RX ファミリ

Flash モジュール、SCI モジュールとデュアルバンク機能を用いたファームウェアアップデートサンプルプログラム
Firmware Integration Technology

要旨

本アプリケーションノートでは、RX ファミリのデュアルバンク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。アプリケーションソフトウェアの制御とデータの転送にシリアル通信を使用します。

対象デバイス

RX65N グループ、RX651 グループ	ROM 容量：1.5MB～2MB
RX72M グループ	ROM 容量：2MB～4MB

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。併せて参照してください。

- 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 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 CMT モジュール Firmware Integration Technology (R01AN1856)
- RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)

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、デュアルバンク機能を使用し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。ホスト PC からシリアル通信を使用してマイコン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としては XMODEM/SUM を使用します。ホスト PC 側のシリアル通信ソフトウェアは XMODEM/SUM 転送機能を持つシリアル通信ソフトウェアを使用してください。

本アプリケーションノートは RX65N グループ、RX651 グループの ROM 容量が 2MB のマイコンと RX72M グループの ROM 容量が 4MB のマイコンのサンプルプログラムを含んでいます。他のデバイスでは「6.2.1 Q:デバイスを変更したい」を参照してください。

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
コンペアマッチタイマ	シリアル通信のタイムアウト確認用タイマ

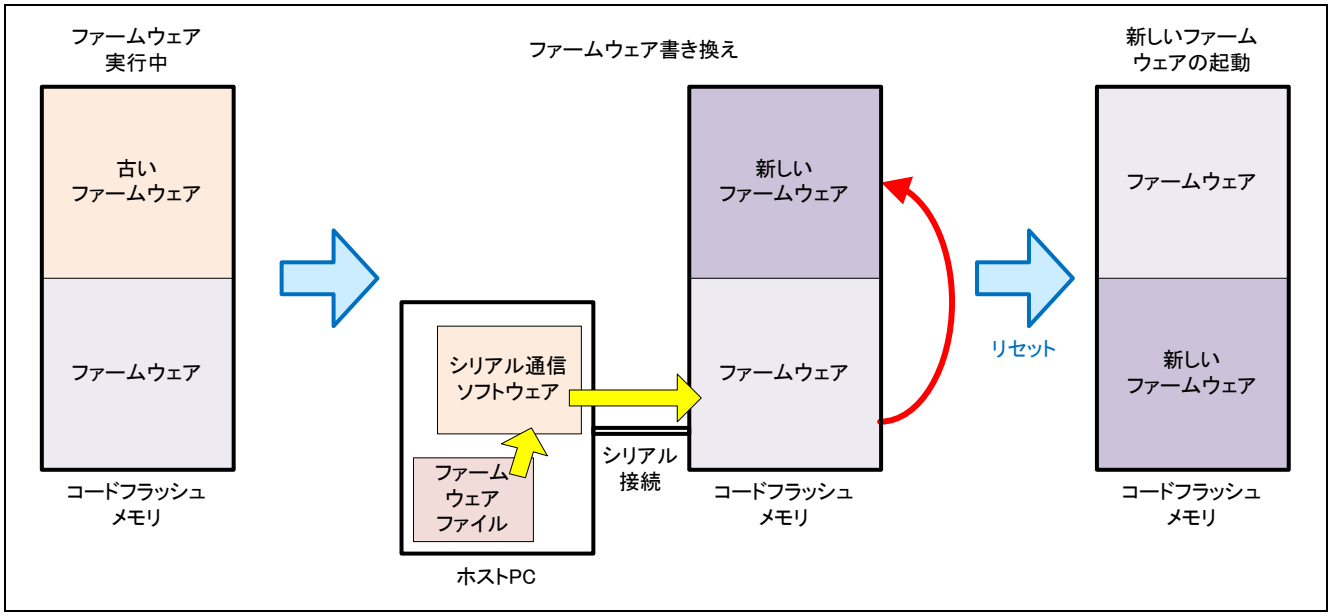


図 1.1　動作概要

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 の条件で動作を確認しています。

表 1.2 動作確認条件

項目	内容
使用マイコン	R5F565NEDDFC (RX65N グループ)
動作周波数	• メインクロック: 24MHz • PLL: 240MHz (メインクロック 1 分周 10 逓倍) • システムクロック(ICLK): 120MHz (PLL 2 分周) • FlashIF クロック(FCLK): 60MHz (PLL 4 分周)
動作電圧	3.3V
統合開発環境	ルネサスエレクトロニクス製 e^2 studio Version 7.5.0
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.3.01.00
	コンパイルオプション -lang = c99
iodefine.h のバージョン	Version 2.3
エンディアン	リトル・エンディアン、ビッグ・エンディアン(注 1)
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
サンプルプログラムバージョン	Version 1.20
フラッシュプログラマ	Renesas Flash Programmer V3.06.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB (製品型名: RTK50565N2C01001BR) (以下、RSK+RX65N-2MB)

項目	内容
使用マイコン	R5F572MNDDBD (RX72M グループ)
動作周波数	• メインクロック: 24MHz • PLL: 240MHz (メインクロック 1 分周 10 逓倍) • システムクロック(ICLK): 240MHz (PLL 1 分周) • FlashIF クロック(FCLK): 60MHz (PLL 4 分周)
動作電圧	3.3V
統合開発環境	ルネサスエレクトロニクス製 e² studio Version 7.5.0
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.3.01.00
	コンパイルオプション -lang = c99
iodefine.h のバージョン	Version 1.0C
エンディアン	リトル・エンディアン、ビッグ・エンディアン(注 1)
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
サンプルプログラムバージョン	Version 1.20
フラッシュプログラマ	Renesas Flash Programmer V3.06.00
使用ボード	Renesas Starter Kit+ for RX72M (製品型名: RTK5572MNDC00000BJ) (以下、RSK+RX72M)

【注 1】サンプルプログラムを構築するプロジェクトの構成はリトル・エンディアンで動作するように作成されています。ビッグ・エンディアンで動作する構成に変更するには「6.2.2 Q:ビッグ・エンディアンに変更したい」を参照してください。

1.3　モジュール構成

図 1.2 にデュアルバンクファームウェアアップデート SCI プロジェクトのモジュール構成を、表 1.3 にデュアルバンクファームウェアアップデート SCI プロジェクトに組み込む FIT モジュールの一覧を示します。

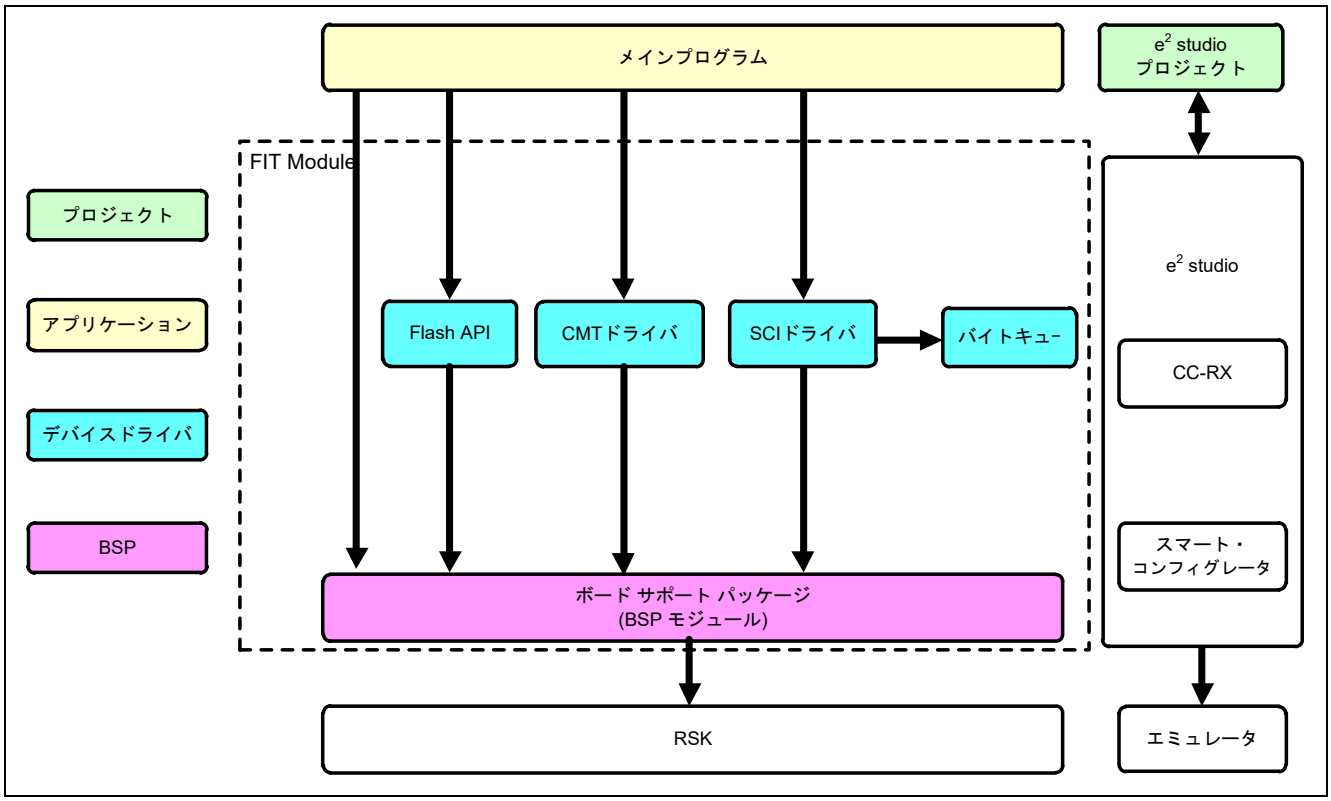


図 1.2　モジュールの構成

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	Rev.
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	5.2
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	1.60
	RX ファミリ CMT モジュール Firmware Integration Technology (R01AN1856)	r_cmt_rx	3.20
	RX ファミリ フラッシュモジュール Firmware Integration Technology (R01AN2184)	r_flash_rx	4.20
	RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)	r_sci_rx	2.0
アプリケーション	メインプログラム	-	1.20

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。

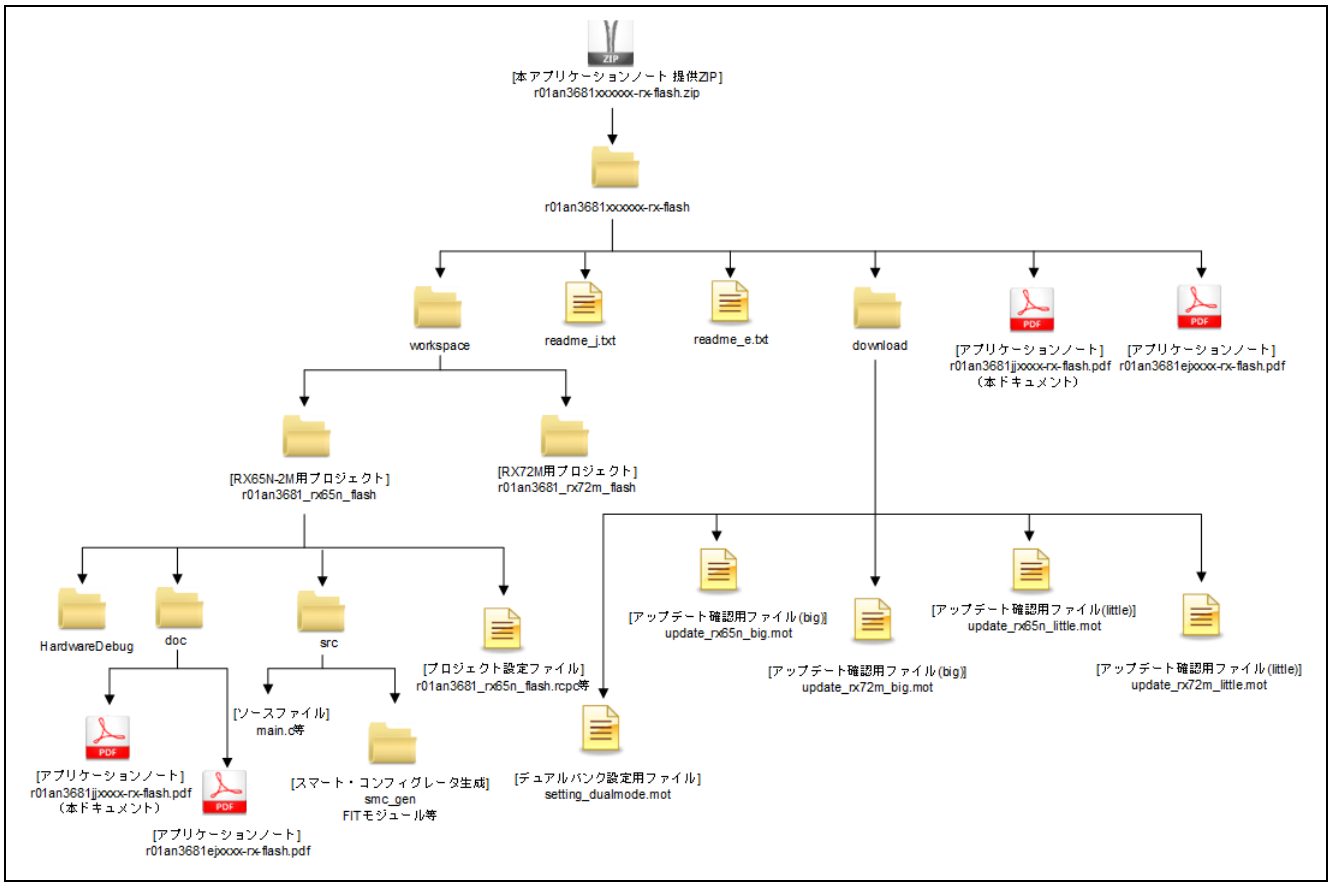


図 1.3　ファイル構成

本アプリケーションノート提供 ZIP ファイルを展開すると、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。

「r01an3681_rx65n_flash」フォルダと「r01an3681_rx72m_flash」フォルダは、本アプリケーションノートのサンプルプログラムを構築するプロジェクトです。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、サンプルプログラム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。プロジェクトには、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。
	Release（Release - No Debug）	デバッグ情報を含まないリリース用の構成です。
デバッグ構成	r01an3681_rx65n_flash HardwareDebug	「HardwareDebug（Debug on Hardware）」で生成したロードモジュールを使用して、E2 エミュレータ Lite 経由でのハードウェアデバッグを行います。
	r01an3681_rx72m_flash HardwareDebug	

また、ターゲット固有の設定は以下のとおりです。

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	V3.01.00
デバッグ・ハードウェア	E2 エミュレータ Lite
エンディアン	リトル・エンディアン
ターゲットの選択	R5F565NEDxFC_DUAL
	R5F572MNDxBD_DUAL
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 V7.5.0 以降を使用することを前提としています。V7.5.0 よりも古い Ver.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 Ver.の e² studio を入手してください。

2.2　コンパイラパッケージの入手方法

以下の URL にアクセスして、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/ja-jp/products/software-tools/tools/compiler-assembler/compiler-package-for-rx-family.html

2.3　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/ja-jp/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.htmling-gui.html

2.4　RSK USB Serial Driver の入手方法

以下の URL にアクセスし、RSK USB Serial Driver をダウンロードしてください。

インストールの詳細は RSK のマニュアルを参照してください。

https://www.renesas.com/ja-jp/software/D6000699.html

3.　プロジェクトのインポート

サンプルプログラムをビルドするためのプロジェクトを e² studio にインポートする手順を説明します。

3.1　ワークスペースの作成

1. e² studio を起動してください。ワークスペースの選択をするダイアログが開きます。ワークスペースの選択をするダイアログが開かない場合は、[ファイル(F)]→[ワークスペースの切り替え(W)]→[その他(O)]をクリックしてください。
2. 任意のワークスペースを入力して、[起動]をクリックしてください。ワークスペースではないフォルダを指定すると新規にワークスペースを作成します。

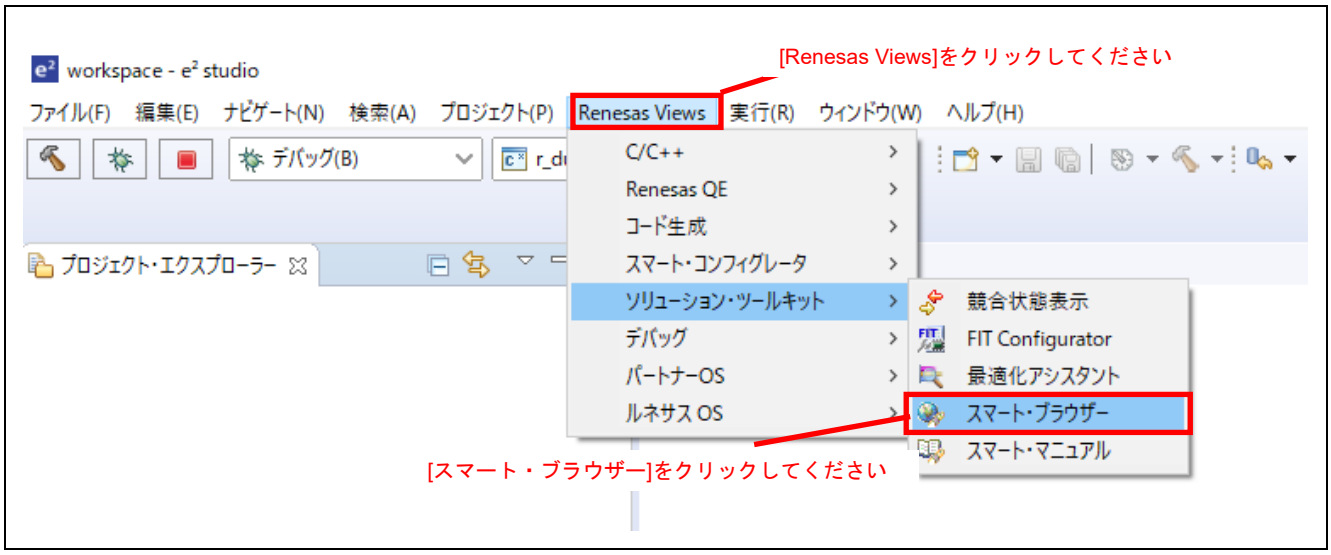
3. 新規にワークスペースを作成した場合は[ようこそ]が開きます。[ワークベンチ]をクリックしてください。

3.2　スマート・ブラウザーからインポート

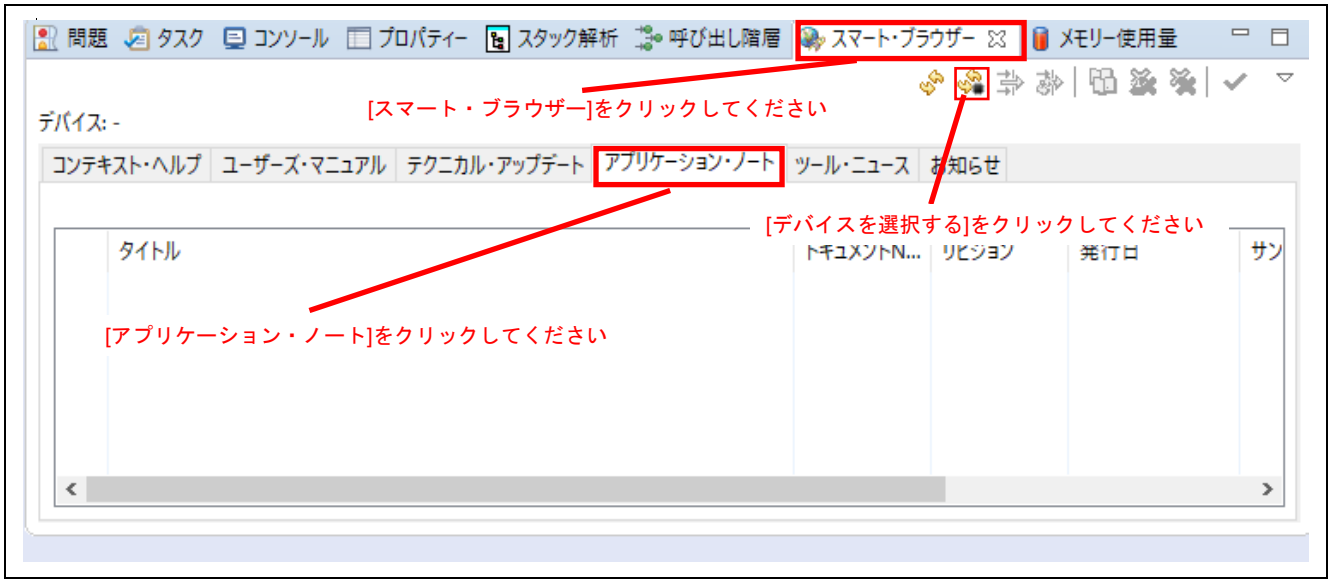
e² studio のスマート・ブラウザーを用いてプロジェクトをインポートする手順について、以下に説明します。

3.2.1　プロジェクトのインポート

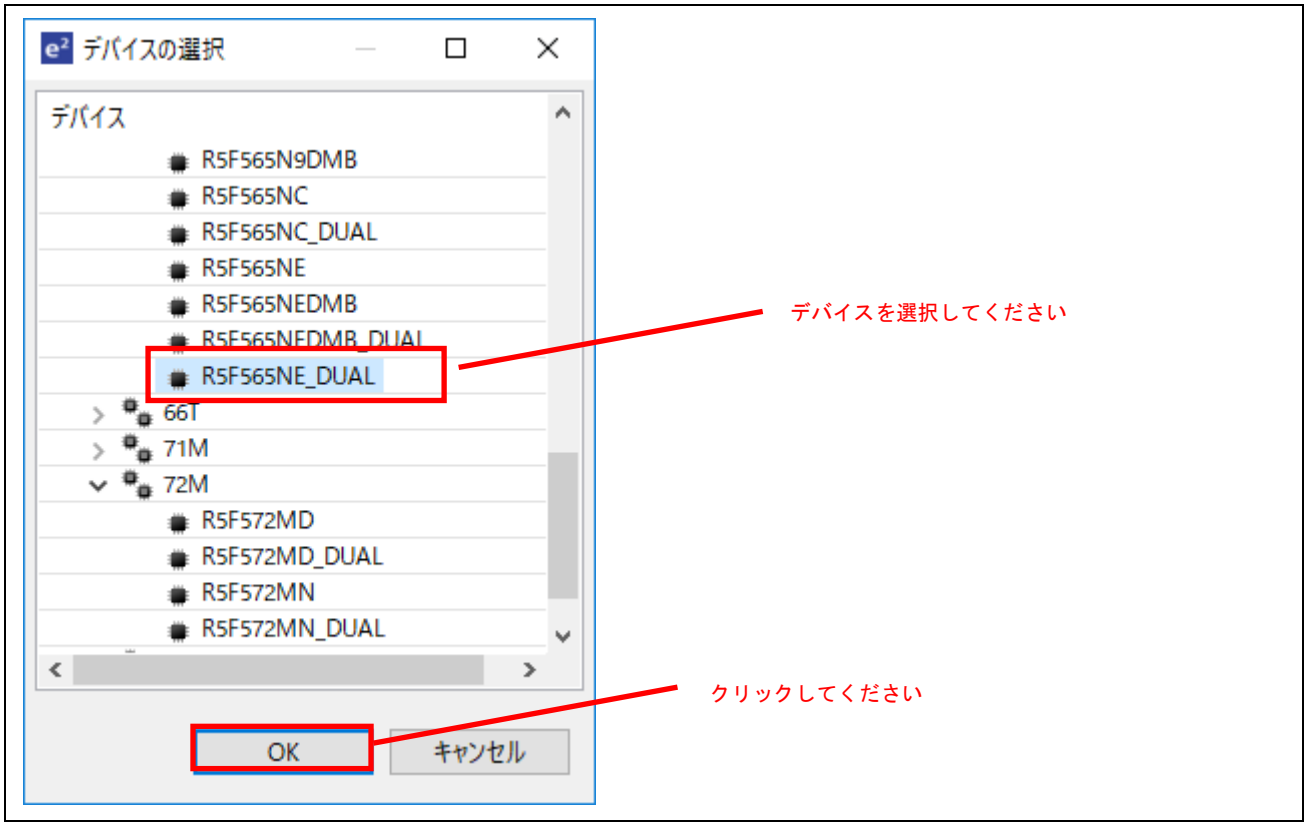
1. [スマート・ブラウザー]タブをクリックしてください。
2. [スマート・ブラウザー]タブが表示されていない場合は、メインメニューの[Renesas Views]をクリックしてください。[ソリューション・ツールキット]の[スマート・ブラウザー]をクリックしてください。



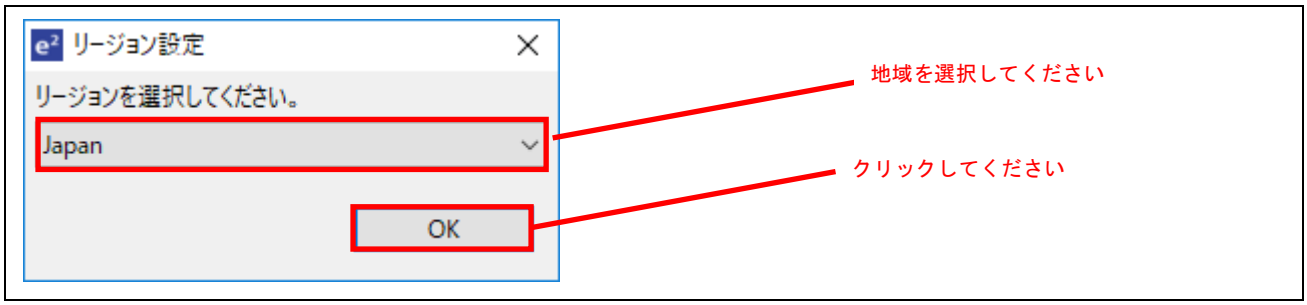
3. [アプリケーション・ノート]タブをクリックしてください。
4. [デバイスを選択する]をクリックしてください。



5. [デバイスの選択]ダイアログが表示されます。使用するデバイスを選択してください。
6. [OK]をクリックしてください。

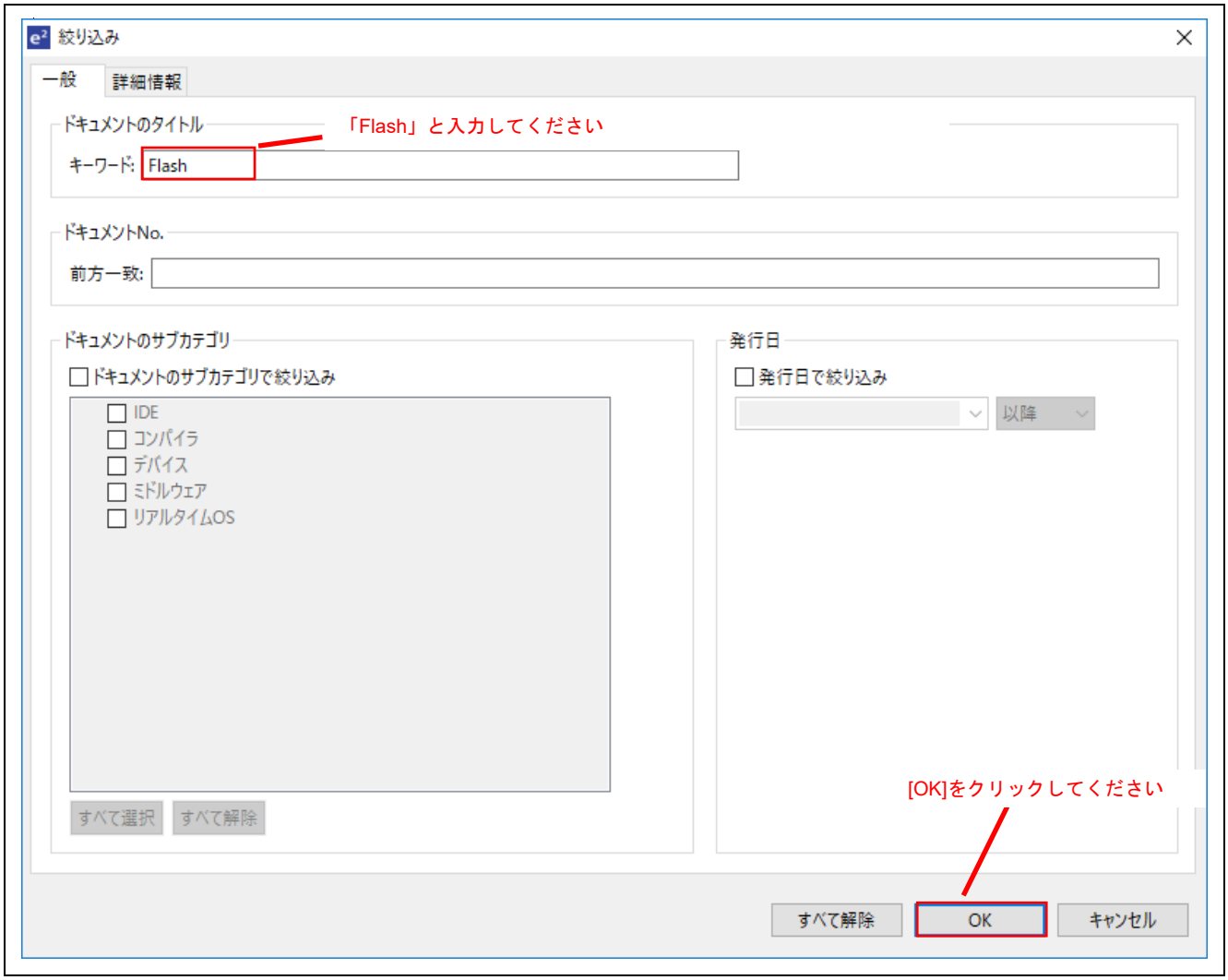


7. 初めてスマート・ブラウザーを使用した場合、[リージョン設定]ダイアログが表示されます。
8. 作業している地域を選択してください。
9. [OK]をクリックしてください。

