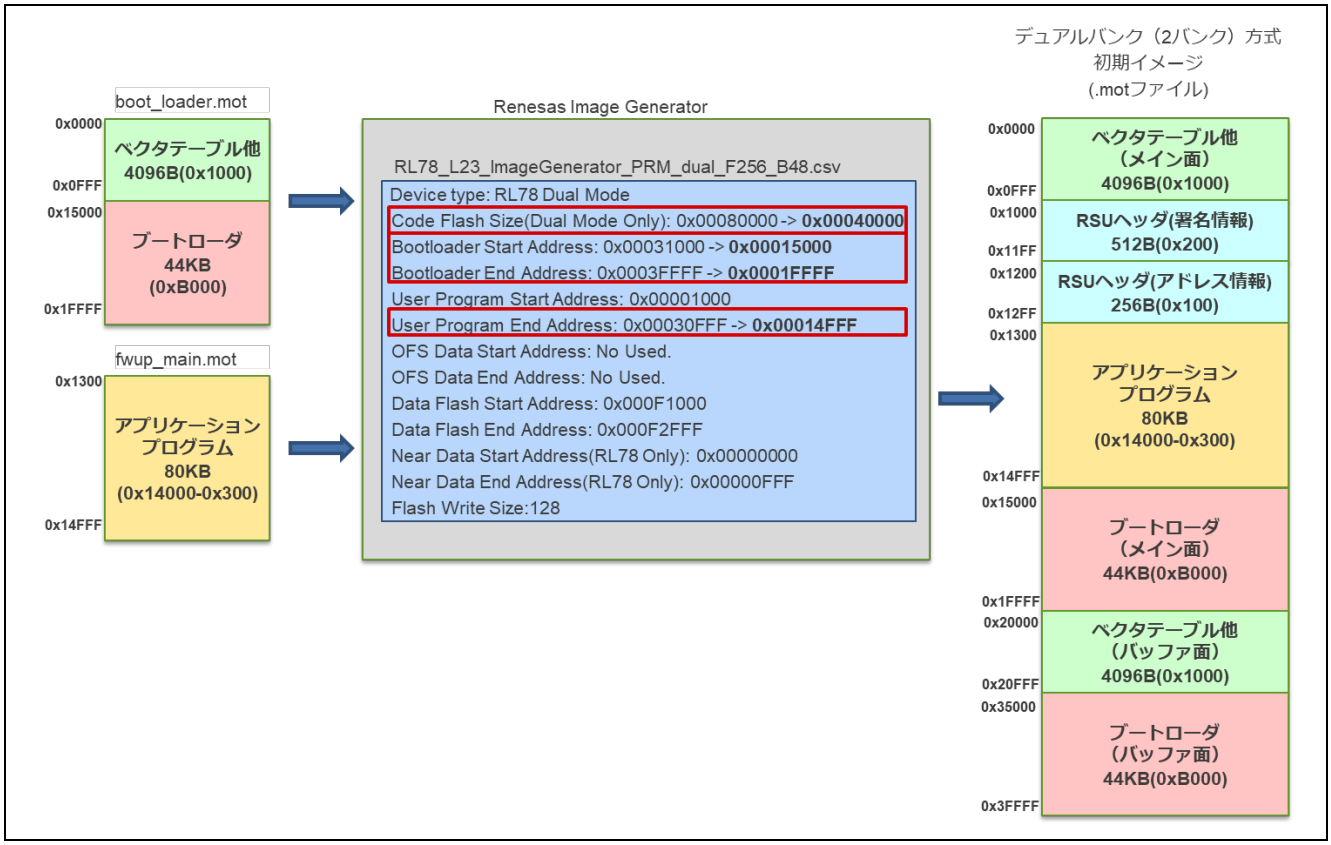


図 4-9 RL78/L23（コードフラッシュ:256KB,ブートローダ:48KB）の
デュアルバンク（2 バンク）方式のパラメータファイル変更例

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：AWS S3 (OTA)専用のイメージを生成します。 生成されるイメージは RSU ヘッダ署名情報（イメージの先頭から 0x200 バイトのデータ）を付加しません。 AWS S3 (OTA)以外で使用される場合は BareMetal を設定してください。 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のファイル名(.mot)、アプリケーションプログラムのファイル名(.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のファイル名(.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。

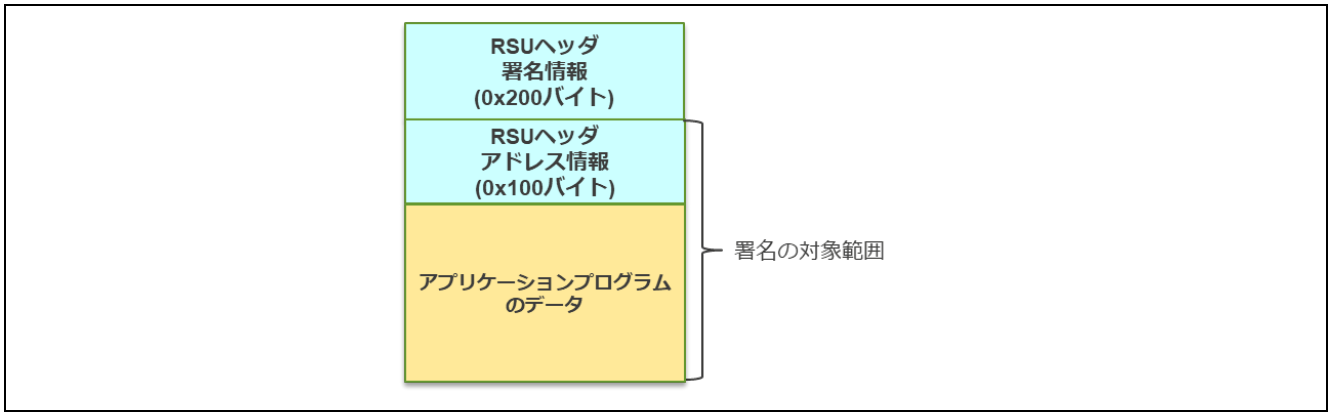


図 5-1 更新イメージファイルの構成

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のデータおよ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、デュアルバンク（2 バンク）方式、半面更新方式、全面更新方式で同じ構成です。

表 5-3 RSU ヘッダ署名情報フォーマット

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 ヘッダアドレス情報フォーマット

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。

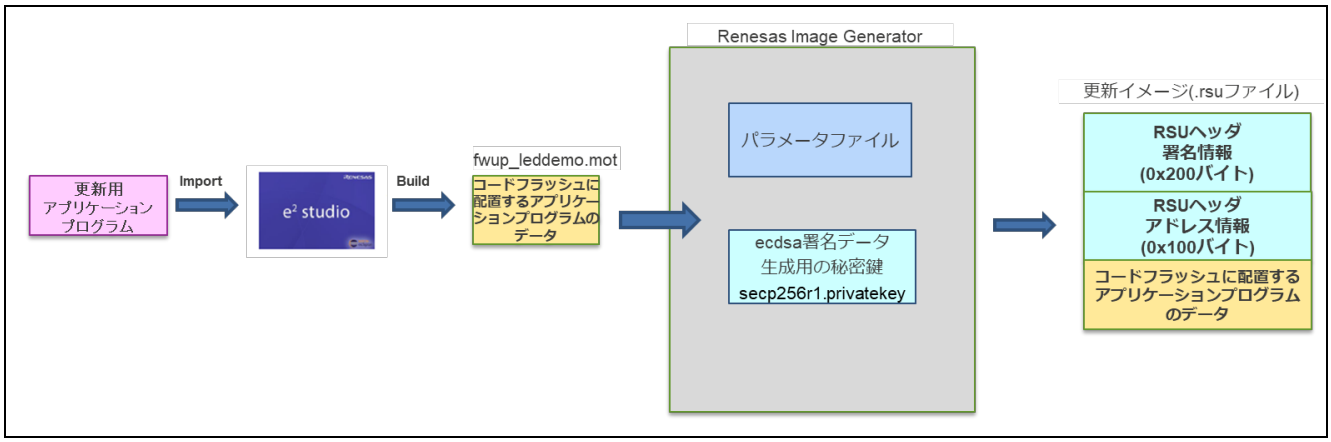


図 5-2 更新イメージファイル生成のメカニズム

・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、図 5-4 に初期イメージファイルの構成図を示します。

