RX100／RX200 シリーズ

スタートアッププログラム保護機能とシリアル通信を使用したファームウェアアップデート方法

要旨

本アプリケーションノートでは、RX100／RX200 シリーズのスタートアッププログラム保護機能を使用したマイコン内蔵コードフラッシュメモリのアップデート方法について説明します。サンプルプログラムの制御とデータの転送にシリアル通信を使用します。

本アプリケーションノートでは、ファームウェアプログラム、およびファームウェアアップデートプログラムを以下のように定義します。

ファームウェアプログラム：コードフラッシュメモリのユーザ領域に書き込むプログラム

ファームウェアアップデートプログラム：ファームウェアプログラムを書き換えるプログラム

本アプリケーションノートのサンプルプログラムは、ファームウェアアップデートプログラムと、ファームウェアアップデートプログラムの動作を確認するためのファームウェアプログラムがあります。

また、本書ではファームウェアアップデートプログラムをスタートアッププログラム保護機能のデフォルト領域に配置することとして説明します。ファームウェアアップデートプログラムを代替領域に配置する場合、本書の「デフォルト領域」と「代替領域」を読み替えてください。

対象デバイス

・RX230 グループ、RX231 グループ

・RX110 グループ、RX111 グループ、RX113 グループ、RX130 グループ、RX140 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合わせて変更し、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。併せて参照してください。

- Firmware Integration Technology ユーザーズマニュアル（R01AN1833）
- RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology（R01AN1723）
- RX ファミリ CS+に組み込む方法 Firmware Integration Technology（R01AN1826）
- 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 バイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology（R01AN1683）

目次

1.　概要

1.1　　本アプリケーションノートについて

本アプリケーションノートでは、スタートアッププログラム保護機能を使用して安全にコードフラッシュメモリをアップデートする方法を解説します。

スタートアッププログラム保護機能のデフォルト領域と定数データ領域にファームウェアアップデートプログラムを配置します。ホスト PC からシリアル通信を使用してファームウェアアップデートプログラムを制御し、コードフラッシュメモリをアップデートします。マイコンの動作モードはシングルチップモードです。書き換え用データとしてモトローラ S-Record フォーマットのデータを使用します。データ転送プロトコルは XMODEM/SUM を使用します。ホスト PC のシリアル通信ソフトウェアは XMODEM/SUM 転送機能を持つターミナルソフトウェアを使用してください。

表 1.1 に使用する周辺機能と用途を、図 1.1 に動作概要を示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
フラッシュメモリ	コードフラッシュメモリの書き換え
シリアルコミュニケーションインタフェース	ホスト PC との調歩同期式シリアル通信

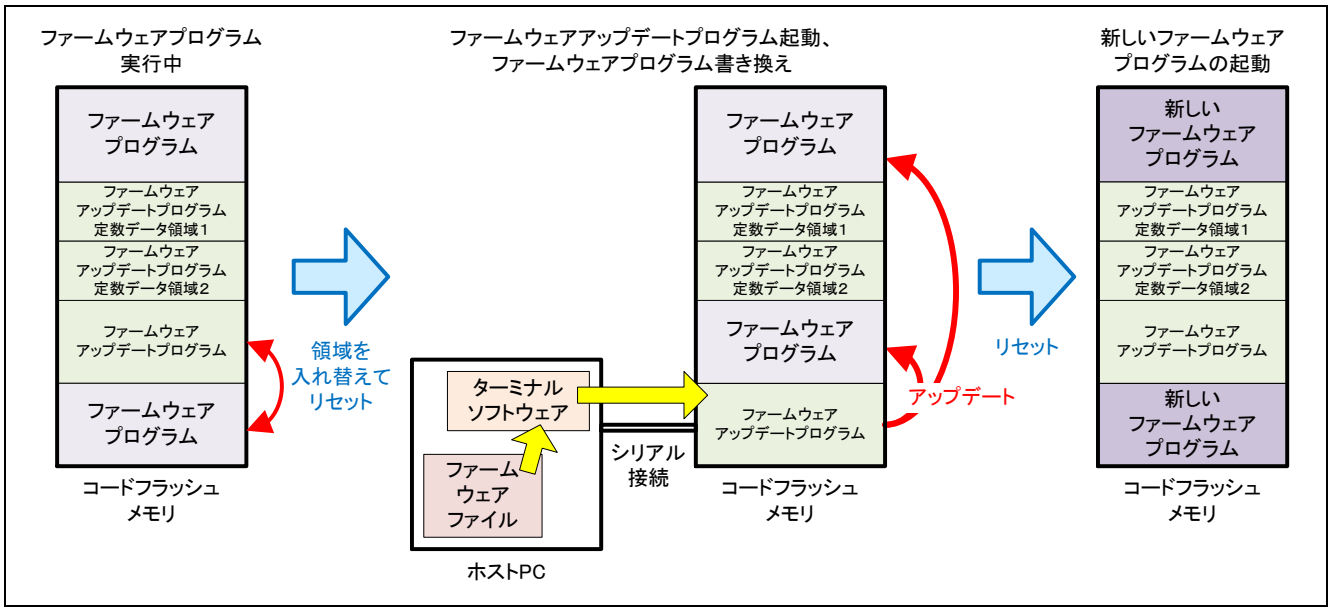


図 1.1　動作概要

本アプリケーションノートのサンプルプログラムは Firmware Integration Technology（以下、FIT と称す）対応モジュールを使用して周辺機能を制御します。本アプリケーションノートで使用する FIT モジュールを以下に示します。

- ボードサポートパッケージ（以下、BSP と称す）
- フラッシュメモリ（以下、フラッシュ FIT モジュールと称す）
- シリアルコミュニケーションインタフェースモジュール（以下、SCI FIT モジュールと称す）
- バイト型キューバッファ（BYTEQ）モジュール

1.2　動作環境

本アプリケーションノートのサンプルプログラムは、表 1.2 の環境で動作を確認しています。

表 1.2　動作環境

項目	内容
使用マイコン	R5F51305ADFN（RX130 グループ） R5F51406BDFN（RX140 グループ） R5F52318ADFP（RX231 グループ）
使用ボード	Renesas Starter Kit for RX130（製品型名: RTK5005130C00000BE） Renesas Starter Kit for RX140 Renesas Starter Kit for RX231（製品型名: R0K505231C000BE）
統合開発環境	ルネサスエレクトロニクス製 e^2 studio Version 2022-04
	ルネサスエレクトロニクス製 CS+ V.8.07.00
C 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 V.3.04.00
	コンパイラオプション –lang = C99
フラッシュプログラマ	Renesas Flash Programmer V.3.09.00
エミュレータ	E2 Lite
エンディアン	リトルエンディアン

1.3　モジュール構成

図 1.2 にサンプルプログラムのモジュール構成を、表 1.3 にサンプルプログラムに組み込む FIT モジュールの一覧を示します。

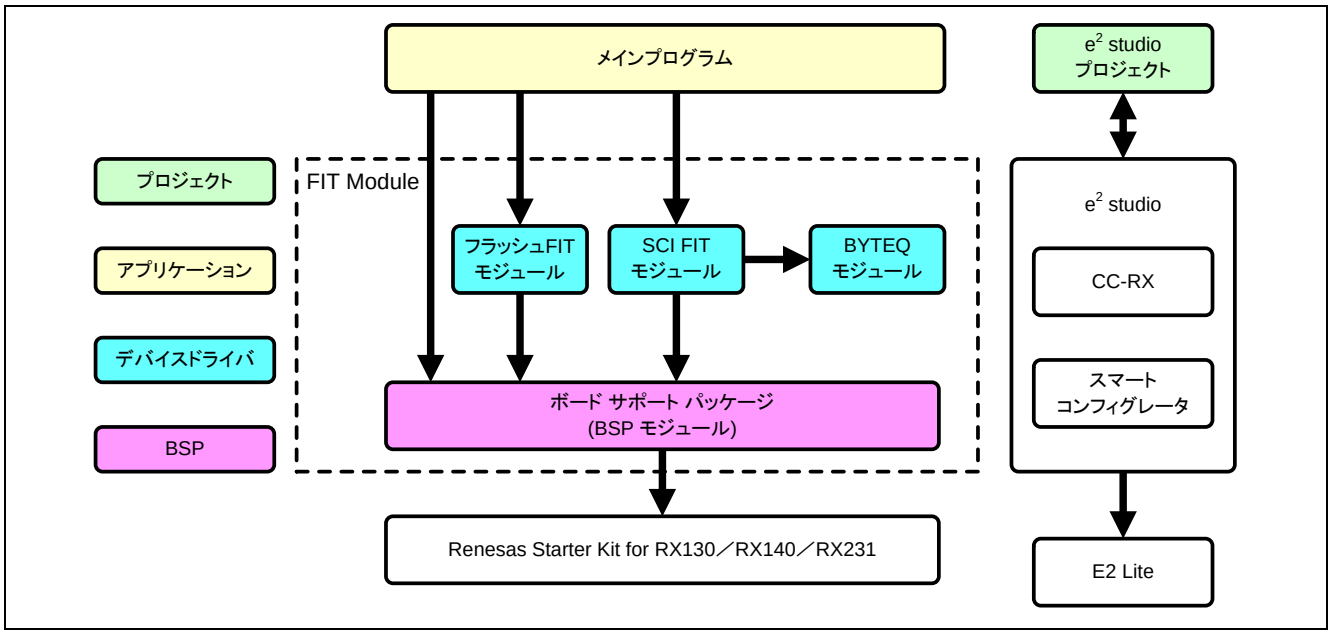


図 1.2　モジュール構成

表 1.3　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 バイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology （R01AN1683）	r_byteq
アプリケーション	メインプログラム	src

1.4　ファイル構成

図 1.3 に本アプリケーションノートのファイル構成を示します。

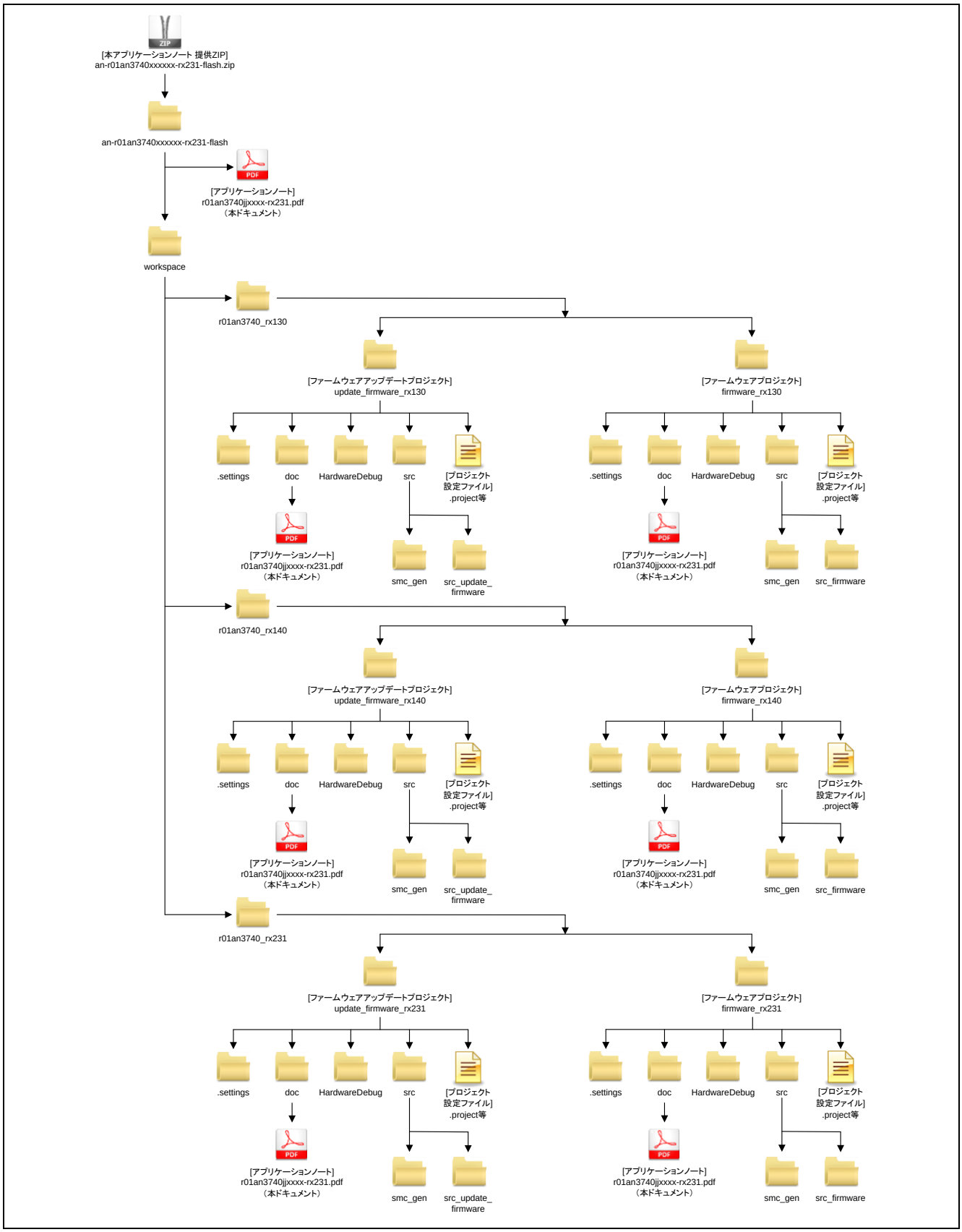


図 1.3　ファイル構成

本アプリケーションノート提供 ZIP ファイルを解凍すると、同名フォルダが作成され、その中に各フォルダやファイルが入っています。

「r01an3740_rx130／r01an3740_rx140／r01an3740_rx231」フォルダの下にある「ファームウェアアップデートプロジェクト（update_firmware_rx130／update_firmware_rx140／update_firmware_rx231）」および「ファームウェアプロジェクト（firmware_rx130／firmware_rx140／firmware_rx231）」は、本アプリケーションノートを構築したプロジェクトです。e² studio のワークスペースにインポートすることでアプリケーションノートの動作確認が可能です。

1.5　プロジェクトについて

本アプリケーションノートには、本アプリケーションノートをビルドおよび評価するための e² studio 用のプロジェクトが付属しています。プロジェクトは、ビルド設定を保存した「ビルド構成」と、デバッグ設定を保存した「デバッグ構成」を登録しています。

表 1.4 に、プロジェクトに登録してあるビルド構成とデバッグ構成の一覧を、表 1.5 にターゲット固有の設定を示します。

表 1.4　プロジェクト設定

	構成例	説明
ビルド構成	HardwareDebug（Debug on hardware）	デバッグ情報付きのロードモジュールを生成するための構成です。
デバッグ構成	HardwareDebug（E2 Lite）	「HardwareDebug（Debug on Hardware）」で生成したロードモジュールを使用して、E2 Lite エミュレータ経由でのハードウェアデバッグを行います。

表 1.5　ターゲット固有の設定

項目	設定
ツールチェーン・バージョン	V3.04.00
デバッグ・ハードウェア	E2 Lite（RX）
データ・エンディアン	Little-endian data
ターゲットの選択（RX130 グループ）	R5F51305ADFN（RX130 LFQFP 80pin)
ターゲットの選択（RX140 グループ）	R5F51406BDFN（RX140 LFQFP 80pin）
ターゲットの選択（RX231 グループ）	R5F52318ADFP（RX231 LQFP 100pin)
Renesas RTOS サポート	None

2. 開発環境の入手

2.1 e² studio の入手方法

以下の URL にアクセスし、e² studio をダウンロードしてください。

https://www.renesas.com/ja-jp/products/software-tools/tools/ide/e2studio.html

なお、本ドキュメントは e² studio V2022-04 以降を使用することを前提としています。V2022-04 よりも古いバージョンを使用した場合、e² studio の一部機能を使用できない可能性があります。ダウンロードする場合、ホームページに掲載されている最新バージョンの e² studio を入手してください。

2.2 CS+の入手方法

以下の URL にアクセスし、CS+をダウンロードしてください。

https://www.renesas.com/jp/ja/software-tool/cs.html

2.3 コンパイラパッケージの入手方法

以下の URL にアクセスし、RX ファミリ用 C/C++コンパイラパッケージをダウンロードしてください。

https://www.renesas.com/jp/ja/software-tool/cc-compiler-package-rx-family.html

2.4 Renesas Flash Programmer の入手方法

以下の URL にアクセスし、Renesas Flash Programmer をダウンロードしてください。

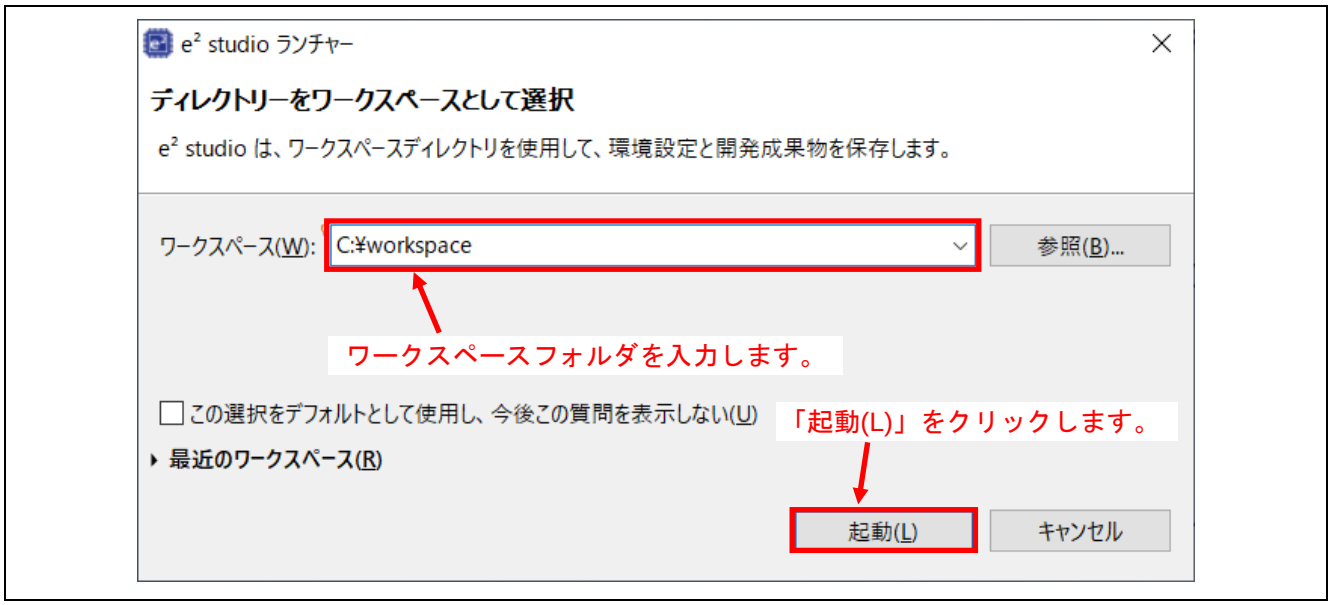
https://www.renesas.com/jp/ja/software-tool/renesas-flash-programmer-programming-gui.html

3.　プロジェクトの構築

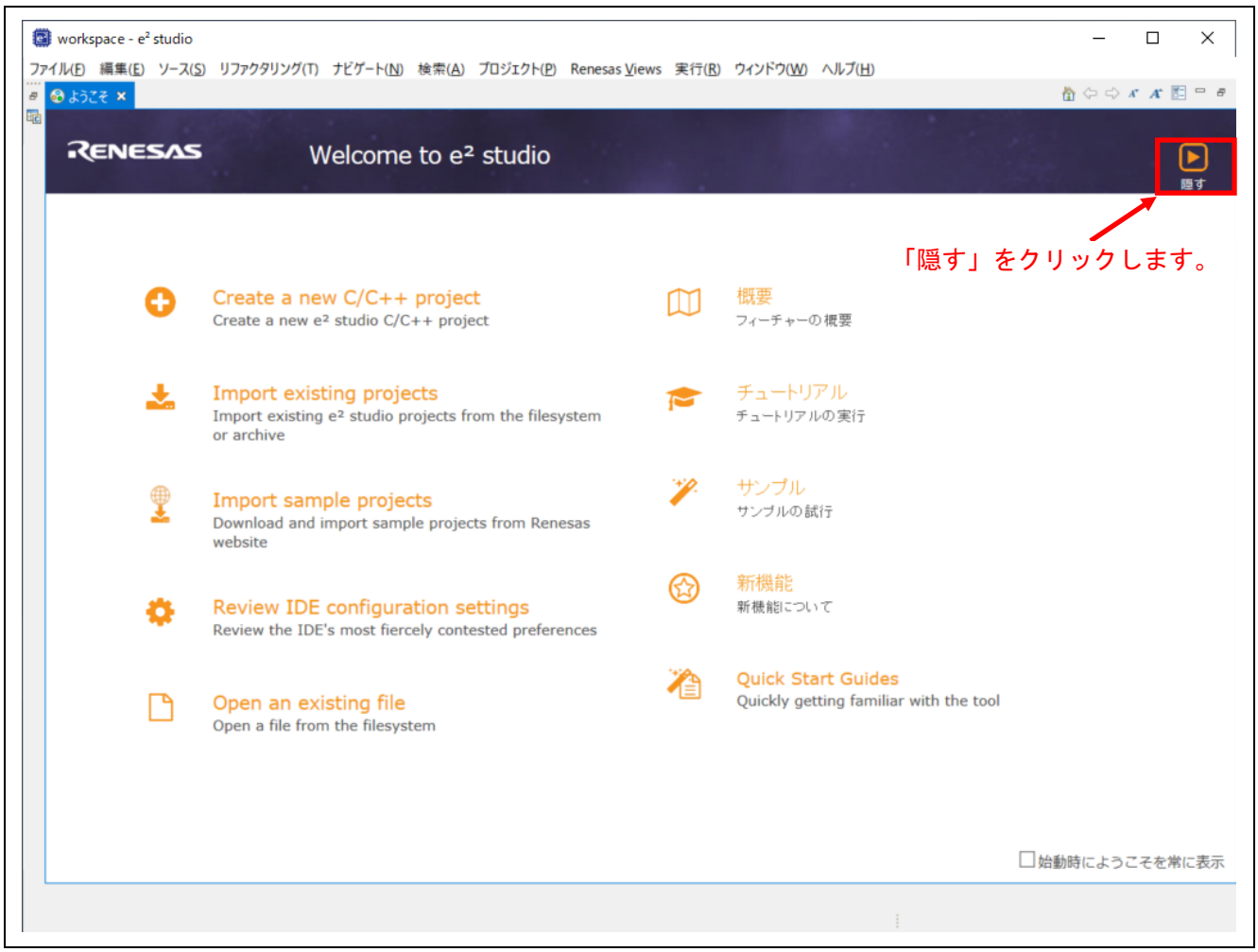
本アプリケーションノートは、環境構築済みのプロジェクトを同梱しています。e² studio のスマート・ブラウザーを用いてプロジェクトをインポートする手順について、以下に説明します。なお、CS+にインポートする場合の手順は「6.1 CS+での手順」を参照してください。

3.1　ワークスペースの作成

1. e² studio を起動します。
2. 表示されたダイアログに、任意のワークスペースを入力して、「起動(L)」をクリックします。



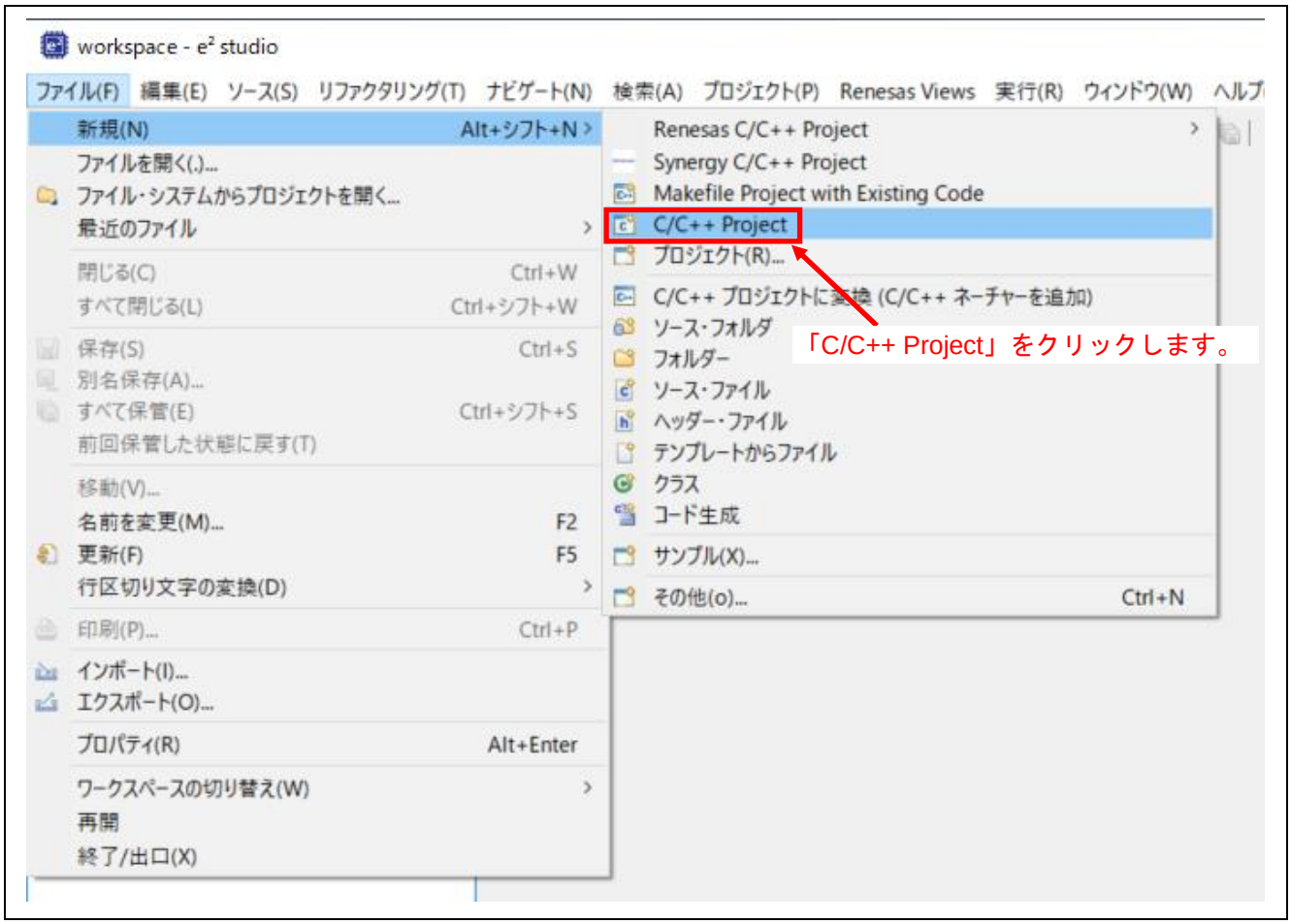
3.　以下の画面が表示されたら、「隠す」をクリックします。



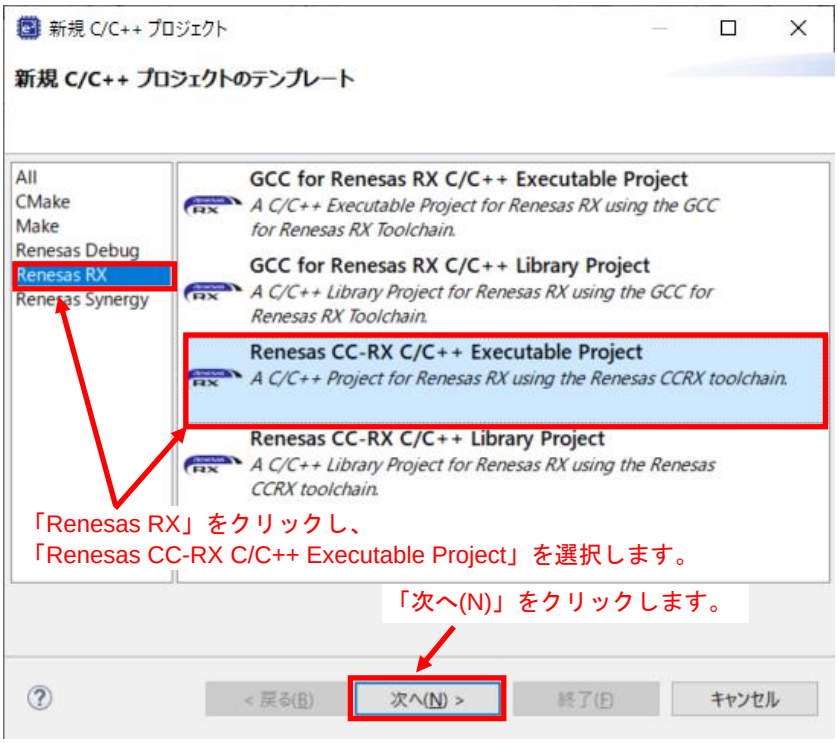
3.2　プロジェクトの作成

スマート・ブラウザーの機能を使用するときは、対象となるプロジェクトあるいはファイルを選択しておく必要があります。スマート・ブラウザーを使用するために、対象デバイスに指定したプロジェクトを作成します。ここで作成するプロジェクトは、スマート・ブラウザーを使用するためのダミープロジェクトです。インポートするプロジェクトの設定は「3.4 変更情報」を参照してください。

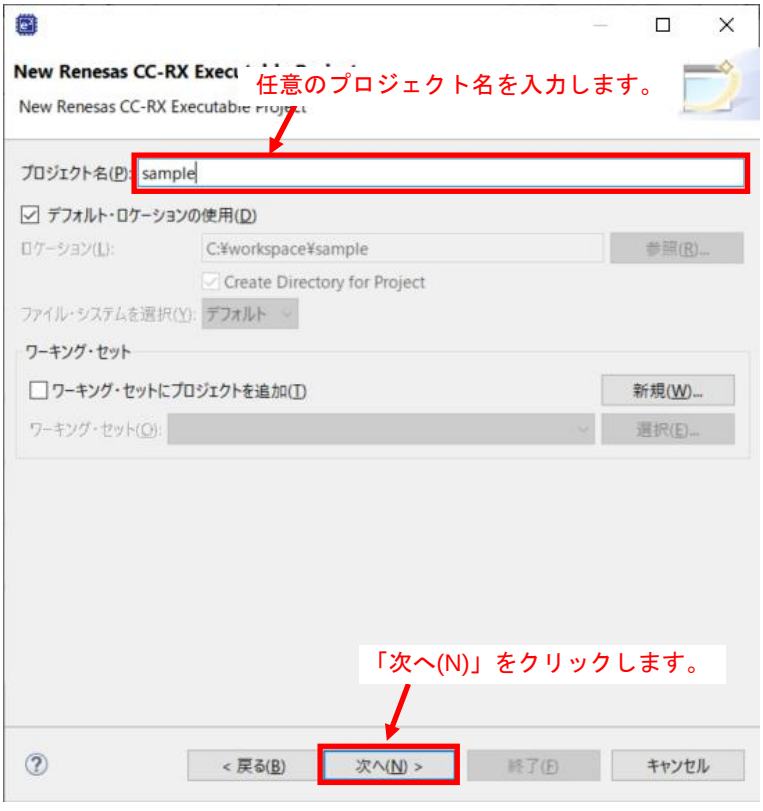
1. [ファイル(F)]→[新規(N)]→[C/C++ Project]の順にクリックして新しい C プロジェクトを作成します。新規プロジェクト作成ウィザードを起動します。



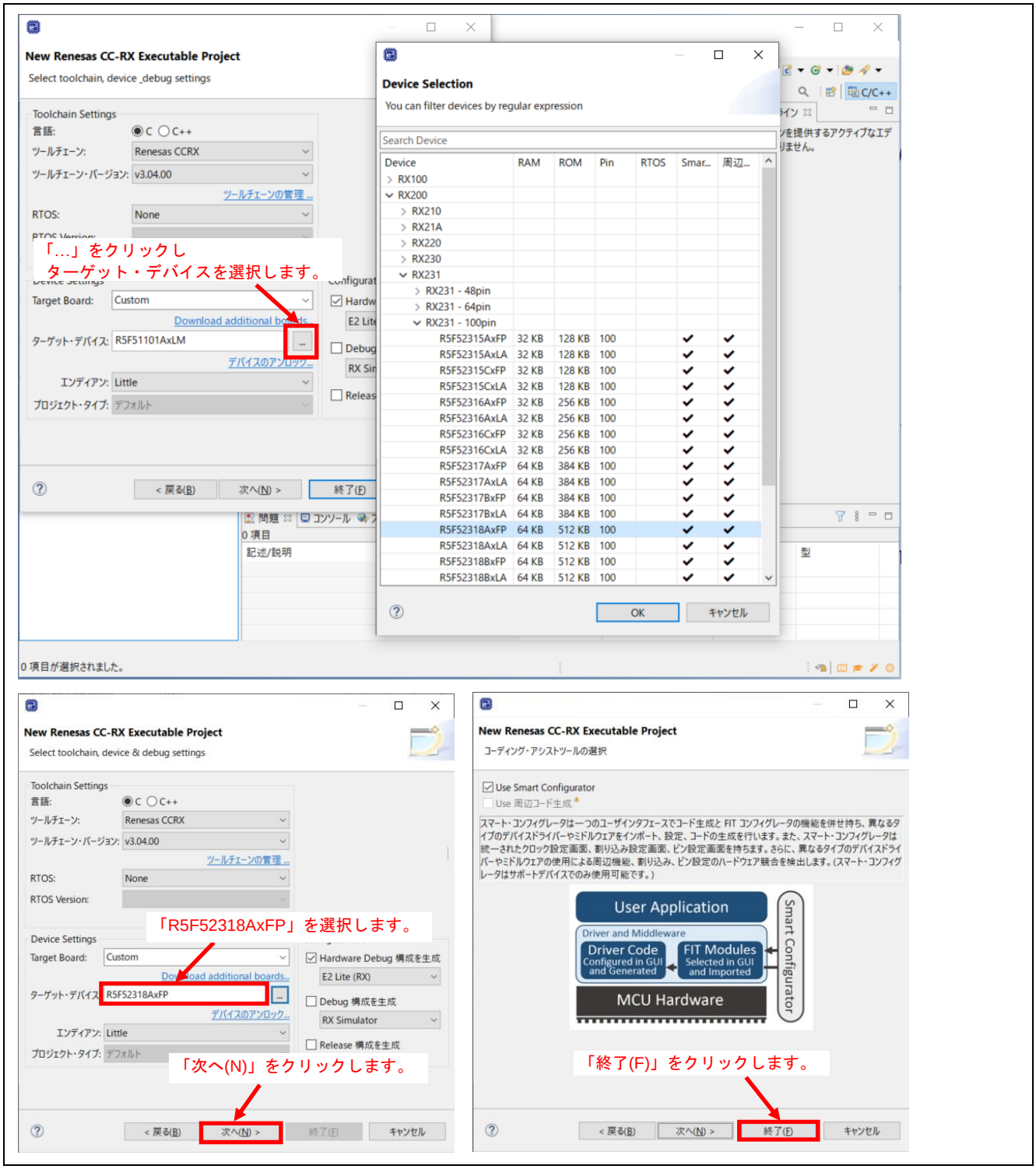
2.　” Renesas CC-RX C/C++ Executable Project”を選択します。[次へ(N)>]をクリックしてください。