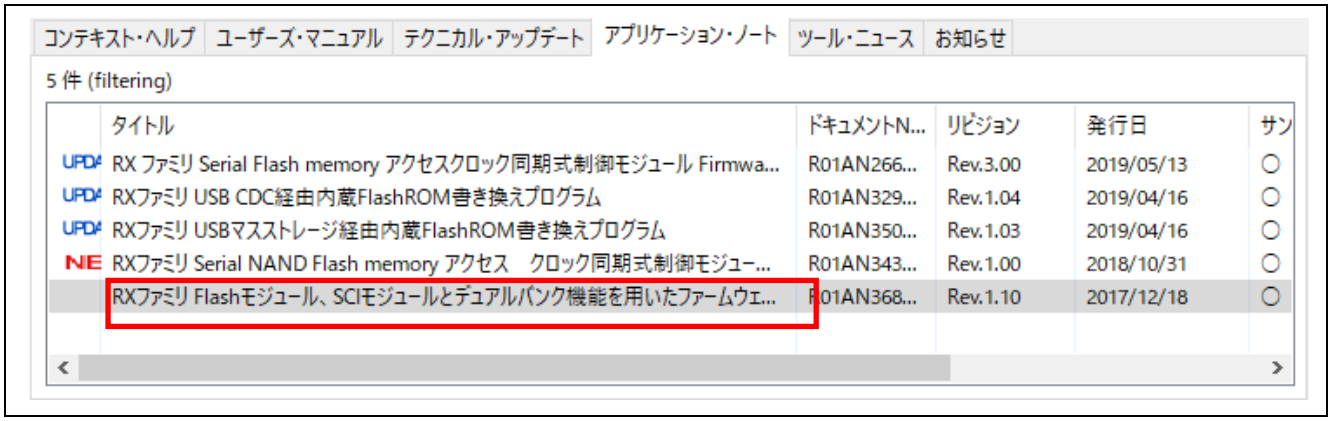


10. しばらくお待ちください。アプリケーションノートの一覧が表示されます。
11. [絞り込み]をクリックしてください。

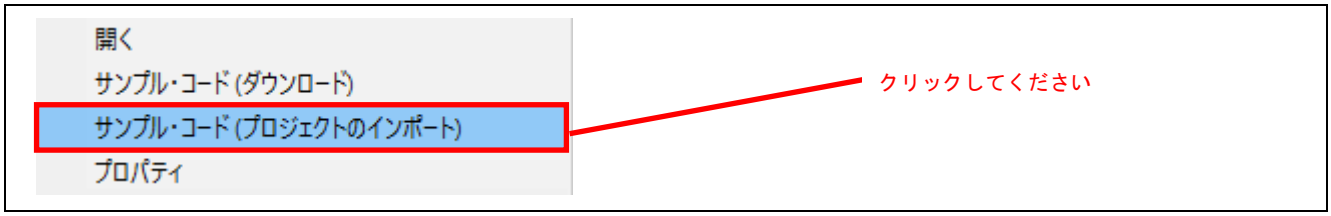
12. ドキュメントのタイトルのキーワードに「Flash」と入力してください。
13. [OK]をクリックしてください。



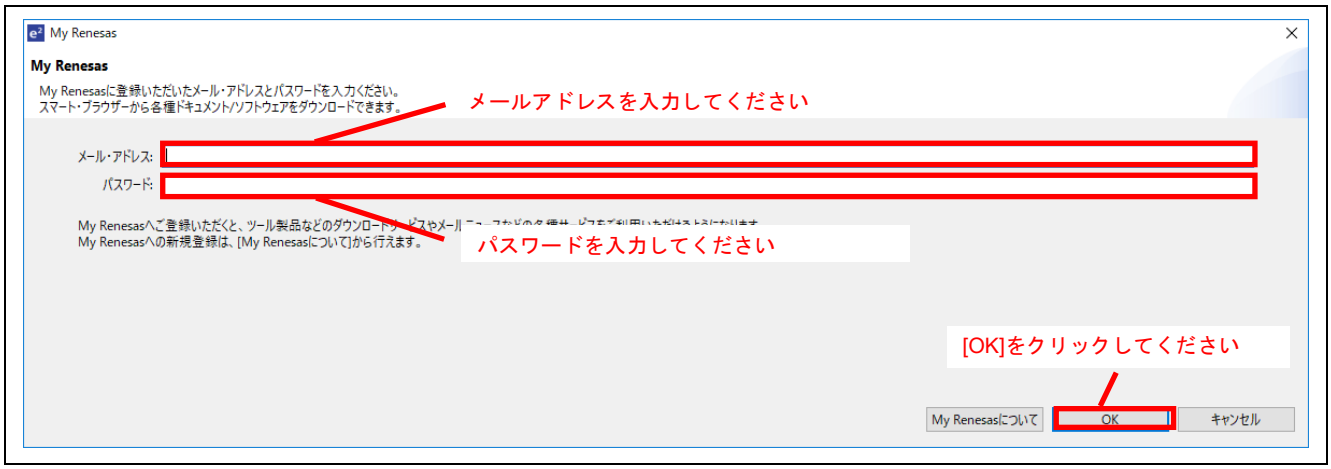
14. 本アプリケーションノートを選択し、右クリックしてください。



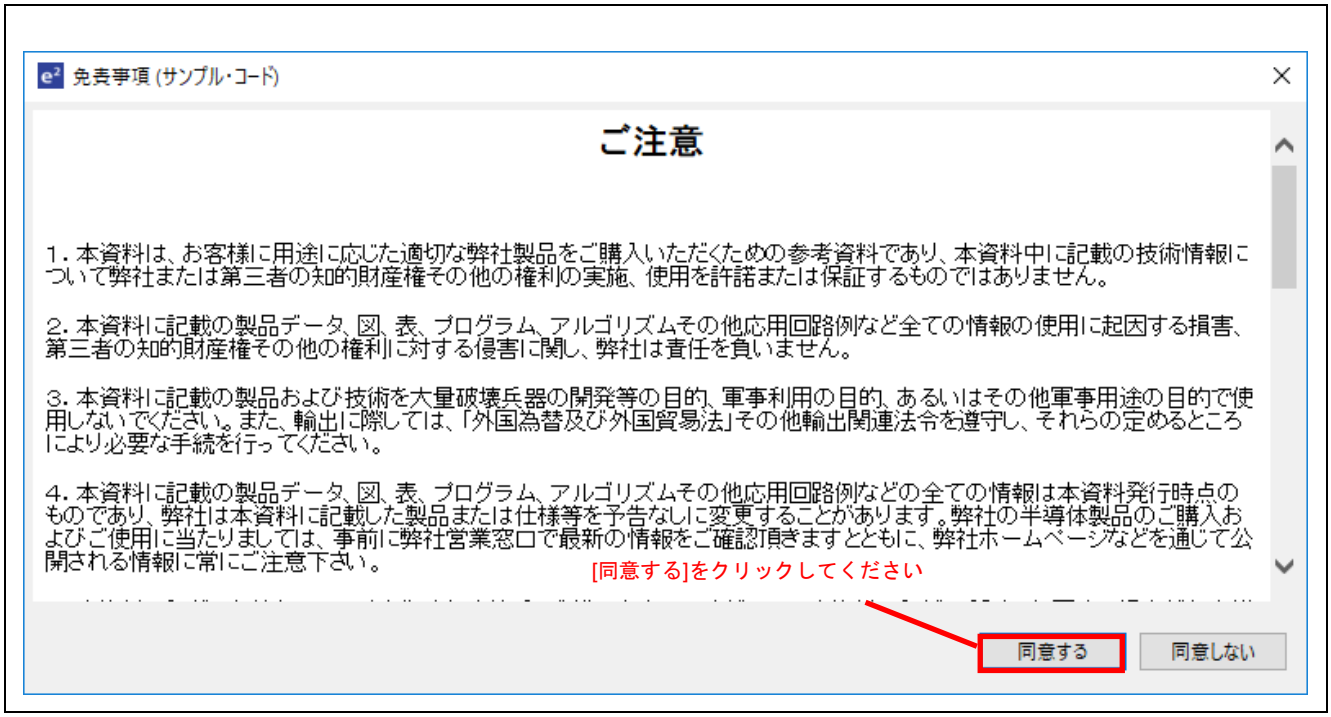
15. コンテキストメニューの[サンプル・コード(プロジェクトのインポート)]をクリックしてください。



16. ファイル保存ダイアログが開きます。保存場所を選んで[保存]をクリックしてください。
17. My Renesas による認証をしていない場合、ファイルをダウンロードする前に[My Renesas]ダイアログが開きます。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。
18. [OK]をクリックしてください。



19. 免責事項に同意できる場合は[同意する]をクリックしてください。



20. [インポート]ダイアログが開きます。[プロジェクト(P):]に表示されている[r01an3681_rx65n_flash]か[r01an3681_rx72m_flash]を使用するデバイスに合わせて選択してください。
21. [終了(F)]をクリックしてください。

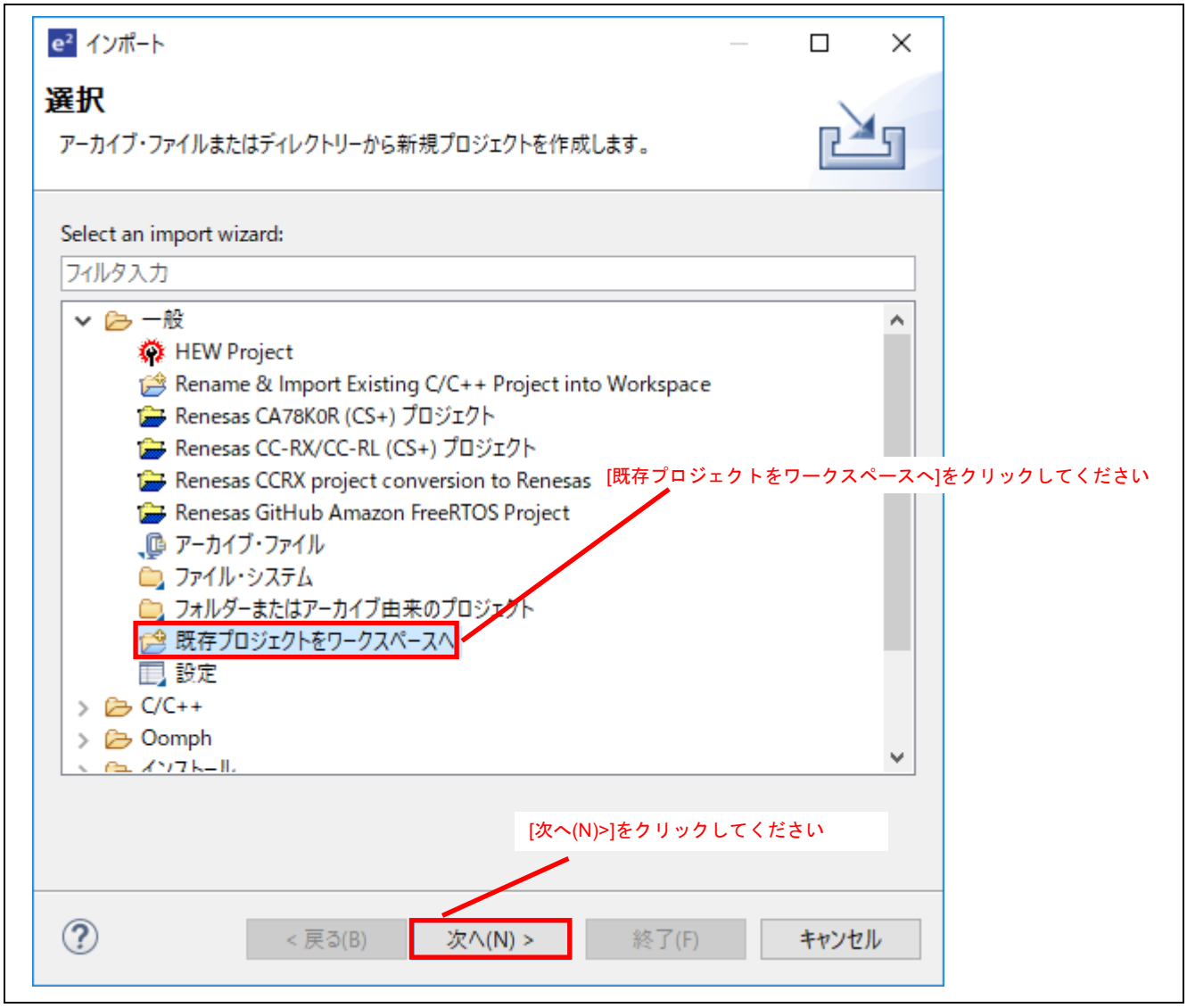
3.3　zip ファイルからインポート

本アプリケーションノートの zip ファイルからプロジェクトをインポートすることが出来ます。

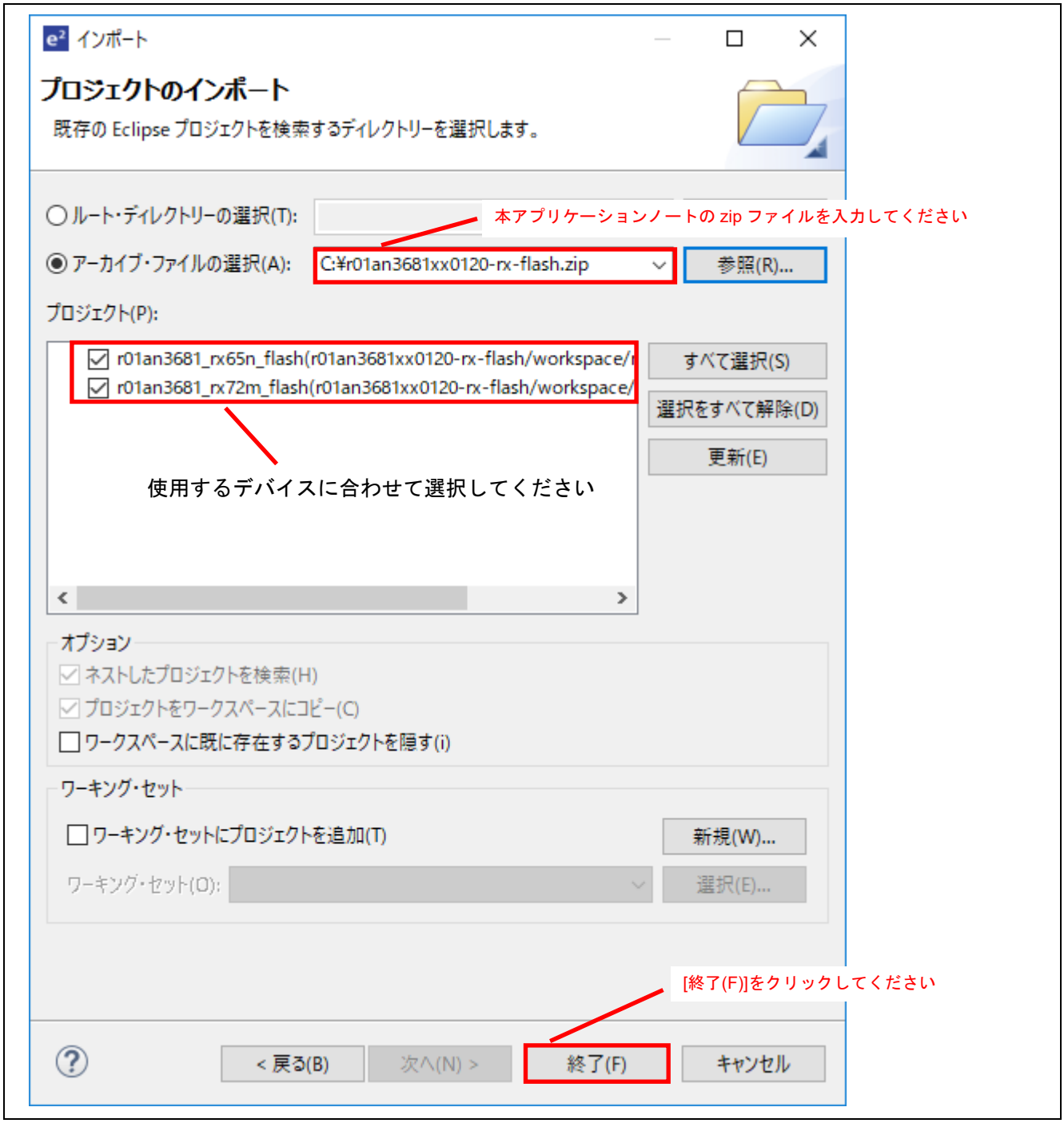
3.3.1　ワークスペースにプロジェクトをインポートする

1. [ファイル(F)]をクリックしてください。
2. [インポート(I)]をクリックしてください。

3. [一般]の [既存プロジェクトをワークスペースへ]をクリックしてください。
4. [次へ(N)>]をクリックしてください。



5. [アーカイブ・ファイルの選択(A):]のコンボボックスに本アプリケーションノートの zip ファイルを入力してください。
6. [プロジェクト(P):]に表示されている[r01an3681_rx65n_flash]か[r01an3681_rx72m_flash]を使用するデバイスに合わせて選択してください。
7. [終了(F)]をクリックしてください。



3.4　変更情報

サンプルプログラムのプロジェクトはソフトウェアコンポーネント設定と[C/C++ビルド]の設定を変更しています。また端子設定も行っています。以下にその詳細を示します。

なお、本変更情報は、新規にプロジェクトを構築する場合に参照してください。インポートしたプロジェクトを使用する場合は「4 動作確認」に進んでください。

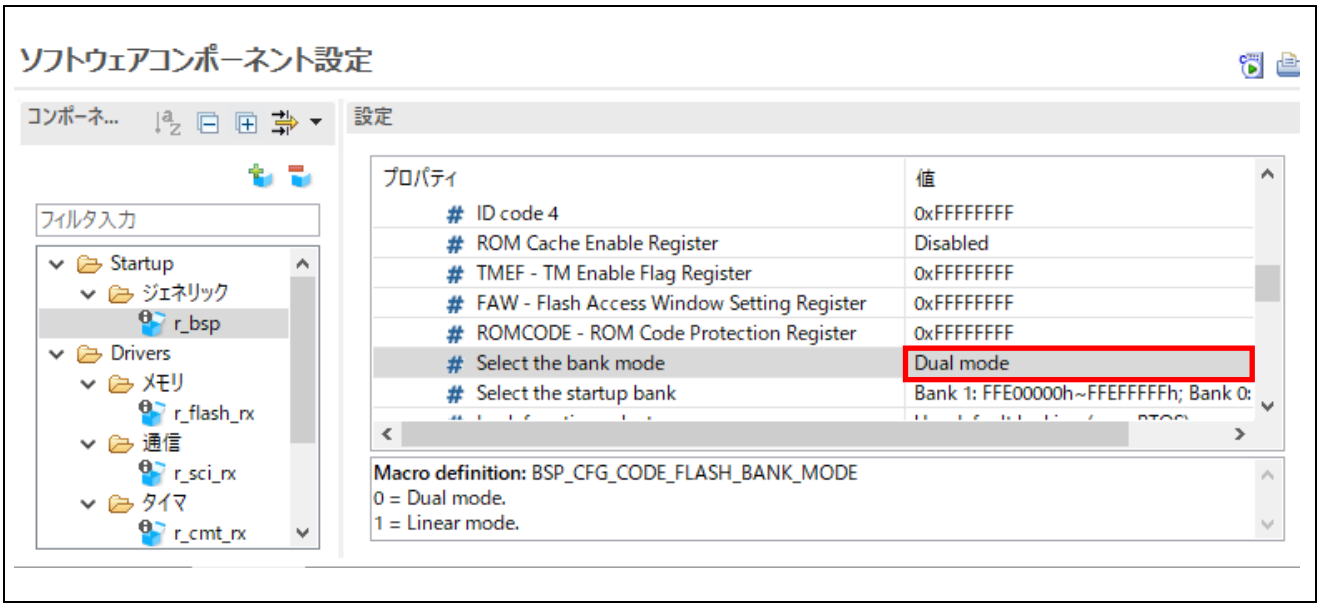
3.4.1　ソフトウェアコンポーネント設定の変更

(1)　BSP の設定変更

【 src/smc_gen/r_config/r_bsp_config.h 】

Startup>ジェネリック>r_bsp

bank mode を Dual mode に変更しています。



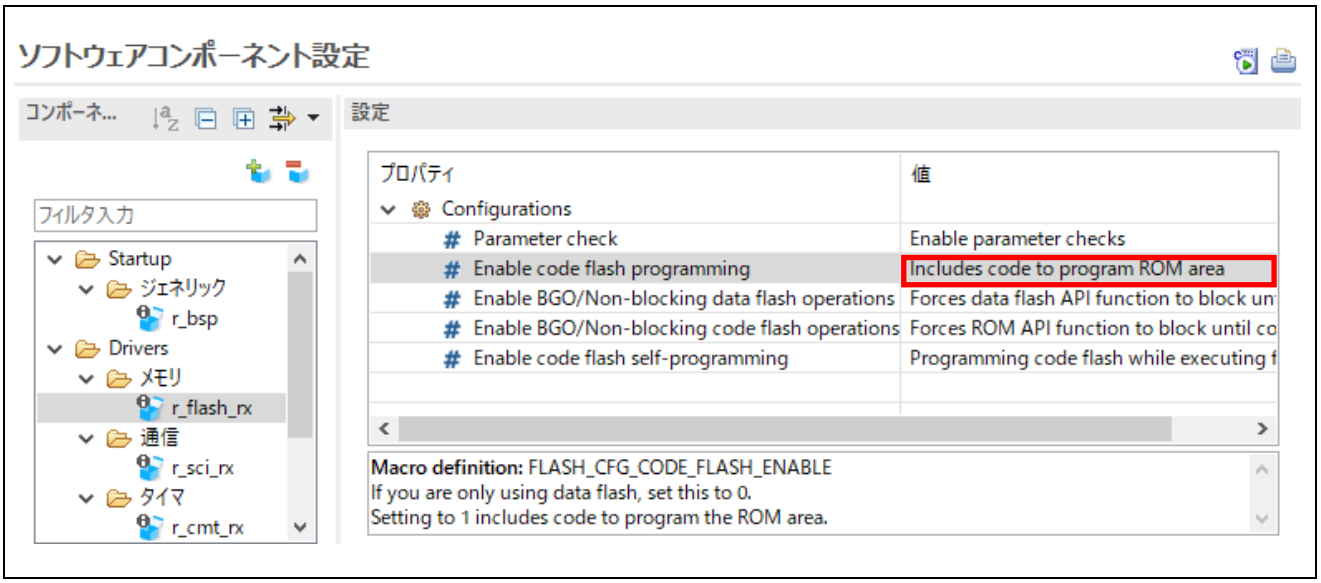
(2)　Flash API の設定変更

Flash API の設定を、以下のように変更しています。

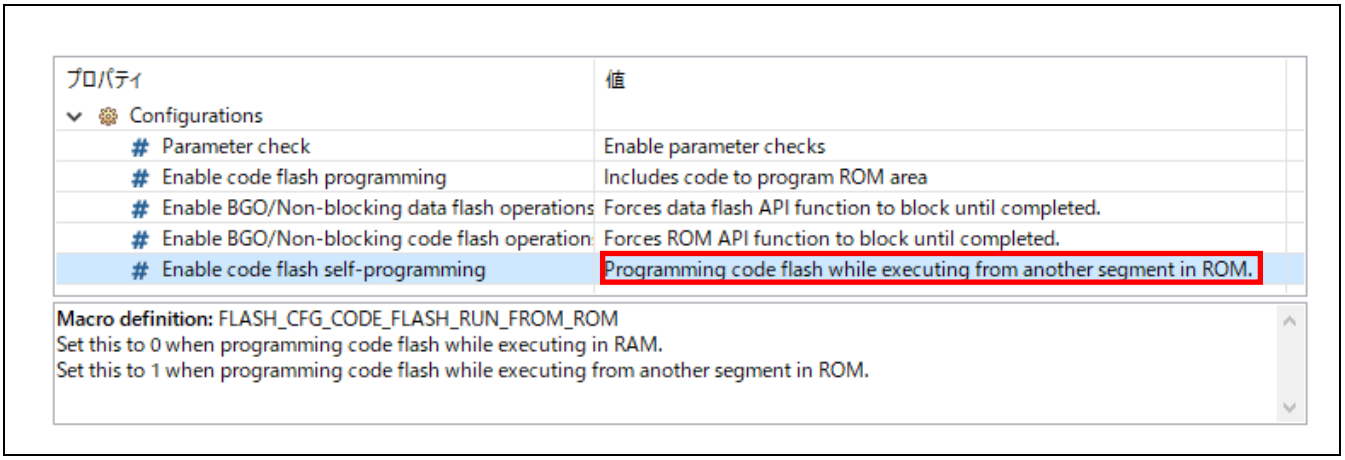
【 src/smc_gen/r_config/r_flash_rx_config.h 】

Drivers>メモリ>r_flash_rx

Flash API でコードフラッシュメモリの書き換えができるように変更しています。



ROM 上でプログラムを実行するように変更しています。

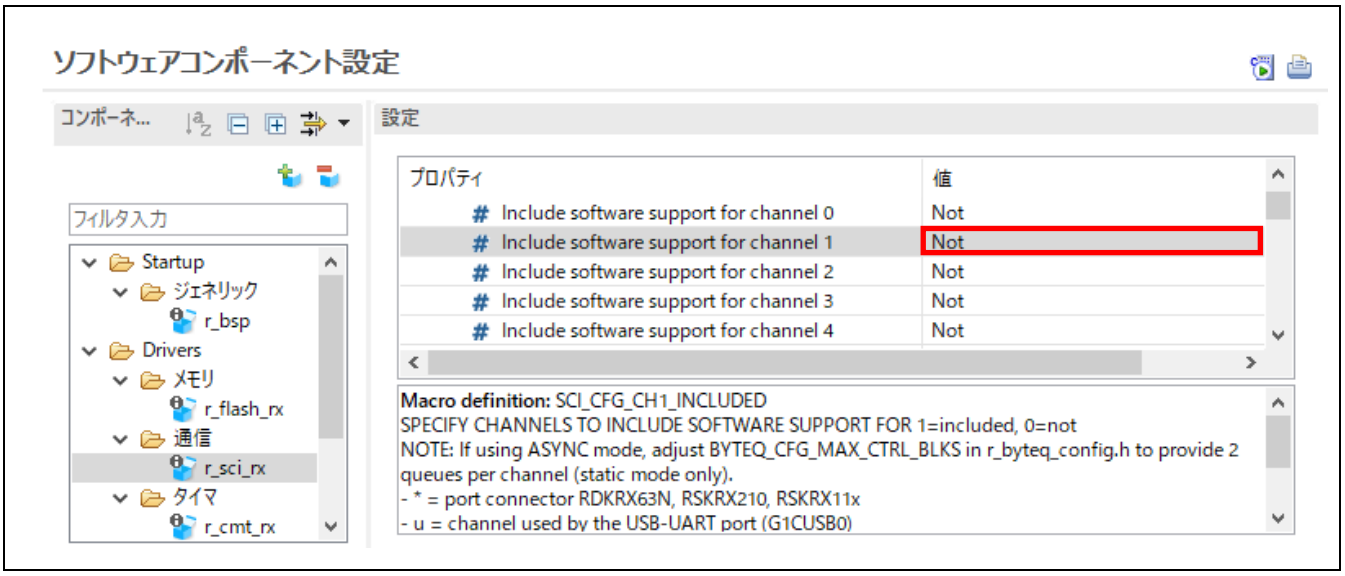


(3)　SCI API の設定変更

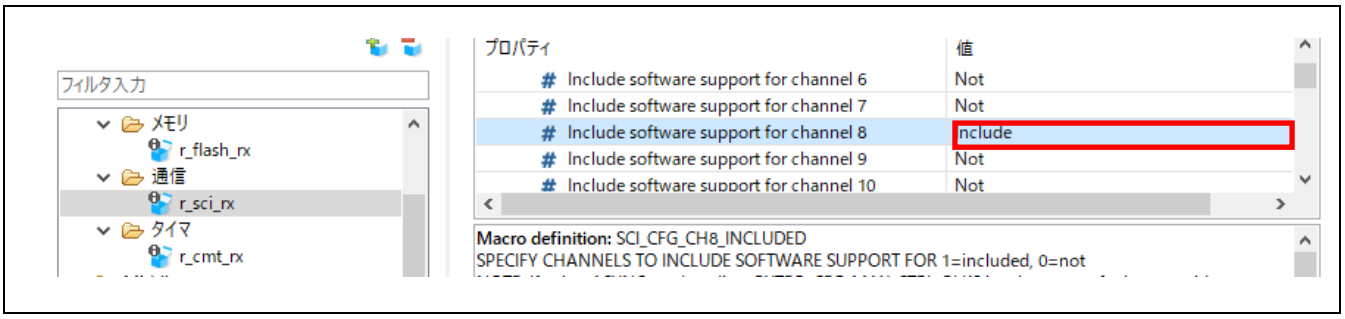
【 src/smc_gen/r_config/r_sci_rx_config.h 】