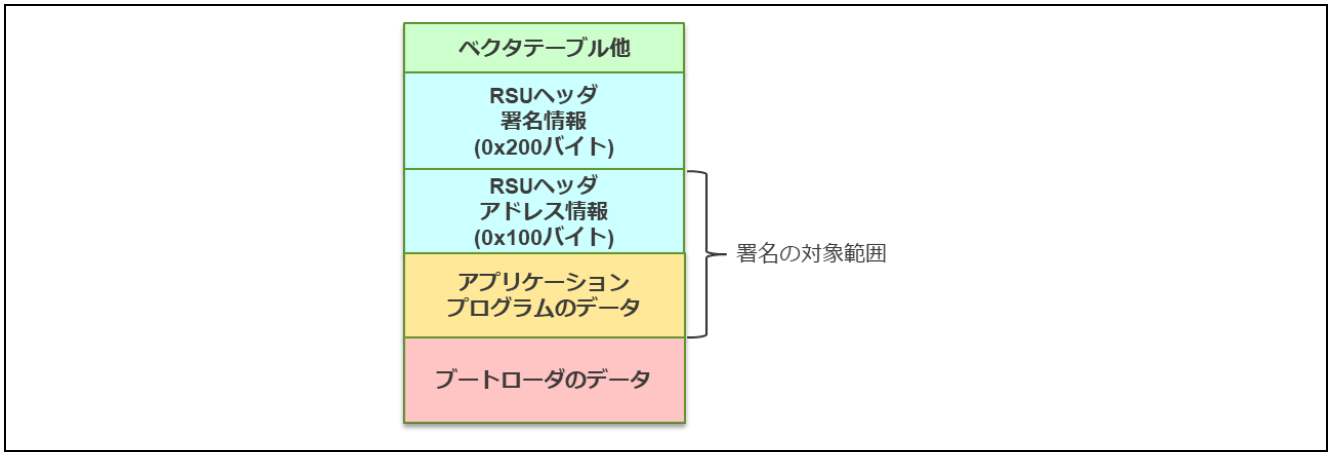


図 5-3 半面／全面更新方式の初期イメージファイルの構成

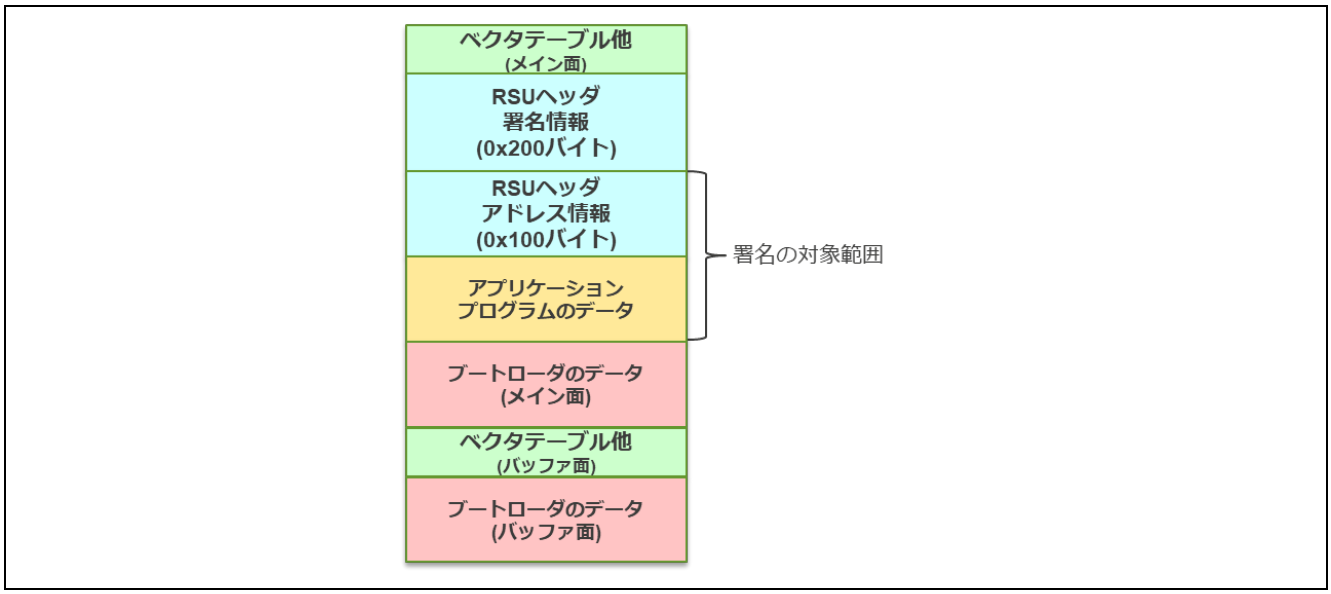


図 5-4 デュアルバンク（2 バンク）方式の初期イメージファイルの構成

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 5-5、図 5-6 を参照してください。

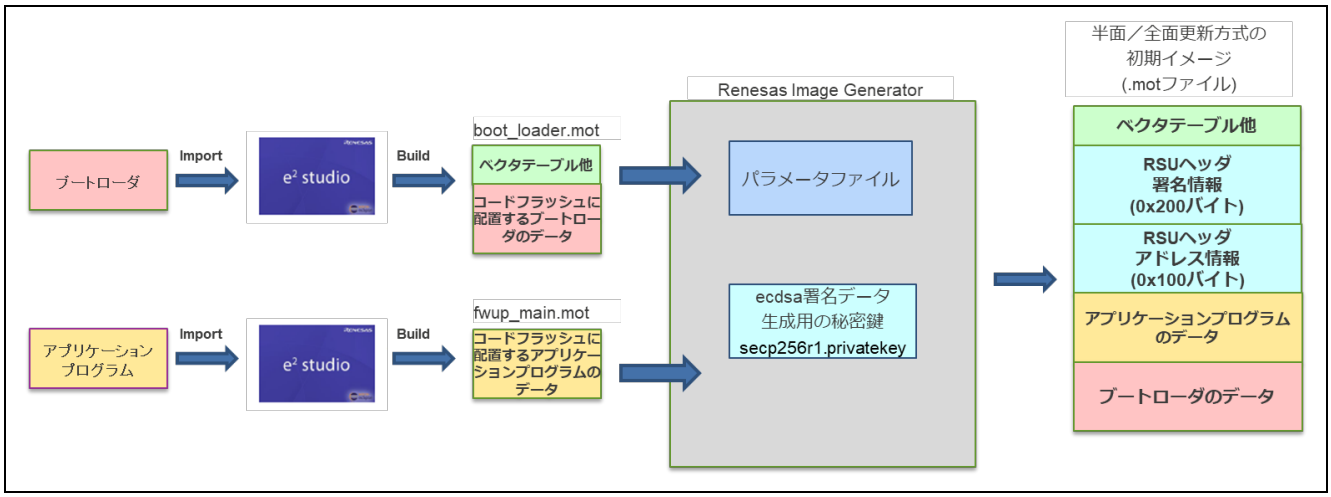


図 5-5 半面／全面更新方式の初期イメージファイル生成のメカニズム

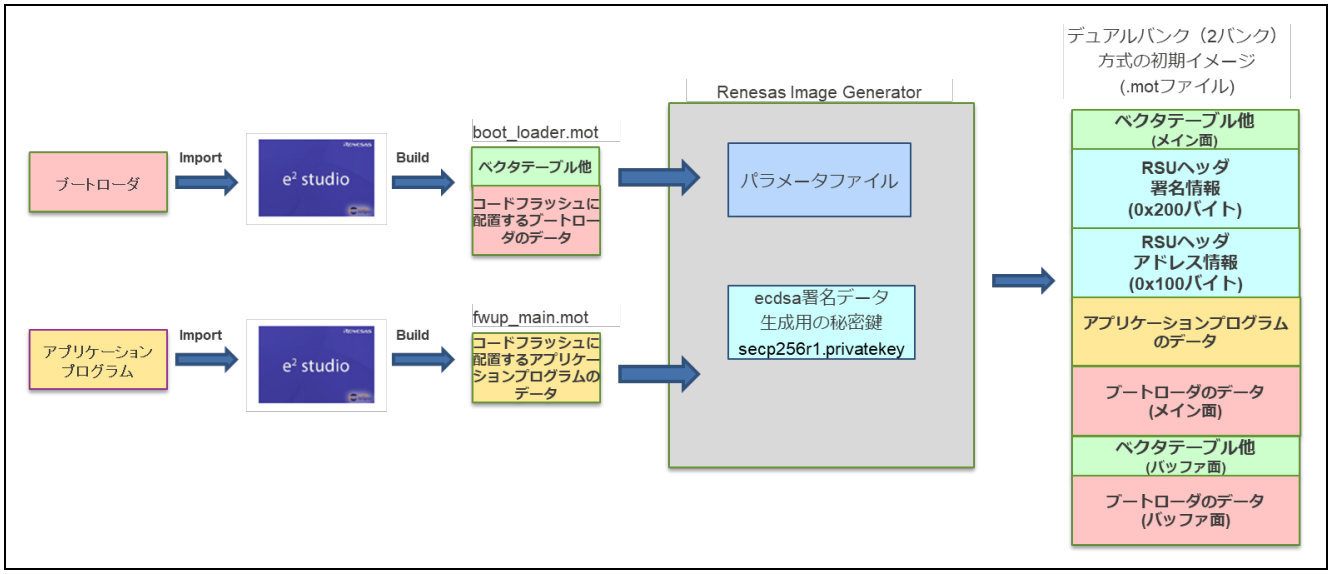


図 5-6 デュアルバンク（2 バンク）方式の初期イメージファイル生成のメカニズム

・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 により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

表 5-5 にパラメータファイルの各パラメータについて示します。

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 パラメータファイルの内容

パラメータ名	説明
Device Type	RL78 Dual Mode：デュアルバンク（2 バンク）方式向け mot ファイル生成 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
Code Flash Size (Dual Mode Only)	コードフラッシュのサイズ （デュアルバンク（2 バンク）方式のみ設定）
Bootloader Start Address	ブートローダの開始アドレス
Bootloader End Address	ブートローダの終了アドレス
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
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
OFS Data Start Address	OFSM データ開始アドレス（RL78 では'No Used.'を設定してください）
OFS Data End Address	OFSM データ終了アドレス（RL78 では'No Used.'を設定してください）
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)
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)
Near Data Start Address (RL78 Only)	RL78 用 Near ブートローダ開始アドレス
Near Data End Address (RL78 Only)	RL78 用 Near ブートローダ終了アドレス
Flash Write Size	フラッシュ書き込みサイズ (フラッシュへ 1 回に書き込むバイト数を 10 進数で指定します)

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。

次にイメージファイルを生成する際に参照するパラメータの詳細を示します。

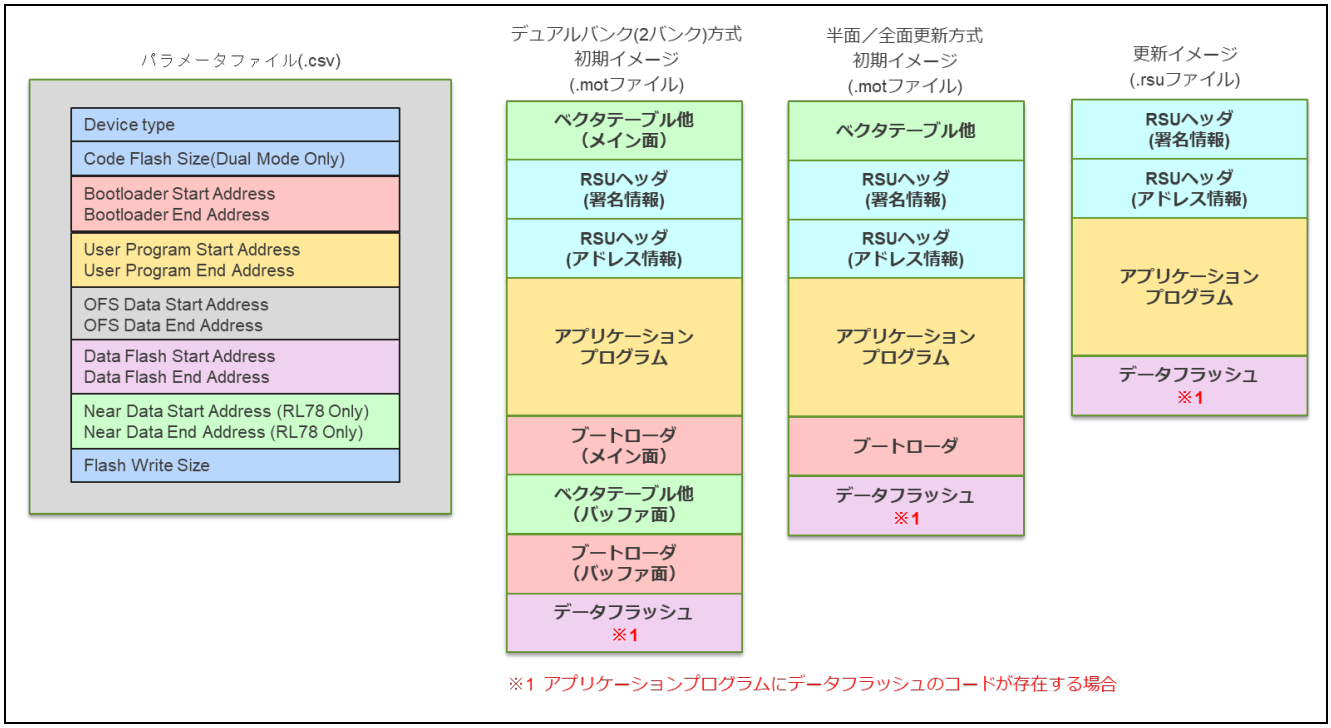


図 5-7 イメージファイルを生成する際に参照するパラメータ

・Device type：
　デュアルバンク（2 バンク）方式の初期イメージを生成するかどうかを判断するために使用します。
　Device type が RL78 Dual Mode の場合には、ブートローダ（メイン面）とブートローダ（バッファ面）を生成し、Linear Mode の場合には、ブートローダ（メイン面）のみを生成します。

・Code Flash Size(Dual Mode Only)：
　ブートローダ（バッファ面）を配置するアドレスを決めるために使用します。

・Bootloader Start Address - Bootloader End Address：
　設定範囲をブートローダのコードとしてイメージファイルを生成します。
　ブートローダファイル（,mot）を入力データとします。

・User Program Start Address - User Program End Address：
　設定範囲をアプリケーションプログラムのコードとしてイメージファイルを生成します。
　アプリケーションプログラムファイル(.mot) を入力データとします。

・Data Flash Start Address - Data Flash End Address：
　設定範囲をデータフラッシュのデータとしてイメージファイルを生成します。（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）
　アプリケーションプログラムファイル(.mot) を入力データとします。

