

RX63N グループ

FIT を用いたフラッシュメモリの書き換え

要旨

本サンプルコードでは、FIT を用いて、特定の内蔵フラッシュメモリ（ROM、および E2 データフラッシュ）のアドレスに特定の値を書き込む方法について説明します。

対象デバイス

- RX63N

内容

1.	仕様.....	3
2.	動作確認条件.....	3
3.	ハードウェア説明.....	3
4.	ソフトウェア説明.....	4
4.1	動作概要.....	4
4.2	ファイル構成.....	6
4.3	オプション設定メモリ.....	7
4.4	定数一覧.....	8
4.5	enum 一覧.....	9
4.6	変数一覧.....	9
4.7	関数一覧.....	10
4.8	関数仕様.....	11
4.9	作成する関数のフローチャート.....	13
4.9.1	メイン処理.....	13
4.9.2	内蔵フラッシュメモリの書き換え.....	14
5.	FIT モジュールのダウンロード方法.....	16
6.	FIT モジュールの組み込み.....	22
7.	CC-RX の設定.....	35
7.1	インクルード・パスの確認.....	35
7.2	ビルド方法の変更.....	36
7.3	C 言語規格の変更.....	36
7.4	可変ベクタ空き領域の設定.....	38
7.5	セクションアドレスの設定.....	39
7.6	デバッグ・ツールの設定.....	42
8.	動作確認方法.....	44
8.1	メモリ.....	44
9.	参考ドキュメント.....	50

1. 仕様

FIT モジュールを用いて、特定の内蔵フラッシュメモリ（ROM、および E2 データフラッシュ）のアドレスに特定の値を書き込みます。

2. 動作確認条件

本サンプルコードは、表 2.1 の条件で動作を確認しています。

表 2.1 動作確認条件

項目	内容
使用マイコン	R5F563NFDDFP (RX63N グループ)
動作周波数	<ul style="list-style-type: none">・メインクロック：12MHz・PLL：192MHz（メインクロック 1 分周 16 通倍）・システムクロック (ICLK)：96MHz（PLL 2 分周）・Flash IF クロック (FCLK)：48MHz（PLL 4 分周）
ボード電源電圧	5V
マイコン動作電圧	3.3V
エンディアン	リトルエンディアン
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
統合開発環境	ルネサスエレクトロニクス製品 CS+ for CC V5.00.00
エミュレータ	ルネサスエレクトロニクス製 E1 エミュレータ
使用ボード	北斗電子製評価ボード HSBRX63NP (R5F563NFDDFP)

3. ハードウェア説明

本サンプルコードでは外部にハードウェアを使用しません。ただし、動作確認用でボードに実装されている LED を制御しています。

4. ソフトウェア説明

4.1 動作概要

BSP(ボードサポートパッケージ)FIT モジュール、Flash API(フラッシュメモリ)FIT モジュールを CS+に組み込み、組み込んだ FIT モジュールを使用して、特定の内蔵フラッシュメモリ (ROM、および E2 データフラッシュ) のアドレスに特定の値を書き込みます。

<FIT>

FIT(Firmware Integration Technology)は、一定のルールに従って作成されたルネサスから提供するファームウェアです。BSP、周辺機能モジュール、ミドルウェアモジュール、インタフェースモジュールで構成されており、これらのモジュールを使用することにより、ソフトウェア開発が容易になります。

<BSP>

BSP は FIT モジュールを使用するプロジェクトの基盤であり、リセットから main 関数までにおけるマイコン初期設定、クロック設定などと基本設定を行うモジュールが含まれています。なお、対象ボードは YRDKRX63N となっていますので、必要に応じて変更する必要があります。

<Flash API>

この FIT モジュールは、周辺機能モジュールの一つであり、内蔵フラッシュメモリのプログラミングおよび消去プロセスを実行するためのモジュールが含まれています。

ROM を書き換える手順を図 4.1 に示します。なお、ROM を書き換える際には、ROM へのアクセスが禁止されます。このため、書き換え制御プログラム、および書き込むデータを RAM へ配置し、動作させる必要があります。RAM へ配置する方法は、「RX ファミリ フラッシュモジュール Firmware Integration Technology」の” 2.13 RAM からコードを実行してコードフラッシュを書き換える”を参照ください。本サンプルコードでは、動作確認用にベリファイをプログラムで実行し、書き込むデータと書き込んだデータが一致した場合に LED1 を点灯するようにしています。