Drivers>通信>r_sci_rx

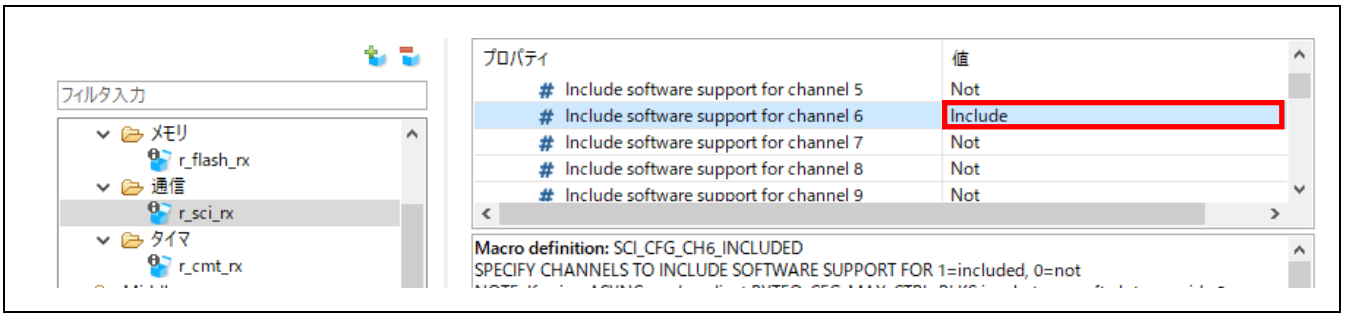
RSK+の USB Serial Port を使用するために SCI のチャネルを変更しています。CH1 を[Not]に変更しています。



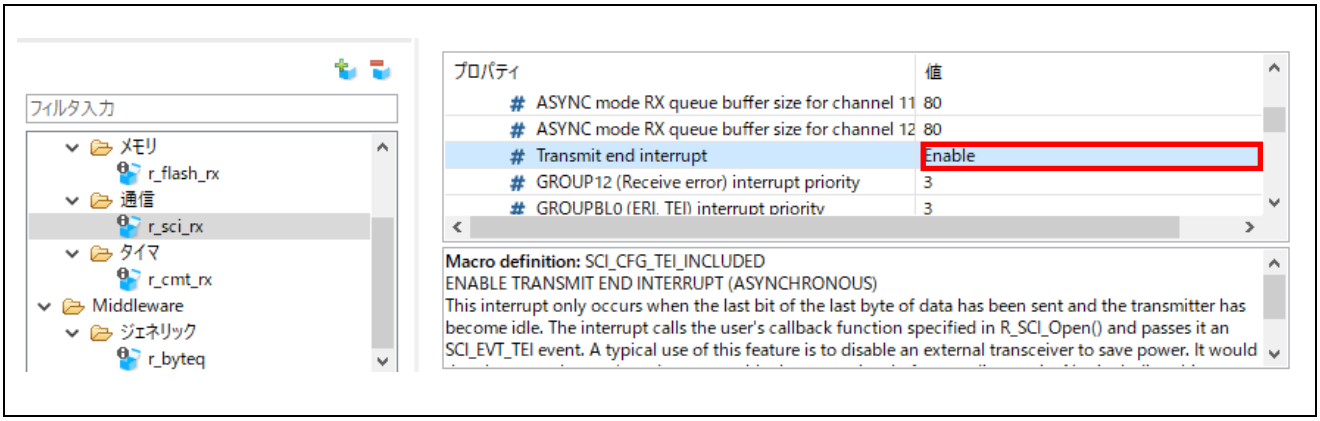
RX65N の場合は CH8 を[Include]に変更しています。



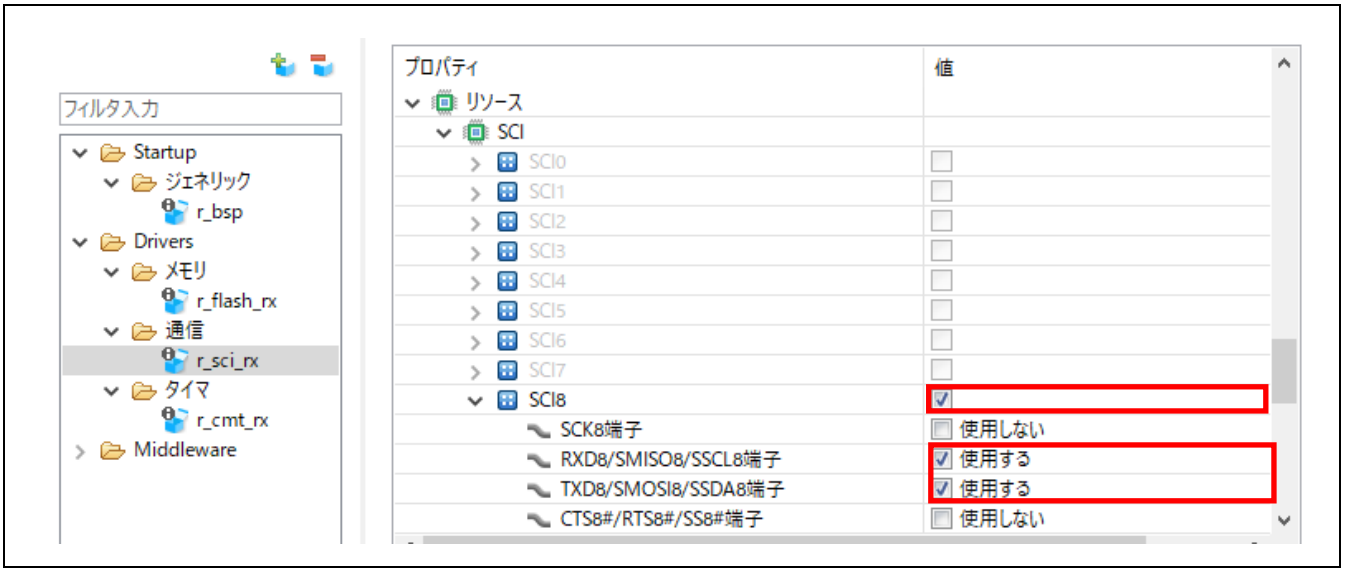
RX72M の場合は CH6 を[Include]に変更しています。



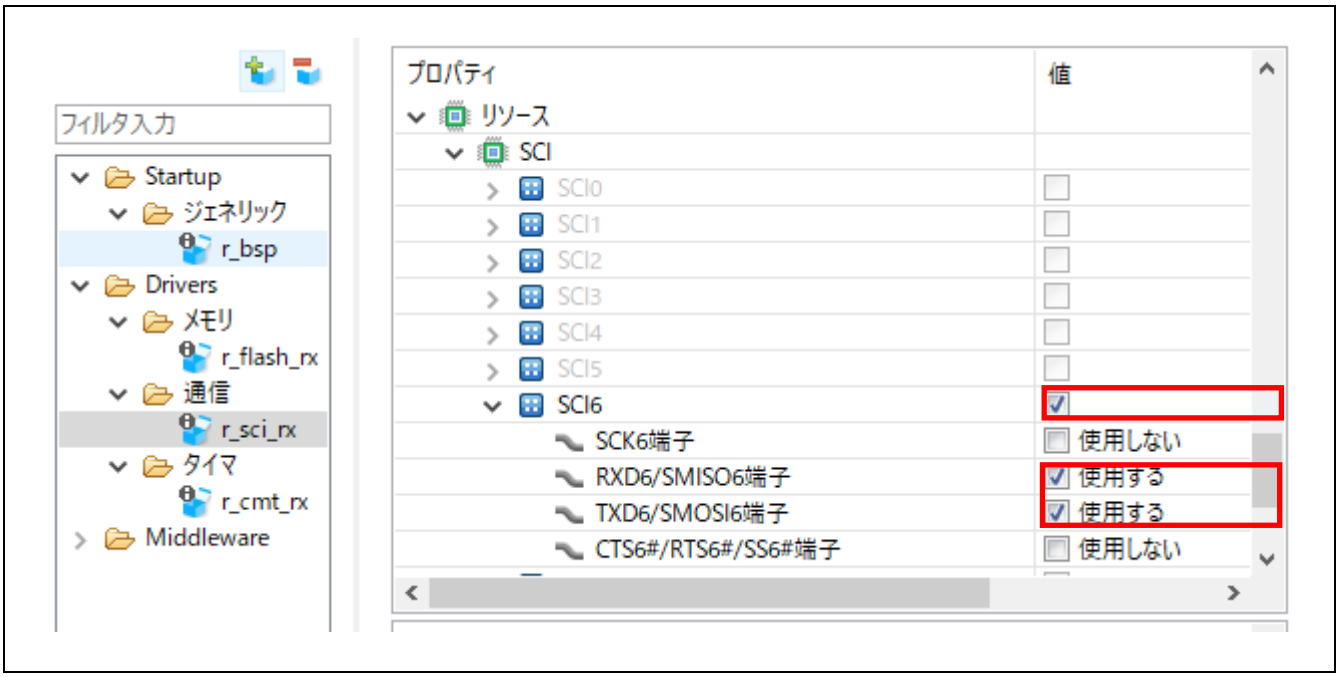
送信完了割り込みを[Enable]に変更しています。



RX65N の場合はリソースの RXD8 と TXD8 を[使用する]に変更しています。



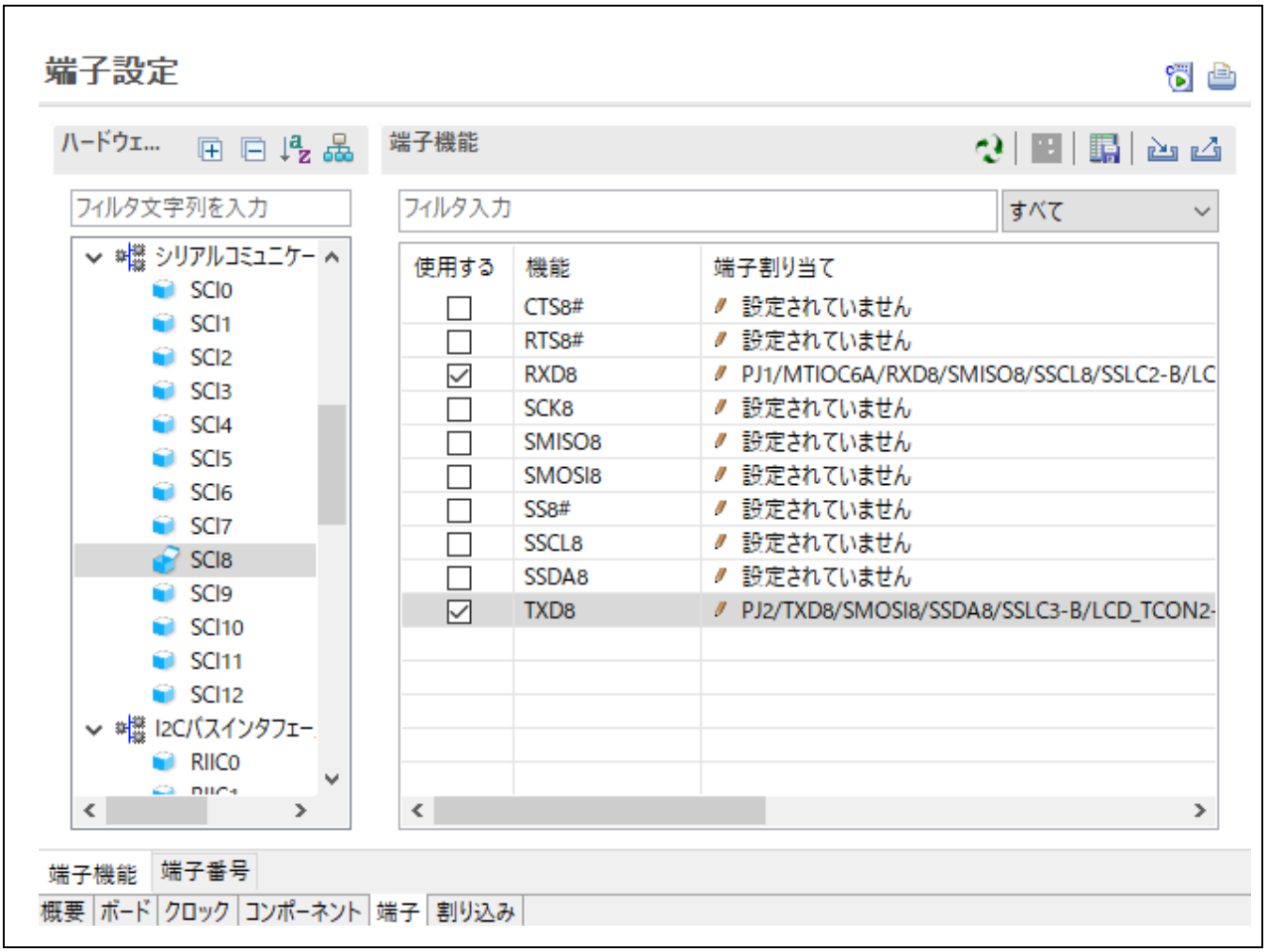
RX72M の場合はリソースの RXD6 と TXD6 を[使用する]に変更しています。



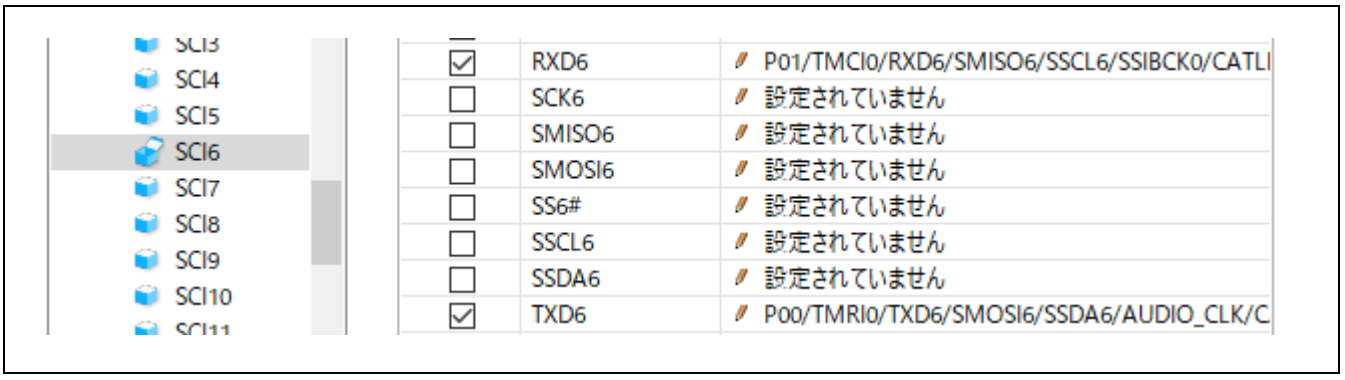
3.4.2　端子設定

[端子]タブをクリックして端子設定の画面にします。

RX65N の場合は RXD8 と TXD8 に割り当てる端子を指定しています。リソースで RXD8 と TXD8 を[使用する]にすると自動でチェックが入ります。



RX72M の場合はリソースの RXD6 と TXD6 に割り当てる端子を指定しています。リソースで RXD6 と TXD6 を[使用する]にすると自動でチェックが入ります。

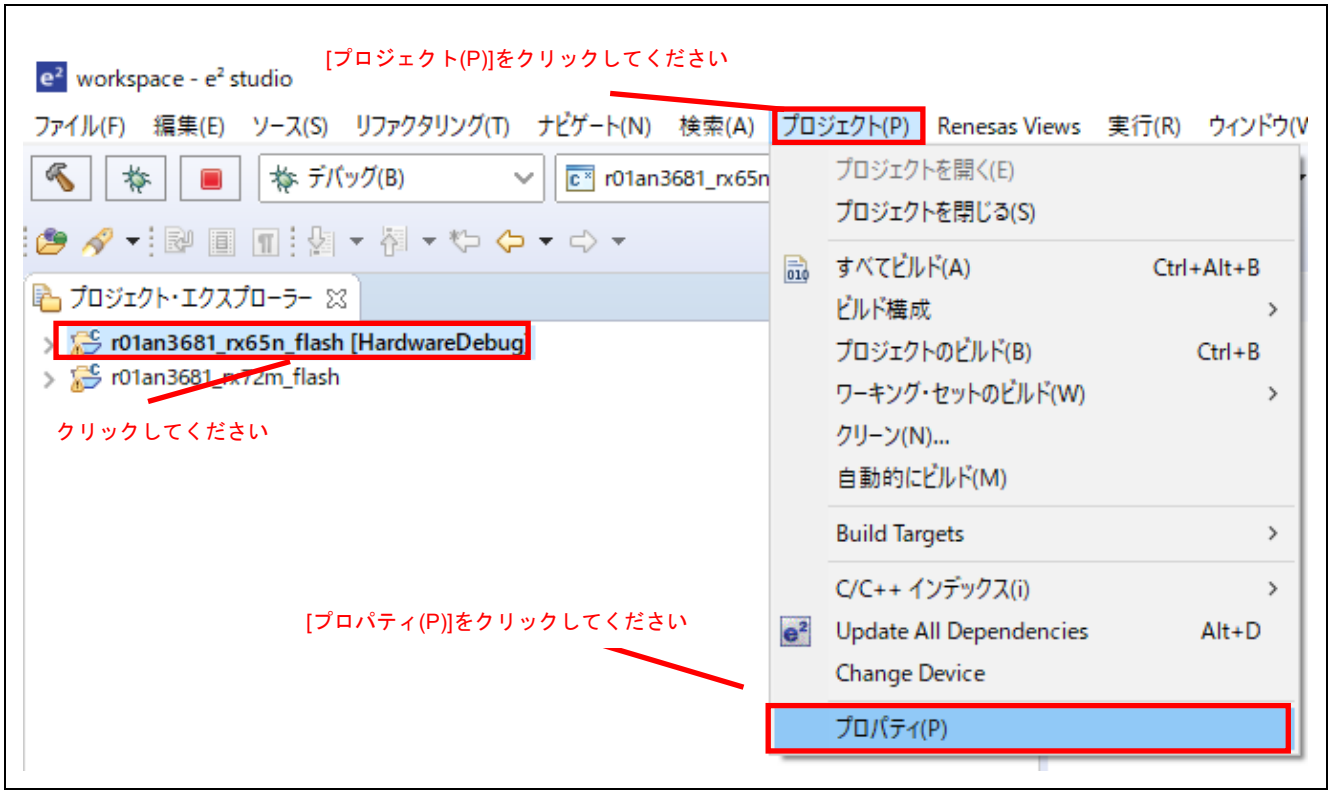


3.4.3　[C/C++ビルド]の設定の変更

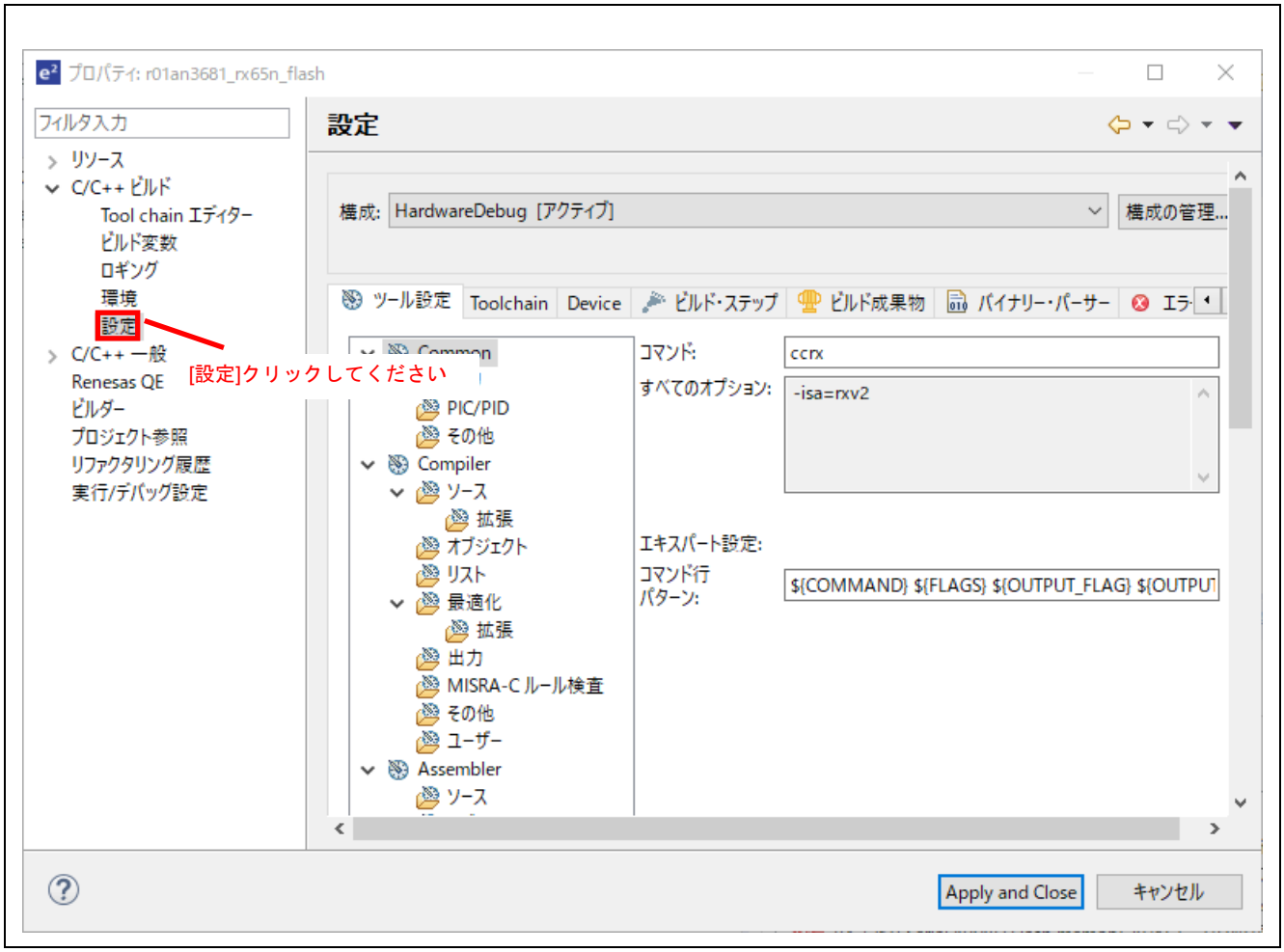
プロジェクトのビルド設定はデフォルト値から、「表 3.1 変更したプロジェクトのビルド設定」に示す内容に変更しています。

プロジェクトの設定を確認する場合は、以下の手順でおこなってください。

1. プロジェクト・エクスプローラーで対象プロジェクトをクリックしてください。
2. [プロジェクト(P)]をクリックしてください。
3. [プロパティ(P)]をクリックしてください。

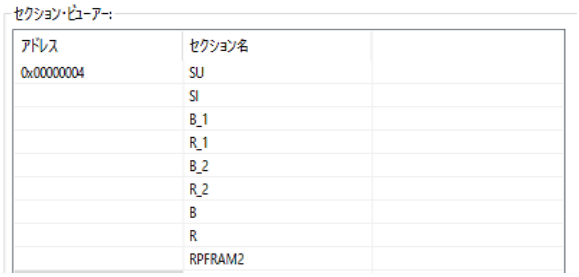
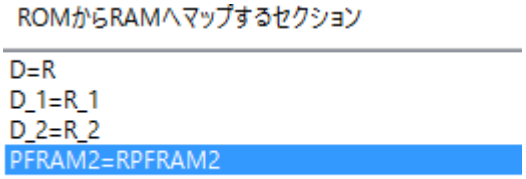
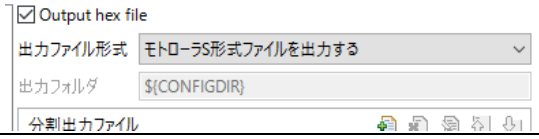


4. [C/C++ビルド]の[設定]をクリックしてください。



5. [ツール設定]タブの内容を「表 3.1 変更したプロジェクトのビルド設定」の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

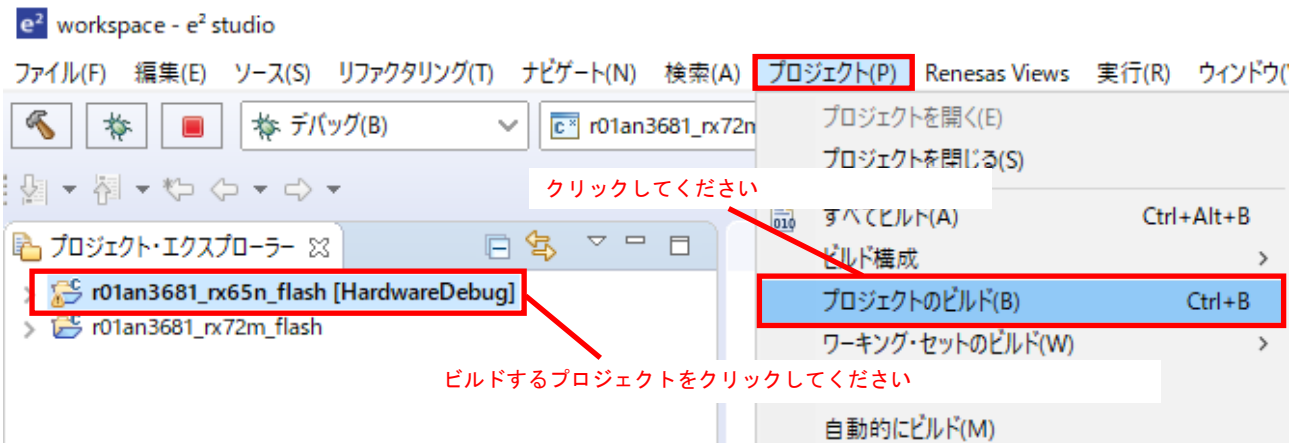
項目	変更内容	説明
Compiler - ソース	[インクルード・ファイルを検索するフォルダ]に src を追加しています。	src 以下のフォルダから src にあるヘッダファイルを使用するために追加しています。
Linker - セクション	RAM 領域に、RPFRAM2 セクションを追加しています。 <設定> 	Flash API が使用する RAM 領域を設定しています。
Linker - セクション - シンボル・ファイル	[ROM から RAM へマップするセクション]に PFRAM2=RPFRAM2 を追加しています。 <設定> 	Flash FIT API で起動バンクを切り替えるためのコードは RAM 上で実行する必要があります。そのため ROM から RAM にマップする設定を追加しています。
Converter -出力	[Output hex file]をチェックしています。 <設定> 	mot ファイルを出力します。

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てください。

1. ビルドするプロジェクトをクリックしてください。
2. [プロジェクト]-[プロジェクトのビルド]をクリックしてください。



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。

```
    -nomessage
rlink   "r01an3681_rx65n_flash.abs" -subcommand="Converterr01an3681_rx65n_flash.tmp"

Renesas Optimizing Linker Completed
'Finished building target:'

'Build complete.'

14:03:04 Build Finished. 0 errors, 0 warnings. (took 21s.932ms)
```

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、評価ボードの準備をしてください。

必要な機器の一覧と構成を「表 4.1　機器構成」に示します。

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-2MB） （Renesas Starter Kit+ for RX72M）	本アプリケーションノートではボードの電源の供給は AC アダプタから行います。
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	環境の例は以下の「図 4.1」を参照してください。

