R01AN5824JJ0106
Rev.1.06
2022.12.28

RX ファミリ
ファームウェアアップデートモジュール Firmware Integration Technology

要旨

本アプリケーションノートは Firmware Integration Technology (FIT)を使った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 FIT モジュールと称します。

本アプリケーションノードは、「ルネサス MCU におけるファームウェアアップデートの設計方針(R01AN5548)」をベースに記載しておりますので、こちらのアプリケーションノートを先に読まれることをお勧めします。

本 FIT モジュールを使って、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 FIT モジュールの使用、およびユーザアプリケーションへの取り込みについて説明します。

動作確認デバイス

RX130 グループ

RX140 グループ

RX230、RX231、RX23E-A、RX23W グループ

RX65N、RX651 グループ

RX66N グループ

RX66T グループ

RX660 グループ

RX671 グループ

RX72M グループ

RX72N グループ

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。

- Renesas ルネサス MCU におけるファームウェアアップデートの設計方針(R01AN5548)
- RX ファミリ RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法(R01AN5549)
- Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
- RX ファミリ e2 studio に組み込む方法 Firmware Integration Technology(R01AN1723)
- RX スマート・コンフィグレータ ユーザーガイド: IAREW 編(R20AN0535)
- RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology(R01AN1685)
- RX ファミリ フラッシュモジュール Firmware Integration Technology(R01AN2184)

- RX ファミリ SCI モジュール Firmware Integration Technology(R01AN1815)
- RX ファミリ イーサネットモジュール Firmware Integration Technology(R01AN2009)
- RX ファミリ CMT モジュール Firmware Integration Technology(R01AN1856)
- RX ファミリ バイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology(R01AN1683)
- RX ファミリ システムタイマモジュール Firmware Integration Technology(R20AN0431)

ターゲットコンパイラ

- ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for RX

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「5.1 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とは

ファームウェアアップデートとは、機器の制御を行うファームウェアを新しいファームウェア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。

RX ファミリの場合、ファームウェアは MCU に内蔵されたフラッシュメモリに書き込まれています。したがって RX ファミリでは、この MCU 内蔵フラッシュメモリの内容を書き換える操作や処理をファームウェアアップデートと呼びます。

通常、MCU に内蔵されたフラッシュメモリの内容を書き換える方法には、以下の 2 種類があります。

- オフボードプログラミング
  外部にフラッシュライタ等のフラッシュ書き換え機器を接続しフラッシュメモリを書き換える方法
- オンボードプログラミング（セルフプログラミング）
  MCU を動作させ MCU 自身で内蔵フラッシュメモリを書き換える方法

ファームウェアアップデートは後者のセルフプログラミング機能を使い、MCU 自身が内蔵しているフラッシュメモリを書き換えます。

フラッシュメモリをセルフプログラミングで書き換える場合、RAM にフラッシュメモリを書き換えるプログラムを配置した後、RAM 上からフラッシュメモリを書き換えるコマンドを実行する必要があります。また、新しいバージョンのファームウェアを各種のインタフェースから取得する必要もあり、お客様のシステムにファームウェアアップデートの機能を構築することは非常に難しい物でした。

本ファームウェアアップデートの FIT モジュールを使うことで、ファームウェアアップデートの機能をお客様のシステムに容易に組み込むことができます。

ファームウェアアップデートモジュールは API として、プロジェクトに組み込んで使用します。本モジュールの組み込み方については、「2.10 FIT モジュールの追加方法」を参照してください。

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はファームウェアをアップデートすることを目的としたミドルウェアです。

ファームウェアアップデートモジュールには OS 無しのシステム向けの機能と、OS 無し(モジュール外通信制御)のシステム向けの機能、及び FreeRTOS（OTA）を用いたシステム向けの機能があります。FreeRTOS（OTA）の詳細については、以下のリンクを参照してください：
https://docs.aws.amazon.com/ja_jp/freertos/latest/userguide/freertos-ota-dev.html

OS 無しのシステムにおけるファームウェアアップデートのシステム構成を図 1-1 に、OS 無し(モジュール外通信制御)のシステムにおけるファームウェアアップデートのシステム構成を図 1-2 に、FreeRTOS（OTA）を用いたシステムにおけるファームウェアアップデートのシステム構成を図 1-3 に示します。

ブートローダモジュールとは、リセット後に最初に実行され、ユーザプログラム（ブートローダ実行後に実行するプログラム）が改ざんされていないことを検証するモジュールです。

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を行うモジュールでユーザプログラムに含まれます。

ファームウェアアップデートで使用する FIT モジュールの一覧を表 1-1 に示します。

各種通信 I/F で取得したアップデート用のファームウェアは、ファームウェアアップデートモジュールとフラッシュ FIT モジュールを介してターゲットデバイス上のコードフラッシュメモリにプログラムされます。

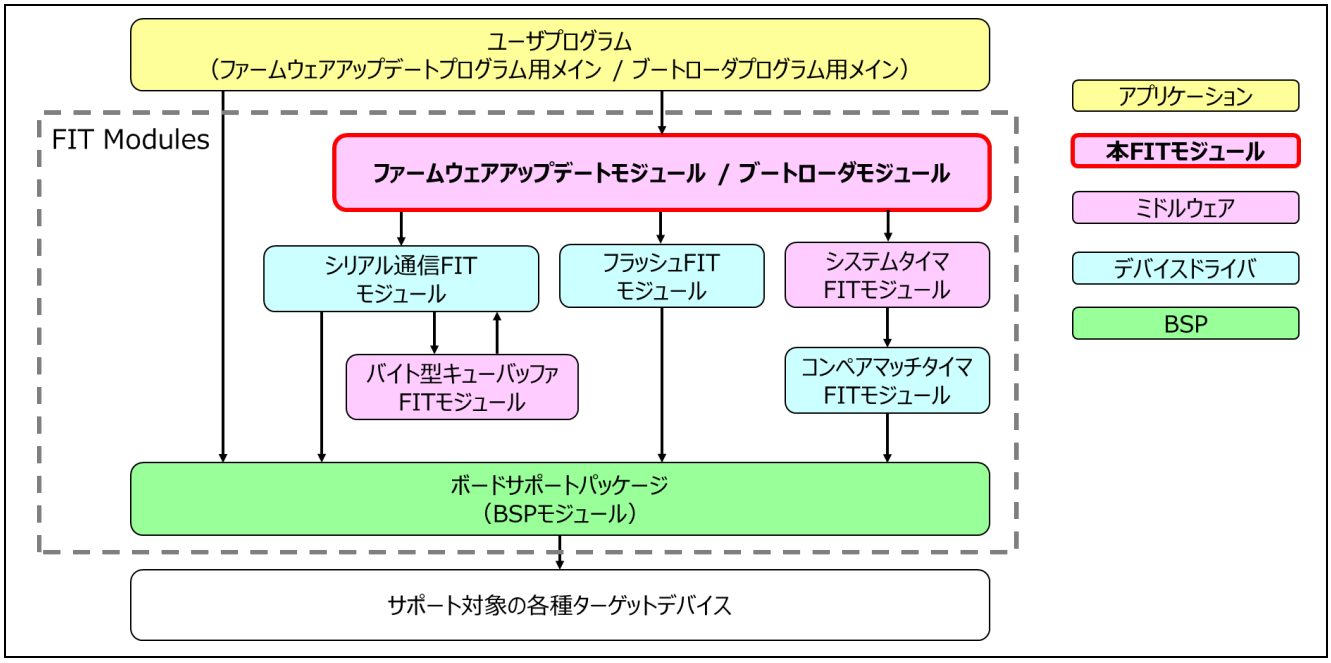


図 1-1 OS 無しのシステムにおけるファームウェアアップデートのシステム構成

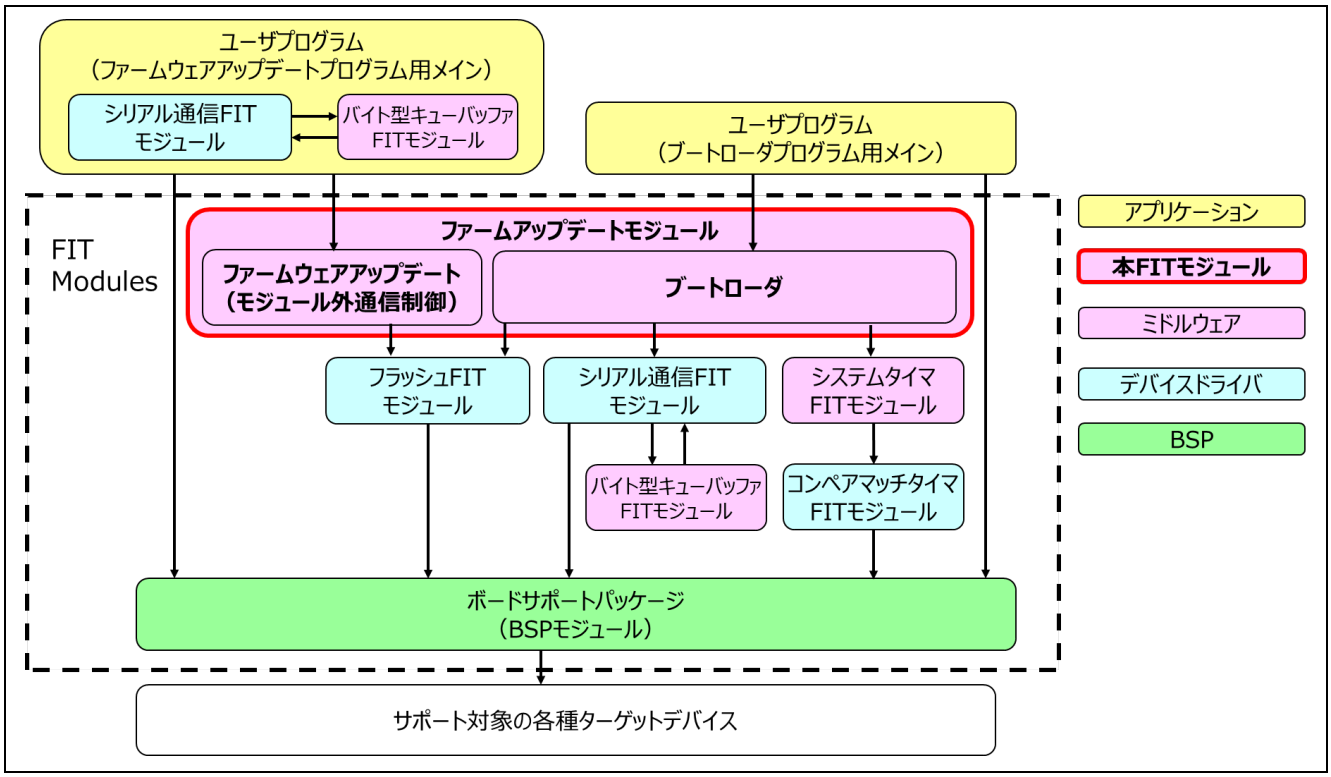


図 1-2 OS 無し(モジュール外通信制御)のシステムにおけるファームウェアアップデートのシステム構成

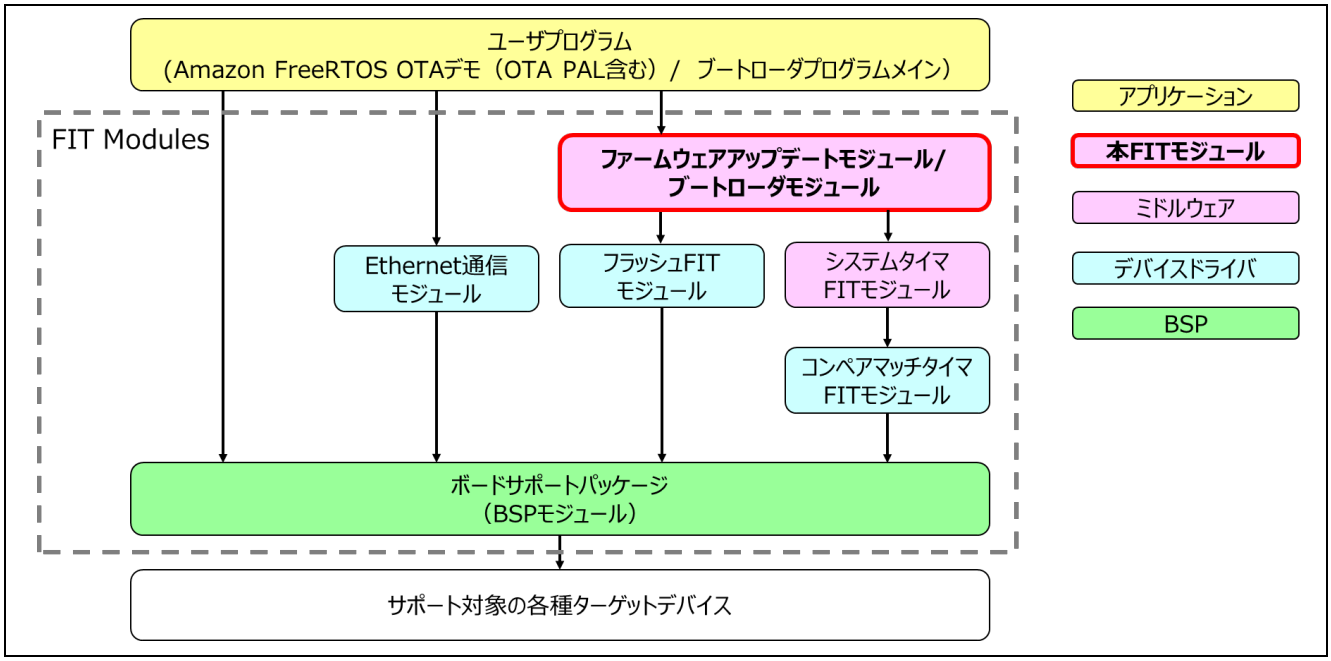


図 1-3 FreeRTOS（OTA）を用いたシステムにおけるファームウェアアップデートのシステム構成

表 1-1 モジュール一覧

種類	アプリケーションノート名（型名）	FIT モジュール名
BSP	RX ファミリボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)	r_bsp
デバイスドライバ	RX ファミリフラッシュモジュール Firmware Integration Technology (R01AN2184)	r_flash_rx
デバイスドライバ	RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)	r_sci_rx
デバイスドライバ	RX ファミリ CMT モジュール Firmware Integration Technology (R01AN1856)	r_cmt_rx
ミドルウェア	RX ファミリバイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology (R01AN1683)	r_byteq
ミドルウェア	RX ファミリ システムタイマモジュー Firmware Integration Technology (R20AN0431)	r_sys_time_rx

1.3　ファームウェアアップデートの動作

RX ファミリには、MCU 内蔵のフラッシュメモリにデュアルバンク機能を持つ製品があります。

デュアルバンク機能を持たない製品もしくは、デュアルバンク機能でリニアモードを使用する場合、フラッシュメモリの書き換えを行うためには、RAM にフラッシュメモリを書き換えるプログラムを配置した後、RAM 上からフラッシュメモリを書き換えるコマンドを実行する必要があります。

デュアルバンク機能でデュアルモードを使用する場合、フラッシュメモリの書き換え領域と実行領域が違う領域にある場合、RAM でプログラムを実行して書き換える必要がありません。このため、システム動作を維持したままフラッシュメモリを書き換えることが非常に容易となります。

本ファームウェアアップデートモジュールは、リニアモードでもデュアルモードでも、ファームウェアアップデートが可能です。

表 1-2 デバイス毎のリニアモード／デュアルモードサポート状況

デバイス	リニアモード	デュアルモード
RX130 グループ	○	—
RX140 グループ	○	—
RX231 グループ	○	—
RX65N グループ	−	○
RX66T グループ	○	—
RX660 グループ	○	—
RX671 グループ	−	○
RX72N グループ	−	○

1.3.1　デュアルモードを使用したファームウェアアップデート動作

フラッシュメモリのデュアルモードを使用したファームウェアアップデートの動作を説明します。

ファームウェアアップデートの動作は、ファームウェアアップデートの準備処理としての内蔵フラッシュの初期設定と、ファームウェアアップデート動作に分けられます。

デュアルモードのファームウェアアップデートの初期設定の動作を示します。

本 FIT モジュールでは、内蔵フラッシュに書き込む初期ファームウェアを生成するためのツール（Renesas Image Generator）を提供しています。このツールでは、ブートローダのみとブートローダとユーザプログラムで構成される 2 種類の初期ファームウェアを生成することができます。

この 2 種類の初期ファームウェアをフラッシュライタ等で書き込むことで、図 1-4 の①または④の状態にすることができます。どちらから開始するかはお客様のシステムに合わせて選択してください。初期ファームウェアの詳細に関しては、5.4 と 5.5 を参照してください。

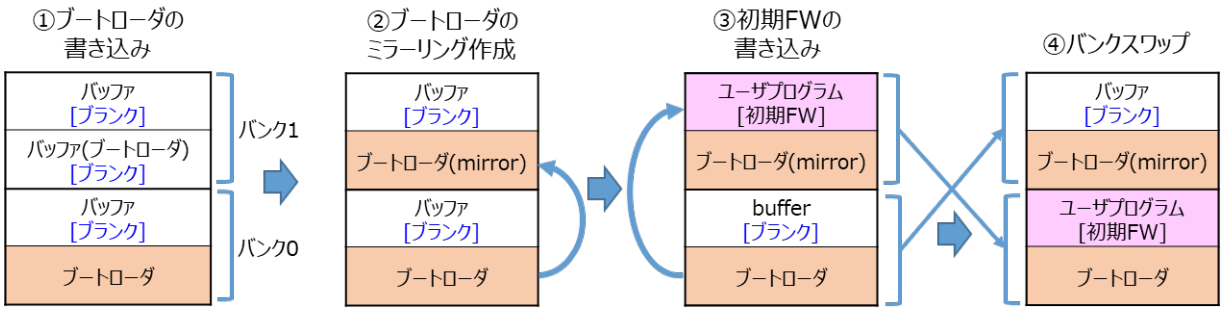


図 1-4 デュアルモードのファームウェアアップデートの初期設定

初期設定を①から開始する場合

① ブートローダをフラッシュライタ等でフラッシュメモリに書き込む。

② ブートローダを実行し、バンク 1 にブートローダをミラーリングする。

③ ブートローダにての初期ファームウェアを書き込み（外部から入力する必要あり）、書き込んだファームウェアを検証する。

④ 検証に問題なければ、バンクスワップを行う。

初期設定を④から開始する場合

④ ブートローダとユーザプログラムで構成される初期ファームウェアを、フラッシュライタ等でフラッシュメモリに書き込む。

デュアルモードのファームウェアアップデートの動作を示します（ただし下記「①初期状態」は初回起動時にブートローダを実行し、バンク１にブートローダをミラーリングした状態）。

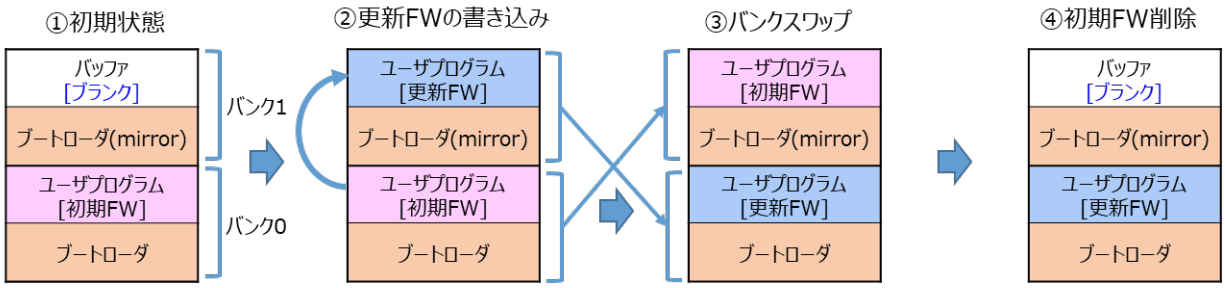


図 1-5 デュアルモードのファームウェアアップデートの動作

① 初期状態。

② ユーザプログラムに含まれるファームウェアアップデートモジュールで更新ファームウェアを書き込み（外部から入力する必要あり）、書き込んだファームウェアを検証する。

③ 検証に問題なければ、バンクスワップを行う。

④ バンク１の初期ファームウェアを削除する。

1.3.2　リニアモードを使用したファームウェアアップデート動作

フラッシュメモリのリニアモードを使用したファームウェアアップデートとして半面書き換え方式と全面書き換え方式の 2 種類の方式が有り、ここではそれらの動作を説明します。

注）全面書き換え方式は、書き換え中の電源遮断等によるアップデート失敗時に、他の 2 方式（デュアルモード及びリニアモードの半面書き換え方式）とは異なり、旧バージョンでの復旧が行えない仕様となります。

1.3.2.1　半面書き換え方式のファームウェアアップデート

半面書き換え方式の初期設定の動作を示します。

本 FIT モジュールでは、内蔵フラッシュに書き込む初期ファームウェアを生成するためのツール（Renesas Image Generator）を提供しています。このツールでは、ブートローダのみとブートローダとユーザプログラムで構成される 2 種類の初期ファームウェアを生成することができます。

この 2 種類の初期ファームウェアをフラッシュライタ等で書き込むことで、図 1-6 の①または②の状態にすることができます。どちらから開始するかはお客様のシステムに合わせて選択してください。初期ファームウェアの詳細に関しては、5.4 と 5.5 を参照してください。

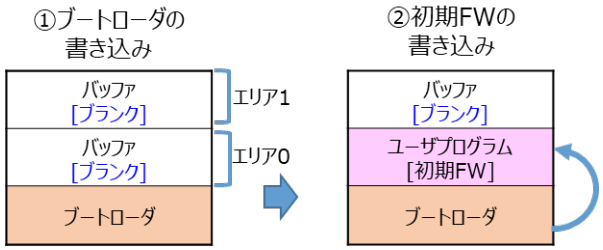


図 1-6 半面書き換え方式のファームウェアアップデートの初期設定

初期設定を①から開始する場合

① ブートローダをフラッシュライタ等でフラッシュメモリに書き込む。

② ブートローダにて初期ファームウェアを書き込み（外部から入力する必要あり）、書き込んだファームウェアを検証し、検証に問題なければ終了する。

初期設定を②から開始する場合

② ブートローダとユーザプログラムで構成される初期ファームウェアを、フラッシュライタ等でフラッシュメモリに書き込む。

半面書き換え方式のファームウェアアップデートの動作を示します。

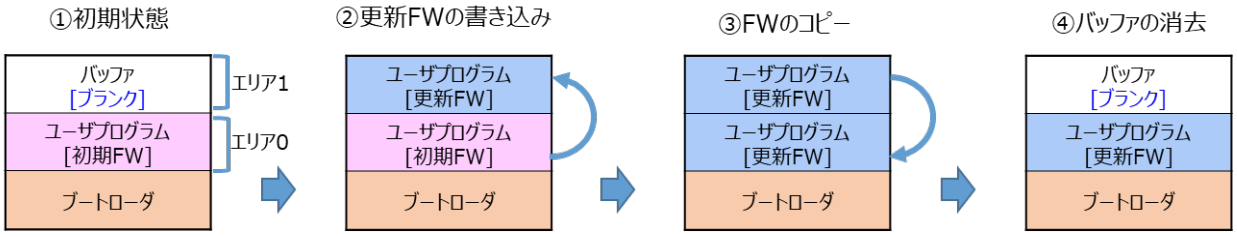


図 1-7 半面書き換え方式のファームウェアアップデートの動作

① 初期状態。

② エリア 0 のユーザプログラムにて更新ファームウェアをエリア 1 のバッファ領域に書き込み（外部から入力する必要あり）、書き込んだファームウェアを検証する。

③ 検証に問題なければ、エリア 1 のバッファ領域からエリア 0 のユーザプログラム領域にコピーする。

④ エリア 1 のバッファ領域を削除する。

1.3.2.2 全面書き換え方式のファームウェアアップデート

全面書き換え方式の初期設定の動作を図 1-8 に示します。

本 FIT モジュールでは、内蔵フラッシュに書き込む初期ファームウェアを生成するためのツール（Renesas Image Generator）を提供しています。このツールでは、ブートローダのみとブートローダとユーザプログラムで構成される 2 種類の初期ファームウェアを生成することができます。

この 2 種類の初期ファームウェアをフラッシュライタ等で書き込むことで、図 1-8 の①または②の状態にすることができます。どちらから開始するかはお客様のシステムに合わせて選択してください。初期ファームウェアの詳細に関しては、5.4 と 5.5 を参照してください。