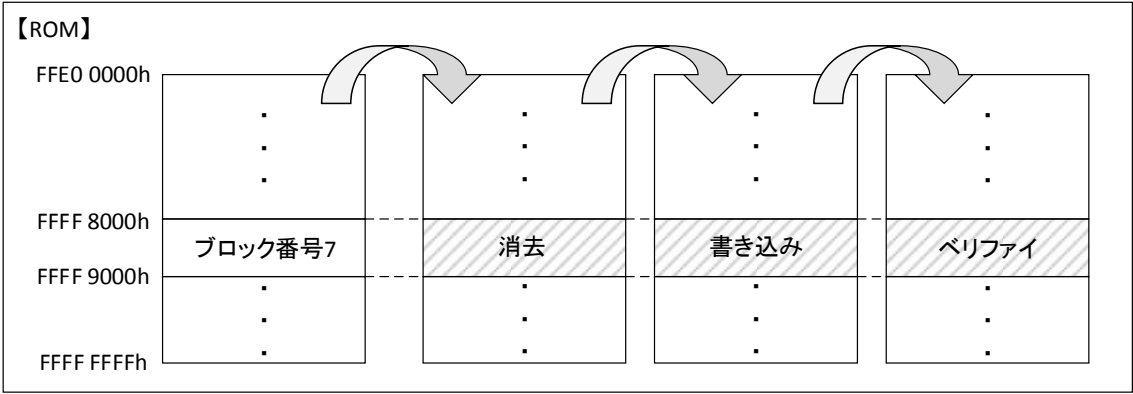


図 4.1 ROM の書き換え手順

E2 データフラッシュを書き換える手順を図 4.2 に示します。E2 データフラッシュを書き換える際には、ROM へのアクセスは可能ですが、読み出し、および P/E 許可・禁止を制御する必要があります。また、消去状態は不定値が読み出され、ブランクチェック機能により対象領域が消去されていることを確認します。本サンプルコードでは、動作確認用にベリファイをプログラムで実行し、書き込むデータと書き込んだデータが一致した場合に LED2 を点灯するようにしています。

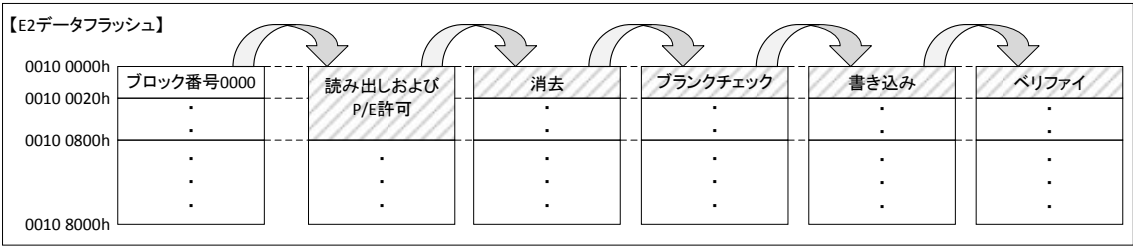


図 4.2 E2 データフラッシュの書き換え手順

4.2 ファイル構成

本サンプルコードを作成するにあたり、編集したファイルを表 4.1 に示します。6. FIT モジュールの組み込みでダウンロードした FIT モジュールを組み込んだファイルについては割愛します。

表 4.1 ファイル名一覧

ファイル名	概要	備考
FIT_Flash_RX63N.c	メインファイル <ul style="list-style-type: none">Flash API の FIT モジュール初期化指定した ROM 領域の消去指定した ROM 領域のブランクチェック指定した ROM 領域の書き込み <ul style="list-style-type: none">E2 データフラッシュへのアクセスを許可指定した E2 データフラッシュ領域の消去指定した E2 データフラッシュ領域のブランクチェック指定した E2 データフラッシュ領域の書き込み	
hwsetup.c	初期設定 <ul style="list-style-type: none">LED 制御用ポートの設定	r_bsp¥board_rdkr x63n¥hwsetup.c 内 の output_ports_con figure 関数 を HSBRX63NP 用に変更しています