・Near Data Flash Start Address (RL78 Only) - Near Data Flash End Address (RL78 Only)：
　設定範囲をブートローダのベクタテーブル他のコードとしてイメージファイルを生成します。
　ブートローダファイル(.mot)を入力データとします。

・Flash Write Size：
　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。

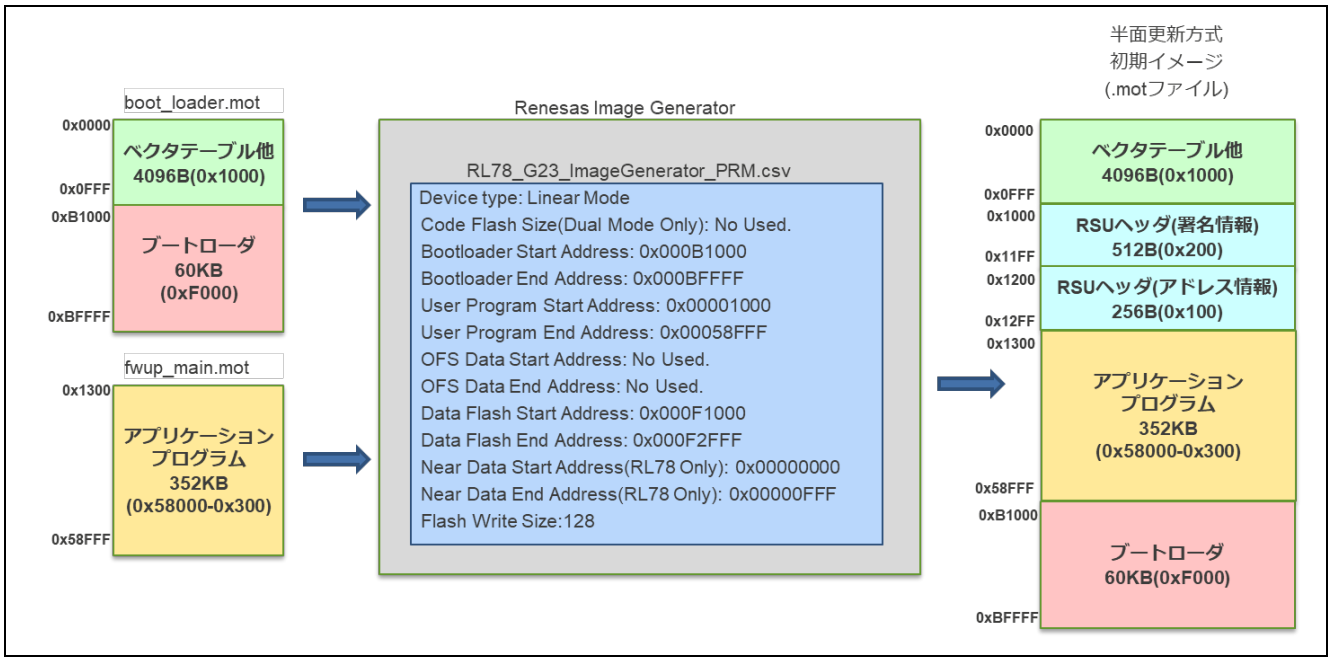


図 5-8 RL78/G23 向けデモプロジェクトの半面更新方式の初期イメージを生成する際のパラメータ

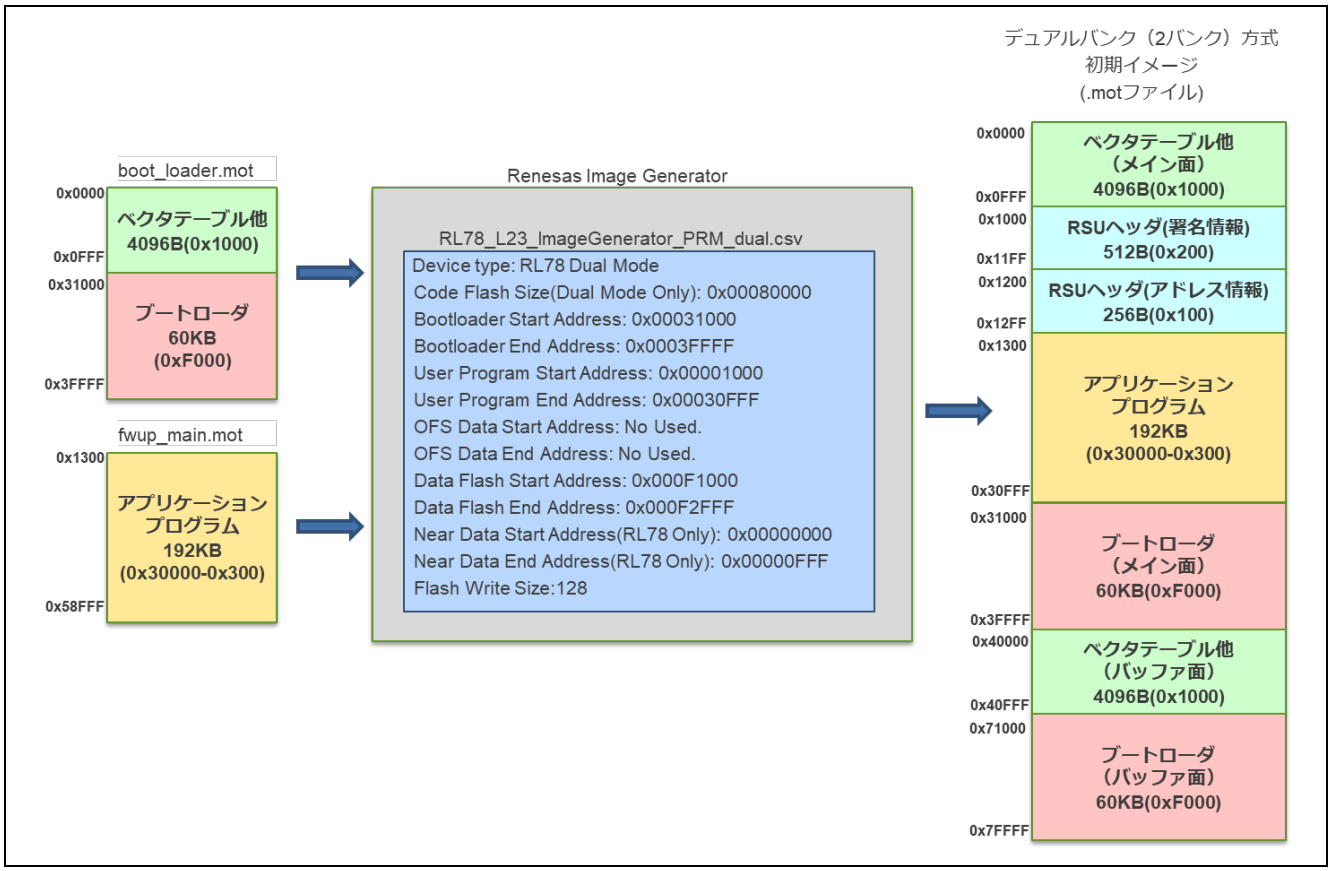


図 5-9 RL78/L23 向けデモプロジェクトのデュアルバンク（2 バンク）方式の初期イメージを生成する際のパラメータ

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 2025-12
C コンパイラ	ルネサスエレクトロニクス製 CC-RL V1.16.00 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.2.05
使用ボード	RL78/G22 Fast Prototyping Board（RTK7RLG220C00000BJ） RL78/G23-128p Fast Prototyping Board（RTK7RLG230CSN000BJ） RL78/G24 Fast Prototyping Board（RTK7RLG240C00000BJ） RL78/L23- Fast Prototyping Board（RTK7RLL230S00001BJ）

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 5.20.2
C コンパイラ	IAR Systems 製 IAR C/C++ Compiler for Renesas RL78 version 5.20.2 IAR Assembler for Renesas RL78 version 5.20.2 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.2.05
使用ボード	RL78/G22 Fast Prototyping Board（RTK7RLG220C00000BJ） RL78/G23-128p Fast Prototyping Board（RTK7RLG230CSN000BJ） RL78/G24 Fast Prototyping Board（RTK7RLG240C00000BJ） RL78/L23- Fast Prototyping Board（RTK7RLL230S00001BJ）

表 6-3　動作確認環境 (LLVM)

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 2025-12
C コンパイラ	LLVM for Renesas RL78 17.0.1.202512
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.2.05
使用ボード	RL78/G22 Fast Prototyping Board（RTK7RLG220C00000BJ） RL78/G23-128p Fast Prototyping Board（RTK7RLG230CSN000BJ） RL78/G24 Fast Prototyping Board（RTK7RLG240C00000BJ） RL78/L23- Fast Prototyping Board（RTK7RLL230S00001BJ）

6.2　デモプロジェクトの動作確認環境

デモプロジェクトはボードやコンパイラ、アップデート方式毎に提供しており、それぞれのデモプロジェクトを使用する際に必要な情報を以下に示します。

6.2.1　RL78/G23 の動作確認環境

動作確認環境の接続図とピンアサインを以下に示します。

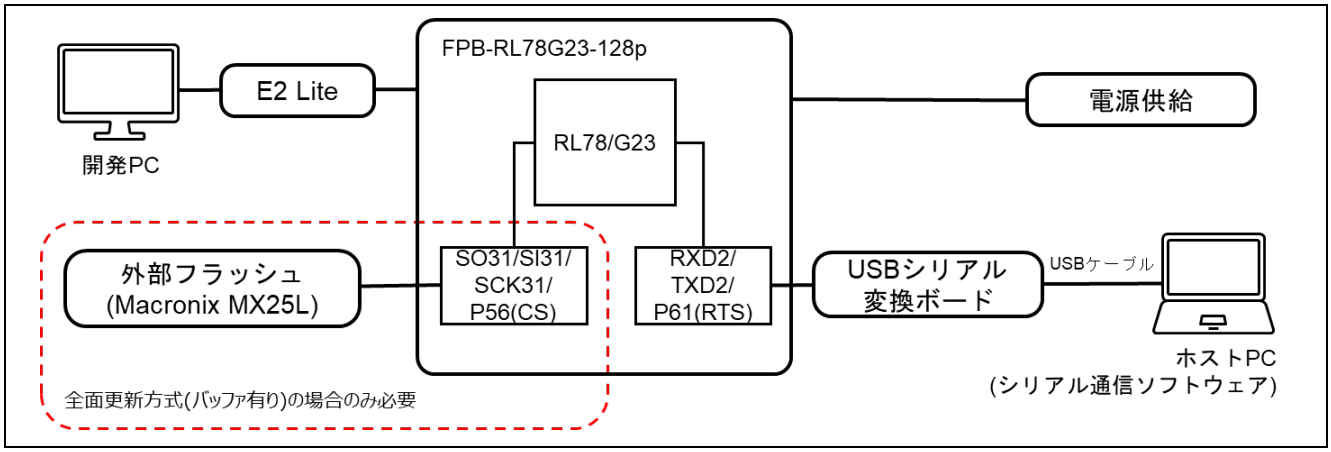


図 6-1 FPB-RL78G23-128p の機器接続図

■ UART(Red)

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

■ External Flash(Green)

MCU Header J1		MX25L	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		MX25L	Note
4	3V3	VCC	
6	GND	GND	