任意のプロジェクト名を入力します。[次へ(N)>]をクリックしてください。



3. 「ターゲット・デバイス：」を設定します。RX130 グループの場合は「R5F51305AxFN」、RX140 グループの場合は「R5F51406BxFN」、RX231 グループの場合は「R5F52318AxFP」としてください。その他の項目は任意の設定で構いません。設定が完了してから[終了(F)]をクリックします。※下図は RX231 グループを使用する場合の例です。

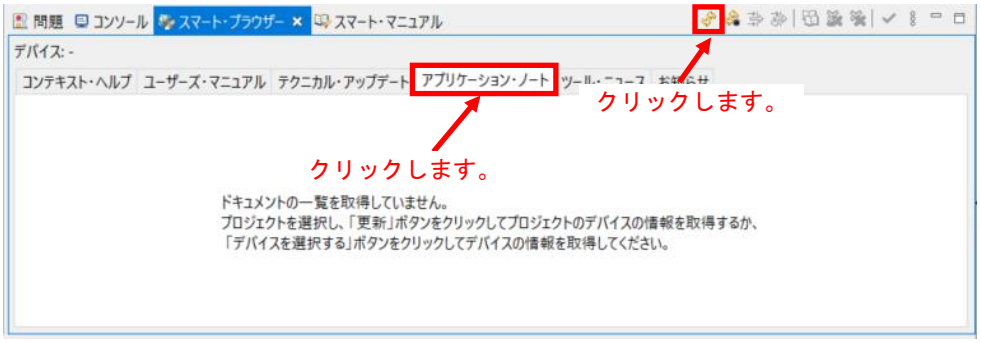


3.3　プロジェクトのインポート

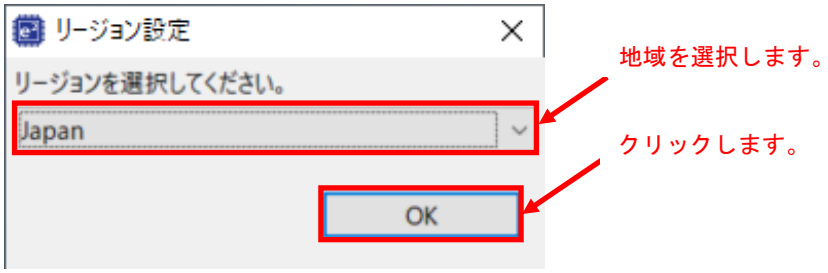
サンプルプログラムのプロジェクトを、作成したワークスペースにインポートします。

1. プロジェクト・エクスプローラーから「3.2 プロジェクトの作成」で作成したプロジェクトを選択します。
2. スマート・ブラウザーを起動します。

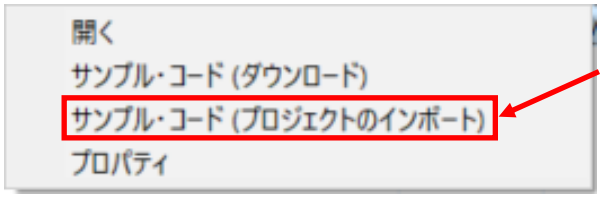
3. [スマート・ブラウザー]タブの[アプリケーションノート]タブをクリックします。
4. [更新]をクリックします。



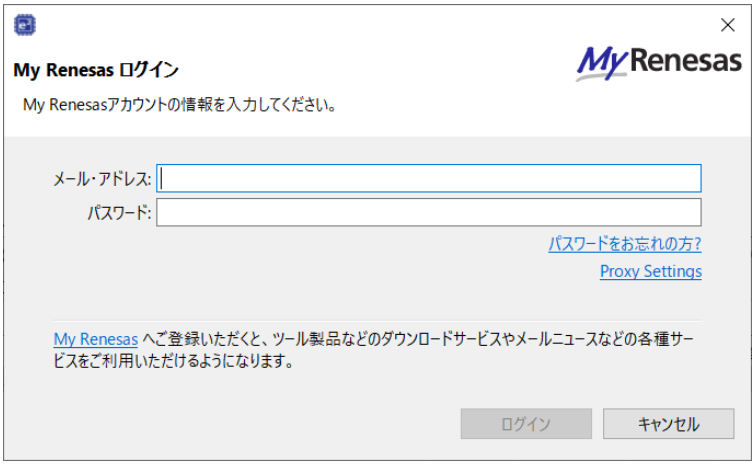
5. 「リージョン設定」ダイアログが表示されます。作業している地域を選択し[OK]をクリックします。



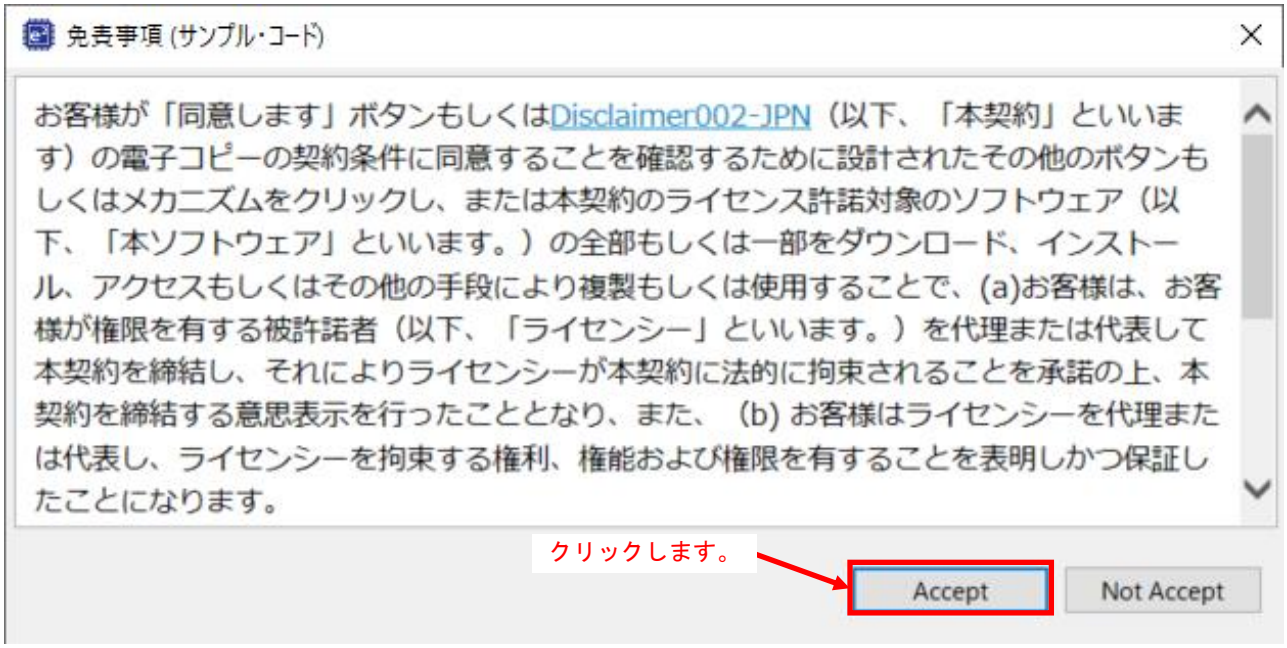
6. 本アプリケーションノートを選択し、右クリックします。コンテキストメニューの[サンプル・コード(プロジェクトのインポート)]をクリックします。*1



【注】*1　My Renesas による認証を一度も行っていない場合、ファイルをダウンロードする前に「My Renesas」ダイアログが表示されます。ルネサス Web サイトで登録しているメールアドレスとパスワードを入力してください。



7. [Accept]をクリックします。



8. 本アプリケーションノートを保存します。

9. RX130 グループの場合：「プロジェクト(P):」に表示されている「update_firmware_rx130」および「firmware_rx130」の両方を選択し、「終了(F)」をクリックします。

   RX140 グループの場合：「プロジェクト(P):」に表示されている「update_firmware_rx140」および「firmware_rx140」の両方を選択し、「終了(F)」をクリックします。

   RX231 グループの場合：「プロジェクト(P):」に表示されている「update_firmware_rx231」および「firmware_rx231」の両方を選択し、「終了(F)」をクリックします。

10. プロジェクトインポート後は、スマート・ブラウザーを使用するために作成したプロジェクト（ここでは sample）は不要ですので削除してください。

3.4　変更情報

本アプリケーションノートでは、サンプルプログラムを構成するために各 FIT モジュールのコンフィギュレーションオプションとプロジェクト設定を変更しています。以下に詳細を示します。

3.4.1　コンフィギュレーションオプション

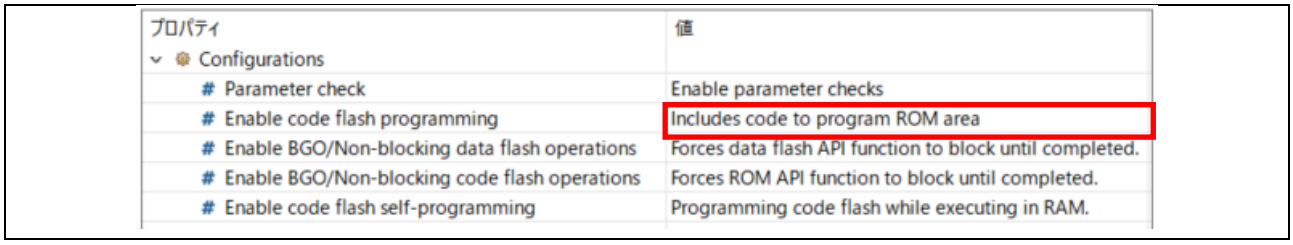
サンプルプログラムを構成する各 FIT モジュールのコンフィギュレーションオプションを変更しています。

コンフィギュレーションオプションの項目と設定内容は、各 FIT モジュールの doc フォルダに入っているマニュアルなどを参照してください。

以下にスマートコンフィグレータのコンフィギュレーションオプションの変更箇所を示します。

(1)　フラッシュ FIT モジュールの設定変更

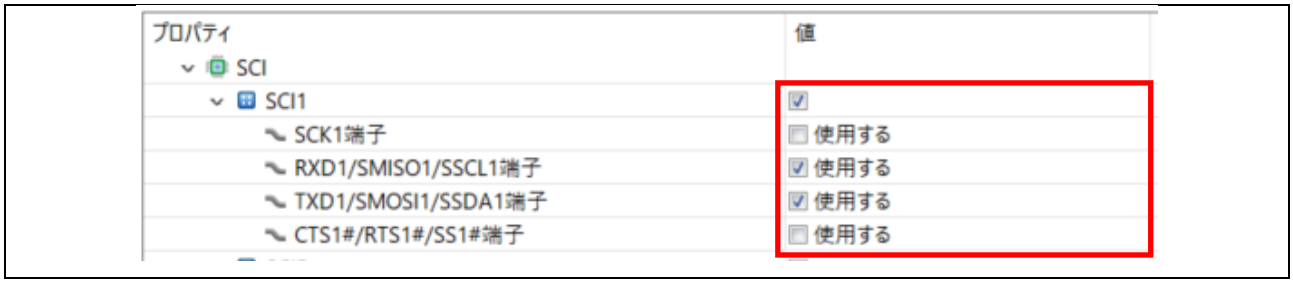
コンポーネント設定画面において、フラッシュ FIT モジュールでコードフラッシュメモリの書き換えができるように設定を変更しています。



(2)　SCI FIT モジュールの設定変更

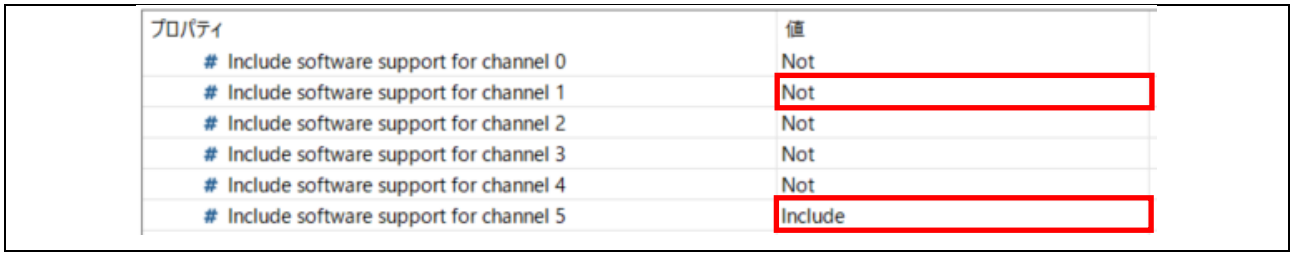
コンポーネント設定画面において、送信データエンプティ割り込みを有効に変更しています。

コンポーネント設定画面において、RXD1 と TXD1 の端子を有効にしています。



(3)　SCI FIT モジュールの設定変更（RX231 グループのみ）

コンポーネント設定画面において、SCI FIT モジュールで使用するチャネルを CH1 から CH5 に変更しています。



コンポーネント設定画面において、RXD5 と TXD5 の端子を有効にしています。

端子設定画面において、端子割り当てを RXD5 は PA3 に変更し、TXD5 は PA4 に変更しています。

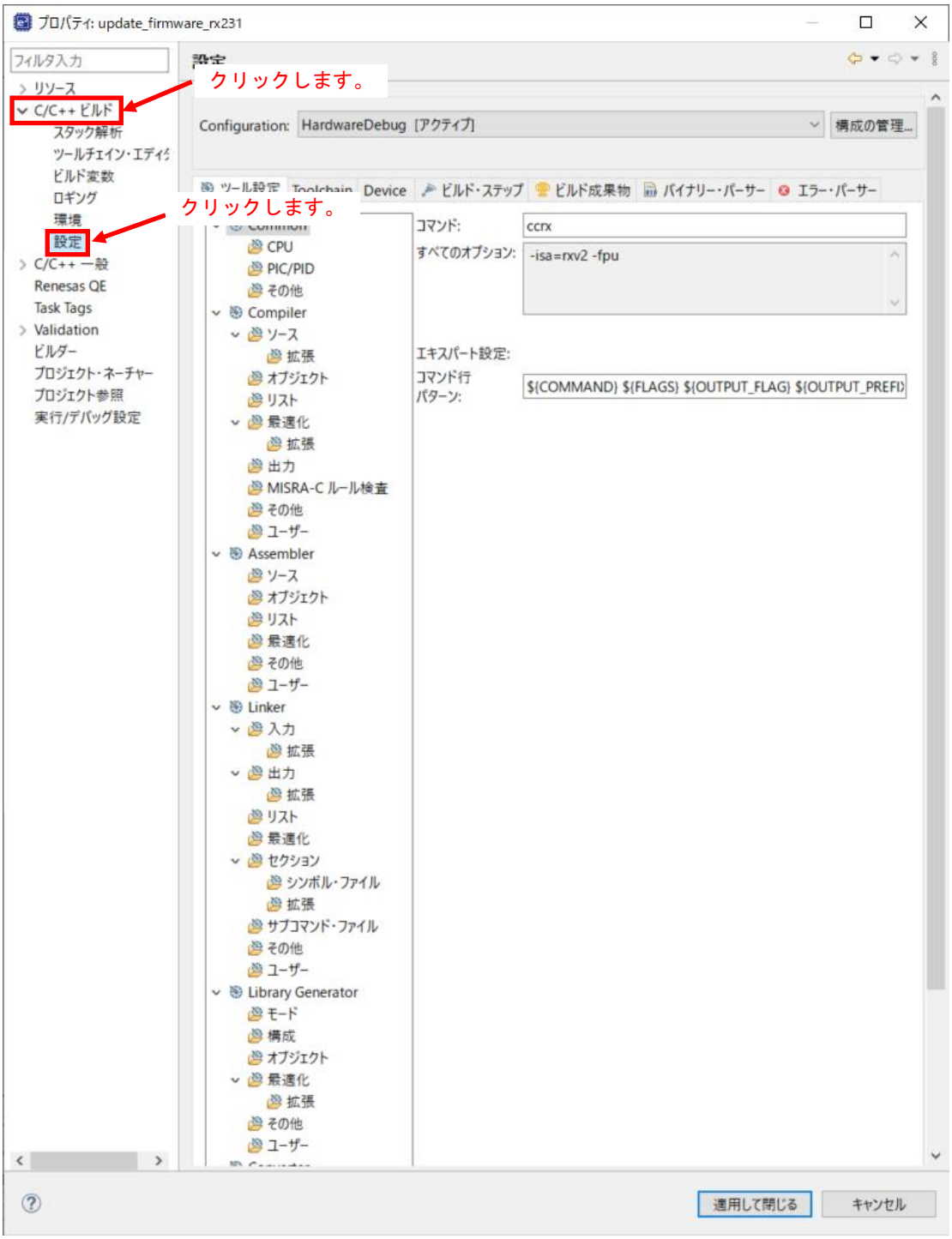
3.4.2　　プロジェクトの設定変更

ファームウェアアップデートプロジェクトのビルド時の設定はデフォルト設定から、表 3.1 および表 3.2 に示す内容に変更しています。ファームウェアプロジェクトのビルド時の設定はデフォルト設定から、表 3.3 に示す内容に変更しています。

プロジェクト設定を確認する場合、以下の手順で行ってください。

1. e² studio の対象プロジェクト、RX130 グループの場合（update_firmware_rx130 または firmware_rx130）を、RX140 の場合（update_firmware_rx140 または firmware_rx140）を、RX231 グループの場合（update_firmware_rx231 または firmware_rx231）を選択して右クリックします。その後「プロパティ(R)」をクリックします。※下図は RX231 グループを使用する場合の例です。

2. 「C/C++ビルド」をクリックして、「設定」をクリックします。※下図は RX231 グループを使用する場合の例です。



3. 「ツール設定」タブから、ファームウェアアップデートプロジェクトは表 3.1 および表 3.2、ファームウェアプロジェクトは表 3.3 に示す内容に変更しています。

表 3.1　変更したプロジェクトのビルド設定（ファームウェアアップデートプロジェクト）（1/2）

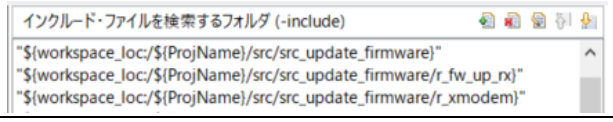
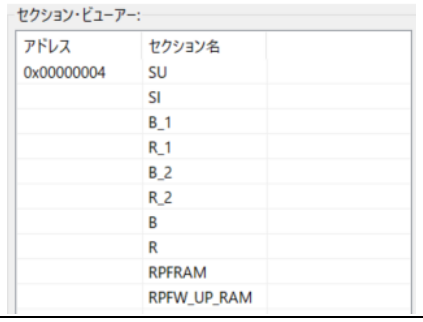
項目	変更内容	説明
Compiler - ソース	「インクルード・ファイル・ディレクトリ」にインクルードパスを追加する。	各 FIT モジュールで設定が必要なインクルードパスを追加します。 「スマートコンフィグレータ」を使用して、各 FIT モジュールを組み込む場合、自動で設定されます。
		サンプルプログラムのインクルードパスを追加します。本プロジェクトでは、 src/src_update_firmware, src/src_update_firmware/r_fw_up_rx, src/src_update_firmware/r_xmodem を追加しています。 <設定例> 
Linker - セクション	RAM 領域に、RPFRAM セクション、RPFW_UP_RAM セクションを追加する。	サンプルプログラムが使用する RAM 領域を設定します。 <設定例> 

表 3.2　変更したプロジェクトのビルド設定（ファームウェアアップデートプロジェクト）（2/2）

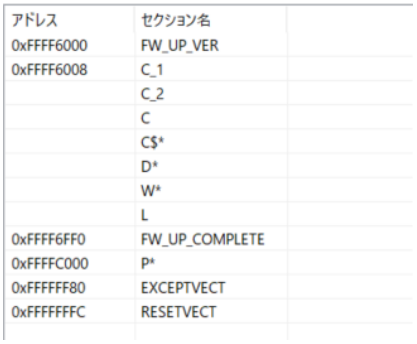
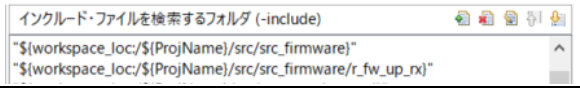
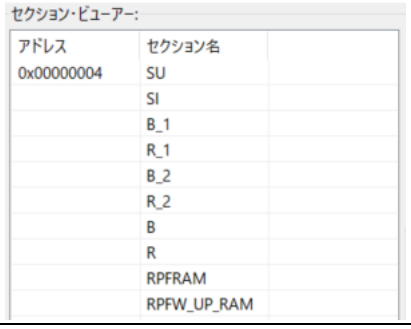
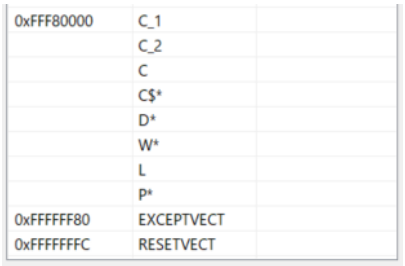
項目	変更内容	説明
Linker - セクション	ROM 領域に、FW_UP_VER セクション、FW_UP_COMPLETE セクションを追加する。 FW_UP_VER セクションの先頭アドレスを、 RX130 グループの場合 0xFFFF6800 RX140 グループおよび RX231 グループの場合 0xFFFF6000 に設定する。 C_1 セクションの先頭アドレスを、 RX130 グループの場合 0xFFFF6808 RX140 グループおよび RX231 グループの場合 0xFFFF6008 に設定する。 FW_UP_COMPLETE セクションの先頭アドレスを、 RX130 グループの場合 0xFFFF73F0 RX140 グループおよび RX231 グループの場合 0xFFFF6FF0 に設定する。 P*セクションの先頭アドレスを、 0xFFFFC000 に設定する。	サンプルプログラムを配置する ROM 領域を設定します。定数データをスタートアッププログラム保護機能の代替領域より上に配置して、P*セクションをスタートアッププログラム保護機能のデフォルト領域に配置します。 <設定例> 
Linker - セクション - シンボル・ファイル	ROM から RAM へマップするセクションに PFRAM=RPFRAM, PFW_UP_RAM=RPFW_UP_RAM を追加する。	サンプルプログラムは、コードフラッシュメモリの書き換えを RAM 上で実行します。そのため ROM から RAM にマップする設定を追加します。

表 3.3　変更したプロジェクトのビルド設定（ファームウェアプロジェクト）

項目	変更内容	説明
Compiler - ソース	「インクルード・ファイル・ディレクトリ」にインクルードパスを追加する。	各 FIT モジュールで設定が必要なインクルードパスを追加します。 「スマートコンフィグレータ」を使用して、各 FIT モジュールを組み込む場合、自動で設定されます。
		サンプルプログラムのインクルードパスを追加します。本プロジェクトでは、src/src_firmware, src/src_firmware/r_fw_up_rx を追加しています。 <設定例> 
Linker - セクション	RAM 領域に、RPFRAM セクション、RPFW_UP_RAM セクションを追加する。	サンプルプログラムが使用する RAM 領域を設定します。 <設定例> 
Linker - セクション	ROM 領域に配置するセクションの先頭アドレスを、 RX130 グループの場合 0xFFFE0000 RX140 グループの場合 0xFFFC0000 RX231 グループの場合 0xFFF80000 に設定する。	サンプルプログラムを配置する ROM 領域を設定します。サンプルプログラムをコードフラッシュメモリの先頭アドレスに配置します。 <設定例>  【注】　ファームウェアプログラムは以下の領域を使用することはできないため、この領域にデータが配置されないようにしてください。 RX130 グループの場合 FFFF_6800h 番地～FFFF_BFFFh 番地 RX140 グループおよび RX231 グループの場合 FFFF_6000h 番地～FFFF_BFFFh 番地
Linker - セクション - シンボル・ファイル	ROM から RAM へマップするセクションに PFRAM=RPFRAM, PFW_UP_RAM=RPFW_UP_RAM を追加する。	サンプルプログラムは、スタートアッププログラム保護機能の領域入れ替えを RAM 上で実行します。そのため ROM から RAM にマップする設定を追加します。

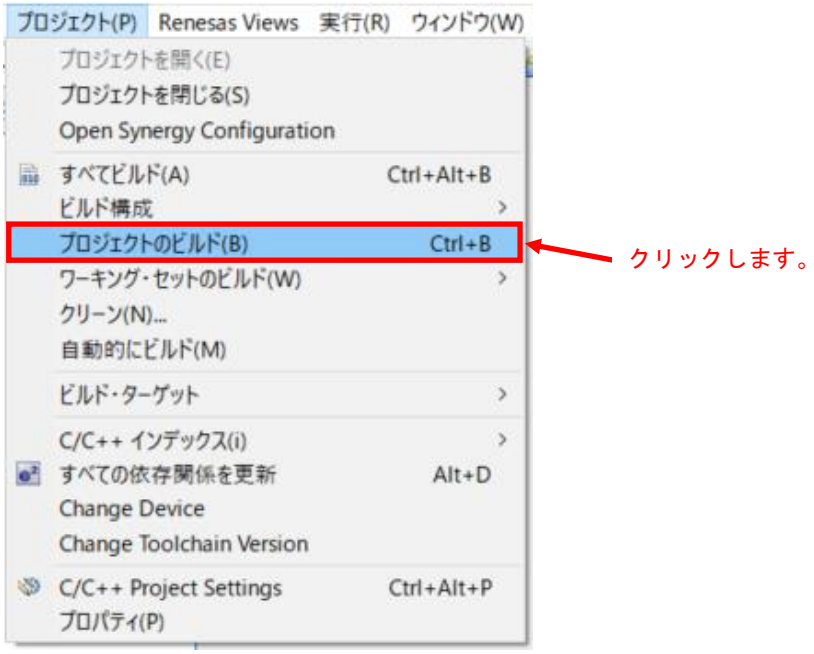
4.　動作確認

4.1　　プロジェクトのビルド

以下の手順に従い、プロジェクトをビルドしてロードモジュールを作成します。

1. ビルドするプロジェクト、RX130 グループの場合（update_firmware_rx130 または firmware_rx130）を、RX140 グループの場合（update_firmware_rx140 または firmware_rx140）を、RX231 グループの場合（update_firmware_rx231 または firmware_rx231）をクリックします。※下図は RX231 グループを使用する場合の例です。

2. 「プロジェクト(P)」メニューの「プロジェクトのビルド(B)」をクリックします。



3. 「コンソール」パネルに「”Build Complete.”」と表示されたらビルド完了です。

```
Renesas Optimizing Linker Completed
'Finished building target:'

'Build complete.'

13:11:24 ビルドが完了しました(所要時間 52s.846ms)
```

4.2　デバッグの準備

4.2.1　機器の準備

デバッグを開始する前に、評価ボードを準備します。表 4.1 に必要な機器の一覧を、図 4.1 にデバッグ構成図に示します。

表 4.1　機器一覧

No.	機器	補足
1	開発 PC	開発する PC です。
2	評価ボード。 (Renesas Starter Kit for RX130／RX140／RX231)	-
3	ホスト PC — 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。
4	USB ケーブル（mini-B タイプ）	Renesas Starter Kit for RX130／RX140／RX231 のシリアル入出力信号は USB シリアル変換されており、ホスト PC と USB 接続することにより仮想 COM ポートとして使用することができます。