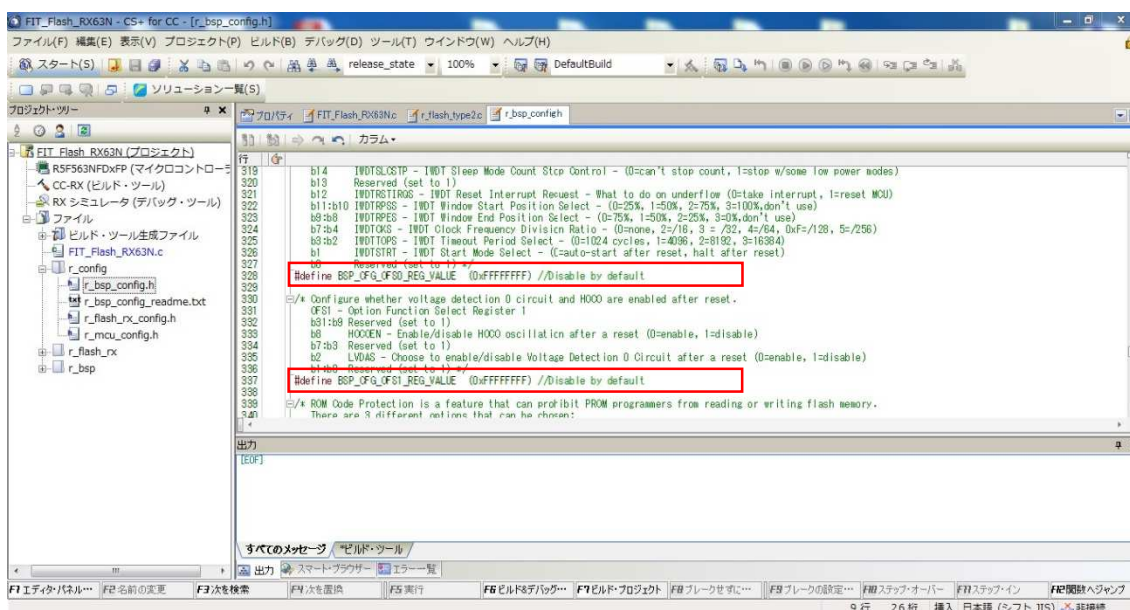
4.3 オプション設定メモリ

表 4.2 に本サンプルコードで使用するオプション設定メモリの状態を示します。

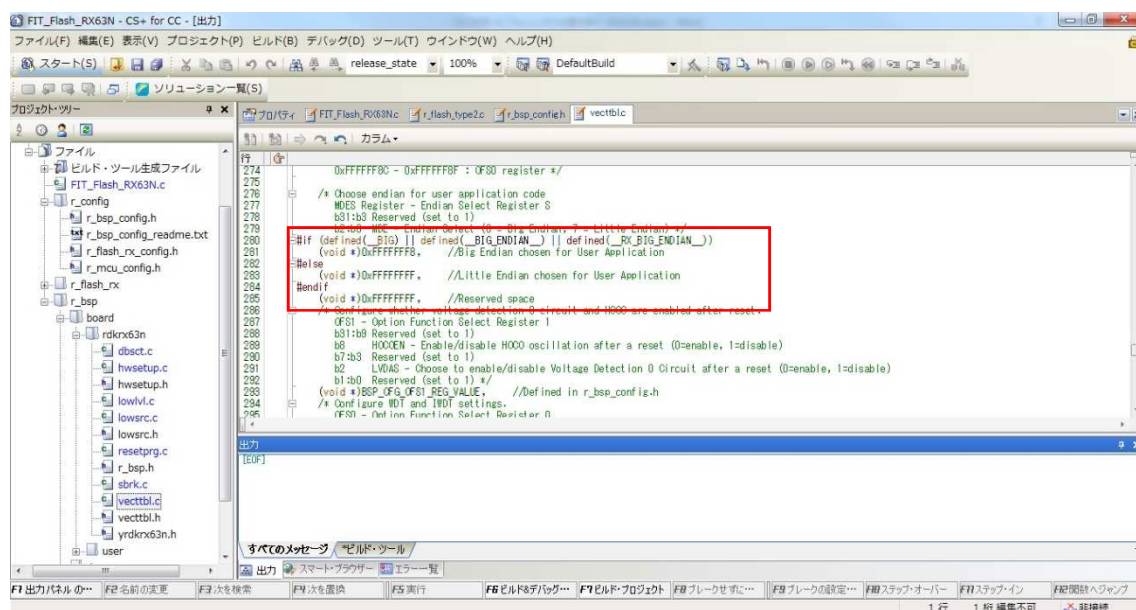
表 4.2 オプション設定メモリー一覧

シンボル	アドレス	設定値	内容
OFS0	FFFF FF8Fh~FFFF FF8Ch	FFFF FFFFh	リセット後、IWDTP は停止 リセット後、WDT は停止
OFS1	FFFF FF8Bh~FFFF FF88h	FFFF FFFFh	リセット後、 電圧監視 0 リセット無効 HOCO(高速オンチップオシレータ)発振が無効
MDES	FFFF FF83h~FFFF FF80h	FFFF FFFFh	リトルエンディアン