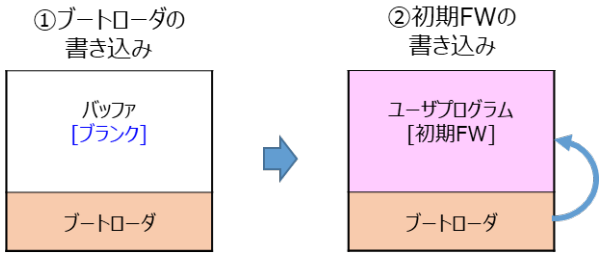


図 1-8 全面書き換え方式のファームウェアアップデートの初期設定

初期設定を①から開始する場合

① ブートローダをフラッシュライタ等でフラッシュメモリに書き込む。

② ブートローダにて初期ファームウェアを書き込み（外部から入力する必要あり）、書き込んだファームウェアを検証し、検証に問題なければ終了する。

初期設定を②から開始する場合

② ブートローダとユーザプログラムで構成される初期ファームウェアを、フラッシュライタ等でフラッシュメモリに書き込む。

全面書き換え方式のファームウェアアップデートの動作を図 1-9 に示します。

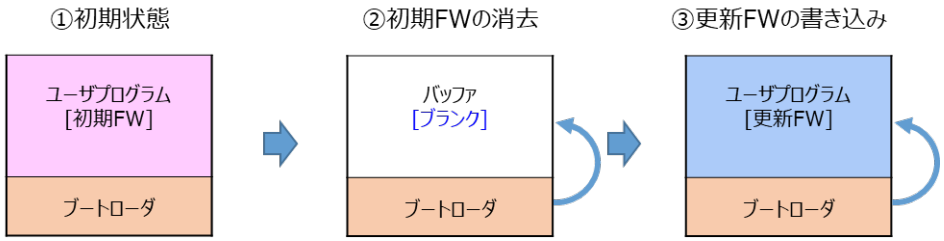


図 1-9 全面書き換え方式のファームウェアアップデートの動作

① 初期状態。

② ユーザプログラムにて初期ファームウェアを使用できない状態に設定しリセットを行う。その後ブートローダが立上り、初期ファームウェアを消去する。

③ ブートローダにて、更新ファームウェアを書き込み（外部から入力する必要あり）、書き込んだファームウェアを検証し問題無ければ、更新されたユーザプログラムを起動する。

1.4　OS 無しのシステムにおけるファームウェアアップデートの通信制御

1.4.1　モジュール内通信制御のファームウェアアップデート

ファームウェアの受信処理から署名検証までをファームウェアアップデートモジュールの API(R_FWUP_Operation 関数)内で実施するシステムです。

ユーザプログラムに含まれるファームウェアアップデートの概要を以下に示します。

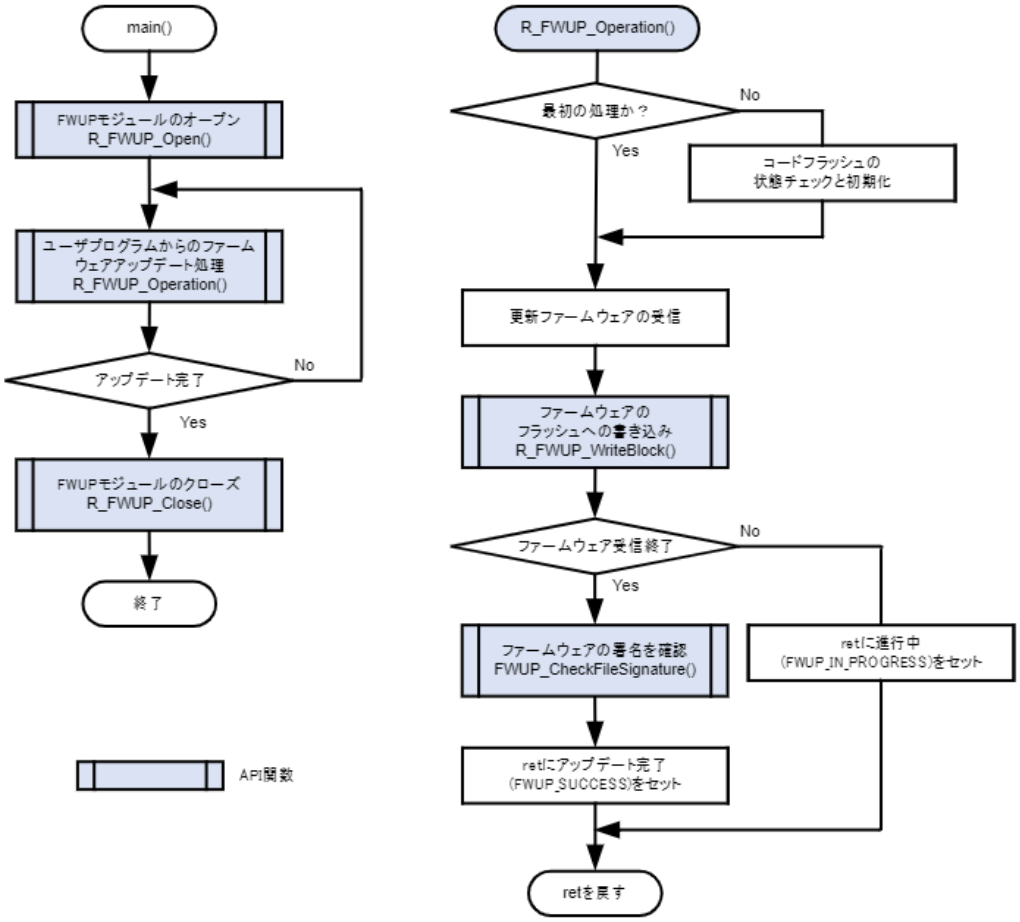


図 1-10 OS 無しのシステムにおけるファームウェアアップデートの動作

ファームウェアアップデートモジュールのオープン(R_FWUP_Open)、ファームウェアアップデート処理(R_FWUP_Operation)、クローズ(R_FWUP_Close)、で構成されます。

R_FWUP_Operation 関数内で通信制御の他、フラッシュの初期化、フラッシュへの書き込み、ファームウェアの署名の確認など、ファームウェアアップデートの一連の処理を行います。

1.4.2　モジュール外通信制御のファームウェアアップデート

ファームウェアの受信処理はユーザプログラムで行い、ファームウェアアップデートモジュールの API で受信済みのデータをフラッシュに書き込むシステムです。

OS 無し(モジュール外通信制御)のシステムにおけるユーザプログラムに含まれるファームウェアアップデートの概要を以下に示します。

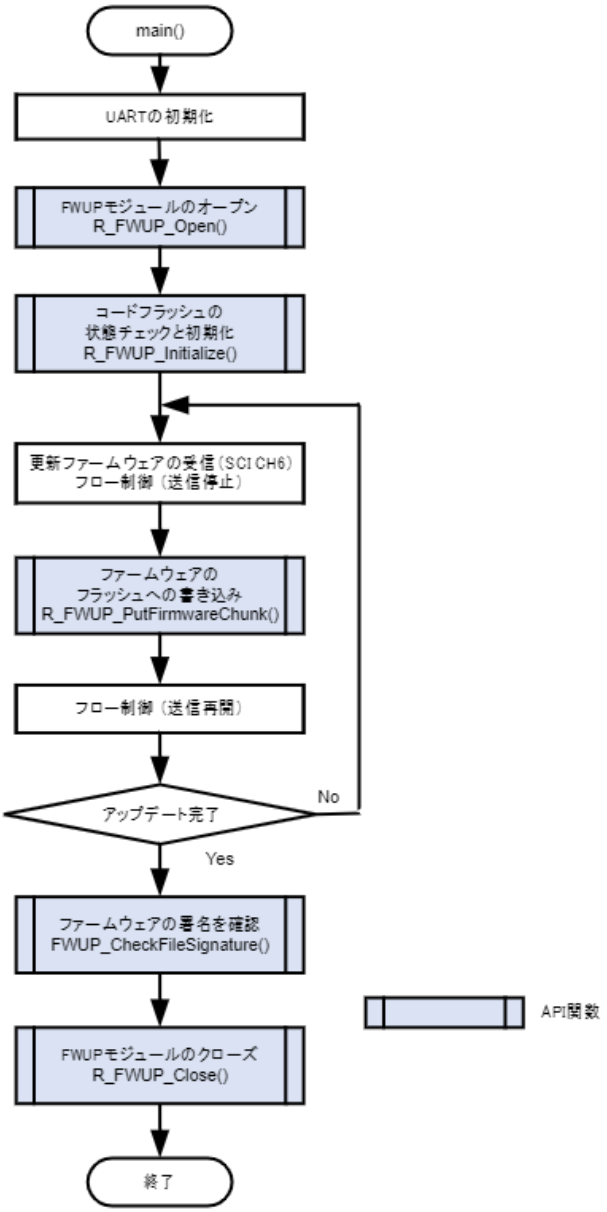


図 1-11 OS 無し(モジュール外通信制御)のシステムにおけるファームウェアアップデートの動作

OS 無し(モジュール外通信制御)のシステムでは、ファームウェアアップデートモジュールのオープン(R_FWUP_Open)、コードフラッシュの状態チェックと初期化(R_FWUP_Initialize)、フラッシュへの書き込み(R_FWUP_PutFirmwareChunk)、クローズ(R_FWUP_Close)、ファームウェアの署名の確認(R_FWUP_CheckFileSigunature)で構成されます。

通信の初期化やファームウェアの受信(含むフロー制御)など、ユーザプログラム側で通信を制御する必要があります。

フロー制御については、デモプロジェクト(fwup_main_owSciDrv)の以下を参照ください。

フロー制御（送信停止）は UART 受信用 callback 関数(uart_receive_fileblock)内で行います。

```
static void uart_receive_fileblock(void *pArgs)
{
    sci_cb_args_t * p_args;

    p_args = (sci_cb_args_t *)pArgs;

    switch (p_args->event)
    {
        case SCI_EVT_RX_CHAR:

            /* From RXI interrupt; received character data is in p_args->byte */
            if ((s_sci_receive_control_block.p_sci_buffer_control->buffer_occupied_byte_size <
                  (sizeof(s_sci_receive_control_block.p_sci_buffer_control->buffer))) &&
                (USR_SCI_RECEIVE_BUFFER_EMPTY == s_sci_receive_control_block.p_sci_buffer_control->buffer_full_flag))
            {
                R_SCI_Receive(p_args->hdl, &s_sci_receive_control_block.p_sci_buffer_control->
                                        buffer[s_sci_receive_control_block.p_sci_buffer_control->
                                        buffer_occupied_byte_size++ ],
                                        1);
                if ((sizeof(s_sci_receive_control_block.p_sci_buffer_control->buffer)) == s_sci_receive_control_block.p_sci_buffer_control->buffer_occupied_byte_size)
                {
                /* RTS HIGH */
                    USR_CFG_CSI_PORT_SYMBOL.PODR.BIT.USR_CFG_CSI_BIT_SYMBOL = 1;  /* Set RTS to HIGH */
                    USR_CFG_CSI_PORT_SYMBOL.PDR.BIT.USR_CFG_CSI_BIT_SYMBOL  = 1;
                    USR_CFG_CSI_PORT_SYMBOL.PMR.BIT.USR_CFG_CSI_BIT_SYMBOL  = 0;  /* Change to general I/O port */

                    s_sci_receive_control_block.total_byte_size +=
                    s_sci_receive_control_block.p_sci_buffer_control->buffer_occupied_byte_size;
                    s_sci_receive_control_block.p_sci_buffer_control->buffer_occupied_byte_size = 0;
                    s_sci_receive_control_block.p_sci_buffer_control->buffer_full_flag          = USR_SCI_RECEIVE_BUFFER_FULL;
```

図 1-12 フロー制御（送信停止）

フラッシュ書き込み(R_FWUP_PutFirmwareChunk)の直前で受信完了待ち(R_BSP_SoftwareDelay)を行います。待機時間は r_fwup_config.h で定義される FWUP_CFG_SCI_RECEIVE_WAIT を用います。

フロー制御（送信再開）はフラッシュ書き込み(R_FWUP_PutFirmwareChunk)の直後に行います。

```
    R_BSP_SoftwareDelay(FWUP_CFG_SCI_RECEIVE_WAIT, BSP_DELAY_MICROSECS);  /* SCI receive wait */

    result_fwup = R_FWUP_PutFirmwareChunk (s_sci_receive_control_block.offset,
                                           firm_data, FWUP_WRITE_BLOCK_SIZE);

    USR_CFG_CSI_PORT_SYMBOL.PMR.BIT.USR_CFG_CSI_BIT_SYMBOL = 1;  /* Return port to UART flow control */
    printf("Flash Write: Address = 0x%X, length = %dbyte\n",
            TEMPORARY_AREA_LOW_ADDRESS + s_sci_receive_control_block.offset, FWUP_WRITE_BLOCK_SIZE);
    s_sci_receive_control_block.offset += FWUP_WRITE_BLOCK_SIZE;
```

図 1-13 フロー制御（送信再開）

1.5　API の概要

表 **1-3** にファームウェアアップデートモジュールに含まれる API 関数を示します。

表 1-3 API 関数一覧

関数	関数説明	ファームウェアアップデートモジュール				ブートローダプログラム
		OS あり		OS レス		
		Free RTOS (OTA)	Azure (ADU)	モジュール外通信制御 (1)	モジュール内通信制御	
R_FWUP_Open	本モジュールのオープン処理行います。	—	○	○	○	○
R_FWUP_Close	本モジュールのクローズ処理を行います。	—	○	○	○	○
R_FWUP_Initialize	コードフラッシュの状態チェック・消去を行います。	—	○	○	—	—
R_FWUP_Operation	ユーザプログラムからのファームウェアアップデート処理を行います。	—	—	—	○	—
R_FWUP_PutFirmwareChunk	指定したオフセットでデータブロックを書き込みます。	—	○	○	—	—
R_FWUP_SoftwareReset	ソフトウェアリセットを行います。	—	○	○	○	—
R_FWUP_DirectUpdate (2)	バッファ領域を使用せずに、ファームウェアアップデートを行います。	—	—	—	○	—
R_FWUP_SetEndOfLife	ユーザプログラムの保守期間終了時 EOL(END OF LIFE)の処理を行います。	○	○	○	○	—
R_FWUP_SecureBoot	ブートローダでのセキュアブート処理を行います。	—	—	—	—	○
R_FWUP_ExecuteFirmware	インストール、もしくはアップデートしたファームウェアへ処理を移します。	—	—	—	—	○
R_FWUP_Abort	OTA 更新処理を中止します。	○	—	—	—	—
R_FWUP_CreateFileForRx	OTA の初期設定を行います。	○	—	—	—	—
R_FWUP_CloseFile	指定したファイルをクローズします。	○	—	—	—	—
R_FWUP_WriteBlock	指定したファイルに、指定したオフセットでデータブロックを書き込みます。	○	—	—	—	—
R_FWUP_ActivateNewImage	新しいファームウェアイメージをアクティブ化または起動します。	○	—	—	—	—
R_FWUP_ResetDevice	ソフトウェアリセットが発生し、以降はブートローダの処理を経て新しいファームウェアを起動します。	○	—	—	—	—

R_FWUP_SetPlatformImageState	ライフサイクルの状態を、引数で指定された状態に設定します。	○	—	—	—	—
R_FWUP_GetPlatformImageState	現在のライフサイクルの状態を返します。	○	—	—	—	—
R_FWUP_CheckFileSignature [OS 無し時に使用]	フラッシュに書き込んだファームウェアの署名を確認します。	—	—	○	—	—
R_FWUP_CheckFileSignature [OTA 時に使用]	フラッシュに書き込んだファームウェアの署名を確認します。	○	—	—	—	—
R_FWUP_ReadAndAssumeCertificate	指定された署名者証明書をファイルシステムから読み込み返します。	○	—	—	—	—
R_FWUP_GetVersion	本モジュールのバージョン番号を返します。	○	○	○	○	○

(1) モジュール外通信制御は、RX66T で半面書き換え方式(CC-RX)のみ動作確認を行っています。
(2) R_FWUP_DirectUpdate 関数は、RX66T で全面書き換え方式(CC-RX)のみ動作確認を行っています。

2. API 情報

本 FIT 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

ご使用になる 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
- シリアルコミュニケーションインタフェース：オプション
- イーサネット：オプション
- システムタイマモジュール

2.2 ソフトウェアの要求

本 FIT モジュールは以下のパッケージに依存しています。

- ボードサポートパッケージ（r_bsp）
- バイト型キューバッファモジュール（r_byteq）
- コンペアマッチタイマ（r_cmt_rx）
- フラッシュモジュール（r_flash_rx）
- シリアルコミュニケーション
  インタフェース（SCI：調歩同期式/クロック同期式）（r_sci_rx）：オプション
- イーサネットモジュール（r_ether_rx）：オプション
- システムタイマモジュール（r_sys_time_rx）

2.3 サポートされているツールチェーン

本 FIT モジュールは「5.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

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 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_IMPLEMENTATION_ENVIRONMENT ※デフォルト値は“0”	本 FIT モジュールを実装するユーザプログラムの環境をコンフィグ設定します。 実装対象により、使用可能な API 関数が変わります。 以下のいずれかの値を設定する 0：ブートローダプログラムに実装する（デフォルト） 1：ユーザプログラムのファームウェアアッププログラム（OS 無しのシステム）に実装する 2：ユーザプログラムのファームウェアアッププログラム（OS 無し,モジュール外通信制御 のシステム）に実装する 3：FreeRTOS（OTA）プログラムに実装する 4：Azure（ADU）プログラムに実装する 実装環境を追加する場合は、設定値を追加する。
FWUP_CFG_COMMUNICATION_FUNCTION ※デフォルト値は“0”	ユーザプログラムのファームウェアアップデート時に、新しいバージョンのファームウェアを取得する通信経路をコンフィグ設定します。 以下のいずれかの値を設定する 0：SCI 通信を接続（デフォルト） 1：Ethernet 通信を接続 2：USB を接続[注 1] 3：SDHI を接続[注 1] 4：QSPI を接続[注 1] 通信経路を追加する場合は、設定値を追加する。
FWUP_CFG_BOOT_PROTECT_ENABLE ※デフォルト値は“0”	ブート保護設定の有効/無効を切り替えます。 0：ブート保護設定無効（デフォルト） 1：ブート保護設定有効 * *）ブートローダを格納する領域を書き換えられなくするための機能です。 一度ブート保護設定を行うと、環境によっては二度と“ブート保護設定無効”には戻せなくなり、アクセス可能領域やスタートアップ領域保護機能の再設定ができなくなります。 ブート保護設定の取り扱いには十分にご注意ください。
FWUP_CFG_OTA_DATA_STORAGE ※デフォルト値は“0”	FreeRTOS（OTA）用データの格納先を設定します。 FreeRTOS の OTA 実行時に、本設定が有効になります。 また、ブートプログラムと FreeRTOS（OTA）プログラムにて設定値を同一にしてください。 0：データフラッシュ（デフォルト） 1：コードフラッシュ
FWUP_CFG_NO_USE_BUFFER ※デフォルト値は“0”	ファームウェアアップデートの書き換え方式を設定します。 0：半面書き換え方式（デフォルト） 1：全面書き換え方式

FWUP_CFG_BOOTLOADER_LOG_DISABLE ※デフォルト値は“0”	ROM サイズ縮小のため、ターミナルソフトへの printf 文での文字列表示を抑制します。 FWUP_CFG_IMPLEMENTATION_ENVIRONMENT を“0”に設定した場合に、本設定が有効になります。 0：ターミナルソフトへ文字列を表示する（デフォルト） 1：ターミナルソフトへ文字列を表示しない
FWUP_CFG_LOG_LEVEL ※デフォルト値は“3”	LOG 出力のレベルを設定します。 FWUP_CFG_IMPLEMENTATION_ENVIRONMENT を“1”に設定した場合に、本設定が有効になります。 0：LOG 出力無し 1：エラーメッセージのみ出力 2：ワーニング および エラーメッセージを出力 3：インフォメーション、ワーニング および エラーメッセージを出力（デフォルト） 4：全ての LOG を出力
FWUP_CFG_SERIAL_TERM_SCI ※デフォルト値は“8”	ファームウェアのダウンロードに使用する SCI チャンネルを設定します。
FWUP_CFG_SERIAL_TERM_SCI_BITRATE ※デフォルト値は“115200”	ファームウェアのダウンロードに使用する UART のボーレート値を設定します。
FWUP_CFG_SERIAL_TERM_SCI_INTERRUPT_PRIORITY ※デフォルト値は“15”	ファームウェアのダウンロードに使用する SCI の割り込み優先レベルを設定します。
FWUP_CFG_SCI_RECEIVE_WAIT ※デフォルト値は“300”	送信停止(RTS を HIGH に設定)後の UART 受信待機時間を指定します。マイクロ秒単位で設定します。
FWUP_CFG_PORT_SYMBOL ※デフォルト値は RSK-RX231 用の“PORTC”	UART の受信要求端子である RTS に使用される I/O ポートのポートシンボルを設定します。
FWUP_CFG_BIT_SYMBOL ※デフォルト値は RSK-RX231 用の“B4”	UART の受信要求端子である RTS に使用される I/O ポートのビットシンボルを設定します。

注１）設定しても動作しない未対応の項目

コンフィグレーションオプション設定の“FWUP_CFG_IMPLEMENTATION_ENVIRONMENT”と“FWUP_CFG_COMMUNICATION_FUNCTION”には設定可能/不可能な組み合わせがあり、設定可能な組み合わせは以下のとおり。

（表中：”数字”＝FWUP_ENV_COMMUNICATION_FUNCTION へ設定値、“—”＝無効な組み合わせ）

表 2-2 コンパイル時設定の設定可能組み合わせ

		FWUP_CFG_COMMUNICATION_FUNCTION				
		0 : SCI	1 : Ethernet	2 : USB	3 : SDHI	4 : QSPI
FWUP_CFG_IMPLEMENTATION_ENVIRONMENT	0 : ブートローダプログラム	0	—	—	—	—
	1 : ユーザプログラムのファームウェアアップデートプログラム（OS 無のシステム） 2 : ユーザプログラムのファームウェアアップデートプログラム（OS 無，モジュール外通信制御のシステム）	1	—	2 注 1	—	3 注 1
	3 : FreeRTOS（OTA）プログラム	4 注 1	5	6 注 1	7 注 1	—
	4 : Azure（ADU）プログラム	—	8	—	—	—

注 1）設定しても動作しない未対応の組み合わせ

実装環境設定と通信経路設定の組み合わせで有効となった組み合わせ条件を、r_fwup_private.h 内にマクロで保持する。

表 2-3 有効組み合わせのマクロ値

マクロ名	値	説明
FWUP_COMM_SCI_BOOTLOADER	0	SCI に PC(COM ポート)を接続し、ブートローダ処理を行う
FWUP_COMM_SCI_PRIMITIVE	1	SCI に PC(COM ポート)を接続し、ターミナルソフト経由で新しいバージョンのファームウェアを取得する
FWUP_COMM_USB_PRIMITIVE	2	USB に PC(COM ポート)を接続し、ターミナルソフト経由で新しいバージョンのファームウェアを取得する
FWUP_COMM_QSPI_PRIMITIVE	3	QSPI に外部ストレージ(SD カード)を接続し、新しいバージョンのファームウェアを取得する
FWUP_COMM_SCI_AFRTOS	4	SCI に無線モジュール(SX-ULPGN, BG96 等)を接続し、FreeRTOS(OTA)で新しいバージョンのファームウェアを取得する
FWUP_COMM_ETHER_AFRTOS	5	Ethernet で接続し、FreeRTOS(OTA)で新しいバージョンのファームウェアを取得する
FWUP_COMM_USB_AFRTOS	6	USB に LTE モデムを接続し、FreeRTOS(OTA)で新しいバージョンのファームウェアを取得する
FWUP_COMM_SDHI_AFRTOS	7	SDHI に無線モジュール(Type 1DX 等)を接続し、FreeRTOS(OTA)で新しいバージョンのファームウェアを取得する
FWUP_COMM_ETHER_AZURE	8	Ethernet (Wi-fi 接続を含む) で接続し、Azure(ADU)で新しいバージョンのファームウェアを取得する

実装環境設定と通信経路設定の組み合わせ設定が追加となった際には、合わせて本マクロに設定を追加します。

ex.)

```
#define FWUP_COMM_SCI_BOOTLOADER    0    // Used for Bootloader with SCI connection from COM port.
#define FWUP_COMM_SCI_PRIMITIVE     1    // SCI connection from COM port using primitive R/W.
#define FWUP_COMM_USB_PRIMITIVE     2    // USB connection from COM port using primitive R/W.
#define FWUP_COMM_QSP_PRIMITIVE     3    // Connect external storage (SD card) to QSPI using primitive R/W.
#define FWUP_COMM_SCI_AFRTOS        4    // Connect wireless module to SCI with Amazon FreeRTOS.
#define FWUP_COMM_ETHER_AFRTOS      5    // Connect Eathernet with Amazon FreeRTOS.
#define FWUP_COMM_USB_AFRTOS        6    // Connect LTE modem to USB with Amazon FreeRTOS.
#define FWUP_COMM_SDHI_AFRTOS       7    // Connect wireless module to SDHI with Amazon FreeRTOS.
#define FWUP_COMM_ETHER_AZURE       8    // Connect Eathernet with Azure ADU.
```