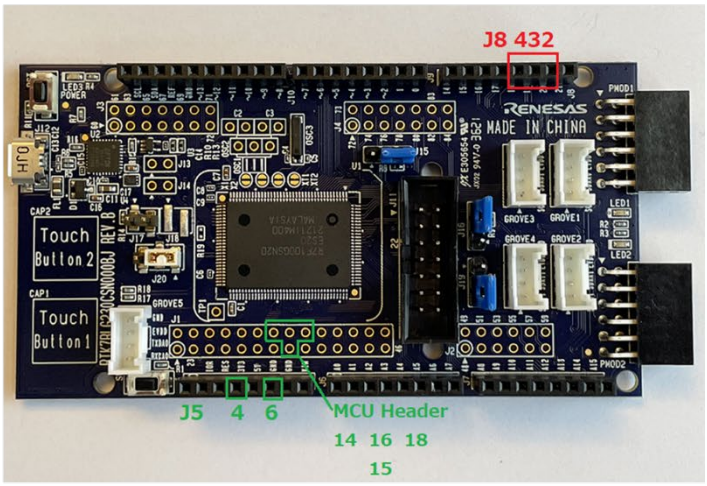


図 6-2 FPB-RL78G23-128p のピン情報

6.2.1.1　半面更新方式のデモプロジェクトの各種情報

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について、以下に示します。

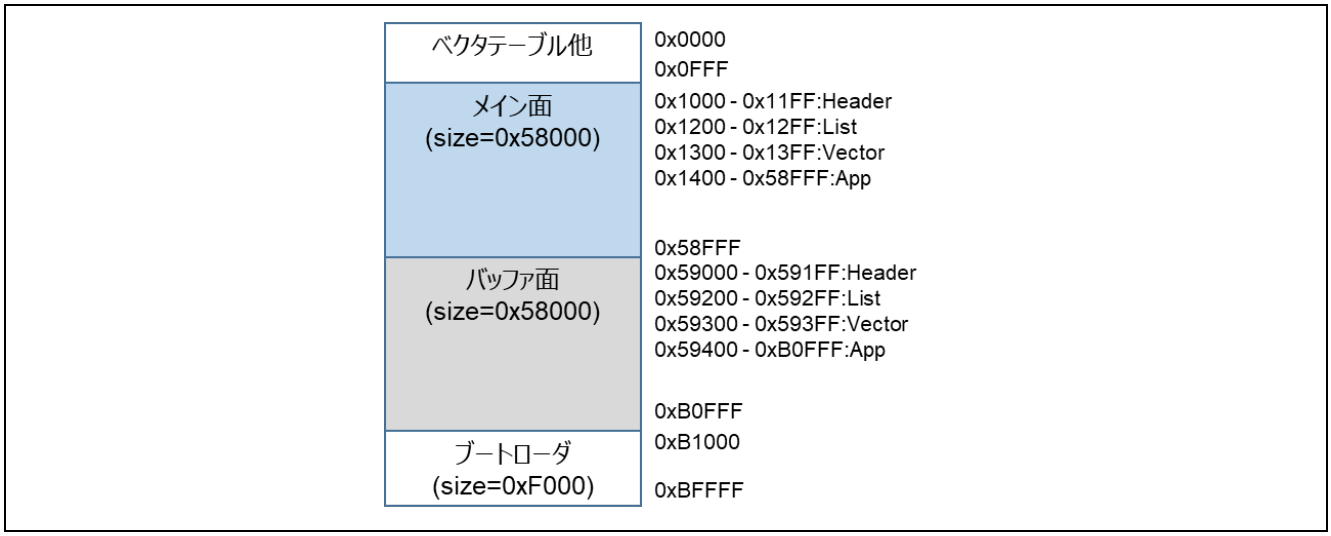


図 6-3 RL78/G23 の半面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-4 RL78/G23 の半面更新方式のコンフィグ設定（CC-RL,IAR の場合）

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	1	1

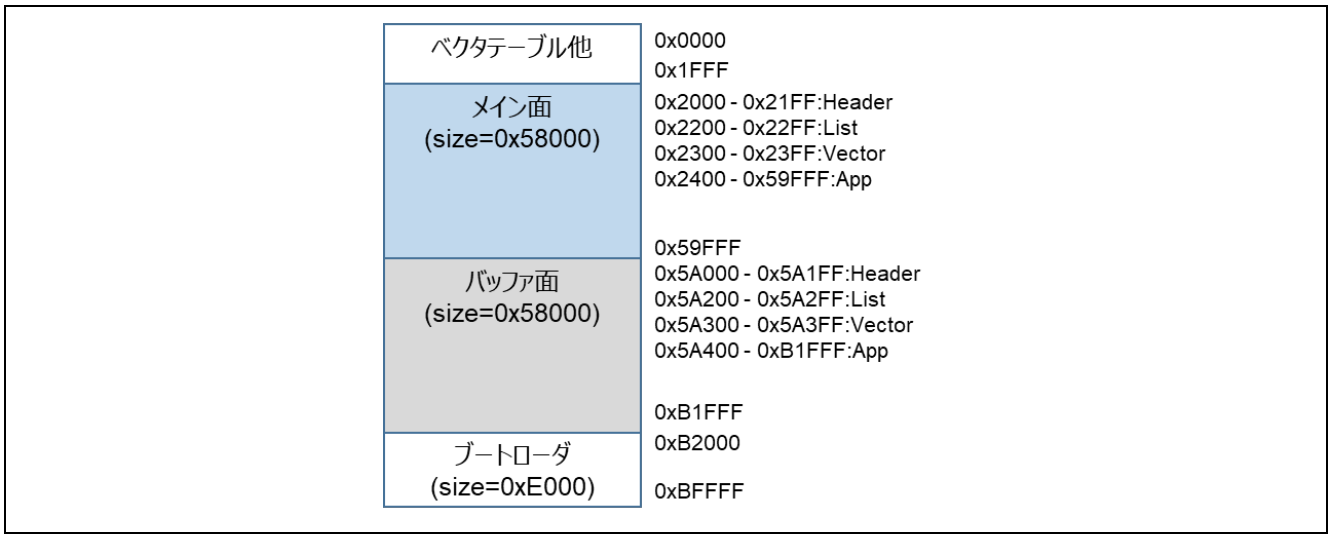


図 6-4 RL78/G23 の半面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-5 RL78/G23 の半面更新方式のコンフィグ設定（LLVM の場合）

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	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x5A000	0x5A000
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	1	1

6.2.1.2　全面更新方式のデモプロジェクトの各種情報

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について、以下に示します。

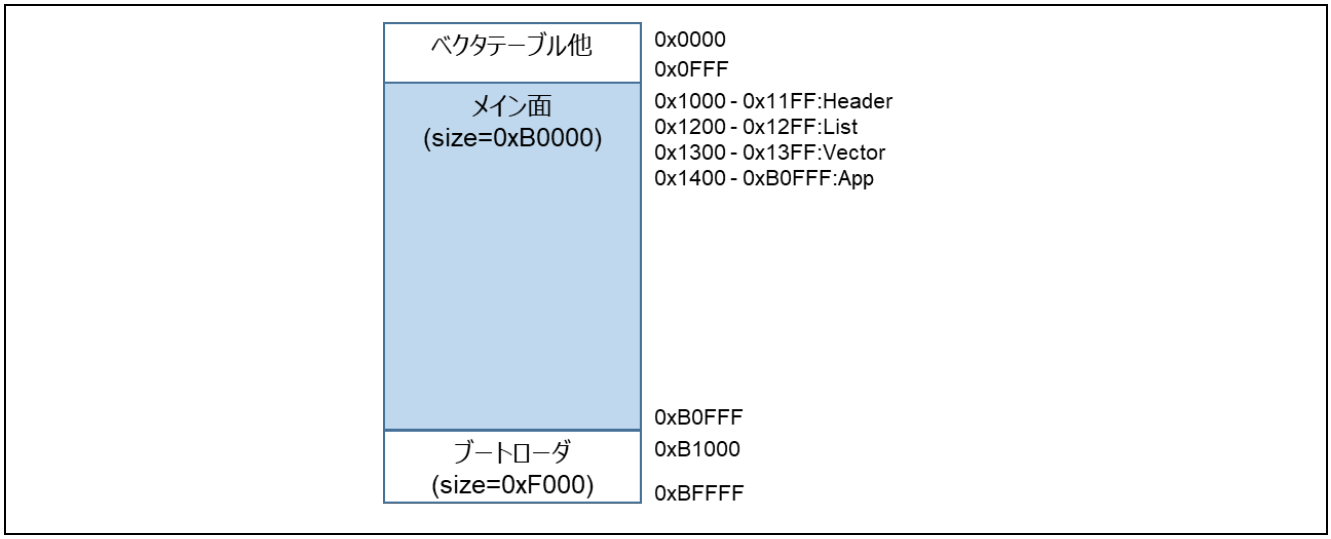


図 6-5　RL78/G23 の全面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-6 RL78/G23 の全面更新方式のコンフィグ設定（CC-RL,IAR の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	2 or 3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x1000	0x1000
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	1	1

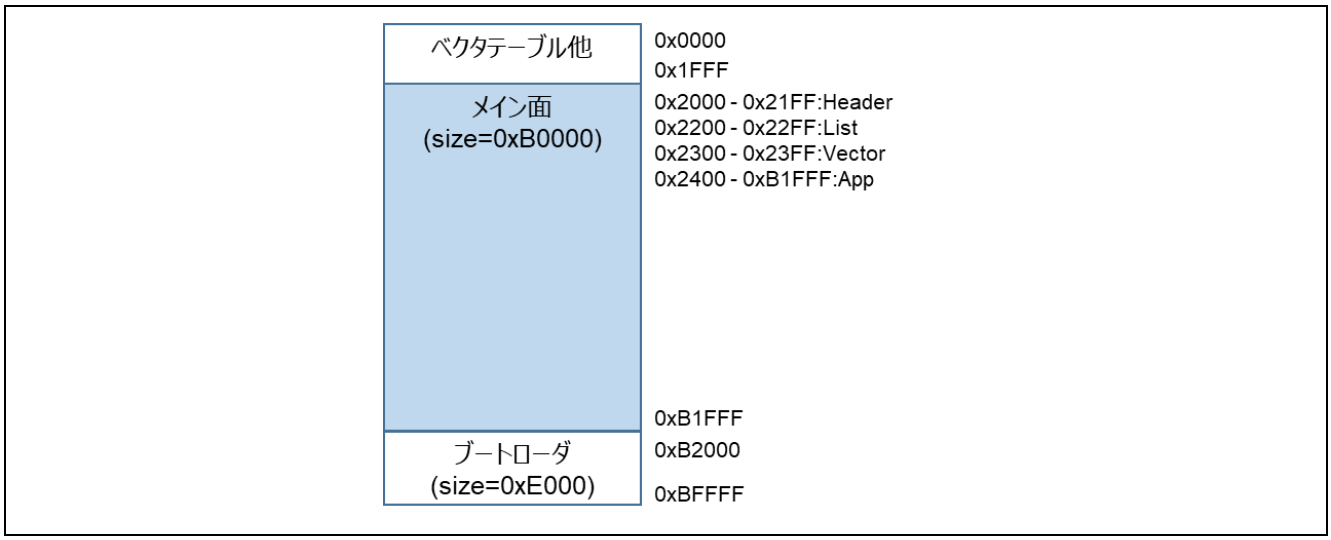


図 6-6　RL78/G23 の全面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-7 RL78/G23 の全面更新方式のコンフィグ設定（LLVM の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	2 or 3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x2000	0x2000
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	1	1