OFS0 と OFS1 は r_config\bsp_config.h(「6. FIT モジュールの組み込み」で組み込んだファイル)に定義されています。



MDES は r_bsp¥board¥rdkrx63n¥vecttbl.c (6. の手順で組み込んだファイル) に定義されています。



4.4 定数一覧

表 4.3 に本サンプルコードで使用する定数一覧を示します。

表 4.3 サンプルコードで使用する定数

定数名	設定値	内容
ACCESS_DF_EN_0000_0063BLOCK	0001h	E2 データフラッシュ 0000-0063 ブロックの読み出しおよび P/E を許可する設定値
CF_TABLE_SIZE	ROM_PROGRAM_SIZE	ROM に書き込むデータのテーブルサイズ
CF_WRITE_SIZE	ROM_PROGRAM_SIZE	ROM に書き込むデータのサイズ
DF_WRITE_SIZE	FLASH_DF_BLOCK_SIZE	E2 データフラッシュに書き込むデータサイズ

4.5 enum 一覧

表 4.4 に本サンプルコードで使用する enum を示します。

表 4.4 サンプルコードで使用する enum

型	列挙定数	内容
flash_target_t	TARGET_CODE_FLASH=0, TARGET_DATA_FLASH	フラッシュメモリの ROM または E2 データフラッシュの判別

4.6 変数一覧