2.6.1　RX130/RX140 環境でのコンパイル時の注意事項

RSK RX130/RX140 で本 FIT を使用する場合は、ボードサポートパッケージ(BSP)のコンフィグレーションオプションで設定されるユーザスタックのサイズ(BSP_CFG_USTACK_BYTES)を、デフォルトから 0x1000(4KB)に増やしてください。

2.7　コードサイズ

本 FIT モジュールのコードサイズを下表に示します。

フラッシュタイプ[注]毎に代表して 1 デバイスずつ掲載しています。

表 2-4 コードサイズ

ROM、RAM およびスタックのコードサイズ					
デバイス	分類	使用メモリ			備考
		C/C++ Compiler Package for RX Family	GCC for Renesas RX	IAR C/C++ Compiler for RX	
RX65N (フラッシュタイプ 4)	ROM	3,213 バイト	3,220 バイト	3,052 バイト	boot_loader Project
		4,560 バイト	3,891 バイト	3,243 バイト	fwup_main Project
		4,560 バイト	3,891 バイト	1,579 バイト	eol_main Project
		5,342 バイト	4,251 バイト	—	aws_demos Project
	RAM	36,968 バイト	36,957 バイト	36,946 バイト	boot_loader Project
		3,261 バイト	3,257 バイト	3,246 バイト	fwup_main Project
		3,261 バイト	3,257 バイト	2,222 バイト	eol_main Project
		1,504 バイト	1,500 バイト	—	aws_demos Project
	最大使用スタックサイズ	1,184 バイト	836 バイト	1,527 バイト	boot_loader Project
		2,712 バイト	2,356 バイト	2,480 バイト	fwup_main Project
		2,712 バイト	2,356 バイト	1,284 バイト	eol_main Project
		1,880 バイト	1,736 バイト	—	aws_demos Project
RX66T (フラッシュタイプ 3)	ROM	4,448 バイト	3,476 バイト	3,274 バイト	boot_loader Project
		4,194 バイト	3,573 バイト	3,132 バイト	fwup_main Project
		4,192 バイト	3,573 バイト	1,528 バイト	eol_main Project
		2,999 バイト	—	—	fwup_main woSciDrv Project
	RAM	36,969 バイト	36,957 バイト	36,946 バイト	boot_loader Project
		3,257 バイト	3,253 バイト	3,242 バイト	fwup_main Project
		3,257 バイト	3,253 バイト	2,218 バイト	eol_main Project
		3,257 バイト	—	—	fwup_main woSciDrv Project
	最大使用スタックサイズ	1,412 バイト	864 バイト	1,560 バイト	boot_loader Project
		2,672 バイト	2,332 バイト	2,472 バイト	fwup_main Project
		2,668 バイト	2,332 バイト	1,276 バイト	eol_main Project
RX231 (フラッシュタイプ 1)	ROM	1,088 バイト	—	—	fwup_main woSciDrv Project
		4,329 バイト	3,393 バイト	3,266 バイト	boot_loader Project
		4,335 バイト	3,686 バイト	3,134 バイト	fwup_main Project
	RAM	4,335 バイト	3,686 バイト	1,529 バイト	eol_main Project
		6,249 バイト	6,237 バイト	6,226 バイト	boot_loader Project
		3,257 バイト	3,253 バイト	3,242 バイト	fwup_main Project
	最大使用スタックサイズ	3,257 バイト	3,253 バイト	2,218 バイト	eol_main Project
		1,412 バイト	856 バイト	1,556 バイト	boot_loader Project
		2,668 バイト	2,344 バイト	2,468 バイト	fwup_main Project

注）フラッシュタイプの詳細に関しては、アプリケーションノート「RX ファミリ フラッシュモジュール Firmware Integration Technology(R01AN2184)」を参照してください。

＜条件＞

C/C++ Compiler Package for RX Family

- ・最適化レベル　：レベル 2
- ・リンク時にモジュール間最適化の対象にする　：チェック有り
- ・最適化方法　：コードサイズ重視の最適化を実施する
- ・一度も参照のない変数/関数を削除する　：チェック無し
- ・FWUP_CFG_BOOTLOADER_LOG_DISABLE（Config）　：1

GCC for Renesas RX

- ・最適化レベル　：Optimize size (-Os)
- ・デバッグレベル　：None
- ・リンクオプション　：-Wl,--no-gc-sections
- ・FWUP_CFG_BOOTLOADER_LOG_DISABLE（Config）　：1

IAR C/C++ Compiler for RX

- ・最適化レベル　：高い (サイズ)
- ・FWUP_CFG_BOOTLOADER_LOG_DISABLE（Config）　：1

【参考】ブートローダの使用 ROM、RAM サイズ

ブートローダプロジェクトが使用する ROM、RAM のサイズを、参考として製品ごとに以下に示します。

表 2-5 ブートローダの使用 ROM、RAM サイズ

ブートローダの使用 ROM、RAM サイズ				
デバイス	分類	使用メモリ		
		C/C++ Compiler Package for RX Family	GCC for Renesas RX	IAR C/C++ Compiler for RX
RX130	ROM	36,240 バイト	52,092 バイト	25,298 バイト
	RAM	11,304 バイト	11,140 バイト	12,457 バイト
RX140	ROM	31,167 バイト	48,288 バイト	23,319 バイト
	RAM	11,803 バイト	11,684 バイト	15,183 バイト
RX231	ROM	35,516 バイト	50,324 バイト	24,747 バイト
	RAM	12,388 バイト	12,164 バイト	15,516 バイト
RX671	ROM	36,838 バイト	55,541 バイト	30,466 バイト
	RAM	41,230 バイト	40,868 バイト	45,724 バイト
RX65N	ROM	30,370 バイト	54,762 バイト	27,097 バイト
	RAM	41,186 バイト	43,388 バイト	43,660 バイト
RX66T	ROM	37,310 バイト	53,036 バイト	25,377 バイト
	RAM	43,628 バイト	43,268 バイト	45,403 バイト
RX660	ROM	38,840 バイト	51,932 バイト	24,722 バイト
	RAM	42,935 バイト	42,660 バイト	44,944 バイト
RX72N	ROM	39,296 バイト	55,818 バイト	30,730 バイト
	RAM	41,350 バイト	40,996 バイト	45,828 バイト

＜条件＞

C/C++ Compiler Package for RX Family

- 最適化レベル　：レベル 2
- リンク時にモジュール間最適化の対象にする　：チェック有り
- 最適化方法　：コードサイズ重視の最適化を実施する
- 一度も参照のない変数/関数を削除する　：チェック無し
- 入出力関数　：簡易版
- FWUP_CFG_BOOTLOADER_LOG_DISABLE（Config）　：1

GCC for Renesas RX

- 最適化レベル　：Optimize size (-Os)
- デバッグレベル　：None
- リンクオプション　：-Wl,--no-gc-sections
- FWUP_CFG_BOOTLOADER_LOG_DISABLE（Config）　：1

IAR C/C++ Compiler for RX

- 最適化レベル　：高い (サイズ)
- FWUP_CFG_BOOTLOADER_LOG_DISABLE（Config）　：1

2.8　引数

API 関数の引数にある構造体は Amazon FreeRTOS(OTA) 202002.00 の環境でのファイルコンテキスト設定を、その他の環境でも流用して使用しています。

流用し使用している構造体は以下の通りです。

【注意】

Amazon FreeRTOS(OTA)の当該設定がバージョンアップなどで変更される可能性があります。バージョンアップされた場合に設定値が変更になっていないか確認が必要です。

OTA 環境での宣言箇所：aws_demos¥libraries¥ota_for_aws¥source¥include¥ota_private.h

表 2-6 OTA のファイルコンテキスト

```
typedef struct
{
    uint16_t size;                          /*!< @brief Size, in bytes, of the signature. */
    uint8_t data[ kOTA_MaxSignatureSize ]; /*!< @brief The binary signature data. */
} Sig256_t;

typedef struct OtaFileContext
{
    uint8_t * pFilePath;          /*!< @brief Update file pathname. */
    uint16_t filePathMaxSize;     /*!< @brief Maximum size of the update file path */
    #if defined( WIN32 ) || defined( __linux__ )
        FILE * pFile;             /*!< @brief File type is stdio FILE structure after file is open for write. */
    #else
        uint8_t * pFile;          /*!< @brief File type is RAM/Flash image pointer after file is open for write. */
    #endif
    uint32_t fileSize;            /*!< @brief The size of the file in bytes. */
    uint32_t blocksRemaining;     /*!< @brief How many blocks remain to be received (a code optimization). */
    uint32_t fileAttributes;      /*!< @brief Flags specific to the file being received (e.g. secure, bundle, archive).
*/
    uint32_t serverFileID;        /*!< @brief The file is referenced by this numeric ID in the OTA job. */
    uint8_t * pJobName;           /*!< @brief The job name associated with this file from the job service. */
    uint16_t jobNameMaxSize;      /*!< @brief Maximum size of the job name. */
    uint8_t * pStreamName;        /*!< @brief The stream associated with this file from the OTA service. */
    uint16_t streamNameMaxSize;   /*!< @brief Maximum size of the stream name. */
    uint8_t * pRxBlockBitmap;     /*!< @brief Bitmap of blocks received (for deduplicating and missing block request).
*/
    uint16_t blockBitmapMaxSize;  /*!< @brief Maximum size of the block bitmap. */
    uint8_t * pCertFilepath;      /*!< @brief Pathname of the certificate file used to validate the receive file. */
    uint16_t certFilePathMaxSize; /*!< @brief Maximum certificate path size. */
    uint8_t * pUpdateUrlPath;     /*!< @brief Url for the file. */
    uint16_t updateUrlMaxSize;    /*!< @brief Maximum size of the url. */
    uint8_t * pAuthScheme;        /*!< @brief Authorization scheme. */
    uint16_t authSchemeMaxSize;   /*!< @brief Maximum size of the auth scheme. */
    uint32_t updaterVersion;      /*!< @brief Used by OTA self-test detection, the version of Firmware that did the
update. */
    bool isInSelfTest;            /*!< @brief True if the job is in self test mode. */
    uint8_t * pProtocols;         /*!< @brief Authorization scheme. */
    uint16_t protocolMaxSize;     /*!< @brief Maximum size of the  supported protocols string. */
    uint8_t * pDecodeMem;         /*!< @brief Decode memory. */
    uint32_t decodeMemMaxSize;    /*!< @brief Maximum size of the decode memory. */
    uint32_t fileType;            /*!< @brief The file type id set when creating the OTA job. */
    Sig256_t * pSignature;        /*!< @brief Pointer to the file's signature structure. */
} OtaFileContext_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 に記載されています。

表 2-7 API の戻り値設定

```
typedef enum e_fwup_err
{
    FWUP_SUCCESS = 0,               // Normally terminated.
    FWUP_FAIL,                      // Illegal terminated.
    FWUP_IN_PROGRESS,               // Firmware update is in progress.
    FWUP_END_OF_LIFE,               // End Of Life process finished.
    FWUP_ERR_ALREADY_OPEN,          // Firmware Update module is in use by another process.
    FWUP_ERR_NOT_OPEN,              // R_FWUP_Open function is not executed yet.
    FWUP_ERR_IMAGE_STATE,           // Platform image status not suitable for firmware update.
    FWUP_ERR_LESS_MEMORY,           // Out of memory.
    FWUP_ERR_FLASH,                 // Detect error of r_flash module.
    FWUP_ERR_COMM,                  // Detect error of communication module.
    FWUP_ERR_STATE_MONITORING,      // Detect error of state monitoring module.
} fwup_err_t;
```

2.10 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。

e² studio の環境では、スマート・コンフィグレータを使用した(1)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)の方法を使用してください。

IAR Embedded Workbench for Renesas RX (IAREW) の環境では(3)の追加方法を使用してください。

(1)　e² studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合

e² studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: e² studio 編 (R20AN0451)」を参照してください。

(2)　e² studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合

e² studio の FIT コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3)　IAR Embedded Workbench for Renesas RX の環境でスマート・コンフィグレータを使用して FIT モジュールを追加する場合

IAR Embedded Workbench for Renesas RX の環境で FIT モジュールを追加する場合は、RX スマート・コンフィグレータを使用して、ユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

(4)　不要なモジュールを削除する

スマート・コンフィグレータを使用して、ユーザプロジェクトにファームウェアアップデートの FIT モジュールを追加する場合、依存関係のあるモジュールも同時に追加されます。
OS 無しモジュール外通信制御のファームウェアアップデートモジュールでは、r_sys_time_rx, r_sci_rx などファームウェアアップデートモジュール内で未使用でも、ユーザアプリケーションで使用されると思われるモジュールも追加されます。必要に応じて不要モジュールを削除してください。

2.11 システムタイマによる状態遷移監視の注意事項

本モジュールでは、システムタイマによる状態遷移監視を行っており、状態が設定時間以上遷移しない場合はエラー終了する仕様となっています。default 値は 1 分です。状態が設定時間以上固定されない様に御注意ください。

3. API 関数

3.1 R_FWUP_Open 関数

表 3-1 R_FWUP_Open 関数仕様

Format	fwup_err_t R_FWUP_Open (void)	
Description	ファームウェアアップデートモジュールやブートローダモジュールのオープン処理をおこないます。 ファームウェアアップデートモジュールやブートローダモジュールで使用する資源のオープン処理、OS の初期設定（OS 使用時）、各変数の初期化処理を行います。	
Parameters	なし	
Return Values	FWUP_SUCCESS	：正常終了
	FWUP_ERR_ALREADY_OPEN	：既にオープン済み
	FWUP_ERR_FLASH	：FLASH モジュールのエラー
	FWUP_ERR_COMM	：通信モジュールのエラー
	FWUP_ERR_STATE_MONITORING	：状態遷移監視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	fwup_err_t R_FWUP_Close (void)	
Description	ファームウェアアップデートモジュールやブートローダモジュールのクローズ処理をおこないます。 ファームウェアアップデートモジュールやブートローダモジュールで使用した資源のクローズ処理、OS の終了設定（OS 使用時）を行います。	
Parameters	なし	
Return Values	FWUP_SUCCESS	：正常終了
	FWUP_ERR_NOT_OPEN	：オープンしていない
Special Notes	ー	

3.3　R_FWUP_Initialize 関数

表 3-3 R_FWUP_Initialize 関数仕様

Format	fwup_err_t R_FWUP_Initialize (void)	
Description	ユーザプログラムからのファームウェアアップデート前のフラッシュ状態チェック、フラッシュ消去を行います。 ● 更新対象のフラッシュの状態が “VALID” または “INITIAL_FIRM_ INSTALLING” 以外の場合は ファームウェアを更新できる状態ではないため、戻り値として”FWUP_ERR_IMAGE_STATE” を返します。 ● 戻り値が “FWUP_IN_PROGRESS” の場合は正常終了です。R_FWUP_PutFirmwareChunk()関数を使用して、ファームウェアアップデート処理を継続してください。	
Parameters	なし	
Return Values	FWUP_IN_PROGRESS	: 正常終了 ファームウェアアップデートを継続
	FWUP_ERR_IMAGE_STATE	: フラッシュのステータスがアップデートできる状態でない
	FWUP_ERR_FLASH	: FLASH モジュールのエラー
Special Notes	ー	

3.4　R_FWUP_Operation 関数

表 3-4 R_FWUP_Operation 関数仕様

Format	fwup_err_t R_FWUP_Operation (void)	
Description	ユーザプログラムからのファームウェアアップデート処理を行います。 コンフィグ設定で指定された通信経路からアップデート用ファームウェアデータを取り出し、フラッシュへプログラム、署名検証の実施を行います。 ● 更新対象のフラッシュの状態が “VALID” または “INITIAL_FIRM_ INSTALLING” 以外の場合は ファームウェアを更新できる状態ではないため、戻り値として”FWUP_ERR_IMAGE_STATE” を返します。 ● 戻り値が “FWUP_IN_PROGRESS” の場合はファームウェアアップデート継続中のため、再度本関数をコールしてください。 ● 戻り値が “FWUP_SUCCESS” の場合はファームウェアアップデート完了です。R_FWUP_SoftwareReset 関数をコールしてください。ソフトウェアリセットを行うことで更新したファームウェアに処理を移します。 ● 戻り値が “FWUP_FAIL” の場合はファームウェアアップデートが失敗を意味します。エラーを解除し再度本関数をコールしてください。	
Parameters	なし	
Return Values	FWUP_SUCCESS	: ファームウェアアップデート正常終了
	FWUP_FAIL	: ファームウェアアップデートエラー発生
	FWUP_IN_PROGRESS	: ファームウェアアップデート継続中
	FWUP_ERR_NOT_OPEN	: オープンしていない
	FWUP_ERR_IMAGE_STATE	: フラッシュのステータスがアップデートできる状態でない
	FWUP_ERR_FLASH	: FLASH モジュールのエラー
	FWUP_ERR_STATE_MONITORING	: ファームウェアアップデートの状態が一定時間以上変わっていない
Special Notes	ー	

3.5　R_FWUP_PutFirmwareChunk 関数

表 3-5 R_FWUP_PutFirmwareChunk 関数仕様

Format	fwup_err_t R_FWUP_PutFirmwareChunk (uint32_t ulOffset, uint8_t * const pData, uint32_t ulBlockSize)
Description	指定されたオフセットでデータブロックを書き込みます。 成功した場合、FWUP_IN_PROGRESS を返します。
Parameters	ulOffset　：コードフラッシュの書き込み先オフセット * pData　：書き込むデータ ulBlockSize　：書き込むデータサイズ
Return Values	FWUP_IN_PROGRESS　：正常終了。
	FWUP_FAIL　：コードフラッシュへの書き込みエラー
Special Notes	OS レス環境で使用する場合は、パラメータの ulOffset と ulBlockSize には対象となるフラッシュメモリ上の領域の最小プログラムサイズの倍数を指定する必要があります。 また、ulBlockSize の最大値は 1024 です。

3.6　R_FWUP_SoftwareReset 関数

表 3-6 R_FWUP_SoftwareReset 関数仕様

Format	void R_FWUP_SoftwareReset (void)
Description	ソフトウェアリセットを行います。
Parameters	なし
Return Values	なし
Special Notes	ー

3.7　R_FWUP_DirectUpdate 関数

表 3-7 R_FWUP_DirectUpdate 関数仕様

Format	fwup_err_t R_FWUP_DirectUpdate (void)
Description	ファームウェア更新処理の開始。 実行中のファームウェアの状態を”BLANK”に設定し、リセットを発生させます。 リセット解除後、ブートローダで現在の実行ファームウェアの消去、 更新ファームウェアのアップデート処理が実行されます。
Parameters	なし
Return Values	FWUP_FAIL　：　ファームウェアアップデートエラー発生。
Special Notes	「全面書き換え方式」専用の関数。 正常終了時は関数内でリセットを実行し、戻り値を返しません。 エラー発生時のみ FWUP_FAIL を返します。

3.8　R_FWUP_SetEndOfLife 関数

表 3-8 R_FWUP_SetEndOfLife 関数仕様

書式	fwup_err_t R_FWUP_SetEndOfLife (void)	
Description	ユーザプログラムの保守期間終了時（END OF LIFE）の処理を行います。 [注意] 本関数コールし正常終了（FWUP_SUCCESS）した状態は、END OF LIFE(EOL)の処理は未完了の状態です。 END OF LIFE(EOL)の処理を完了させるためには本関数実施後に R_FWUP_SoftwareReset 関数をコールしソフトウェアリセット（デュアルバンクモードの場合はバンクスワップを伴うソフトウェアリセット）を発生させ、ブートローダで残りの END OF LIFE 処理を実施する必要があります。	
Parameters	なし	
Return Values	FWUP_SUCCESS	：正常終了
	FWUP_ERR_NOT_OPEN	：オープンしていない
	FWUP_ERR_FLASH	：FLASH モジュールのエラー
Special Notes	ー	

3.9　R_FWUP_SecureBoot 関数

表 3-9 R_FWUP_SecureBoot 関数仕様

Format	int32_t R_FWUP_SecureBoot (void)	
Description	ブートローダでのセキュアブート処理を行う。 ● インストール済みのファームウェアを起動前に、改ざんチェックのため署名検証します。 ● ファームウェアがインストールされていない場合、コンフィグ設定で指定された通信経路からインストール用ファームウェアデータを取り出し、フラッシュへプログラム、署名検証の実施を行います。 ● ユーザプログラムでアップデート用ファームウェアが設定された場合、起動用ファームウェアに置き換えます。 ● ユーザプログラムで保守期間終了時処理（EOL 処理）が設定された場合、本関数でファームウェアを削除します。 ● 戻り値が“FWUP_IN_PROGRESS”の場合はセキュアブート継続中のため、再度本関数をコールしてください。 ● 戻り値が“FWUP_SUCCESS”の場合はセキュアブート完了です。R_FWUP_ExecuteFirmware 関数をコールしインストールし、更新したファームウェアへ処理を移してください。 ● 戻り値が“FWUP_END_OF_LIFE”の場合はユーザプログラムの保守期間終了時（END OF LIFE）の処理が完了です。 ● 戻り値が“FWUP_FAIL”の場合はセキュアブートの失敗を意味します。必要であればエラーを解除し再度本関数をコールしてください。 「全面書き換え方式」向け ● 「全面書き換え方式」を実行した場合、取得したファームウェアは直接コードフラッシュの実行領域に書き込まれます。コードフラッシュへの書き込み及び署名検証で Fail した場合はコードフラッシュを消去して FWUP_FAIL を返します。（次回起動時は、ブートローダから初期ファームウェアを要求します）	
Parameters	なし	
Return Values	FWUP_SUCCESS	: セキュアブート正常終了
	FWUP_FAIL	: セキュアブートエラー発生
	FWUP_IN_PROGRESS	: セキュアブート継続中
	FWUP_END_OF_LIFE	: END OF LIFE(EOL)処理完了
	FWUP_ERR_NOT_OPEN	: オープンしていない
	FWUP_ERR_STATE_MONITORING	: ファームウェアアップデートの状態が一定時間以上変わっていない
Special Notes	ー	

3.10 R_FWUP_ExecuteFirmware 関数

表 3-10 R_FWUP_ExecuteFirmware 関数仕様

Format	void R_FWUP_ExecuteFirmware (void)
Description	インストール、もしくはアップデートしたファームウェアへ処理を移します。 [注意] 処理を移行するファームウェアの起動アドレスは、MCU のファミリやシリーズにより異なる場合があります。 実装する環境に応じてファームウェアの起動アドレスを取得する処理を実装する必要があります。 [例：RX65N の場合] マクロ USER_RESET_VECTOR_ADDRESS に設定されているアドレスに処理を移す
Parameters	なし
Return Values	なし
Special Notes	ー

3.11 R_FWUP_Abort 関数

表 3-11 R_FWUP_Abort 関数仕様

Format	OtaPalStatus_t R_FWUP_Abort (OTA_FileContext_t * const C)	
Description	OTA 更新処理を中止します。	
Parameters	* C	：ファイルコンテキスト
Return Values	OtaPalSuccess	：正常終了
	OtaPalFileClose	：ファイルコンテキストのクローズエラー
Special Notes	ー	

3.12 R_FWUP_CreateFileForRx 関数

表 3-12 R_FWUP_CreateFileForRx 関数仕様

Format	OtaPalStatus_t R_FWUP_CreateFileForRx (OTA_FileContext_t * const C)	
Description	OTA の初期設定を行います。 受信したデータを保存するファイルを作成します。	
Parameters	* C	：ファイルコンテキスト
Return Values	OtaPalSuccess	：正常終了
	OtaPalRxFileCreateFailed	：ファイル作成エラー
Special Notes	ー	

3.13 R_FWUP_CloseFile 関数

表 3-13 R_FWUP_CloseFile 関数仕様

Format	OtaPalStatus_t R_FWUP_CloseFile (OTA_FileContext_t * const C)
Description	指定されたファイルをクローズします。 テンポラリエリアのバッファ領域にダウンロードしたファームウェアイメージに対する署名検証を行います。 テンポラリエリアのバッファ領域のヘッダ情報を書き込みます。
Parameters	* C　：ファイルコンテキスト
Return Values	OtaPalSuccess　：正常終了
	OtaPalFileClose　：ファイルのクローズエラー
	OtaPalSignatureCheckFailed　：署名検証エラー
Special Notes	ー

3.14 R_FWUP_WriteBlock 関数

表 3-14 R_FWUP_WriteBlock 関数仕様

Format	int16_t R_FWUP_WriteBlock (OTA_FileContext_t * const C, uint32_t ulOffset, uint8_t * const pacData, uint32_t ulBlockSize)
Description	指定されたファイルに、指定されたオフセットでデータブロックを書き込みます。 成功した場合、書き込まれたバイト数を返します。
Parameters	* C　：ファイルコンテキスト ulOffset　：コードフラッシュの書き込み先オフセット * pacData　：書き込むデータ ulBlockSize　：書き込むデータサイズ
Return Values	R_OTA_ERR_QUEUE_SEND_FAIL (-2)　：コードフラッシュへの書き込みエラー
	上記以外　：コードフラッシュに書き込んだバイト数
Special Notes	ー

3.15 R_FWUP_ActivateNewImage 関数

表 3-15 R_FWUP_ActivateNewImage 関数仕様

Format	OtaPalStatus_t R_FWUP_ActivateNewImage (void)
Description	新しいファームウェアイメージをアクティブ化または起動します。 R_FWUP_ResetDevice()関数をコールし、ソフトウェアリセットします。
Parameters	なし
Return Values	OtaPalSuccess　：正常終了
Special Notes	ー

3.16 R_FWUP_ResetDevice 関数

表 3-16 R_FWUP_ResetDevice 関数仕様

Format	OtaPalStatus_t R_FWUP_ResetDevice (void)
Description	本関数コールでソフトウェアリセットが発生し、以降はブートローダの処理を経て新しいファームウェアを起動します。
Parameters	なし
Return Values	OtaPalSuccess　：正常終了
Special Notes	オープンしている周辺回路をすべてクローズしてから本関数を実行してください。 本関数は関数内でソフトウェアリセットが発生するため、リターンしません。 リターンする場合は、システムはリセットしていないかエラーが発生しています。

3.17 R_FWUP_SetPlatformImageState 関数

表 3-17 R_FWUP_SetPlatformImageState 関数仕様

Format	OtaPalStatus_t R_FWUP_SetPlatformImageState (OTA_ImageState_t eState)	
Description	ライフサイクルの状態を、Parameters で指定された状態に設定します。 新しいファームウェアへの更新が完了した場合、テンポラリエリアのバッファ領域を消去します。	
Parameters	eState	：設定する状態
Return Values	OtaPalSuccess	：正常終了
	OtaPalCommitFailed	：コミットエラー
	OtaPalBadImageState	：指定された OTA イメージの状態が範囲外
Special Notes	ー	

3.18 R_FWUP_GetPlatformImageState 関数

表 3-18 R_FWUP_GetPlatformImageState 関数仕様

Format	OtaPalImageState_t R_FWUP_GetPlatformImageState (void)	
Description	現在のライフサイクルの状態を返します。	
Parameters	なし	
Return Values	OtaPalImageStatePendingCommit	：更新待ち
	OtaPalImageStateValid	：有効
	OtaPalImageStateInvalid	：無効
Special Notes	ー	

3.19 R_FWUP_CheckFileSignature 関数　[OS 無し時に使用]

表 3-19 R_FWUP_CheckFileSignature 関数仕様

Format	fwup_err_t R_FWUP_CheckFileSignature(void)	
Description	フラッシュに書き込んだファームウェアの署名を確認します。	
Parameters	なし	
Return Values	FWUP_SUCCESS	：正常終了
	FWUP_FAIL	：署名検証エラー
Special Notes	ー	

3.20 R_FWUP_CheckFileSignature 関数　[OTA 時に使用]

表 3-20 R_FWUP_CheckFileSignature 関数仕様

Format	OtaPalStatus_t R_FWUP_CheckFileSignature (OTA_FileContext_t * const C)	
Description	フラッシュに書き込んだファームウェアの署名を確認します。	
Parameters	* C	：ファイルコンテキスト
Return Values	OtaPalSuccess	：正常終了
	OtaPalSignatureCheckFailed	：署名検証エラー
	OtaPalBadSignerCert	：署名者証明書が読み取り可能でなかったか、長さがゼロ
Special Notes	ー	

3.21 R_FWUP_ReadAndAssumeCertificate 関数

表 3-21 R_FWUP_ReadAndAssumeCertificate 関数仕様

Format	uint8_t * R_FWUP_ReadAndAssumeCertificate (const uint8_t * const pucCertName uint32_t * const ulSignerCertSize)	
Description	指定された署名者証明書をファイルシステムから読み込み返します。	
Parameters	* pucCertName	：証明書ファイル名
	* ulSignerCertSize	：証明書のサイズ
Return Values	証明書データへのポインタ	
Special Notes	ー	

3.22 R_FWUP_GetVersion 関数

表 3-22 R_FWUP_GetVersion 関数仕様

Format	uint32_t R_FWUP_GetVersion (void)
Description	本 FIT モジュールのバージョン番号を返します。
Parameters	なし
Return Values	バージョン番号
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サンプルプログラムです。

デモプロジェクトは、本 FIT 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。4.1 の一覧に示した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では、以下のプロジェクトで構成されています。

デュアルモードのフォルダ構成：□□¥dualbank¥△△¥ の下
リニアモード（半面書き換え方式）のフォルダ構成：□□¥non-dualbank2 ¥△△¥ の下
リニアモード（全面書き換え方式）のフォルダ構成：□□¥non-dualbank3 ¥△△¥ の下

□□：デバイス名
△△：コンパイラ（ccrx/gcc/iar）

- boot_loader：ブートローダプログラム
  リセット後に最初に実行され、ユーザプログラムが改ざんされていないことを検証し、問題無ければユーザプログラムを起動するプログラムです。
- fwup_main：ユーザプログラム（初期ファームウェア／更新ファームウェア）
  更新ファームウェアをダウンロードするユーザプログラム（初期ファームウェア）です。
  また、本プログラムを編集し、ユーザプログラム（更新ファームウェア）としても使用します。
- eol_main：
  EOL 処理をするプログラムです。EOL 処理ではシステムを安全に廃棄するために、コードフラッシュのユーザプログラムとデータフラッシュの内容を消去します。

4.1　デモプロジェクト一覧

本パッケージに含まれるデモプロジェクト一式を、以下に示します。