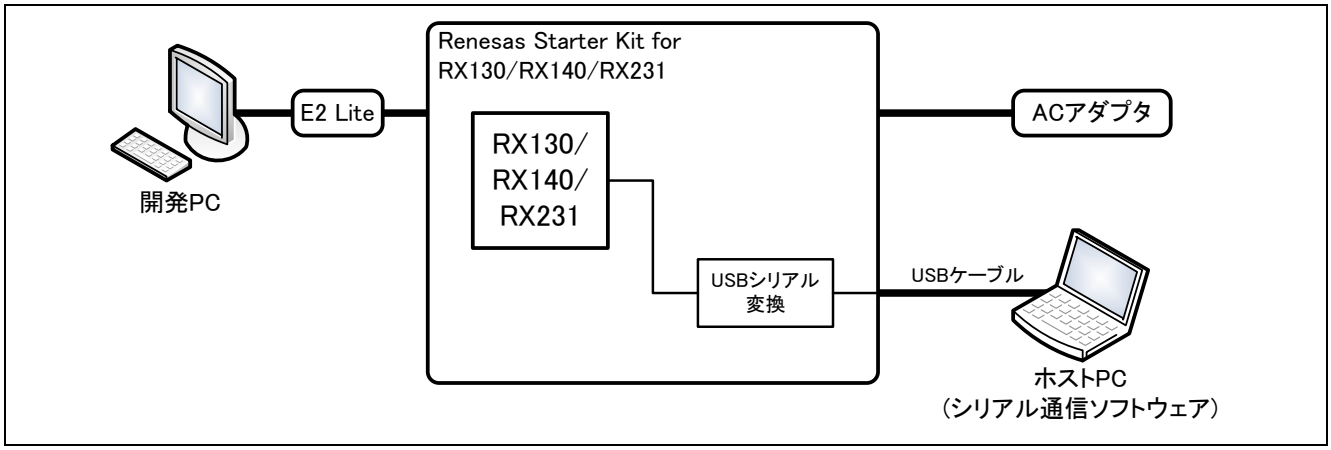


図 4.1　デバッグ構成図

4.2.2　ホスト PC の設定

デバイスとホスト PC のシリアル通信仕様を表 4.2 に示します。ターミナルソフトウェアの設定方法はターミナルソフトウェアのマニュアルを参照してください。

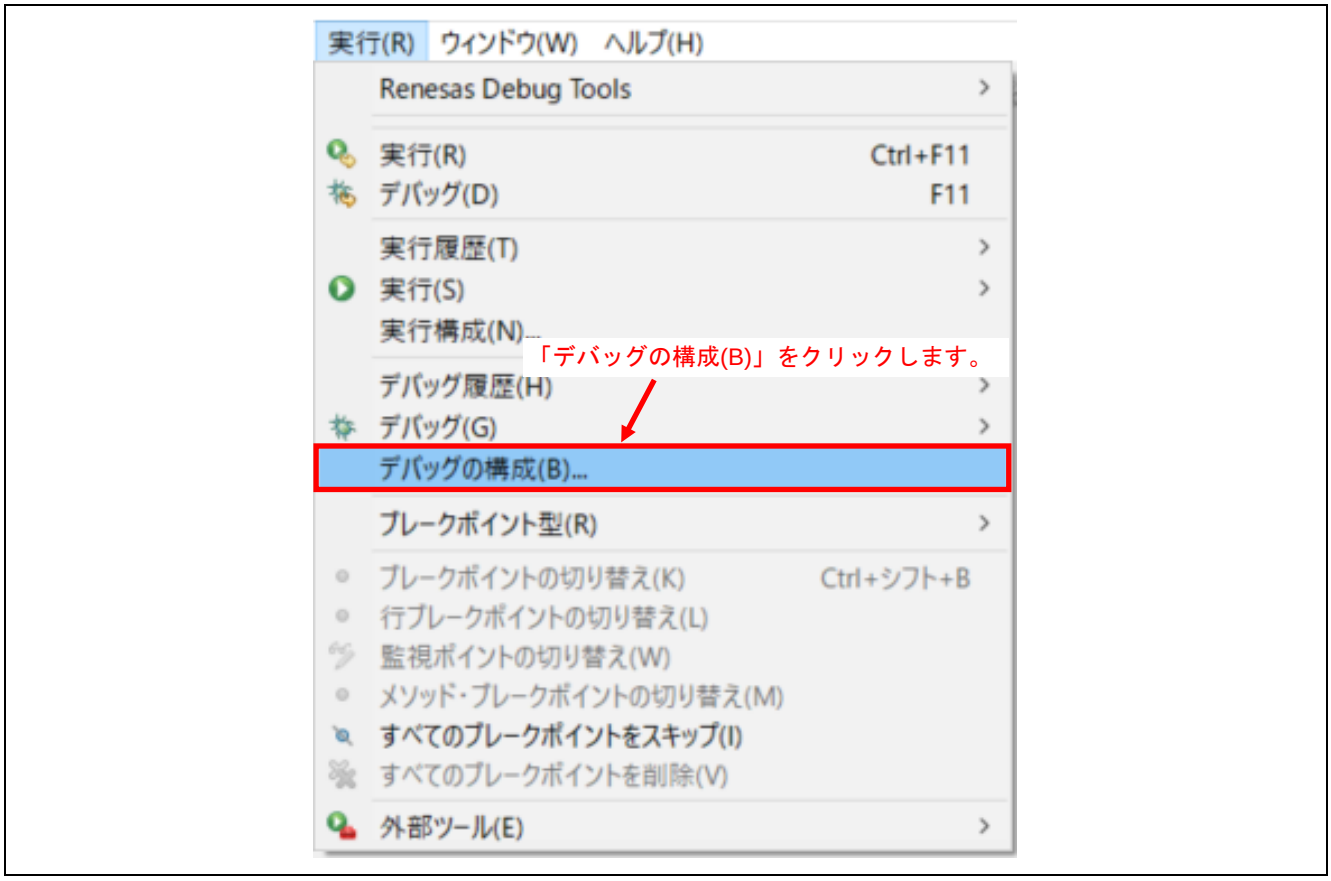
表 4.2　通信仕様

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

4.3　プロジェクトのデバッグ

以下の手順に従い、ファームウェアアップデートプロジェクトのデバッグを開始します。ファームウェアプロジェクトも同様の手順です。

1. e² studio の「実行」メニューの「デバッグの構成」をクリックします。



2.　RX130 グループの場合：

「Renesas GDB Hardware Debugging」の「update_firmware_rx130 HardwareDebug」をクリックします。

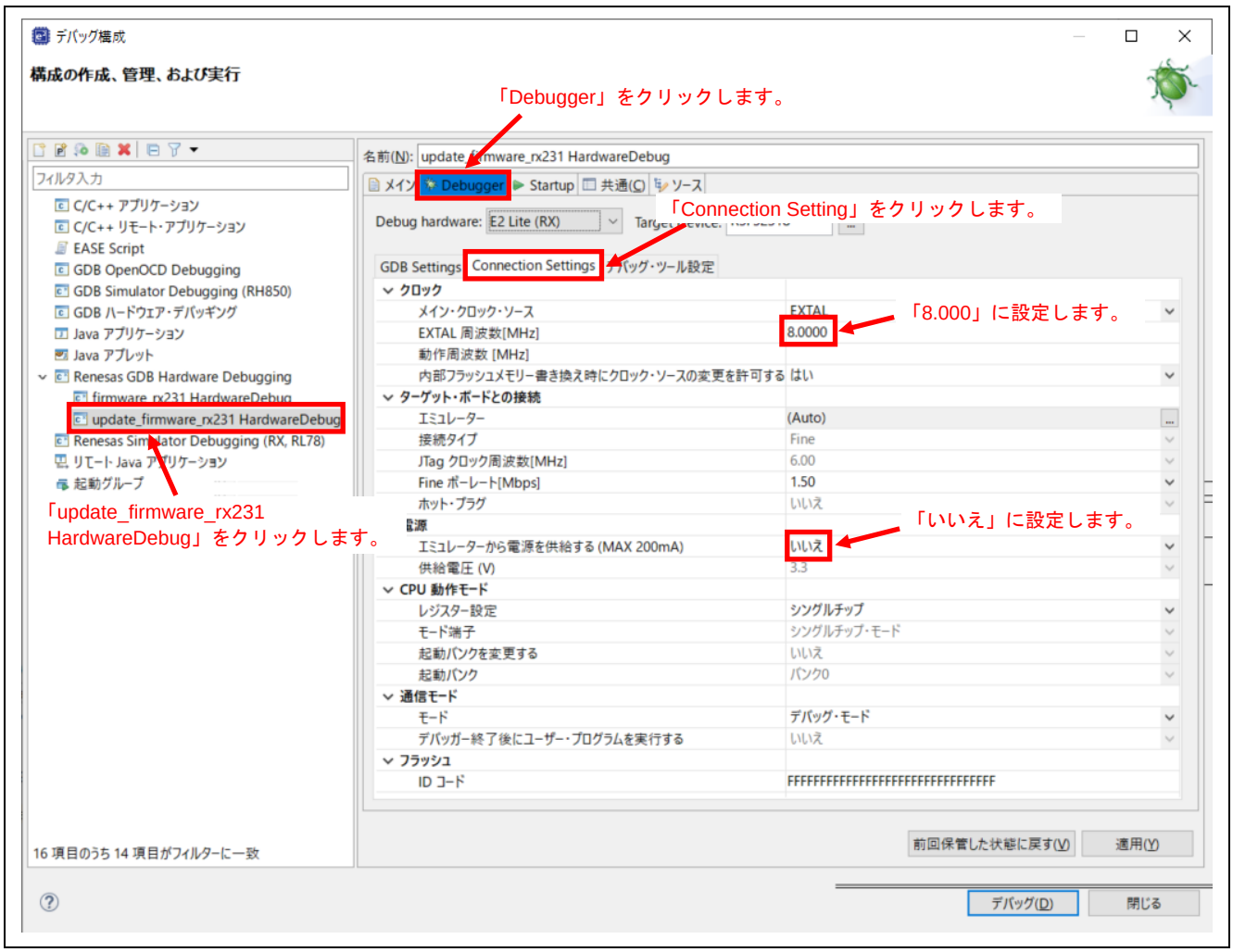
RX140 グループの場合：

「Renesas GDB Hardware Debugging」の「update_firmware_rx140 HardwareDebug」をクリックします。

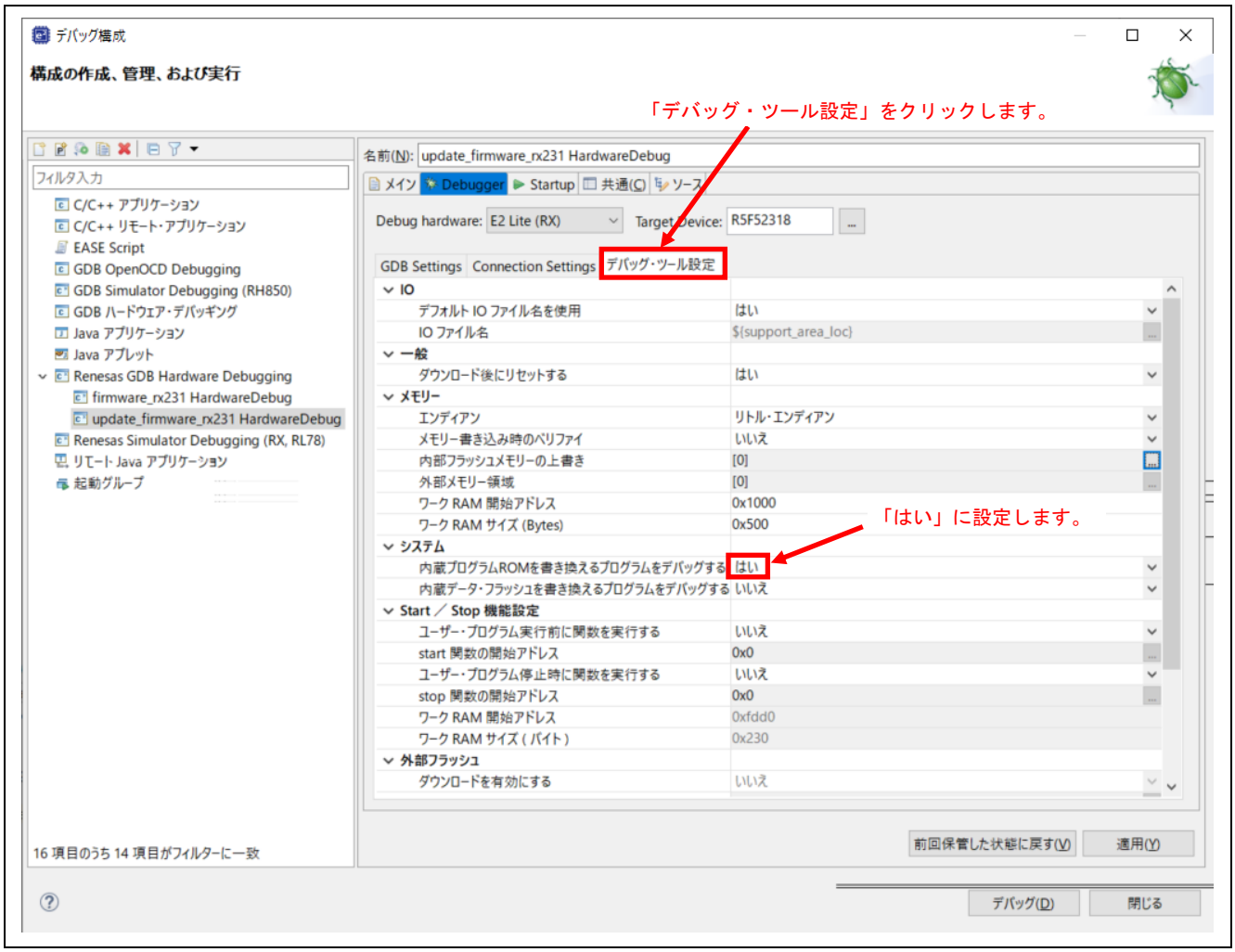
RX231 グループの場合：

「Renesas GDB Hardware Debugging」の「update_firmware_rx231 HardwareDebug」をクリックします。

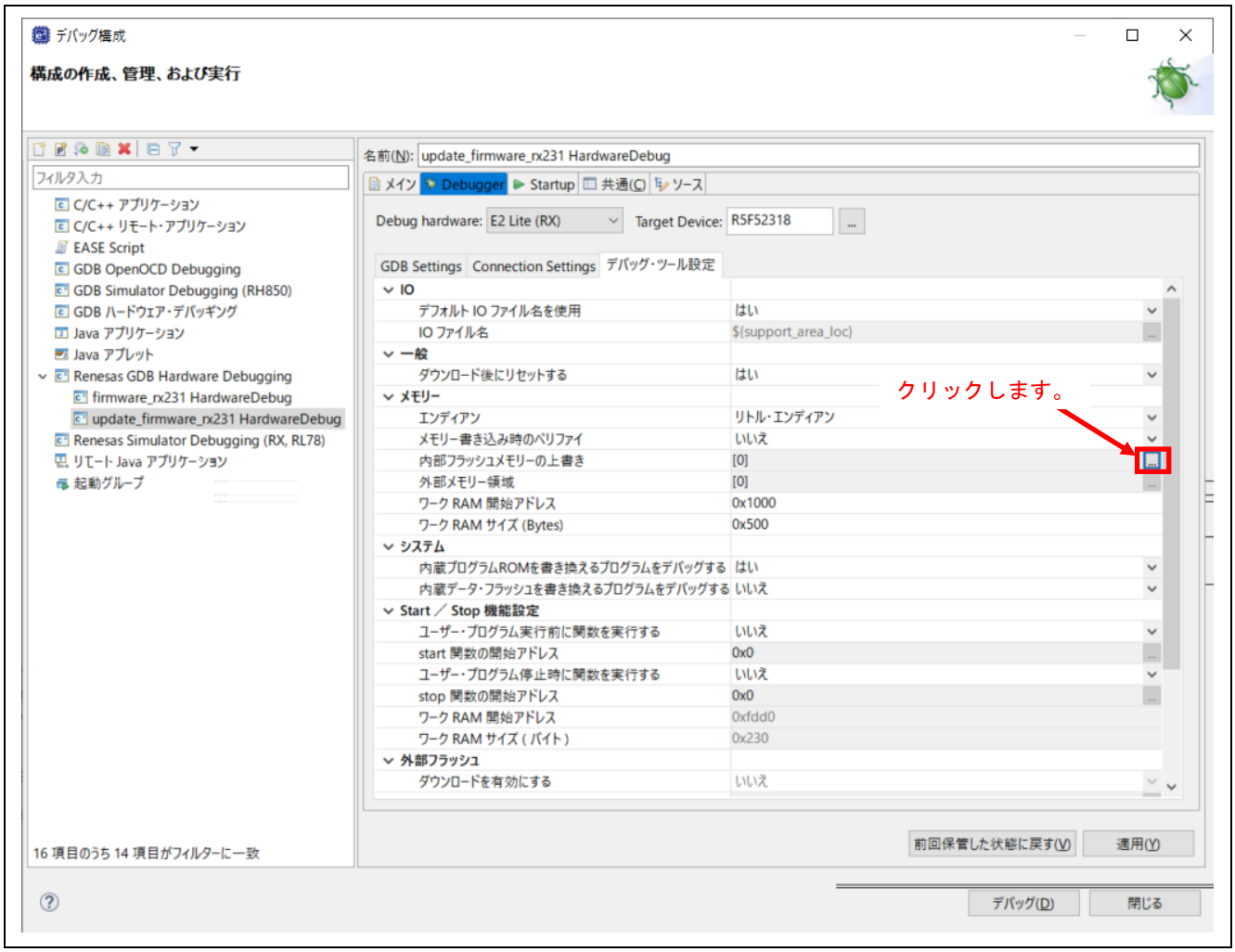
「Debugger」タブをクリックして、「Connection Settings」タブをクリックします。「EXTAL 周波数」を「8.0000」に修正して、「エミュレータから電源を供給する」を「いいえ」に変更します。※下図は RX231 グループを使用する場合の例です。



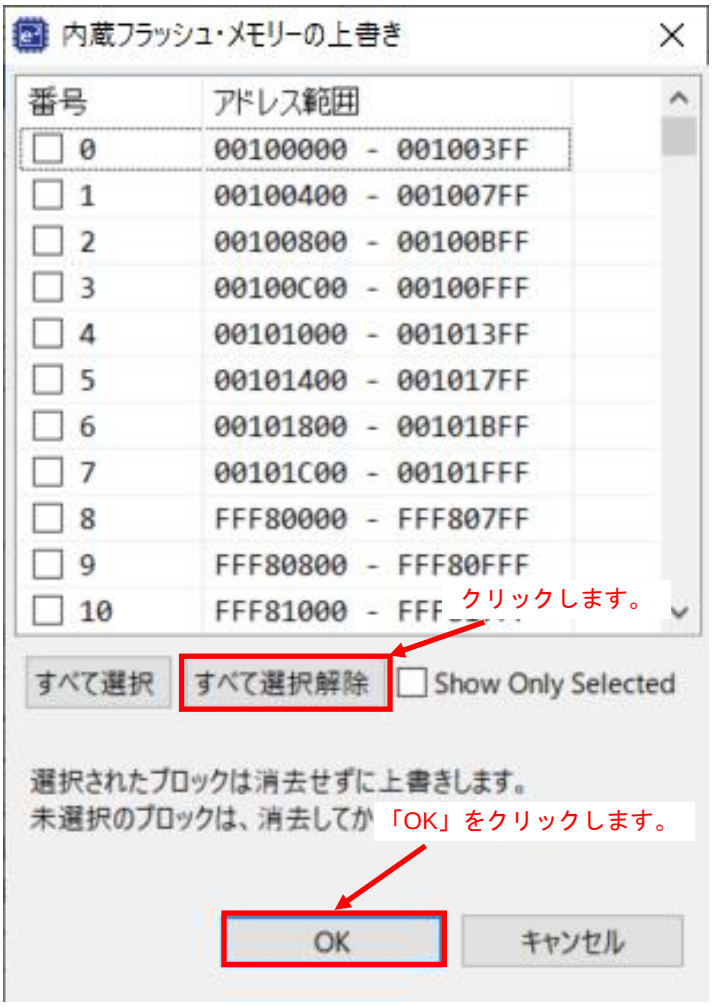
3. 「デバッグ・ツール設定」タブをクリックします。「内蔵プログラム ROM を書き換えるプログラムをデバッグする」を「はい」に変更します。※下図は RX231 グループを使用する場合の例です。



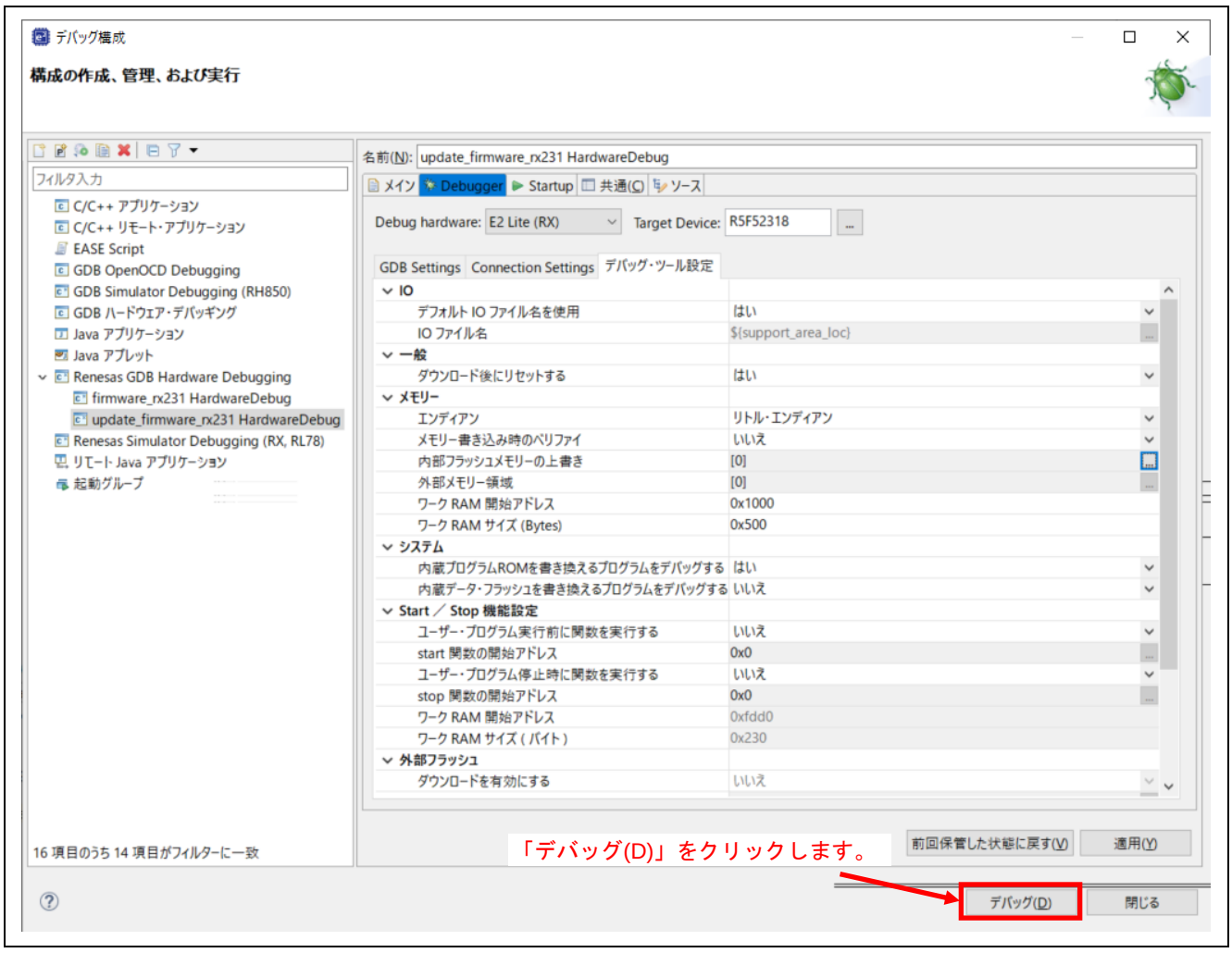
4. 「内蔵フラッシュメモリーの上書き」にある右端のボタンをクリックします。※下図は RX231 グループを使用する場合の例です。



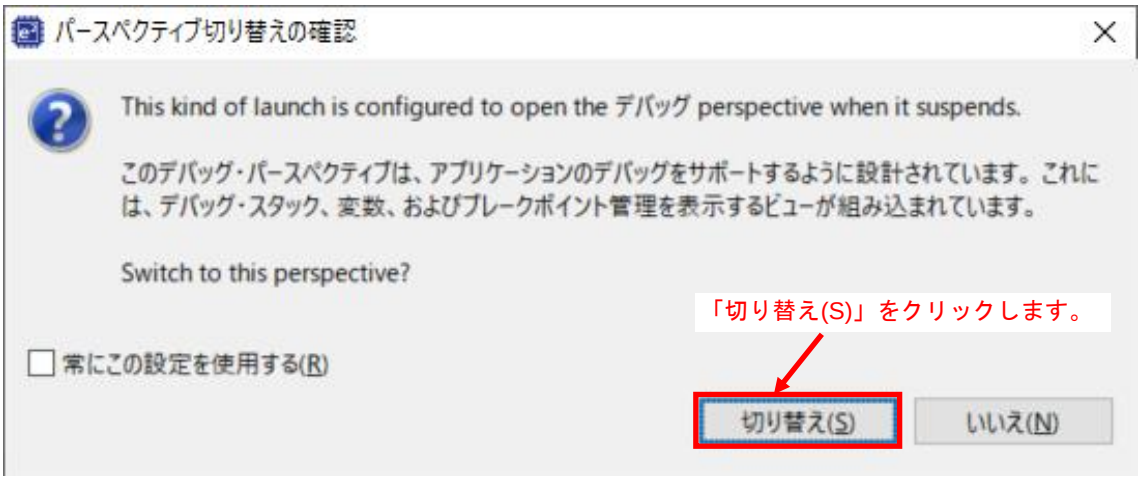
5. コードフラッシュメモリの全ブロックを消去してから上書きするように設定します。「すべて選択解除」をクリックしてから、「OK」をクリックします。



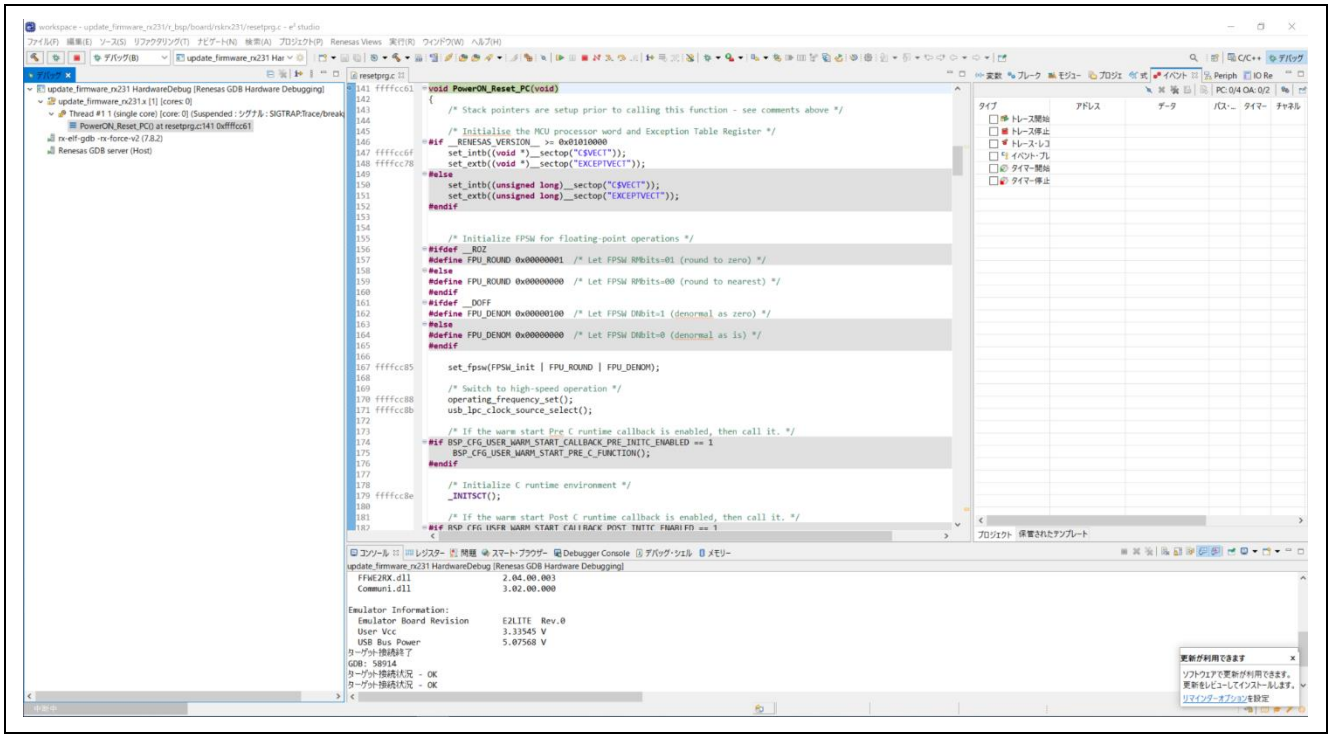
6.　「デバッグ(D)」をクリックします。※下図は RX231 グループを使用する場合の例です。



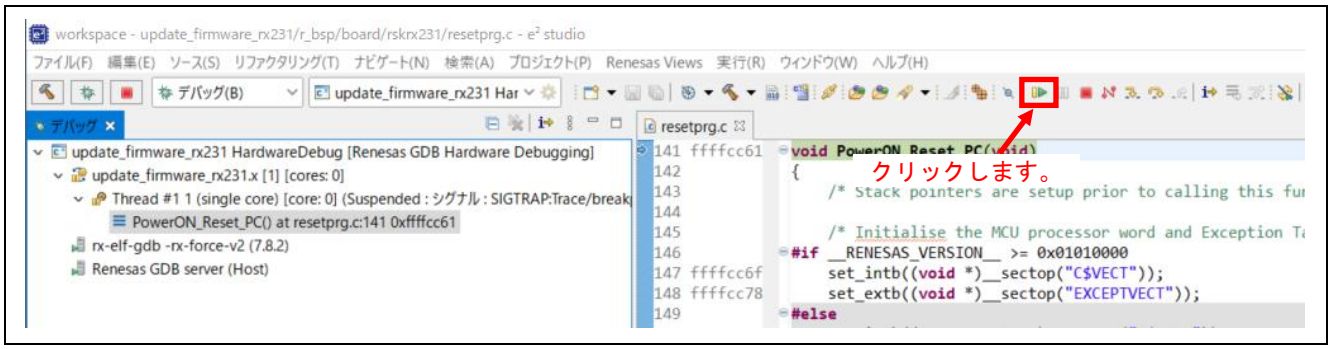
以下のメッセージが表示されたら、「切り替え(S)」をクリックします。



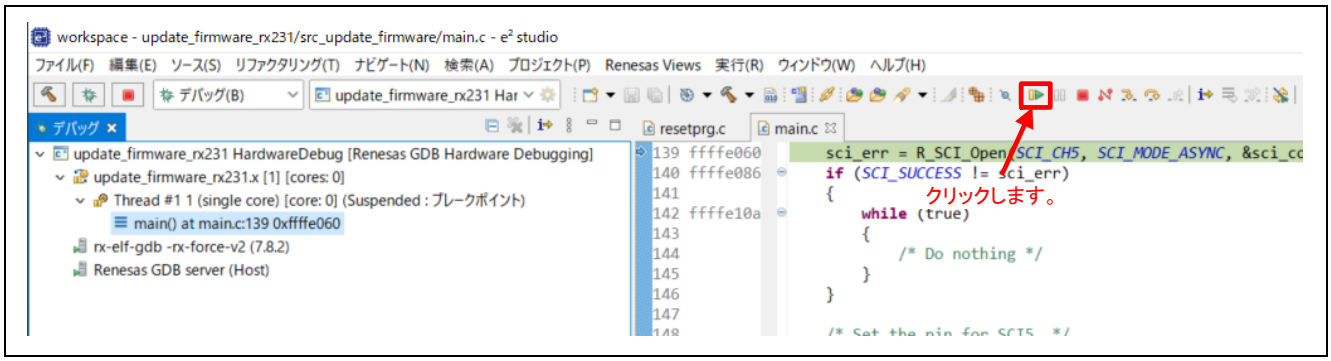
ロードモジュールのダウンロードが完了すると、「デバッグ」パースペクティブが開きます。※下図は RX231 グループを使用する場合の例です。



7. ツールバーの「再開」をクリックします。プログラムが実行され、main 関数の先頭でブレークします。※下図は RX231 グループを使用する場合の例です。



8. main 関数の先頭でブレークした後に、もう一度ツールバーの「再開」をクリックします。※下図は RX231 グループを使用する場合の例です。



9. ターミナルソフトウェアに以下のメッセージが出力されることを確認します。
RX130 グループの場合：

```
RX130 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

RX140 グループの場合：