表 4.5 に本サンプルコードで使用する変数一覧を示します。

表 4.5 サンプルコードで使用する変数

型	変数名	内容
const uint8_t	cf_data_table[CF_TABLE_SIZE]	ROM データテーブル ※ブロック番号 7 の先頭番地 から配置
const uint8_t	df_write_data[DF_WRITE_SIZE]	E2 データフラッシュに書き込 むデータのテーブル
uint8_t	cf_write_data[CF_WRITE_SIZE]	ROM に書き込むデータのテー ブル ※初期化済み変数で RAM へ配 置

4.7 関数一覧

表 4.6 に関数一覧を掲載します。本サンプルコードで新規作成、編集した関数のみ記載しています。フラッシュモジュール FIT の関数に関しましては、「RX ファミリ フラッシュモジュール Firmware Integration Technology」を参照ください。

表 4.6 関数一覧

関数名	概要
main	メイン処理
er_wr_flash	ROM および E2 データフラッシュの書き換え処理

4.8 関数仕様

main

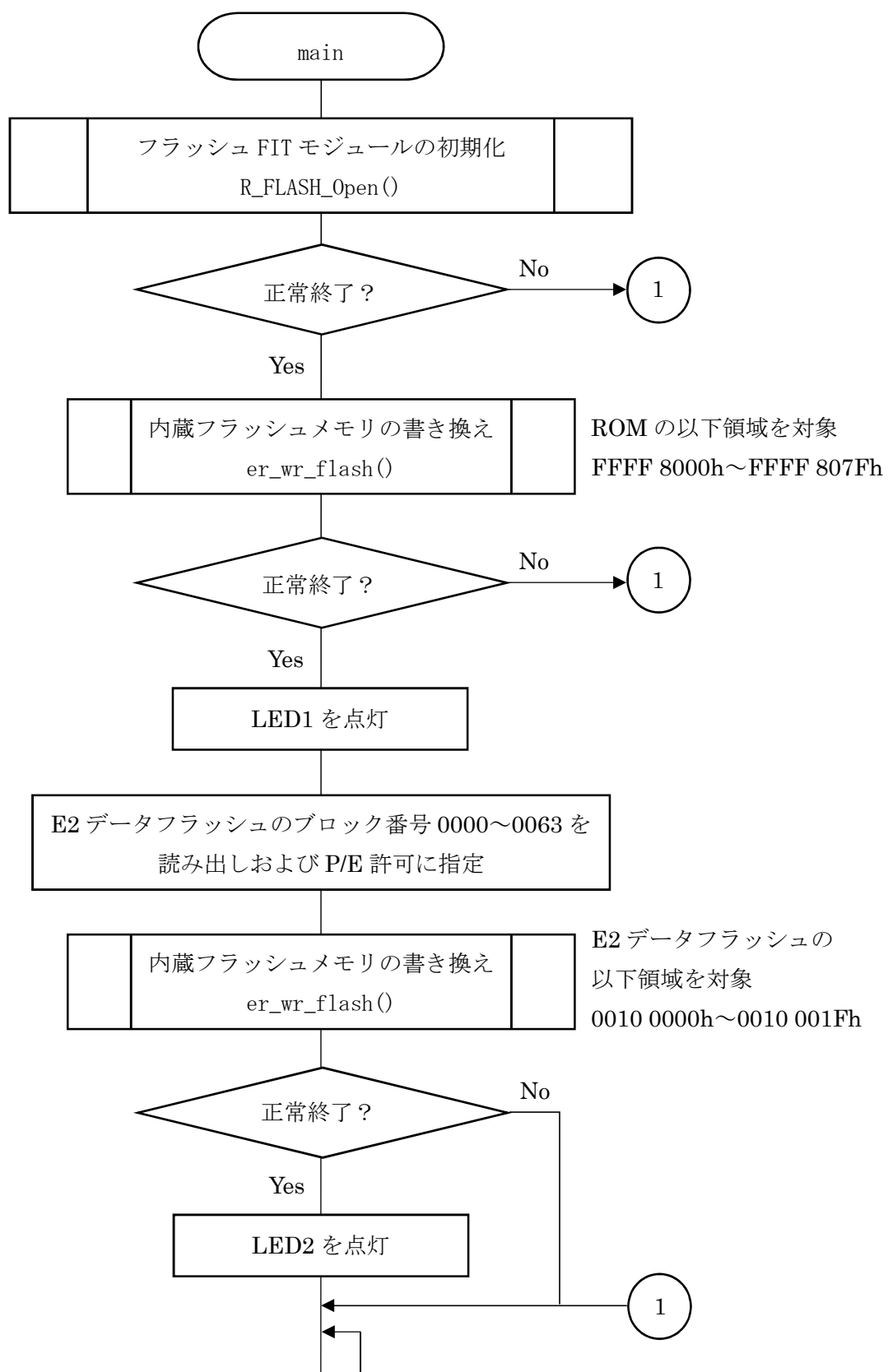
概要	メイン処理
ヘッダ	なし
宣言	void main(void)
説明	ROM および E2 データフラッシュの書き換え処理関数の呼び出し
引数	なし
リターン値	なし