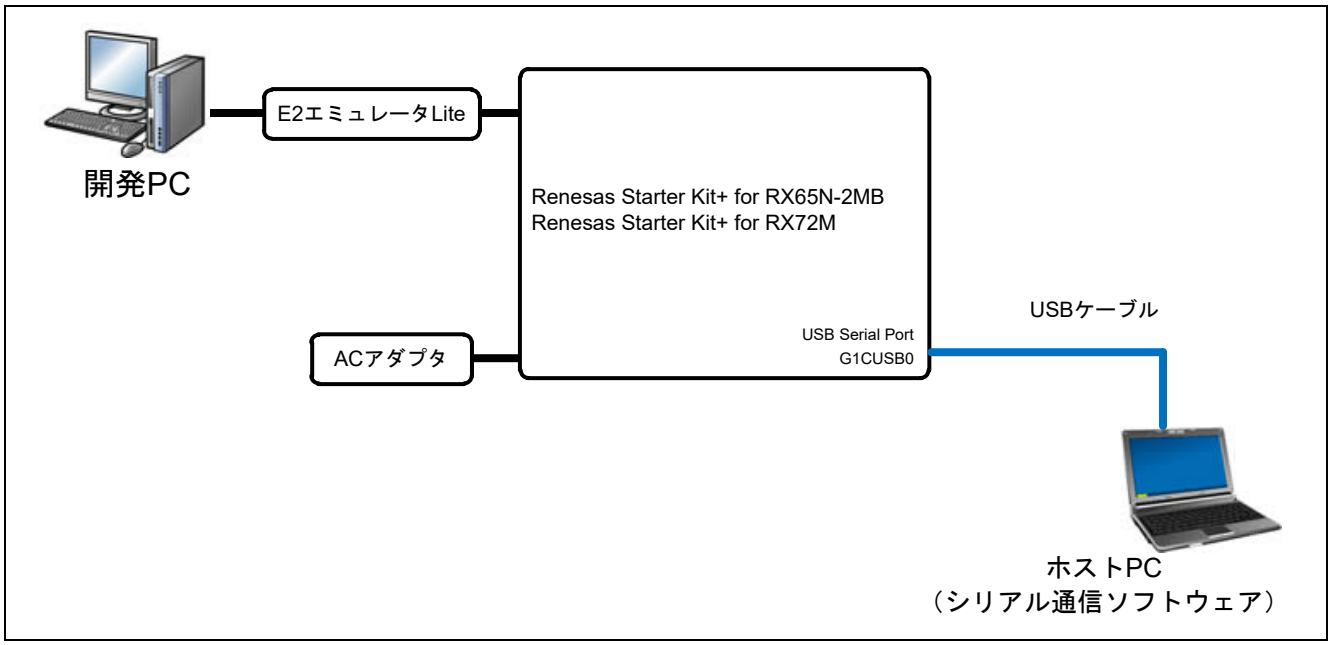


図 4.1　デバッグ構成図

4.2.2　ホスト PC の設定

シリアル通信ソフトウェアの通信仕様を表 表 4.2　通信仕様に示します。シリアル通信ソフトウェアの設定方法はシリアル通信ソフトウェアの説明書をご覧ください。

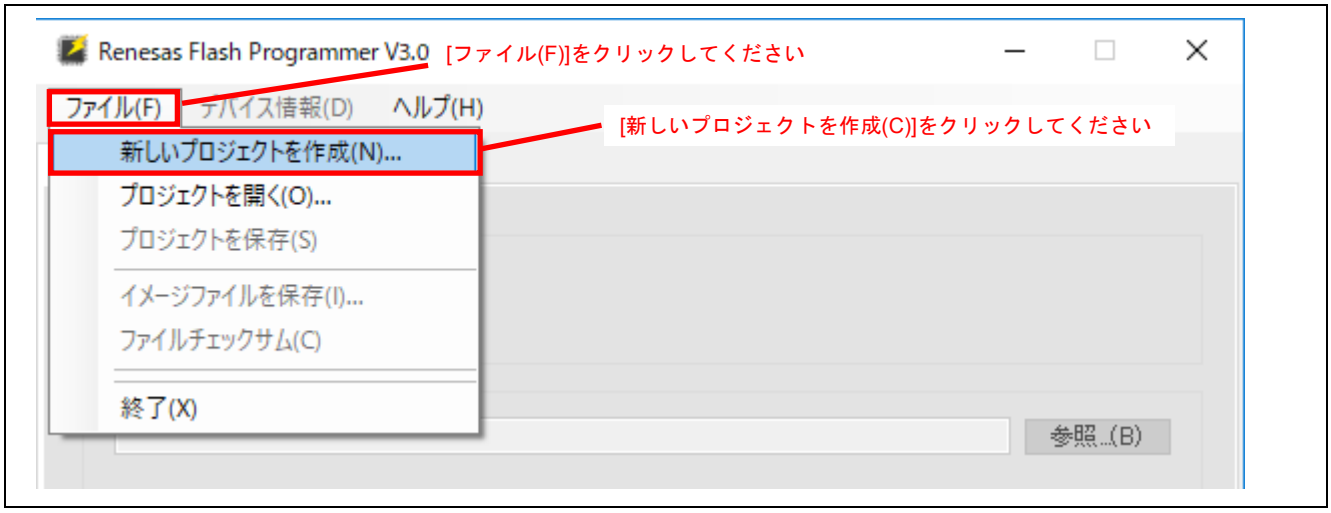
表 4.2　通信仕様

項目	内容
通信方式	調歩同期式通信
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

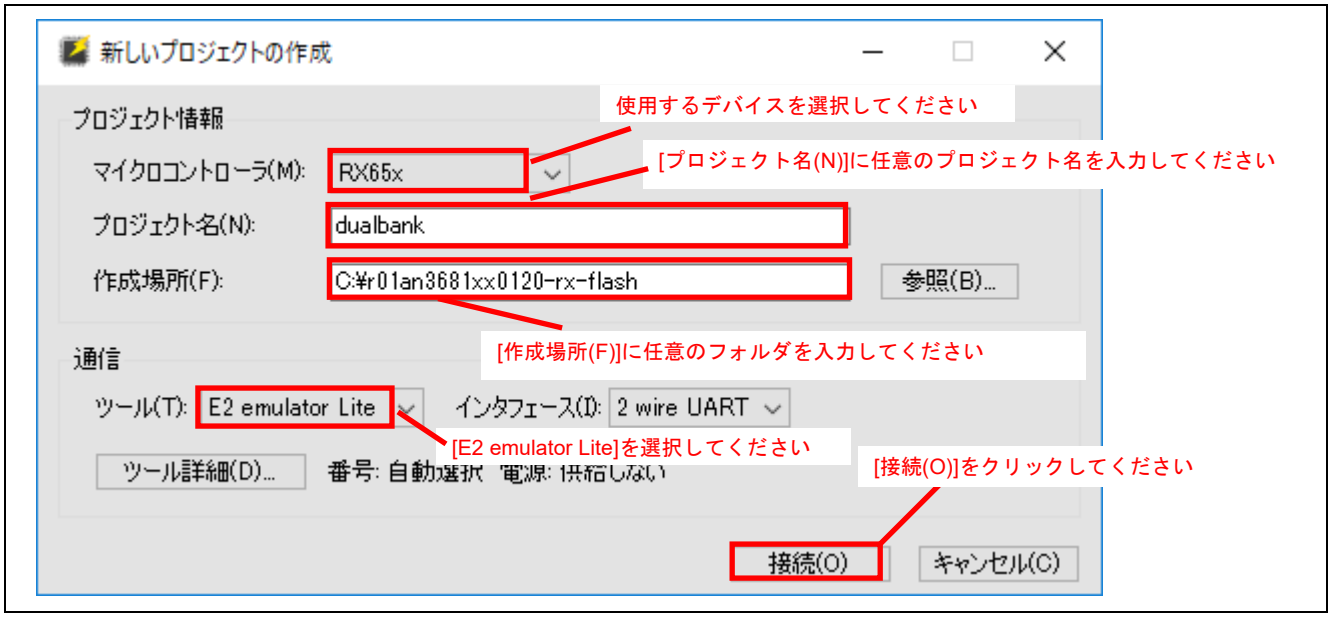
4.2.3　マイコンの事前設定

デュアルバンク機能を使用するためには最初にマイコンをデュアルモードに設定する必要があります。以下の手順でデュアルモードに設定してください。

1. Renesas Flash Programmer を起動してください。
2. [ファイル(F)]をクリックしてください。
3. [新しいプロジェクトを作成(C)]をクリックしてください。

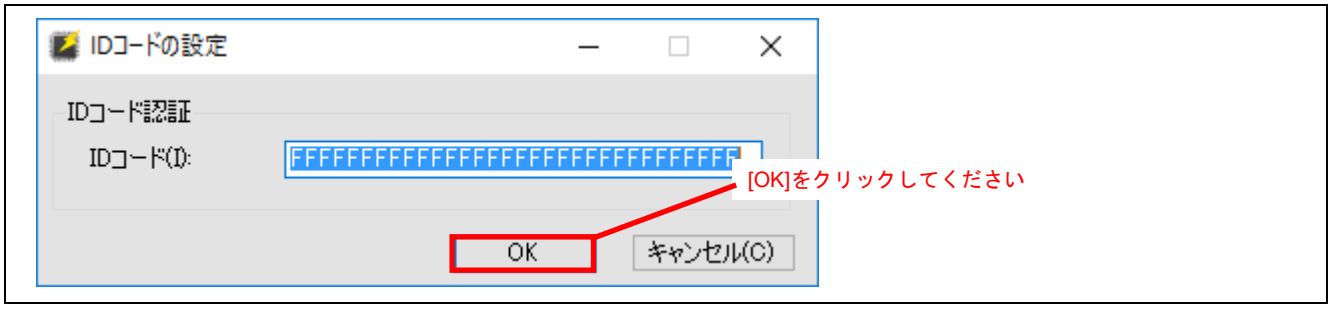


4. [新しいプロジェクトの作成]ダイアログが開きます。
5. [マイクロコントローラ(M):]は使用するデバイスを選択してください。
6. [プロジェクト名(N):]に任意のプロジェクト名を入力してください。
7. [作成場所(F):]に任意のフォルダを入力してください。
8. [ツール(T):]は[E2 emulator Lite]を選択してください。
9. [接続(O)]をクリックしてください。

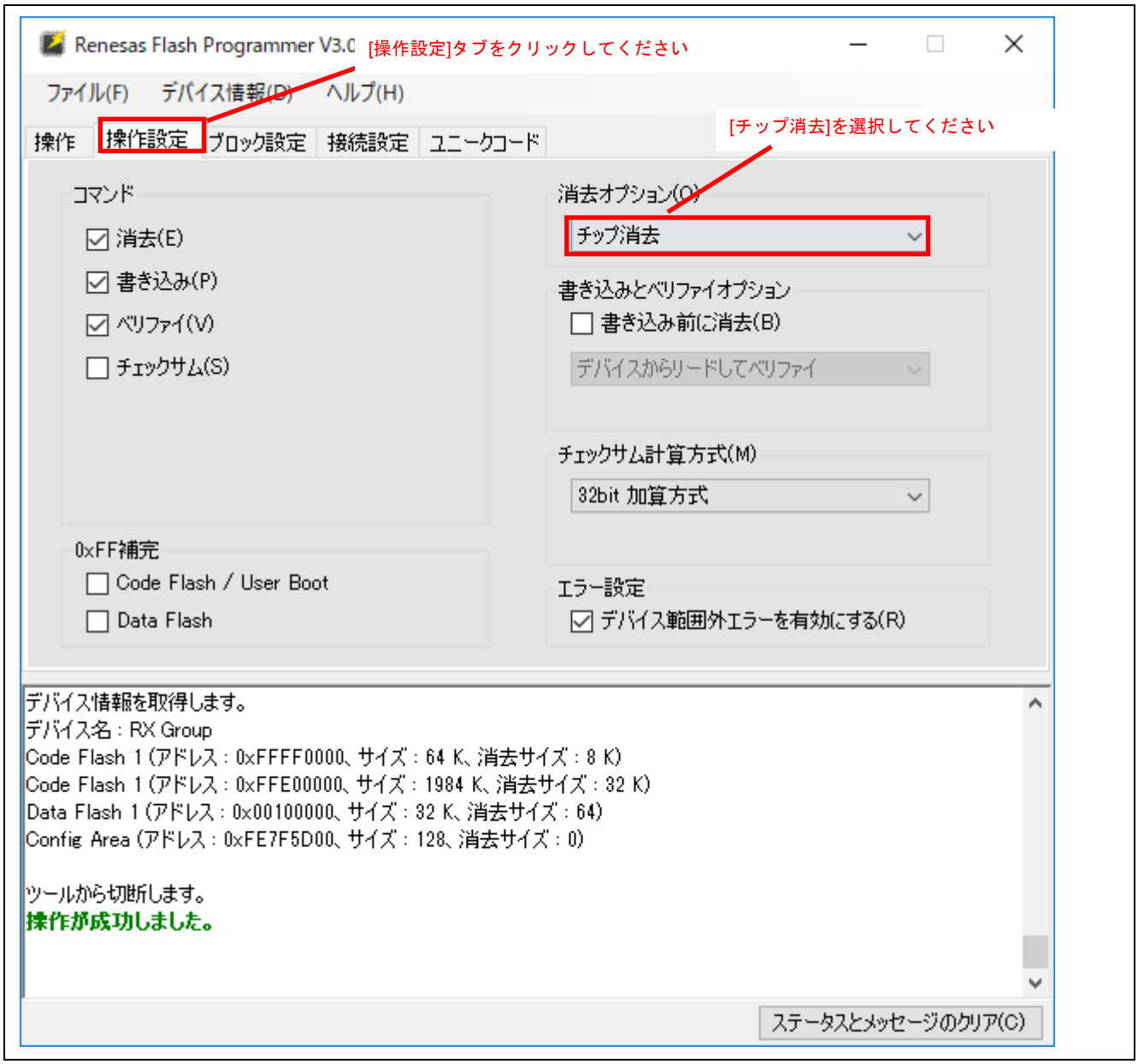


10. [ID コードの設定]ダイアログが開きます。

11. [OK]をクリックしてください。

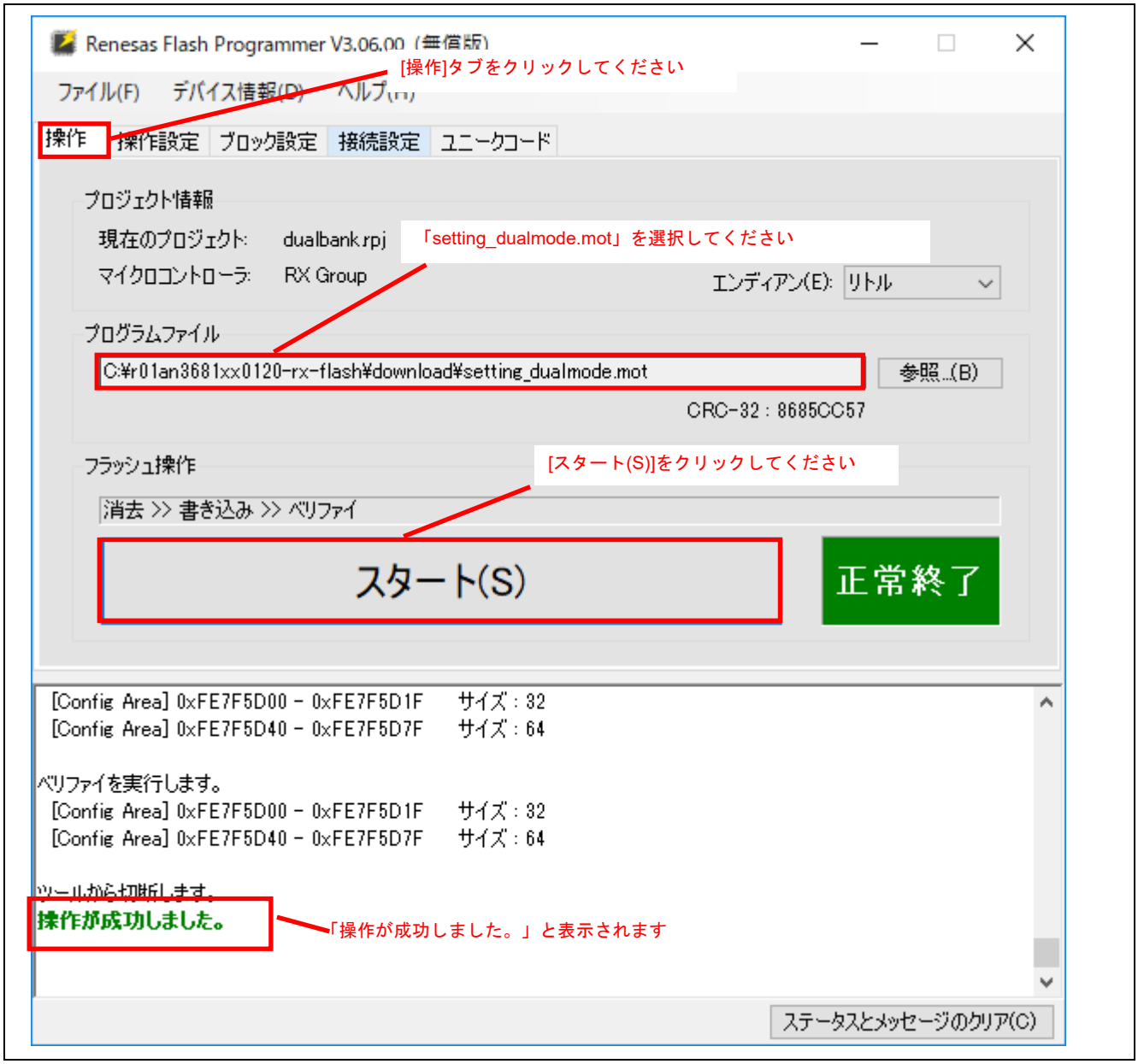


12. [操作設定]タブをクリックしてください。
13. [消去オプション(O)]で[チップ消去]を選択してください。



14. [操作]タブをクリックしてください。

15. [プログラムファイル]に本アプリケーションノートの ZIP ファイルを展開したフォルダにある「download」フォルダの下の「setting_dualmode.mot」ファイルを選択してください。
16. [スタート]をクリックしてください。
17. 「操作が成功しました。」とメッセージが表示されます。これでマイコンがデュアルモードになりました。



Renesas Flash Programmer はデュアルモードとリニアモードで別のプロジェクトが必要になります。リニアモードとして作成したプロジェクトではデュアルモードでは接続できません。Renesas Flash Programmer で再度接続したい場合はプロジェクトを再度作成してください。

4.3　デバッグ構成

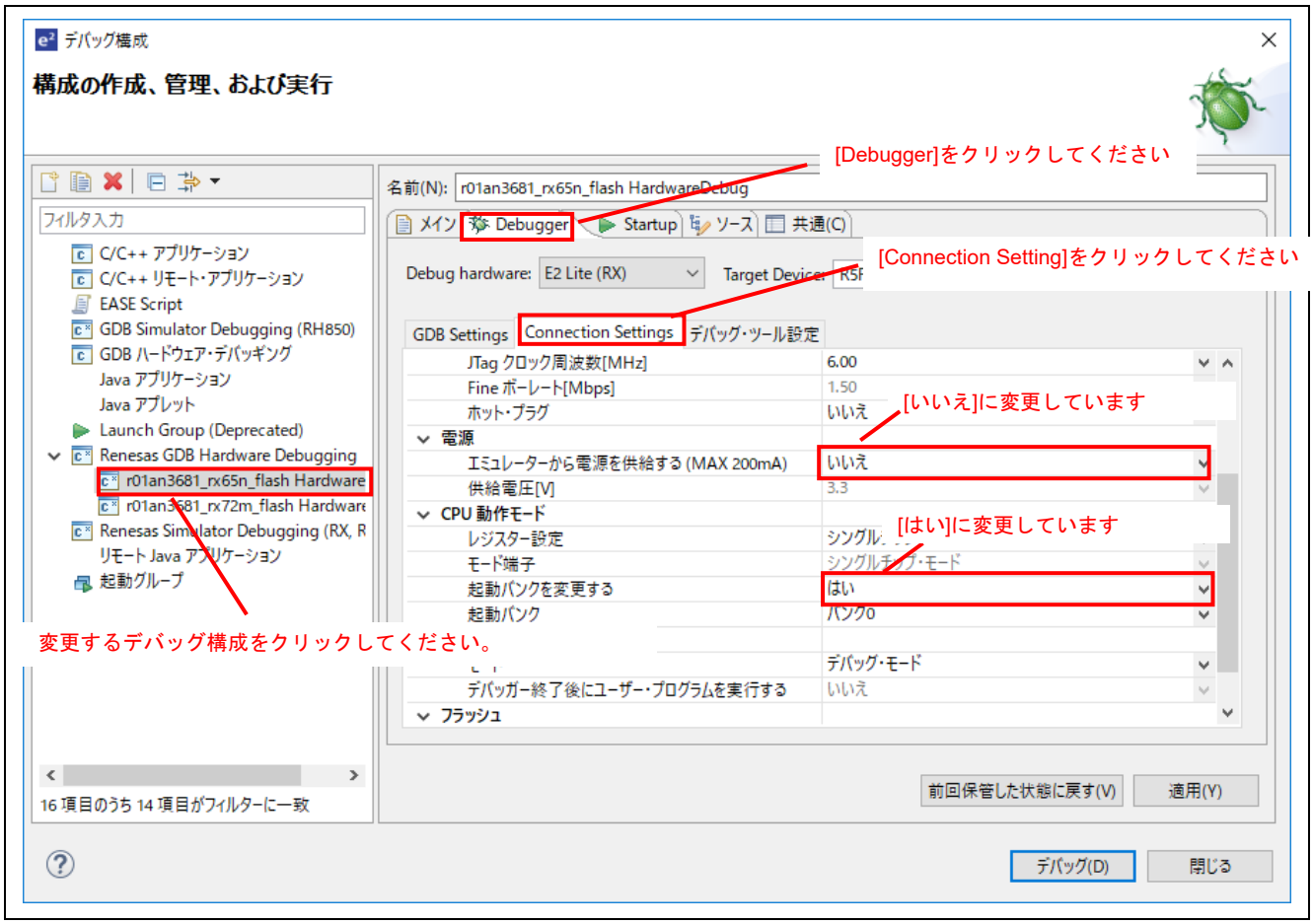
プロジェクトのデバッグ構成はデフォルト値から変更しています。

プロジェクトのデバッグ構成を確認する場合は、以下の手順でおこなってください。

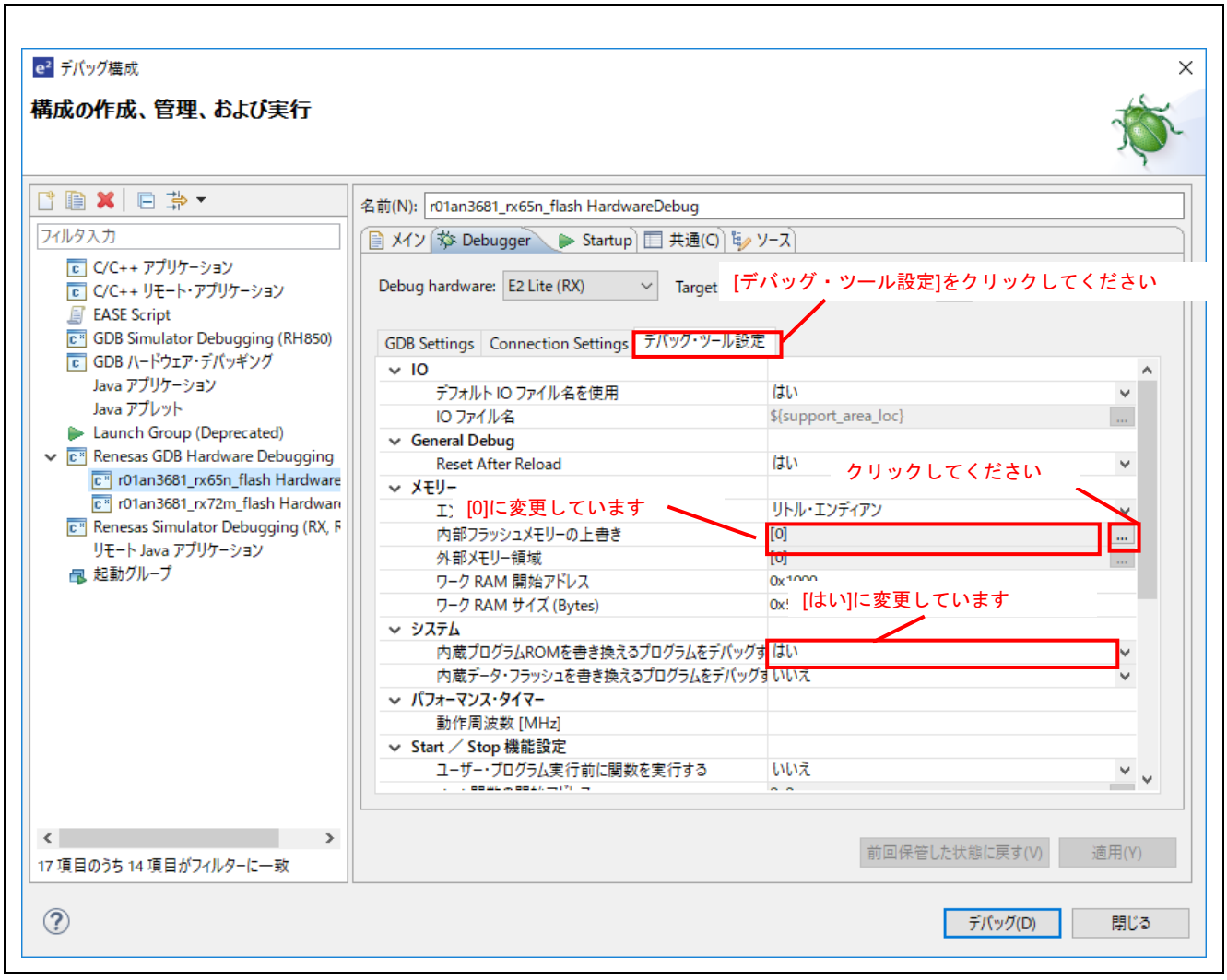
なお、この変更手順は、新規にプロジェクトを構築する場合に参照してください。デバッグ実行を行うには「4.4 デバッグ」に進んでください。

1. e² studio の[実行(R)]メニューをクリックしてください。
2. [デバッグ構成(B)]をクリックしてください。

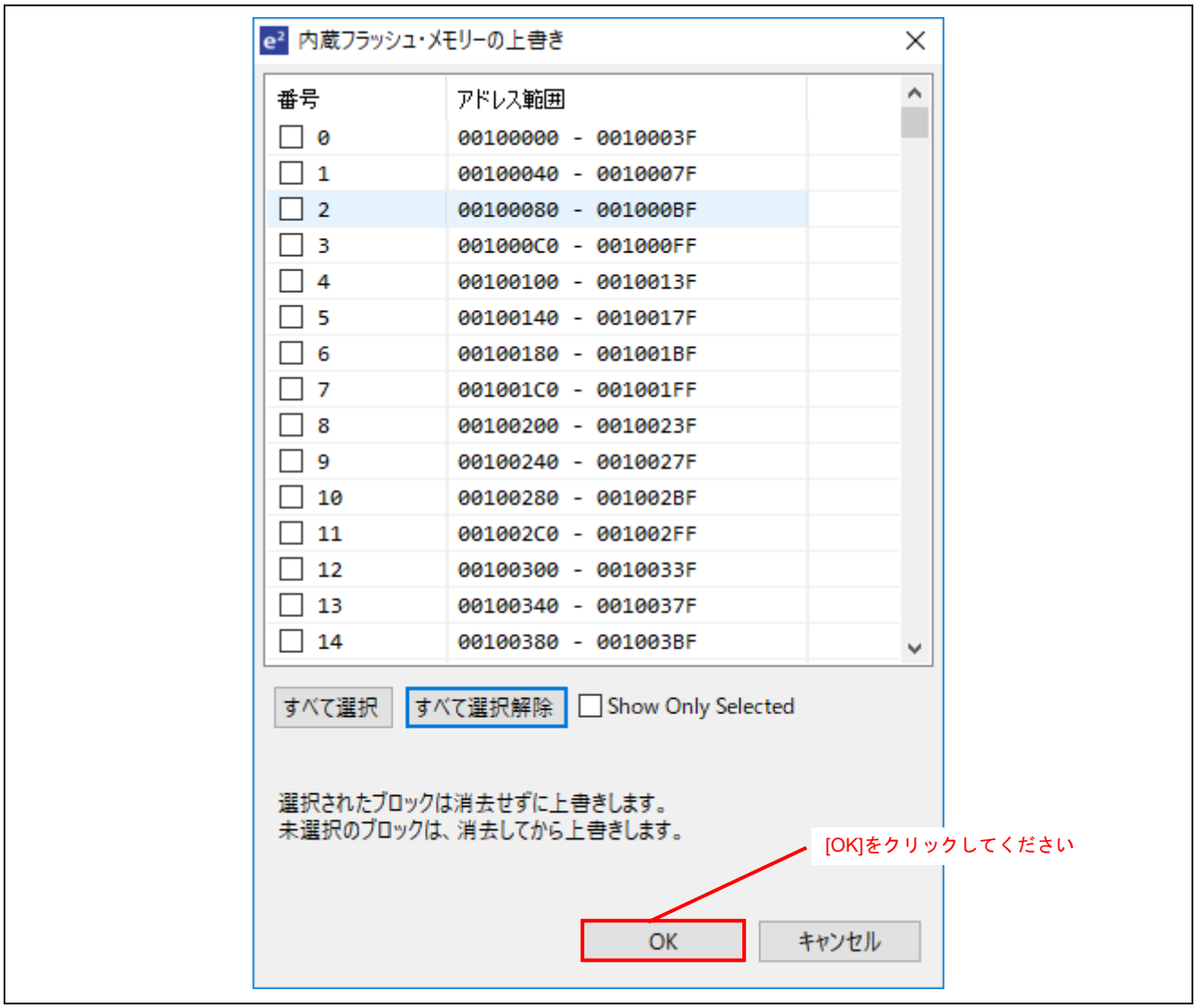
3. [Renesas GDB Hardware Debugging]の変更するデバッグ構成をクリックしてください。
4. [Debugger]タブをクリックしてください。
5. [Connection Setting]タブをクリックしてください。
6. [エミュレータから電力を供給する]を[いいえ]に変更しています。
7. [起動バンクを変更する]を[はい]に変更しています。



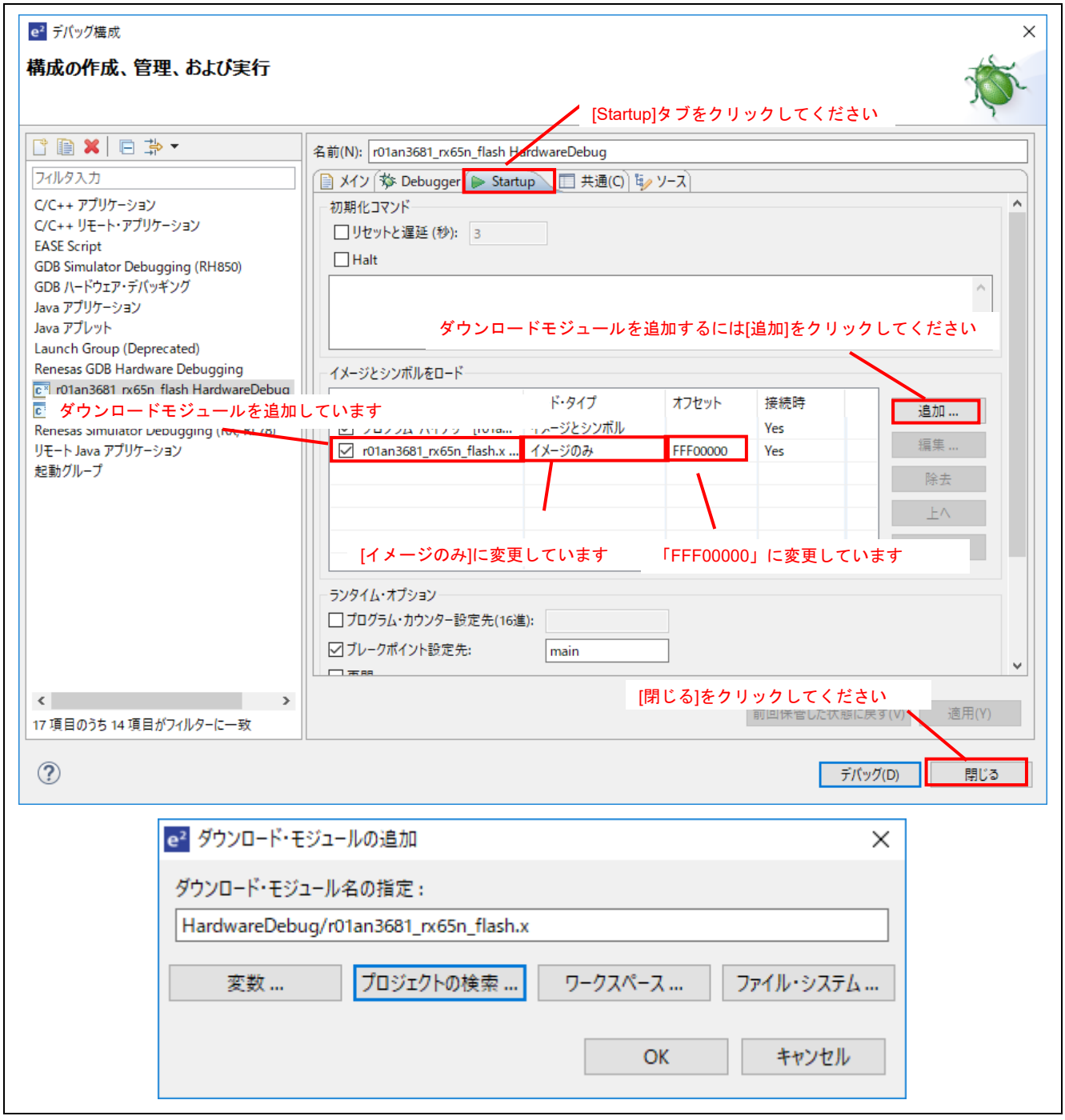
8. [デバッグ・ツール設定]タブをクリックしてください。
9. [内蔵プログラム ROM を書き換えるプログラムをデバッグする]を[はい]に変更しています。
10. RX72M の場合は[パフォーマンス・タイマー]を「240」設定しています。
11. [内蔵フラッシュメモリーの上書き]を[0]に変更しています。ここを変更するには右端のボタンをクリックしてください。