```
+-rx65n-rsk                          : Demo project set folder using RSK-RX65N starter kit
|  |  amazon-freertos-gcc.zip         :   Amazon FreeRTOS (OTA), CC-RX version demo project
|  |  amazon-freertos.zip             :   Amazon FreeRTOS (OTA), GCC version demo project
|  |
|  +-dualbank
|      +-ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|      |      boot_loader.zip        :   Project of boot loader
|      |      eol_main.zip           :   Project of EOL processing
|      |      fwup_main.zip          :   Project of Firmware update
|      |
|      +-gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|      |      boot_loader_gcc.zip    :   Project of boot loader
|      |      eol_main_gcc.zip       :   Project of EOL processing
|      |      fwup_main_gcc.zip      :   Project of Firmware update
|      |
|      +-iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|             boot_loader_iar.zip    :   Project of boot loader
|             eol_main_iar.zip       :   Project of EOL processing
|             fwup_main_iar.zip      :   Project of Firmware update
|
+-rx66t-rsk                          : Demo project set folder using RSK-RX66T starter kit
|  +-non-dualbank2
|  |   +-ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|  |   |      boot_loader.zip        :   Project of boot loader
|  |   |      eol_main.zip           :   Project of EOL processing
|  |   |      fwup_main.zip          :   Project of Firmware update
|  |   |      fwup_main_woSciDrv.zip :   Project of Firmware update (w/o SCI Driver)
|  |   |
|  |   +-gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|  |   |      boot_loader_gcc.zip    :   Project of boot loader
|  |   |      eol_main_gcc.zip       :   Project of EOL processing
|  |   |      fwup_main_gcc.zip      :   Project of Firmware update
|  |   |
|  |   +-iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|  |          boot_loader_iar.zip    :   Project of boot loader
|  |          eol_main_iar.zip       :   Project of EOL processing
|  |          fwup_main_iar.zip      :   Project of Firmware update
|  |
|  +-non-dualbank3
|      +-ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|             boot_loader.zip        :   Project of boot loader
|             eol_main.zip           :   Project of EOL processing
|             fwup_main.zip          :   Project of Firmware update
|
+-rx72n-rsk                          : Demo project set folder using RSK-RX72N starter kit
|   +-dualbank
|      +-ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|      |      boot_loader.zip        :   Project of boot loader
|      |      eol_main.zip           :   Project of EOL processing
|      |      fwup_main.zip          :   Project of Firmware update
|      |
|      +-gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|      |      boot_loader_gcc.zip    :   Project of boot loader
|      |      eol_main_gcc.zip       :   Project of EOL processing
|      |      fwup_main_gcc.zip      :   Project of Firmware update
|      |
|      +-iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|             boot_loader_iar.zip    :   Project of boot loader
|             eol_main_iar.zip       :   Project of EOL processing
|             fwup_main_iar.zip      :   Project of Firmware update
|
+-rx130-rsk                          : Demo project set folder using RSK-RX130 starter kit
|  +-non-dualbank2
|      +-ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|      |      boot_loader.zip        :   Project of boot loader
|      |      eol_main.zip           :   Project of EOL processing
|      |      fwup_main.zip          :   Project of Firmware update
|      |
|      +-gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|      |      boot_loader_gcc.zip    :   Project of boot loader
|      |      eol_main_gcc.zip       :   Project of EOL processing
|      |      fwup_main_gcc.zip      :   Project of Firmware update
|      |
|      +-iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|             boot_loader_iar.zip    :   Project of boot loader
|             eol_main_iar.zip       :   Project of EOL processing
|             fwup_main_iar.zip      :   Project of Firmware update
```

図 4-1 デモプロジェクト一覧 (1/2)

```
+--rx140-rsk                          : Demo project set folder using RSK-RX140 starter kit
|  +--non-dualbank2
|      +--ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|      |      boot_loader.zip         :   Project of boot loader
|      |      eol_main.zip            :   Project of EOL processing
|      |      fwup_main.zip           :   Project of Firmware update
|      |
|      +--gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|      |      boot_loader_gcc.zip     :   Project of boot loader
|      |      eol_main_gcc.zip        :   Project of EOL processing
|      |      fwup_main_gcc.zip       :   Project of Firmware update
|      |
|      +--iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|             boot_loader_iar.zip     :   Project of boot loader
|             eol_main_iar.zip        :   Project of EOL processing
|             fwup_main_iar.zip       :   Project of Firmware update
|
+--rx231-rsk                          : Demo project set folder using RSK-RX231 starter kit
|  +--non-dualbank2
|      +--ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|      |      boot_loader.zip         :   Project of boot loader
|      |      eol_main.zip            :   Project of EOL processing
|      |      fwup_main.zip           :   Project of Firmware update
|      |
|      +--gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|      |      boot_loader_gcc.zip     :   Project of boot loader
|      |      eol_main_gcc.zip        :   Project of EOL processing
|      |      fwup_main_gcc.zip       :   Project of Firmware update
|      |
|      +--iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|             boot_loader_iar.zip     :   Project of boot loader
|             eol_main_iar.zip        :   Project of EOL processing
|             fwup_main_iar.zip       :   Project of Firmware update
|
+--rx660-rsk                          : Demo project set folder using RSK-RX660 starter kit
|  +--dualbank
|      +--ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
|      |      boot_loader.zip         :   Project of boot loader
|      |      eol_main.zip            :   Project of EOL processing
|      |      fwup_main.zip           :   Project of Firmware update
|      |
|      +--gcc                         : Update firmware demo set folder : for GCC for Renesas RX
|      |      boot_loader_gcc.zip     :   Project of boot loader
|      |      eol_main_gcc.zip        :   Project of EOL processing
|      |      fwup_main_gcc.zip       :   Project of Firmware update
|      |
|      +--iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
|             boot_loader_iar.zip     :   Project of boot loader
|             eol_main_iar.zip        :   Project of EOL processing
|             fwup_main_iar.zip       :   Project of Firmware update
|
+--rx671-rsk                          : Demo project set folder using RSK-RX671 starter kit
   +--dualbank
       +--ccrx                        : Update firmware demo set folder : for C/C++ Compiler Package for RX Family
       |      boot_loader.zip         :   Project of boot loader
       |      eol_main.zip            :   Project of EOL processing
       |      fwup_main.zip           :   Project of Firmware update
       |
       +--gcc                         : Update firmware demo set folder : for GCC for Renesas RX
       |      boot_loader_gcc.zip     :   Project of boot loader
       |      eol_main_gcc.zip        :   Project of EOL processing
       |      fwup_main_gcc.zip       :   Project of Firmware update
       |
       +--iar                         : Update firmware demo set folder : for IAR C/C++ Compiler for RX
              boot_loader_iar.zip     :   Project of boot loader
              eol_main_iar.zip        :   Project of EOL processing
              fwup_main_iar.zip       :   Project of Firmware update
```

図 4-1 デモプロジェクト一覧 (2/2)

4.2　デモプロジェクトのビルド

ファームウェアアップデートを実施するために、ブートローダプログラム、ユーザプログラム（初期ファームウェア）、ユーザプログラム（更新ファームウェア）を準備します。

4.2.1　事前準備

事前準備として、以下を実施してください。

4.2.1.1　統合開発環境の準備

本デモプロジェクトは、5.1 に記載した環境で動作確認を行っています。デモを実施するデバイスやコンパイラに対応した統合開発環境を準備してください。

4.2.1.2　署名検証用の公開鍵／秘密鍵情報の生成

プロジェクトを更新するファームウェアの完全性を確保するために、更新ファームウェアのデジタル署名（ECDSA+SHA256）にて完全性の検証を行います。検証を行うには、署名検証用の公開鍵情報を各プログラムに組み込む必要があります。

また、署名検証用の秘密鍵情報（secp256r1.privatekey）は、4.3 で Image Generator に入力するファイルとして使用します。
以下のリンクを参考に ECDSA+SHA256 用の鍵ペアを生成してください。

https://github.com/renesas/amazon-freertos/wiki/OTA の活用#手順まとめ に記載の

「4.ファームウェア検証に使用する鍵を OpenSSL で作成する」と

「5.ファームウェア検証に ECDSA+SHA256 を使用するため、ブートローダに署名検証用の公開鍵=secp256r1.publickey を仕込む」を参考に署名検証用の公開鍵（code_signer_public_key.h）を生成します。

4.2.2　ブートローダプログラム

任意のプロジェクトの boot_loader.zip を解凍し、ブートローダプログラム用のワークスペースを作成し、boot_loader をインポートします。

4.2.1.2 で生成した署名検証用の公開鍵（code_signer_public_key.h）を boot_loader¥src¥key 以下に組み込み、ビルドを行います。HardwareDebug フォルダに boot_loader.mot ファイルが生成されます。
boot_loader.mot ファイルは Image Generator の入力ファイルとして使用します。（4.3 参照）

4.2.3　ユーザプログラム（初期ファームウェア）

任意のプロジェクトの fwup_main.zip を解凍し、ユーザプログラム（初期ファームウェア）用のワークスペースを作成し、fwup_main をインポートします。

4.2.1.2 で生成した署名検証用の公開鍵（code_signer_public_key.h）を boot_loader¥src¥key 以下に組み込み、ビルドを行います。HardwareDebug フォルダに fwup_main.mot ファイルが生成されます。
fwup_main.mot ファイルは Image Generator の入力ファイルとして使用します。（4.3 参照）

4.2.4　ユーザプログラム（更新ファームウェア）

任意のプロジェクトの fwup_main.zip を解凍し、ユーザプログラム（更新ファームウェア）用のワークスペースを作成し、fwup_main をインポートします。

4.2.1.2 で生成した署名検証用の公開鍵（code_signer_public_key.h）を boot_loader¥src¥key 以下に組み込みます。

fwup_main¥src¥main.c ファイルを開き、以下のコメント行を有効化します。