er_wr_flash

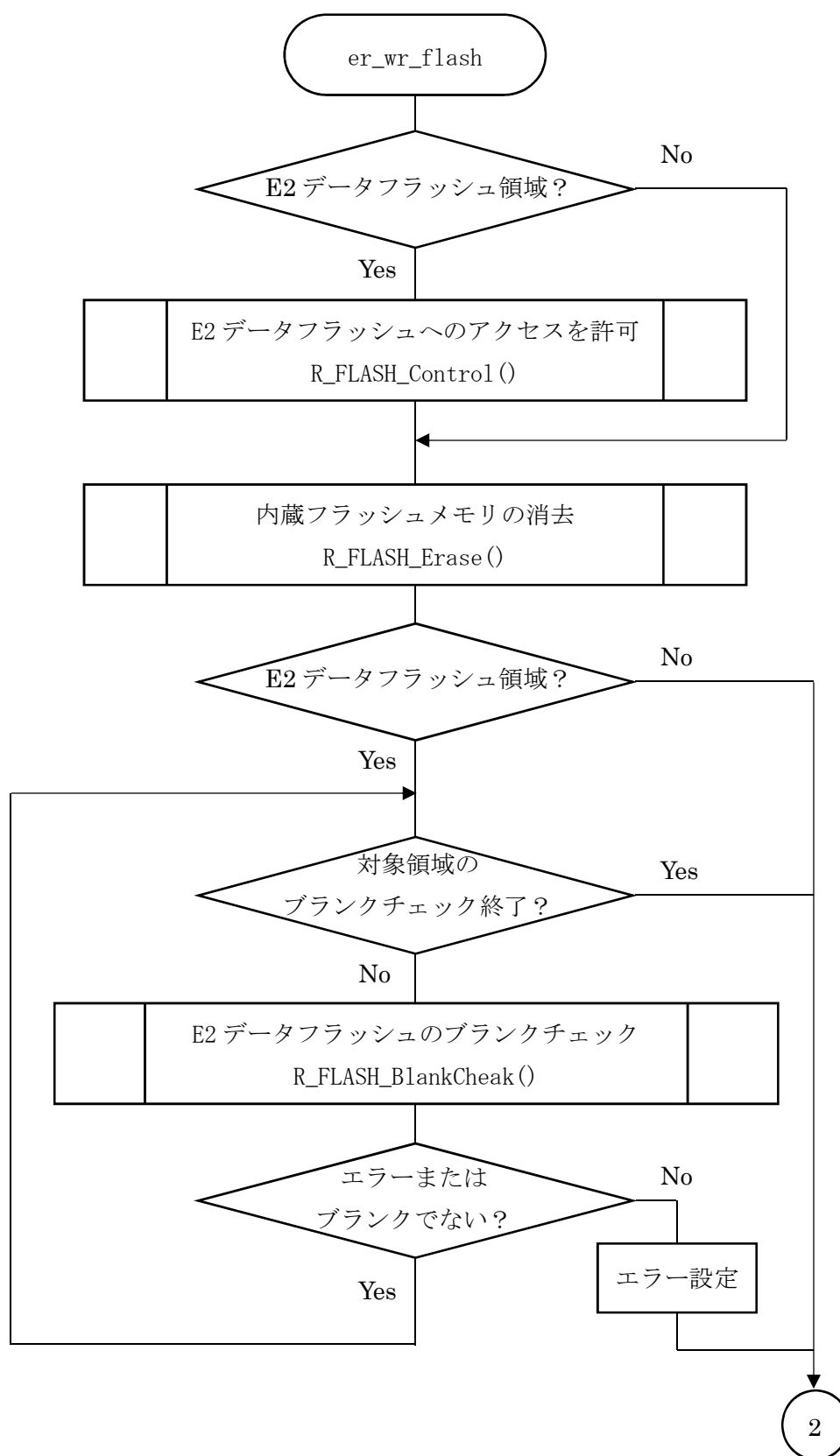
概要	ROM および E2 データフラッシュの書き換え処理
ヘッダ	なし
宣言	<pre>flash_err_t er_wr_flash (flash_access_window_config_t access_info, flash_block_address_t block_start_adr, uint32_t erase_num_blocks, uint32_t src_adr, uint32_t dest_adr, uint32_t write_size)</pre>
説明	Flash API 関数呼び出し、およびベリファイ
引数	<p>flash_access_window_config_t access_info : E2 データフラッシュのアクセス制御</p> <p>flash_block_address_t block_start_adr : 対象のブロック先頭アドレス</p> <p>uint32_t erase_num_blocks : 消去対象のブロック数</p> <p>uint32_t src_adr : 書き込むデータが配置されている先頭アドレス</p> <p>uint32_t dest_adr : 書き換え対象の先頭アドレス</p> <p>uint32_t write_size : 書き込むデータサイズ</p>
リターン値	<p>実行結果</p> <p>FLASH_SUCCESS : 正常終了</p> <p>FLASH_ERR_BUSY : フラッシュモジュールがビジー状態</p> <p>FLASH_ERR_ACCESSW : アクセスウィンドウのエラー</p> <p>FLASH_ERR_FAILURE : プログラミングエラー、イレーズエラー、ブランクチェックエラー、ベリファイエラー</p> <p>FLASH_ERR_BLOCKS : 指定されたブロック数のエラー</p> <p>FLASH_ERR_ADDRESS : 指定されたアドレスのエラー</p> <p>FLASH_ERR_BYTES : 指定されたブロック数、プログラムサイズのエラー</p> <p>FLASH_ERR_NULL_PTR : 指定されたコマンドに使用する引数” pcfg” が NULL</p> <p>FLASH_ERR_LOCKED : フラッシュの制御回路がコマンドロック状態</p> <p>FLASH_ERR_PARAM : 指定されたコマンドが無効</p>