6.2.2　RL78/G24 の動作確認環境

動作確認環境の接続図とピンアサインを以下に示します。

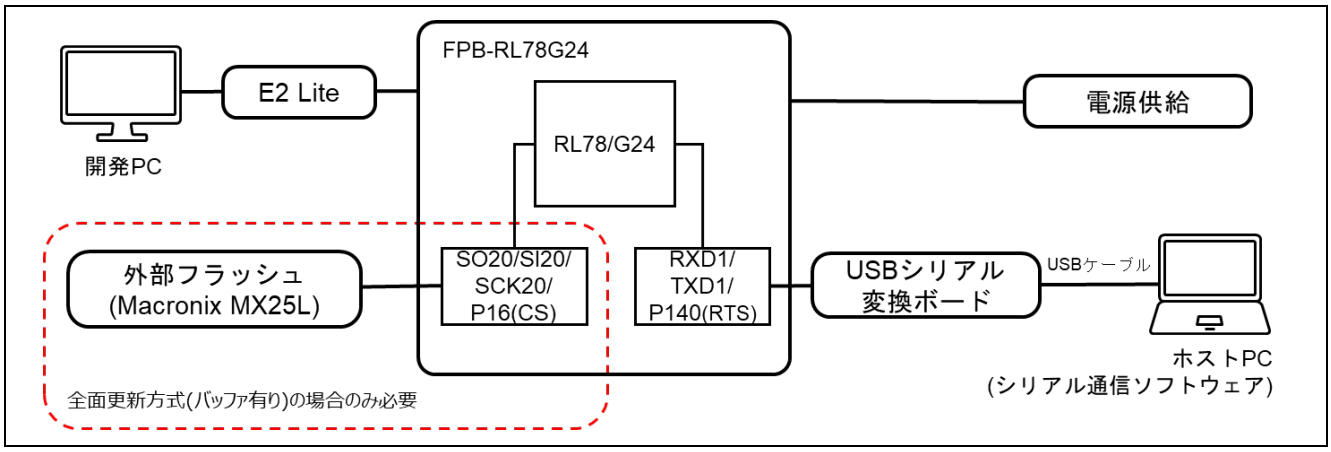


図 6-7 FPB-RL78G24 の機器接続図

■ UART(Red)

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

■ External Flash(Green)

Arduino J5,J6		MX25L	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		MX25L	Note
4	3V3	VCC	
6	GND	GND	

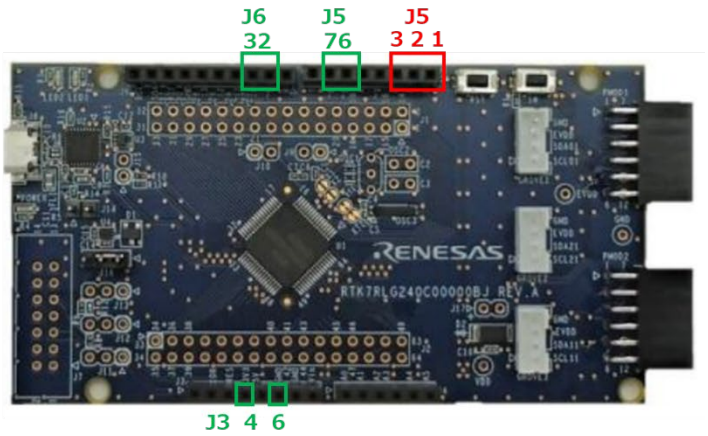


図 6-8 FPB-RL78G24 のピン情報

6.2.2.1　半面更新方式のデモプロジェクトの各種情報

RL78/G24 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について、以下に示します。

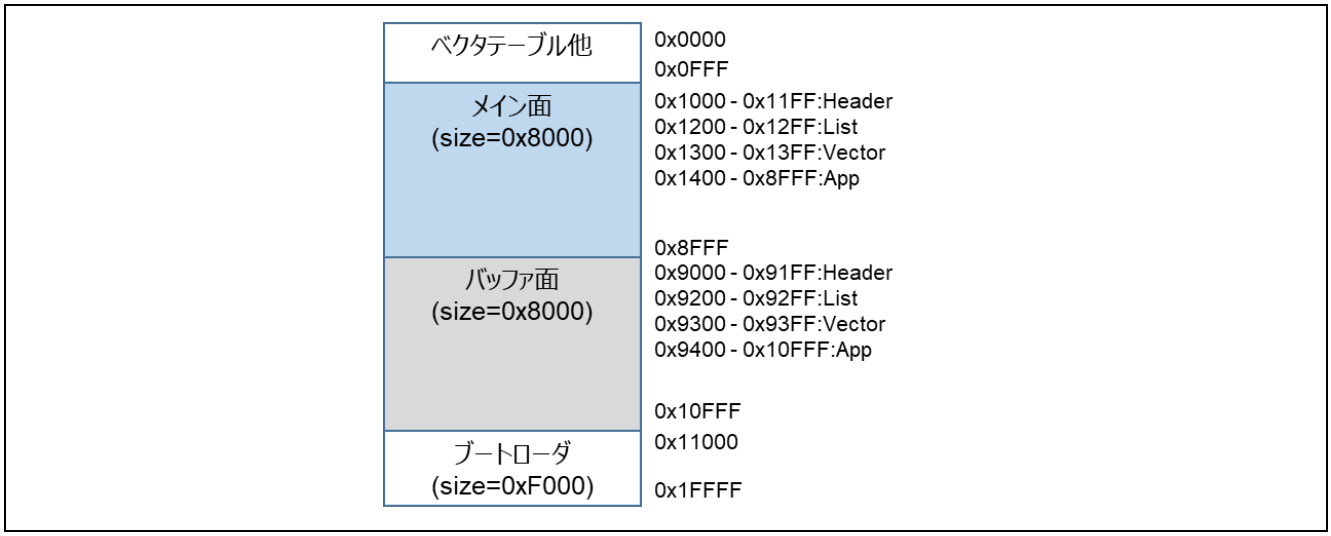


図 6-9 RL78/G24 の半面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-8 RL78/G24 の半面更新方式のコンフィグ設定（CC-RL,IAR の場合）

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	1	1

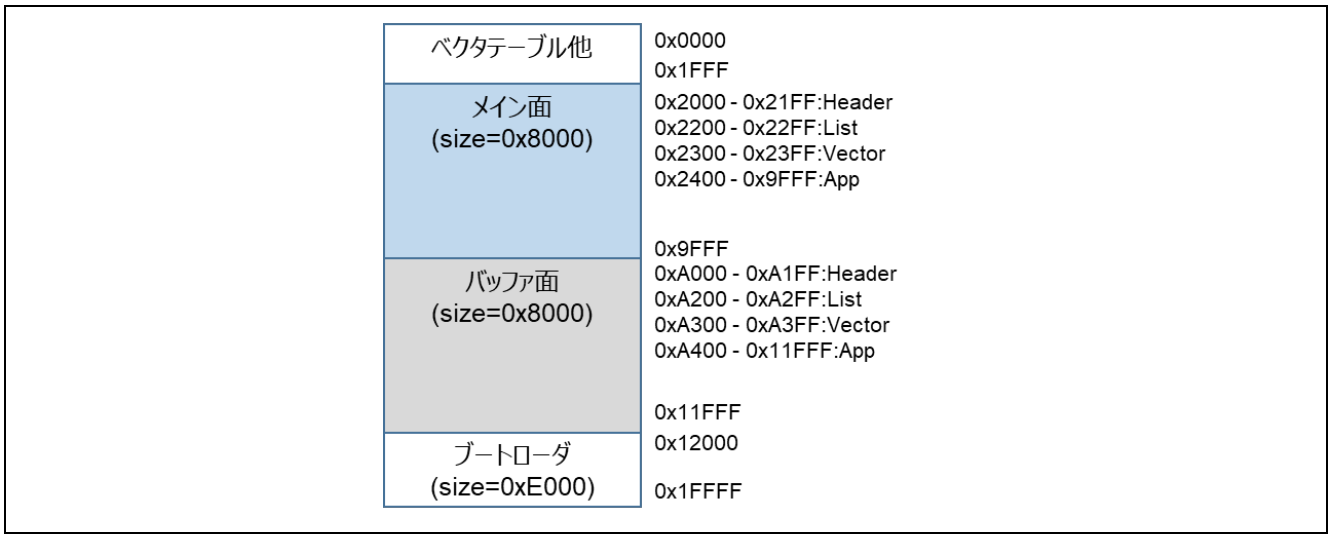


図 6-10 RL78/G24 の半面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-9 RL78/G24 の半面更新方式のコンフィグ設定（LLVM の場合）

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	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0xA000	0xA000
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	1	1

6.2.2.2　全面更新方式のデモプロジェクトの各種情報

RL78/G24 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について、以下に示します。

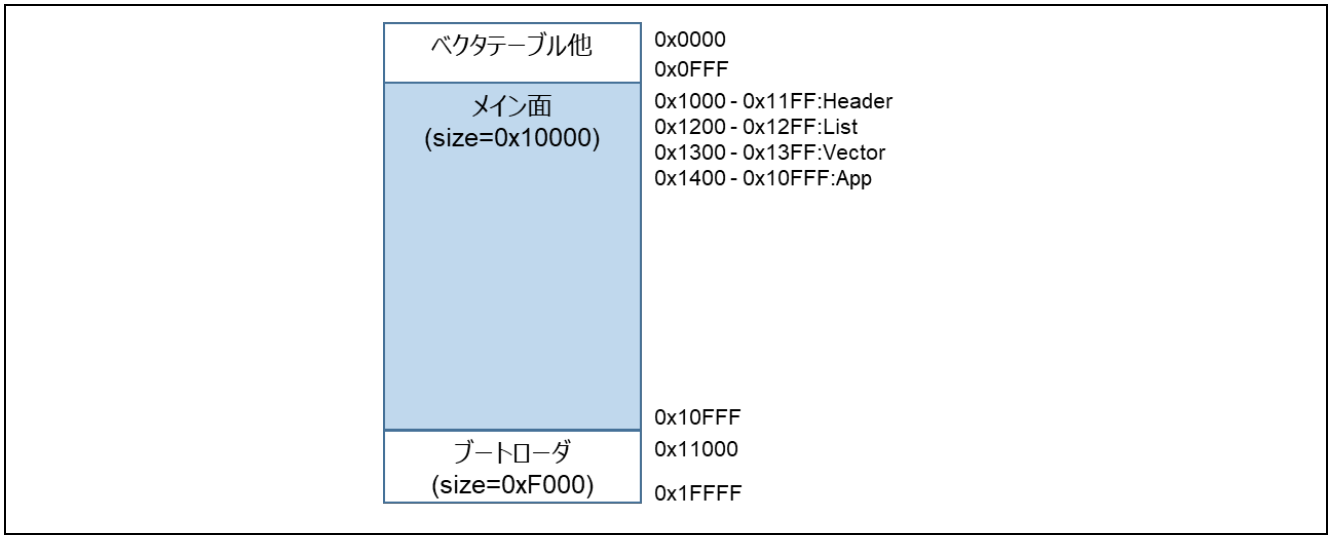


図 6-11　RL78/G24 の全面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-10 RL78/G24 の全面更新方式のコンフィグ設定（CC-RL,IAR の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	2 or 3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x1000	0x1000
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	1	1