12. 内蔵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ています。[すべて選択解除]をクリックするとチェックが全てはずれます。
13. [OK]をクリックしてください。



14. 起動バンクとは逆側のバンクにロードするイメージを追加しています。[Startup]タブをクリックしてください。
15. ダウンロードモジュールを追加するには[追加]をクリックしてください。[ダウンロード・モジュールの追加]ダイアログが開きます。プロジェクト内のファイルを追加する場合は、[プロジェクトの検索]から追加してください。

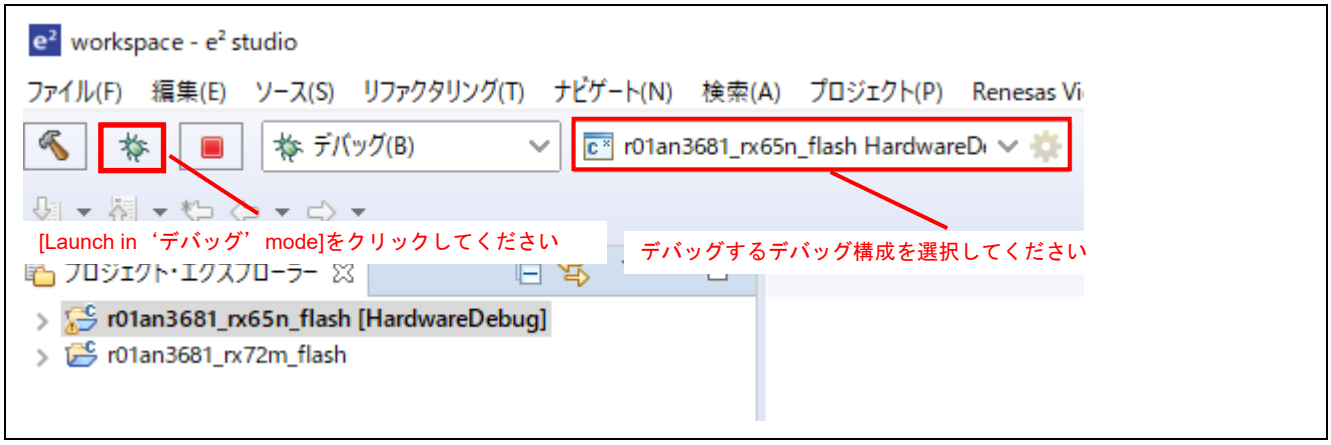


16. HardwareDebug の拡張子が x のファイルを追加しています。
17. [ロード・タイプ]を[イメージのみ]に変更しています。
18. RX65N の場合は[オフセット]を「FFF00000」に変更しています。RX72M の場合は[オフセット]を「FFE00000」に変更しています。(注 1)
19. [閉じる]をクリックしてください。

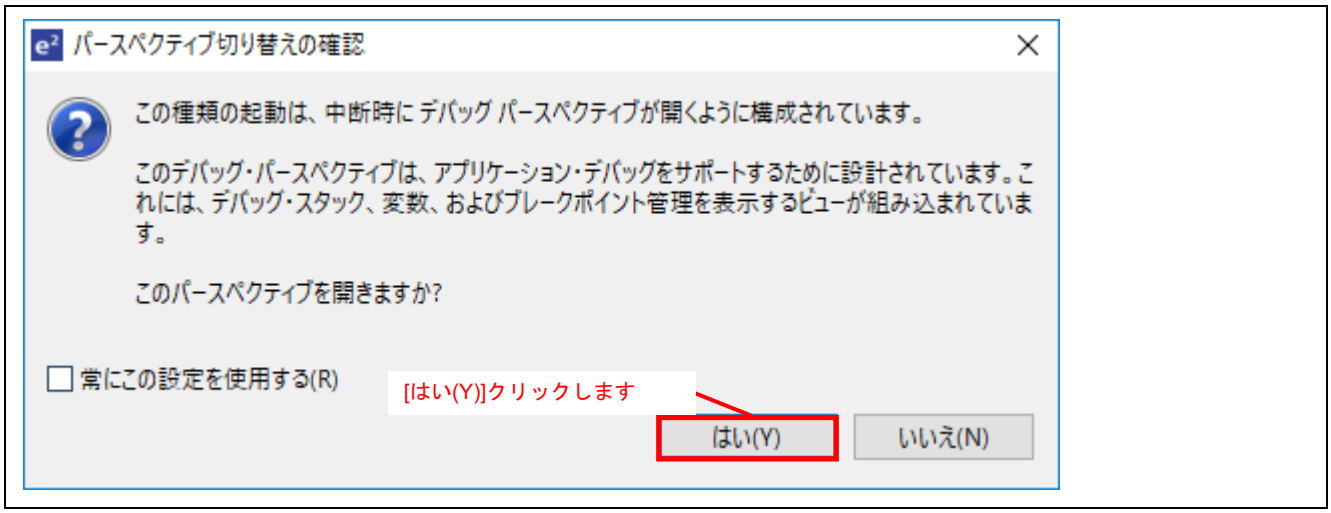
【注 1】ここで指定する値はアドレス値ではなくオフセット値です。RX65N の場合はアドレス値を-100000H オフセットします。マイナス値を入力できないため 2 の補数で表現した値である FFF00000H を入力します。FFF00000H オフセットすると起動バンクのアドレス値が逆側のバンクのアドレス値になります。RX72M の場合はアドレス値を-200000H オフセットするので FFE00000H を入力します。

4.4　デバッグ

1. デバッグするデバッグ構成を選択してください。
2. [Launch in‘デバッグ’mode]をクリックしてください。

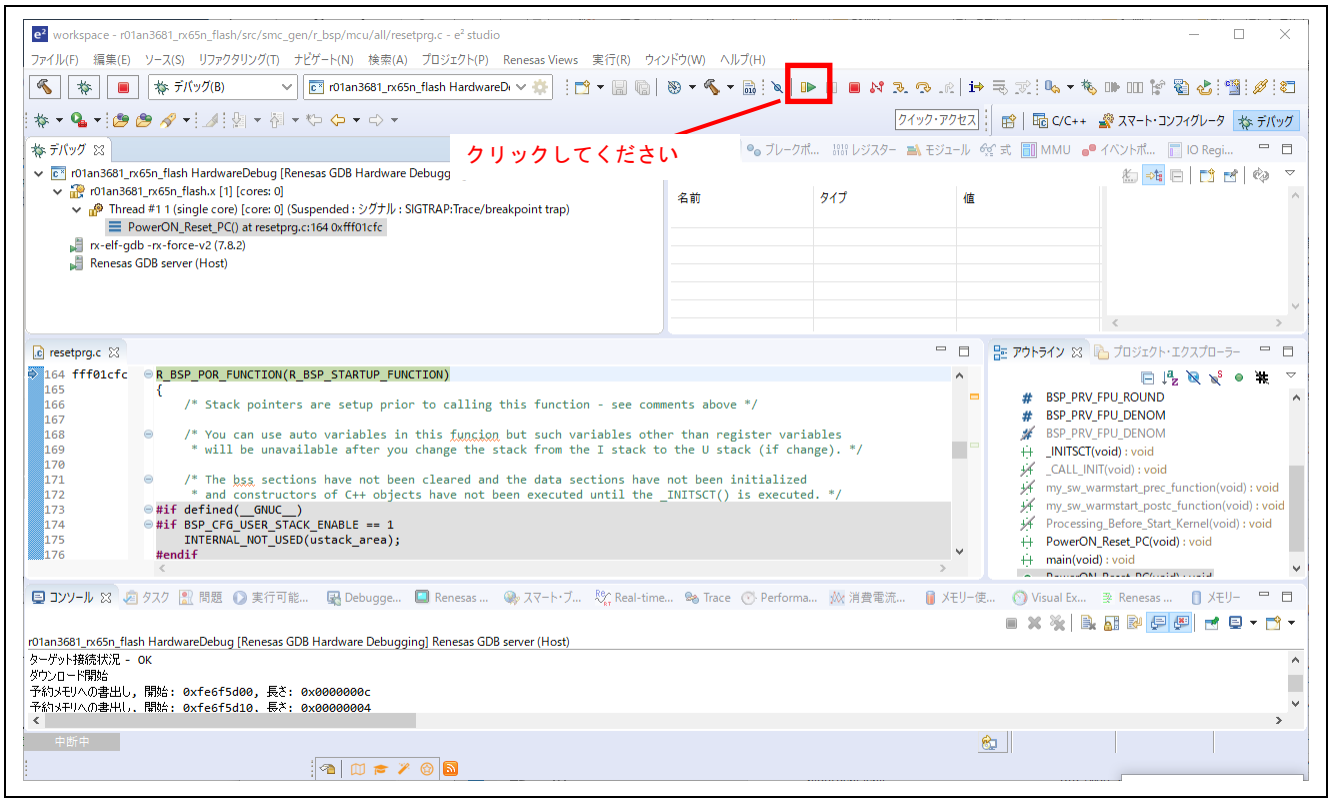


3. 以下のメッセージが表示されたら、[はい(Y)]をクリックしてください。

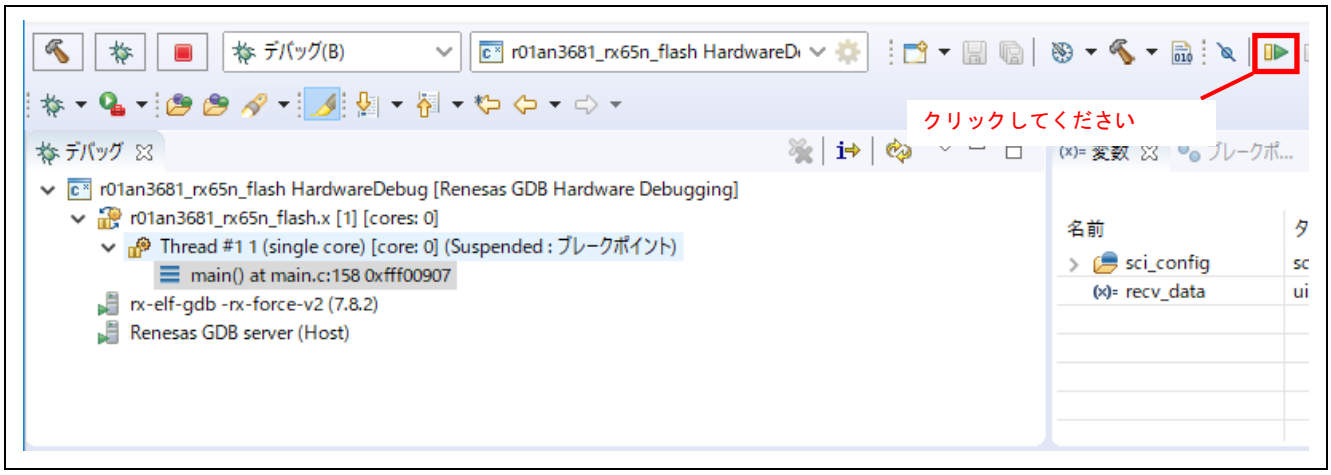


4. ロードモジュールのダウンロードが完了すると、[デバッグ]パースペクティブが開きます。

5. ツールバーの[再開]をクリックしてください。プログラムが実行され、main 関数の先頭でブレークします。



6. 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]をクリックしてください。



7. シリアル通信ソフトの画面に以下のメッセージが表示されます。

```
Dualbank firmware update menu ver1.20A
1...Update
2...Changing the Startup Bank and Reset
```

4.5　アップデート確認用 mot ファイル

本アプリケーションノートではアップデート確認用 mot ファイルを用意しています。この mot ファイルにアップデートされるとメニュー表示の文字列が「ver1.20B」に変わります。

```
Dualbank firmware update menu ver1.20B
1...Update
2...Changing the Startup Bank and Reset
```

mot ファイルは本アプリケーションノートの ZIP ファイルの中にある「download」フォルダに格納しています。デバイスとエンディアンの組み合わせで 4 種類のアップデート確認用 mot ファイルがあります。

5.　サンプルプログラム概要

5.1　動作概要

サンプルプログラムは、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（mot ファイル）をマイコンに転送し、コードフラッシュメモリに書き込みます。ファームウェア自体はデュアルバンクの起動バンク側で動作します。アドレス範囲が FFF0 0000h～FFFF FFFFh の mot ファイルを起動バンクとは逆側のバンクにあたる FFE0 0000h～FFEF FFFFh の範囲に書き込みます。正常に書き込めた事を確認後、起動バンクを変更してソフトウェアリセットを行うことによって安全にファームウェアを更新することができます。アドレス範囲が FFF0 0000h～FFFF FFFFh 以外の場合は書き込みを行いません。

なお FFF0 0000h、FFE0 0000h、FFEF FFFFh は RX65N-2MB の場合の値です。RX72M の ROM 容量 4MB のデバイスではそれぞれ FFE0 0000h、FFC0 0000h、FFDF FFFFh になります。図 5.1～図 5.6 も同様です。

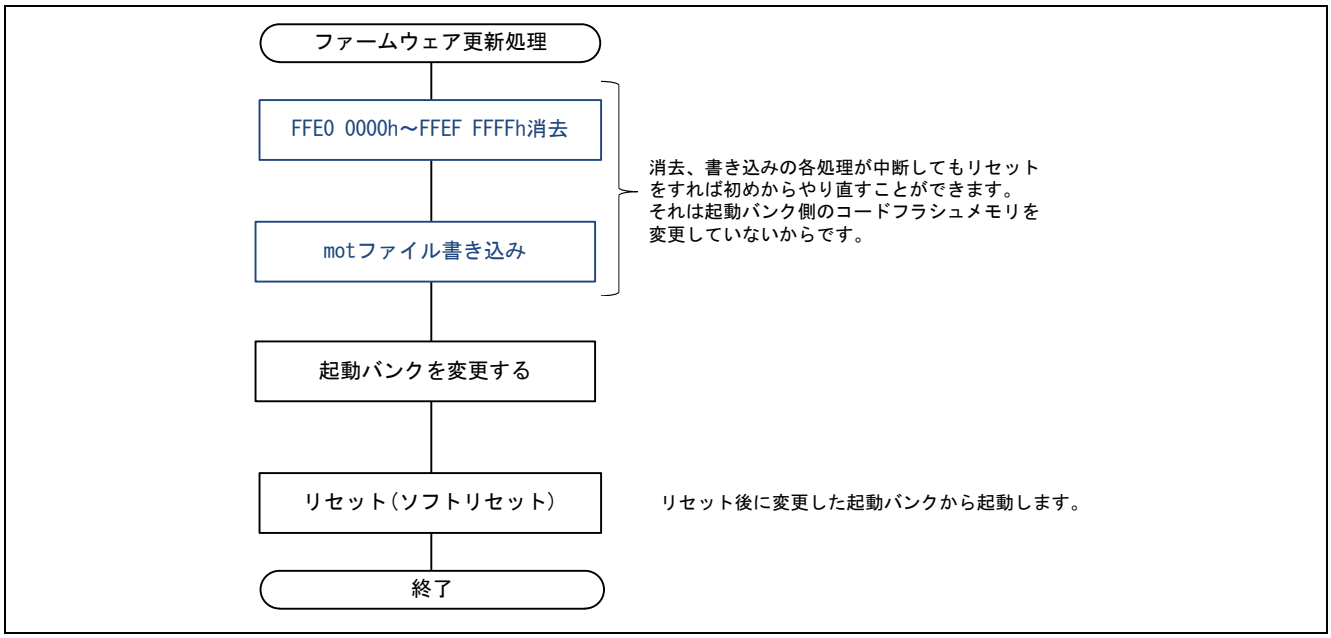


図 5.1　ファームウェアの更新処理

5.1.1　ファームウェアの更新

サンプルプログラムを使用してファームウェアを更新する動作のフローを以下に示します。

1. サンプルプログラムを起動します。サンプルプログラムはホスト PC のシリアル通信ソフトウェアにメニューを表示します。

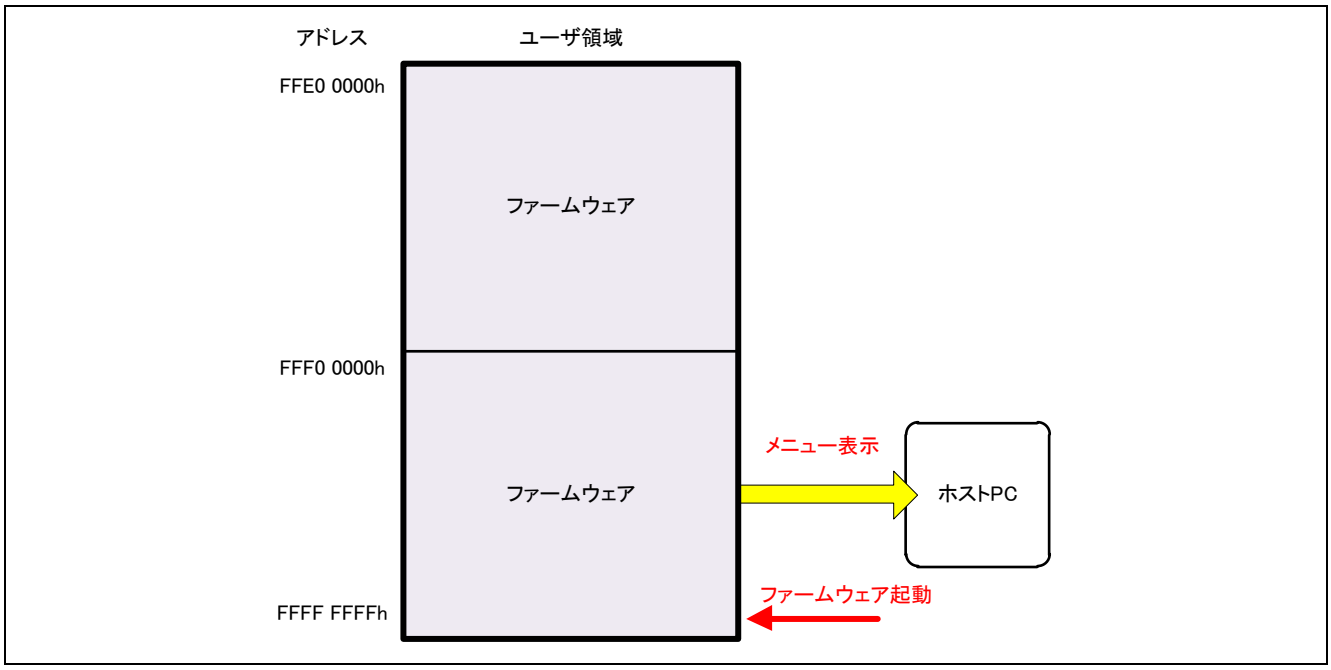


図 5.2　サンプルプログラム起動

2. Update コマンドを実行するとサンプルプログラムは起動バンクと逆側のバンクのコードフラッシュメモリを消去します。

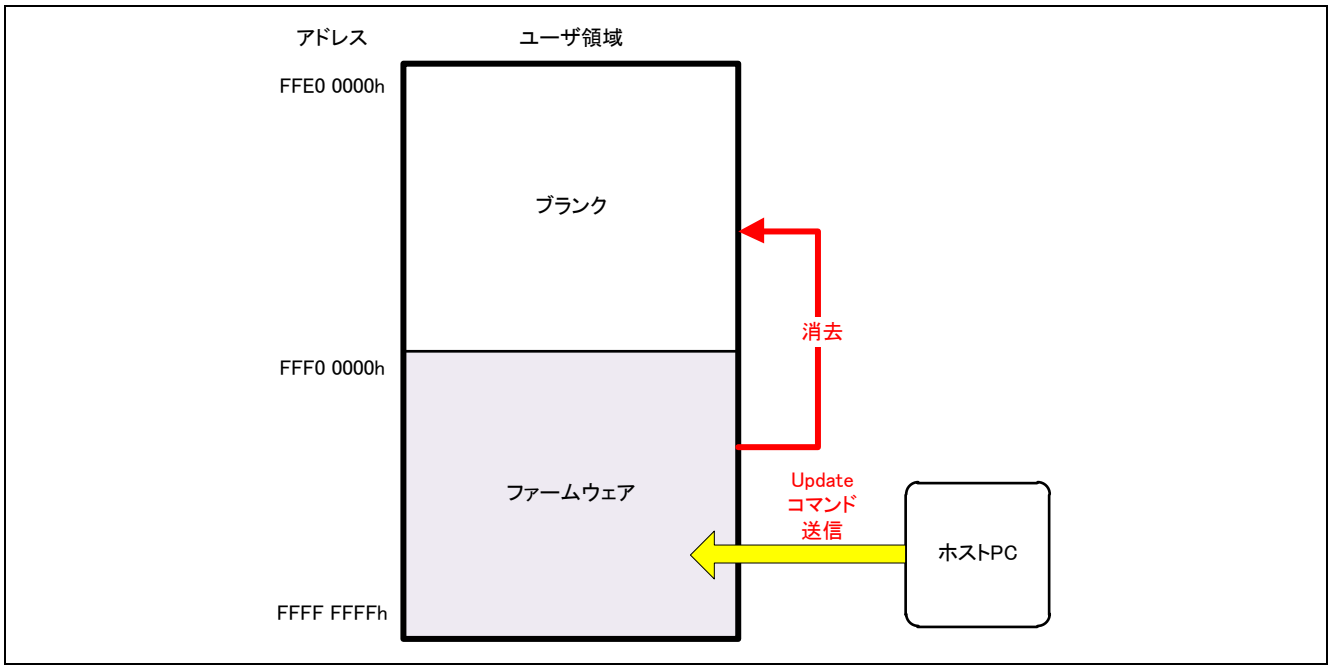


図 5.3　コードフラッシュメモリ消去

3. シリアル通信ソフトウェアを使用して、ファームウェアを送信します。サンプルプログラムは、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みます。

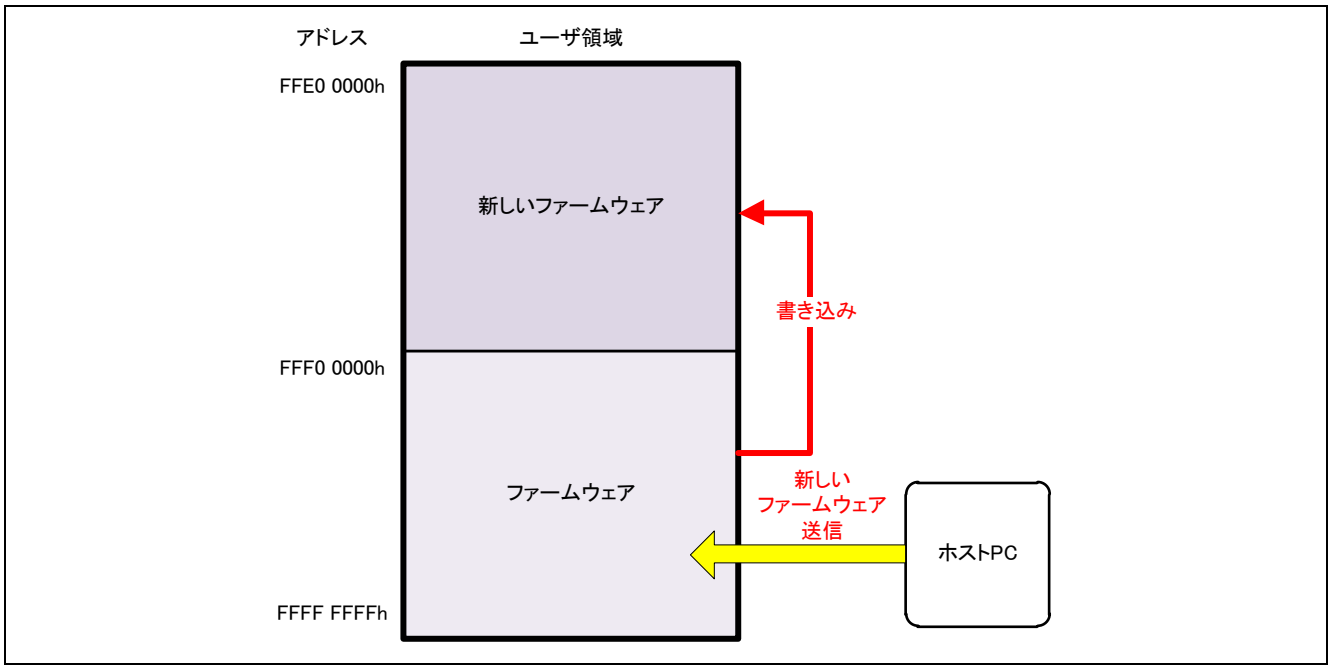


図 5.4　新しいファームウェアの書き込み

4. コードフラッシュメモリに新しいファームウェアの書き込みが完了するとサンプルプログラムはホスト PC のシリアル通信ソフトウェアにメニューを表示します。

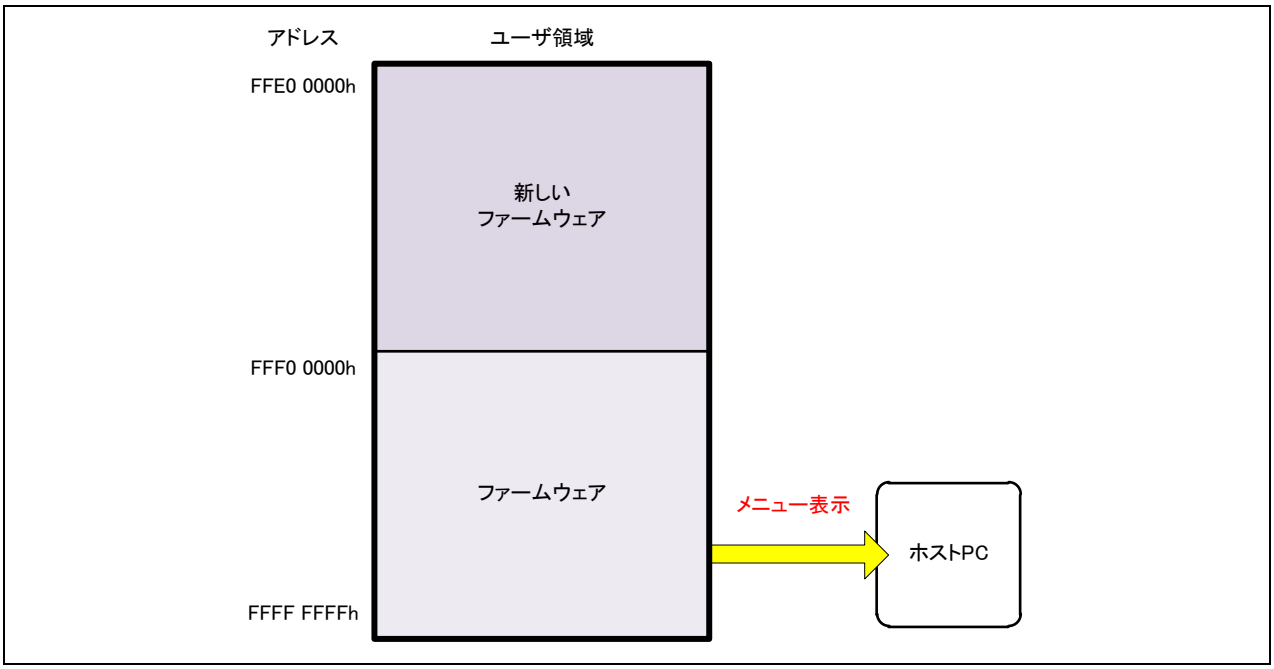


図 5.5　新しいファームウェアの書き込み完了

5. Changing the Startup Bank and Reset コマンドを実行するとサンプルプログラムは、オプション設定メモリのバンク選択レジスタを書き換えたのちソフトウェアリセットを実行します。リセット後に起動バンクが切り替わり、新しいファームウェアが起動します。