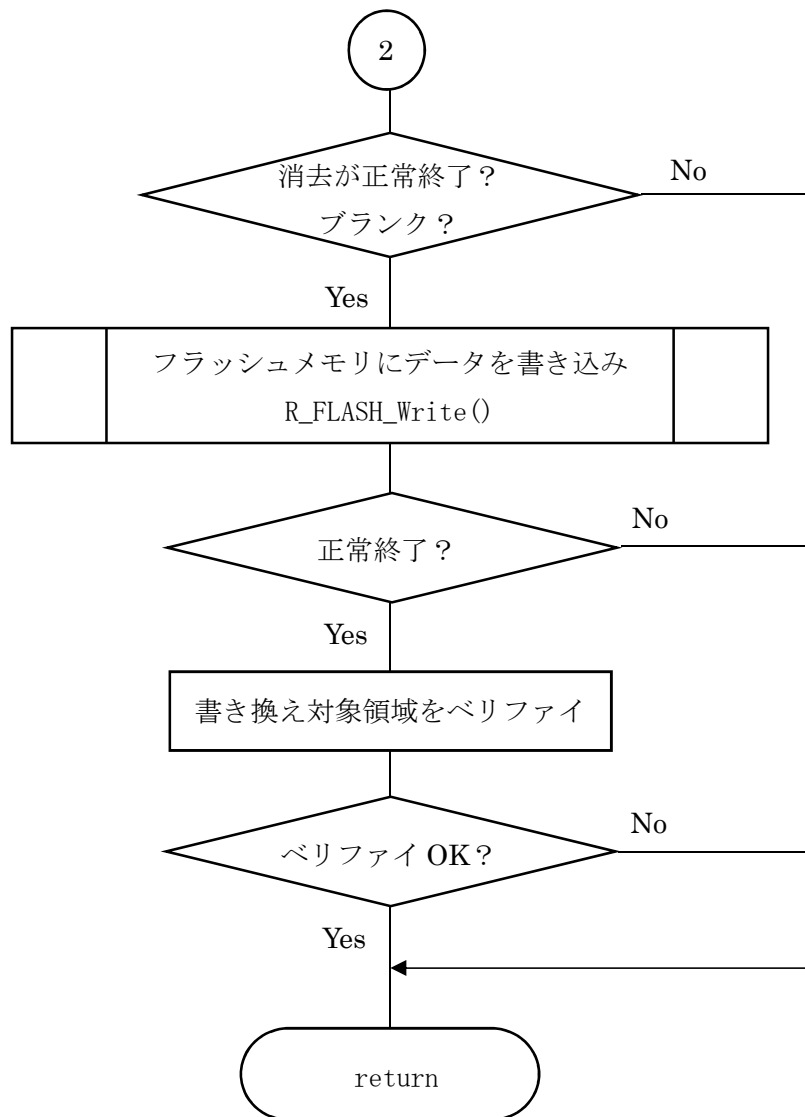
4.9 作成する関数のフローチャート

4.9.1 メイン処理



4.9.2 内蔵フラッシュメモリの書き換え





5. FIT モジュールのダウンロード方法

FIT モジュールをダウンロードの手順を説明します。

ルネサスエレクトロニクスの公式 HP にアクセスします。アクセスしましたら、右上にある Search をクリックし、「Firmware Integration Technology」と入力し、検索します。



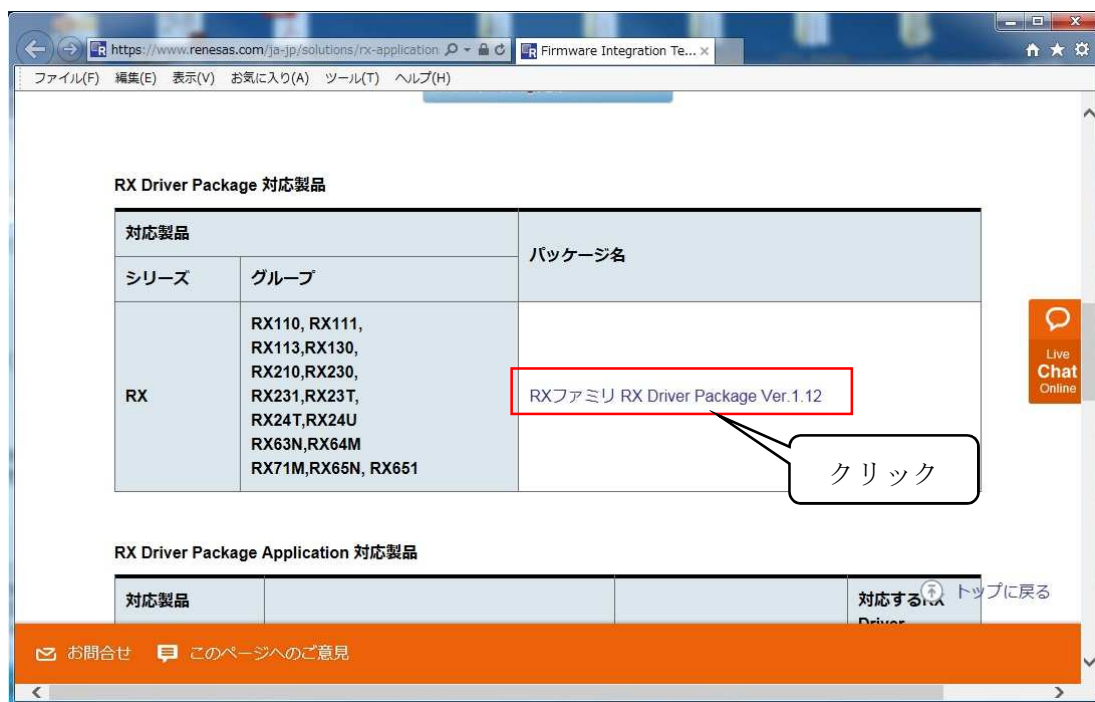
検索後に出てくる画面を下にスクロールして、「Firmware Integration Technology (FIT)」をクリックします。