```
RX140 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

RX231 グループの場合：

```
RX231 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

5.　アプリケーション概要

5.1　　ファームウェアアップデートプログラムの構成

本アプリケーションノートのサンプルプログラムであるファームウェアアップデートプログラムの構成について説明します。プログラムはスタートアッププログラム保護機能のデフォルト領域に格納しています。変数の初期値や文字列リテラルなどの定数データは ROM 内の定数データ領域に格納しています。定数データ領域は定数データ領域 1 と定数データ領域 2 があり、定数データはどちらか一方に格納されます。

定数データ領域は、バージョン情報格納領域、定数データ格納領域、書き込み完了情報格納領域の 3 つの領域で構成されています。

図 5.1 にファームウェアアップデートプログラムのメモリマップを、表 5.1 に定数データ領域の構成を示します。

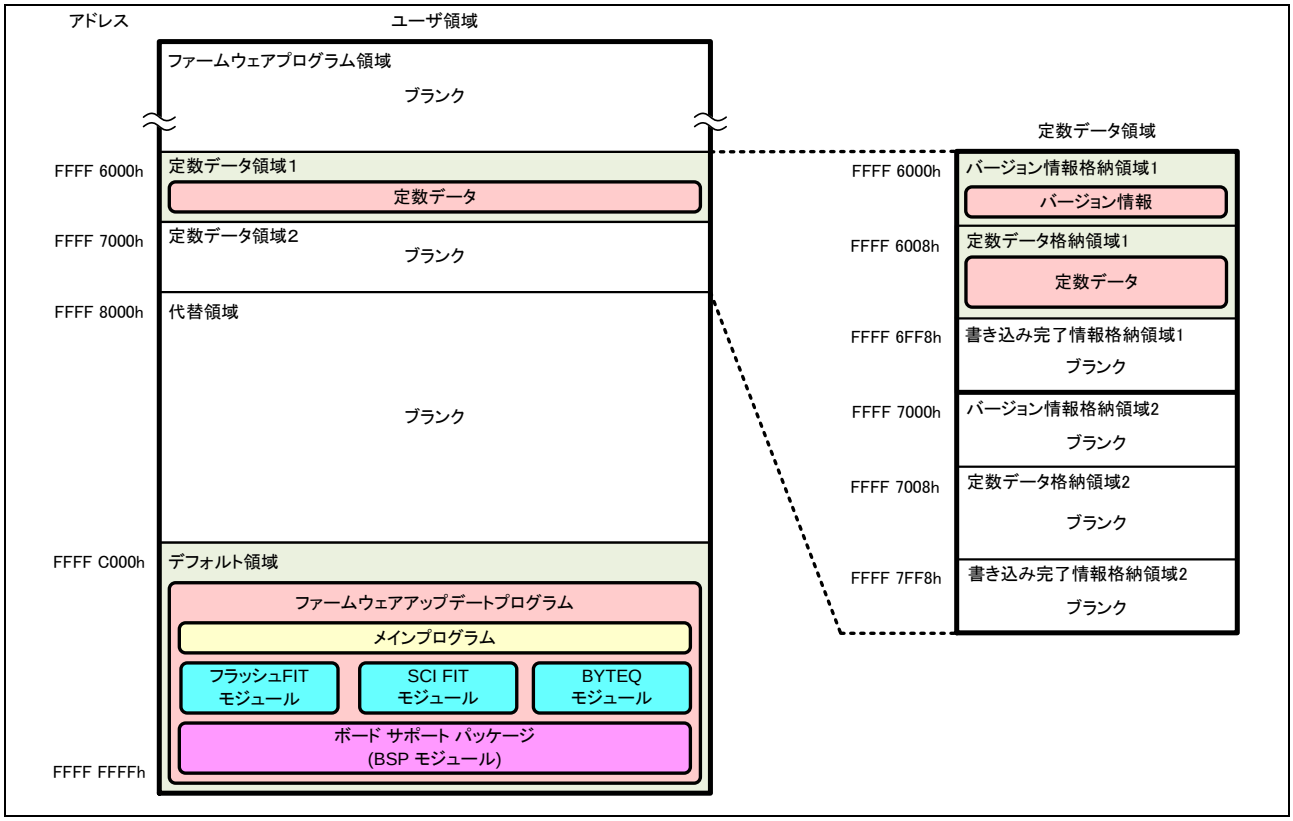


図 5.1　ファームウェアアップデートプログラムのメモリマップ

表 5.1　定数データ領域の構成

領域名	概要
バージョン情報格納領域	ファームウェアアップデートプログラムバージョン情報を格納する領域
定数データ格納領域	定数データを格納する領域
書き込み完了情報格納領域	ファームウェアアップデートプログラムの更新時にバージョン情報を書き込む領域

5.2　動作概要

本アプリケーションノートのサンプルプログラムであるファームウェアアップデートプログラムとファームウェアプログラムの動作について説明します。

ファームウェアアップデートプログラムはスタートアッププログラム保護機能のデフォルト領域と定数データ領域 1 に格納します。ファームウェアアップデートプログラムは、XMODEM/SUM プロトコルを使用したシリアル通信でファームウェアプログラム（.mot ファイル）を受信し、コードフラッシュメモリに書き込みます。このとき、スタートアッププログラム保護機能のデフォルト領域と代替領域の配置を一時的に変更し、デフォルト領域と定数データ領域以外のコードフラッシュメモリの書き換えを行います。これにより、電源の瞬断などによりコードフラッシュメモリの書き換えが失敗してもファームウェアアップデートプログラムは保護されており、ファームウェアアップデートプログラムを再度起動することにより、ファームウェアプログラムのアップデートを再度行うことができます。

ファームウェアプログラムは、ファームウェアアップデートプログラムを使用して、スタートアッププログラム保護領域のデフォルト領域と定数データ領域以外に書き込みます。ファームウェアプログラムはシリアル通信を使用してホスト PC にメッセージを出力します。ホスト PC からコマンドを受信すると、デフォルト領域と代替領域を入れ替えた後ソフトウェアリセットを実行します。これにより、ファームウェアアップデートプログラムが再度起動します。

ファームウェアアップデートプログラムとファームウェアプログラムの機能比較を表 5.2 に示します。

表 5.2　サンプルプログラムの機能比較

機能	ファームウェア アップデートプログラム	ファームウェア プログラム
コードフラッシュメモリの消去、書き込み	対応	非対応
スタートアッププログラム保護機能の領域入れ替えとソフトウェアリセット	対応	対応

5.2.1　ファームウェアアップデートプログラムの書き込み

「4.3 プロジェクトのデバッグ」の手順に従い、ファームウェアアップデートプログラムをスタートアッププログラム保護領域のデフォルト領域と定数データ領域 1 に書き込みます。または、ブートモードで起動し、Renesas Flash Programmer を用いて、ファームウェアアップデートプログラムをスタートアッププログラム保護領域のデフォルト領域と定数データ領域 1 に書き込みます。

なお、Renesas Flash Programmer の使用方法は、Renesas Flash Programmer のユーザーズマニュアルを参照してください。

5.2.2　　ファームウェアプログラムの書き込み

ファームウェアアップデートプログラムを使用してファームウェアプログラムを書き込む動作のフローを以下に示します。

1. シングルチップモードでファームウェアアップデートプログラムを起動します。ファームウェアアップデートプログラムは SCI を起動して、ホスト PC のターミナルソフトウェアにメニュー表示を出力します。

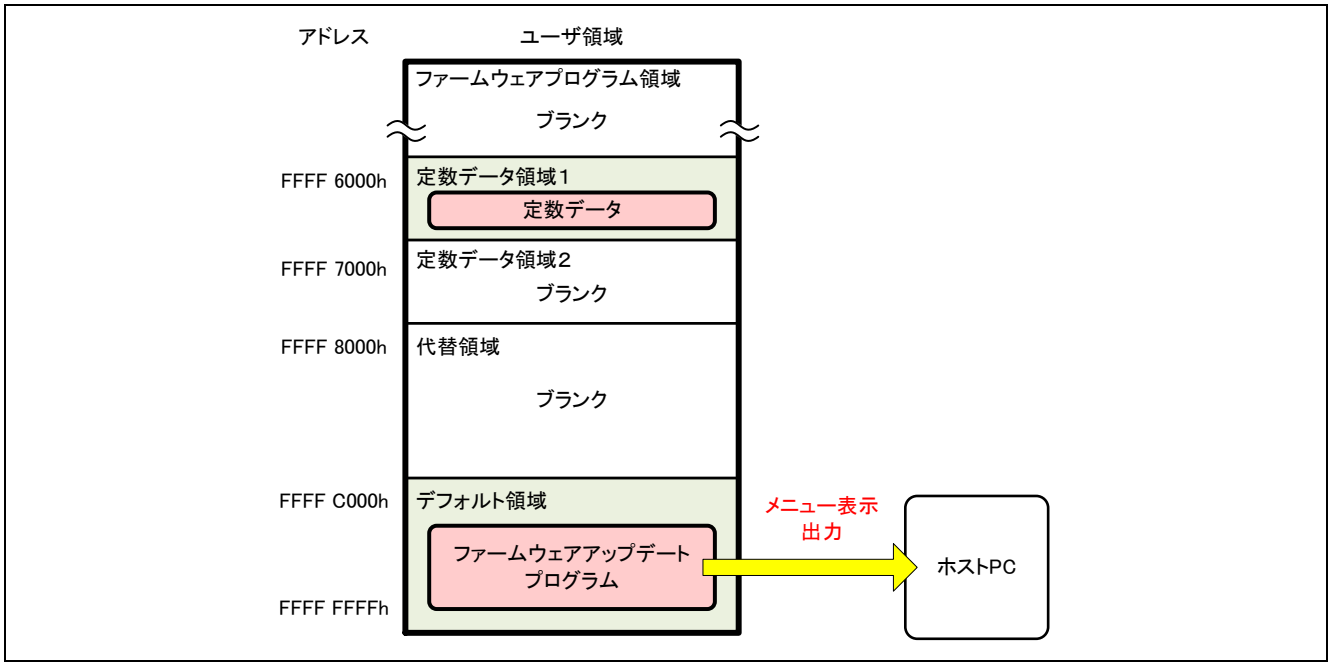


図 5.2　ファームウェアアップデートプログラム起動（RX231 グループの場合）

2. ファームウェアプログラムの書き込みを行うため、ターミナルソフトウェアからファームウェアプログラムアップデートコマンドを送信します。ファームウェアアップデートプログラムはコードフラッシュメモリの書き換えを内蔵 RAM 上で行うため、内蔵 RAM にフラッシュメモリ書き換え処理を展開します。

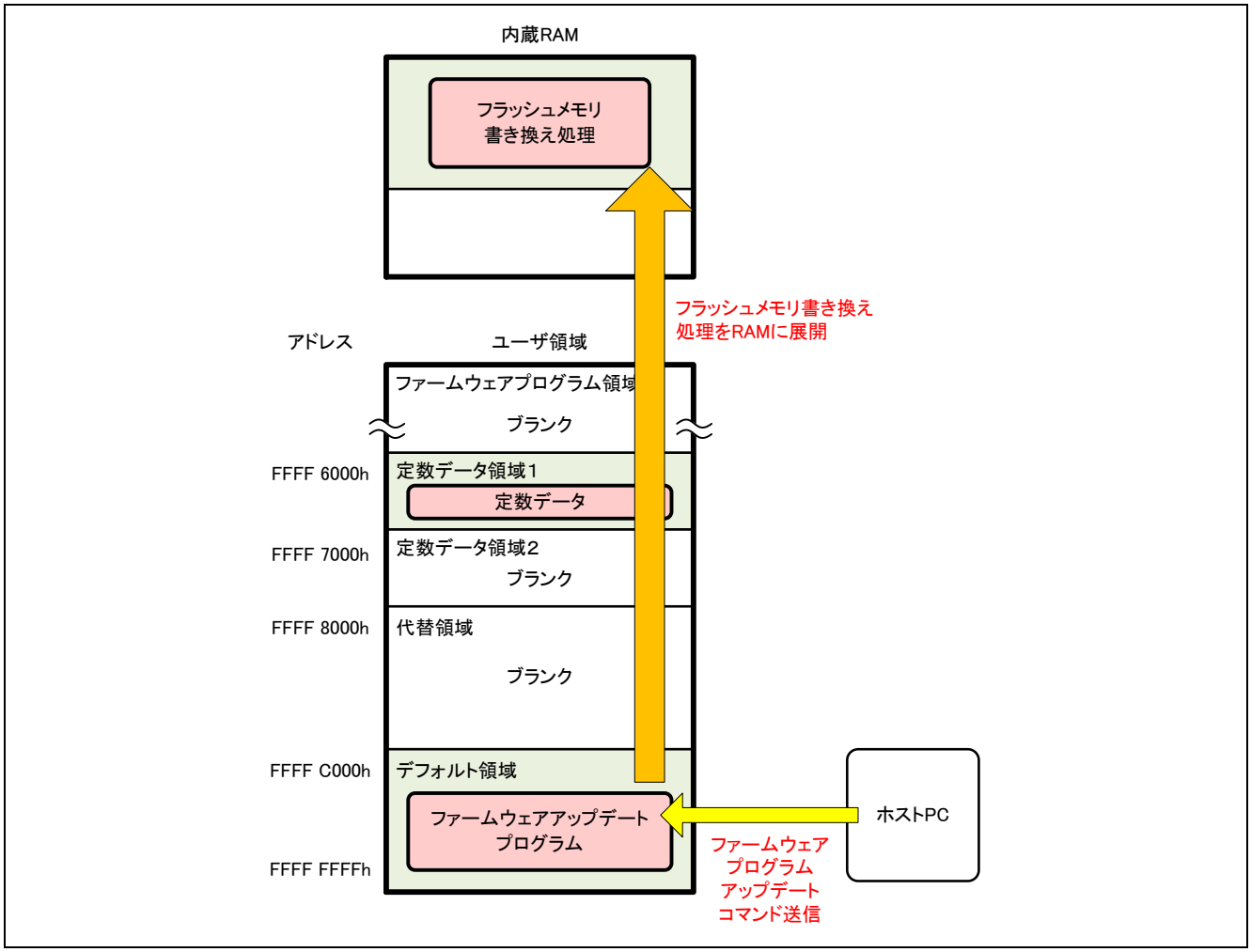


図 5.3　ファームウェアプログラムアップデートコマンド送信（RX231 グループの場合）

3. ファームウェアアップデートプログラムは内蔵 RAM 上のフラッシュメモリ書き換え処理に分岐して、代替領域とファームウェアプログラム領域を消去します。コードフラッシュメモリ消去後、デフォルト領域のファームウェアアップデートプログラムに復帰します。

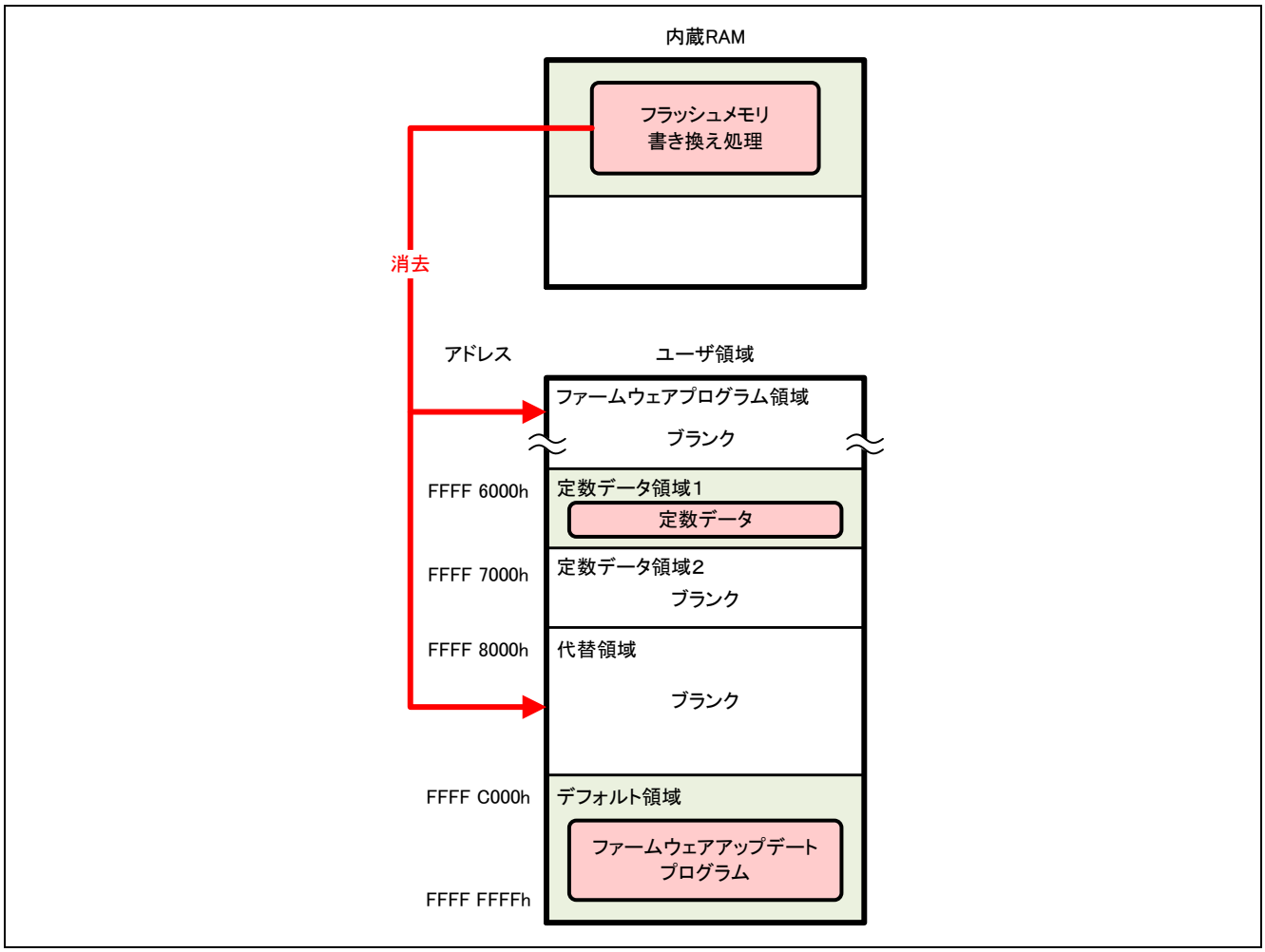


図 5.4　コードフラッシュメモリ消去（RX231 グループの場合）

4. ターミナルソフトウェアを使用して、ファームウェアプログラムを送信します。ファームウェアアップデートプログラムは、受信したデータを解析して、コードフラッシュメモリに書き込むデータを内蔵 RAM の書き込みバッファに格納します。

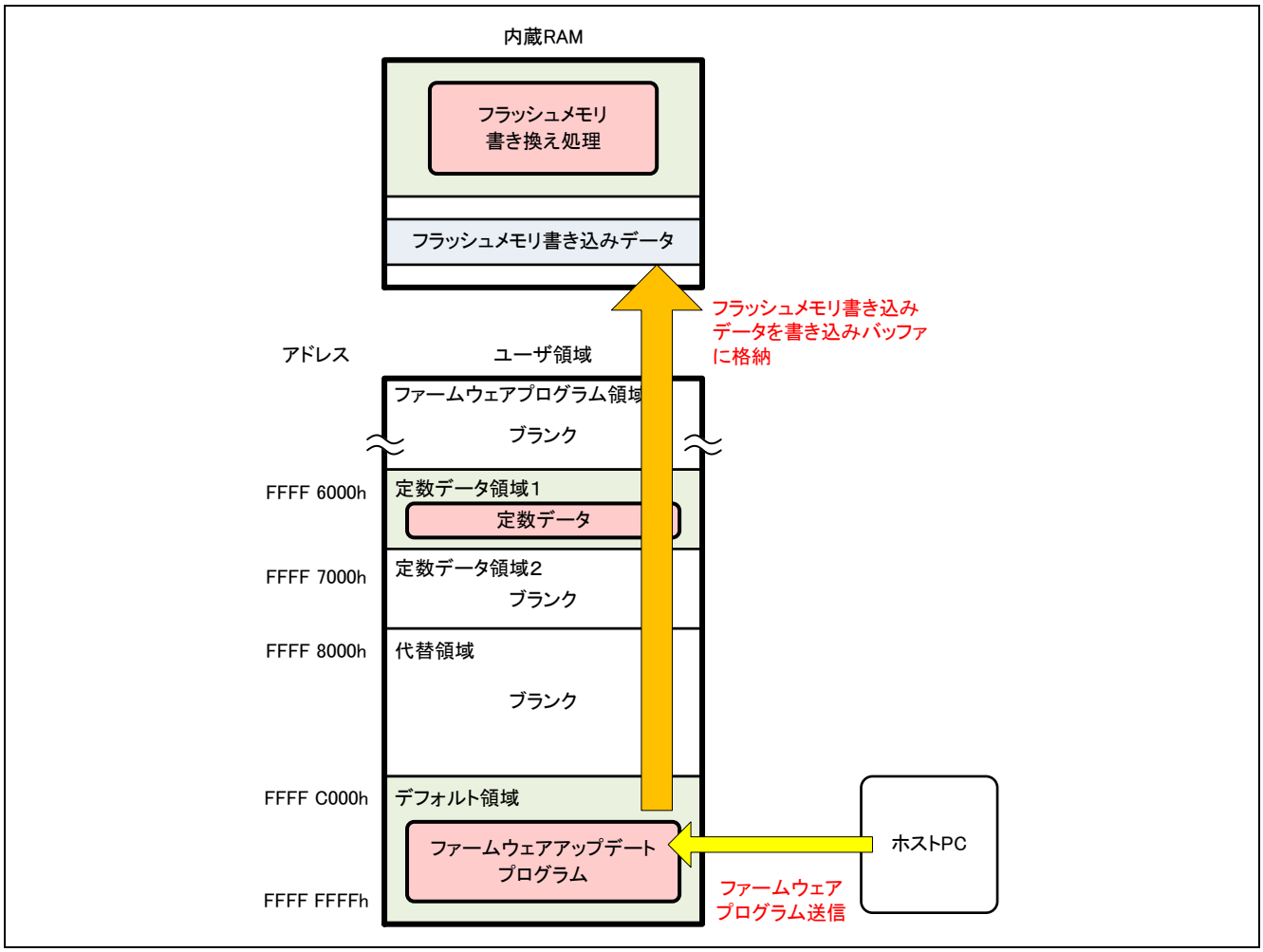


図 5.5　ファームウェアプログラムの送信（RX231 グループの場合）

5. 内蔵 RAM の書き込みバッファに書き込みデータが溜まると、ファームウェアアップデートプログラムは内蔵 RAM のフラッシュメモリ書き換え処理に分岐します。フラッシュメモリ書き換え処理は、フラッシュ初期設定レジスタ（FISR）の設定により、スタートアッププログラム保護機能のデフォルト領域と代替領域を一時的に入れ替えます。その後、書き込みバッファの書き込みデータをコードフラッシュメモリに書き込みます。書き込みが終了するとスタートアッププログラム保護機能の領域を元に戻します。

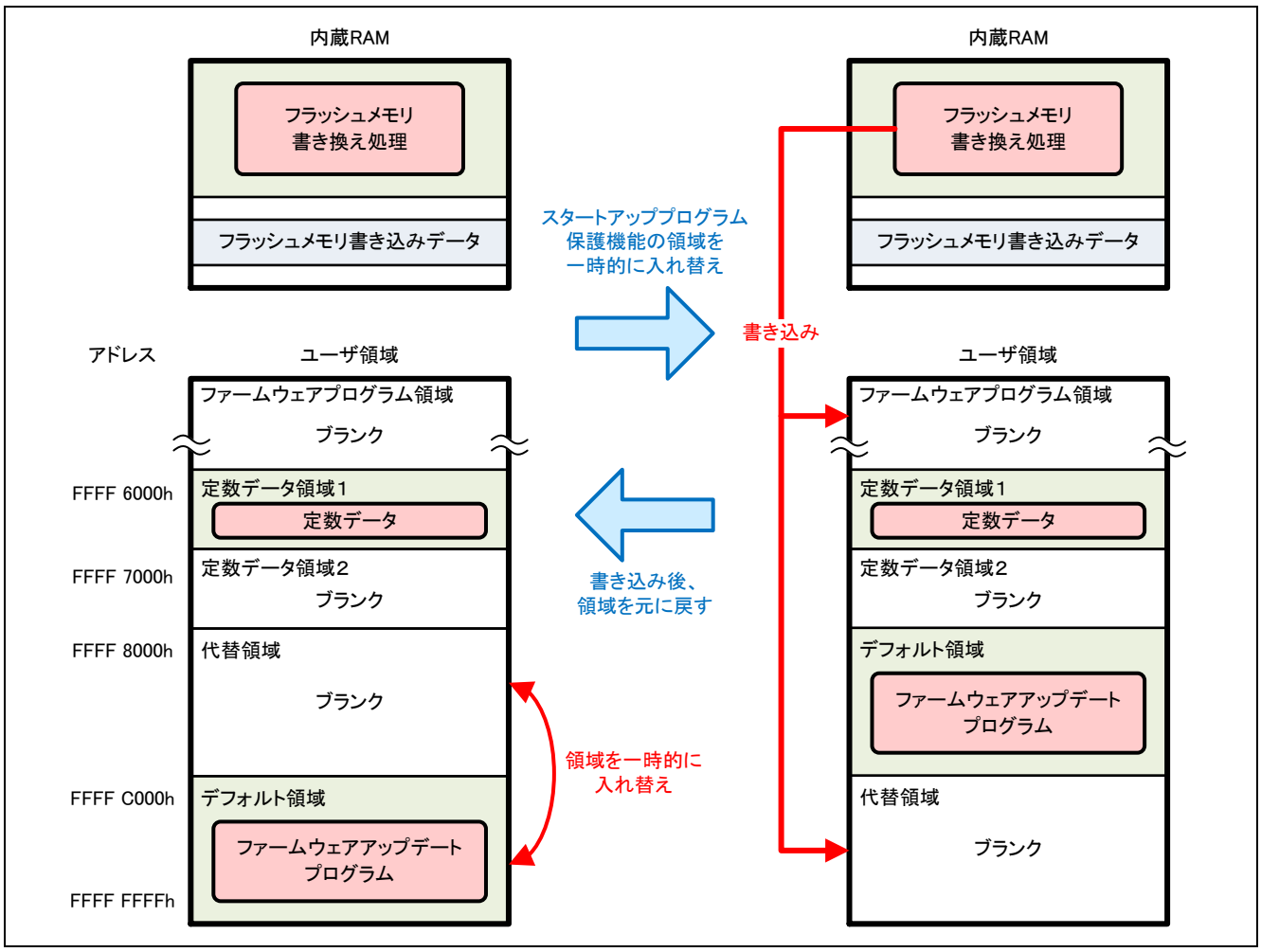


図 5.6　ファームウェアプログラムの書き込み（RX231 グループの場合）

6.　4 と 5 の手順を繰り返し、ファームウェアプログラムの全データを書き込みます。