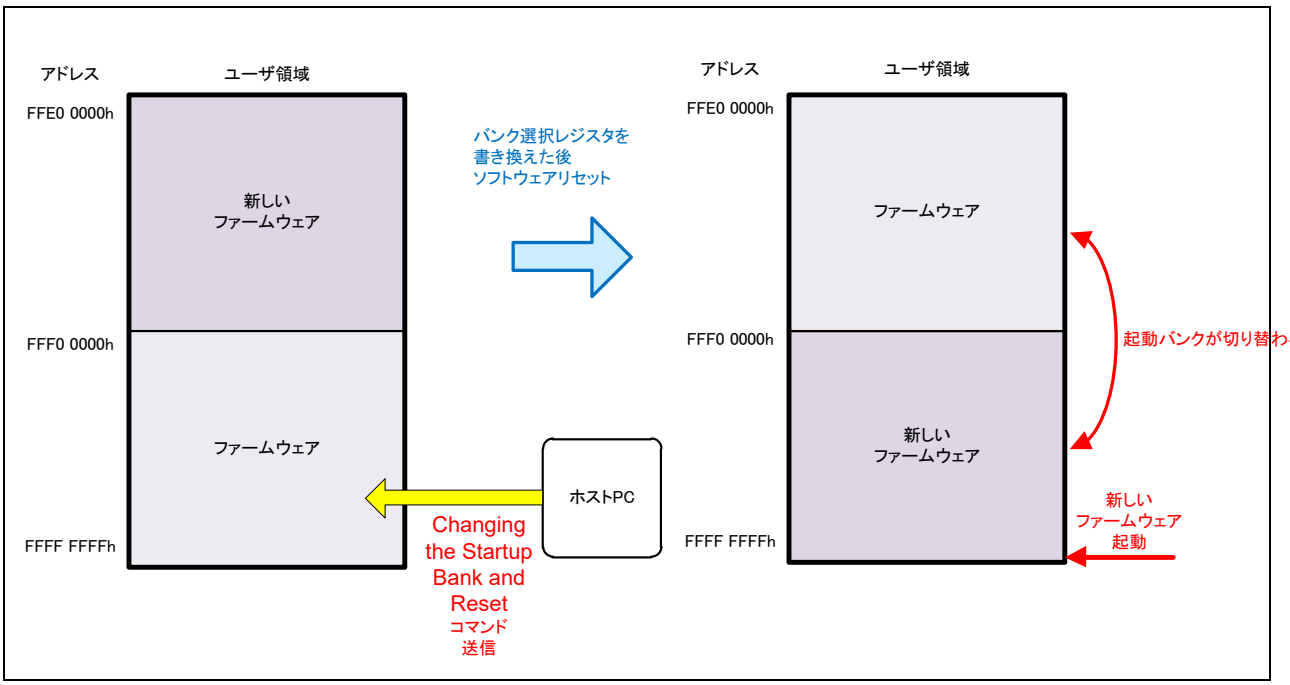


図 5.6　ソフトウェアリセットと新しいファームウェアの起動

5.2　概略フローと画面出力

サンプルプログラムはシリアル通信を使用して、ホスト PC のシリアル通信ソフトウェアにメッセージを出力します。また、シリアル通信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。

5.2.1　メイン処理

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 のシリアル通信ソフトウェアにメニューを表示します。その後、シリアル通信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 5.7 にメイン処理の概略フロー示します。

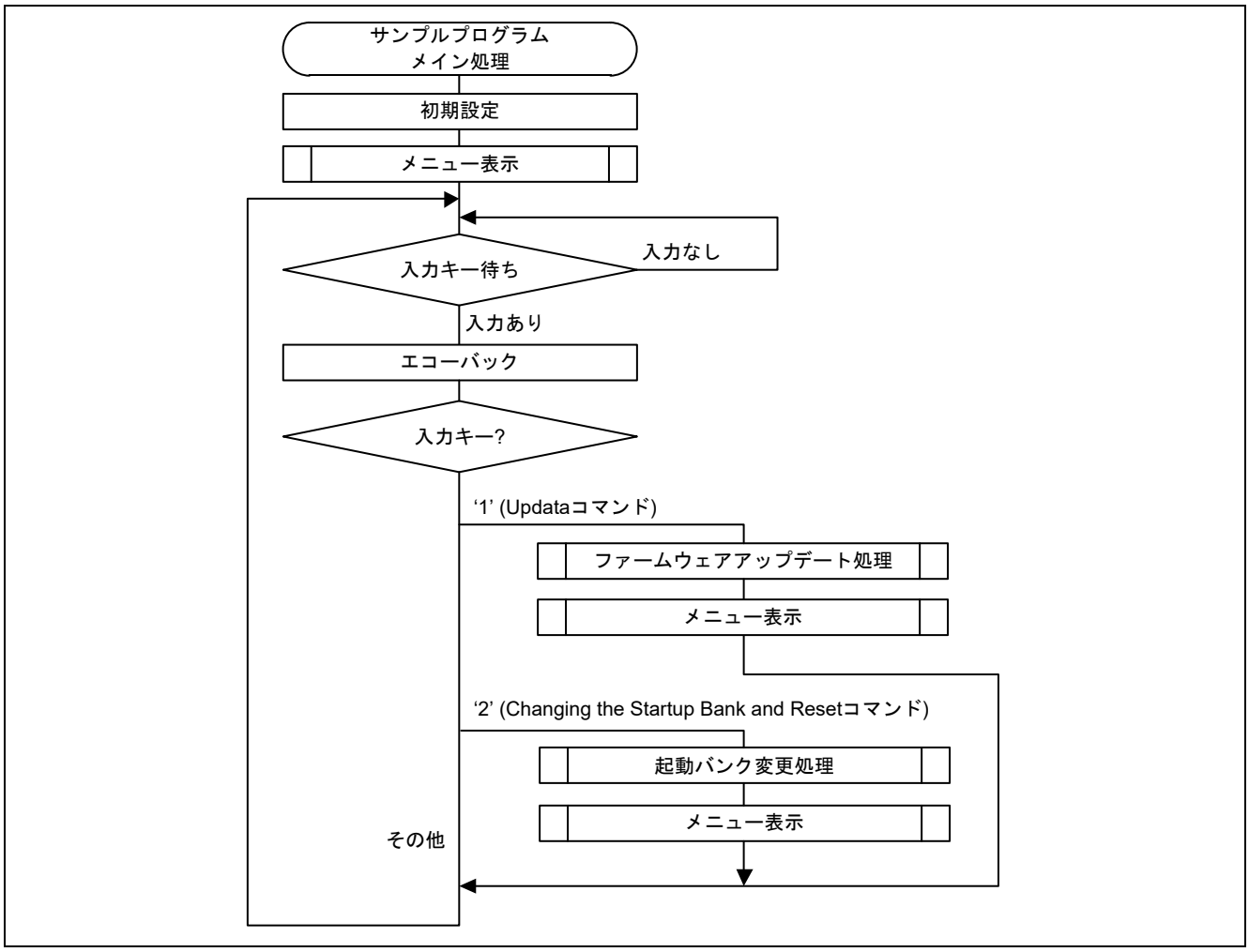


図 5.7　メイン処理の概略フロー

(2)　シリアル通信ソフトウェア画面出力

サンプルプログラムは起動すると、シリアル通信ソフトウェアにメニューを表示します。

```
Dualbank firmware update menu ver1.20A
1...Update
2...Changing the Startup Bank and Reset
```

シリアル通信ソフトウェア上で 1 を入力すればファームウェアアップデート処理を行います。2 を入力すると起動バンク変更処理を行います。

5.2.2　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’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ファームウェアを受信し、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 5.8 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。

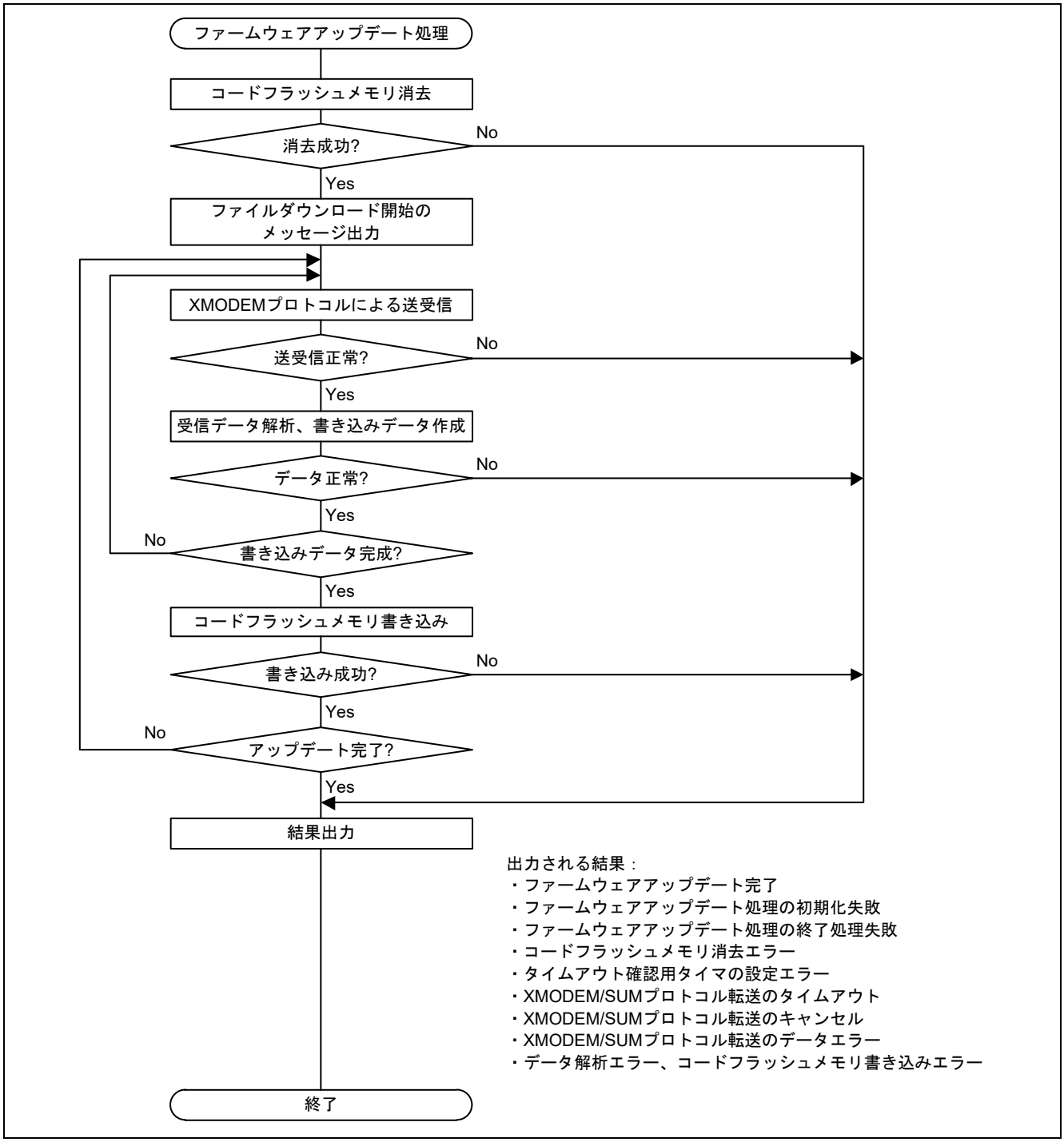


図 5.8　ファームウェアアップデート処理の概略フロー

(2)　シリアル通信ソフトウェア画面出力

1.　ファイルダウンロード開始

FFE0 0000h～FFEF FFFFh の範囲のコードフラッシュメモリを消去します。正常終了したらダウンロード開始のメッセージを出力し XMODEM/SUM プロトコルの受信待ち処理を開始します。

Ready

シリアル通信ソフトウェアから XMODEM/SUM プロトコルでファームウェアを送信してください。

2.　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を完了すると、以下のメッセージを出力してファームウェアアップデート処理を終了します。

Finish

3.　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は、その内容に応じて以下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
Erase error	:コードフラッシュメモリ消去エラー
CMT error	:タイムアウト確認用タイマの設定エラー
Timeout	:XMODEM/SUM プロトコル転送のタイムアウト
Received CAN	:XMODEM/SUM プロトコル転送のキャンセル
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

5.2.3　起動バンク変更

メイン処理において、‘2’を入力すると起動バンク変更処理に移行します。起動バンクを切り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新しいファームウェアが起動します。

(1)　概略フロー

図 5.9 に起動バンク変更処理の概略フローを示します。

オプション設定メモリは BGO 動作の書き換え対象領域ではありません。そのためオプション設定メモリ書き換え中はコードフラッシュメモリをリード出来ません。割り込みベクタはコードフラッシュメモリ上に配置しているため起動バンク切り替え中は割り込みを禁止にします。割り込みを禁止したくない場合は割り込みベクタを RAM に配置してください。

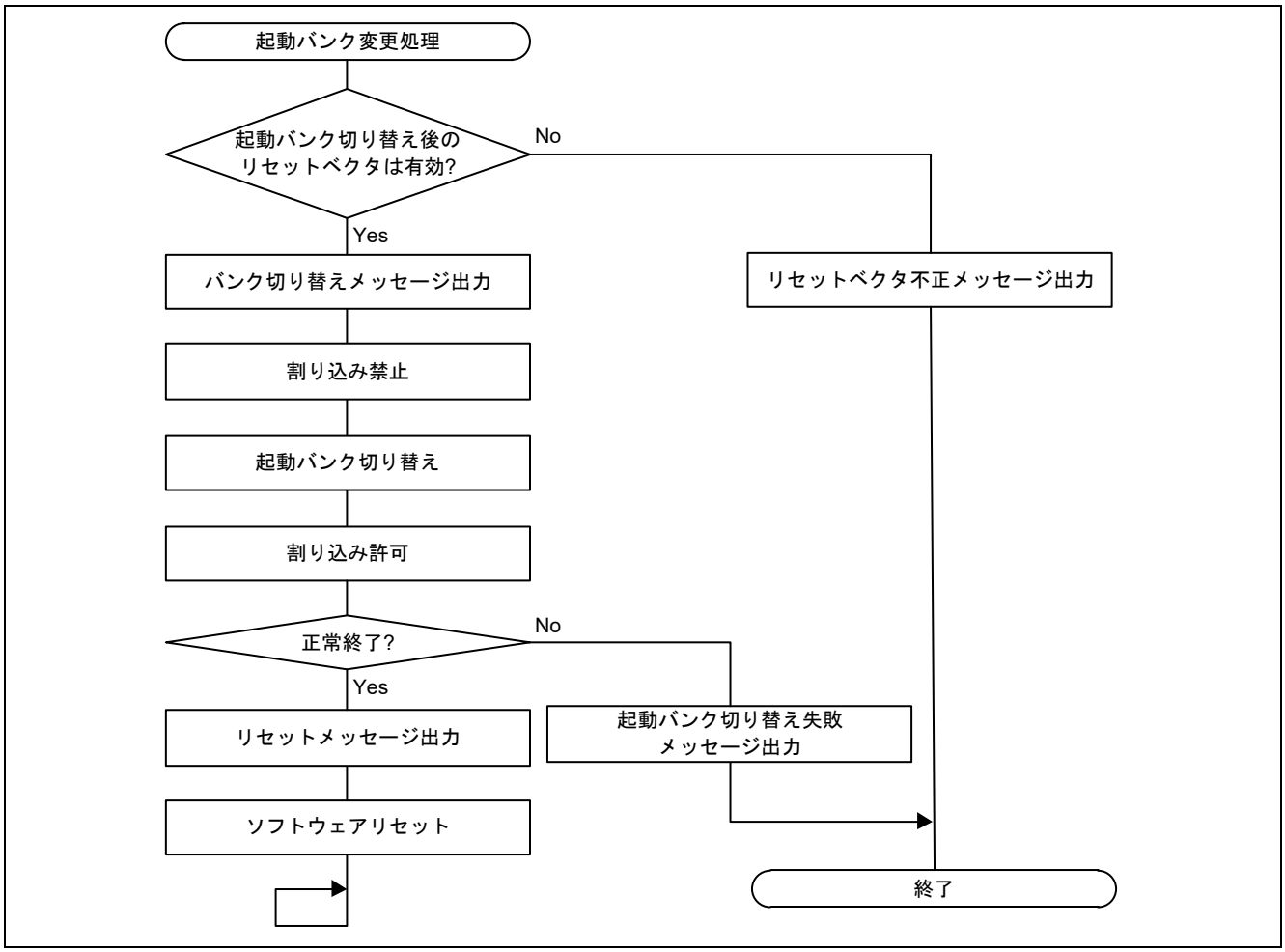


図 5.9　起動バンク変更処理の概略フロー

(2)　シリアル通信ソフトウェア画面出力

1.　起動バンク切り替え
以下のメッセージを出力し、起動バンクを切り替えます。

```
Changing the Startup Bank
```

2.　ソフトウェアリセット
起動バンクの変更が成功した場合、以下のメッセージを出力しソフトウェアリセットを行います。

```
Reset
```

本アプリケーションノートに同梱した各デバイス用アップデートファイルでアップデートを行うと以下のメニュー画面が表示されます。
バージョンが 1.20A から 1.20B に変更されました。

```
Dualbank firmware update menu ver1.20B
1...Update
2...Changing the Startup Bank and Reset
```

3.　エラー出力
起動バンク変更中にエラーが発生した場合は、その内容に応じて以下のメッセージを出力します。

```
Reset vector is invalid   :リセットベクタ不正
Flash error               :起動バンク切り替え失敗
```

5.3　サンプルプログラム詳細

5.3.1　ファイル構成

表 5.1 にサンプルプログラムで使用するファイル、表 5.2 にサンプルプログラムで使用する標準ヘッダ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 5.1　サンプル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 5.2　サンプルプログラムで使用する標準ヘッダファイル

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	記憶領域管理等の 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、複写等を行うライブラリです。

5.3.2　定数一覧

表 5.3～表 5.5 にサンプルプログラムで使用する定数を示します。

表 5.3　サンプルプログラムで使用する定数(main.c)

定数名	設定値	内容
SCI_CH	((uint8_t)SCI_CH8)	RX65N の場合に使用する SCI のチャネル
	((uint8_t)SCI_CH6)	RX72M の場合に使用する SCI のチャネル
TX_MAX_SIZE	SCI_CFG_CH8_TX_BUFSIZ	SCI8 を使用する場合の送信バッファサイズ
	SCI_CFG_CH6_TX_BUFSIZ	SCI6 を使用する場合の送信バッファサイズ
RECV_BYTE_SIZE	((uint16_t)1)	受信用 1 バイトのサイズ
SEND_BYTE_SIZE	((uint16_t)1)	送信用 1 バイトのサイズ
CHARACTER_RETURN	((uint8_t)'r')	改行コード
COMMAND_UPDATE	((uint8_t)'1')	Update コマンド用文字コード
COMMAND_CHANGE_BANK	((uint8_t)'2')	Changing the Startup Bank and Reset コマンド用文字コード
CMT_FREQUENCY_Hz	(2u)	コンペアマッチタイマの割り込み周期(2Hz)
STRING_MAX_SIZE	((uint16_t) TX_MAX_SIZE)	出力する文字列の最大サイズ
STRING_SIZE	((uint16_t)((sizeof((s))) – 1u))	文字列配列のサイズを uint16_t 型で示すマクロ

表 5.4　サンプル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	((uint8_t)0x01)	XMODEM コントロールコード(SOH)
XM_EOT	((uint8_t)0x04)	XMODEM コントロールコード(EOT)
XM_ACK	((uint8_t)0x06)	XMODEM コントロールコード(ACK)
XM_NAK	((uint8_t)0x15)	XMODEM コントロールコード(NAK)
XM_CAN	((uint8_t)0x18)	XMODEM コントロールコード(CAN)
XM_HEADER_SIZE	((uint8_t)(1+1+1))	XMODEM データブロックのヘッダーサイズ(バイト数)
XM_DATA_SIZE	((uint8_t)128)	XMODEM データブロックのデータサイズ(バイト数)
XM_SUM_SIZE	((uint8_t)1)	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	((uint8_t)10)	XMODEM データ転送タイムアウト時のリトライ回数
UINT8T_0	((uint8_t)0)	uint8_t 型の 0
UINT8T_1	((uint8_t)1)	uint8_t 型の 1

表 5.5　サンプルプログラムで使用する定数(r_fw_up_rx_private.h)

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_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

表 5.6　サンプル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	(0x02)	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	(0x04)	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	(0x06)	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	(0x08)	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	(1)	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	(0)	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。

5.3.3　型定義一覧

サンプルプログラムで使用する型定義を示します。

r_xmodem.c の型定義

```
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 *precv_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;
```

r_xmodem_if.h の型定義

```
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);
```

r_fw_up_rx_if.h の型定義

```
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_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;
```

r_fw_up_buf.h の型定義

```
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 *paddress;
    uint8_t *pdata;
    uint8_t type;
    uint8_t act;
    struct MotSBufS *pnext;
} 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 *pnext;
    struct WriteDataS *pprev;
} fw_up_write_data_t;
```

5.3.4　変数一覧

表 5.7～表 5.10 に static 型変数を、表 5.11 に const 型変数を示します。

表 5.7　サンプルプログラムで使用する定数(r_fw_up_buf.h)

型	変数名	内容	使用関数
static sci_hdl	sci_handle	SCI モジュール制御ハンドル	main send_byte_xm recv_byte_xm send_string_sci
static volatile bool	sci_send_end_flag	SCI 送信完了判定用フラグ	sci_callback send_string_sci
static uint32_t	cmt_ch	コンペアマッチタイマのチャンネル番号	update_dualbank
static volatile int32_t	timeout_count	タイムアウト判定用カウンタ	cmt_callback recv_byte_xm
static volatile bool	timeout_flag	タイムアウト判定用フラグ	cmt_callback recv_byte_xm
static volatile bool	start_timer_flag	タイムアウト判定有効フラグ	cmt_callback recv_byte_xm

表 5.8　static 型変数(r_xmodem.c)

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

表 5.9　static 型変数(r_fw_up_rx.c)

型	変数名	内容	使用関数
static bool	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 write_firmware fw_up_put_data fw_up_get_data

表 5.10　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	*papp_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	*papp_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	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	*papp_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	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	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	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	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 5.11　const 型変数(main.c)

型	変数名	内容	使用関数
static const uint8_t	string_menu0[]	“RX65N-2MB Dualbank firmware update menu ver1.00A¥r¥n”	show_menu_dualbank
static const uint8_t	string_menu1[]	“1...Update¥r¥n”	show_menu_dualbank
static const uint8_t	string_menu2[]	“2...Changing the Startup Bank and Reset¥r¥n”	show_menu_dualbank
static const uint8_t	string_change_bank[]	"Changing the Startup Bank¥r¥n"	change_startup_bank
static const uint8_t	string_reset[]	"Reset¥r¥n"	change_startup_bank
static const uint8_t	string_crlf[]	“¥r¥n”	main
static const uint8_t	string_start_xmodem[]	"Ready¥r¥n"	update_dualbank
static const uint8_t	string_finish_xmodem[]	"Finish¥r¥n"	update_dualbank
static const uint8_t	string_timeout[]	"Timeout¥r¥n"	update_dualbank
static const uint8_t	string_cmt_err[]	"CMT error¥r¥n"	update_dualbank
static const uint8_t	string_erase_err[]	"Erase error¥r¥n"	update_dualbank
static const uint8_t	string_flash_err[]	"Flash error¥r¥n"	change_startup_bank
static const uint8_t	string_block_err[]	"Block processing error¥r¥n"	update_dualbank
static const uint8_t	string_data_err[]	“Data error¥r¥n”	update_dualbank
static const uint8_t	string_recv_can[]	"Received CAN¥r¥n"	update_dualbank
static const uint8_t	string_initialize_update_err[]	"initialize update error¥r¥n"	update_dualbank
static const uint8_t	string_finalize_update_err[]	"finalize update error¥r¥n"	update_dualbank
static const uint8_t	string_reset_vector_invalid[]	"Reset vector is invalid¥r¥n"	change_startup_bank

5.3.5　関数一覧

表 5.12 にサンプルプログラムの関数を、表 5.13 にサンプルプログラムが使用する FIT モジュールの関数を、表 5.14 にサンプルプログラムが使用する 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を示します。

表 5.12　サンプルプログラムの関数

関数名	概要	記載ファイル
main	メイン処理	main.c
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
cmt_callback	CMT FIT モジュールのコールバック関数、CMT を使用したタイムアウトの確認	main.c
send_string_sci	文字列送信処理	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
show_menu_dualbank	メニュー表示	main.c
update_dualbank	ファームウェアアップデート処理	main.c
change_startup_bank	起動バンク変更処理	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
erase_another_bank	コードフラッシュメモリ消去	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
bank_toggle	起動バンク切り替え	r_fw_up_rx.c
fw_up_soft_reset	ソフトウェアリセット実行	r_fw_up_rx.c
fw_up_check_reset_vector	リセットベクタの確認	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
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
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 5.13　サンプルプログラムが使用する 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_another_bank
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
R_FLASH_Control	フラッシュ FIT モジュール	起動バンク切り替え	bank_toggle
R_BSP_RegisterProtectDisable	BSP モジュール	SWRR のプロテクト解除	fw_up_soft_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
R_CMT_CreatePeriodic	CMT FIT モジュール	タイムアウト確認用タイマ生成	update_dualbank
R_CMT_Stop	CMT FIT モジュール	タイムアウト確認用タイマ停止	update_dualbank

表 5.14　サンプルプログラムが使用する 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

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_SCI8	SCI FIT モジュール	SCI8 を使用する場合の SCI 用端子設定	main
R_SCI_PinSet_SCI6	SCI FIT モジュール	SCI6 を使用する場合の SCI 用端子設定	main

6. 付録

6.1 トラブルシューティング

6.1.1 mot ファイルを Renesas Flash Programmer で書き込む時に異常終了する

「エラー(E3000107): デバイスが接続情報と一致しません。操作は失敗しました。」と表示され異常終了することがあります。[ファイル]の[新しいプロジェクトを作成]で新しいプロジェクトを作成してください。

このエラーが発生するのはプロジェクトの接続情報とデバイスが一致しないためです。Renesas Flash Programmer の持つ接続情報はリニアモードとデュアルモードを区別します。リニアモードのプロジェクトではデュアルモードで動作しているマイコンと接続できません。またデュアルモードのプロジェクトもリニアモードで動作しているマイコンと接続できません。

6.1.2 XMODEM でファイルの転送が途中で停止する

XMODEM 転送が停止する原因は以下の 2 つが考えられます。

(1) mot ファイルではないファイルを転送した場合

シリアル通信ソフトウェアの XMODEM 処理を終了させてください。あるいはシリアル通信ソフトウェアの XMODEM 処理がタイムアウトするまでお待ちください。その後、もう一度、Update の処理を行ってください。XMODEM 処理終了の操作方法はシリアル通信ソフトウェアの説明書をご覧ください。

(2) mot ファイルの選択に 10 秒以上かかった場合

シリアル通信ソフトウェアの XMODEM 処理を終了させてください。サンプルプログラムを再実行してください。その後、もう一度、Update の処理を行ってください。XMODEM 処理終了の操作方法はシリアル通信ソフトウェアの説明書をご覧ください。

この現象が発生する理由は、シリアル通信ソフトウェアが、XMODEM 送信ファイルの選択時に NAK を正常に処理できないためです。この場合、通信のタイミングにずれが生じ、データブロックの送信が繰り返されます。