```
main.c
//    printf("[FWUP_main DEMO] Firmware update demonstration completed.¥r¥n");
//    while(1)
//    {
//        /* infinity loop */
//    }
```

ビルドを行います。HardwareDebug フォルダに fwup_main.mot ファイルが生成されます。fwup_main.mot ファイルは Image Generator の入力ファイルとして使用します。（4.3 参照）

4.3　Image Generator によるファームウェアアップデートイメージファイルの変換

4.2 でビルドして生成された mot ファイルは、Image Generator により、ファームウェアアップデートイメージを構成（5.4 参照）するファイルに変換します。

Image Generator は、以下の URL から mot-file-converter/Renesas Image Generator/bin/Debug/ フォルダごとダウンロードしてください。（Renesas Image Generator.exe と一緒に置かれているファイルも必要となります）

Release mot file converter tool · renesas/mot-file-converter · GitHub

Renesas Image Generator.exe をダブルクリックして Image Generator を起動します。

4.3.1　ブートローダとユーザプログラム（初期ファームウェア）のイメージファイルの生成

ファームウェアアップデートの初期設定をブートローダとユーザプログラム（初期ファームウェア）が書き込まれた状態（デュアルモードは図 1-4④、リニアモード（半面書き換え方式は、図 1-6②、全面書き換え方式は図 1-8②）から開始する場合、ブートローダをビルドして生成された mot ファイル（4.2.2 参照）とユーザプログラム（初期ファームウェア）をビルドして生成された mot ファイル（4.2.3 参照）、また、ECDSA+SHA256 用の署名検証用秘密鍵（4.2.1.2 参照）を Image Generator に入力し、mot ファイルに変換します。なお、イメージ生成のメカニズムについては、5.5.1 を参照願います。

Image Generator を起動し、以下のパラメータブロックを設定して mot ファイルを生成します。
生成した mot ファイルは、4.4.1 で使用します。

①[Initial Firm]タブを選択します。（デフォルトは、[Initial Firm]が選択されています）

②Settings の Select MCU に対象となる MCU を選択します。

③Settings の Select Firmware Verification Type に「sig-sha256-ecdsa」を選択します。

④Settings の Private Key Path に 4.2.1.2 で生成したファイル（secp256r1.privatekey）のパスを設定します。

⑤Settings の Select Output format に「Bank0 User Program + Boot Loader (Motorola S Format)」を設定します。

⑥Boot Loader の File Path (Motorola Format) に 4.2.2 で生成した boot_loader.mot のファイルパスを設定します。

⑦Bank0 User Program の Firmware Sequence Number に 1 を入力します。

⑧Bank0 User Program の File Path (Motorola Format) に 4.2.3 で生成した fwup_main.mot のファイルパスを設定します。

⑨[Generate]をクリックし、生成する userprog.mot（Motrola S format）のファイルを指定します。

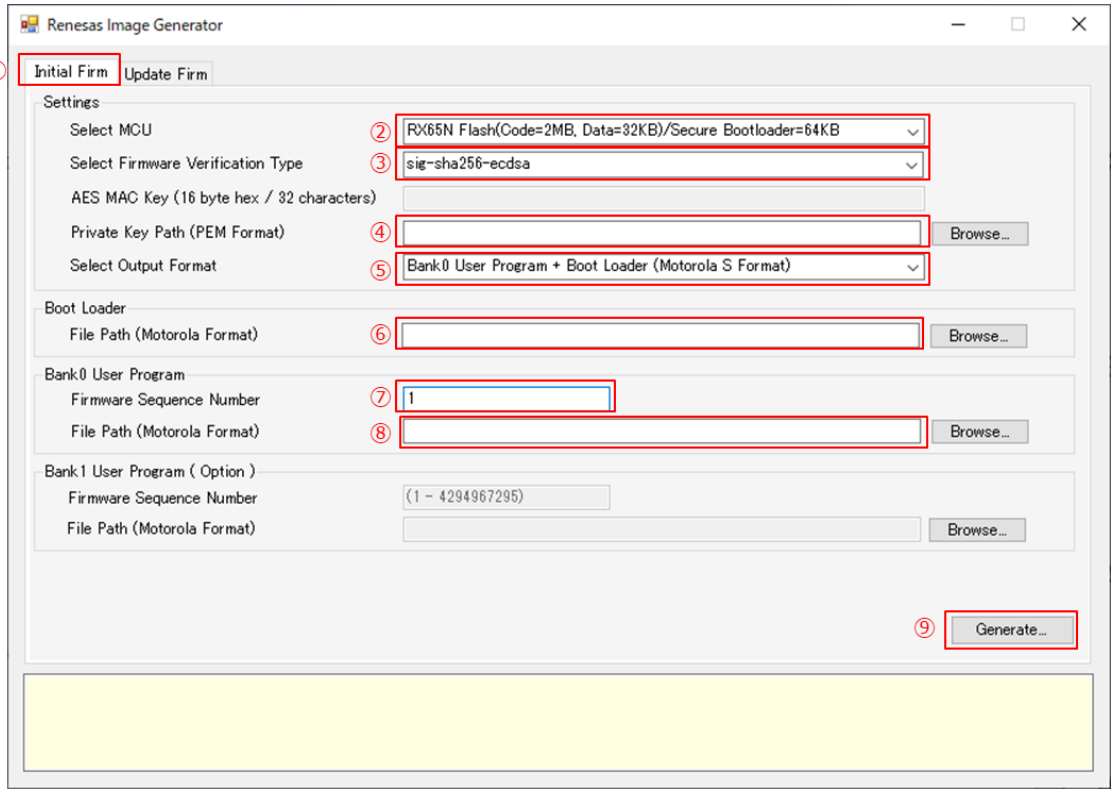


図 4-2 ブートローダとユーザプログラム（初期ファームウェア）のイメージファイル生成

4.3.2　ユーザプログラム（更新ファームウェア）の RSU イメージファイルの生成

更新ファームウェアを生成する場合、ユーザプログラム（更新ファームウェア）をビルドして生成された mot ファイル（4.2.4 参照）、また、ECDSA+SHA256 用の署名検証用秘密鍵（4.2.1.2 参照）を Image Generator に入力し、RSU ファイルに変換します。なお、イメージ生成のメカニズムについては、5.5.2 を参照願います。

Image Generator を起動し、以下のパラメータブロックを設定して RSU ファイルを生成します。
生成した RSU ファイルは、4.4 で使用します。

①[Update Firm]タブを選択します。（デフォルトは、[Initial Firm]が選択されています）

②Settings の Select MCU に対象となる MCU を選択します。

③Settings の Select Firmware Verification Type に「sig-sha256-ecdsa」を選択します。

④Settings の Private Key Path に 4.2.1.2 で生成したファイル（secp256r1.privatekey）のパスを設定します。

⑤Bank0 User Program の Firmware Sequence Number に 1 を入力します。

⑥Bank0 User Program の File Path (Motorola Format) に 4.2.4 で生成した fwup_main.mot のファイルパスを設定します。

⑦[Generate]をクリックし、生成する userprog.rsu（Renesas Secure Update）のファイルを指定します。

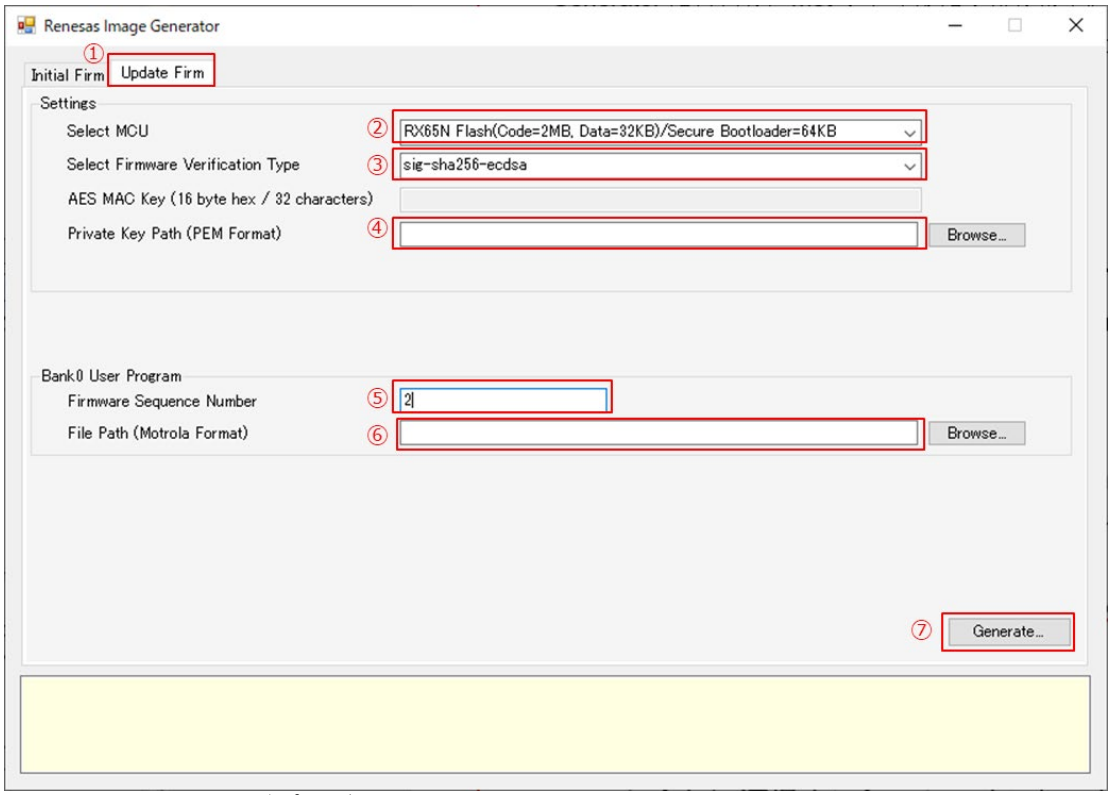


図 4-3 ユーザプログラム（更新ファームウェア）の RSU イメージファイル生成

4.3.3　ユーザプログラム（初期ファームウェア）の RSU イメージファイルの生成

ファームウェアアップデートの初期設定をブートローダが書き込まれた状態（デュアルモードは図 1-4①、リニアモード（半面書き換え方式は、図 1-6①、全面書き換え方式は、図 1-8①）から開始する場合、最初にブートローダをビルドして生成された mot ファイル（4.2.2 参照）をフラッシュライタで MCU ボードに書き込みます。その後、ユーザプログラム（初期ファームウェア）をビルドして生成された mot ファイル（4.2.3 参照）と ECDSA+SHA256 用の署名検証用秘密鍵（4.2.1.2 参照）を Image Generator に入力し、RSU ファイルに変換します。なお、イメージ生成のメカニズムについては、5.5.3 を参照願います。

Image Generator を起動し、以下のパラメータブロックを設定して RSU ファイルを生成します。
生成した RSU ファイルは、4.4 で使用します。

①[Initial Firm]タブを選択します。（デフォルトは、[Initial Firm]が選択されています）

②Settings の Select MCU に対象となる MCU を選択します。

③Settings の Select Firmware Verification Type に「sig-sha256-ecdsa」を選択します。

④Settings の Private Key Path に 4.2.1.2 で生成したファイル（secp256r1.privatekey）のパスを設定します。

⑤Settings の Select Output format に「Bank0 User Program (Binary Format)」を設定します。

⑥Bank0 User Program の Firmware Sequence Number に 1 を入力します。

⑦Bank0 User Program の File Path (Motorola Format) に 4.2.3 で生成した fwup_main.mot のファイルパスを設定します。

⑧[Generate]をクリックし、生成する userprog.rsu（Renesas Secure Update）のファイルを指定します。

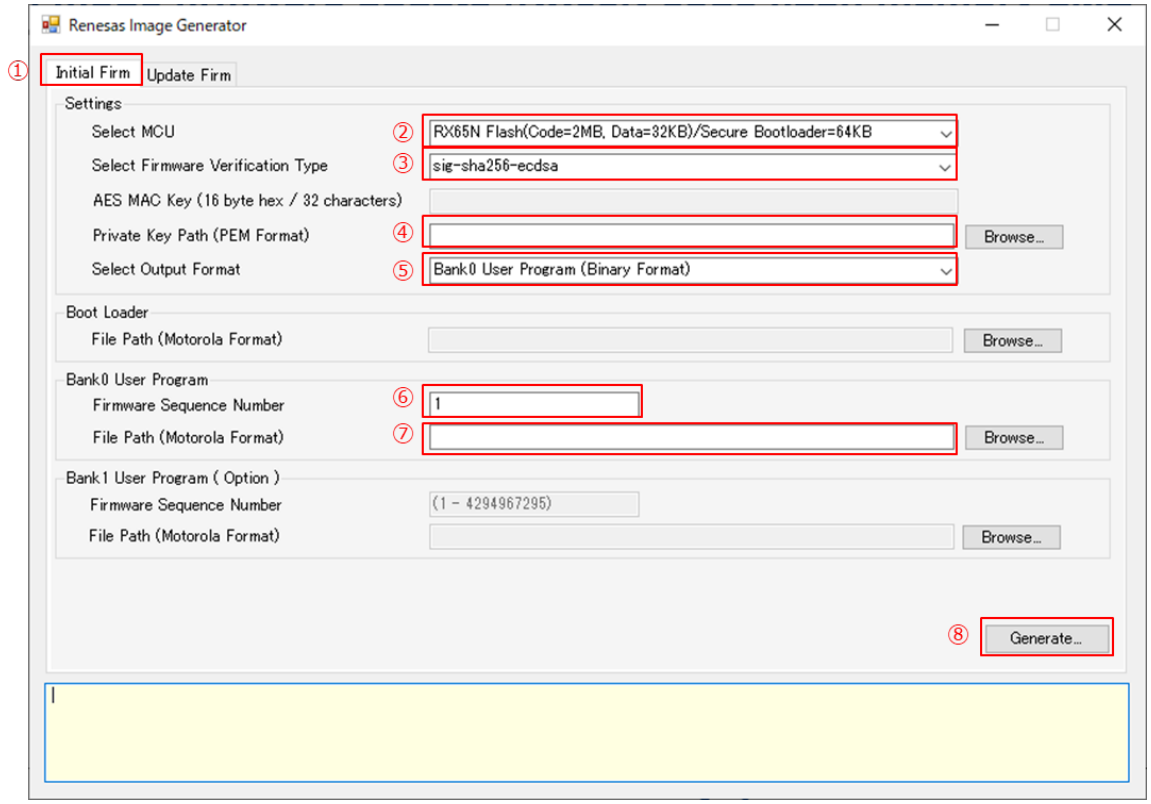


図 4-4 ユーザプログラム（初期ファームウェア）の RSU イメージファイル生成

4.4　シリアル通信インターフェイス(SCI)を用いたファームウェアアップデート

本章では、デュアルモード、リニアモード（半面書き換え方式）、リニアモード（全面書き換え方式）における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デモの実施方法について説明します。UART として構成された SCI チャネルを介してターミナルと通信を行い、ファームウェアをアップデートします。

4.4.1　デュアルモードのファームウェアアップデート

本章では例として、RSK RX65N スターターキットの RX65N 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デモの実施方法について説明します。

4.4.1.1　実行環境の準備

本ファームウェアアップデートのデモでは、PMOD1 にインタフェースされているシリアルポート SCI6 を使用しています。PMOD1 コネクタには USB シリアル変換ボードを接続します。

ターミナルソフトウェアを実行している PC が入出力用に必要となります。

表 4-1 機器構成

No.	機器	補足
1	開発 PC	開発を行う PC です。
2	評価ボード （Renesas Starter Kit for RX65N）	―
3	ホスト PC（TeraTerm 等のターミナルソフトウェア）	XMODEM/SUM 転送プロトコルに対応したシリアル通信ソフトウェア（開発 PC でも代用可能です） デモでは、TeraTerm 4.105 を使用して動作確認を行っています。
4	USB シリアル変換ボード	Renesas Starter Kit for RX65N のシリアル入出力信号を USB シリアル変換し、ホスト PC と USB 接続します。
5	USB ケーブル	USB シリアル変換ボードとホスト PC を USB 接続します。

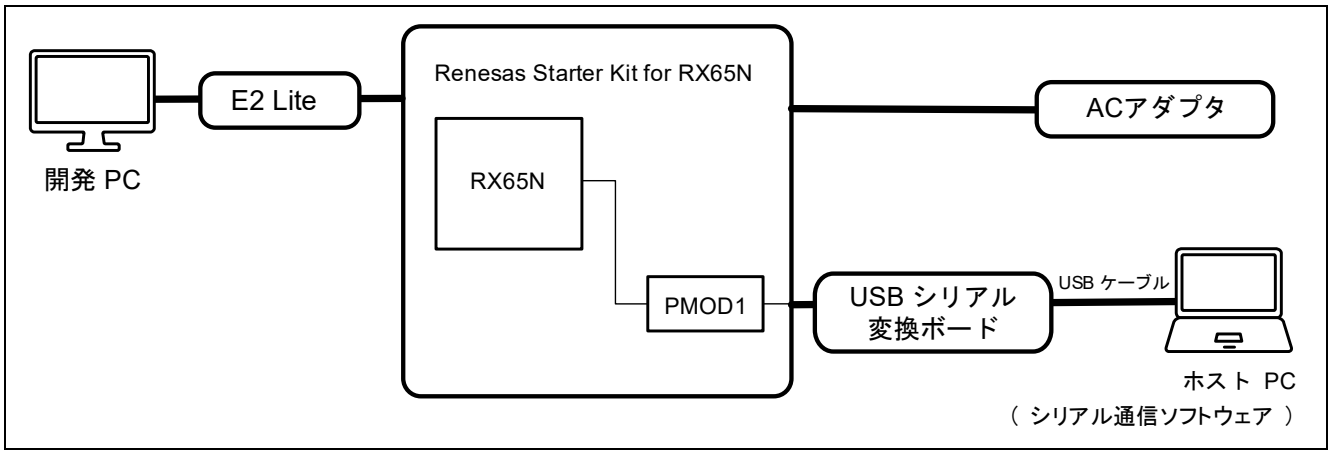


図 4-5 RSK RX65N 機器接続図

表 4-2 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	なし

4.4.1.2 ブートローダとユーザプログラム（初期ファームウェア）の書き込み

ファームウェアアップデートの初期設定をブートローダと初期ファームウェアが書き込まれた状態（図 1-4④）から開始する場合、4.3.1 で生成したブートローダとユーザプログラム（初期ファームウェア）の mot ファイルをフラッシュライタで MCU ボードに書き込みます。書き込みを行った後は、ボードの電源を OFF し、エミュレータ接続などを外しておいてください。

4.4.1.3 ファームウェアアップデートの実行

初期ファームウェアは、シリアル通信による更新ファームウェアの転送を待ちます。転送したプログラムはコードフラッシュに書き込みを行います。転送が完了すると、転送された更新ファームウェアは、署名検証を行った後、ファームウェアがアップデートされます。（1.3.1 参照）

以下の手順により、ファームウェアアップデートを実施してください。

1. 「図 4-5 RSK RX65N 機器接続図」の様に PC の USB ポート, USB シリアル変換ボード, RSK ボードの PMOD1 を接続してください。
2. PC 上のターミナルソフトウェア（TeraTerm 4.105）を開きます。そして、USB シリアル変換ボードに割り当てられたシリアル COM ポートを選択します。
3. このサンプルアプリケーションの設定と一致するように、ターミナルのシリアル設定を行います。115200bps、8 ビットデータ、パリティなし、1 ストップビット、フロー制御なし。
4. ボードの電源を ON します。以下のメッセージが出力されます。

```
jump to user program
[INFO] Receive file created.
----------------------------------------
FIRMWARE UPDATE demo version 0.1.1
FWUP FIT module version 1.06
----------------------------------------
The firmware update will start.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。

.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
[INFO] Flash Write: Address = 0xFFE00000, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFE00400, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFE00800, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFE00C00, length = 1024byte ... 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。

```
jump to user program
[INFO] Receive file created.
```

6. 更新ファームウェアで以下のメッセージが出力されるとデモの正常終了です。

```
[FWUP_main DEMO] Firmware update demonstration completed.
```

4.4.1.4 ユーザプログラム（初期ファームウェア）の書き込み

ファームウェアアップデートの初期設定をブートローダが書き込まれた状態（図 1-4①）から開始する場合、4.2.2 でビルドしたブートローダ（mot ファイル）を事前にフラッシュライタで MCU ボードに書き込んでおきます。

ボードの電源を ON することによりブートローダが起動し、シリアル通信による初期ファームウェアの転送を待ちます。

転送したプログラムはコードフラッシュに書き込みを行い、転送が完了した後、署名検証を行い、初期ファームウェアを起動します。初期ファームウェアが起動すると、更新ファームウェアの転送を待ちます。転送したプログラムはコードフラッシュに書き込みを行います。転送が完了すると、更新ファームウェアの署名検証を行った後、ファームウェアがアップデートされます。（1.3.1 参照）以下の手順により、ファームウェアアップデートを実施してください。

1. 「図 4-5 RSK RX65N 機器接続図」の様に PC の USB ポート, USB シリアル変換ボード, RSK ボードの PMOD1 を接続してください。
2. PC 上のターミナルソフトウェア（以下、ターミナル）を開きます。そして、USB シリアル変換ボードに割り当てられたシリアル COM ポートを選択します。
3. このサンプルアプリケーションの設定と一致するように、ターミナルのシリアル設定を行います。115200bps、8 ビットデータ、パリティなし、1 ストップビット、フロー制御なし。
4. ソフトウェアを実行すると以下のメッセージが出力されます。

```
send "userprog.rsu" via UART.
```

ターミナルの機能で「ファイル送信」を選び、4.3.3 で生成した初期ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
installing firmware...0%(1/960KB).
installing firmware...0%(2/960KB).
installing firmware...0%(3/960KB).
installing firmware...0%(4/960KB).
```

5. インストールと署名検証が終了すると初期ファームウェアが起動され、更新ファームウェアの入力を促すメッセージが出力されます。

```
jump to user program
[INFO] Receive file created.
----------------------------------------
FIRMWARE UPDATE demo version 0.1.1
FWUP FIT module version 1.06
----------------------------------------
The firmware update will start.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。

.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
[INFO] Flash Write: Address = 0xFFE00000, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFE00400, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFE00800, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFE00C00, length = 1024byte ... OK
```

6. 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。

```
jump to user program
[INFO] Receive file created.
```

7. 更新ファームウェアで以下のメッセージが出力されるとデモの正常終了です。

```
[FWUP_main DEMO] Firmware update demonstration completed.
```

4.4.2　リニアモード（半面書き換え方式）のファームウェアアップデート

本章では例として、RSK RX66T スターターキットの RX66T 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デモの実施方法について説明します。

4.4.2.1　実行環境の準備

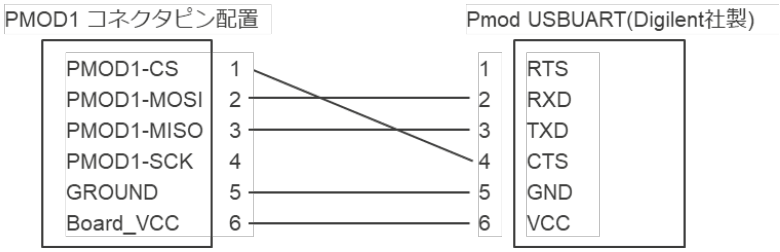
本ファームウェアアップデートのデモでは、PMOD1 にインタフェースされているシリアルポート SCI6 を使用しています。PMOD1 コネクタには USB シリアル変換ボードを接続します。

ターミナルソフトウェアを実行している PC が入出力用に必要となります。

表 4-3 機器構成

No.	機器	補足
1	開発 PC	開発を行う PC です。
2	評価ボード （Renesas Starter Kit for RX66T）	5V 電源の電源供給が必要となるため、J7 設定をショートさせてください。
3	ホスト PC（TeraTerm 等のターミナルソフトウェア）	XMODEM/SUM 転送プロトコルに対応したシリアル通信ソフトウェア（開発 PC でも代用可能です） デモでは、TeraTerm 4.105 を使用して動作確認を行っています。
4	USB シリアル変換ボード	Renesas Starter Kit for RX66T のシリアル入出力信号を USB シリアル変換し、ホスト PC と USB 接続します。（※1）
5	USB ケーブル	USB シリアル変換ボードとホスト PC を USB 接続します。

※1 ･･･ 本デモプロジェクトでは、Digilent 社製の Pmod USBUART を使用して動作確認を行っています。
Pmod USBUART と PMOD1 は、以下のような信号の接続が必要となります。



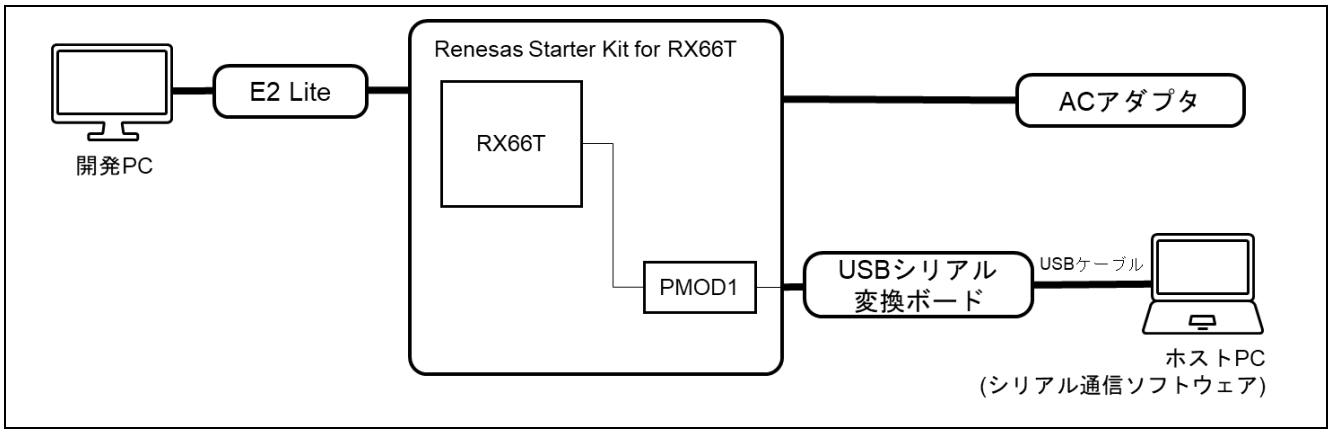


図 4-6 RSK RX66T 機器接続図

表 4-4 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	RTS/CTS

4.4.2.2　ブートローダとユーザプログラム（初期ファームウェア）の書き込み

ファームウェアアップデートの初期設定をブートローダと初期ファームウェアが書き込まれた状態（図 1-6②）から開始する場合、4.3.1 で生成したブートローダとユーザプログラム（初期ファームウェア）の mot ファイルをフラッシュライタで MCU ボードに書き込みます。書き込みを行った後は、ボードの電源を OFF し、エミュレータ接続などを外しておいてください。

4.4.2.3　ファームウェアアップデートの実行

初期ファームウェアは、シリアル通信による更新ファームウェアの転送を待ちます。転送したプログラムはコードフラッシュに書き込みを行います。転送が完了すると、転送された更新ファームウェアは、署名検証を行った後、ファームウェアがアップデートされます。（1.3.2.1 参照）

以下の手順により、ファームウェアアップデートを実施してください。

1. 「図 4-6 RSK RX66T 機器接続図」の様に PC の USB ポート, USB シリアル変換ボード, RSK ボードの PMOD1 を接続してください。
2. PC 上のターミナルソフトウェア（TeraTerm 4.105）を開きます。そして、USB シリアル変換ボードに割り当てられたシリアル COM ポートを選択します。
3. このサンプルアプリケーションの設定と一致するように、ターミナルのシリアル設定を行います。115200bps、8 ビットデータ、パリティなし、1 ストップビット、フロー制御（RTS/CTS）
4. ボードの電源を ON します。以下のメッセージが出力されます。

```
jump to user program
[INFO] Receive file created.
----------------------------------------
FIRMWARE UPDATE demo version 0.1.1
FWUP FIT module version 1.06
----------------------------------------
The firmware update will start.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。転送されたファイルは、初期ファームウェアにより、データ受信・ユーザプログラムエリア 1 への書き込みを行います。データ受信・書き込み中は以下のメッセージが出力されます。

```
[INFO] Flash Write: Address = 0xFFF80000, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFF80400, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFF80800, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFF80C00, length = 1024byte ... 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、ユーザプログラムエリア 1 に書き込まれた更新ファームウェアをユーザプログラムエリア 0 にコピーした後、ユーザプログラムエリア 1 を消去し、ユーザプログラムエリア 0（更新ファームウェア）が起動し、プログラムが実行されます。

```
jump to user program
[INFO] Receive file created.
```

6. 更新ファームウェアで以下のメッセージが出力されるとデモの正常終了です。

```
[FWUP_main DEMO] Firmware update demonstration completed.
```

4.4.2.4　ユーザプログラム（初期ファームウェア）の書き込み

ファームウェアアップデートの初期設定をブートローダが書き込まれた状態（図 1-6①）から開始する場合、4.2.2 でビルドしたブートローダ（mot ファイル）を事前にフラッシュライタで MCU ボードに書き込んでおきます。

ボードの電源を ON することによりブートローダが起動し、シリアル通信による初期ファームウェアの転送を待ちます。

転送したプログラムはコードフラッシュおよびデータフラッシュに書き込みを行い転送が完了した後、署名検証を行い、初期ファームウェアを起動します。初期ファームウェアが起動すると、更新ファームウェアの転送を待ちます。転送したプログラムはコードフラッシュに書き込みを行います。転送が完了すると、転送された更新ファームウェアは、署名検証を行った後、ファームウェアがアップデートされます。（1.3.2.1 参照）以下の手順により、ファームウェアアップデートを実施してください。

1. 「図 4-6 RSK RX66T 機器接続図」の様に PC の USB ポート, USB シリアル変換ボード, RSK ボードの PMOD1 を接続してください。
2. PC 上のターミナルソフトウェア（以下、ターミナル）を開きます。そして、USB シリアル変換ボードに割り当てられたシリアル COM ポートを選択します。
3. このサンプルアプリケーションの設定と一致するように、ターミナルのシリアル設定を行います。115200bps、8 ビットデータ、パリティなし、1 ストップビット、フロー制御（RTS/CTS）
4. ソフトウェアを実行すると以下のメッセージが出力されます。

```
send "userprog.rsu" via UART.
```

ターミナルの機能で「ファイル送信」を選び、4.3.3 で生成した初期ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。

.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
installing firmware...0%(1/960KB).
installing firmware...0%(2/960KB).
installing firmware...0%(3/960KB).
installing firmware...0%(4/960KB).
```

5. インストールと署名検証が終了すると初期ファームウェアが起動され、更新ファームウェアの入力を促すメッセージが出力されます。

```
jump to user program
[INFO] Receive file created.
----------------------------------------
FIRMWARE UPDATE demo version 0.1.1
FWUP FIT module version 1.06
----------------------------------------
The firmware update will start.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。

.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
[INFO] Flash Write: Address = 0xFFF80000, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFF80400, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFF80800, length = 1024byte ... OK
[INFO] Flash Write: Address = 0xFFF80C00, length = 1024byte ... OK
```

6. 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、ユーザプログラムエリア 1 に書き込まれた更新ファームウェアをユーザプログラムエリア 0 にコピーした後、ユーザプログラムエリア 1 を消去し、ユーザプログラムエリア 0（更新ファームウェア）を再起動し、プログラムが実行されます。

```
jump to user program
[INFO] Receive file created.
```

7. 更新ファームウェアで以下のメッセージが出力されるとデモの正常終了です。

```
[FWUP_main DEMO] Firmware update demonstration completed.
```

4.4.3　リニアモード（全面書き換え方式）のファームウェアアップデート

本章では例として、RSK RX66T スターターキットの RX66T 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デモの実施方法について説明します。

4.4.3.1　実行環境の準備

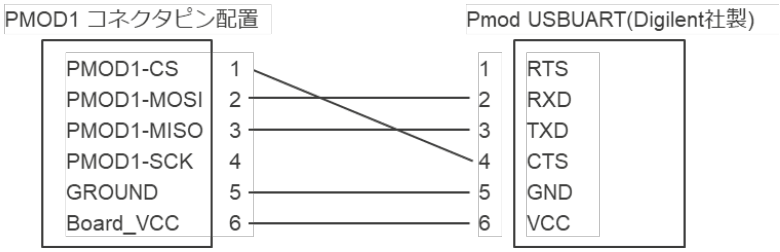
本ファームウェアアップデートのデモでは、PMOD1 にインタフェースされているシリアルポート SCI6 を使用しています。PMOD1 コネクタには USB シリアル変換ボードを接続します。

ターミナルソフトウェアを実行している PC が入出力用に必要となります。

表 4-5 機器構成

No.	機器	補足
1	開発 PC	開発を行う PC です。
2	評価ボード （Renesas Starter Kit for RX66T）	5V 電源の電源供給が必要となるため、J7 設定をショートさせてください。
3	ホスト PC（TeraTerm 等のターミナルソフトウェア）	XMODEM/SUM 転送プロトコルに対応したシリアル通信ソフトウェア（開発 PC でも代用可能です） デモでは、TeraTerm 4.105 を使用して動作確認を行っています。
4	USB シリアル変換ボード	Renesas Starter Kit for RX66T のシリアル入出力信号を USB シリアル変換し、ホスト PC と USB 接続します。（※1）
5	USB ケーブル	USB シリアル変換ボードとホスト PC を USB 接続します。

※1 ･･･ 本デモプロジェクトでは、Digilent 社製の Pmod USBUART を使用して動作確認を行っています。
Pmod USBUART と PMOD1 は、以下のような信号の接続が必要となります。