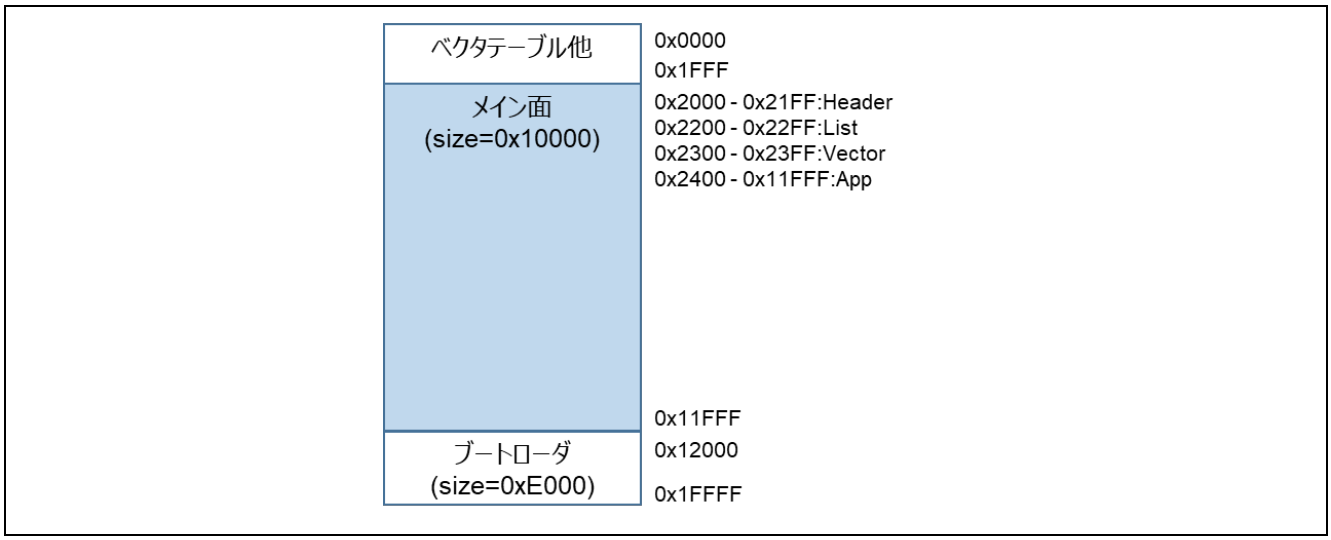


図 6-12　RL78/G24 の全面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-11 RL78/G24 の全面更新方式のコンフィグ設定（LLVM の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	2 or 3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x2000	0x2000
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	1	1

6.2.3　RL78/G22 の動作確認環境

動作確認環境の接続図とピンアサインを以下に示します。

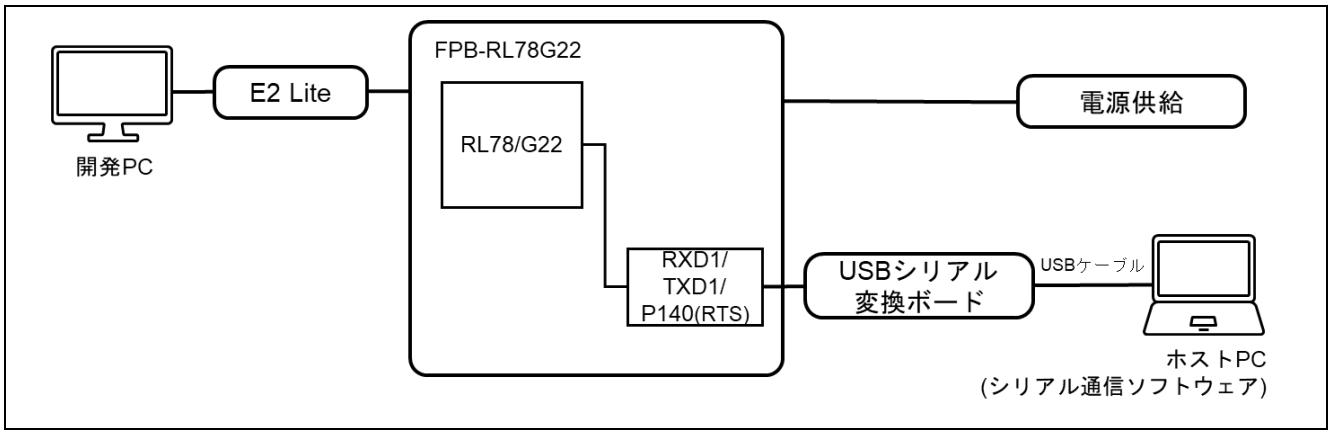


図 6-13 FPB-RL78G22 の機器接続図

■ UART(Red)

Arduino J7		USB-UART	Note
1	RXD1	TX	
2	TXD1	RX	
3	P140(RTS)	CTS	

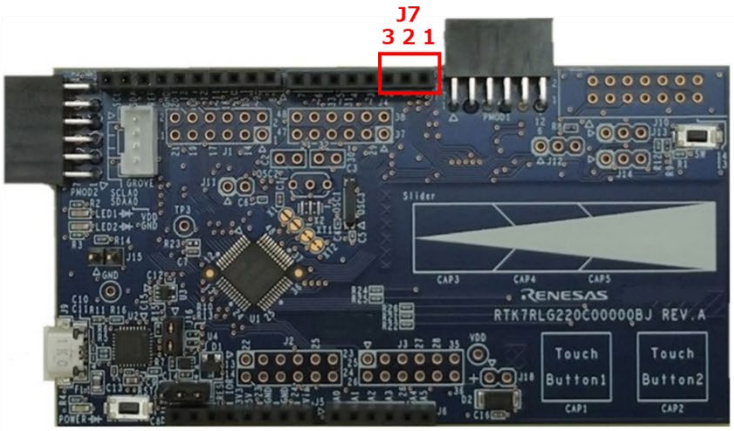


図 6-14 FPB-RL78G22 のピン情報

6.2.3.1　全面更新方式のデモプロジェクトの各種情報

RL78/G22 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について、以下に示します。

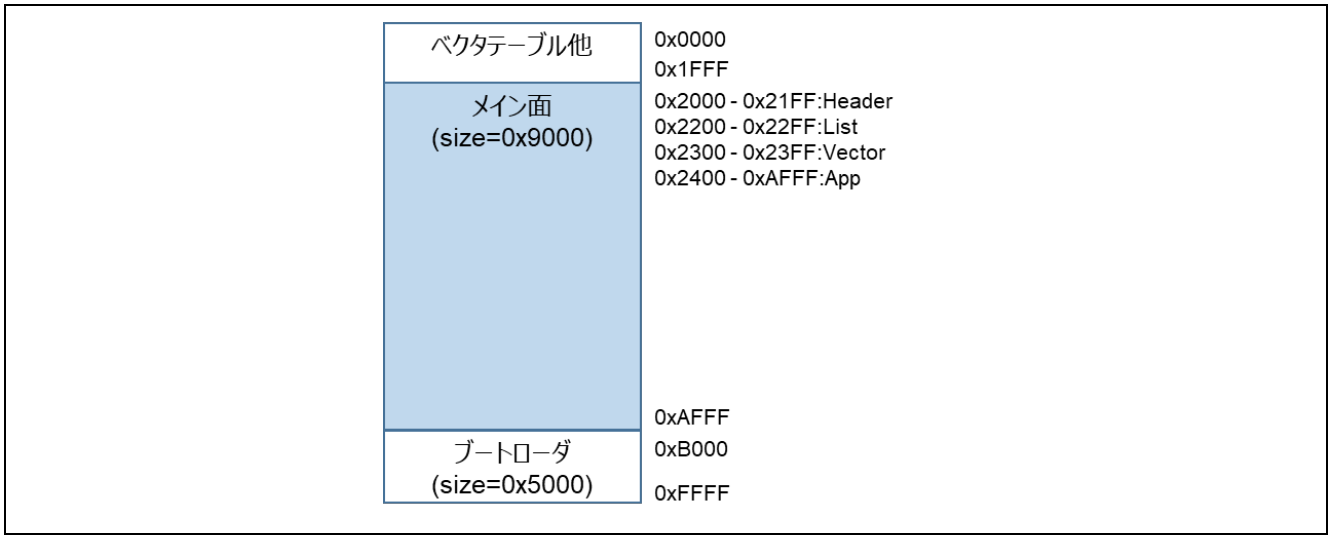


図 6-15　RL78/G22 の全面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-12 RL78/G22 の全面更新方式のコンフィグ設定（CC-RL,IAR の場合）

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	1

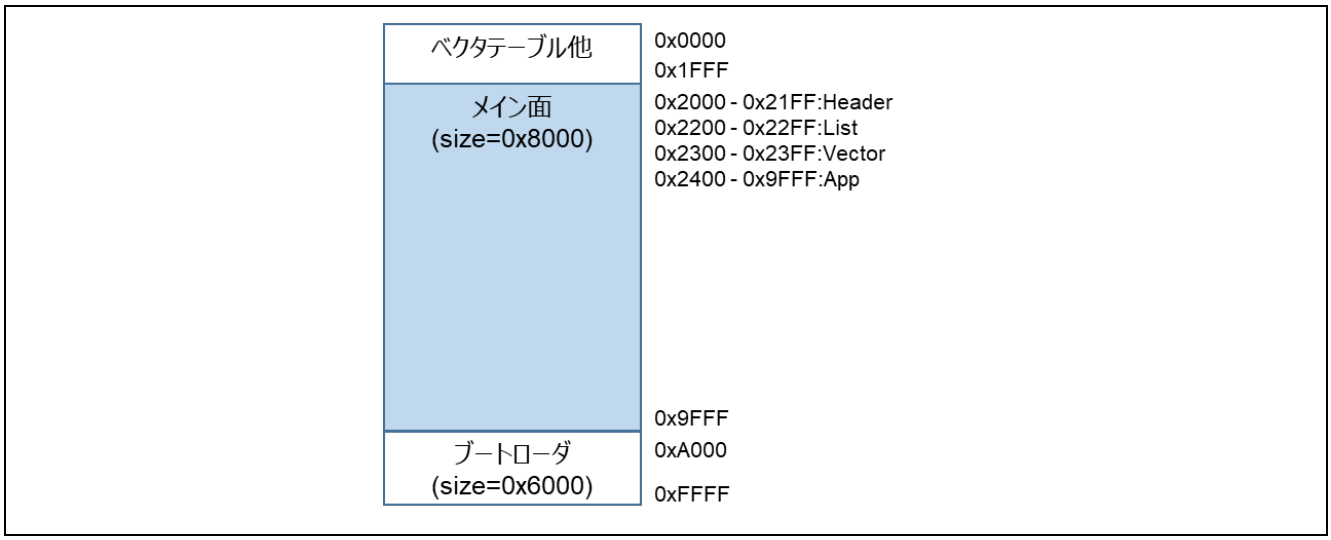


図 6-16　RL78/G22 の全面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-13 RL78/G22 の全面更新方式のコンフィグ設定（LLVM の場合）

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	0x8000
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	1

6.2.4　RL78/L23 の動作確認環境

動作確認環境の接続図とピンアサインを以下に示します。

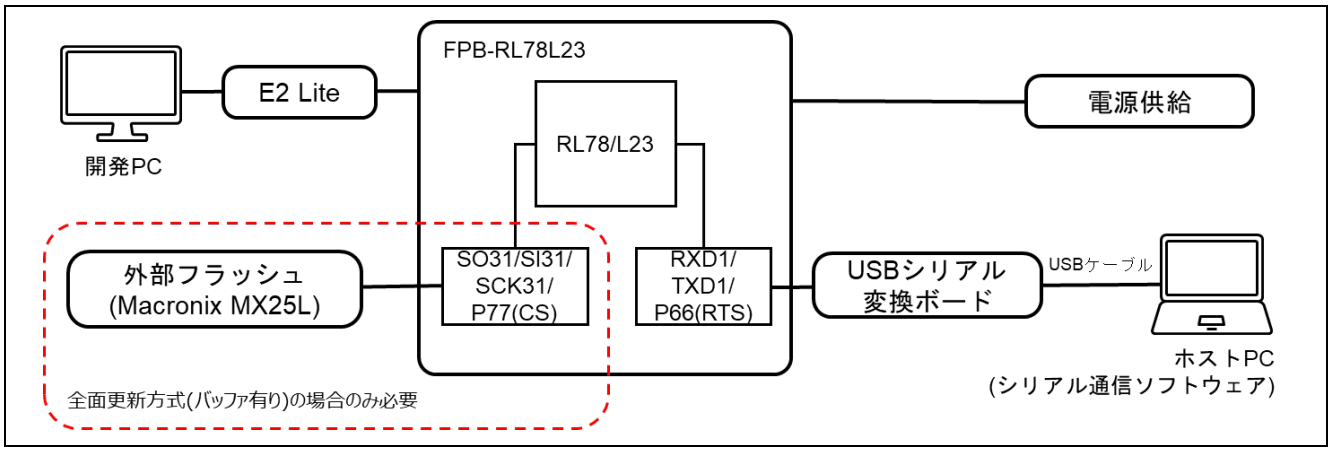


図 6-17 FPB-RL78L23 の機器接続図

■ UART(Red)