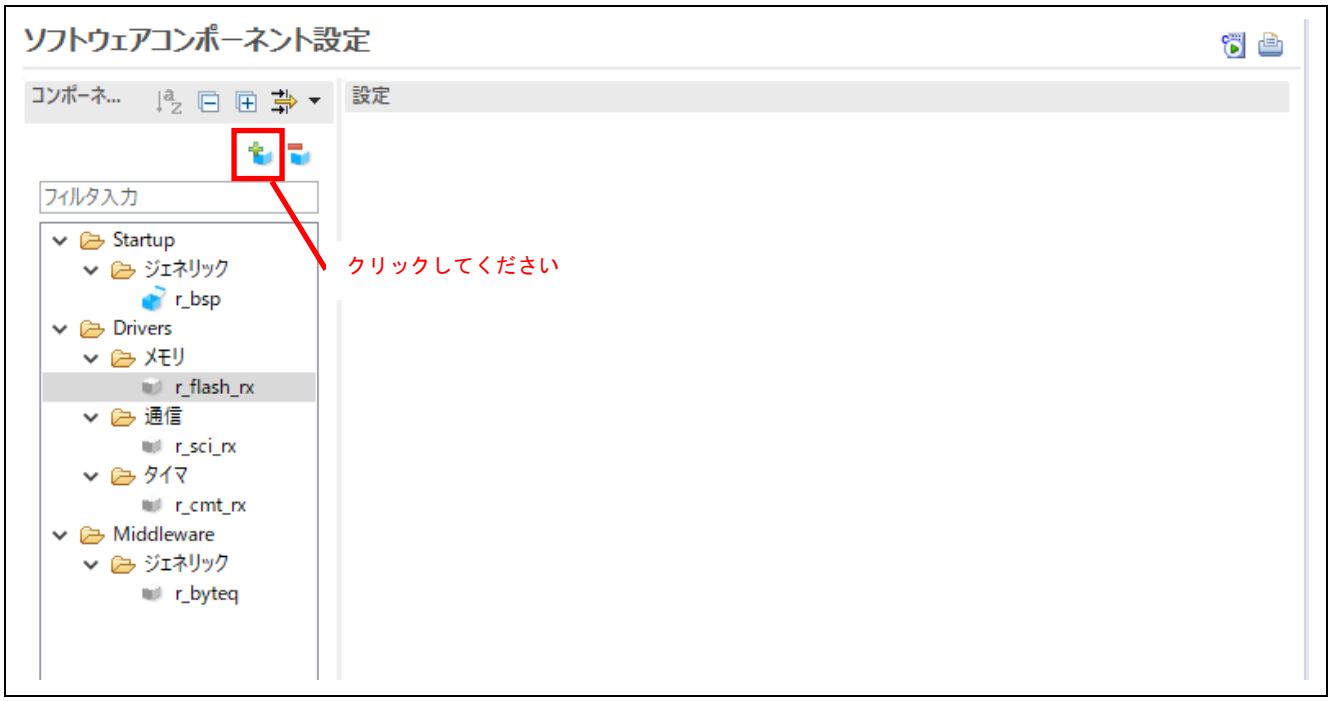
6.1.3 XMODEM 転送が終了したのに何も表示されない

シリアル通信ソフトウェアに 1、2 以外の任意の文字を入力してください。エコーバックが表示されればサンプルプログラムも XMODEM 処理を終了しています。

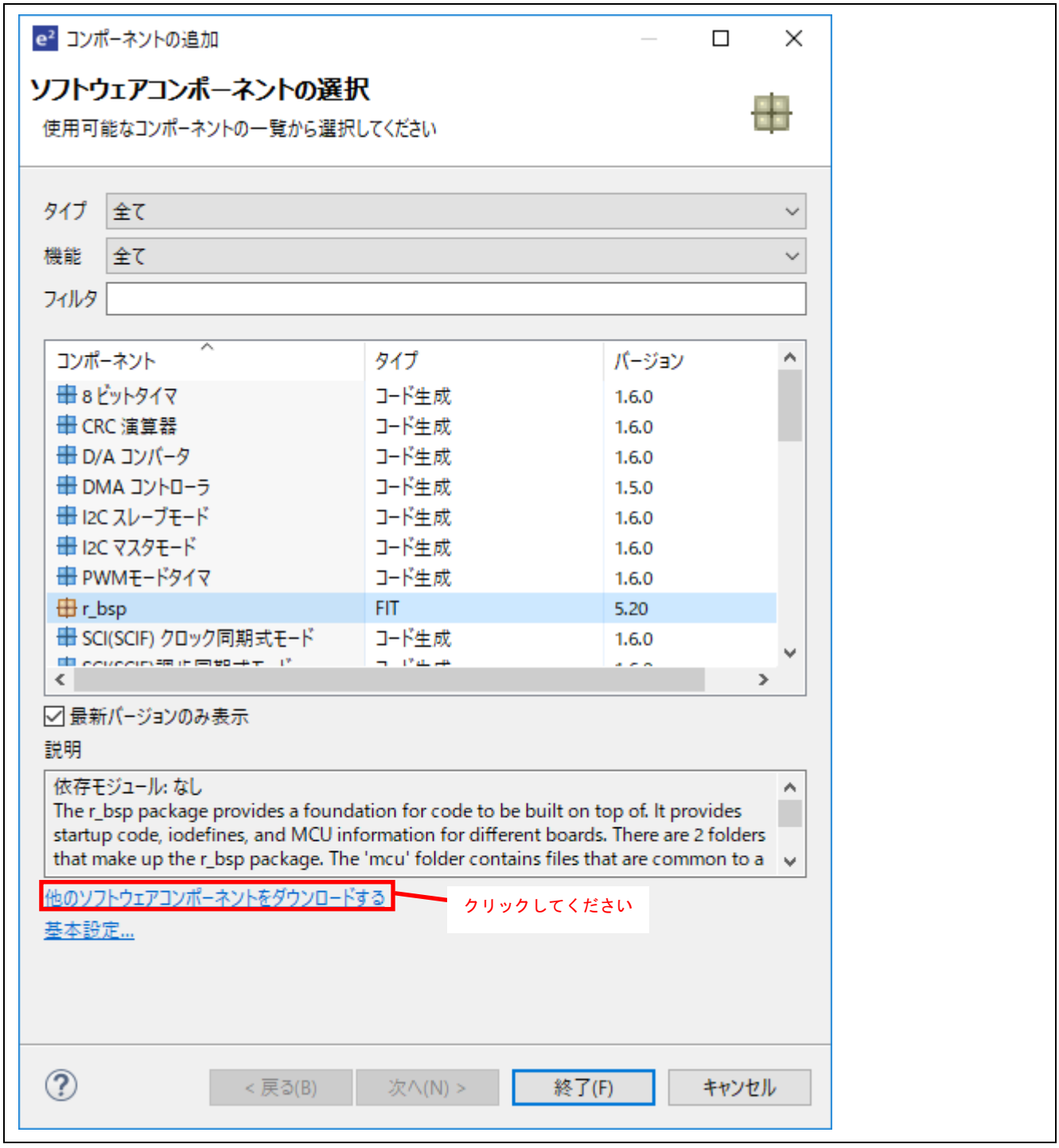
この現象が発生する理由はシリアル通信ソフトウェアが、XMODEM 処理と通常の通信処理の切り替えに時間がかかりサンプルプログラムが送信しているメッセージを受信できないためです。

6.1.4　ソフトウェアコンポーネント設定で FIT モジュールの設定項目が表示されない

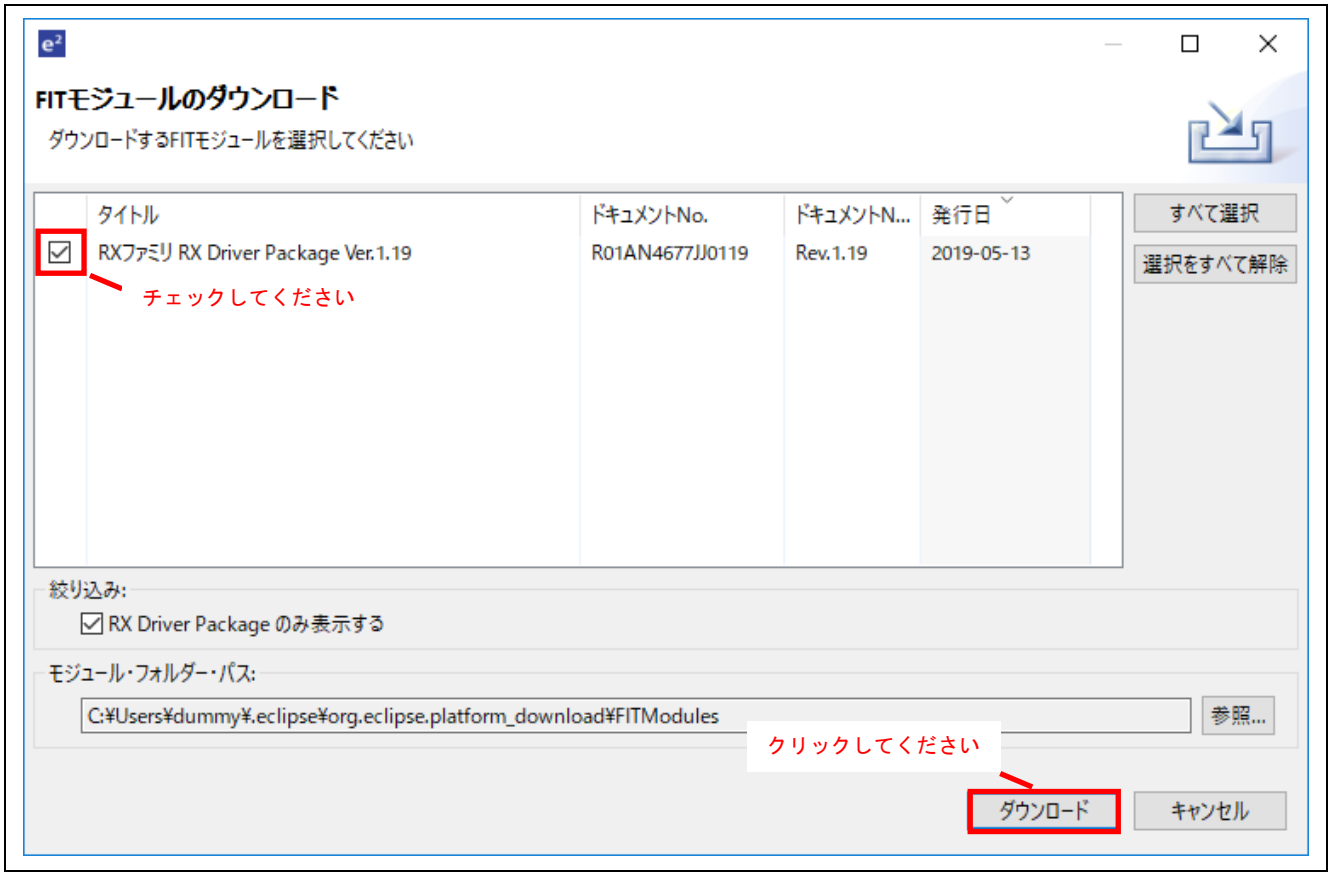
[ソフトウェアコンポーネント設定]で FIT モジュールが灰色で表示され設定項目が表示されない場合、スマート・コンフィギュレータが認識しているフォルダにプロジェクトで使用されているバージョンの FIT モジュールがダウンロードされていない可能性があります。最新の FIT モジュールのダウンロードを行います。[コンポーネントの追加]をクリックしてください。



[他のソフトウェアコンポーネントをダウンロードする]をクリックしてください。

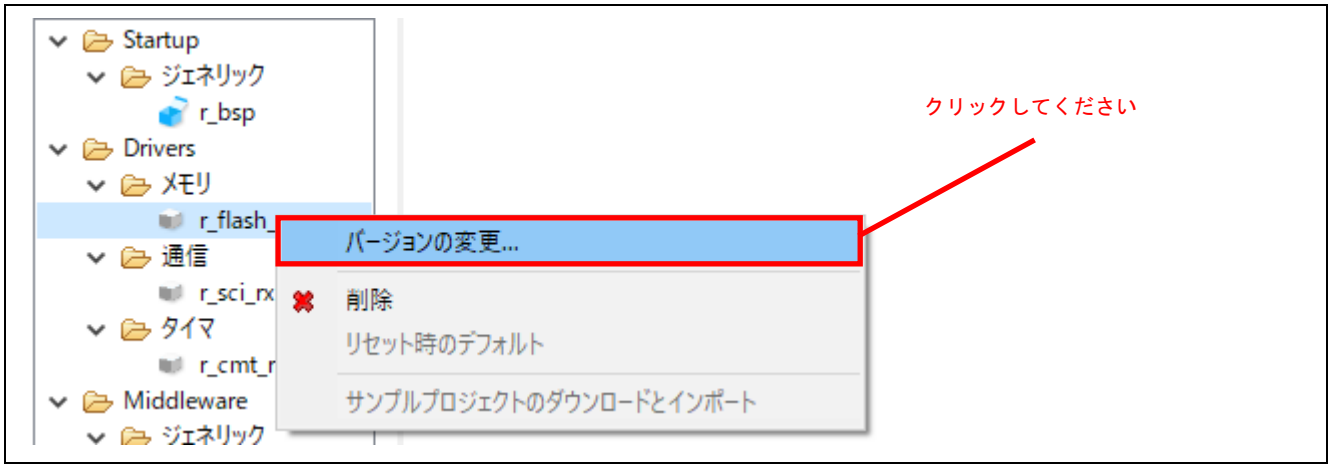


最新の RX Driver Package が表示されるのでチェックを入れダウンロードしてください。



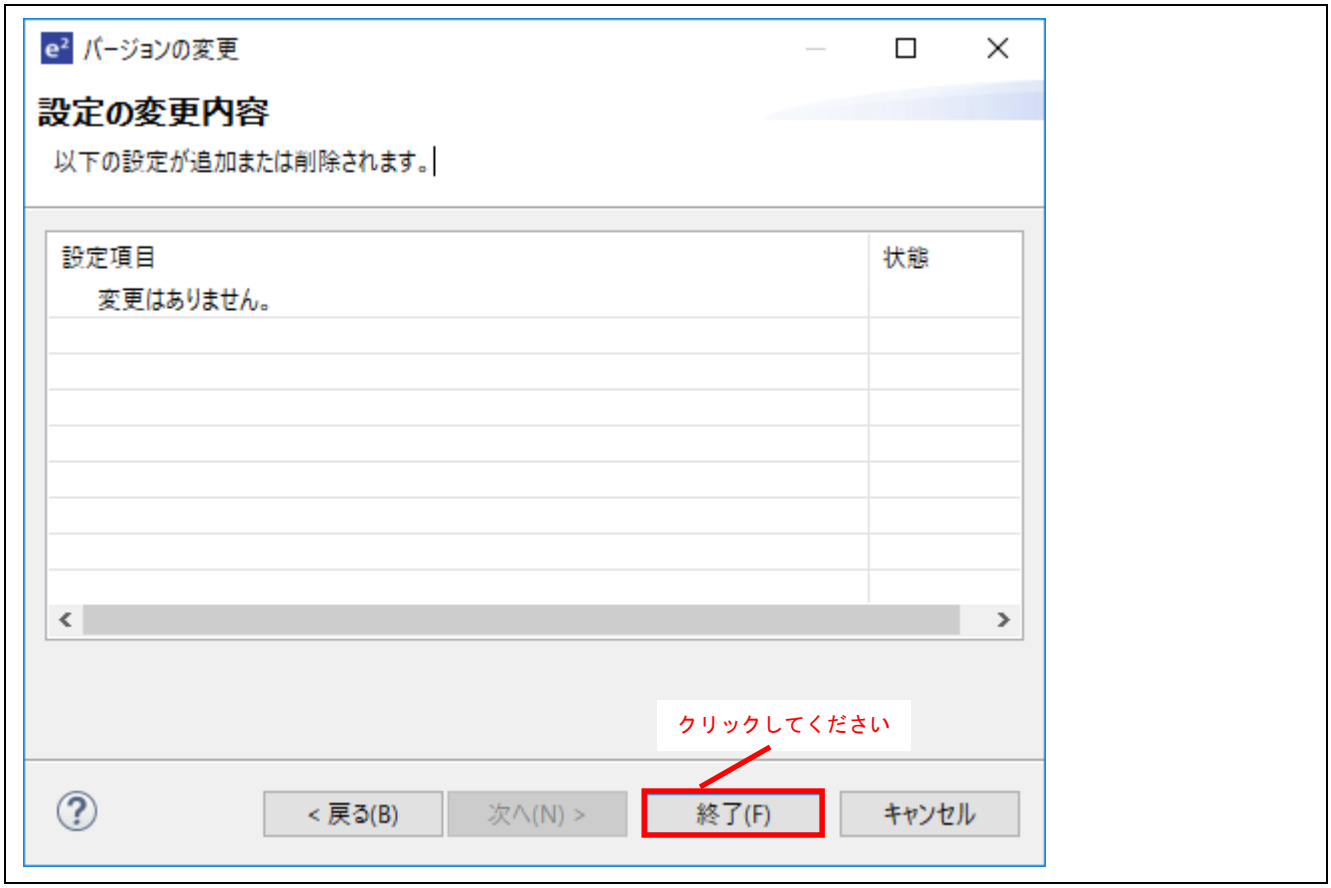
最新の FIT モジュールをダウンロードしても灰色から変化のない場合はプロジェクトで使用している FIT モジュールを最新のものにバージョンアップしてください。

コンテキストメニューから[バージョンの変更]をクリックしてください。

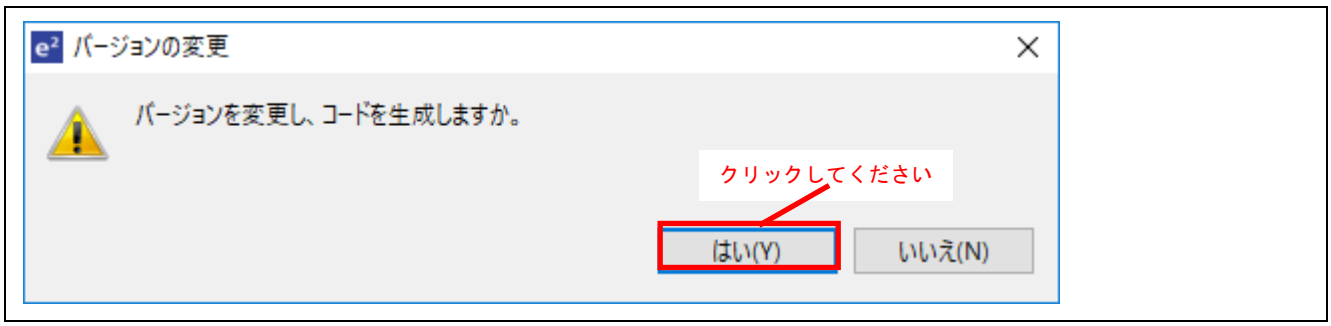


[次へ]をクリックしてください。

[終了]をクリックしてください。

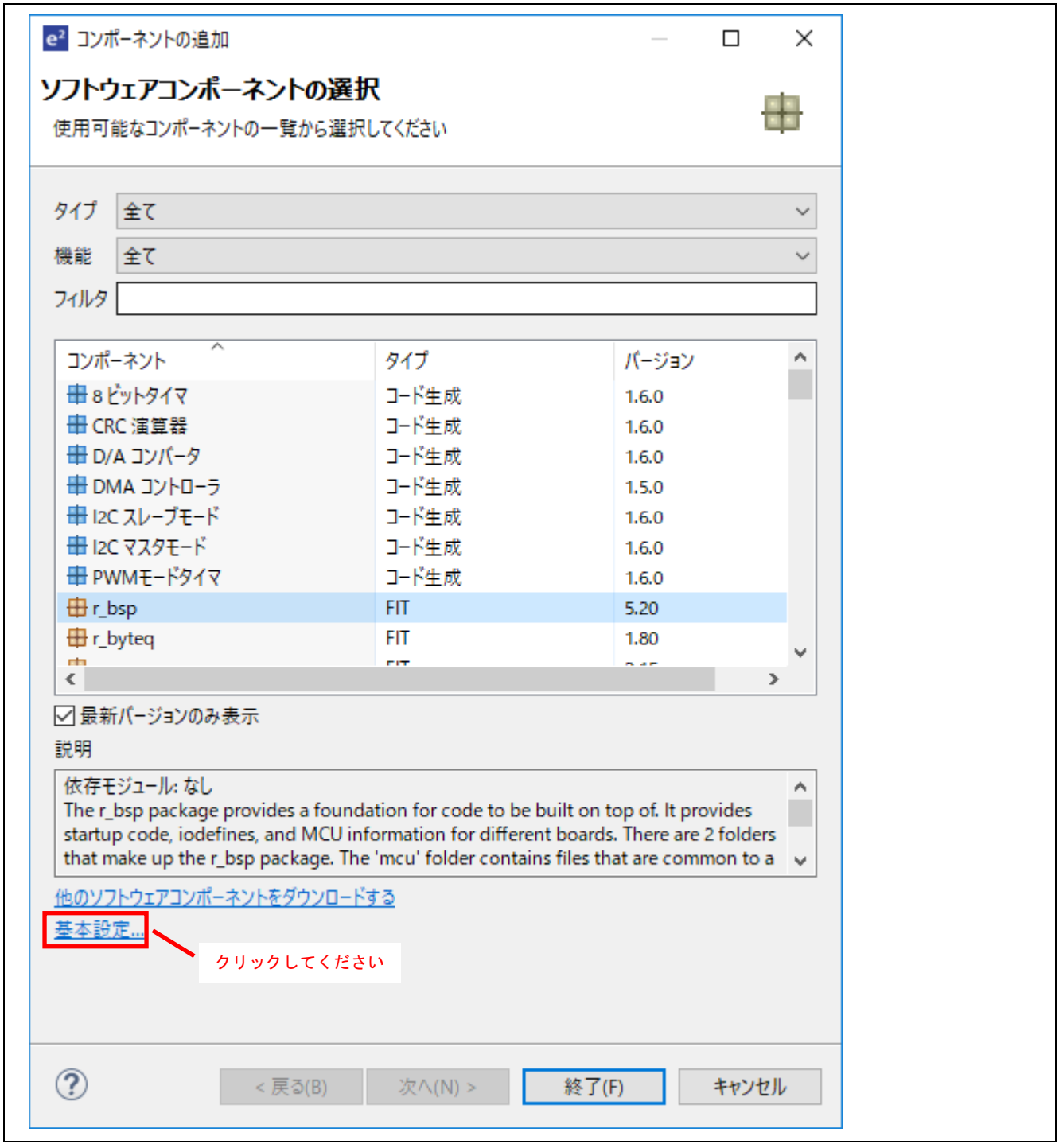


[はい]をクリックしてください。

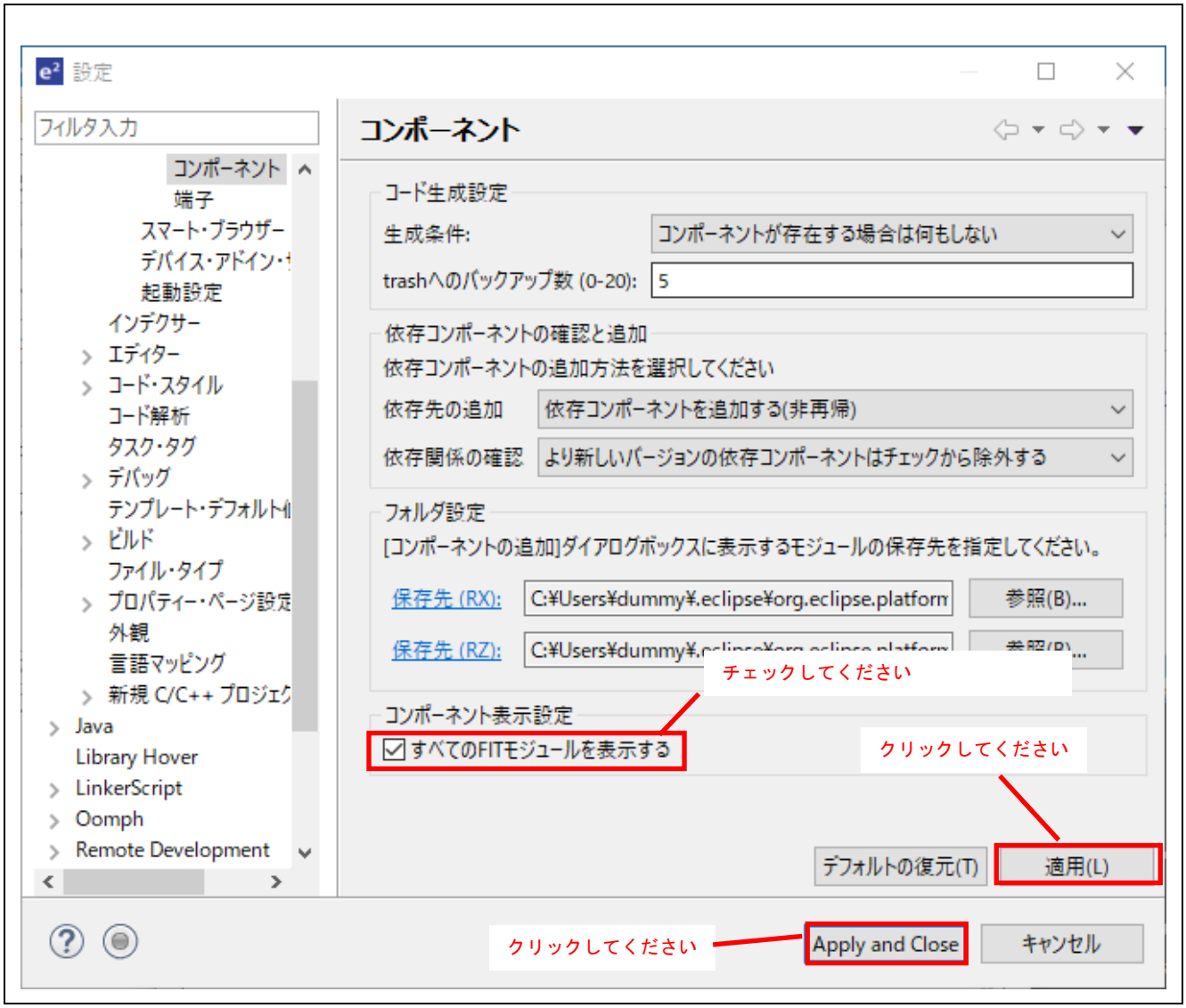


6.1.5　[コンポーネントの追加]ダイアログで SCI FIT モジュールが表示されない

[コンポーネントの追加]ダイアログで[基本設定]をクリックしてください。



[すべての FIT モジュールを表示する]をチェックしてください。
[適用]をクリックしてください。
[Apply and Close]をクリックしてください。



6.2　FAQ

6.2.1　Q:デバイスを変更したい

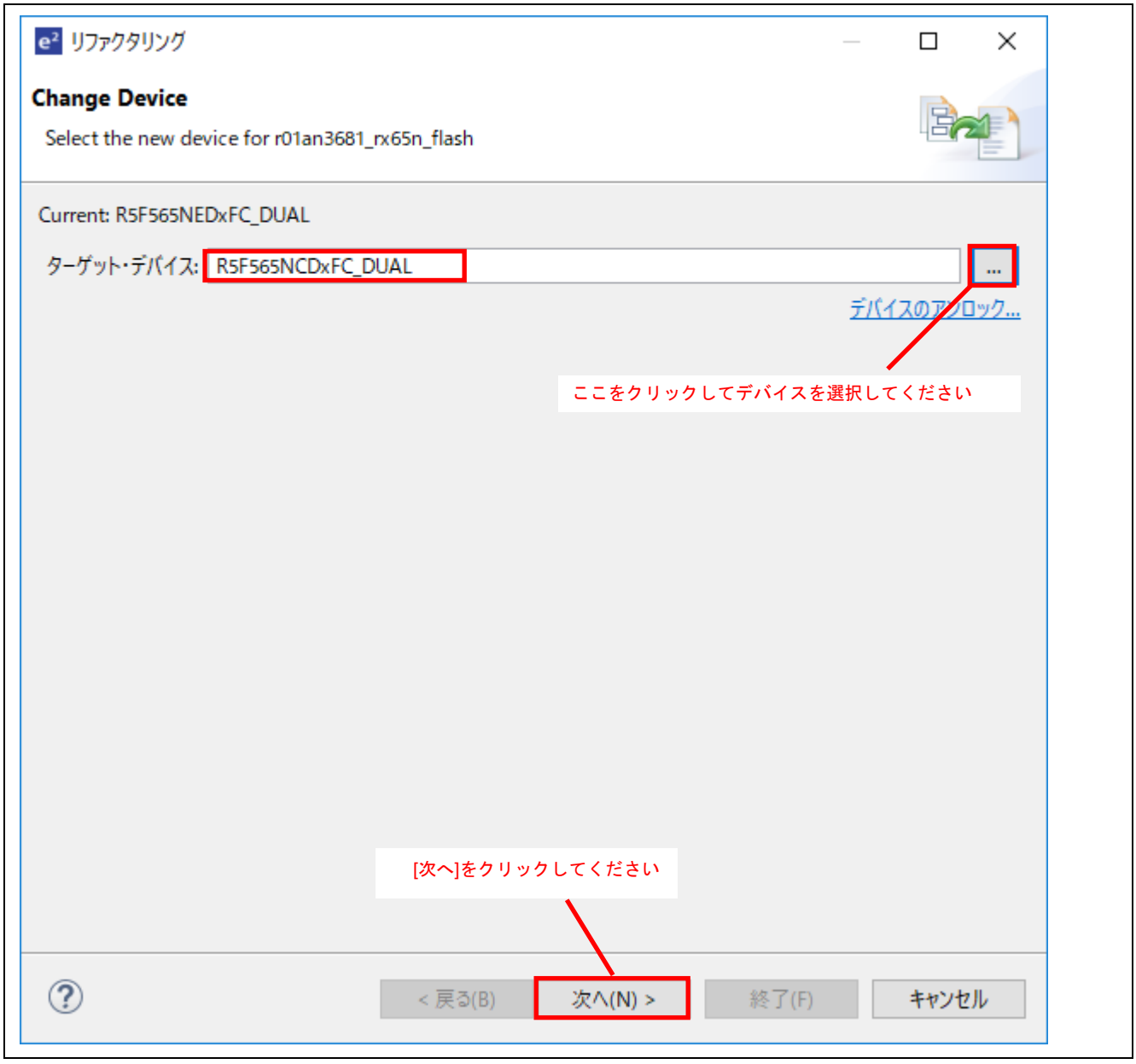
RX65N の 2MB 用のプロジェクトを RX65N の 1.5MB のデバイス用に変更することを例に説明します。