7.　ファームウェアプログラムの書き込みが終了したのち、ファームウェア起動処理に分岐します。ファームウェアアップデートプログラムは、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、ソフトウェアリセットを実行します。ファームウェアプログラムが起動します。

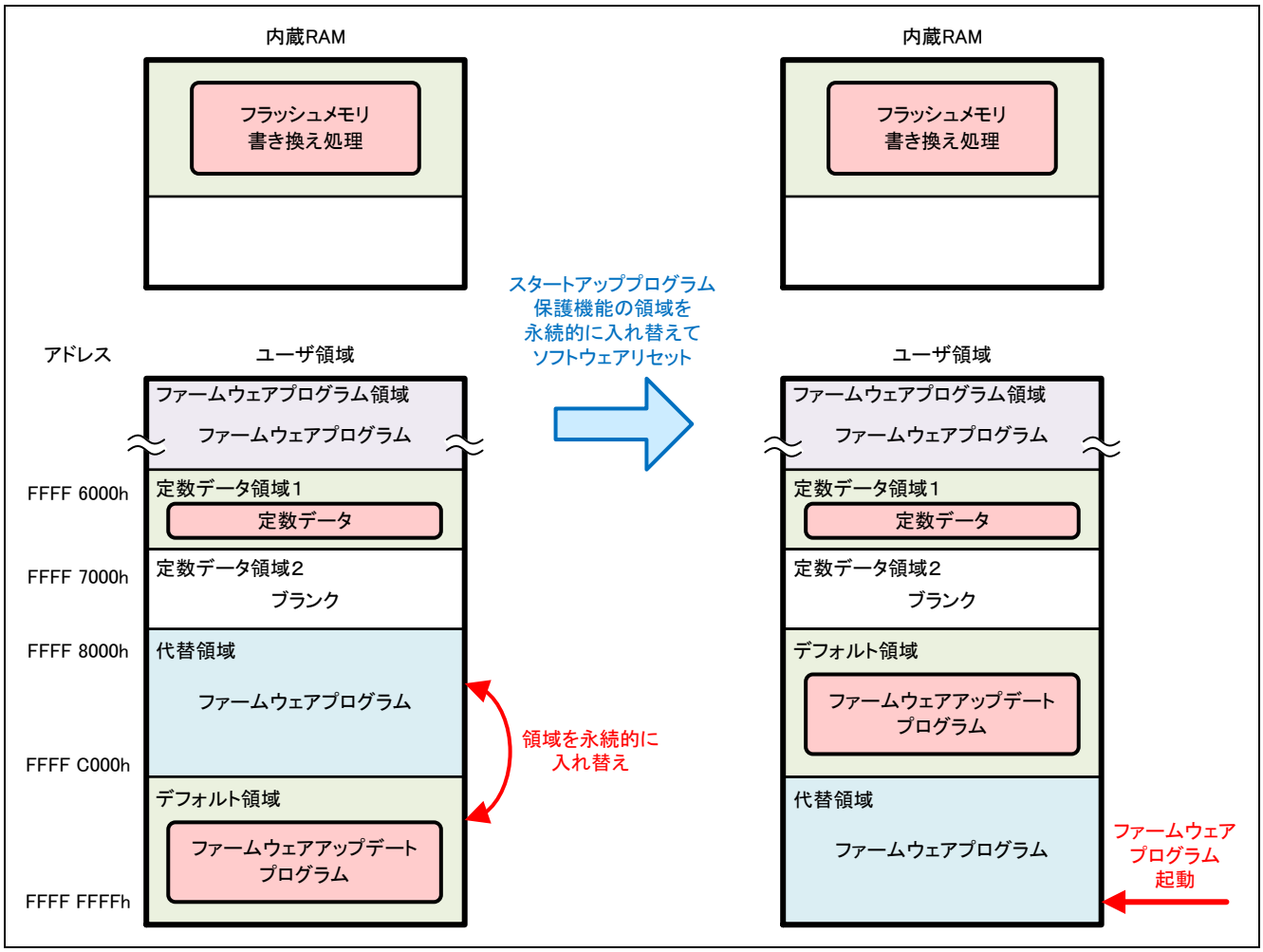


図 5.7　ソフトウェアリセットとファームウェアプログラムの起動（RX231 グループの場合）

5.2.3　　ファームウェアプログラムの更新

ファームウェアアップデートプログラムを使用してファームウェアプログラムを更新する動作のフローを以下に示します。

1. 更新前のファームウェアプログラムにおいて、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、デバイスをリセットします。

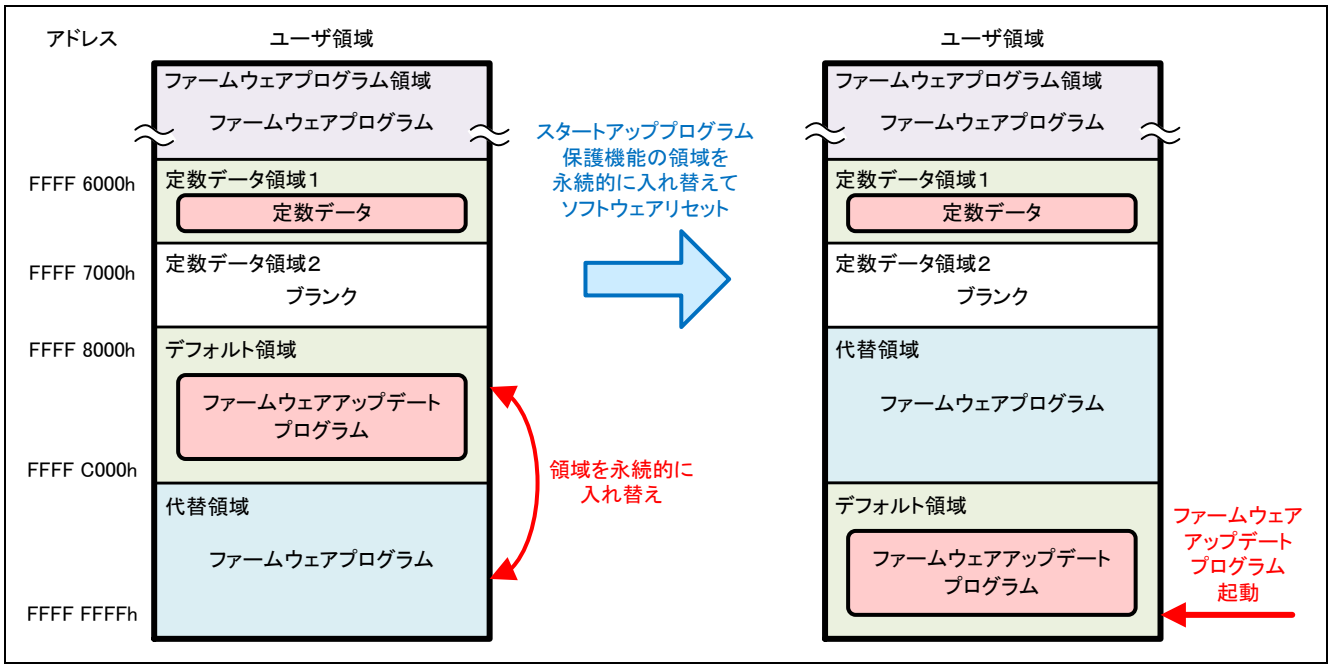


図 5.8　更新前のファームウェアプログラムによる動作（RX231 グループの場合）

2. ファームウェアアップデートプログラムが起動します。ファームウェアアップデートプログラムは SCI を起動して、ホスト PC のターミナルソフトウェアにメニュー表示を出力します。

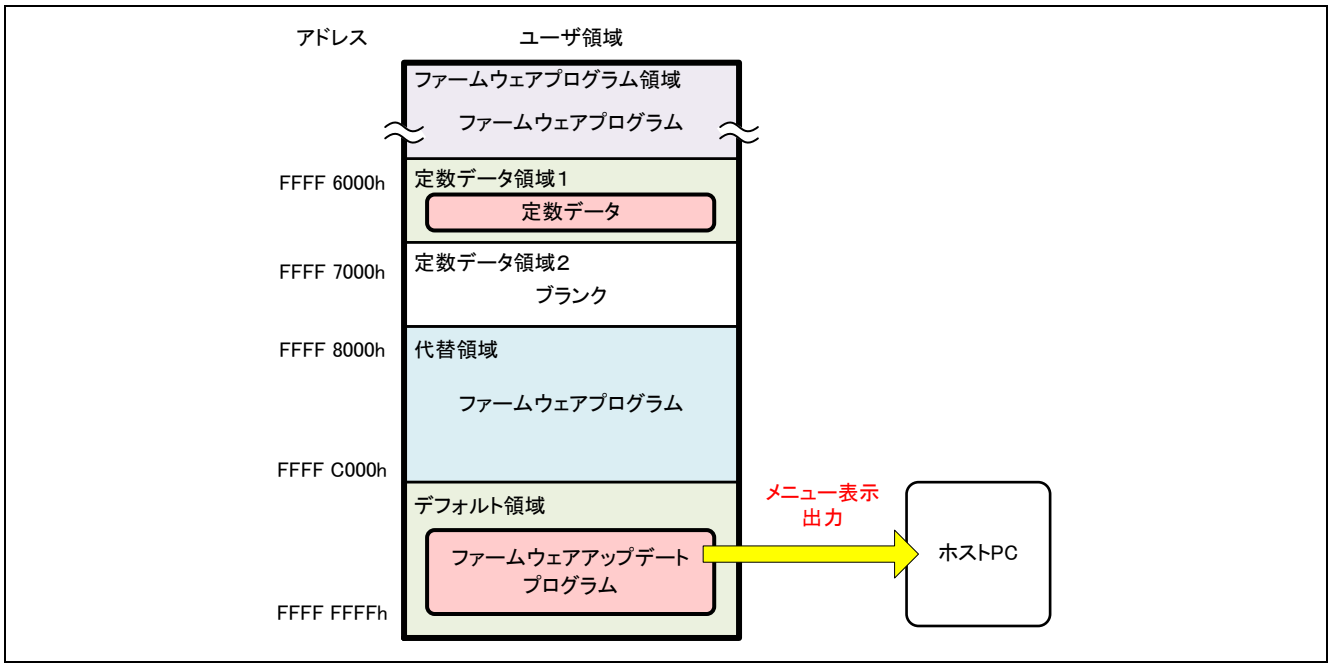


図 5.9　ファームウェアアップデートプログラム起動（RX231 グループの場合）

3. ファームウェアプログラムの更新を行うため、ターミナルソフトウェアからファームウェアプログラムアップデートコマンドを送信します。ファームウェアプログラムアップデートプログラムはコードフラッシュメモリの書き換えを内蔵 RAM 上で行うため、内蔵 RAM にフラッシュメモリ書き換え処理を展開します。

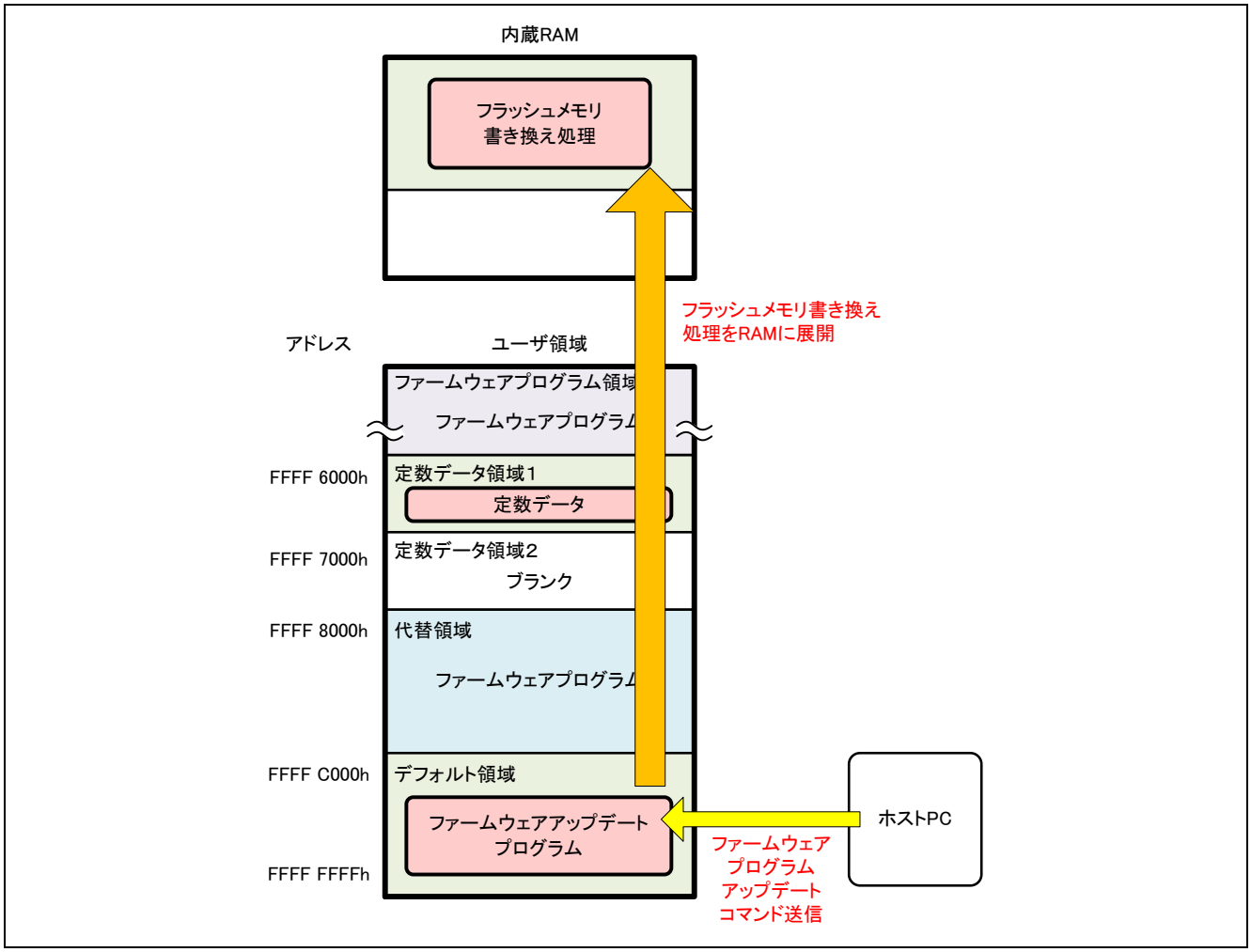


図 5.10　ファームウェアプログラムアップデートコマンド送信（RX231 グループの場合）

4. ファームウェアアップデートプログラムは内蔵 RAM 上のフラッシュメモリ書き換え処理に分岐して、代替領域とファームウェアプログラム領域を消去します。コードフラッシュメモリ消去後、デフォルト領域のファームウェアアップデートプログラムに復帰します。

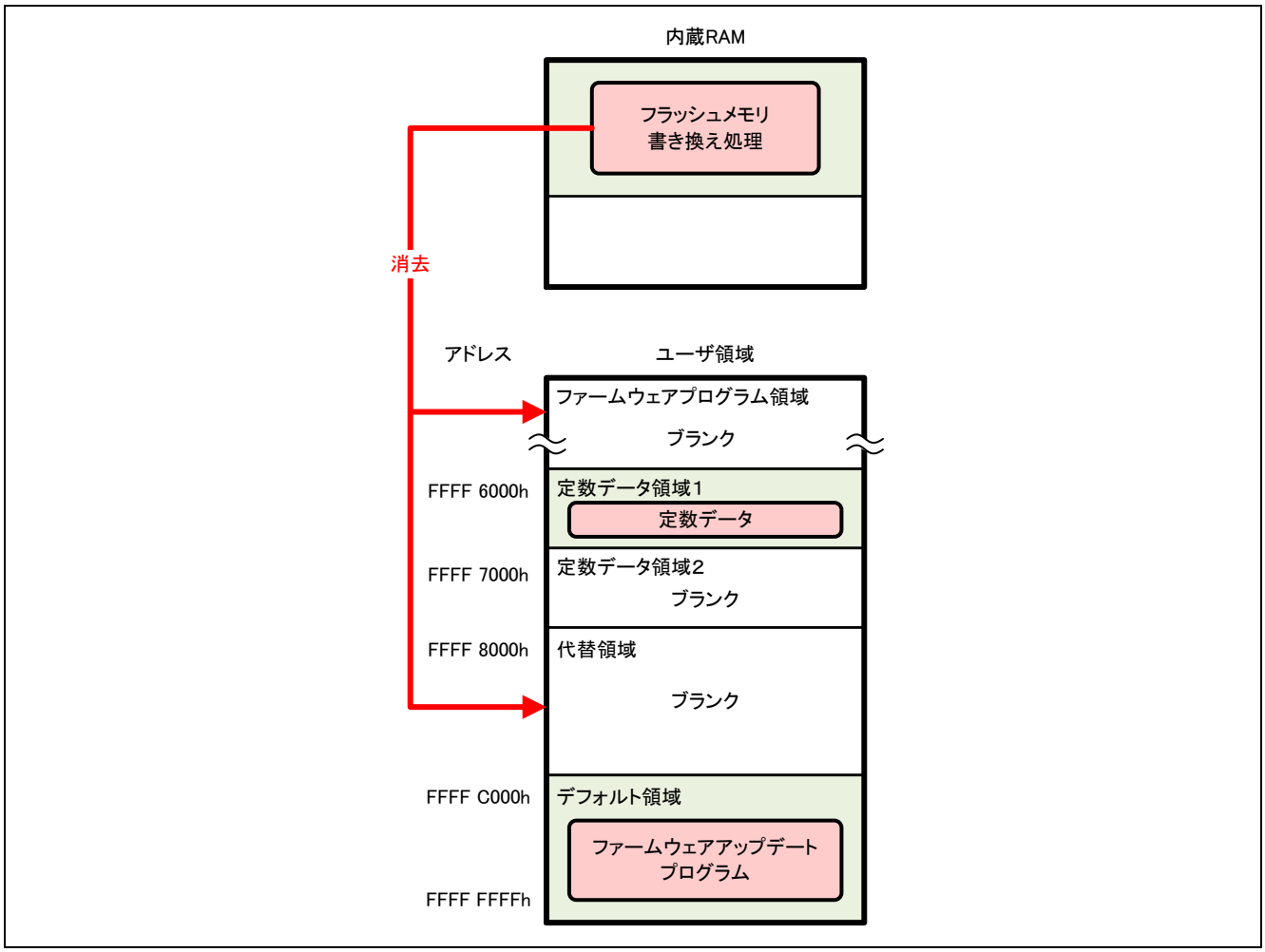


図 5.11　コードフラッシュメモリ消去（RX231 グループの場合）

5. ターミナルソフトウェアを使用して、新しいファームウェアプログラムを送信します。ファームウェアアップデートプログラムは、受信したデータを解析して、コードフラッシュメモリに書き込むデータを内蔵 RAM の書き込みバッファに格納します。

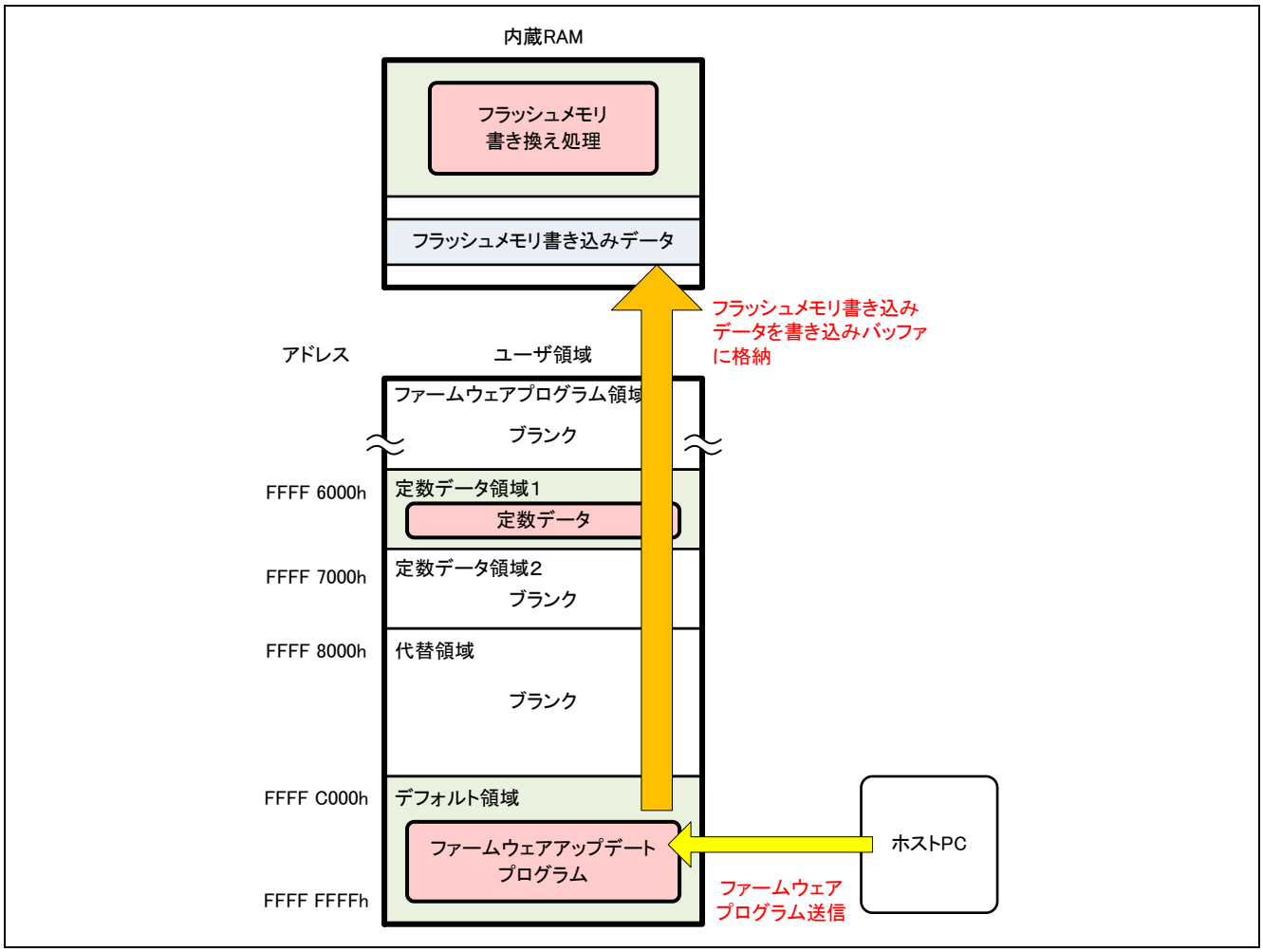


図 5.12　ファームウェアプログラムの送信（RX231 グループの場合）

6. 内蔵 RAM の書き込みバッファに書き込みデータが溜まると、ファームウェアアップデートプログラムは内蔵 RAM のフラッシュメモリ書き換え処理に分岐します。フラッシュメモリ書き換え処理は、フラッシュ初期設定レジスタ（FISR）の設定により、スタートアッププログラム保護機能のデフォルト領域と代替領域を一時的に入れ替えます。その後、書き込みバッファの書き込みデータをコードフラッシュメモリに書き込みます。書き込みが終了するとスタートアッププログラム保護機能の領域を元に戻します。

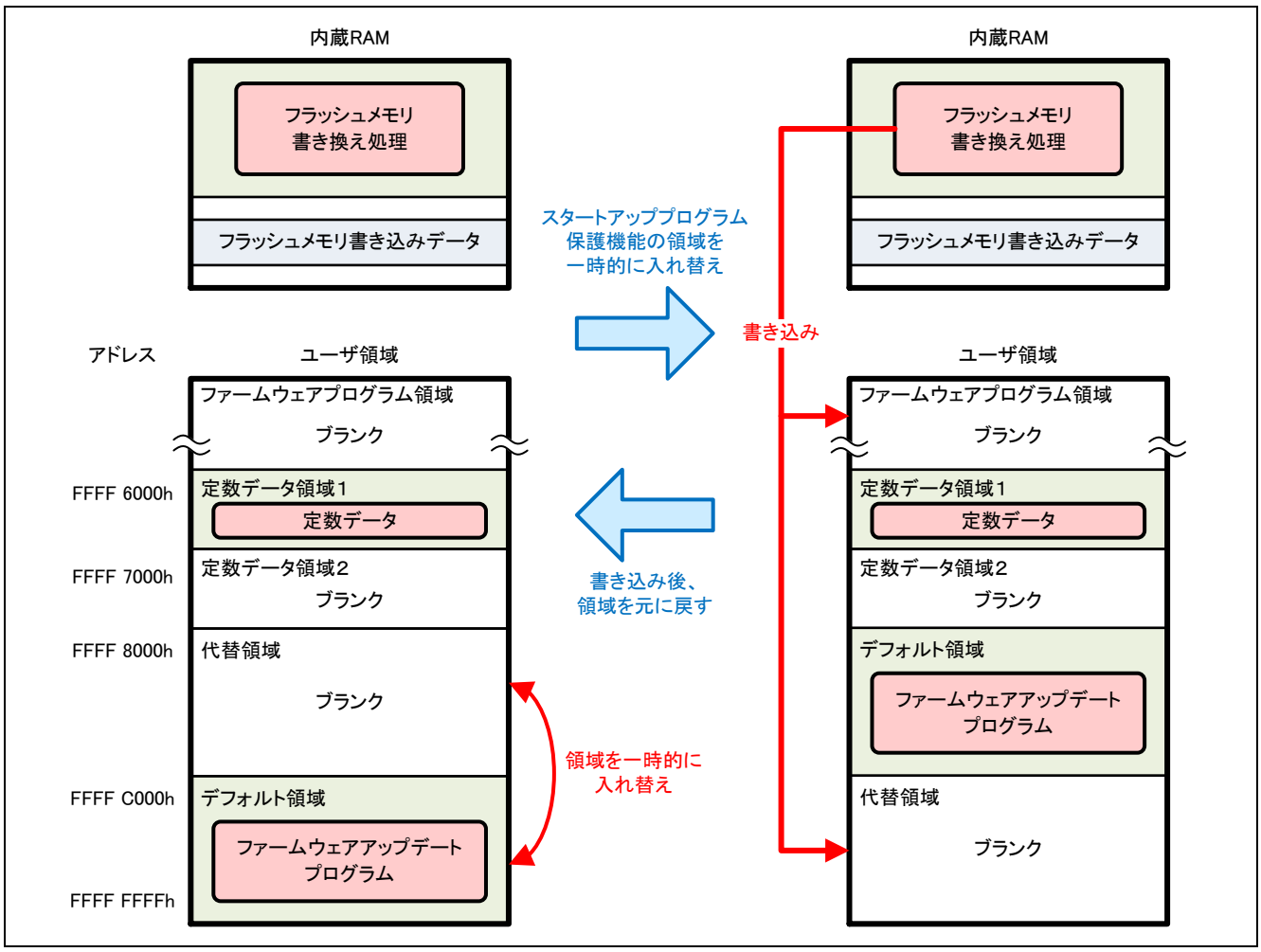


図 5.13　ファームウェアプログラムの書き込み（RX231 グループの場合）

7. 5 と 6 の手順を繰り返し、新しいファームウェアプログラムの全データを書き込みます。

8. 新しいファームウェアプログラムの書き込みが終了したのち、ファームウェア起動処理に分岐します。ファームウェアアップデートプログラムは、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、ソフトウェアリセットを実行します。更新されたファームウェアプログラムが起動します。

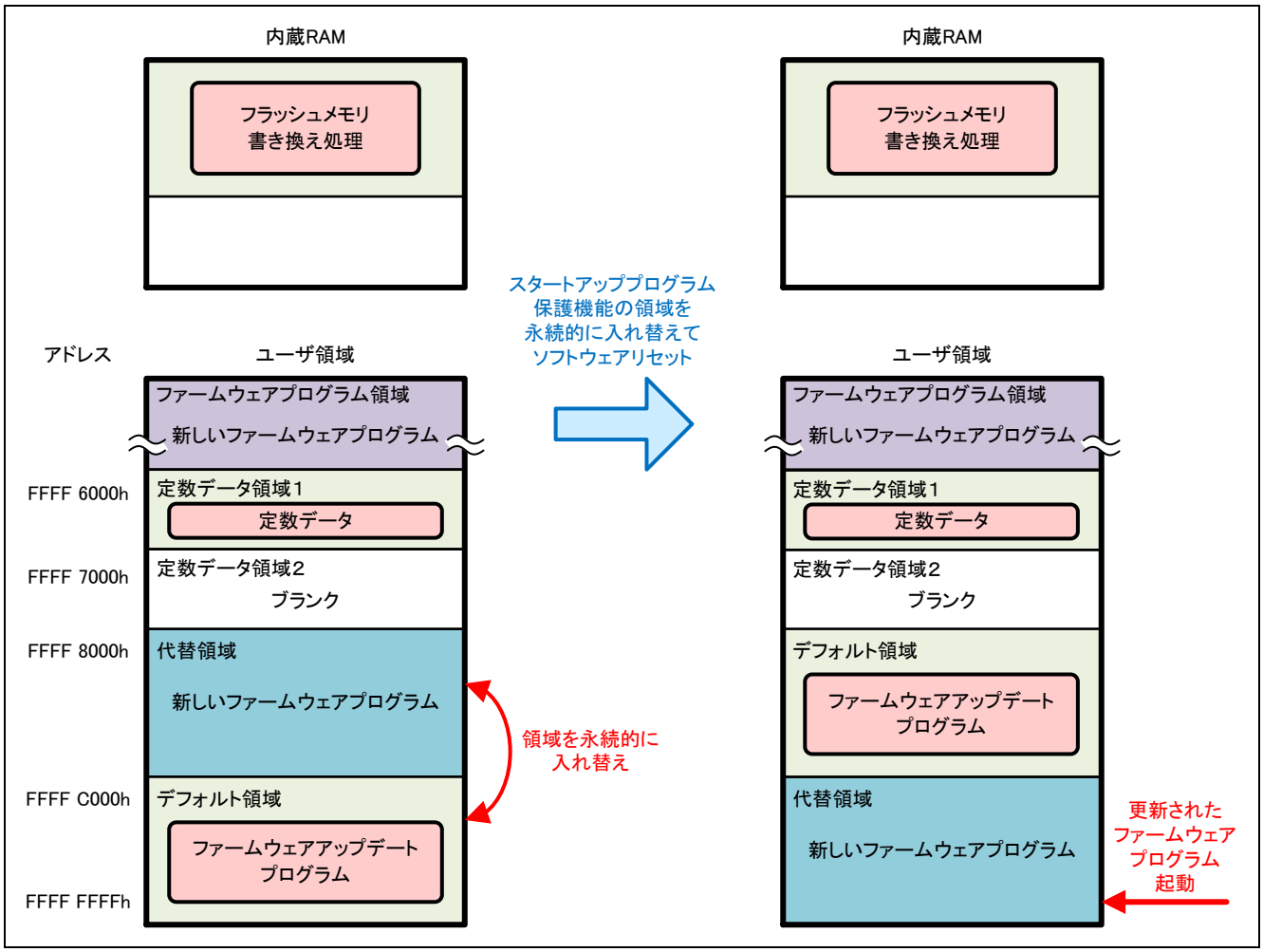


図 5.14　ソフトウェアリセットと更新されたファームウェアプログラムの起動（RX231 グループの場合）

5.2.4　ファームウェアアップデートプログラムの更新

ファームウェアアップデートプログラムを使用してファームウェアアップデートプログラムを更新する動作のフローを以下に示します。なお、ファームウェアアップデートプログラムの更新時にコードフラッシュメモリに書き込まれているファームウェアプログラムを消去するため、ファームウェアアップデートプログラムを更新したのちファームウェアプログラムの書き込みを行ってください。また、更新に使用するファームウェアアップデートプログラムは設定を変更する必要があります。変更内容については「5.2.7 更新用ファームウェアアップデートプログラムの作成方法」を参照してください。

1. ファームウェアプログラムにおいて、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、デバイスをリセットします。

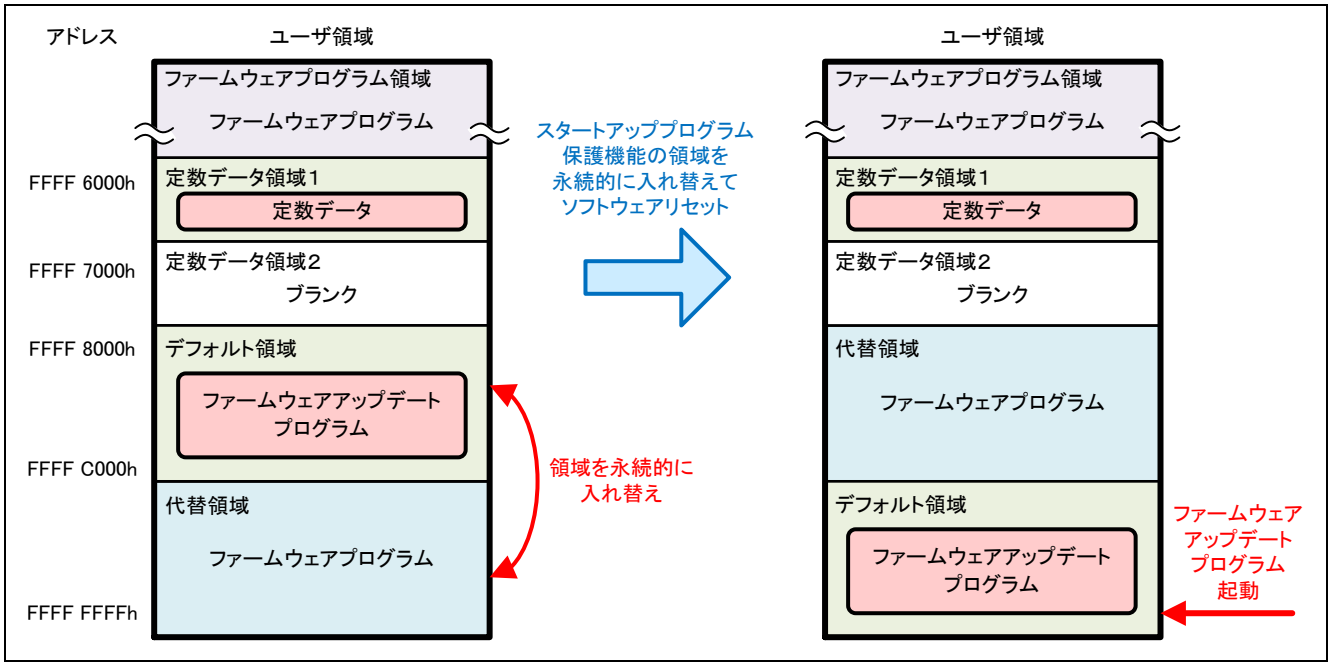


図 5.15　ファームウェアプログラムによる動作（RX231 グループの場合）

2. ファームウェアアップデートプログラムが起動します。ファームウェアアップデートプログラムは SCI を起動して、ホスト PC のターミナルソフトウェアにメニュー表示を出力します。

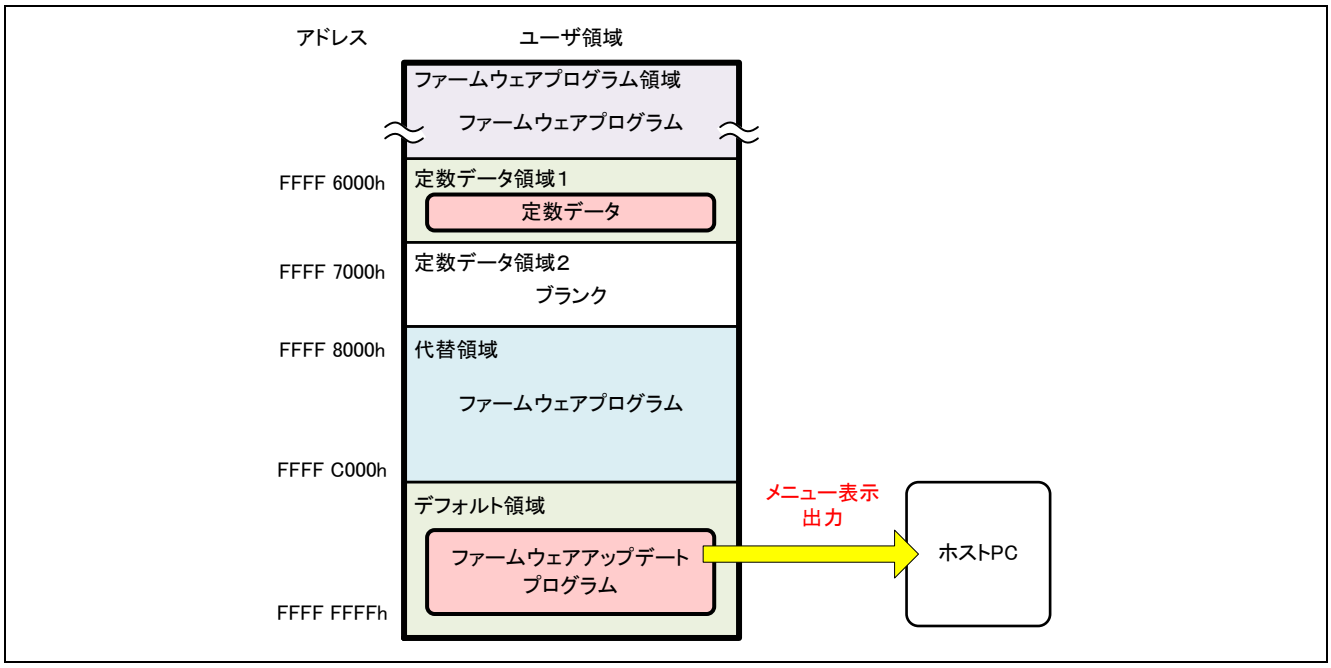


図 5.16　ファームウェアアップデートプログラム起動（RX231 グループの場合）

3. ファームウェアアップデートプログラムの更新を行うため、ターミナルソフトウェアからファームウェアアップデートプログラムアップデートコマンドを送信します。ファームウェアアップデートプログラムはコードフラッシュメモリの書き換えを内蔵 RAM 上で行うため、内蔵 RAM にフラッシュメモリ書き換え処理を展開します。

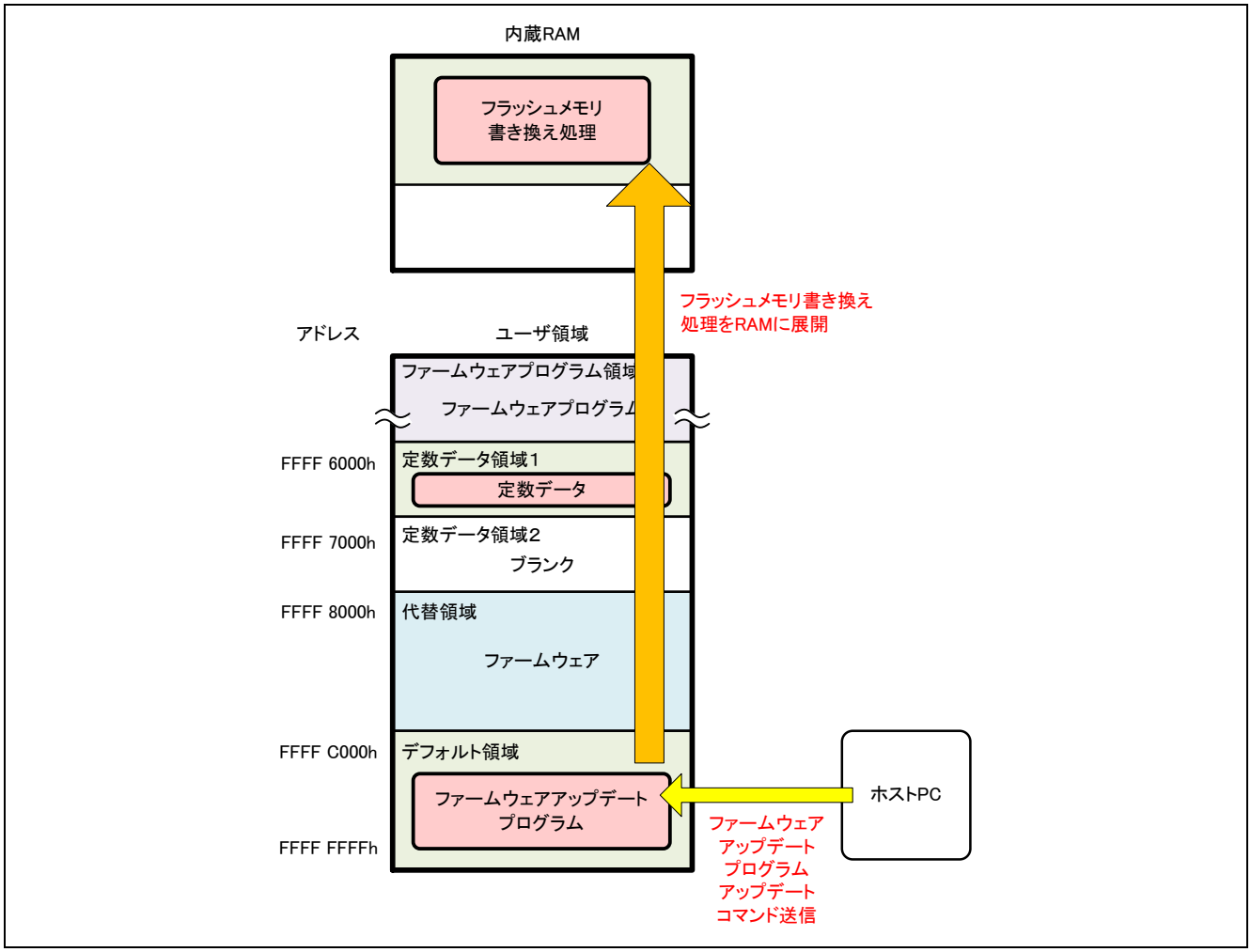


図 5.17　ファームウェアアップデートプログラムアップデートコマンド送信（RX231 グループの場合）

4. ファームウェアアップデートプログラムは内蔵 RAM 上のフラッシュメモリ書き換え処理に分岐して、代替領域と定数データ領域 2 を消去します。*1 コードフラッシュメモリ消去後、デフォルト領域のファームウェアアップデートプログラムに復帰します。

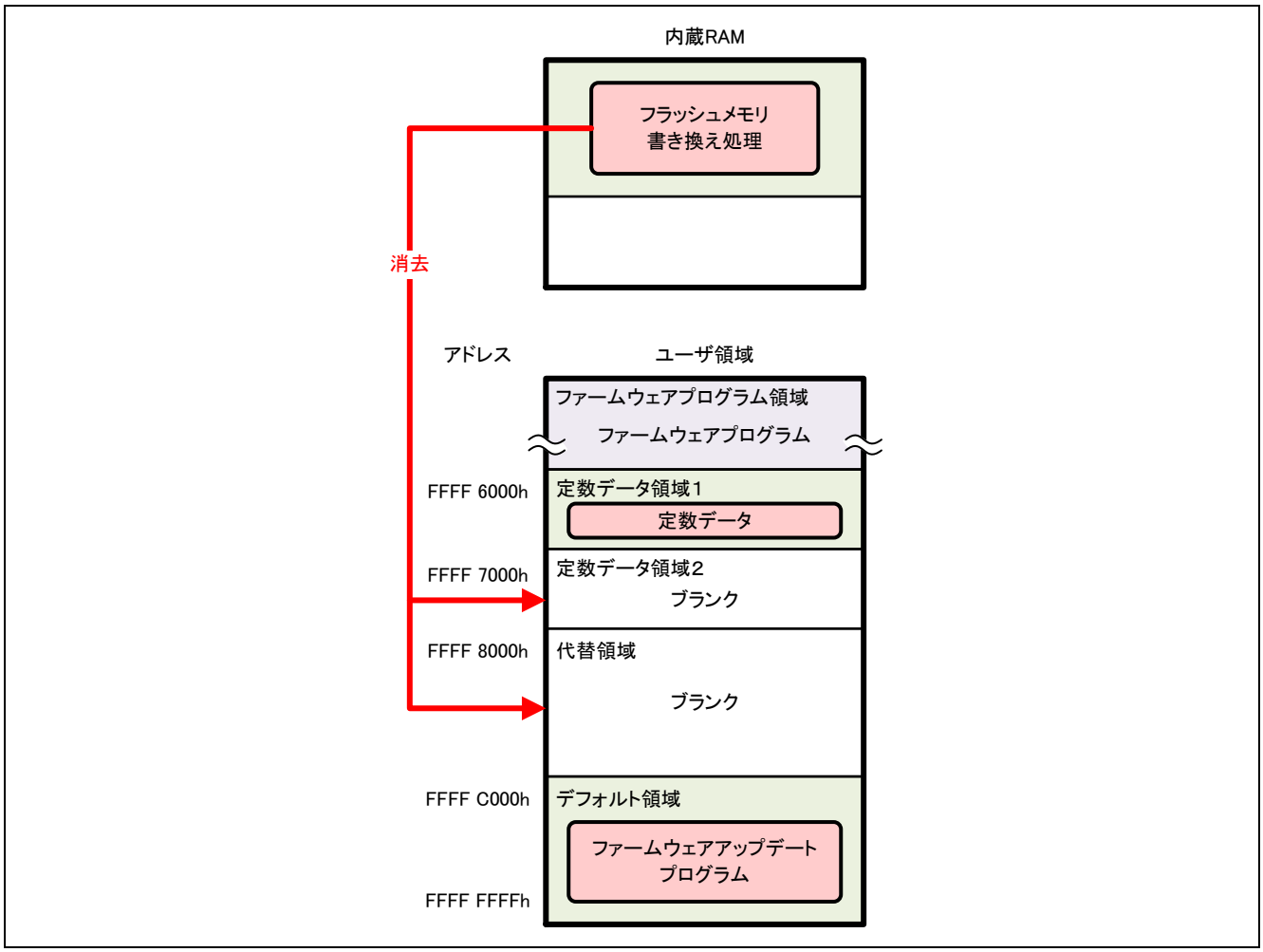


図 5.18　コードフラッシュメモリ消去（RX231 グループの場合）

【注】*1 ファームウェアアップデートプログラムが定数データ領域 2 を使用している場合、代替領域と定数データ領域 1 を消去します。

5. ターミナルソフトウェアを使用して、新しいファームウェアアップデートプログラムを送信します。更新前のファームウェアアップデートプログラムは、受信したデータを解析して、コードフラッシュメモリに書き込むデータを内蔵 RAM の書き込みバッファに格納します。