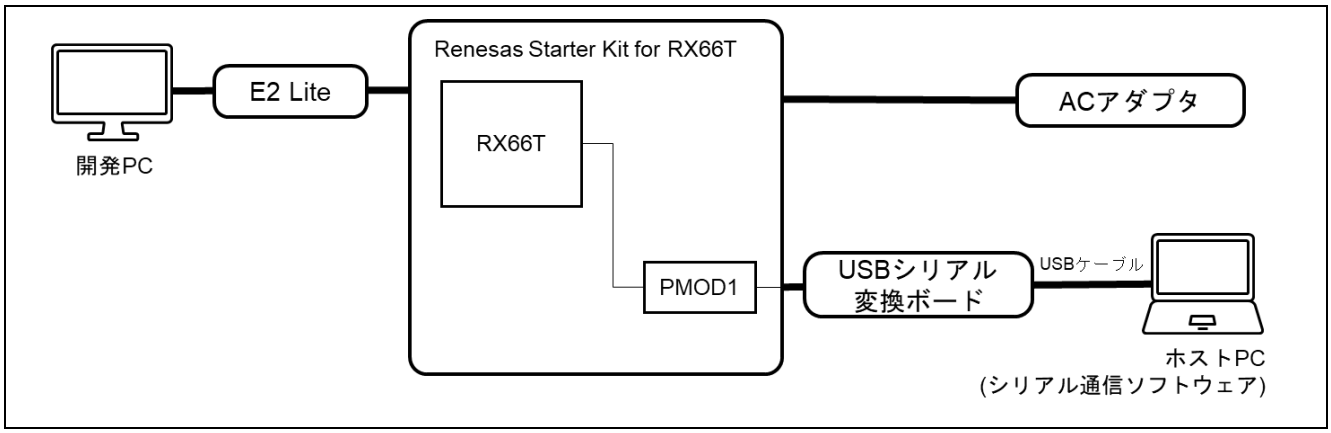


図 4-7 RSK RX66T 機器接続図

表 4-6 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	RTS/CTS

4.4.3.2　ブートローダとユーザプログラム（初期ファームウェア）の書き込み

ファームウェアアップデートの初期設定をブートローダと初期ファームウェアが書き込まれた状態（図 1-6②）から開始する場合、4.3.1 で生成したブートローダとユーザプログラム（初期ファームウェア）の mot ファイルをフラッシュライタで MCU ボードに書き込みます。書き込みを行った後は、ボードの電源を OFF し、エミュレータ接続などを外しておいてください。（初期設定をブートローダのみが書き込まれた状態から開始する場合は、4.2.2 でビルドしたブートローダ（mot ファイル）を事前にフラッシュライタで MCU ボードに書き込んでおいてください）

4.4.3.3　ファームウェアアップデートの実行

ボードの電源を ON することにより、初期ファームウェアが起動した後、直ぐに更新ファイルへのファームウェアアップデートを実施させるためのイメージフラグを空に設定し、ソフトウェアリセットを行います。ソフトウェアリセットにより、ブートローダが起動し、ユーザプログラムエリアを消去し、更新ファームウェアの転送を待ちます。転送したプログラムはコードフラッシュに書き込みを行います。転送が完了すると、転送された更新ファームウェアの署名検証を行った後、ファームウェアがアップデートされます。（1.3.2.2 参照）

以下の手順により、ファームウェアアップデートを実施してください。

1. 「図 4-7 RSK RX66T 機器接続図」の様に PC の USB ポート, USB シリアル変換ボード, RSK ボードの PMOD1 を接続してください。
2. PC 上のターミナルソフトウェア（TeraTerm 4.105）を開きます。そして、USB シリアル変換ボードに割り当てられたシリアル COM ポートを選択します。
3. このサンプルアプリケーションの設定と一致するように、ターミナルのシリアル設定を行います。115200bps、8 ビットデータ、パリティなし、1 ストップビット、フロー制御（RTS/CTS）
4. ボードの電源を ON します。以下のメッセージが出力されます。

```
------------------------------------------------
BOOTLOADER demo version 0.1.1
FWUP FIT module version 1.06
------------------------------------------------
RX66T secure boot program
------------------------------------------------
Checking flash ROM status.
bank 0 status = 0xff [LIFECYCLE_STATE_BLANK]
bank 1 status = 0xff [LIFECYCLE_STATE_BLANK]
start installing user program.
========== install user program phase ==========
erase install area (data flash): OK
erase install area (code flash): OK
send "userprog.rsu" via UART.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。転送されたファイルは、ブートローダにより、データ受信・ユーザプログラムエリアへの書き込みを行います。データ受信・書き込み中は以下のメッセージが出力されます。

```
installing firmware...0%(1/960KB).
installing firmware...0%(2/960KB).
installing firmware...0%(3/960KB).
installing firmware...0%(4/960KB).
```

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、ユーザプログラムエリア（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。

```
jump to user program
[INFO] Receive file created.
```

6. 更新ファームウェアで以下のメッセージが出力されるとデモの正常終了です。

```
[FWUP_main DEMO] Firmware update demonstration completed.
```

4.4.3.4　ユーザプログラム（初期ファームウェア）の書き込み

ファームウェアアップデートの初期設定をブートローダが書き込まれた状態（図 1-8①）から開始する場合、4.2.2 でビルドしたブートローダ（mot ファイル）を事前にフラッシュライタで MCU ボードに書き込んでおきます。

ボードの電源を ON することにより、初期ファームウェアが起動した後、直ぐに更新ファイルへのファームウェアアップデートを実施させるためのイメージフラグを空に設定し、ソフトウェアリセットを行います。ソフトウェアリセットにより、ブートローダが起動し、ユーザプログラムエリアを消去し、更新ファームウェアの転送を待ちます。転送したプログラムはコードフラッシュに書き込みを行います。転送が完了すると、転送された更新ファームウェアの署名検証を行った後、ファームウェアがアップデートされます。（1.3.2.2 参照）以下の手順により、ファームウェアアップデートを実施してください。

1. 「図 4-7 RSK RX66T 機器接続図」の様に PC の USB ポート, USB シリアル変換ボード, RSK ボードの PMOD1 を接続してください。
2. PC 上のターミナルソフトウェア（以下、ターミナル）を開きます。そして、USB シリアル変換ボードに割り当てられたシリアル COM ポートを選択します。
3. このサンプルアプリケーションの設定と一致するように、ターミナルのシリアル設定を行います。115200bps、8 ビットデータ、パリティなし、1 ストップビット、フロー制御（RTS/CTS）
4. ソフトウェアを実行すると以下のメッセージが出力されます。

```
send "userprog.rsu" via UART.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。

.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
installing firmware...0%(1/960KB).
installing firmware...0%(2/960KB).
installing firmware...0%(3/960KB).
installing firmware...0%(4/960KB).
```

5. インストールと署名検証が終了すると初期ファームウェアが起動され、更新ファームウェアの入力を促すメッセージが出力されます。

```
jump to user program
[INFO] Receive file created.
----------------------------------------
FIRMWARE UPDATE demo version 0.1.1
FWUP FIT module version 1.06
----------------------------------------
The firmware update will start.
[INFO] Update ExeHeader ImageFlag : OK
[INFO] Resetting the device.
------------------------------------------------
BOOTLOADER demo version 0.1.1
FWUP FIT module version 1.06
------------------------------------------------
RX66T secure boot program
------------------------------------------------
Checking flash ROM status.
bank 0 status = 0xff [LIFECYCLE_STATE_BLANK]
bank 1 status = 0xff [LIFECYCLE_STATE_BLANK]
start update user program.
========== install user program phase ==========
erase install area (data flash): SKIP
erase install area (code flash): OK
send update firmware via UART.
```

ターミナルの機能で「ファイル送信」を選び、4.3.2 で生成した更新ファームウェア（.RSU ファイル）を送信ターミナルの機能で「ファイル送信」を選択し、（送信オプションとしてバイナリの送信を設定してください）ファイルを転送します。1 分以内にファイル送信を開始してください。

.RSU ファイルのデータ受信・コードフラッシュへの書き込み中は以下のメッセージが出力されます。

```
installing firmware...0%(1/960KB).
installing firmware...0%(2/960KB).
installing firmware...0%(3/960KB).
installing firmware...0%(4/960KB).
```

6. インストールと署名検証が終了すると更新ファームウェアが起動されます。

```
jump to user program
[INFO] Receive file created.
```

7. 更新ファームウェアで以下のメッセージが出力されるとデモの正常終了です。

```
[FWUP_main DEMO] Firmware update demonstration completed.
```

5. 付録

5.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 5-1 動作確認環境 (CC-RX)

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 2022 10
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.04.00 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.1.06
使用ボード	Renesas Starter Kit+ for RX65N（製品型名：RTK50565N2SxxxxxBE） Renesas Starter Kit+ for RX72N（製品型名：RTK5572NNxxxxxxxBE） Renesas Starter Kit+ for RX671（製品型名：RTK55671EHS10000BE） Renesas Starter Kit for RX66T（製品型名：RTK50566T0S00000BE） Renesas Starter Kit for RX660（製品型名：RTK556609HCxxxxxBJ） Renesas Starter Kit+ for RX231（製品型名：R0K505231SxxxBE） Renesas Starter Kit for RX130-512KB（製品型名：RTK5051308SxxxxxBE） Renesas Starter Kit for RX140-256KB（製品型名：RTK551406BxxxxxBJ）
USB シリアル変換ボード	Pmod USBUART（DIGILENT 製） https://reference.digilentinc.com/reference/pmod/pmodusbuart/start

表 5-2 動作確認環境 (GCC)

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 2022 10
C コンパイラ	GCC for Renesas RX 8.3.0.202202 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 -std=gnu99
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.1.06
使用ボード	Renesas Starter Kit+ for RX65N（製品型名：RTK50565N2SxxxxxBE） Renesas Starter Kit+ for RX72N（製品型名：RTK5572NNxxxxxxxBE） Renesas Starter Kit+ for RX671（製品型名：RTK55671EHS10000BE） Renesas Starter Kit for RX66T（製品型名：RTK50566T0S00000BE） Renesas Starter Kit for RX660（製品型名：RTK556609HCxxxxxBJ） Renesas Starter Kit+ for RX231（製品型名：R0K505231SxxxBE） Renesas Starter Kit for RX130-512KB（製品型名：RTK5051308SxxxxxBE） Renesas Starter Kit for RX140-256KB（製品型名：RTK551406BxxxxxBJ）
USB シリアル変換ボード	Pmod USBUART（DIGILENT 製） https://reference.digilentinc.com/reference/pmod/pmodusbuart/start

表 5-3 動作確認環境 (IAR)

項目	内容
統合開発環境	IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	IAR C/C++ Compiler for Renesas RX 4.20.3 コンパイルオプション：統合開発環境のデフォルト設定
スマート・コンフィグレータ	RX スマート・コンフィグレータ V2.14.0
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.1.06
使用ボード	Renesas Starter Kit+ for RX65N（製品型名：RTK50565N2SxxxxxBE） Renesas Starter Kit+ for RX72N（製品型名：RTK5572NNxxxxxxxBE） Renesas Starter Kit+ for RX671（製品型名：RTK55671EHS10000BE） Renesas Starter Kit for RX66T（製品型名：RTK50566T0S00000BE） Renesas Starter Kit for RX660（製品型名：RTK556609HCxxxxxBJ） Renesas Starter Kit+ for RX231（製品型名：R0K505231SxxxBE） Renesas Starter Kit for RX130-512KB（製品型名：RTK5051308SxxxxxBE） Renesas Starter Kit for RX140-256KB（製品型名：RTK551406BxxxxxBJ）
USB シリアル変換ボード	Pmod USBUART（DIGILENT 製） https://reference.digilentinc.com/reference/pmod/pmodusbuart/start

ファームウェアアップデートの動作確認のために、デモプロジェクトで使用した FIT モジュールのバージョン一覧を以下に示します。

(1) ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family の環境

表 5-4 FIT モジュールのバージョン一覧(CC-RX)

デバイス	プロジェクト	r_bsp	r_byteq	r_flash_rx	r_fwup	r_sys_time_rx	r_sci_rx	r_cmt_rx
RX130	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX140	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX231	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX65N	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
	aws_demos	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX66T (non-dualbank2)	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
	fwup_main_woSciDrv	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX66T (non-dualbank3)	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX660	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX671	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX72N	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20

(2) GCC for Renesas RX の環境

表 5-5 FIT モジュールのバージョン一覧(GCC)

デバイス	プロジェクト	r_bsp	r_byteq	r_flash_rx	r_fwup	r_sys_time_rx	r_sci_rx	r_cmt_rx
RX130	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX140	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX231	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX65N	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
	aws_demos	7.00	2.00	4.90	1.06	1.01	4.40	5.20
RX66T	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX660	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX671	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX72N	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20

(3) IAR C/C++ Compiler for RX の環境

表 5-6 FIT モジュールのバージョン一覧(IAR)

デバイス	プロジェクト	r_bsp	r_byteq	r_flash_rx	r_fwup	r_sys_time_rx	r_sci_rx	r_cmt_rx
RX130	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX140	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX231	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX65N	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX66T	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX660	boot_loader fwup_main eol_main	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX671	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20
RX72N	boot_loader_gcc fwup_main_gcc eol_main_gcc	7.20	2.00	4.90	1.06	1.01	4.40	5.20

5.2　コンパイラ依存の設定

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

5.2.1　Renesas Electronics C/C++ Compiler Package for RX Family を使用する場合

コンパイラとして Renesas Electronics C/C++ Compiler Package for RX Family を使用する場合について示します。

リンカのセクションの設定は e^2 studio で行う必要があります。

5.2.1.1　コンパイルオプション

統合開発環境のデフォルト設定に以下のオプションを追加してください。

-lang = c99

5.2.1.2　フラッシュメモリ上の配置アドレスの変更

フラッシュメモリ上の実行領域にブートローダとユーザプログラムを配置するために、リンカのセクション設定を変更します。

1) 「プロジェクト・エクスプローラー」においてデバッグ対象のプロジェクトをクリックします。
2) 「ファイル」→「プロパティ」の順にクリックし、「プロパティ」ウィンドウを開きます。
3) 「プロパティ」ウィンドウで、「C/C++ビルド」→「設定」の順にクリックします。
4) 「ツール設定」タブを押下し、「Linker」→「セクション」の順にクリックして、「…」ボタンを押下し、「セクション・ビューアー」ウィンドウを開きます。

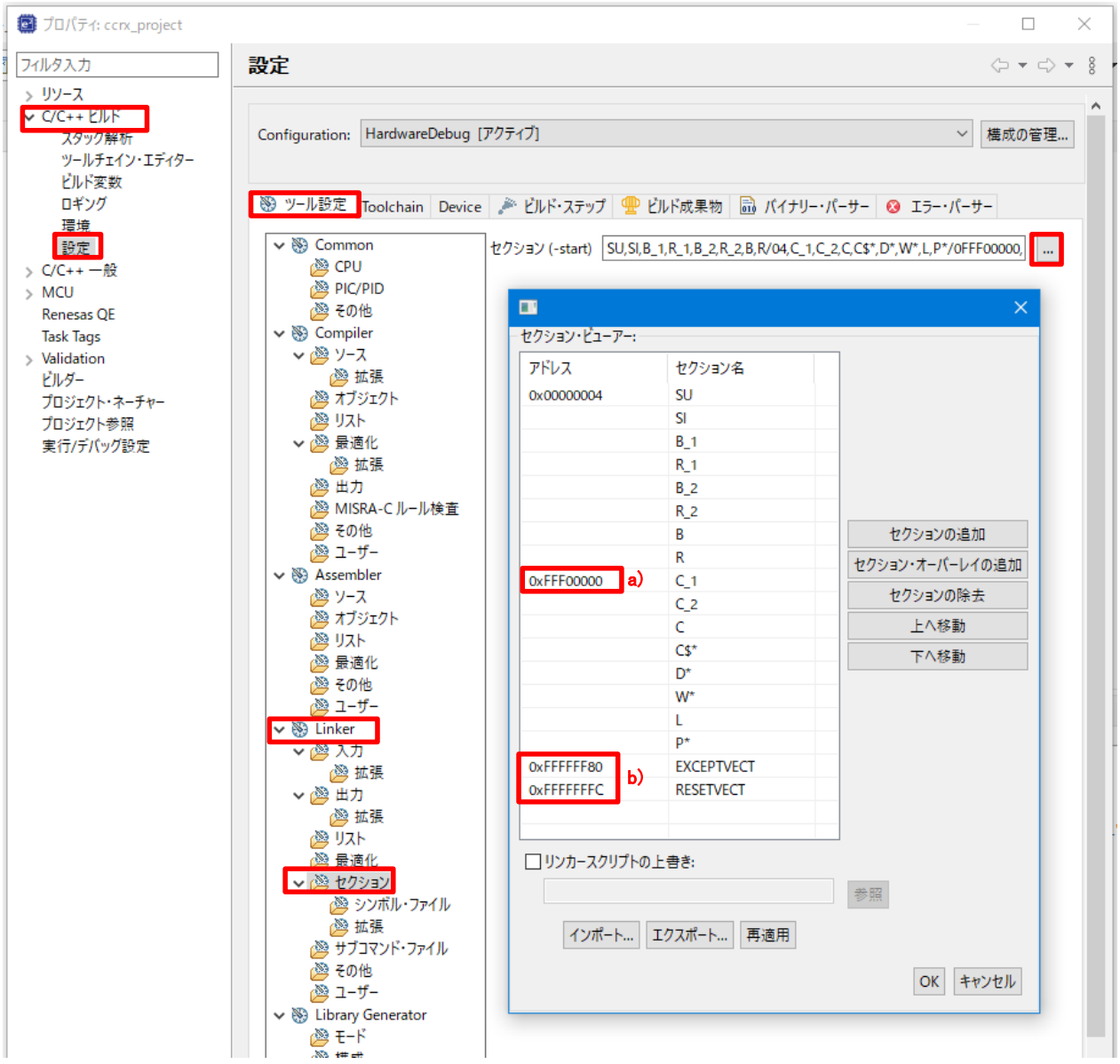


図 5-1 Renesas Electronics C/C++ Compiler Package for RX Family でのセクション設定

5) 「セクション・ビューアー」ウィンドウの a)～b)を、ユーザの環境に合わせて変更してください。
例) RX65N、デュアルモード、ブートローダのサイズ=64KB の場合の各設定値は以下のとおり。

記号	説明	ブートローダの設定	ユーザプログラムの設定
a)	フラッシュメモリ上での開始アドレス	0xFFFF0000	0xFFF00300
b)	例外ベクタとリセットベクタの配置アドレス	0xFFFFFF80 0xFFFFFFFC	0xFFFEFF80 0xFFFEFFFC

5.2.1.3　フラッシュメモリ書き換えの設定

ユーザプログラムおよびブートプログラムに、フラッシュメモリ書き換えのための設定をする必要があります。設定の詳細は、以下アプリケーションノートを参照してください。

「フラッシュモジュール Firmware Integration Technology (R01AN2184)」の
「5.3.1 Renesas Electronics C/C++ Compiler Package for RX Family を使用する場合」

5.2.2　GCC for Renesas RX を使用する場合

コンパイラとして GCC for Renesas RX を使用する場合について示します。

リンカの設定は e^2 studio で生成されるリンカ設定ファイルを編集する必要があります。

5.2.2.1　コンパイルオプション

1) コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加してください。

-std=gnu99

2) リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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してください。

-Wl,--no-gc-sections

これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。

3) コンパイルオプション：ブートローダをデバッグする場合、統合開発環境のデフォルト設定から以下のオプションを変更してください。

最適化レベル：Optimize for debug (-Og)

5.2.2.2　フラッシュメモリ上の配置アドレスの変更

内蔵フラッシュ上の実行領域にブートローダとユーザプログラムを配置するために、リンカ設定を変更します。

1) プロジェクト・エクスプローラーからリンカ設定ファイル(linker_script.ld)を右クリックして、「開く」を選択します。

2) 「linker_script.ld」ウィンドウで、「linker_script.ld」タブをクリックします。

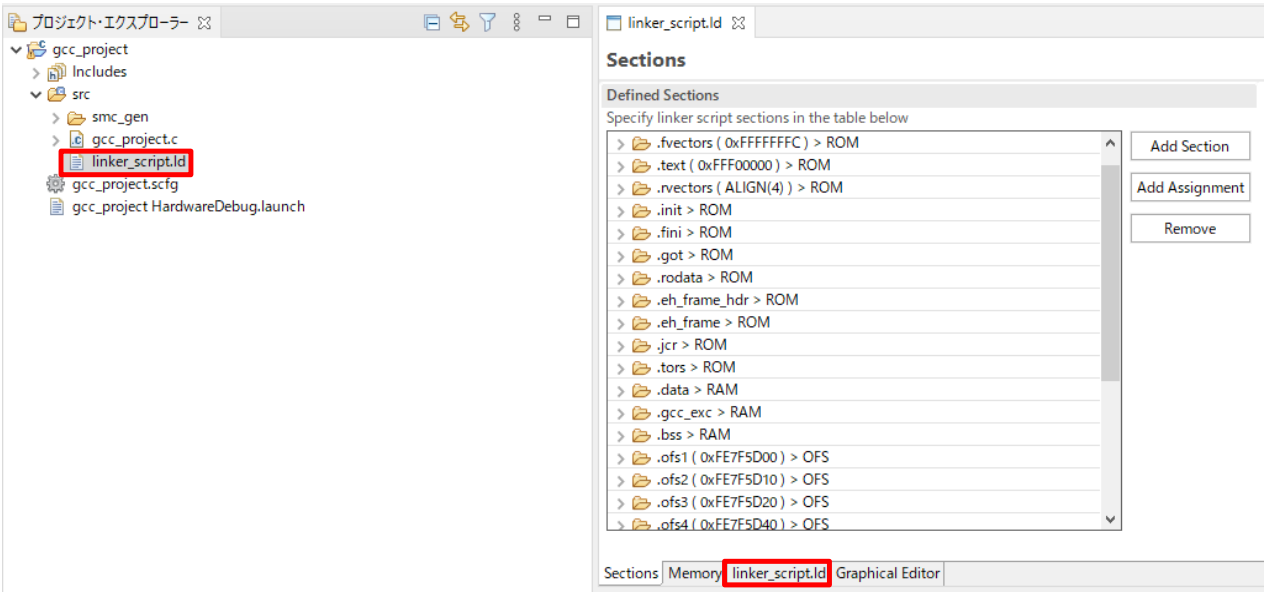


図 5-2 GCC for Renesas RX でのセクション設定 (1/2)

3) 以下の(a)～(d)のアドレスを、ユーザの環境に合わせて変更してください。

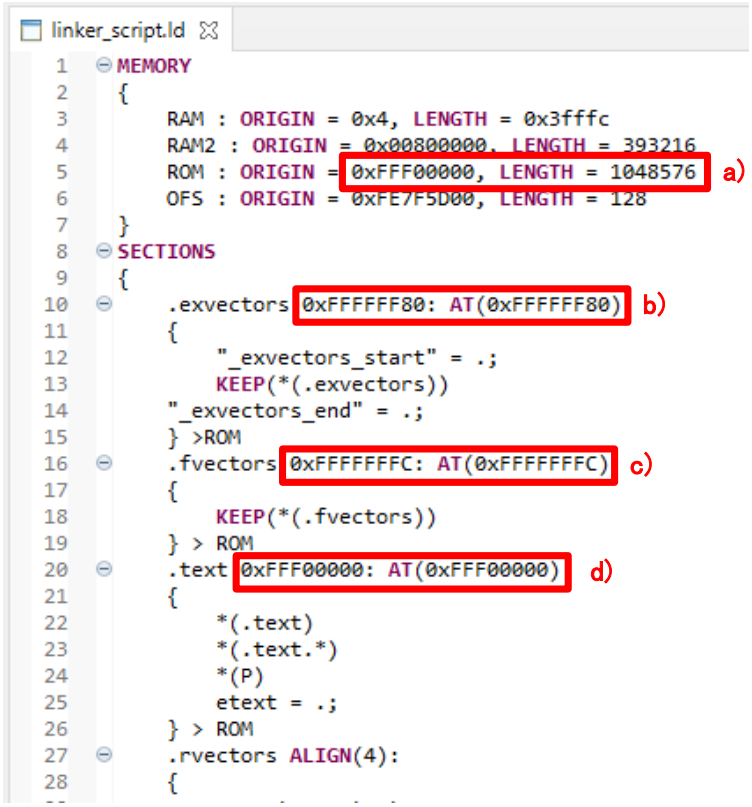


図 5-3 GCC for Renesas RX でのセクション設定 (2/2)

例) RX65N、デュアルモード、ブートローダのサイズ=64KB の場合の各設定値は以下のとおり。

記号	説明	ブートローダの設定	ユーザプログラムの設定
a)	コードフラッシュの開始アドレスと、コードフラッシュのサイズ	ORIGIN = 0xFFFF0000 LENGTH = 65536	ORIGIN = 0xFFF00300 LENGTH = 982272
b)	例外ベクタの配置アドレス	0xFFFFFF80	0xFFFEFF80
c)	リセットベクタの配置アドレス	0xFFFFFFFC	0xFFFEFFFC
d)	コードフラッシュの開始アドレス ＝ a)と同じアドレス	0xFFFF0000	0xFFF00300

5.2.2.3　フラッシュメモリ書き換えの設定

ユーザプログラムおよびブートプログラムに、フラッシュメモリ書き換えのための設定をする必要があります。設定の詳細は、以下アプリケーションノートを参照してください。

「フラッシュモジュール Firmware Integration Technology (R01AN2184)」の
「5.3.2 GCC for Renesas RX を使用する場合」

5.2.2.4　ビルド時のワーニングについて

本 FIT モジュールをビルドする際に、関数のスタック使用量が、-Wstack-usage オプションで指定したバイトサイズを超えた（“warning: stack usage is XXX bytes [-Wstack-usage=]”）ことを示すワーニングが出ます（デフォルト 100 バイト）。問題が有る場合は、ビルドオプションの設定を変更してください。

5.2.3　IAR C/C++ Compiler for Renesas RX を使用する場合

コンパイラとして IAR C/C++ Compiler for Renesas RX を使用する場合について示します。

5.2.3.1　コンパイルオプション

IAR Embedded Workbench for Renesas RX でのプロジェクトのオプション設定で、
出力コンバータ → 出力 の設定で、Motorola S-records を出力する設定にしてください。

その際に出力ファイルの拡張子をデフォルトの “*.srec” から “*.mot” に変更してください。

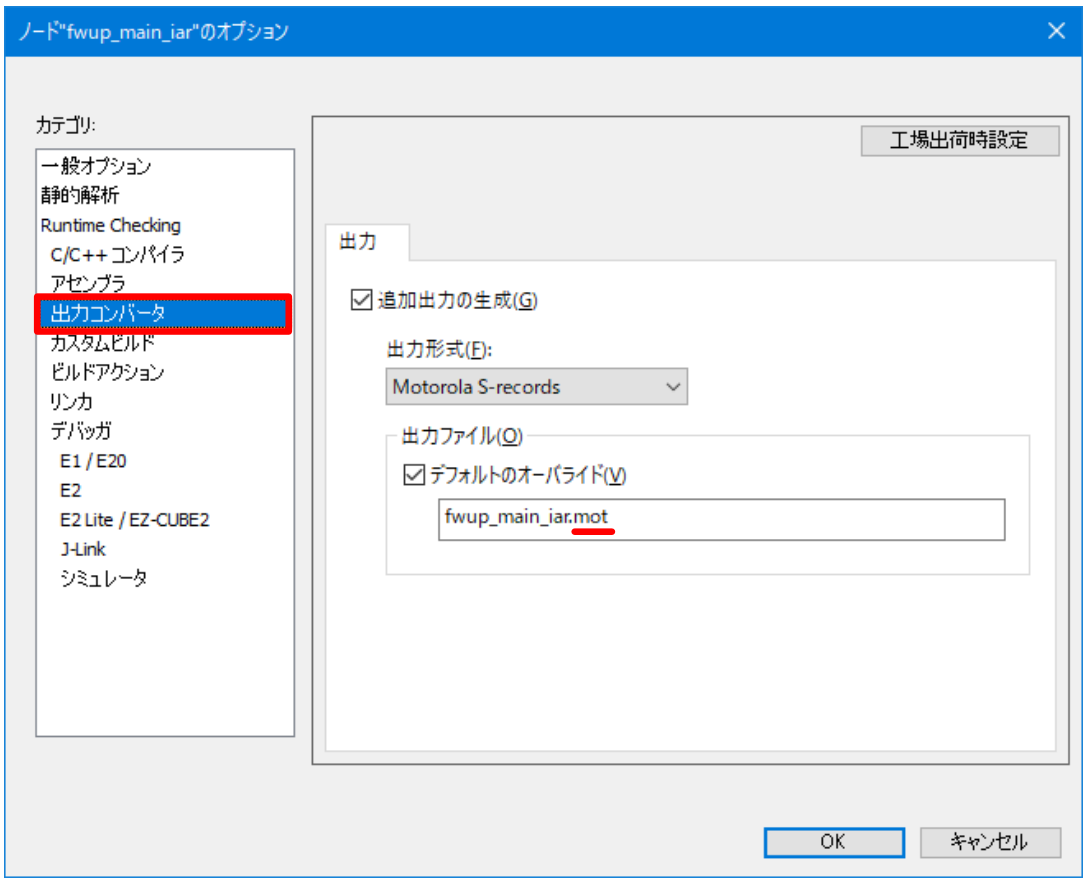


図 5-4 出力ファイルの拡張子の変更

5.2.3.2　フラッシュメモリ書き換えの設定

ユーザプログラムおよびブートプログラムに、フラッシュメモリ書き換えのための設定をする必要があります。設定の詳細は、以下アプリケーションノートを参照してください。

「フラッシュモジュール Firmware Integration Technology (R01AN2184)」の
「5.3.3 IAR C/C++ Compiler for Renesas RX を使用する場合」

5.2.3.3　フラッシュメモリ上の配置アドレスの変更

内蔵フラッシュ上の実行領域にブートローダとユーザプログラムを配置するために、5.2.3.2 で作成したリンカ設定ファイルを変更します。

1) 5.2.3.2 で作成したリンカ設定ファイル (*.icf) をエディタで開きます。

2) 以下の(a)～(c)のアドレスを、ユーザの環境に合わせて変更してください。
（例：RX65N ブートローダのリンカ設定ファイル）