プロジェクト・エクスプローラーで対象プロジェクトのコンテキストメニューを表示してください。

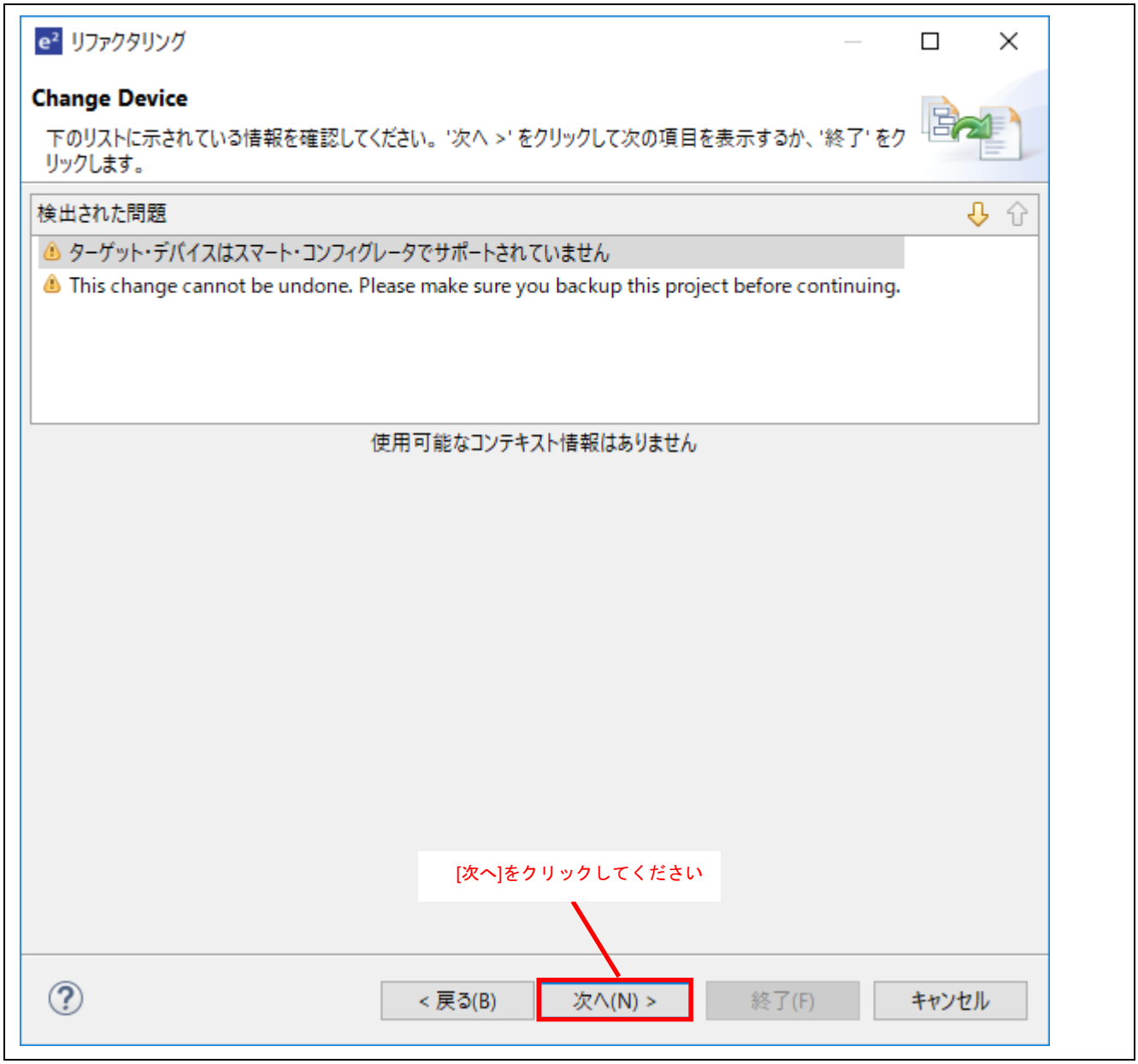
[Change Device]をクリックしてください。

変更するデバイスを選択します。

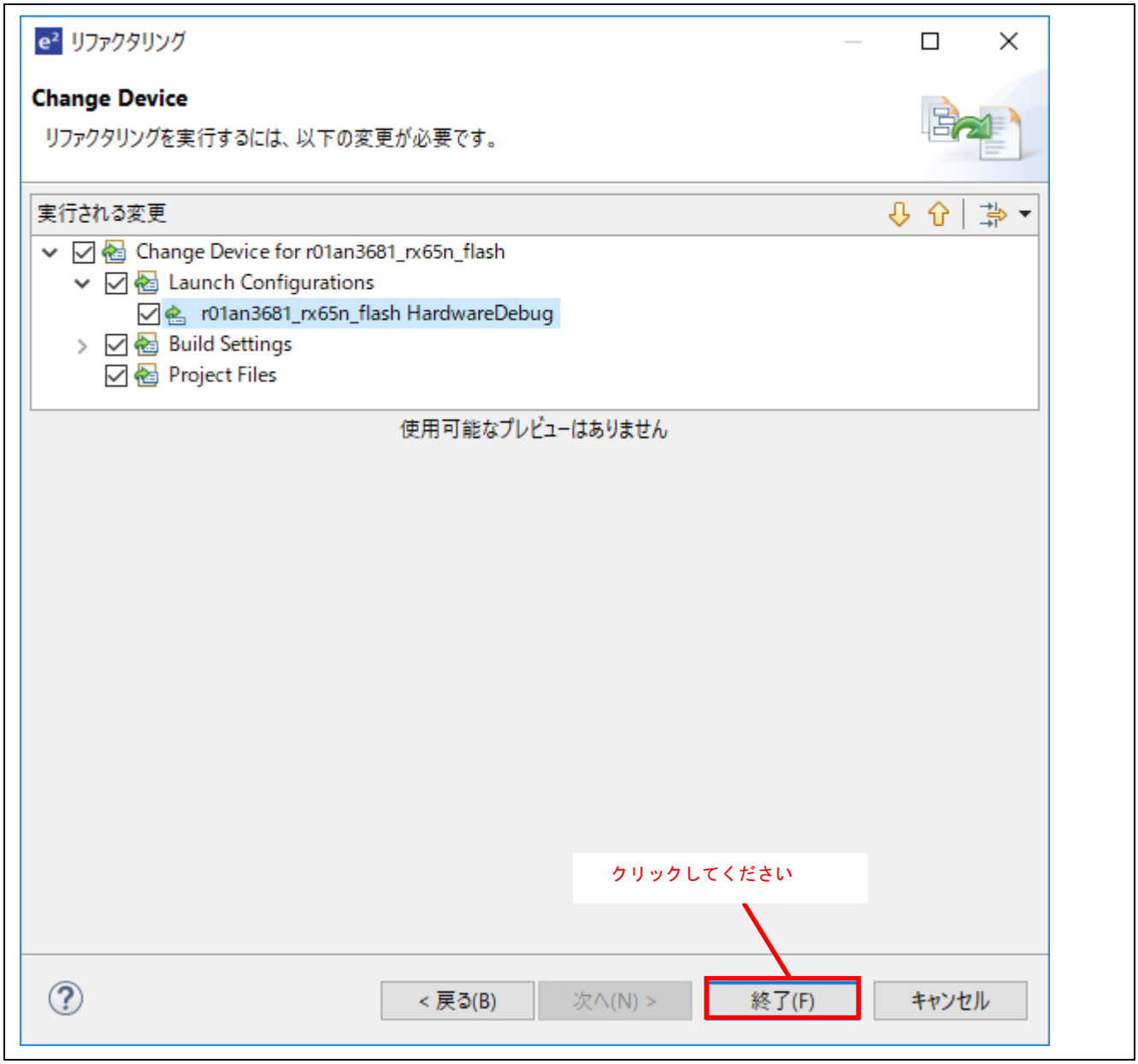
選択したら[次へ]をクリックしてください。



警告が出ていますが[次へ]をクリックしてください。



[終了]をクリックしてください。

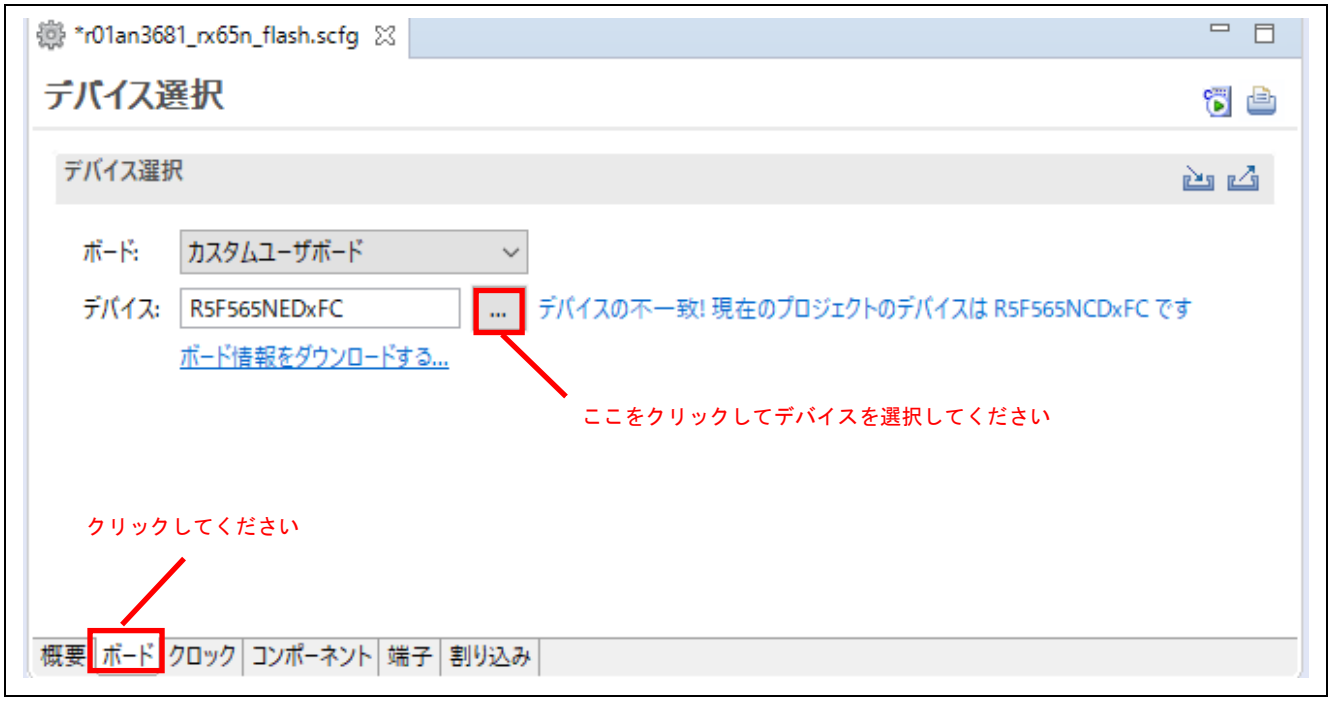


スマート・コンフィギュレータで選択されているデバイスを変更します。

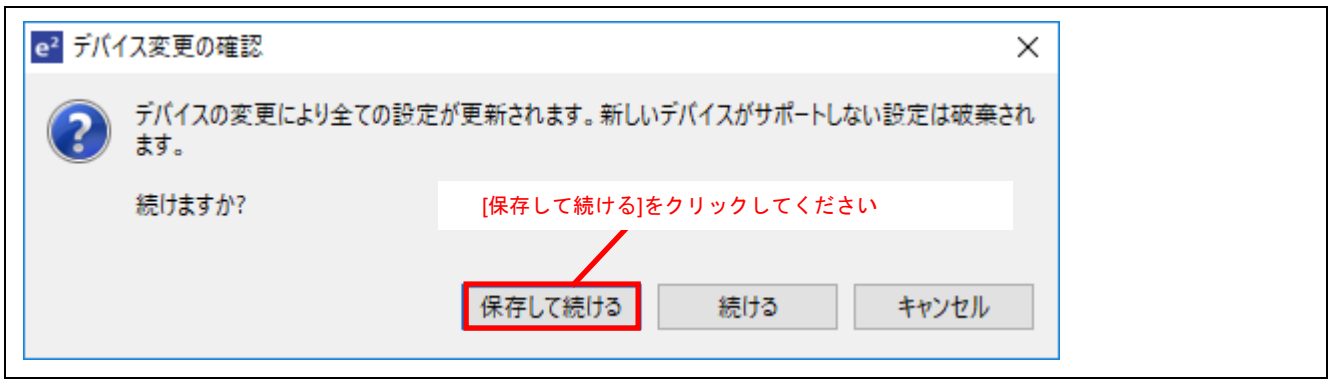
プロジェクト・エクスプローラーから r01an3681_rx65n_flash.scfg をダブルクリックしてください。

[ボード]タブをクリックしてください。

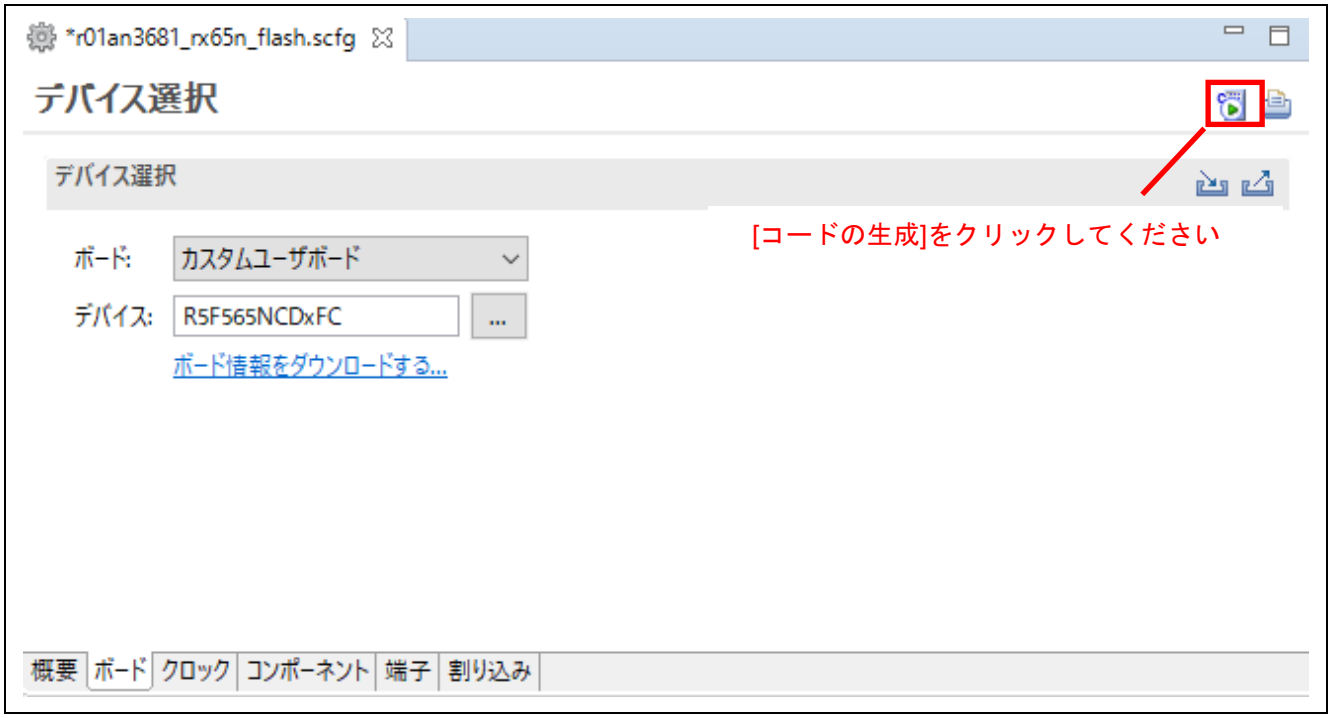
先ほど選択したデバイスをここでも選択します。



デバイス変更の確認ダイアログが表示されます。[保存して続ける]をクリックしてください。



[コードの生成]をクリックしてください。



RAM 領域に追加した RPFRAM2 セクションの設定が消えています。「3.4.3[C/C++ビルド]の設定の変更」を参考にして RPFRAM2 セクションを追加してください。

使用する SCI のチャネルを変更する場合は「3.4.1(3)SCI API の設定変更」、「3.4.2 端子設定」、「5.3.2 定数一覧」、「5.3.5 関数一覧」を参考にしてください。

6.2.2　Q:ビッグ・エンディアンに変更したい

変更が必要な箇所が 2 つあります。[C/C++ビルド]と[デバッグ構成]です。

プロジェクトの[プロパティ]をクリックしてください。

[C/C++ビルド]の[設定]をクリックしてください。

[ツール設定]タブをクリックしてください。

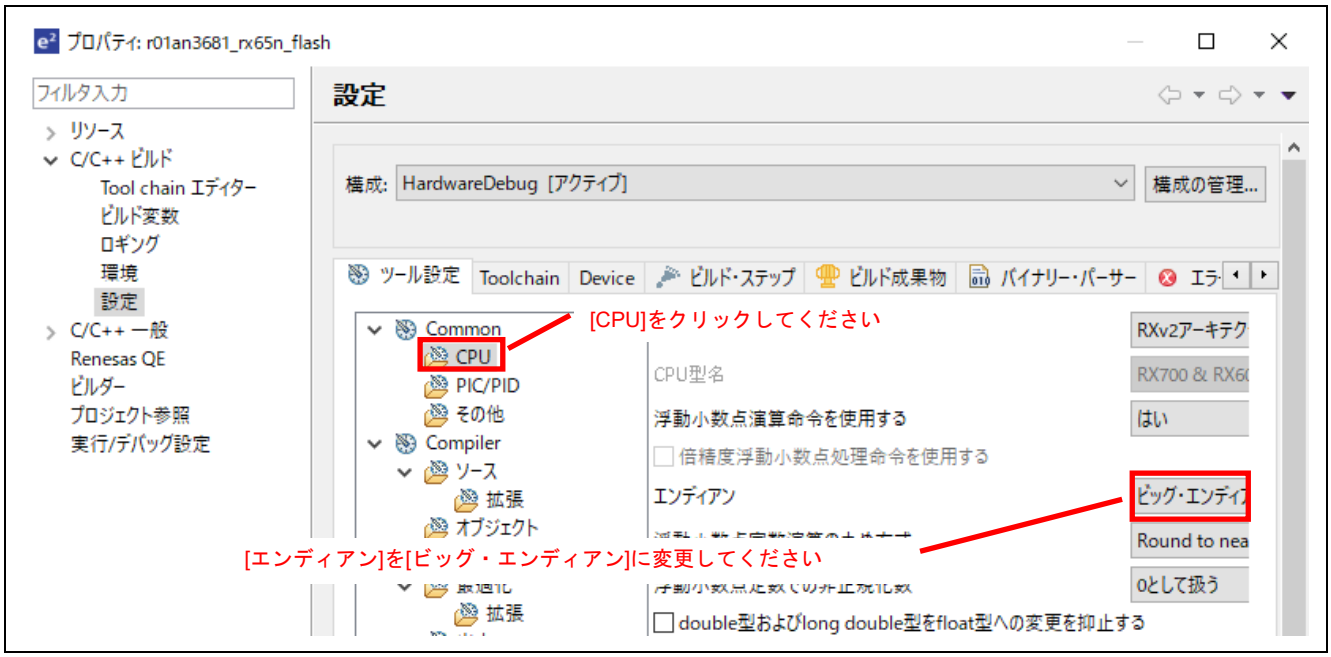
[Common]の[CPU]をクリックしてください。

[エンディアン]を[ビッグ・エンディアン]に変更してください。

HardwareDebug 構成と Release 構成はそれぞれ変更する必要があります。

構成を[全ての構成]にすると両方の構成を一度に変更できます。

詳細な手順は「3.4.2 プロジェクトの設定の変更」を参照してください。



[プロジェクトのクリーン]を行ってから、[プロジェクトのビルド]を行ってください。

HardwareDebug 構成と Release 構成でそれぞれ行う必要があります。

[実行]の[デバッグ構成]をクリックしてください。

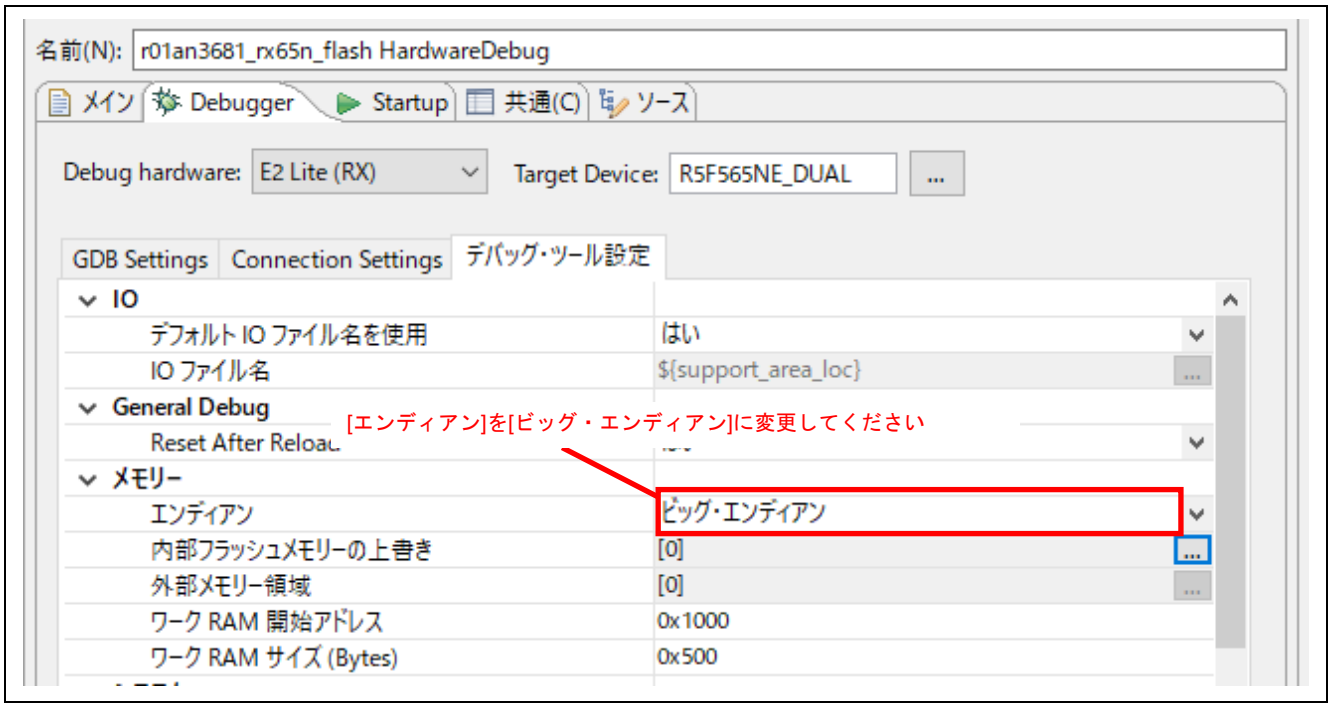
[Renesas GDB Hardware Debugging]の変更するデバッグ構成をクリックしてください。

[Debugger]タブをクリックしてください。

[デバッグ・ツール設定]タブをクリックしてください。

[メモリー]の[エンディアン]を[ビッグ・エンディアン]に変更してください。

詳細な手順は「4.3 デバッグ構成」を参照してください。



6.2.3　Q:SCI 以外のインタフェースに変更したい

インタフェースを変更する場合、変更が必要な関数は main.c とスマート・コンフィグレータが生成する端子設定用関数になります。

変更が必要な関数を表 6.1 に示しました。それぞれの関数をインタフェースに合わせた内容に変更してください。sci_callback は送信完了を検出する目的で使用しているコールバック関数です。インタフェースに合わせた送信完了の検出方法に変更してください。送信完了を検出する手段がない場合は send_string_sci 関数もしくはそれを呼び出している箇所にウェイト処理を入れるなどの変更を加えてください。

FIT モジュールの端子設定はスマート・コンフィグレータを使用して生成してください。 [コンポーネント]タブのプロパティで設定できます。

表 6.1　変更が必要な関数

関数名	用途	ファイル
main	インタフェースの初期化、XMODEM プロトコル以外の受信を行う。	main.c
sci_callback	文字列の送信完了を検出する	main.c
send_string_sci	文字列を送信する	main.c
send_byte_xm	XMODEM プロトコルで 1 バイト送信する。戻り値として正常終了か送信エラーを返す。	main.c
recv_byte_xm	XMODEM プロトコルで 1 バイト受信する。10 秒でタイムアウトする。戻り値として正常終了、受信エラー、タイムアウトを返す。	main.c
R_SCI_PinSet_SCI8	端子設定を行う。	r_sci_rx_pinset.c

6.2.4　Q:オプション設定メモリを消去したい

Renesas Flash Programmer で以下の手順を行ってください。

1. [操作設定]タブをクリックしてください。
2. [コマンド]の[消去(E)]をチェックしてください。
3. [消去オプション(O)]を[チップ消去]にしてください。
4. [書き込み(P)]、[チェックサム(S)]、[ベリファイ(V)]のチェックを外してください。
5. [操作]タブをクリックしてください。
6. [スタート]ボタンをクリックしてください。
7. オプション設定メモリが消去されます。

データフラッシュメモリとコードフラッシュメモリも同時に消去されるので注意してください。消去オプションが[チップ消去]ではない場合、Renesas Flash Programmer はオプション設定メモリを消去しません。

オプション設定メモリの詳細については「RX65N グループ、RX651 グループ ユーザーズマニュアルハードウェア編」,「RX72M グループ ユーザーズマニュアルハードウェア編」を参照してください。

6.2.5　Q:起動バンクと逆側のバンクをリードしたい

デバッグパースペクティブの時に、[ウィンドウ]の[ビューの表示]から[メモリ]をクリックしてください。[メモリー・モニターの追加]ボタンをクリックしてください。[メモリー・モニター]ダイアログが開くので逆側のバンクのアドレスを入力してください。

またデバッグ構成の[Debugger]-[デバッグ・ツール設定]-[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]が[はい]になっている事を確認してください。[いいえ]の場合、書き換えた内容がデバッガに反映しません。

6.2.6　Q:どちらのバンクが起動バンクになっているか確認したい

起動直後に BANKSEL レジスタをリードすることでどちらが起動バンクか確認ができます。BANKSEL レジスタの値を変更した場合、その値はリセット後の起動バンクを示します。

BANKSEL レジスタの詳細については「RX65N グループ、RX651 グループ ユーザーズマニュアルハードウェア編」,「RX72M グループ ユーザーズマニュアルハードウェア編」を参照してください。

6.2.7　Q:CS+にサンプルプログラムをインポートしたい

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。

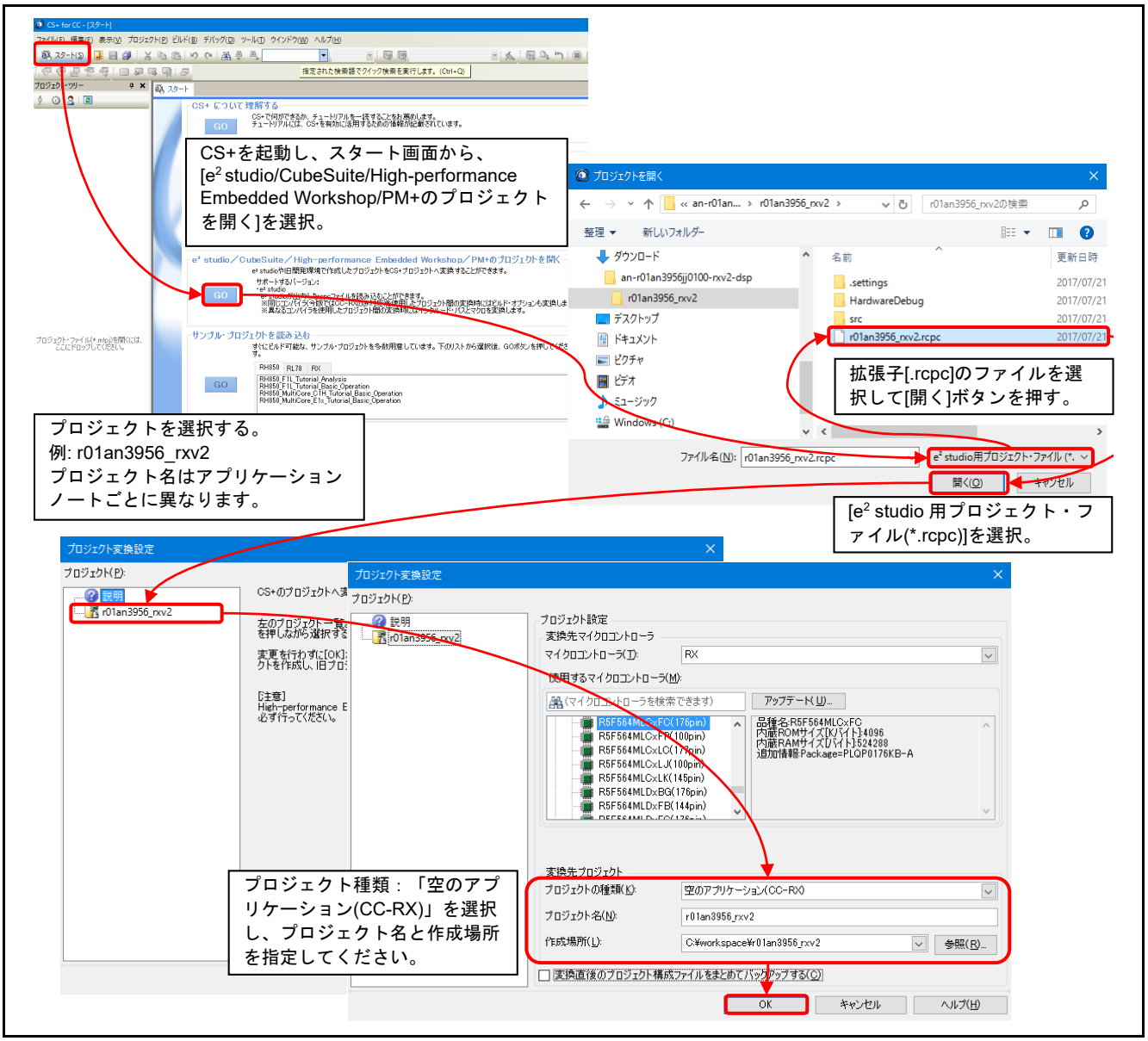


図 6.1　プロジェクトを CS+にインポートする方法

7.　補足

7.1　無償評価版の「RX ファミリ用 C/C++コンパイラパッケージ」を利用する場合の注意事項

無償評価版の「RX ファミリ用 C/C++コンパイラパッケージ」には、使用期限と使用制限があります。使用期限が過ぎた場合、リンクサイズが 128K バイト以内に制限されるためロードモジュールが正しく生成されなくなる場合があります。

詳しくは、ルネサスのホームページにある、無償版ソフトウェアツールのページを参照してください。

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.0	2017/10/02	-	初版
1.1	2017/12/18	-	Rev1.00 に同梱しているフラッシュ FIT モジュール Rev3.2 が評価版であったので正式版に更新した。
1.2	2019/09/01	-	RX72M グループ対応。 統合開発環境を e^2 studio Version 7.5.0 に変更。 FIT モジュールのコンフィギュレーションはスマート・コンフィギュレータを使用した内容に変更。 ROM 容量の変更を e^2 studio の機能で行うように変更。 デバッグ・ハードウェアを E2 エミュレータ Lite に変更。

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。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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