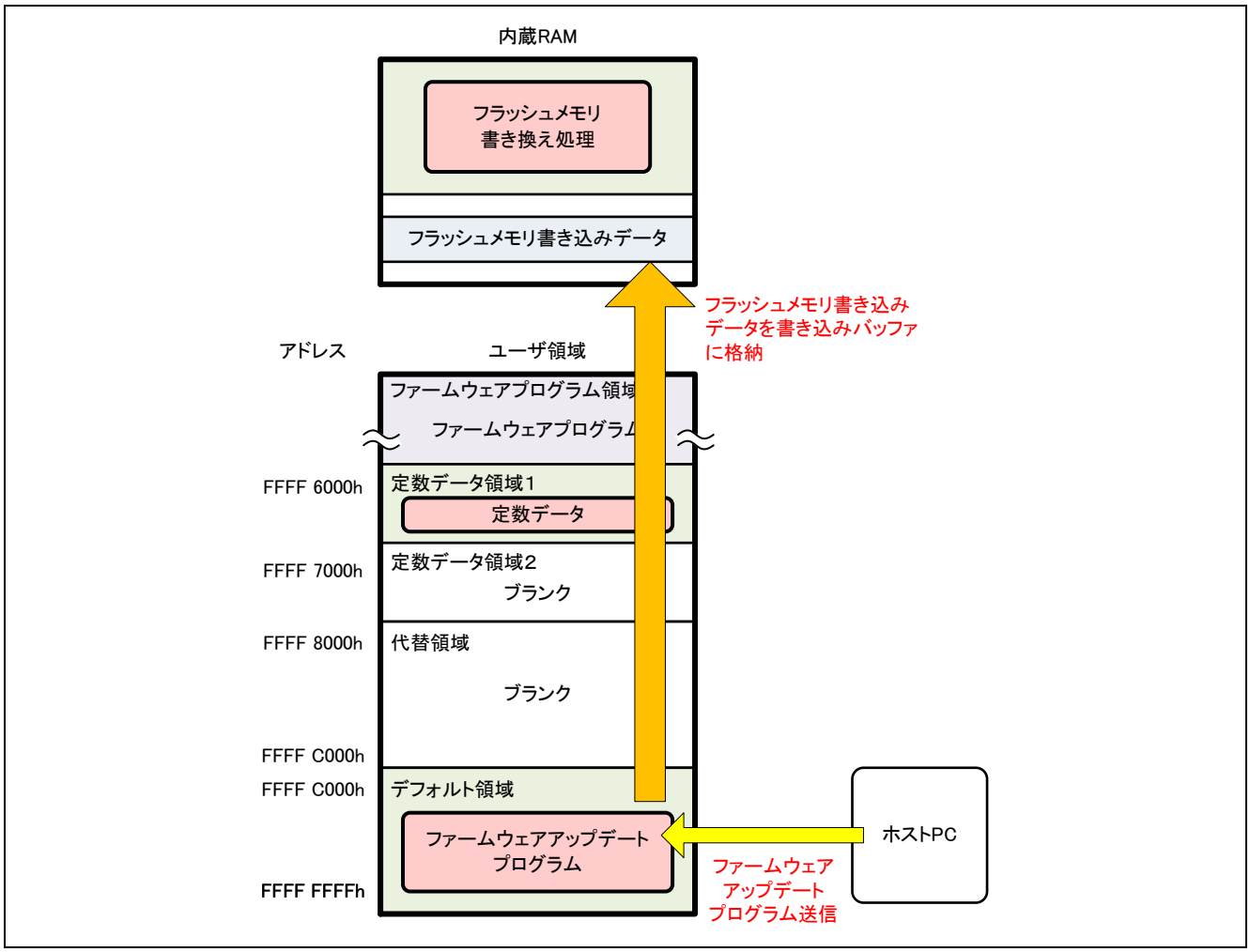


図 5.19　ファームウェアアップデートプログラムの送信（RX231 グループの場合）

6. 内蔵 RAM の書き込みバッファに書き込みデータが溜まると、ファームウェアアップデートプログラムは内蔵 RAM のフラッシュメモリ書き換え処理に分岐します。フラッシュメモリ書き換え処理は、フラッシュ初期設定レジスタ（FISR）の設定により、スタートアッププログラム保護機能のデフォルト領域と代替領域を一時的に入れ替えます。その後、書き込みバッファの書き込みデータをコードフラッシュメモリに書き込みます。書き込みが終了するとスタートアッププログラム保護機能の領域を元に戻します。

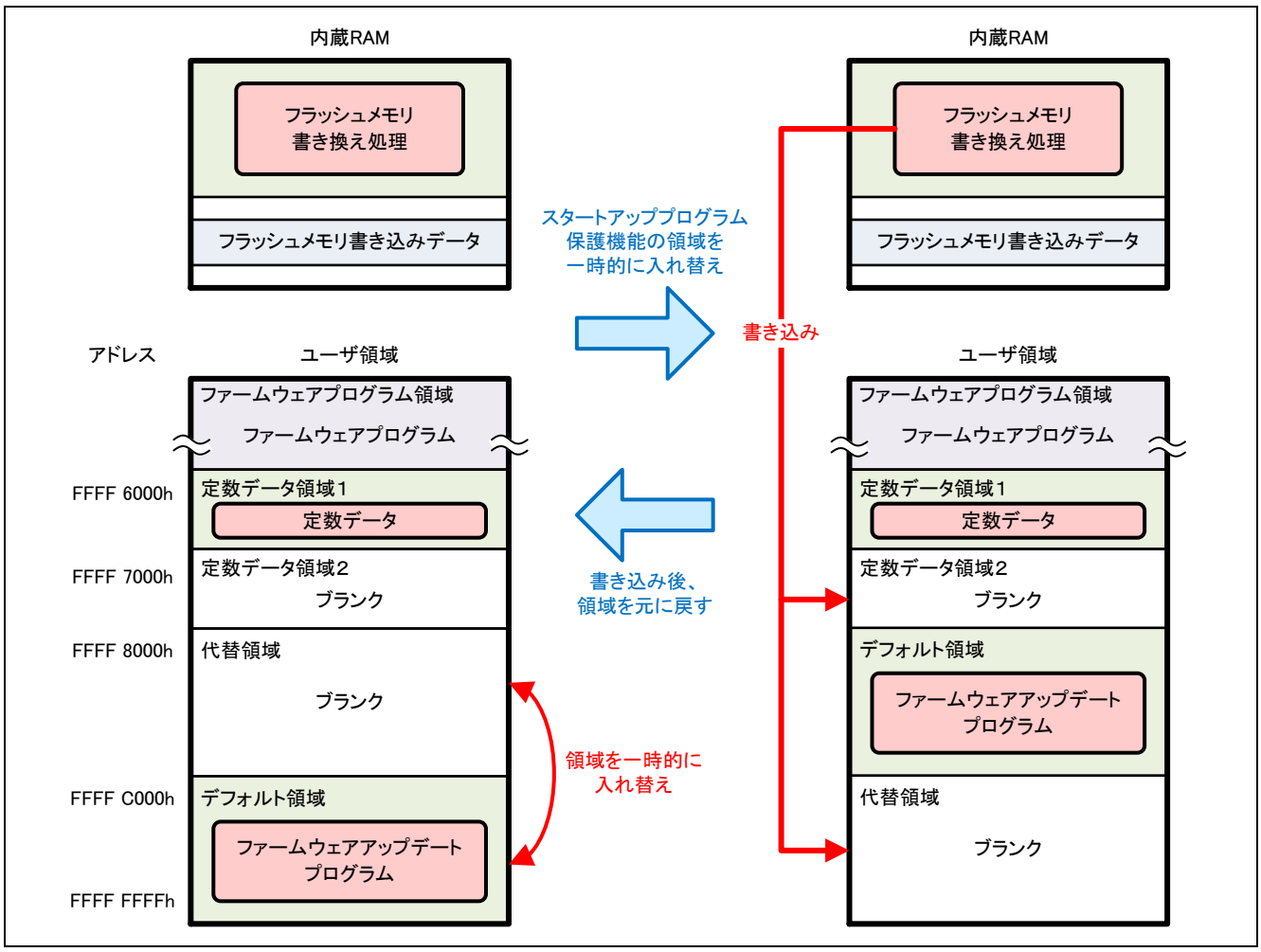


図 5.20　ファームウェアアップデートプログラムの書き込み（RX231 グループの場合）

7. 5 と 6 の手順を繰り返し、新しいファームウェアアップデートプログラムの全データを書き込みます。

8. 新しいファームウェアアップデートプログラムの書き込みが終了したのち、定数データ領域 2 のバージョン情報格納領域 2 から新しいファームウェアアップデートプログラムのバージョン情報を取得して、書き込み完了情報格納領域 2 に書き込みます。

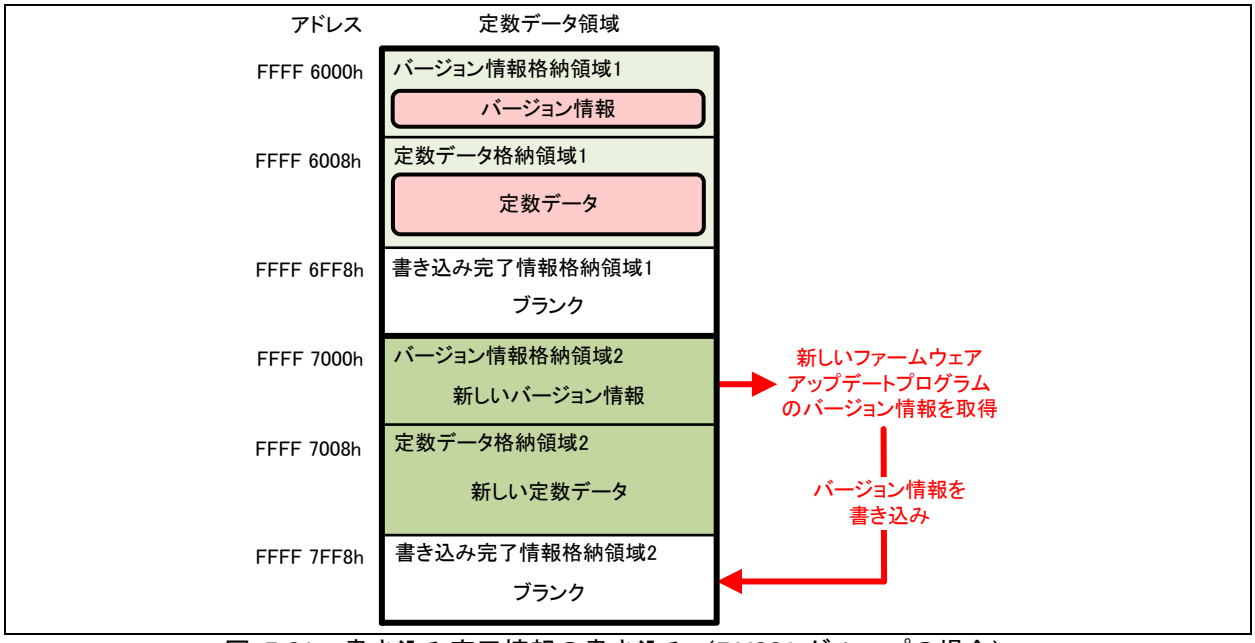


図 5.21　書き込み完了情報の書き込み（RX231 グループの場合）

9. 書き込み完了情報格納領域への書き込みが終了したのち、ファームウェア起動処理に分岐します。更新前のファームウェアアップデートプログラムは、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、ソフトウェアリセットを実行します。更新されたファームウェアアップデートプログラムが起動します。

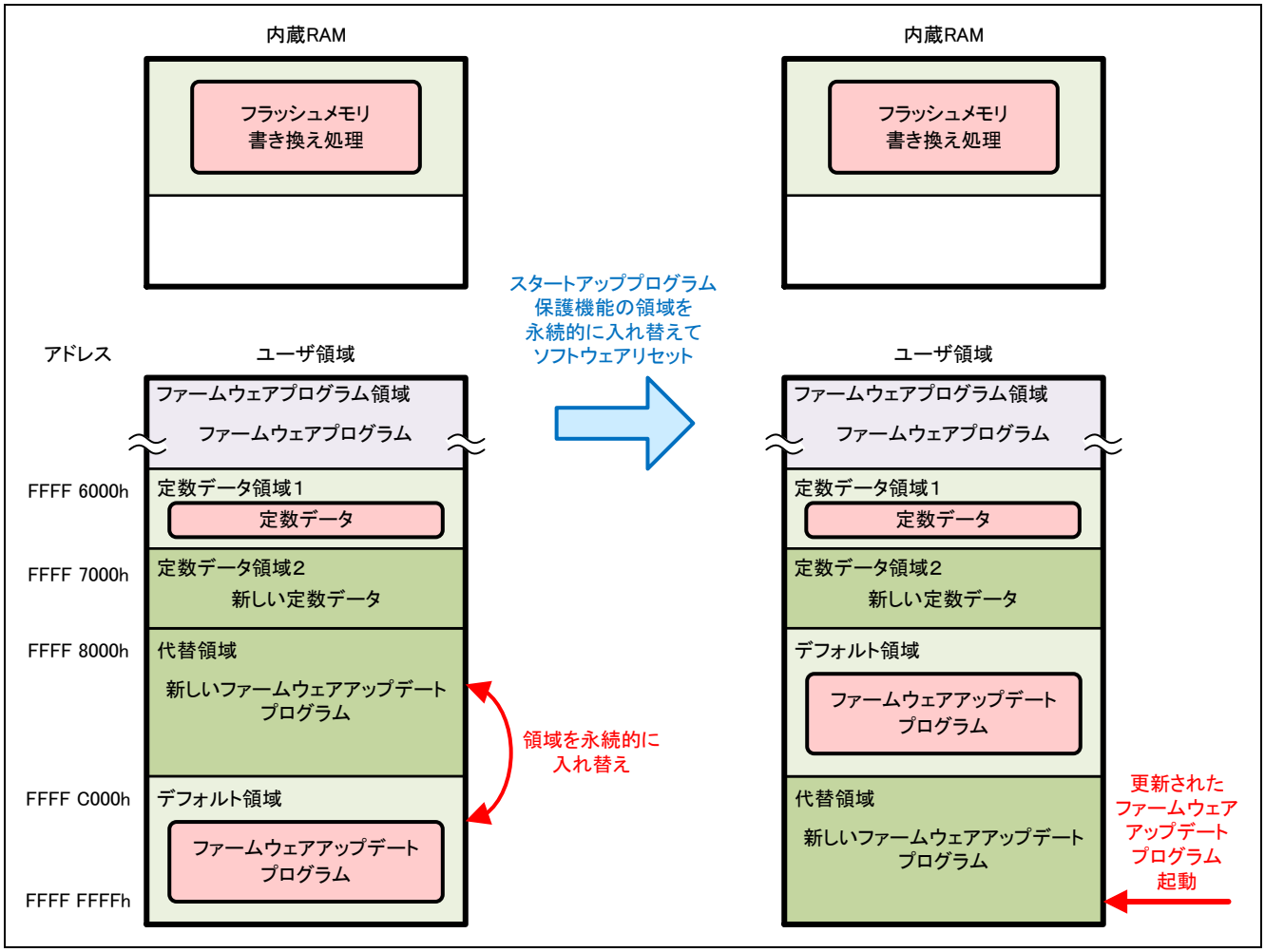


図 5.22　ソフトウェアリセットと更新されたファームウェアアップデートプログラムの起動（RX231 グループの場合）

5.2.5　　ファームウェアプログラムの動作

ファームウェアプログラムを使用したファームウェアアップデートプログラムの動作確認のフローを以下に示します。

1.　「5.2.2 ファームウェアプログラムの書き込み」の手順に従って、ファームウェアプログラムを書き込みます。

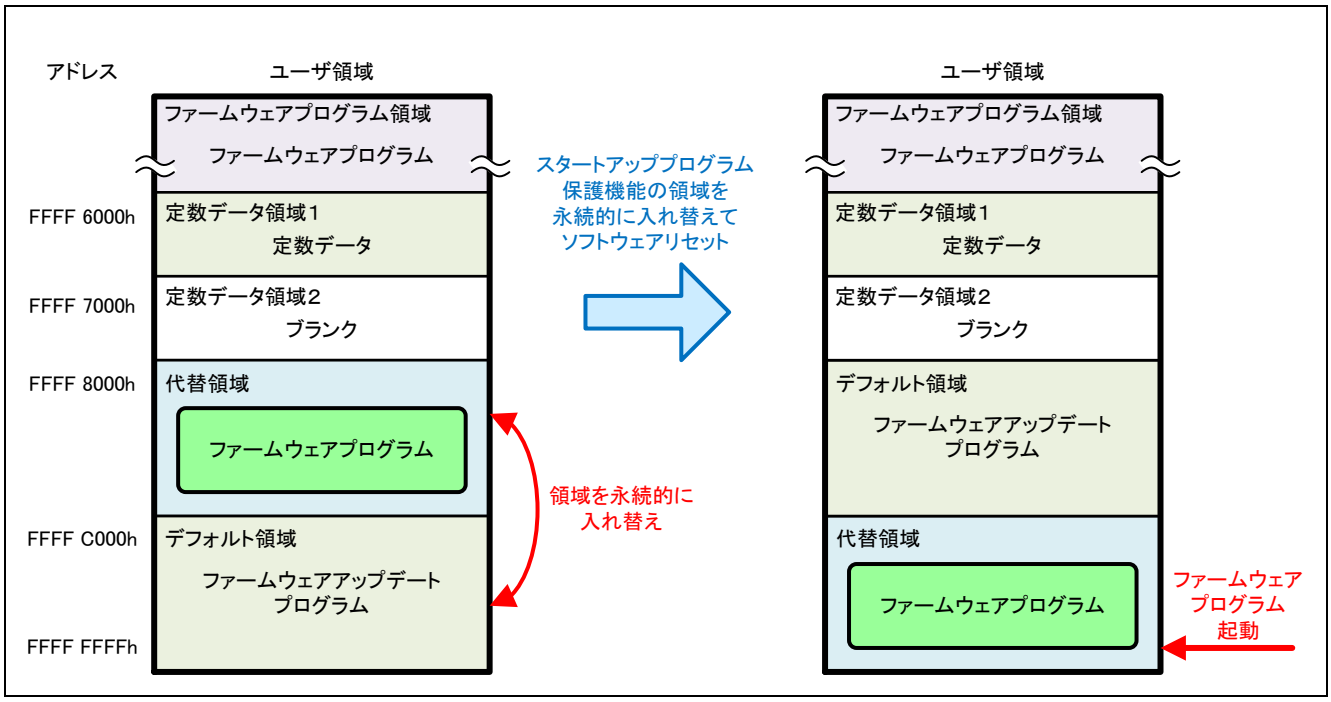


図 5.23　ファームウェアアップデートプログラムによるファームウェアプログラムの書き込みと起動
（RX231 グループの場合）

2. ファームウェアプログラムが起動します。ファームウェアプログラムは SCI を起動して、ホスト PC のターミナルソフトウェアにメッセージを出力します。

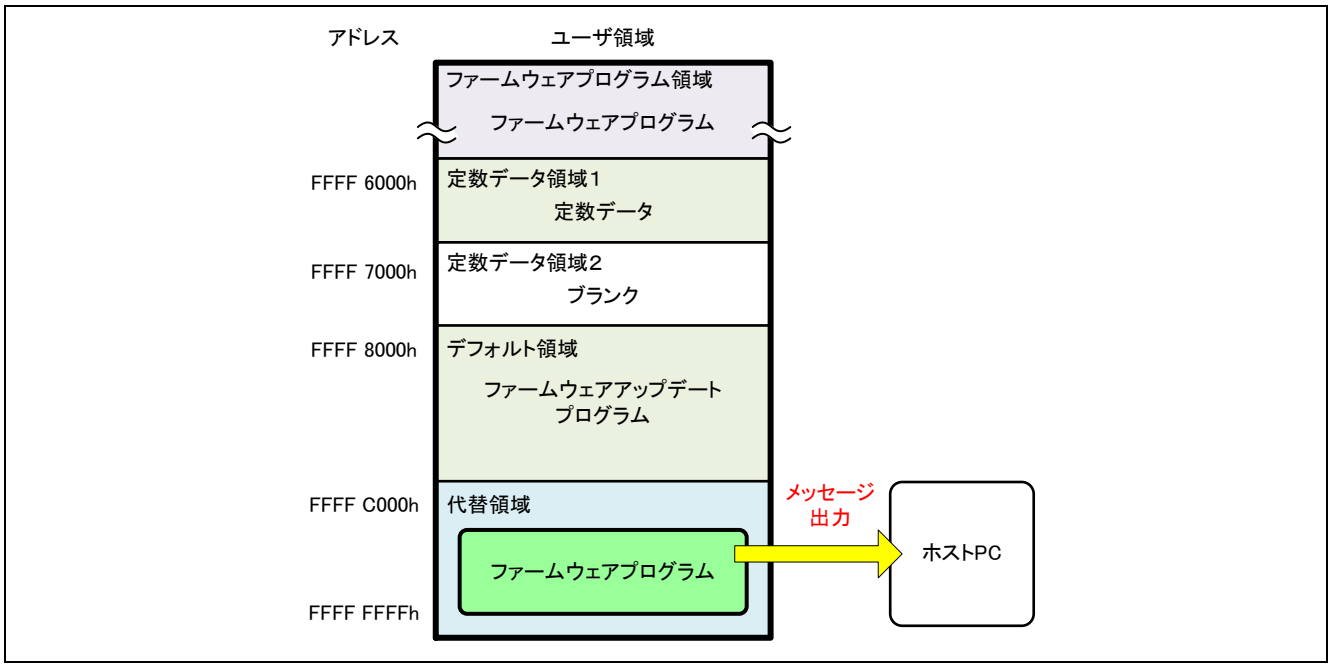


図 5.24　メッセージ出力（RX231 グループの場合）

3. ターミナルソフトウェアからコマンドを送信します。ファームウェアプログラムはスタートアッププログラム保護機能の領域の入れ替えとソフトウェアリセットを内蔵 RAM 上で行うため、内蔵 RAM にフラッシュメモリ操作処理を展開します。

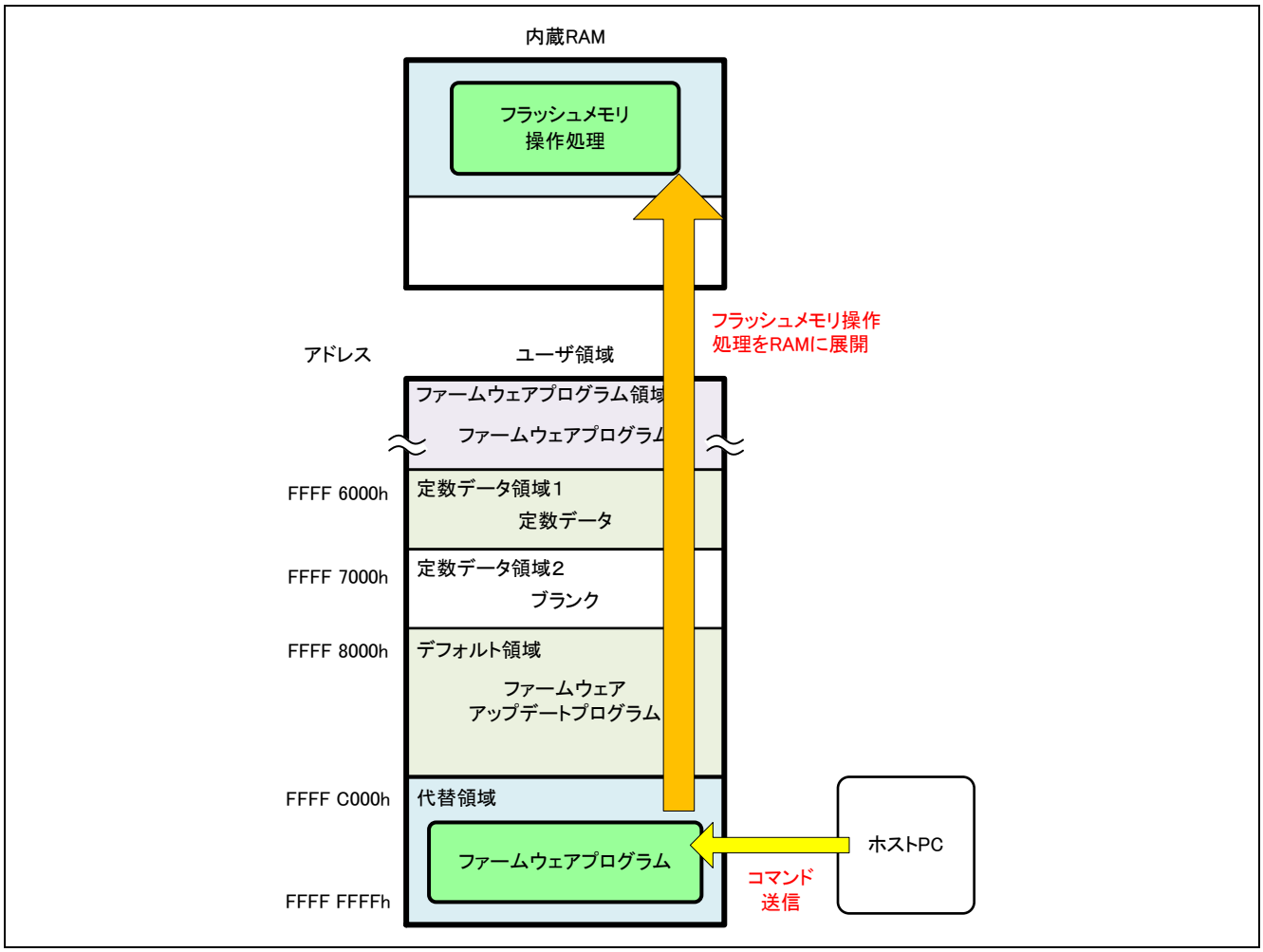


図 5.25　コマンド受信（RX231 グループの場合）

4. ファームウェアプログラムは、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、ソフトウェアリセットを実行します。ファームウェアアップデートプログラムが起動します。

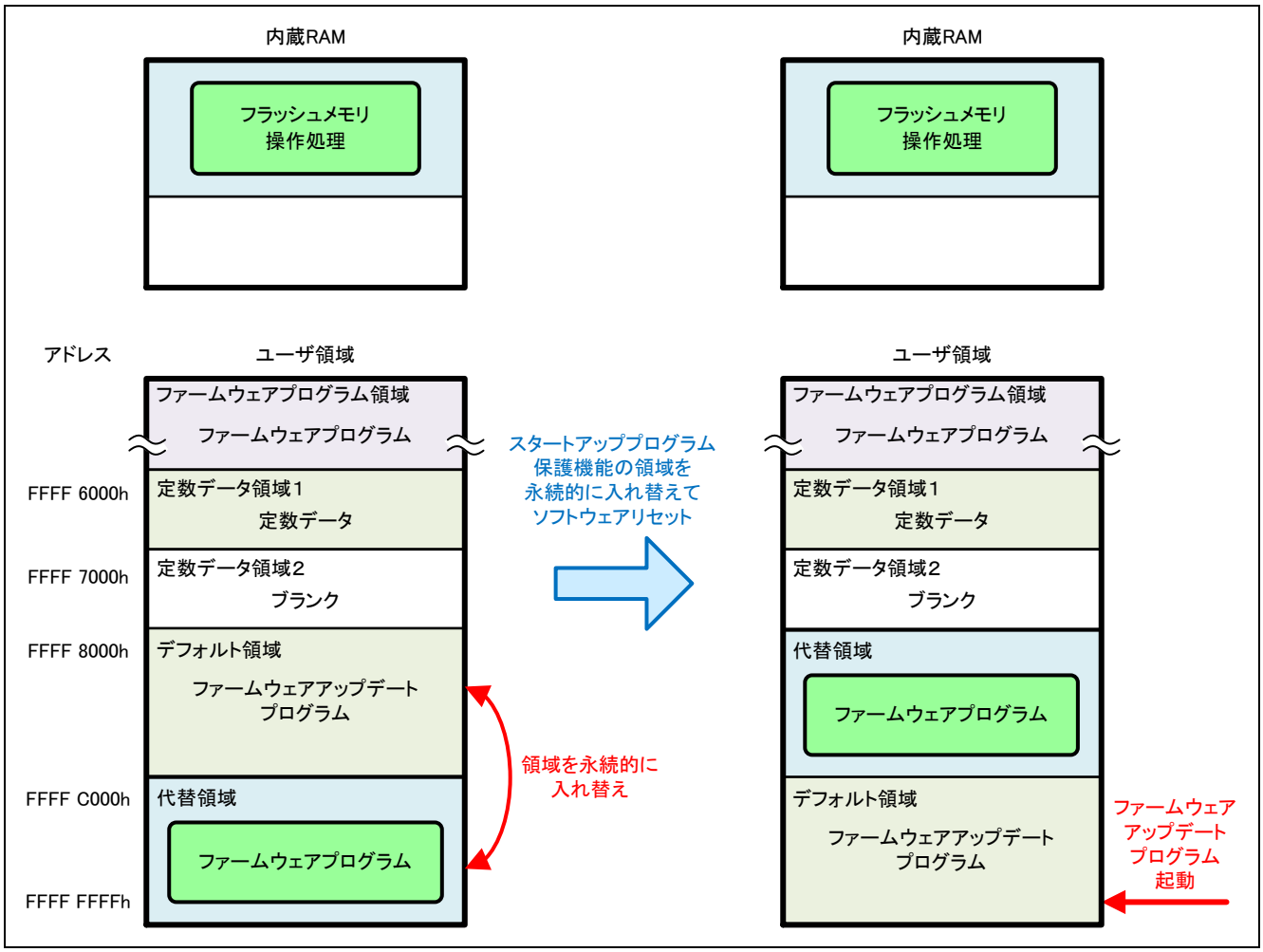


図 5.26　ソフトウェアリセットとファームウェアアップデートプログラムの起動（RX231 グループの場合）

5.2.6　書き込みが失敗した場合の復旧方法

ファームウェアアップデートプログラムを使用してコードフラッシュメモリの書き換えるとき、電源の瞬断などにより書き換えが失敗した場合、デバイスをリセットすることにより、コードフラッシュメモリの書き換えを再度行うことができます。コードフラッシュメモリの書き換えを再度行うことができる理由を以下に示します。

ファームウェアアップデートプログラムは、コードフラッシュメモリの消去および書き込みを行う前に、フラッシュ初期設定レジスタ（FISR）の設定によりスタートアッププログラム保護機能のデフォルト領域と代替領域を一時的に入れ替えます。その後、デフォルト領域以外のコードフラッシュメモリの書き換えを行います。フラッシュ初期設定レジスタ（FISR）の設定はデバイスのリセットにより初期化され、初期化後のデフォルト領域と代替領域の配置はエクストラ領域のスタートアップ領域設定に従います。ファームウェアアップデートプログラムによってコードフラッシュメモリを書き換えるとき、エクストラ領域のスタートアップ領域設定はデフォルト領域から起動する設定になっています。このため、デバイスをリセットすることにより、デフォルト領域に配置されているファームウェアアップデートプログラムが起動し、コードフラッシュメモリの書き換えを再度行うことができます。

5.2.7　更新用ファームウェアアップデートプログラムの作成方法

ファームウェアアップデートプログラムを更新するときに使用する更新用ファームウェアアップデートプログラムを作成するときは、以下に示す項目を変更してください。

(1)　セクション設定を変更

セクション設定でセクションの先頭アドレスを変更します。更新用ファームウェアアップデートプログラムは、動作中のファームウェアアップデートプログラムが使用していない定数データ領域に定数データを格納する必要があります。そのため、更新前のプログラムが定数データ領域 2 を使用している場合、表 5.3 に示す先頭アドレスを設定してください。更新前のプログラムが定数データ領域 1 を使用している場合、表 5.4 に示す先頭アドレスを設定してください。

表 5.3　定数データ領域 1 に定数データを格納する場合のセクションの先頭アドレス

デバイス	セクション		
	FW_UP_VER	C_1	FW_UP_COMPLETE
RX130 グループ	0xFFFF6800	0xFFFF6808	0xFFFF73F0
RX140 グループ RX231 グループ	0xFFFF6000	0xFFFF6008	0xFFFF6FF0

表 5.4　定数データ領域 2 に定数データを格納する場合のセクションの先頭アドレス

デバイス	セクション		
	FW_UP_VER	C_1	FW_UP_COMPLETE
RX130 グループ	0xFFFF7400	0xFFFF7408	0xFFFF7FF0
RX140 グループ RX231 グループ	0xFFFF7000	0xFFFF7008	0xFFFF7FF0

(2)　バージョン情報を変更

ファームウェアアップデートプログラムのバージョン情報を変更します。バージョン情報を変更するには main.c の定数「FW_UP_PROGRAM_VERSION」の設定値を変更します。バージョン情報は 4 桁で 0～9 の 16 進数を設定してください。

以下にバージョン情報の設定例を示します。

- Ver3.05 の場合