PMOD1		USB-UART	Note
2	RXD1	TX	
3	TXD1	RX	
4	P66(RTS)	CTS	

■ External Flash(Green)

Arduino J10		MX25L	Note
2	P77(CS)	CE#	1Kohm pull up
4	SO31	SI	1Kohm pull up
5	SI31	SO	1Kohm pull up
6	SCK31	SCLK	1Kohm pull up

Arduino J7		MX25L	Note
4	3V3	VCC	
6	GND	GND	

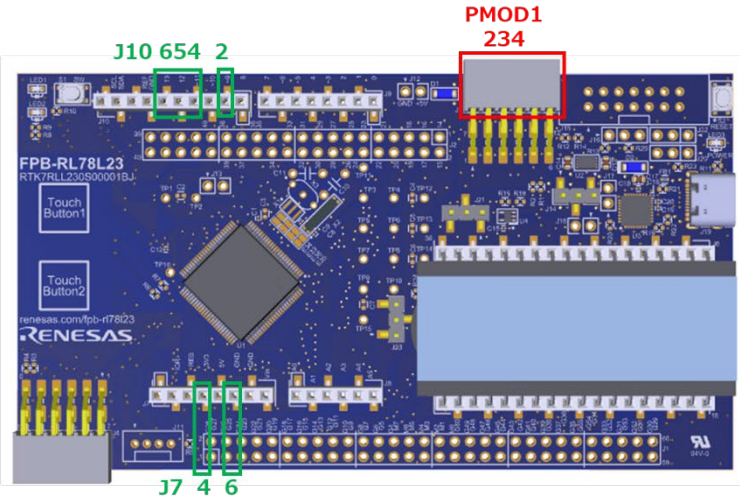


図 6-18 FPB-RL78L23 のピン情報

6.2.4.1　デュアルバンク（2 バンク）方式のデモプロジェクトの各種情報

RL78/L23 のデュアルバンク（2 バンク）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について、以下に示します。

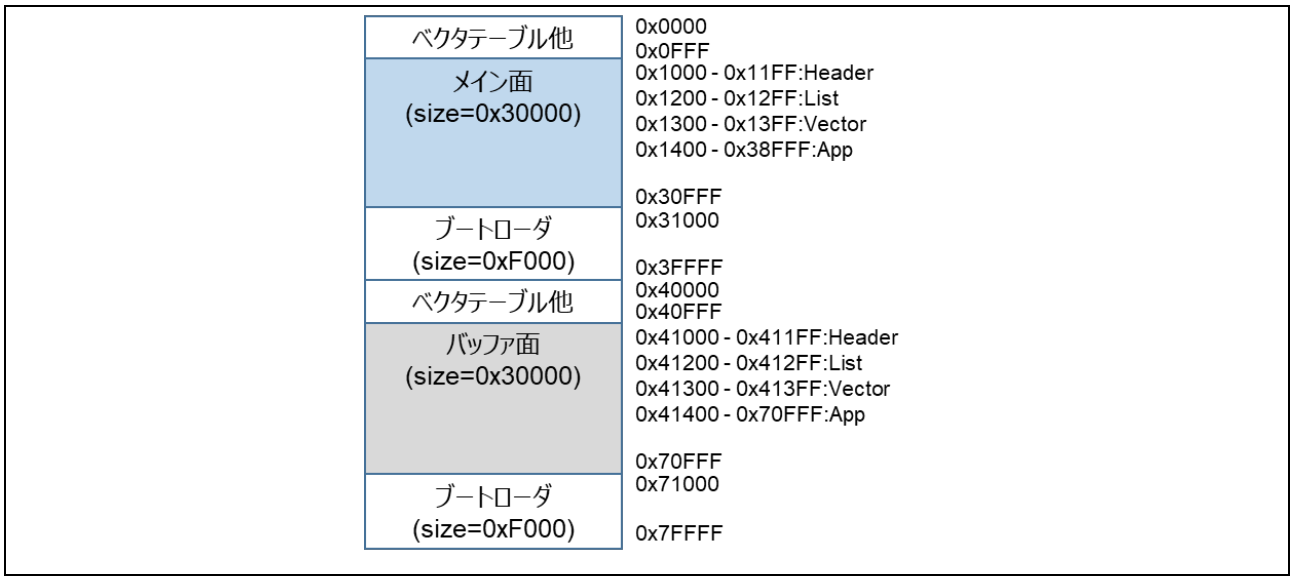


図 6-19 RL78/L23 デュアルバンク（2 バンク）方式のデモプロジェクトのメモリマップ
（CC-RL,IAR の場合）

表 6-14 RL78/L23 のデュアルバンク（2 バンク）方式のコンフィグ設定（CC-RL,IAR の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	0	0
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x41000	0x41000
FWUP_CFG_AREA_SIZE	0x30000	0x30000
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	1	1

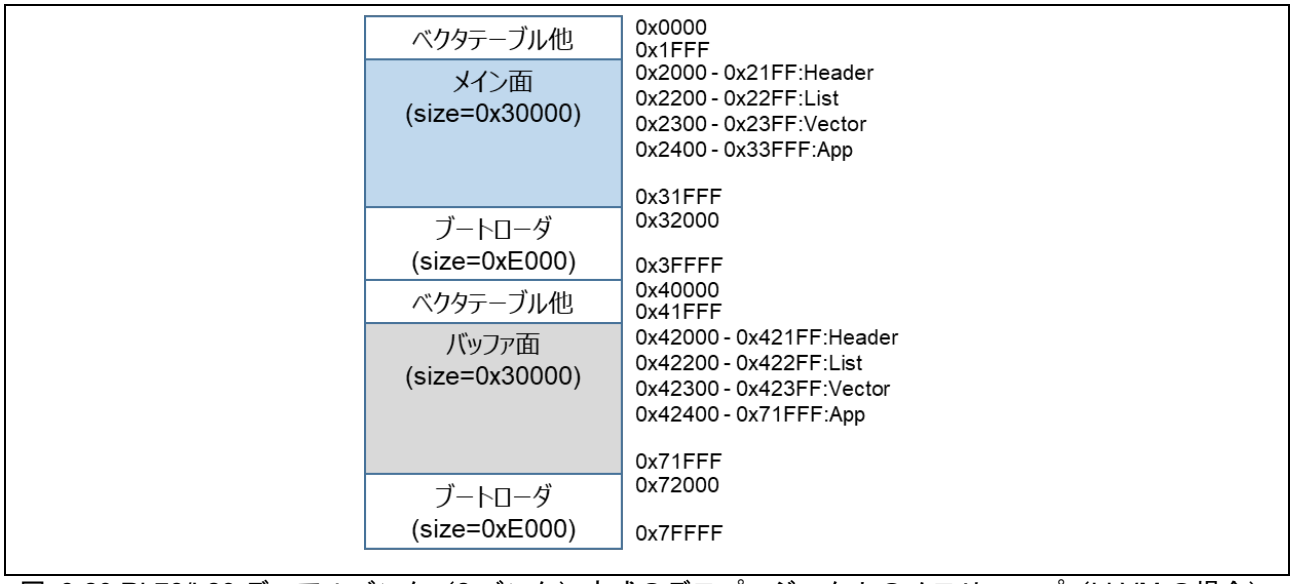


図 6-20 RL78/L23 デュアルバンク（2 バンク）方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-15 RL78/L23 のデュアルバンク（2 バンク）方式のコンフィグ設定（LLVM の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	0	0
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x42000	0x42000
FWUP_CFG_AREA_SIZE	0x30000	0x30000
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	1	1

6.2.4.2　半面更新方式のデモプロジェクトの各種情報

RL78/L23 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について、以下に示します。

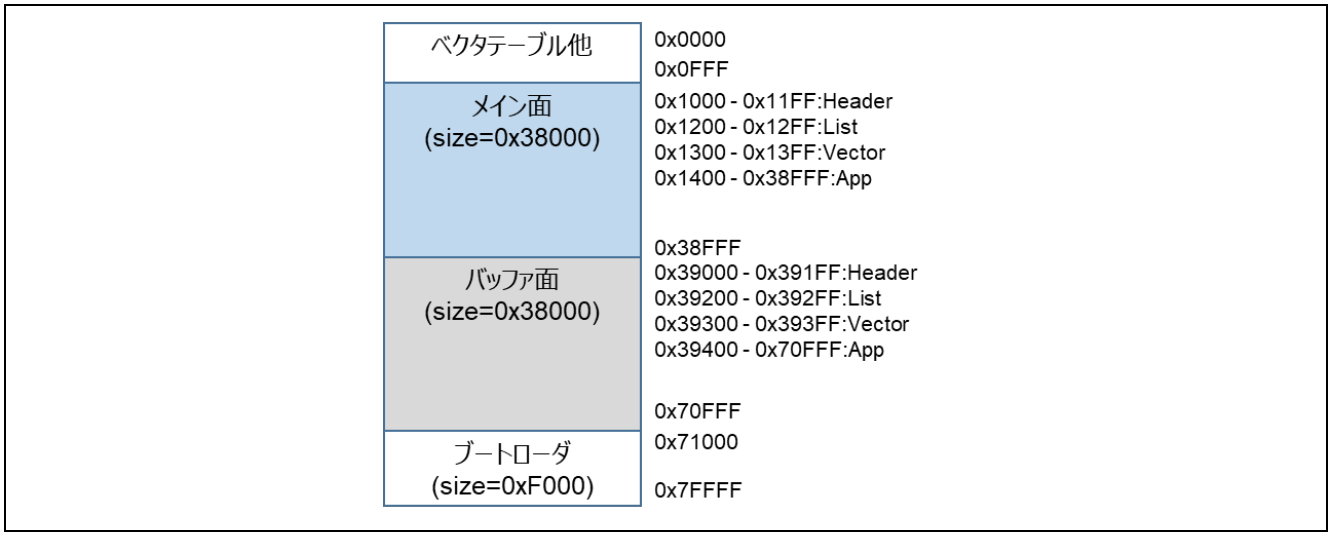


図 6-21 RL78/L23 の半面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-16 RL78/L23 の半面更新方式のコンフィグ設定（CC-RL,IAR の場合）

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	0x39000	0x39000
FWUP_CFG_AREA_SIZE	0x38000	0x38000
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	1	1