```
define region RAM_region1  = mem:[from 0x00000004 to 0x0003FFFF];
define region RAM_region2  = mem:[from 0x00800000 to 0x0085FFFF];

define region RAM_region16 = mem:[from 0x00000004 to 0x00007FFF];
define region RAM_region24 = RAM_region1 | RAM_region2;
define region RAM_region32 = RAM_region1 | RAM_region2;

define region STANDBY_RAM  = mem:[from 0x000A4000 to 0x000A5FFF];

define region ROM_region16 = mem:[from 0xFFFF0000 to 0xFFFFFFFF];
define region ROM_region24 = mem:[from 0xFFFF0000 to 0xFFFFFFFF];  a)
define region ROM_region32 = mem:[from 0xFFFF0000 to 0xFFFFFFFF];

define region DATA_FLASH   = mem:[from 0x00100000 to 0x00107FFF];
```

```
place at address mem:0xFE7F5D00       { ro section .option_mem };
place at address mem:0xFFFFFFFC  b)   { ro section .resetvect };
place at address mem:0xFFFFFF80  c)   { ro section .exceptvect };
```

例) RX65N、デュアルモード、ブートローダのサイズ=64KB の場合の各設定値は以下のとおり。

記号	説明	ブートローダの設定	ユーザプログラムの設定
a)	コードフラッシュの開始アドレスと、終了アドレス	from 0xFFFF0000 to 0xFFFFFFFF	from 0xFFF00300 to 0xFFFEFFFF
b)	リセットベクタの配置アドレス	0xFFFFFFFC	0xFFFEFFFC
c)	例外ベクタの配置アドレス	0xFFFFFF80	0xFFFEFF80

5.3　FreeRTOS OTA 用データの格納先（RX65N-2MB のみ）

FreeRTOS の OTA 動作時に使用する PKCS11 データ（コード署名証明書など）の格納先を、コンフィグレーションオプションによりコードフラッシュ または データフラッシュから選択できます。本選択は、RX65N-2MB 製品のみ対応しています。

5.3.1　格納先の選択

以下のコンフィグレーションオプションにより、PKCS11 データの格納先を選択します。

但し、FreeRTOS の OTA 実行時に本設定は有効になります。また、ブートプログラムと FreeRTOS（OTA）プログラムにて設定値を同一にしてください。

FWUP_CFG_OTA_DATA_STORAGE

0：データフラッシュ（デフォルト）

1：コードフラッシュ

データフラッシュの格納領域は、0x00100000-0x00107FFF（32KB）

コードフラッシュの格納領域は、0xFFE00000-0xFFE07FFF（32KB）

5.3.2　セクション設定

PKCS11 データをコードフラッシュに置く場合は、FreeRTOS(OTA)プログラムのセクションを、以下の図を参照して設定してください。

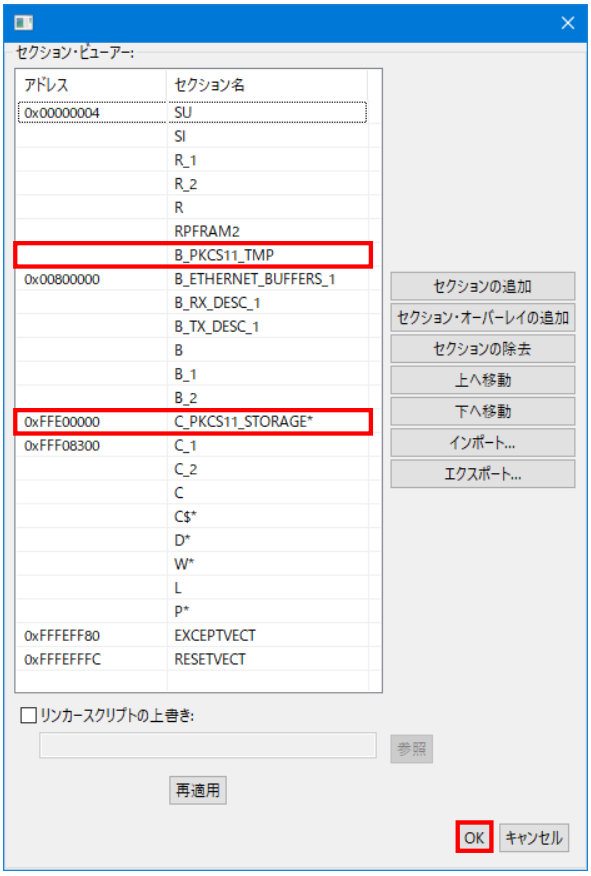


図 5-5 コードフラッシュを選択した場合のセクション設定

PKCS11 データをデータフラッシュに置く場合は、サンプルプログラムのセクション設定を参照ください。

5.3.3　コードフラッシュを選択した場合の.RSU ファイルへの変換方法

コードフラッシュを選択した場合の.RSU ファイルへの変換方法を以下に説明します。

FreeRTOS(OTA)のプログラムをビルドし、Renesas Image Generator で.mot ファイルから.RSU ファイルへ変換します。

Renesas Image Generator の[Initial Firm]タブで、Select MCU を RX65N Flash(Code=2MB, Data=0KB)を選択し、変換してください。

ファイル変換方法は、1 を参照ください。

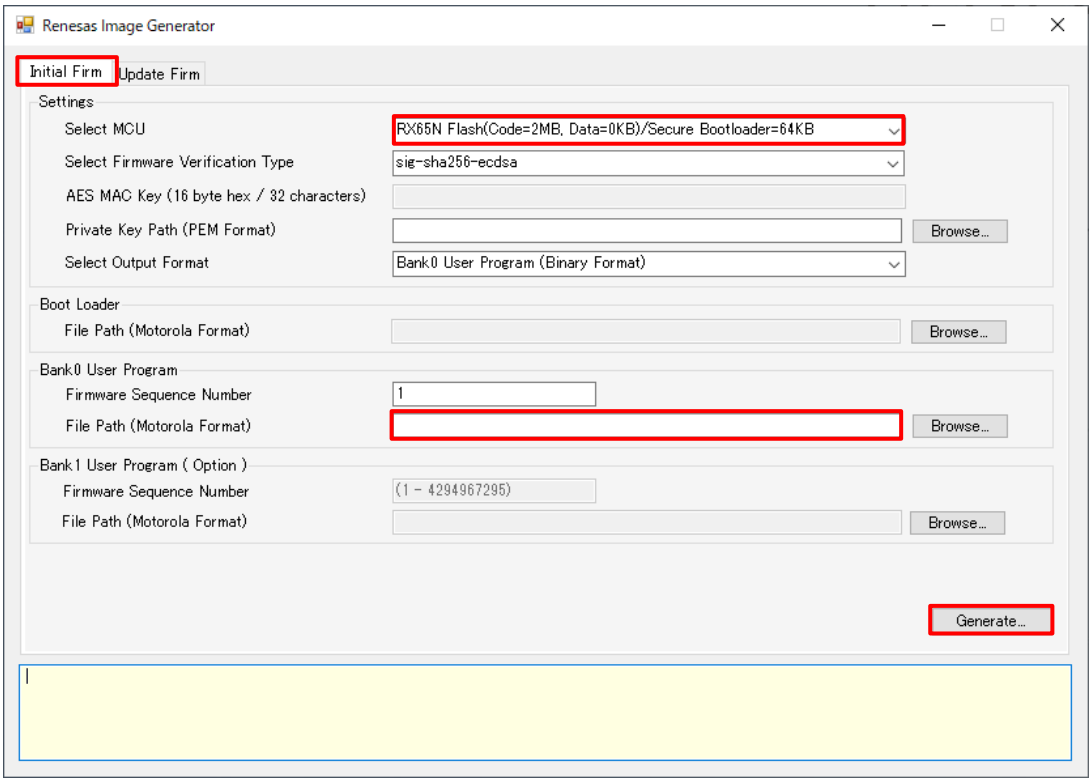


図 5-6 コードフラッシュ選択時の.RSU ファイルへの変換(Initial Firm)

Renesas Image Generator の[Update Firm]タブで、Select MCU を RX65N Flash(Code=2MB, Data=0KB)を選択し、変換してください。

ファイル変換方法は、1 を参照ください。

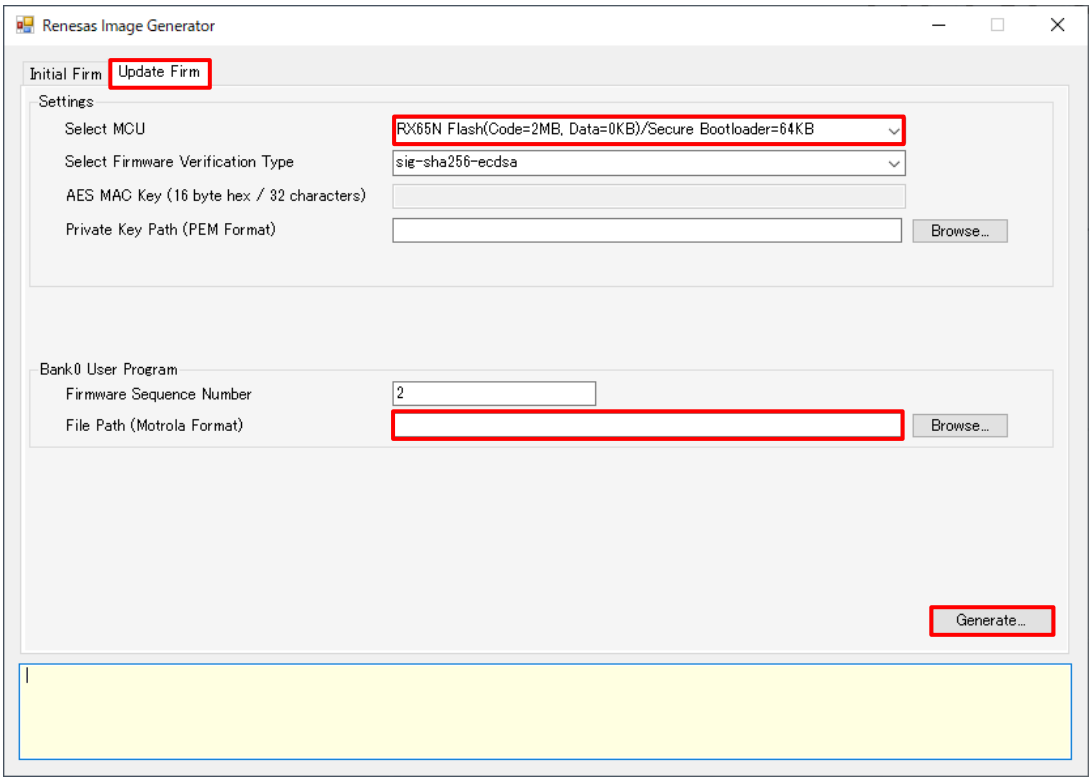


図 5-7 コードフラッシュ選択時の.RSU ファイルへの変換(Update Firm)

5.4　Image Generator によるファームウェアアップデートイメージの構成

ファームウェアアップデートのメモリ構成は、デュアルモードとリニアモードで異なります。

5.4.1　デュアルモードのメモリ構成

デュアルモードの場合、バンク 0 のエリアに初期ファームウェア（ブートローダとユーザプログラム）を配置し、バンク 1 エリアに更新用のユーザプログラムを配置します。なお、ユーザプログラムの上部には、Image Generator により RSU ヘッダを付加します。（表 5-7 参照）

デュアルモードによるファームウェアアップデートの動作仕様については、1.3.1 をご参照ください。以下の図は、RX65N（2MB）のコードフラッシュメモリにファームウェアアップデートを構成する場合の例となります。

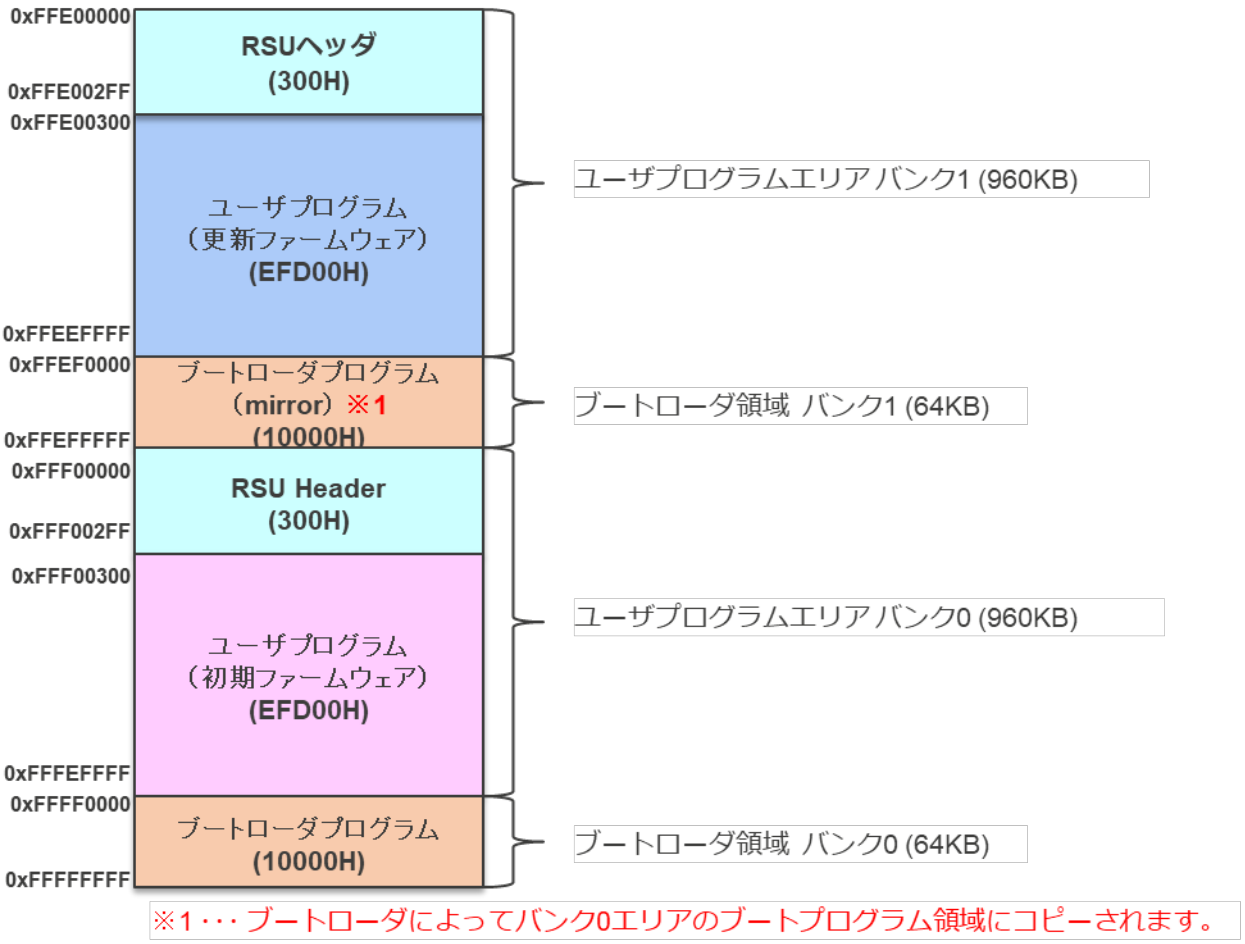


図 5-8 RX65N コードフラッシュ（2MB）ファームウェアアップデートのメモリ構成

5.4.2　リニアモード（半面書き換え方式）のメモリ構成

リニアモード（半面書き換え方式）の場合、ブートローダをコードフラッシュの後方 64KB に配置し、残りの領域の上位半分（エリア 1）を更新用ユーザプログラム、下位半分（エリア 0）を初期用ユーザプログラムの使用領域として割り当てます。なお、ユーザプログラムの上部には、Image Generator により RSU ヘッダを付加します。（表 5-7 参照）

リニアモード（半面書き換え方式）によるファームウェアアップデートの動作仕様については、1.3.2.1 を参照ご参照ください。

以下の図は、RX66T（512KB）のコードフラッシュメモリにファームウェアアップデートを構成する場合の例となります。

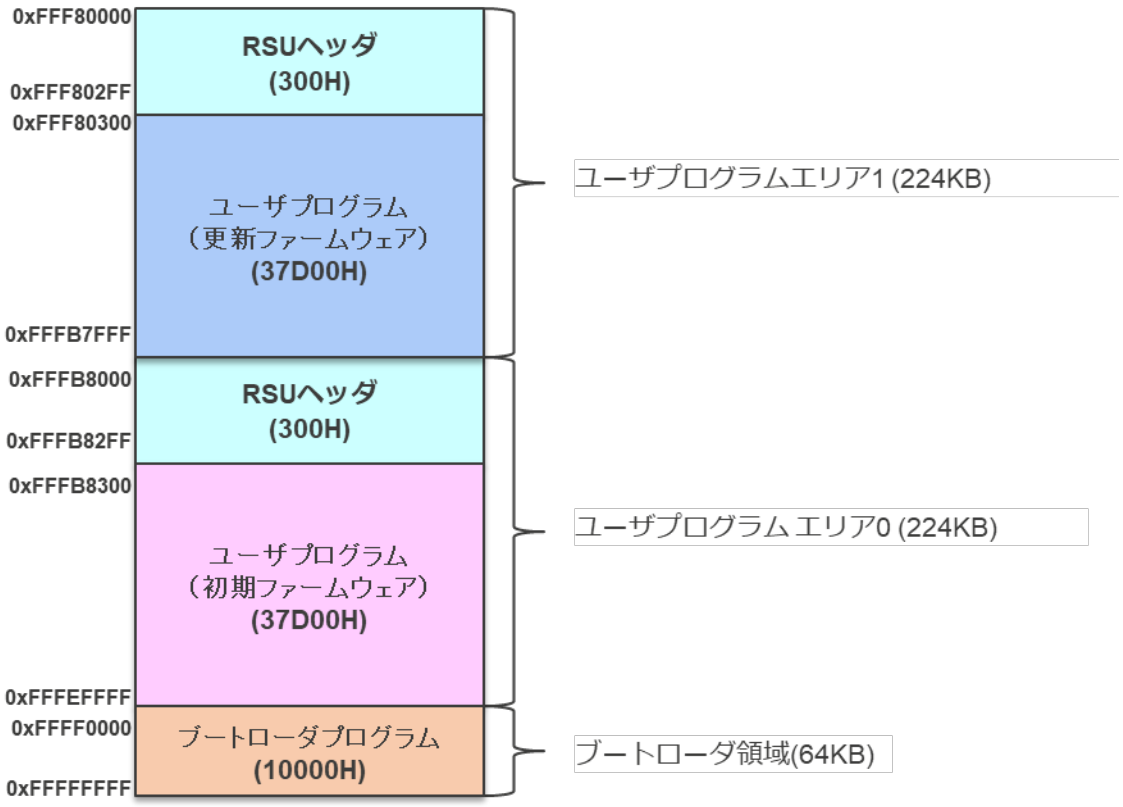


図 5-9 RX66T コードフラッシュ（512KB）ファームウェアアップデートのメモリ構成

5.4.3　リニアモード（全面書き換え方式）のメモリ構成

リニアモード（全面書き換え方式）の場合、ブートローダをコードフラッシュの後方 64KB に配置し、残りの領域をユーザプログラム（初期ファームウェアと更新用ファームウェア）として割り当てます。なお、ユーザプログラムの上部には、Image Generator により RSU ヘッダを付加します。（表 5-7 参照）

リニアモード（全面書き換え方式）によるファームウェアアップデートの動作仕様については、1.3.2.2 をご参照ください。

以下の図は、RX66T（512KB）のコードフラッシュメモリにファームウェアアップデートを構成する場合の例となります。

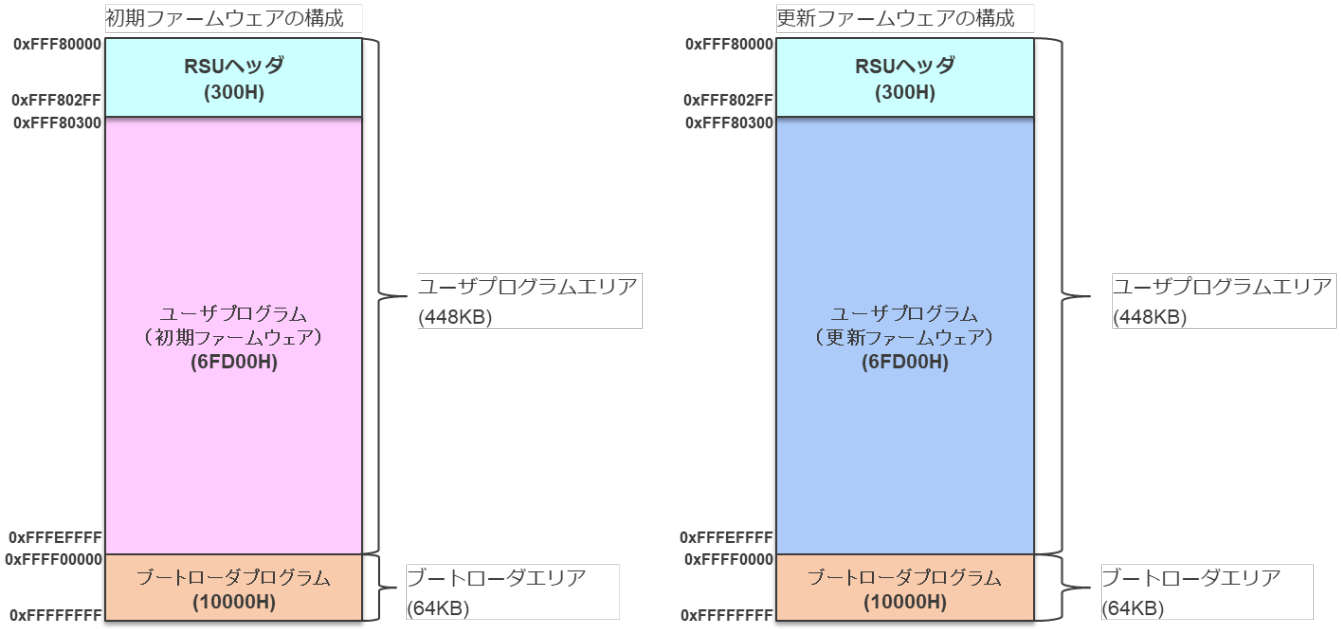


図 5-10 RX66T コードフラッシュ（512KB）ファームウェアアップデートのメモリ構成

5.5　Image Generator によるファームウェアアップデートイメージ詳細

Image Generator は、ビルドして生成されたブートローダやユーザプログラムをもとに 5.4 に示したファームウェアアップデートのメモリ構成をするためのイメージを生成します。なお、ユーザプログラム（.mot）ファイルは、ブートローダがユーザプログラムの署名検証などに必要な情報がないため、以下の RSU ヘッダ（0x300h）を生成し、ユーザプログラムの先頭に付加します。

表 5-7 RSU ヘッダ詳細

オフセット	項目	長さ (Byte)	説明
0x00000000	Magic Code	7	マジックコード（”Renesas”）
0x00000007	Image Flags	1	ブートローダがユーザプログラムの検証を行うための状態フラグ
0x00000008	Firmware Verification Type	32	ファームウェア検証方式を指定するための識別子（Image Generator で指定）
0x00000028	Signature size	4	ファームウェア検証に用いる署名値や MAC 値やハッシュ値などのデータサイズ
0x0000002C	Signature	256	ファームウェア検証に用いる署名値や MAC 値やハッシュ値
0x0000012C	Data Flash Flag	4	データフラッシュ用のデータが含まれるか否かを表すフラグ
0x00000130	Data Flash Start Address	4	データフラッシュ開始アドレス
0x00000134	Data Flash End Address	4	データフラッシュ終了アドレス
0x00000138	Reserved(0x00)	200	予約領域
0x00000200	Sequence Number	4	シーケンス番号（Image Generator に指定した値を設定）
0x00000204	Start Address	4	ユーザプログラムを書き込むシリアルフラッシュの開始ドレス（Image Generator が自動で設定）
0x00000208	End Address	4	ユーザプログラムを書き込むシリアルフラッシュの終了ドレス（Image Generator が自動で設定）
0x0000020C	Execution Address	4	予約領域
0x00000210	Hardware ID	4	Image Generator で「Select MCU」を選択することによって決定されるハードウェア ID でブートローダによって使用。
0x00000214	Reserved(0x00)	236	予約領域

5.5.1　ブートローダとユーザプログラム（初期ファームウェア）のイメージ詳細

ファームウェアアップデートの初期設定をブートローダとユーザプログラム（初期ファームウェア）が書き込まれた状態（デュアルモードは図 1-4④、リニアモード（半面書き換え方式は、図 1-6②、全面書き換え方式は、図 1-8②）から開始する場合、ブートローダとユーザプログラム（初期ファームウェア）をビルドして生成された mot ファイルと、4.2.1.2 で生成した署名検証用の秘密鍵を Image Generator に入力し、mot ファイルを生成します。生成された mot ファイルは、フラッシュライタなどで MCU に書きみます。

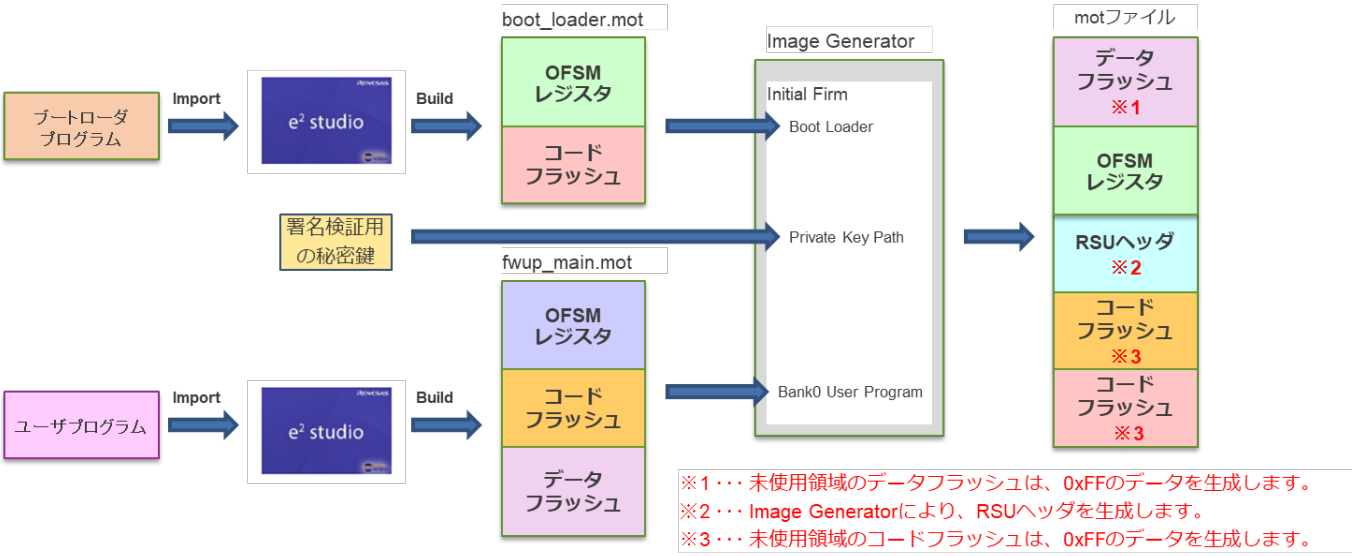


図 5-11 ブートローダとユーザプログラム（初期ファームウェア）のイメージ詳細

5.5.2　ユーザプログラム（更新ファームウェア）の RSU イメージ詳細

更新ファームウェアを生成する場合、ユーザプログラム（更新ファームウェア）をビルドして生成された mot ファイルと、4.2.1.2 で生成した署名検証用の秘密鍵を Image Generator に入力し、RSU ファイルを生成します。生成された RSU ファイルは、ブートローダが書き込むユーザプログラム（更新ファームウェア）となります。

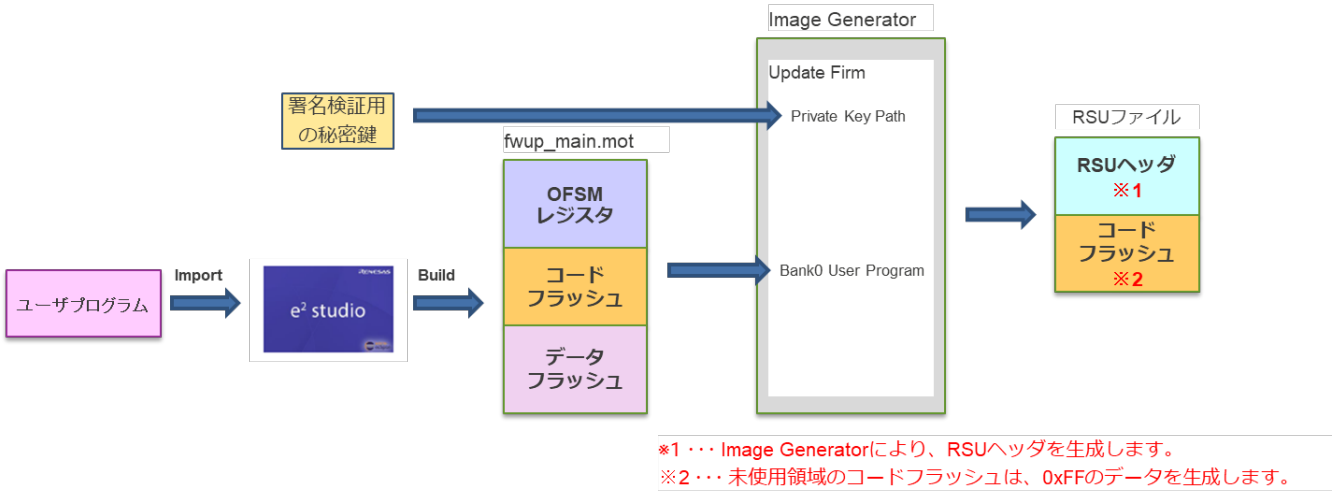


図 5-12 ユーザプログラム（更新ファームウェア）のイメージ詳細

5.5.3　ユーザプログラム（初期ファームウェア）の RSU イメージ詳細

ファームウェアアップデートの初期設定をブートローダが書き込まれた状態（デュアルモードは図 1-4①、リニアモード（半面書き換え方式は、図 1-6①、全面書き換え方式は、図 1-8①）から開始する場合、ユーザプログラム（初期ファームウェア）をビルドして生成された mot ファイルと、4.2.1.2 で生成した署名検証用の秘密鍵を Image Generator に入力し、RSU ファイルを生成します。生成された RSU ファイルは、ブートローダが書き込むユーザプログラム（初期ファームウェア）となります。