```
#define FW_UP_PROGRAM_VERSION (0x0305u)
```

- Ver10.20 の場合

```
#define FW_UP_PROGRAM_VERSION (0x1020u)
```

5.3　ファームウェアアップデートプログラムの概略フローと画面出力

ファームウェアアップデートプログラムはシリアル通信を使用して、ホスト PC のターミナルソフトウェアにメッセージを出力します。また、ターミナルソフトウェアからの入力コマンドに応じて各処理へ分岐します。

5.3.1　メイン処理

ファームウェアアップデートプログラムのメイン処理は、SCI FIT モジュールとフラッシュ FIT モジュールの初期設定を行ったのち、SCI を使用してホスト PC のターミナルソフトウェアにメニュー表示を出力します。その後、ターミナルソフトウェアからの入力キー待ちを行います。入力されたキーに応じて各処理へ分岐します。

(1)　概略フロー

図 5.27 にメイン処理の概略フロー示します。

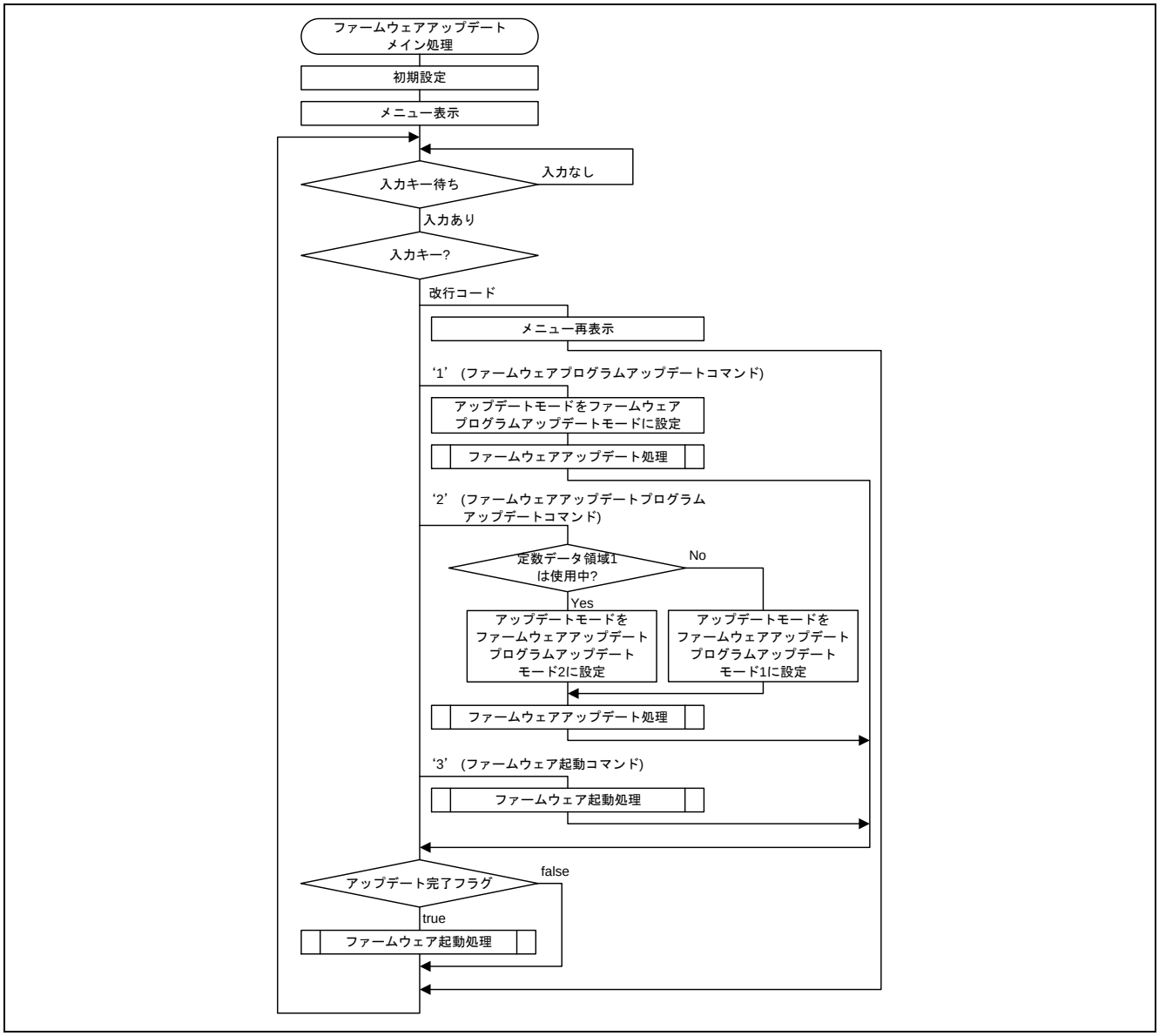


図 5.27　メイン処理の概略フロー

(2)　ターミナルソフトウェア画面出力

ファームウェアアップデートプログラムは起動後、下記のメッセージを出力します。

RX130 グループの場合：

```
RX130 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

図 5.28　メイン処理の画面出力（RX130 グループ）

RX140 グループの場合：

```
RX140 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

図 5.29　メイン処理の画面出力（RX140 グループ）

RX231 グループの場合：

```
RX231 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

図 5.30　メイン処理の画面出力（RX231 グループ）

’1’を送信するとファームウェアプログラムの更新を行います。アップデートモードにファームウェアプログラムアップデートモードが設定されます。アップデートモードの設定後、ファームウェアアップデート処理に分岐します。

‘2’を送信するとファームウェアアップデートプログラムの更新を行います。アップデートモードにファームウェアアップデートプログラムアップデートモード 1 またはファームウェアアップデートプログラムアップデートモード 2 が設定されます。アップデートモードの設定後、ファームウェアアップデート処理に分岐します。

’3’を送信するとファームウェア起動処理に分岐します。

改行コードを送信すると、メニューを再び表示します。

(3)　アップデートモード

ファームウェアアップデートプログラムを使用してプログラムの更新をするときは、表 5.5 に示すアップデートモードを設定してファームウェアアップデート処理に分岐します。

表 5.5　アップデートモードの概要

アップデートモード	概要
ファームウェアプログラム アップデートモード	ファームウェアプログラムを更新するモードです。 ファームウェアプログラム領域とスタートアッププログラム保護機能の代替領域に対してのみプログラム/イレーズを行います。
ファームウェアアップデート プログラムアップデートモード 1	ファームウェアアップデートプログラムを更新するモードです。 定数データ領域 1 とスタートアッププログラム保護機能の代替領域に対してのみプログラム/イレーズを行います。
ファームウェアアップデート プログラムアップデートモード 2	ファームウェアアップデートプログラムを更新するモードです。 定数データ領域 2 とスタートアッププログラム保護機能の代替領域に対してのみプログラム/イレーズを行います。

5.3.2　ファームウェアアップデート処理

メイン処理において、‘1’または‘2’を入力するとファームウェアアップデート処理に移行します。XMODEM/SUM プロトコルを使用したシリアル通信で更新用のプログラムを受信し、コードフラッシュメモリに書き込みます。

(1)　概略フロー

図 5.31 にファームウェアアップデート処理の概略フローを示します。

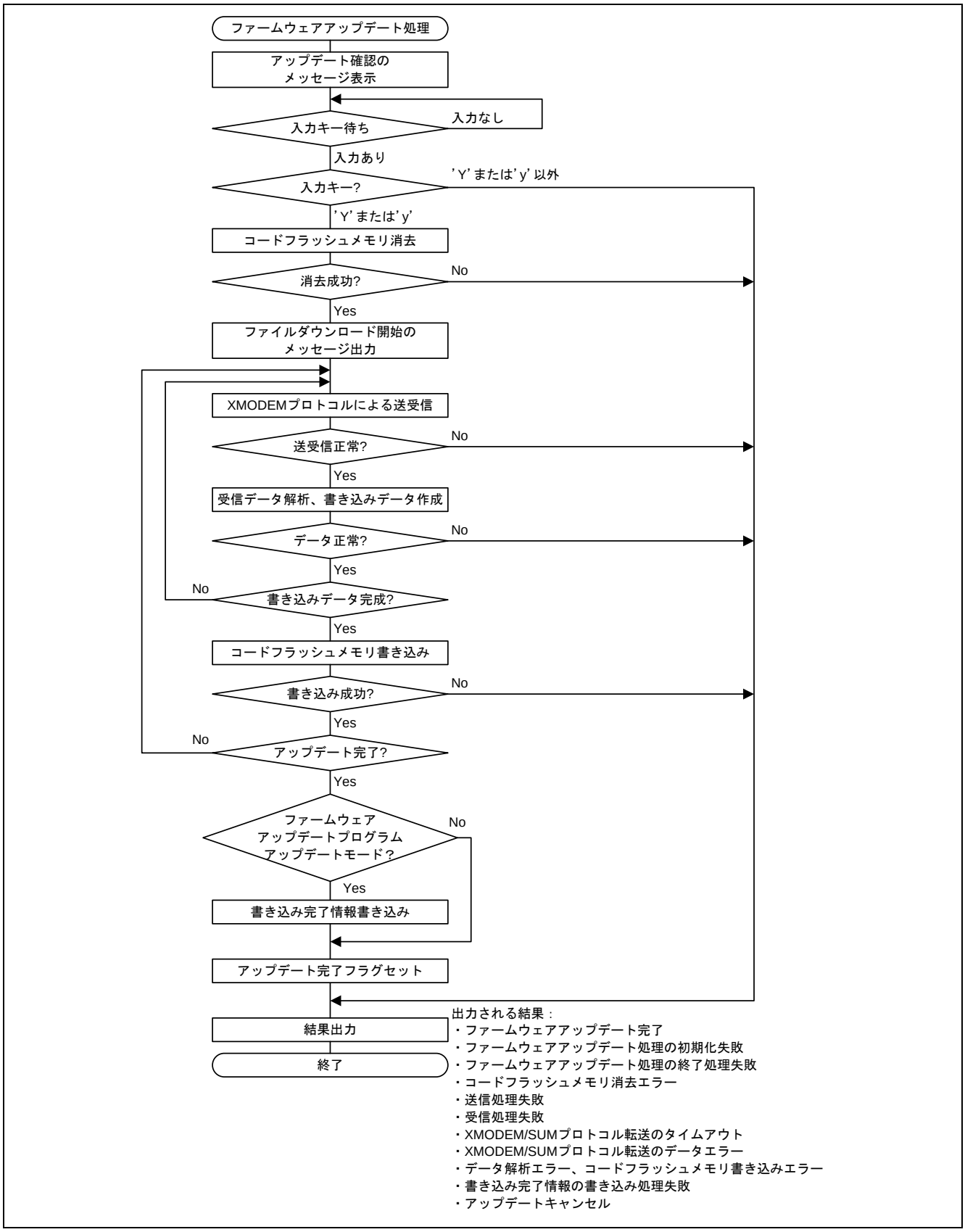


図 5.31　ファームウェアアップデート処理の概略フロー

(2)　ターミナルソフトウェア画面出力

1.　アップデート確認

ファームウェアアップデート処理は、アップデート確認のため図 5.32 のメッセージを出力します。

```
Erase flash memory and write firmware (Y/N)?
```

図 5.32　アップデート確認の画面出力

2.　ファイルダウンロード開始

‘Y’または’y’を送信すると、コードフラッシュメモリを消去したのち、図 5.33 のメッセージを出力してファームウェア受信待ち状態になります。ターミナルソフトウェアから XMODEM/SUM プロトコルを使用して.mot ファイルを送信してください。なお、ターミナルソフトウェアからの XMODEM/SUM プロトコルでのファイル送信については、ターミナルソフトウェアのマニュアルを参照してください。

```
Erasing has been done.
Start Xmodem download...
```

図 5.33　ファイルダウンロード開始の画面出力

3.　ファームウェアアップデート完了

ファームウェアアップデートが完了すると、図 5.34 のメッセージを出力します。

```
Updating firmware has been done.
>
```

図 5.34　ファームウェアアップデート完了の画面出力

4.　エラー出力

ファームウェアアップデート中にエラーが発生した場合、その内容に応じて図 5.35 のメッセージを出力します。

```
Initialize update error.    :ファームウェアアップデート処理の初期化失敗
Finalize update error.      :ファームウェアアップデート処理の終了処理失敗
Erasing error.              :コードフラッシュメモリ消去エラー
Send error.                 :送信処理失敗
Receive error.              :受信処理失敗
Timeout.                    :XMODEM/SUM プロトコル転送のタイムアウト
Data error.                 :XMODEM/SUM プロトコル転送のデータエラー
Block processing error.     :データ解析エラー、コードフラッシュメモリ書き込みエラー
Set write complete information error. :書き込み完了情報の書き込み処理失敗
```

図 5.35　エラー発生時の画面出力

5.　アップデートキャンセル

1.アップデート確認のとき、’Y’または’y’以外のコマンドを送信すると、図 5.36 のメッセージを出力しアップデートをキャンセルします。

```
Command canceled.
>
```

図 5.36　アップデートキャンセルの画面出力

5.3.3　　ファームウェア起動処理

メイン処理において、‘3’を入力するとファームウェア起動処理に移行します。スタートアッププログラム保護機能の領域を入れ替えたのちソフトウェアリセットを実行することにより、プログラムが起動します。

ファームウェアプログラムまたはファームウェアアップデートプログラムを更新した場合、メニューを表示させずに直接ファームウェア起動処理に移行します。

(1)　概略フロー

図 5.37 にファームウェア起動処理の概略フローを示します。

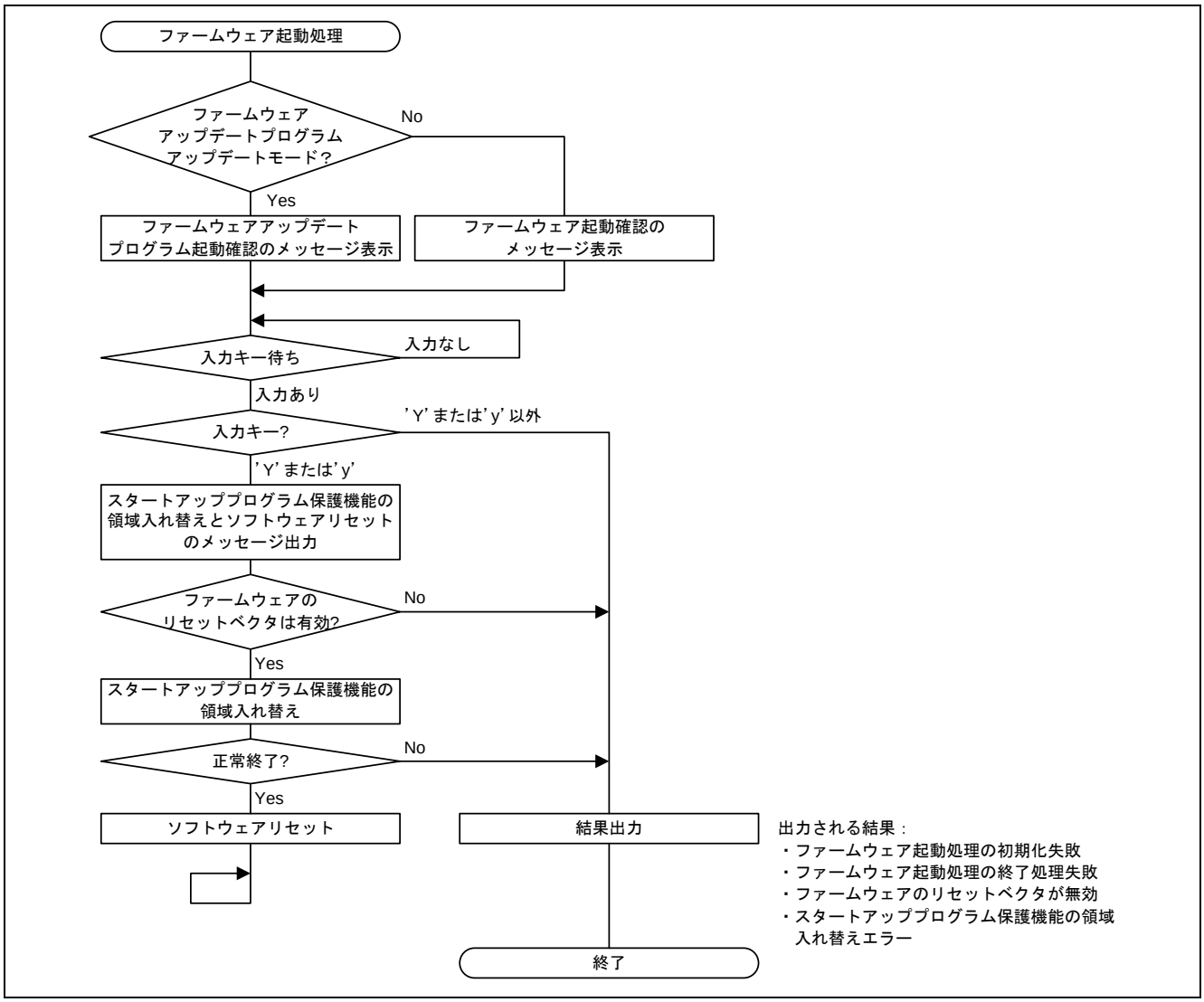


図 5.37　ファームウェア起動処理の概略フロー

(2)　ターミナルソフトウェア画面出力

1.　ファームウェア起動確認

ファームウェア起動処理は、ファームウェア起動確認のため図 5.38 のメッセージを出力します。

```
Execute firmware (Y/N)?
```

図 5.38　ファームウェア起動確認の画面出力

2.　ファームウェアアップデートプログラム起動確認

ファームウェアアップデートプログラム更新後のファームウェア起動処理では、新しいファームウェアアップデートプログラムのバージョン確認のため図 **5.39** のメッセージを出力します。

```
Execute new firmware update program Ver[新しいファームウェアアップデートプログラムのバージョン](Y/N)?
```

図 5.39　新しいファームウェアアップデートプログラム起動確認の画面出力

3.　ファームウェア起動

'Y'または'y'を送信すると、図 5.40 のメッセージを出力して、スタートアッププログラム保護機能の領域を入れ替えたのちソフトウェアリセットを実行します。

```
Switch Start-Up area and do software reset.
```

図 5.40　スタートアッププログラム保護機能の領域入れ替えとソフトウェアリセットの画面出力

4.　エラー出力

ファームウェア起動中にエラーが発生した場合、その内容に応じて図 5.41 のメッセージを出力します。

```
Initialize update error.                    :ファームウェア起動処理の初期化失敗
Finalize update error.                      :ファームウェア起動処理の終了処理失敗
Reset vector of the firmware is invalid.    :ファームウェアのリセットベクタが無効
Switching Start-Up area error.              :スタートアッププログラム保護機能の領域入れ替えエラー
```

図 5.41　エラー発生時の画面出力

5.　ファームウェア起動キャンセル

1.ファームウェア起動確認のとき、'Y'または'y'以外のコマンドを送信すると、図 5.42 のメッセージを出力しアップデートをキャンセルします。

```
Command canceled.
>
```

図 5.42　ファームウェア起動キャンセルの画面出力

5.4　ファームウェアプログラムの概略フローと画面出力

ファームウェアプログラムはシリアル通信を使用して、ホスト PC のターミナルソフトウェアにメッセージを出力します。また、ターミナルソフトウェアから改行コードを送信すると、スタートアッププログラム保護機能の領域を入れ替えたのちソフトウェアリセットを実行して、ファームウェアアップデートプログラムを起動します。

(1)　概略フロー

図 5.43 にファームウェアプログラムの概略フローを示します。

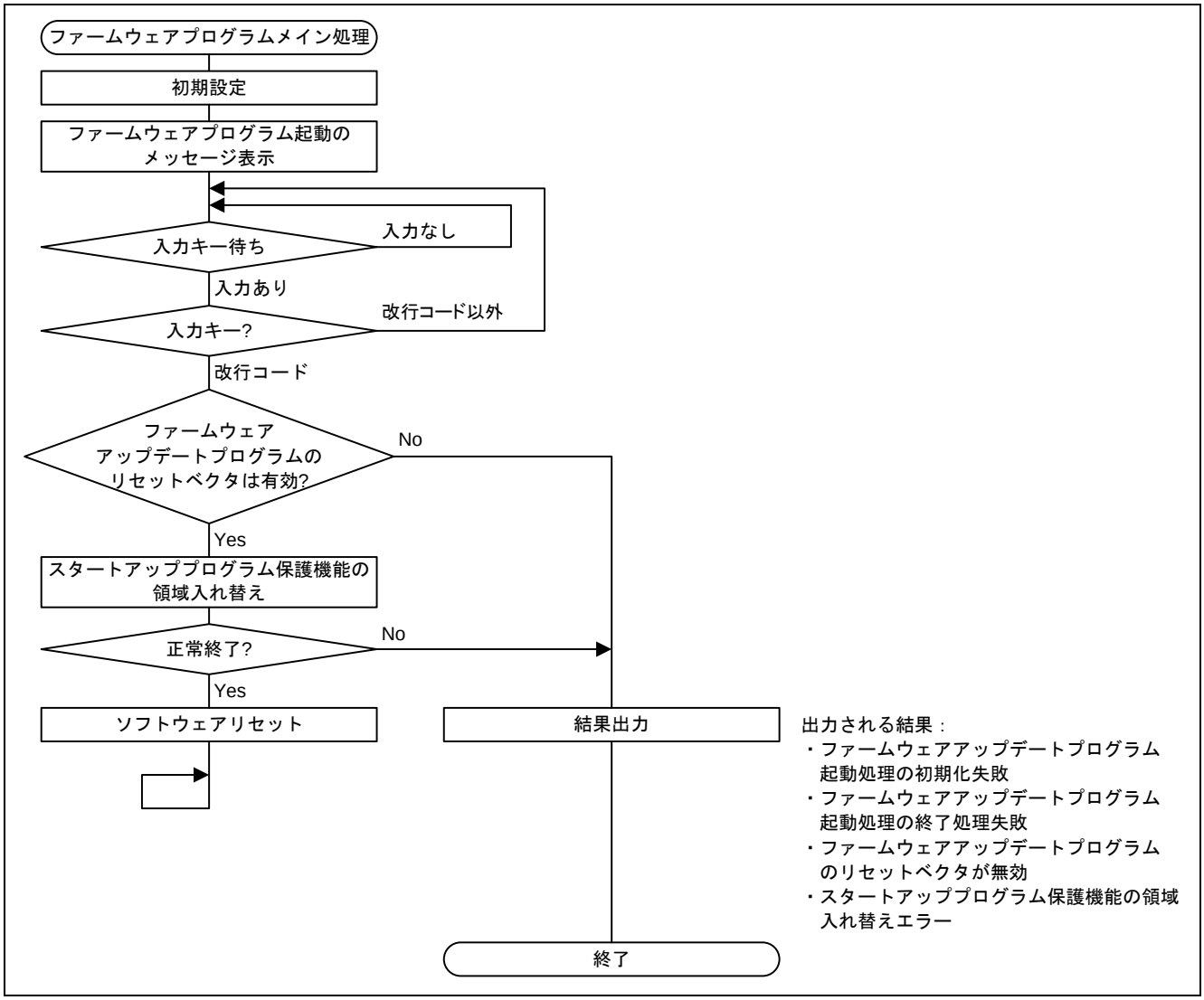


図 5.43　ファームウェアプログラムの概略フロー

(2)　ターミナルソフトウェア画面出力

1.　ファームウェアプログラム起動

　ファームウェアプログラムは、ファームウェアプログラム起動の通知のため図 5.44 のメッセージを出力します。

```
This program is the sample firmware.
Push Enter key to execute firmware update.
>
```

図 5.44　ファームウェアプログラム起動の画面出力

　改行コードを送信すると、スタートアッププログラム保護機能の領域を入れ替えたのちソフトウェアリセットを実行して、ファームウェアアップデートプログラムを起動します。

2.　エラー出力

　ファームウェアアップデートプログラム起動中にエラーが発生した場合、その内容に応じて図 5.45 のメッセージを出力します。

```
Initialize update error.    :ファームウェアアップデートプログラム起動処理の初期化失敗
Finalize update error.      :ファームウェアアップデートプログラム起動処理の終了処理失敗
Reset vector of the firmware is invalid.
                            :ファームウェアアップデートプログラムのリセットベクタが無効
Switching Start-Up area error.
                            :スタートアッププログラム保護機能の領域入れ替えエラー
```

図 5.45　エラー発生時の画面出力

5.5　ファームウェアアップデートプログラム詳細

5.5.1　ファイル構成

表 5.6 にファームウェアアップデートプログラムで使用するファイルを、表 5.7 にファームウェアアップデートプログラムで使用する標準インクルードファイルを示します。なお、FIT モジュールおよび統合開発環境で自動生成されるファイルは除きます。

表 5.6　ファームウェアアップデートプログラムで使用するファイル

ファイル名	概要
main.c	メインソースファイル
main.h	メインインタフェースファイル
r_xmodem.c	XMODEM ソースファイル
r_xmodem_if.h	XMODEM インタフェースファイル
r_fw_up_rx.c	ファームウェアアップデートソースファイル
r_fw_up_rx_if.h	ファームウェアアップデートインタフェースファイル
r_fw_up_rx_private.h	ファームウェアアップデートヘッダファイル
r_fw_up_buf.c	ファームウェアデータ用バッファ処理ソースファイル
r_fw_up_buf.h	ファームウェアデータ用バッファ処理ヘッダファイル

表 5.7　ファームウェアアップデートプログラムで使用する標準インクルードファイル

ファイル名	概要
stdbool.h	論理型、および論理値に関するマクロを定義します。
stdint.h	指定した幅の整数型を宣言してマクロを定義します。
stdlib.h	記憶領域管理などの C プログラムで標準的処理を行うライブラリです。
string.h	文字列の比較、複写などを行うライブラリです。

5.5.2　定数一覧

表 5.8～表 5.13 にファームウェアアップデートプログラムで使用する定数を示します。

表 5.8　ファームウェアアップデートプログラムで使用する定数（main.c）

定数名	設定値	内容
FW_UP_PROGRAM_VERSION	(0x0100u)	バージョン情報
DUMMY_DATA	(0xAA55AA55AA55AA55u)	書き込み完了情報格納領域の前に格納するダミーデータ
MASK_NUM	(0x0Fu)	下位 4 ビット取得用マスク
ASCII_CODE_NUM	(0x30u)	「0」の文字コード
ASCII_CODE_POINT	(0x2Eu)	「.」の文字コード
DELAY_NUM	(1u)	R_BSP_SoftwareDelay 関数に引数として渡す遅延時間
TIMEOUT_NUM	(10000u)	10 秒のタイムアウト時間判定用カウント数
RECV_BYTE_SIZE	(1u)	受信用 1 バイトのサイズ
SEND_BYTE_SIZE	(1u)	送信用 1 バイトのサイズ
COMMAND_UPDATE_FIRM	(’1’)	入力コマンド用文字コード（ファームウェアプログラムアップデートコマンド）
COMMAND_UPDATE_FIRM_UPDATE	(’2’)	入力コマンド用文字コード（ファームウェアアップデートプログラムアップデートコマンド）
COMMAND_EXEC_PROGRAM	(’3’)	入力コマンド用文字コード（ファームウェア起動コマンド）
COMMAND_YES_UPPER	(’Y’)	入力コマンド用文字コード（“Y”）
COMMAND_YES_LOWER	(’y’)	入力コマンド用文字コード（“y”）
COMMAND_CR	(’¥r’)	入力コマンド用文字コード（改行コード）
STRING_MAX_SIZE	RX130 グループ：(SCI_CFG_CH1_TX_BUFSIZ) RX140 グループ：(SCI_CFG_CH1_TX_BUFSIZ) RX231 グループ：(SCI_CFG_CH5_TX_BUFSIZ)	出力する文字列の最大サイズ

表 5.9　ファームウェアアップデートプログラムで使用する定数（r_xmodem.c）

定数名	設定値	内容
XM_SOH	(0x01u)	XMODEM コントロールコード（SOH）
XM_EOT	(0x04u)	XMODEM コントロールコード（EOT）
XM_ACK	(0x06u)	XMODEM コントロールコード（ACK）
XM_NAK	(0x15u)	XMODEM コントロールコード（NAK）
XM_CAN	(0x18u)	XMODEM コントロールコード（CAN）
XM_HEADER_SIZE	(1+1+1)	XMODEM データブロックのヘッダサイズ（バイト数）
XM_DATA_SIZE	(128u)	XMODEM データブロックのデータサイズ（バイト数）
XM_SUM_SIZE	(1u)	XMODEM データブロックのチェックサムサイズ（バイト数）
XM_BLOCK_SIZE	(XM_HEADER_SIZE + XM_DATA_SIZE + XM_SUM_SIZE)	XMODEM データブロックサイズ（バイト数）
XM_RETRY_COUNT	(10u)	XMODEM データ転送タイムアウト時のリトライ回数
UINT8T_0	(0u)	uint8_t 型の 0
UINT8T_1	(1u)	uint8_t 型の 1
COMPLEMENT_CHECK	(0xFFu)	ブロック番号の補数を確認するための数値

表 5.10　ファームウェアアップデートプログラムで使用する定数（r_fw_up_rx_private.h）（1/3）

定数名	設定値	内容
FW_UP_BINARY_BUF_SIZE	(256u)	コードフラッシュメモリ書き込み用データのバッファサイズ
FW_UP_BINARY_BUF_NUM	(2u)	コードフラッシュメモリ書き込み用データのバッファ数
FW_UP_BUF_NUM	(60u)	解析したモトローラ S レコードフォーマットデータの内容を格納する配列の数
FW_UP_FIRM_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_127) RX140 グループ： (FLASH_CF_BLOCK_127) RX231 グループ： (FLASH_CF_BLOCK_255)	ファームウェアプログラムを書き込む領域の先頭アドレス
FW_UP_FIRM_EN_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_37 – 1) RX140 グループ： (FLASH_CF_BLOCK_19 – 1) RX231 グループ： (FLASH_CF_BLOCK_19 – 1)	ファームウェアプログラムを書き込む領域の最終アドレス
FW_UP_CONST_1_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_37) RX140 グループ： (FLASH_CF_BLOCK_19) RX231 グループ： (FLASH_CF_BLOCK_19)	定数データ領域 1 の開始アドレス
FW_UP_CONST_1_FIN_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_34 - 256) RX140 グループ： (FLASH_CF_BLOCK_17 - 256) RX231 グループ： (FLASH_CF_BLOCK_17 - 256)	定数データ領域 1 の最終書き込み処理に使用する先頭アドレス
FW_UP_COMPLETE_1_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_34 - 8) RX140 グループ： (FLASH_CF_BLOCK_17 - 8) RX231 グループ： (FLASH_CF_BLOCK_17 - 8)	書き込み完了情報格納領域 1 の先頭アドレス
FW_UP_CONST_1_EN_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_34 - 1) RX140 グループ： (FLASH_CF_BLOCK_17 - 1) RX231 グループ： (FLASH_CF_BLOCK_17 - 1)	定数データ領域 1 の最終アドレス

表 5.11　ファームウェアアップデートプログラムで使用する定数（r_fw_up_rx_private.h）（2/3）

定数名	設定値	内容
FW_UP_CONST_2_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_34) RX140 グループ： (FLASH_CF_BLOCK_17) RX231 グループ： (FLASH_CF_BLOCK_17)	定数データ領域 2 の先頭アドレス
FW_UP_CONST_2_FIN_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_31 - 256) RX140 グループ： (FLASH_CF_BLOCK_15 - 256) RX231 グループ： (FLASH_CF_BLOCK_15 - 256)	定数データ領域 2 の最終書き込み処理に使用する先頭アドレス
FW_UP_COMPLETE_2_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_31 - 8) RX140 グループ： (FLASH_CF_BLOCK_15 - 8) RX231 グループ： (FLASH_CF_BLOCK_15 - 8)	書き込み完了情報格納領域 2 の開始アドレス
FW_UP_CONST_2_EN_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_31 - 1) RX140 グループ： (FLASH_CF_BLOCK_15 - 1) RX231 グループ： (FLASH_CF_BLOCK_15 - 1)	定数データ領域 2 の最終アドレス
FW_UP_STUP_ST_ADDRESS	RX130 グループ： (FLASH_CF_BLOCK_15) RX140 グループ： (FLASH_CF_BLOCK_7) RX231 グループ： (FLASH_CF_BLOCK_7)	スタートアッププログラム保護機能のデフォルト領域の先頭アドレス
FW_UP_STUP_EN_ADDRESS	(FLASH_CF_BLOCK_END)	スタートアッププログラム保護機能のデフォルト領域の最終アドレス
FW_UP_FIRM_BLOCK_NUM	RX130 グループ：(90u) RX140 グループ：(108u) RX231 グループ：(236u)	ファームウェアプログラムを書き込む領域のブロック数
FW_UP_CONST_BLOCK_NUM	RX130 グループ：(3u) RX140 グループ：(2u) RX231 グループ：(2u)	定数データ領域のブロック数
FW_UP_STUP_BLOCK_NUM	RX130 グループ：(16u) RX140 グループ：(8u) RX231 グループ：(8u)	スタートアッププログラム保護機能のデフォルト領域のブロック数