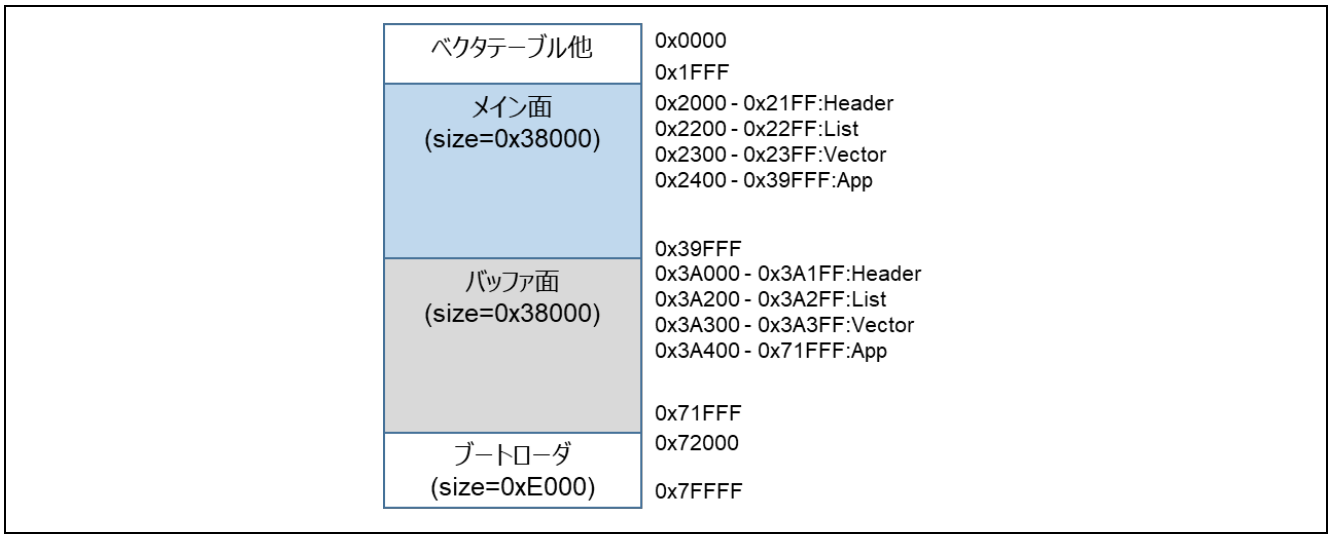


図 6-22 RL78/L23 の半面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-17 RL78/L23 の半面更新方式のコンフィグ設定（LLVM の場合）

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	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x3A000	0x3A000
FWUP_CFG_AREA_SIZE	0x38000	0x38000
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	1	1

6.2.4.3　全面更新方式のデモプロジェクトの各種情報

RL78/L23 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について、以下に示します。

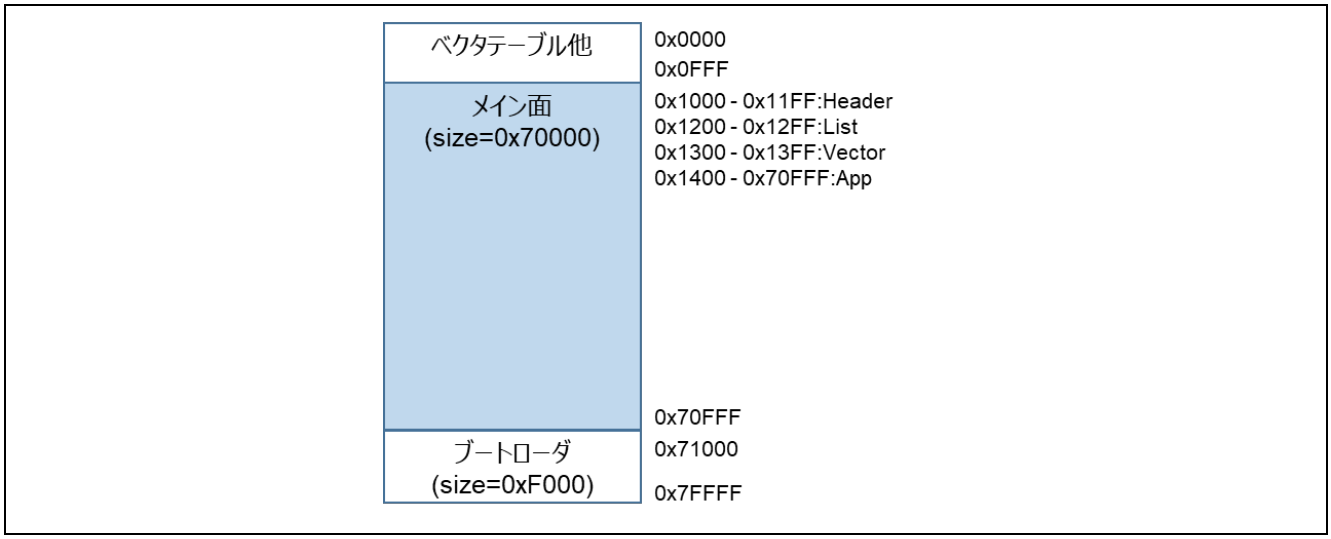


図 6-23 RL78/L23 の全面更新方式のデモプロジェクトのメモリマップ（CC-RL,IAR の場合）

表 6-18 RL78/L23 の全面更新方式のコンフィグ設定（CC-RL,IAR の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	2 or 3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_BUF_AREA_ADDR_L	0x1000	0x1000
FWUP_CFG_AREA_SIZE	0x70000	0x70000
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	1	1

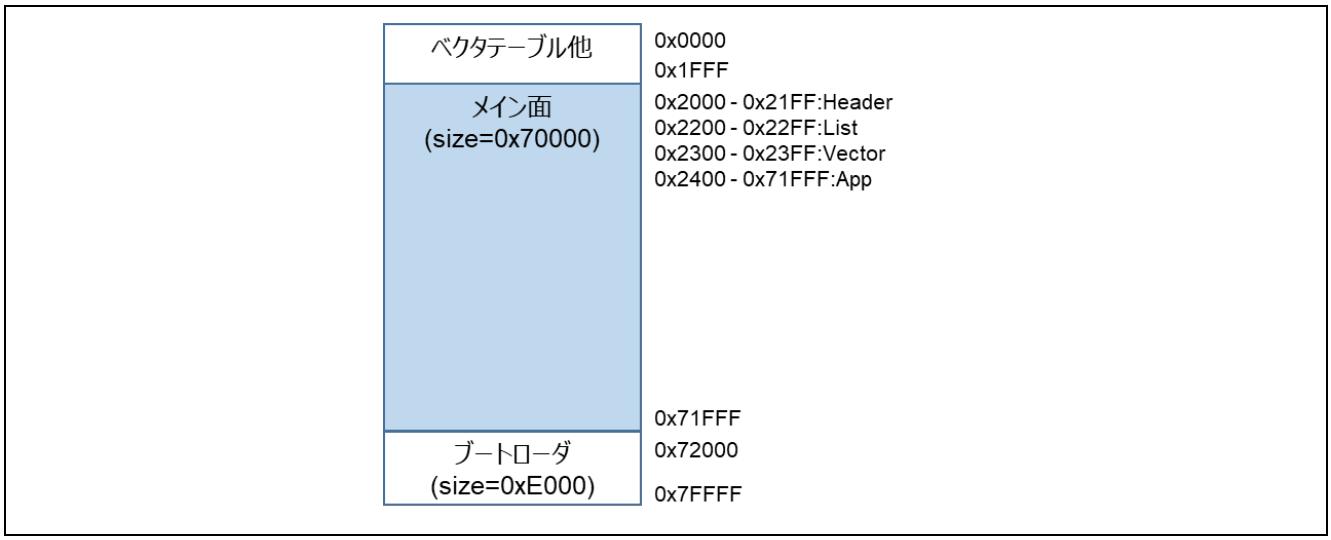


図 6-24 RL78/L23 の全面更新方式のデモプロジェクトのメモリマップ（LLVM の場合）

表 6-19 RL78/L23 の全面更新方式のコンフィグ設定（LLVM の場合）

Configuration options in r_fwup _config.h		
パラメータ名	boot_loader	fwup_main
FWUP_CFG_UPDATE_MODE	2 or 3	3
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0x2000	0x2000
FWUP_CFG_BUF_AREA_ADDR_L	0x2000	0x2000
FWUP_CFG_AREA_SIZE	0x70000	0x70000
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	1	1

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 TinyCrypt 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。
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	・RL78/G24 追加
		12-13	・フォルダ構成にデバイス追加
		17	・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
		19	・ROM/RAM/スタックにデバイス追加
		22	・R_FWUP_EraseArea 関数のパラメータ追加
		22	・R_FWUP_GetImageSize 関数の説明追加
		22	・R_FWUP_WriteImageHeader 関数の戻り値追加
		23	・R_FWUP_WriteImageProgram 関数のパラメータ追加
		23	・R_FWUP_WriteImage 関数の戻り値追加
		24	・R_FWUP_VerifyImage 関数の戻り値追加
		55	・動作確認環境の使用ボード追加
		56-65	・動作確認環境にデバイス追加
		67	・注意事項を追加
2.02	2024.12.13	9-12	・1.4 章を見直し
		21-22	・1.6 章の図を 2.10 章に移動
		27	・3.6 R_FWUP_WriteImageHeader 関数を 3.13 章に移動
		27	・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
		35-51	・4.6 デモプロジェクトのデバッグ方法 を追加
		52-60	・5. Renesas Image Generator を見直し
		61-69	・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
		82	
2.03	2025.4.18	1	・要旨 ターゲットコンパイラに LLVM を追加
		13	・1.5 パッケージ構成の表に LLVM の内容を追加
		17	・2.6 ログ表示設定のデフォルトを Disable に変更
		18	・2.7 デモプロジェクトのコードサイズに LLVM を追加
		21, 22	・2.10 ブートローダ実装例のフローを変更
		39-49	・4.4 LLVM の内容を追記
		72	・6.1 LLVM の動作確認環境を追加
		75, 76	・6.2.1.1 LLVM の図と表を追加
		77, 78	・6.2.1.2 LLVM の図と表を追加
2.04	2025.8.27	1	・動作確認デバイス RL78/L23 を追加
		1	・関連アプリノート Flash Driver RL78 Type11 を追加
		6, 9, 13, 14	・1.1, 1.3.1, 1.4.1, 1.4.4 デュアルバンク方式関連の記述を追加
		16, 17	・表 1-2 RL78/L23 FPB のフォルダを追加
		19	・2.2 ソフトウェア要求に Flash Driver RL78 Type11 を追加
		20	・表 2-1 アップデート方式にデュアルバンク方式を追加
		23	・2.7.4 RL78/L23 のコードサイズを追加
		25, 26	・2.10.1 デュアルバンク方式の実装例を追加
		38	・3.15.2.6 バンクスワップ処理の内容を追加
		47-51	・4.4 デュアルバンク方式のデモプロジェクト実行手順を追加
		78, 79	・5.2.2 デュアルバンク方式の初期イメージファイルを追加
		96-100	・6.2.4 RL78/L23 の動作確認環境を追加

2.05	2026.5.22	6 8 16, 17 50, 54, 58, 61 71–73 74–77 78–81 92 93–112	・1.1 アップデート方式の名称を一部変更 ・表 1-2 各製品のアップデート方式のサポート状況を追加 ・表 1-3 パッケージのフォルダ構成変更 ・4.4.5, 4.5.5, 4.6.5, 4.7.5 のコンパイラ毎のログ出力内容を統一 ・4.9 デモプロジェクトの更新イメージ取得経路の変更方法を追加 ・4.10 デモプロジェクトへの割り込み処理追加方法を追加 ・4.11 デモプロジェクトと異なるメモリ構成で使用する場合のイメージの生成方法を追加 ・6.1 動作確認環境に LLVM を追加 ・6.2 デモプロジェクトの動作確認環境に LLVM 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。