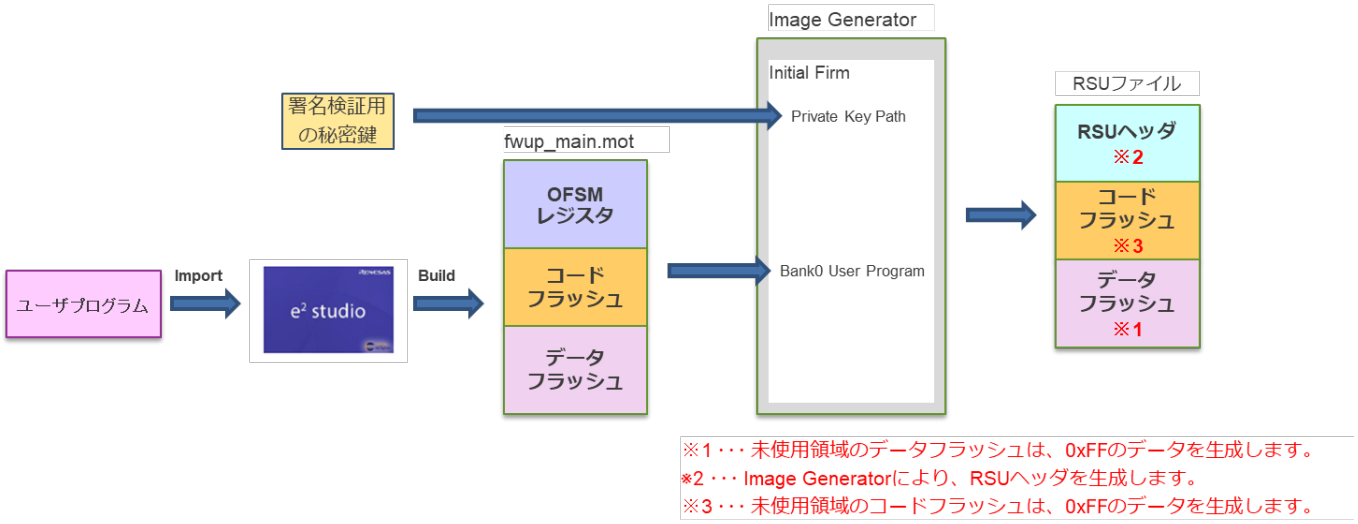


図 5-13 ユーザプログラム（初期ファームウェア）のイメージ詳細

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021.4.16	-	初版発行
1.01	2021.6.21	表紙	動作確認デバイスに RX72N グループ、RX66T グループ、RX130 グループを追加。
		5	「1.概要」の内容を見直し
		13	「2.6 コンパイル時の設定」に設定するオプションを追記。 - FWUP_CFG_SERIAL_TERM_SCI - FWUP_CFG_SERIAL_TERM_SCI_BITRATE - FWUP_CFG_SERIAL_TERM_SCI_INTERRUPT_PRIORITY 説明の見直し
		16	「2.6.1 RX130 環境でのコンパイル時の注意事項」を追記。
		18	「2.8 引数」に OTA のファイルコンテキストの説明を追記。
		19	該当箇所がないため、「2.12 for 文、while 文、do while 文について」の記載を削除
		25	「3.13 R_FWUP_ResetDevice 関数」に Special Notes を追記。
		33	動作確認環境（Rev.1.01）を追記。
1.02	2021.10.29	表紙	動作確認デバイスに RX671 グループを追加。
		5	「1.2 ファームウェアアップデートモジュールの構成」で、OS なしのシステムと FreeRTOS（OTA）のシステムにおけるファームウェアアップデートモジュールとブートローダモジュールの説明を削除
		6	図 1-1 のシリアル通信モジュールとバイト型キューバッファモジュールの接続を修正
		8	表 1-2 に RX671 グループを追加
		11	表 1-3 の説明を変更 • ブートローダモジュールの説明を変更 ブートローダモジュールで R_FWUP_Open, R_FWUP_Close を使用するよう変更
		17	「2.7 コードサイズ」に GCC for Renesas RX でのコードサイズを追記。
			「2.7 コードサイズ」の GCC for Renesas RX に、RX65N の ”boot_loader Project” と “aws_demos Project” 以外のコードサイズを追記。
		19	ブートローダの使用 ROM、RAM サイズの記述を追加
		21	表 3-1 の説明にブートローダモジュールを追記
			表 3-2 の説明にブートローダモジュールを追記
		34	「5.1 動作確認環境」の表 5-1 を Rev.1.02 の動作確認環境に修正。
			デモプロジェクトで使用した FIT モジュールのバージョン一覧として表 5-2、表 5-3 を追記
		35	「5.2 コンパイラ依存の設定」を追加。
		36	表 5-3 に、RX65N 以外の GCC 環境のデモプロジェクトで使用した FIT モジュールのバージョンを追記
		39	5.2.2.1 コンパイルオプションにブートローダをデバッグする際の最適化レベル設定を追記

V1.03	2021.12.28	表紙	関連アプリケーションノートに RX スマート・コンフィグレータ ユーザーガイド: IAREW 編(R20AN0535)を追加
			ターゲットコンパイラに IAR C/C++ Compiler for RX を追加
		11	表 1-3 の R_FWUP_SoftwareReset の誤記を修正 R_FWUP_ActiveNewImage から R_FWUP_ActivateNewImage に修正
		13	表 2-1 の FWUP_CFG_IMPLEMENTATION_ENVIRONMENT 設定値のうち、3 に注 1 を追加
		14	表 2-2 に設定しても動作しない未対応の組み合わせの注意を追加
		17	表 2-4 に IAR C/C++ Compiler for RX 環境でのコードサイズを追加
		18	表 2-4 のサイズ計測時の IAR C/C++ Compiler for RX の条件を追加
		19	表 2-5 に IAR C/C++ Compiler for RX 環境でのブートローダの使用 ROM、RAM サイズとサイズ計測時の IAR C/C++ Compiler for RX の条件を追加
		21	「2.10 FIT モジュールの追加方法」に IAR Embedded Workbench for Renesas RX 環境での FIT モジュールの追加方法を追加
		23	R_FWUP_Close 関数の戻り値を修正
		24	R_FWUP_Operation 関数の戻り値を修正
		25	・R_FWUP_SetEndOfLife 関数の説明を追加 ・R_FWUP_SetEndOfLife 関数の戻り値を修正
		26	R_FWUP_SecureBoot 関数の説明と返却値の説明を追加
		28	R_FWUP_CloseFile 関数の戻り値を修正
		29	関数名の誤記を修正 R_FWUP_ActiveNewImage から R_FWUP_ActivateNewImage に修正
		30	・R_FWUP_SetPlatformImageState 関数の戻り値を修正 ・R_FWUP_CheckFileSignature 関数の戻り値を修正
		35	src/main.c のソースイメージを変更
		38	表 5-2 に IAR の動作確認環境を追加
		40	表 5-5 に IAR C/C++ Compiler for RX 環境での動作確認時の FIT モジュールのバージョン一覧を追加
		45	「5.2.3 IAR C/C++ Compiler for Renesas RX」の章を追加
V1.04	2022.5.24	表紙	動作確認デバイスに RX140 グループを追加
		8	表 1-2 に RX140 グループを追加
			「1.3.1 デュアルモードを使用したファームウェアアップデート動作」の説明文を修正
		14	表 2-1 を修正。 FWUP_CFG_USE_SERIAL_FLASH_FOR_BUFFER、FWUP_CFG_SIGNATURE_VERIFICATION を削除。 FWUP_CFG_PRINTF_DISABLE を FWUP_CFG_BOOTLOADER_LOG_DISABLE に名称変更。 FWUP_CFG_OTA_DATA_STORAGE、FWUP_CFG_LOG_LEVEL を追加。
		17	2.6.1 章の説明に RX140 を追加
		20	表 2-5 に RX140 の ROM、RAM サイズを追加
		21	表 2-6 OTA のファイルコンテキストを変更
		27	R_FWUP_Abort の戻り値を変更
		27	R_FWUP_CreateFileForRx の戻り値を変更
		28	R_FWUP_CloseFile の戻り値を変更
		28	R_FWUP_ActivateNewImage の戻り値を変更

		29	R_FWUP_ResetDevice の戻り値を変更
		29	R_FWUP_SetPlatformImageState の戻り値を変更
		30	R_FWUP_CheckFileSignature の戻り値を変更
		33	図 4-2 を変更
		34	図 4-3 を変更
		35,36	LOG の更新
		37,38	表 5-1 と表 5-2 と表 5-3 の動作確認環境を変更
		39-41	表 5-4、表 5-5、表 5-6 の FIT モジュールのバージョン一覧を変更
		49	「5.3 FreeRTOS OTA 用データの格納先（RX65N-2MB のみ）」を追加
V1.05	2022.8.10	表紙	動作確認デバイスに RX660 グループを追加
		8	表 1-2 に RX660 グループを追加
		12	表 1-3 に R_FWUP_ResetDevice の説明を追加
		14	表 2-1 の FWUP_CFG_BOOTLOADER_LOG_DISABLE に説明を追加
		18	コードサイズ測定時のビルド条件を変更
		19	表 2-5 に RX660 の ROM、RAM サイズを追加
		19	ブートローダが使用する ROM、RAM サイズ測定時のビルド条件を変更
		23	R_FWUP_Open の Return Values から FWUP_ERR_LESS_MEMORY と FWUP_ERR_IMAGE_STATE を削除
		26	R_FWUP_SecureBoot の Return Value から FWUP_ERR_ALREADY_OPEN を削除し、FWUP_ERR_NOT_OPEN の説明を追加
		29	R_FWUP_ResetDevice の Special Notes に説明を追加
		31,32	図 4-1 デモプロジェクト一覧を追加
		33	4.2 章の説明を修正
		34	デモプログラムの説明を追加
		34	4.2.1 の ②Base64 でコード関数追加方法の削除と ③Key ファイルのファイル取得の説明を削除
		39-41	RX660 の動作確認環境を追加
		42	表 5-7 に RX660 の説明追加と RX65N,RX671,RX72N の FIT モジュールのバージョンを更新
		43	表 5-8 に RX660 の説明追加と RX65N,RX671,RX72N の FIT モジュールのバージョンを更新
		44	表 5-9 に RX660 の説明追加と RX65N,RX671,RX72N の FIT モジュールのバージョンを更新
		53	RSU ファイルへの変換方法の誤記を修正
		54	RSU ファイルへの変換方法の誤記を修正
V1.06	2022.12.28	7	OS 無し通信ドライバ無しのシステム向けのモジュールに関する文言を追加
		8	図 1-2 OS 無し(モジュール外通信制御)のシステムにおけるファームウェアアップデートのシステム構成 を追加
		13,14	半面書き換え方式のファームウェアアップデートを追加
		14,15	全面書き換え方式のファームウェアアップデートを追加
		16	1.4 OS 無しのシステムにおけるファームウェアアップデートの通信制御 を追加
		19	表 **1-3** API 関数一覧 に OS･通信ドライバレスと Azure ADU の項目を追加

		22	表 2-1 コンフィグレーション設定に FWUP_CFG_IMPLEMENTATION_ENVIRONMENT にメンバ追加と 4 の説明を変更
		24	表 2-2 コンパイル時設定の設定可能組み合わせ の FWUP_CFG_IMPLEMENTATION_ENVIRONMENT にメンバ追加と 4 の説明を変更
		25	表 2-3 有効組み合わせのマクロ値の 8 の内容を変更し、9～11 を削除
		26	表 **2-4** コードサイズ の RX66T に fwup_main_woSciDrv を追加
		30	2.10 FIT モジュールの追加方法 に、「(4) 不要なモジュールを削除する」を追加
		33	3.3 R_FWUP_Initialize 関数 を追加
		34	3.5 R_FWUP_PutFirmwareChunk 関数 を追加 3.7 R_FWUP_DirectUpdate 関数 を追加
		36	3.9 R_FWUP_SecureBoot 関数に全面書き換え方式向けの説明を追加
		40	3.20 R_FWUP_CheckFileSignature 関数　[OS 無し時に使用]を追加
		42	図 **4–1** デモプロジェクト一覧（1/2）を更新
		44	4.2 デモプロジェクト を追加
		45	4.3 Image Generator によるファームウェアイメージファイルの変換を追加
		49	4.4 を見直し
		63	表 5-1 に Rev.1.06 の動作確認環境 (CC-RX) を追加 旧確認環境を削除。
		63	表 5-2 に Rev.1.06 の動作確認環境 (GCC) を追加 旧確認環境を削除。
		64	表 5-3 に Rev.1.06 の動作確認環境 (IAR) を追加 旧確認環境を削除。
		65	表 5-4 FIT モジュールのバージョン一覧(CC-RX) の FIT モジュールのバージョン更新と、RX66T に fwup_main_woSciDrv を追加
		66	表 5-5 FIT モジュールのバージョン一覧(GCC)の FIT モジュールのバージョンを更新
		67	表 5-6 FIT モジュールのバージョン一覧(IAR)の FIT モジュールのバージョンを更新
		78	5.4 Image Generator によるファームウェアアップデートイメージの構成 を追加
		80	5.5 Image Generator によるファームウェアアップデートアップデートイメージ詳細 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。