表 5.12　ファームウェアアップデートプログラムで使用する定数（r_fw_up_rx_private.h）（3/3）

定数名	設定値	内容
FW_UP_FIRM_RESETVECT	RX130 グループ： (FLASH_CF_BLOCK_15 - 4) RX140 グループ： (FLASH_CF_BLOCK_7 - 4) RX231 グループ： (FLASH_CF_BLOCK_7 - 4)	ファームウェアプログラムのリセットベクタのアドレス
FW_UP_BLANK_VALUE	(0xFFFFFFFFu)	コードフラッシュメモリがブランク時の読み出し値
PRCR_KEY	(0xA500u)	PRCR レジスタのキーコード
SET_PRC1	(0x0002u)	PRCR レジスタの PRC1 ビットを設定するための値
MCU_RESET	(0xA501u)	SWRR レジスタに設定して MCU をリセットさせる値

表 5.13　ファームウェアアップデートプログラムで使用する定数（r_fw_up_buf.h）

定数名	設定値	内容
MOT_S_CHECK_SUM_FIELD	(0x02u)	モトローラ S レコードフォーマットのチェックサムフィールドの文字数
ADDRESS_LENGTH_S1	(0x04u)	モトローラ S レコードフォーマットのアドレスフィールドの文字数（S1 タイプ）
ADDRESS_LENGTH_S2	(0x06u)	モトローラ S レコードフォーマットのアドレスフィールドの文字数（S2 タイプ）
ADDRESS_LENGTH_S3	(0x08u)	モトローラ S レコードフォーマットのアドレスフィールドの文字数（S3 タイプ）
BUF_LOCK	(1u)	指定したモトローラ S レコードフォーマットのバッファはロックされている。
BUF_UNLOCK	(0u)	指定したモトローラ S レコードフォーマットのバッファは開放されている。
MOT_RECORD_S0	(0u)	モトローラ S レコードフォーマットのレコードタイプ（S0 タイプ）
MOT_RECORD_S1	(1u)	モトローラ S レコードフォーマットのレコードタイプ（S1 タイプ）
MOT_RECORD_S2	(2u)	モトローラ S レコードフォーマットのレコードタイプ（S2 タイプ）
MOT_RECORD_S3	(3u)	モトローラ S レコードフォーマットのレコードタイプ（S3 タイプ）
MOT_RECORD_S7	(7u)	モトローラ S レコードフォーマットのレコードタイプ（S7 タイプ）
MOT_RECORD_S8	(8u)	モトローラ S レコードフォーマットのレコードタイプ（S8 タイプ）
MOT_RECORD_S9	(9u)	モトローラ S レコードフォーマットのレコードタイプ（S9 タイプ）
MASK_LOWER_BYTE	(0x000000FFu)	下位 1 バイト取得用マスク
ASCII_CODE_0	(0x30u)	「0」の文字コード
ASCII_CODE_9	(0x39u)	「9」の文字コード
ASCII_CODE_UPPER_A	(0x41u)	「A」の文字コード
ASCII_CODE_UPPER_F	(0x46u)	「F」の文字コード
ASCII_CODE_LOWER_A	(0x61u)	「a」の文字コード
ASCII_CODE_LOWER_F	(0x66u)	「f」の文字コード
CONVERT_HEX_NUM	(0x0Fu)	0～9 の文字コードを 16 進数に変換するための値
CONVERT_HEX_UPPER_CHAR	(0x37u)	A～F の文字コードを 16 進数に変換するための値
CONVERT_HEX_LOWER_CHAR	(0x57u)	a～f の文字コードを 16 進数に変換するための値

5.5.3　　型定義一覧

図 5.46～図 5.49 にファームウェアアップデートプログラムで使用する型定義を示します。

```
typedef enum e_xmodem_proc_stage
{
    XMODEM_PROC_END = 0,
    XMODEM_PROCESSING,
    XMODEM_SOH_RECEIVED
} e_xmodem_proc_stage_t;

typedef struct st_xmodem_states
{
    uint8_t               retry_counter;
    uint8_t               expected_block_number;
    uint8_t               recv_buf_index;
    uint8_t               can_counter;
    uint8_t             * p_recv_buf;
    e_xmodem_proc_stage_t   proc_stage;
    xm_recv_func_t         recv_func;
    xm_send_func_t         send_func;
    xm_exec_func_t         exec_func;
} st_xmodem_states_t;
```

図 5.46　ファームウェアアップデートプログラムで使用する型定義（r_xmodem.c）

```
typedef enum e_xmodem_err
{
    XMODEM_SUCCESS,
    XMODEM_SEND_ERR,
    XMODEM_RECV_ERR,
    XMODEM_TIMEOUT,
    XMODEM_PROC_BLOCK_ERR,
    XMODEM_RECV_CAN,
    XMODEM_DATA_ERR
} e_xmodem_err_t;

typedef e_xmodem_err_t (*xm_recv_func_t)(uint8_t* p_arg);
typedef e_xmodem_err_t (*xm_send_func_t)(uint8_t arg);
typedef e_xmodem_err_t (*xm_exec_func_t)(const uint8_t* p_buf, uint16_t size);
```

図 5.47　ファームウェアアップデートプログラムで使用する型定義（r_xmodem_if.h）

```
typedef enum e_update_mode_t
{
    UPDATE_FW,
    UPDATE_FW_UP_1,
    UPDATE_FW_UP_2,
} update_mode_t;

typedef enum e_fw_up_return_t
{
    FW_UP_SUCCESS,
    FW_UP_ERR_OPENED,
    FW_UP_ERR_NOT_OPEN,
    FW_UP_ERR_NULL_PTR,
    FW_UP_ERR_INVALID_RECORD,
    FW_UP_ERR_BUF_FULL,
    FW_UP_ERR_BUF_EMPTY,
    FW_UP_ERR_INITIALIZE,
    FW_UP_ERR_ERASE,
    FW_UP_ERR_WRITE,
    FW_UP_ERR_VERIFY,
    FW_UP_ERR_SWITCH_AREA,
    FW_UP_ERR_INVALID_ADDRESS,
    FW_UP_ERR_INVALID_RESETVECT,
    FW_UP_ERR_INTERNAL
} fw_up_return_t;

typedef struct st_fw_up_fl_data_t
{
    uint32_t src_addr;
    uint32_t dst_addr;
    uint32_t len;
    uint16_t count;
} fw_up_fl_data_t;
```

図 5.48　ファームウェアアップデートプログラムで使用する型定義（r_fw_up_rx_if.h）

```
typedef enum fw_up_mot_s_cnt_t
{
    STATE_MOT_S_RECORD_MARK = 0,
    STATE_MOT_S_RECORD_TYPE,
    STATE_MOT_S_LENGTH_1,
    STATE_MOT_S_LENGTH_2,
    STATE_MOT_S_ADDRESS,
    STATE_MOT_S_DATA,
    STATE_MOT_S_CHKSUM_1,
    STATE_MOT_S_CHKSUM_2
} fw_up_mot_s_cnt_t;

typedef struct MotSBufS
{
    uint8_t          addr_length;
    uint8_t          data_length;
    uint8_t        * p_address;
    uint8_t        * p_data;
    uint8_t          type;
    uint8_t          act;
    struct MotSBufS * p_next;
} fw_up_mot_s_buf_t;

typedef struct WriteDataS
{
    uint32_t           addr;
    uint32_t           len;
    uint8_t            data[FW_UP_BINARY_BUF_SIZE];
    struct WriteDataS * p_next;
    struct WriteDataS * p_prev;
} fw_up_write_data_t;
```

図 5.49　ファームウェアアップデートプログラムで使用する型定義（r_fw_up_buf.h）

5.5.4　変数一覧

表 5.14～表 5.17 にファームウェアアップデートプログラムで使用する static 型変数を、表 5.18 および表 5.19 にファームウェアアップデートプログラムで使用する const 型変数を示します。

表 5.14　ファームウェアアップデートプログラムで使用する static 型変数（main.c）

型	変数名	内容	使用関数
static uint8_t	s_update_mode	アップデートのモード	main block_proc_xm update_firmware exec_firmware set_write_complete_information
static uint8_t	s_update_complete_flag	アップデート完了フラグ	main update_firmware exec_firmware
static sci_hdl	s_sci_handle	SCI モジュール制御ハンドル	main send_byte_xm recv_byte_xm update_firmware exec_firmware send_string_sci
static volatile bool	s_sci_send_end_flag	SCI 送信完了判定用フラグ	sci_callback send_string_sci
static volatile int32_t	s_timeout_count	タイムアウト判定用カウンタ	recv_byte_xm

表 5.15　ファームウェアアップデートプログラムで使用する static 型変数（r_xmodem.c）

型	変数名	内容	使用関数
static uint8_t	s_recv_buf[XM_BLOCK_SIZE]	XMODEM 受信データ用バッファ	exec_xmodem

表 5.16　ファームウェアアップデートプログラムで使用する static 型変数（r_fw_up_rx.c）

型	変数名	内容	使用関数
static bool	s_is_opened	ファームウェアアップデート初期設定完了フラグ	fw_up_open fw_up_close fw_up_put_data fw_up_get_data erase_firmware write_firmware switch_start_up_and_reset

表 5.17　ファームウェアアップデートプログラムで使用する static 型変数（r_fw_up_buf.c）

型	変数名	内容	使用関数
static fw_up_mot_s_buf_t	*sp_app_put_mot_s_buf	モトローラ S フォーマット解析処理で現在使用しているモトローラ S レコードデータバッファへのポインタ	fw_up_buf_init fw_up_put_mot_s
static fw_up_mot_s_buf_t	*sp_app_get_mot_s_buf	コードフラッシュメモリ書き込み用データ作成処理で現在使用しているモトローラ S レコードデータバッファへのポインタ	fw_up_buf_init fw_up_get_binary
static fw_up_mot_s_buf_t	s_mot_s_buf[FW_UP_BUF_NUM]	モトローラ S レコードフォーマットデータの内容を格納するバッファ	fw_up_buf_init fw_up_memory_init
static fw_up_write_data_t	*sp_app_write_buf	現在使用しているコードフラッシュメモリ書き込み用データバッファへのポインタ	fw_up_buf_init fw_up_get_binary
static fw_up_write_data_t	s_write_buf[FW_UP_BINARY_BUF_NUM]	コードフラッシュメモリ書き込み用データを格納するバッファ	fw_up_buf_init
static fw_up_mot_s_cnt_t	s_mot_s_data_state	モトローラ S レコードフォーマットデータの解析状態	fw_up_buf_init fw_up_put_mot_s
static uint32_t	s_write_current_address	現在のコードフラッシュメモリ書き込み先アドレス	fw_up_buf_init fw_up_get_binary
static bool	s_detect_terminal_flag	終端レコード検出フラグ	fw_up_buf_init fw_up_put_mot_s fw_up_get_binary

表 5.18　ファームウェアアップデートプログラムで使用する const 型変数（main.c）（1/2）

型	変数名	内容	使用関数
const uint32_t	g_program_version	ファームウェアアップデートプログラムのバージョン	show_menu_start_up
const uint64_t	g_dummy_data	書き込み完了情報格納領域の前に格納するダミーデータ	-
static const uint8_t	s_string_menu0[]	RX130 グループ： “RX130 firmware update using Start-Up Program Protection menu ver” RX140 グループ： “RX140 firmware update using Start-Up Program Protection menu ver” RX231 グループ： “RX231 firmware update using Start-Up Program Protection menu ver”	show_menu_start_up
static const uint8_t	s_string_menu1[]	“1…Update firmware program¥r¥n”	show_menu_start_up
static const uint8_t	s_string_menu2[]	"2...Update firmware update program¥r¥n"	show_menu_start_up
static const uint8_t	s_string_menu3[]	“3…Execute firmware¥r¥n”	show_menu_start_up
static const uint8_t	s_string_input[]	“> ”	show_menu_start_up update_firmware exec_firmware
static const uint8_t	s_string_crlf[]	“¥r¥n”	main update_firmware exec_firmware
static const uint8_t	s_string_update[]	“Erase flash memory and write firmware (Y/N)?”	update_firmware
static const uint8_t	s_string_erase_success[]	“Erasing has been done.¥r¥n”	update_firmware
static const uint8_t	s_string_download[]	“Start Xmodem download…¥r¥n”	update_firmware
static const uint8_t	s_string_finish_xmodem[]	“Updating firmware has been done.¥r¥n”	update_firmware
static const uint8_t	s_string_exec_firm[]	“Execute firmware (Y/N)?”	exec_firmware
static const uint8_t	s_string_reset[]	“Switch Start-Up area and do software reset.¥r¥n”	exec_firmware
static const_uin8_t	s_string_exec_firm_update[]	"Execute new firmware update program Ver"	exec_firmware
static const uint8_t	s_string_y_n[]	" (Y/N)?"	exec_firmware
static const uint8_t	s_string_cancel[]	“Command canceled.¥r¥n”	update_firmware exec_firmware

表 5.19　ファームウェアアップデートプログラムで使用する const 型変数（main.c）（2/2）

型	変数名	内容	使用関数
static const uint8_t	s_string_flash_err[]	“Flash module error.¥r¥n”	main
static const uint8_t	s_string_erase_err[]	“Erasing error.¥r¥n”	update_firmware
static const uint8_t	s_string_set_info_err[]	"Set write complete information error.¥r¥n"	update_firmware
static const uint8_t	s_string_send_err[]	“Send error.¥r¥n”	update_firmware
static const uint8_t	s_string_recv_err[]	“Receive error.¥r¥n”	update_firmware
static const uint8_t	s_string_timeout[]	“Timeout.¥r¥n”	update_firmware
static const uint8_t	s_string_block_err[]	“Block processing error.¥r¥n”	update_firmware
static const uint8_t	s_string_data_err[]	“Data error.¥r¥n”	update_firmware
static const uint8_t	s_string_fin_update_err[]	“Finalize update error.¥r¥n”	update_firmware exec_firmware
static const uint8_t	s_string_init_update_err[]	“Initialize update error.¥r¥n”	update_firmware exec_firmware
static const uint8_t	s_string_resetvect_err[]	“Reset vector of the firmware is invalid.¥r¥n”	exec_firmware
static const uint8_t	s_string_switch_err[]	“Switching Start-Up area error.¥r¥n”	exec_firmware

5.5.5　関数一覧

表 5.20 にファームウェアアップデートプログラムの関数を、表 5.21 にファームウェアアップデートプログラムで使用する FIT モジュールの関数を、表 5.22～表 5.24 にファームウェアアップデートプログラムで使用する e2 studio のスマートコンフィグレータにより生成される関数を示します。

表 5.20　ファームウェアアップデートプログラムの関数

関数名	概要	記載ファイル
main	メイン処理	main.c
show_menu_start_up	メニュー表示	main.c
sci_callback	SCI FIT モジュールのコールバック関数、SCI 送信完了の確認	main.c
send_byte_xm	XMODEM プロトコル用コールバック関数、1 バイトのデータを送信	main.c
recv_byte_xm	XMODEM プロトコル用コールバック関数、1 バイトのデータを受信	main.c
block_proc_xm	XMODEM プロトコル用コールバック関数、1 データブロックのデータ処理	main.c
update_firmware	ファームウェアアップデート処理	main.c
exec_firmware	ファームウェア起動処理	main.c
send_string_sci	文字列送信処理	main.c
set_write_complete_information	書き込み完了情報格納領域にバージョン情報を書き込む処理	main.c
show_version	バージョン表示	main.c
exec_xmodem	XMODEM プロトコル処理	r_xmodem.c
xmodem_recv_soh	XMODEM プロトコルのデータブロックのヘッダ受信	r_xmodem.c
xmodem_check_eot	XMODEM プロトコルのデータブロックのヘッダチェック	r_xmodem.c
xmodem_recv_block	XMODEM プロトコルの 1 データブロック受信	r_xmodem.c
xmodem_analyze_block	XMODEM プロトコルのデータブロック解析	r_xmodem.c
xmodem_proc_data	XMODEM プロトコルの 1 データブロックのデータ処理	r_xmodem.c
xmodem_send_response	XMODEM プロトコルの応答処理	r_xmodem.c
fw_up_open_flash	フラッシュ FIT モジュールの初期化	r_fw_up_rx.c
fw_up_open	ファームウェアアップデートの初期化	r_fw_up_rx.c
fw_up_close	ファームウェアアップデートの終了処理	r_fw_up_rx.c
copy_update_ramprog	RAM プログラムのコピー	r_fw_up_rx.c
analyze_and_write_data	受信データ解析とコードフラッシュメモリ書き込み処理	r_fw_up_rx.c
fw_up_put_data	受信データ解析	r_fw_up_rx.c
fw_up_get_data	コードフラッシュメモリ書き込みデータ取得	r_fw_up_rx.c
erase_firmware	コードフラッシュメモリ消去	r_fw_up_rx.c
write_firmware	コードフラッシュメモリ書き込み	r_fw_up_rx.c
switch_start_up_and_reset	スタートアッププログラム保護機能の領域入れ替えとソフトウェアリセット実行	r_fw_up_rx.c
fw_up_buf_init	ファームウェアアップデートで使用するバッファの初期化	r_fw_up_buf.c
fw_up_memory_init	バッファへのポインタの初期化	r_fw_up_buf.c
fw_up_put_mot_s	モトローラ S レコードフォーマットデータの解析	r_fw_up_buf.c
fw_up_get_binary	コードフラッシュメモリ書き込みデータの取得	r_fw_up_buf.c
fw_up_ascii_to_hexbyte	アスキー形式データからバイナリ形式データへの変換	r_fw_up_buf.c

表 5.21　ファームウェアアップデートプログラムで使用する FIT モジュールの関数

関数名	FIT モジュール	用途	使用している関数
R_FLASH_Open	フラッシュ FIT モジュール	フラッシュ FIT モジュールの初期化	fw_up_open_flash
R_FLASH_Erase	フラッシュ FIT モジュール	コードフラッシュメモリの消去	erase_firmware
R_FLASH_Write	フラッシュ FIT モジュール	コードフラッシュメモリの書き込み	write_firmware set_write_complete_information
R_FLASH_Control	フラッシュ FIT モジュール	スタートアッププログラム保護機能の領域の入れ替え	erase_firmware write_firmware switch_start_up_and_reset
R_SCI_Open	SCI FIT モジュール	SCI の起動	main
R_SCI_Control	SCI FIT モジュール	送信完了割り込みの有効化	main
R_SCI_Send	SCI FIT モジュール	SCI データ送信	send_byte_xm send_string_sci
R_SCI_Receive	SCI FIT モジュール	SCI データ受信	main recv_byte_xm update_firmware exec_firmware

表 5.22　ファームウェアアップデートプログラムで使用する e^2 studio のスマートコンフィグレータにより生成される関数（RX130 グループ）

関数名	対応する FIT モジュール	用途	使用している関数
R_SCI_PinSet_SCI1	SCI FIT モジュール	SCI 用端子設定	main

表 5.23　ファームウェアアップデートプログラムで使用する e^2 studio のスマートコンフィグレータにより生成される関数（RX140 グループ）

関数名	対応する FIT モジュール	用途	使用している関数
R_SCI_PinSet_SCI1	SCI FIT モジュール	SCI 用端子設定	main

表 5.24　ファームウェアアップデートプログラムで使用する e^2 studio のスマートコンフィグレータにより生成される関数（RX231 グループ）

関数名	対応する FIT モジュール	用途	使用している関数
R_SCI_PinSet_SCI5	SCI FIT モジュール	SCI 用端子設定	main

5.6　ファームウェアプログラム詳細

5.6.1　ファイル構成

表 5.25 にファームウェアプログラムで使用するファイル、表 5.26 にファームウェアプログラムで使用する標準インクルードファイルを示します。なお、FIT モジュールおよび統合開発環境で自動生成されるファイルは除きます。

表 5.25　ファームウェアプログラムで使用するファイル

ファイル名	概要
main.c	メインソースファイル
main.h*1	メインインタフェースファイル
r_fw_up_rx.c*1	ファームウェアアップデートソースファイル
r_fw_up_rx_if.h*1	ファームウェアアップデートインタフェースファイル
r_fw_up_rx_private.h*1	ファームウェアアップデートヘッダファイル
r_fw_up_buf.c*1	ファームウェアデータ用バッファ処理ソースファイル
r_fw_up_buf.h*1	ファームウェアデータ用バッファ処理ヘッダファイル

【注】*1 ファームウェアアップデートプログラムと同一のファイルを使用しています。

表 5.26　ファームウェアプログラムで使用する標準インクルードファイル

ファイル名	概要
stdbool.h	論理型、および論理値に関するマクロを定義します。
stdint.h	指定した幅の整数型を宣言してマクロを定義します。
stdlib.h	記憶領域管理などの C プログラムで標準的処理を行うライブラリです。
string.h	文字列の比較、複写などを行うライブラリです。

5.6.2　　定数一覧

表 5.27 にファームウェアプログラムで使用する定数（main.c）を示します。なお、ファームウェアアップデートプログラムと同一のファイルに記載されている定数は「5.5.2 定数一覧」を参照してください。

表 5.27　ファームウェアプログラムで使用する定数（main.c）

定数名	設定値	内容
RECV_BYTE_SIZE	(1)	受信用 1 バイトのサイズ
COMMAND_CR	('¥r')	入力コマンド用文字コード（改行コード）
STRING_MAX_SIZE	RX130 グループ： (SCI_CFG_CH1_TX_BUFSIZ) RX140 グループ： (SCI_CFG_CH1_TX_BUFSIZ) RX231 グループ： (SCI_CFG_CH5_TX_BUFSIZ)	出力する文字列の最大サイズ

5.6.3　　型定義一覧

型定義は「5.5.3 型定義一覧」を参照してください。

5.6.4　変数一覧

表 5.28 にファームウェアプログラムで使用する static 型変数（main.c）を、表 5.29 にファームウェアプログラムで使用する const 型変数（main.c）を示します。なお、ファームウェアアップデートプログラムと同一のファイルに記載されている変数は「5.5.4 変数一覧」を参照してください。

表 5.28　ファームウェアプログラムで使用する static 型変数（main.c）

型	変数名	内容	使用関数
static sci_hdl	s_sci_handle	SCI モジュール制御ハンドル	main send_string_sci
static volatile bool	s_sci_send_end_flag	SCI 送信完了判定用フラグ	sci_callback send_string_sci

表 5.29　ファームウェアプログラムで使用する const 型変数（main.c）

型	変数名	内容	使用関数
static const uint8_t	s_string_menu0[]	“This program is the sample firmware.¥r¥n”	show_menu_start_up
static const uint8_t	s_string_menu1[]	“Push Enter key to execute firmware update.¥r¥n”	show_menu_start_up
static const uint8_t	s_string_input[]	“> ”	show_menu_start_up
static const uint8_t	s_string_crlf[]	“¥r¥n”	main
static const uint8_t	s_string_reset[]	“Switch Start-Up area and do software reset.¥r¥n”	main
static const uint8_t	s_string_flash_err[]	“Flash module error.¥r¥n”	main
static const uint8_t	s_string_switch_err[]	“Switching Start-Up area error.¥r¥n”	main
static const uint8_t	s_string_init_update_err[]	“Initialize update error.¥r¥n”	main
static const uint8_t	s_string_fin_update_err[]	“Finalize update error.¥r¥n”	main
static const uint8_t	s_string_resetvect_err[]	“Reset vector of the firmware update is invalid.¥r¥n”	main

5.6.5　関数一覧

表 5.30 にファームウェアプログラムの関数を、表 5.31 にファームウェアプログラムで使用する FIT モジュールの関数を、表 5.32～表 5.34 にファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される関数を示します。なお、ファームウェアアップデートプログラムと同一のファイルに記載されており、ファームウェアプログラムでは使用していない関数は除きます。

表 5.30　ファームウェアプログラムの関数

関数名	概要	記載ファイル
main	メイン処理	main.c
show_menu_start_up	メニュー表示	main.c
sci_callback	SCI FIT モジュールのコールバック関数、SCI 送信完了の確認	main.c
send_string_sci	文字列送信処理	main.c
fw_up_open_flash	フラッシュ FIT モジュールの初期化	r_fw_up_rx.c
fw_up_open	ファームウェアアップデートの初期化	r_fw_up_rx.c
fw_up_close	ファームウェアアップデートの終了処理	r_fw_up_rx.c
copy_update_ramprog	RAM プログラムのコピー	r_fw_up_rx.c
switch_start_up_and_reset	スタートアッププログラム保護機能の領域入れ替えとソフトウェアリセット実行	r_fw_up_rx.c
fw_up_buf_init	ファームウェアアップデートで使用するバッファの初期化	r_fw_up_buf.c
fw_up_memory_init	バッファへのポインタの初期化	r_fw_up_buf.c

表 5.31　ファームウェアプログラムで使用する FIT モジュールの関数

関数名	FIT モジュール	用途	使用している関数
R_FLASH_Open	フラッシュ FIT モジュール	フラッシュ FIT モジュールの初期化	fw_up_open_flash
R_FLASH_Control	フラッシュ FIT モジュール	スタートアッププログラム保護機能の領域の入れ替え	switch_start_up_and_reset
R_SCI_Open	SCI FIT モジュール	SCI の起動	main
R_SCI_Control	SCI FIT モジュール	送信完了割り込みの有効化	main
R_SCI_Send	SCI FIT モジュール	SCI データ送信	send_string_sci
R_SCI_Receive	SCI FIT モジュール	SCI データ受信	main

表 5.32　ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される関数（RX130 グループ）

関数名	対応する FIT モジュール	用途	使用している関数
R_SCI_PinSet_SCI1	SCI FIT モジュール	SCI 用端子設定	main

表 5.33　ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される関数（RX140 グループ）

関数名	対応する FIT モジュール	用途	使用している関数
R_SCI_PinSet_SCI1	SCI FIT モジュール	SCI 用端子設定	main

表 5.34　ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される関数（RX231 グループ）

関数名	対応する FIT モジュール	用途	使用している関数
R_SCI_PinSet_SCI5	SCI FIT モジュール	SCI 用端子設定	main

6.　プロジェクトをインポートする方法

サンプルプログラムは e^2 studio のプロジェクト形式で提供しています。本章では、CS+へプロジェクトをインポートする方法を示します。インポート完了後、ビルドおよびデバッグの設定を確認してください。

6.1　CS+での手順

CS+でご使用になる際は、下記の手順で CS+にインポートしてください。

なお、CS+で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、 空白文字の他、半角カナ文字、全角文字、半角記号(特に'$','#','%') が混じらないようにしてください。

(使用する CS+のバージョンによっては画面が異なる場合があります。)

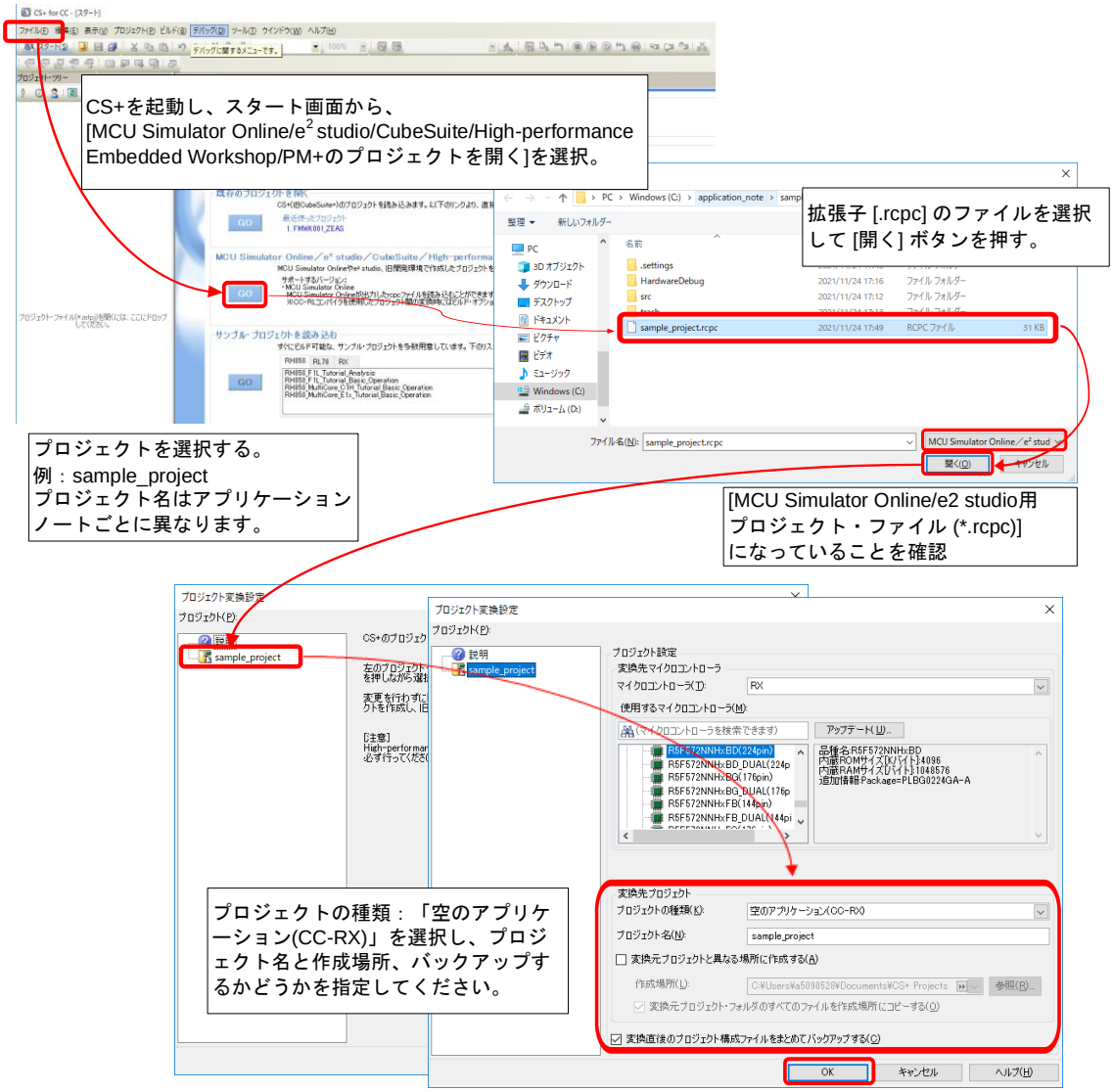


図 6.1　プロジェクトを CS+にインポートする方法

7.　参考ドキュメント

- RX110 グループ ユーザーズマニュアル ハードウェア編 （R01UH0421）
- RX111 グループ ユーザーズマニュアル ハードウェア編 （R01UH0365）
- RX113 グループ ユーザーズマニュアル ハードウェア編 （R01UH0448）
- RX130 グループ ユーザーズマニュアル ハードウェア編 （R01UH0560）
- RX140 グループ ユーザーズマニュアル ハードウェア編 （R01UH0905）
- RX230 グループ、RX231 グループ ユーザーズマニュアル ハードウェア編 （R01UH0496）
- （最新版をルネサスエレクトロニクスホームページから入手してください）
- テクニカルアップデート/テクニカルニュース
  （最新の情報をルネサスエレクトロニクスホームページから入手してください）
- C コンパイラマニュアル
- RX ファミリ用 C/C++コンパイラパッケージ
  （最新版をルネサスエレクトロニクスホームページから入手してください）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jul.01.17	-	初版発行
1.10	Sep.26.18	-	RX100 シリーズを追加
1.20	Mar.24.22	全体	対象デバイスに以下を追加 ・RX140 シリーズ 以下の理由でサンプルプログラムの仕様を変更 ファームウェアアップデートプログラム容量削減のため ・CMT モジュールを削除 ・プログラム部と定数部を分離 プログラム更新時の利便性を向上させるため ・書き換え完了後のファームウェア起動手順を変更 表 1.2 の内容を変更 表 1.5 の設定内容を変更 「2.2.コンパイラパッケージの入手方法」と「2.3.Renesas Flash Programmer の入手方法」の URL を変更 「3.プロジェクトの構築」と「4.動作確認」を更新 「5.1.ファームウェアアップデートプログラムの構成」を追加 「5.2.動作概要」で使用する図を変更 「5.2.7.更新用ファームウェアアップデートプログラムの作成方法」を追加 「5.3.ファームウェアアップデートプログラムの概略フローと画面出力」と「5.4.ファームウェアプログラムの概略フローと画面出力」で使用する図を変更 「5.5.ファームウェアアップデートプログラム詳細」と「5.6.ファームウェアプログラム詳細」を更新 「6.プロジェクトをインポートする方法」の内容を変更 「7.参考ドキュメント」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。