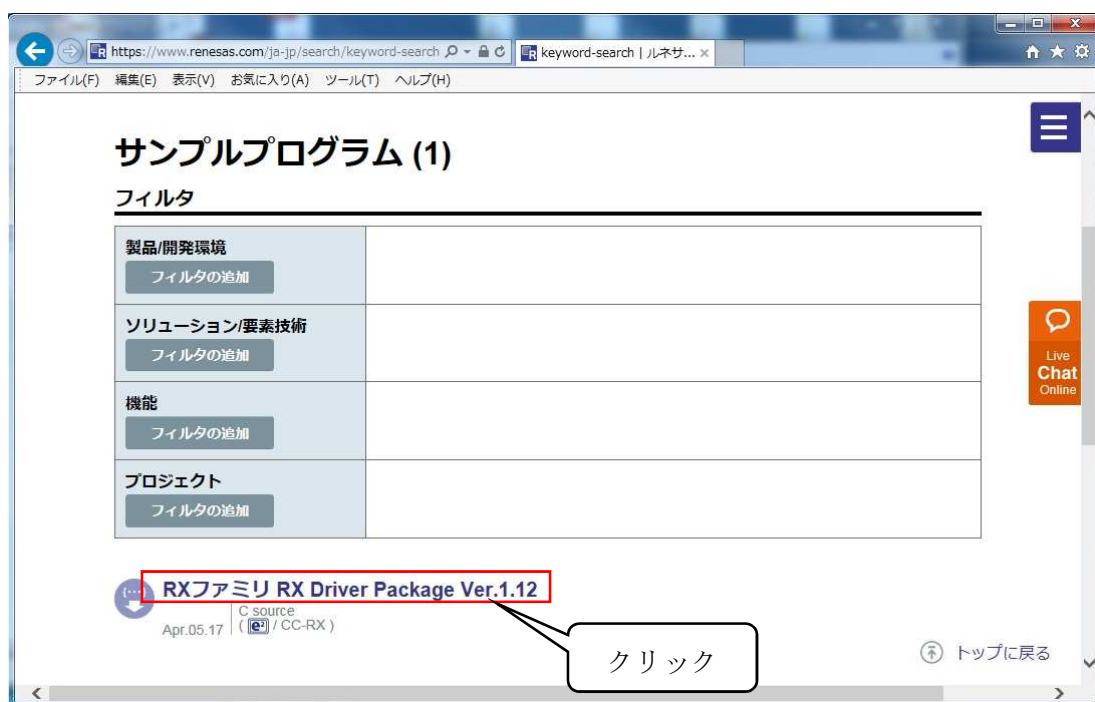
クリックすると、「Firmware Integration Technology (FIT)」のページに移動します。



下にスクロールして、「RX ファミリ RX Driver Package Ver.1.12」をクリックします。

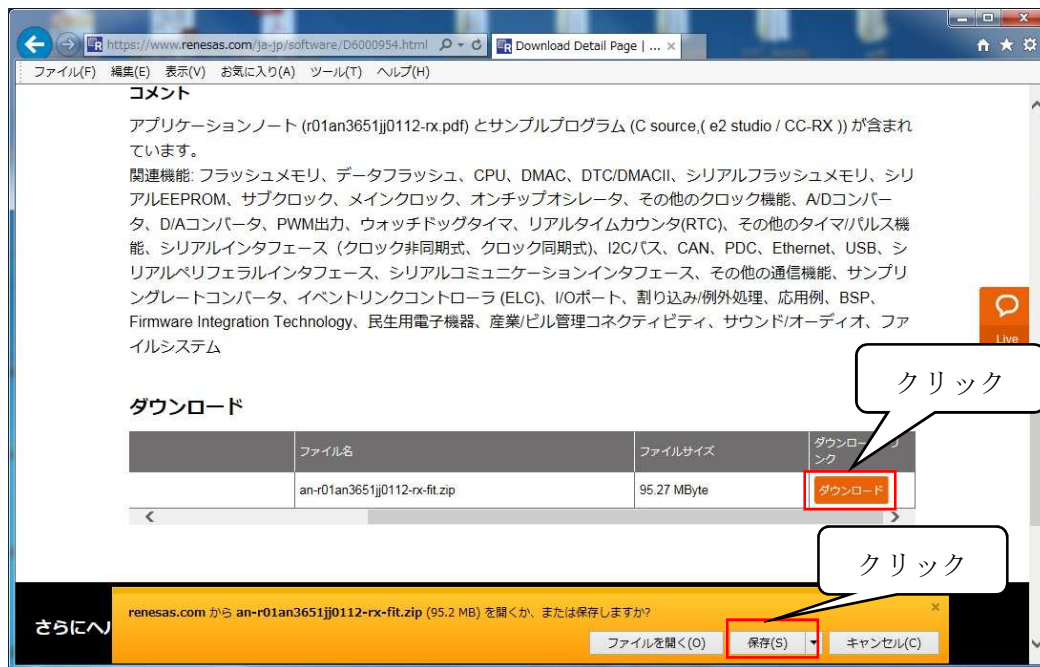


「サンプルプログラム」ページに移動されますので、下にスクロールして、「RX ファミリ RX Driver Package Ver.1.12」をクリックします。

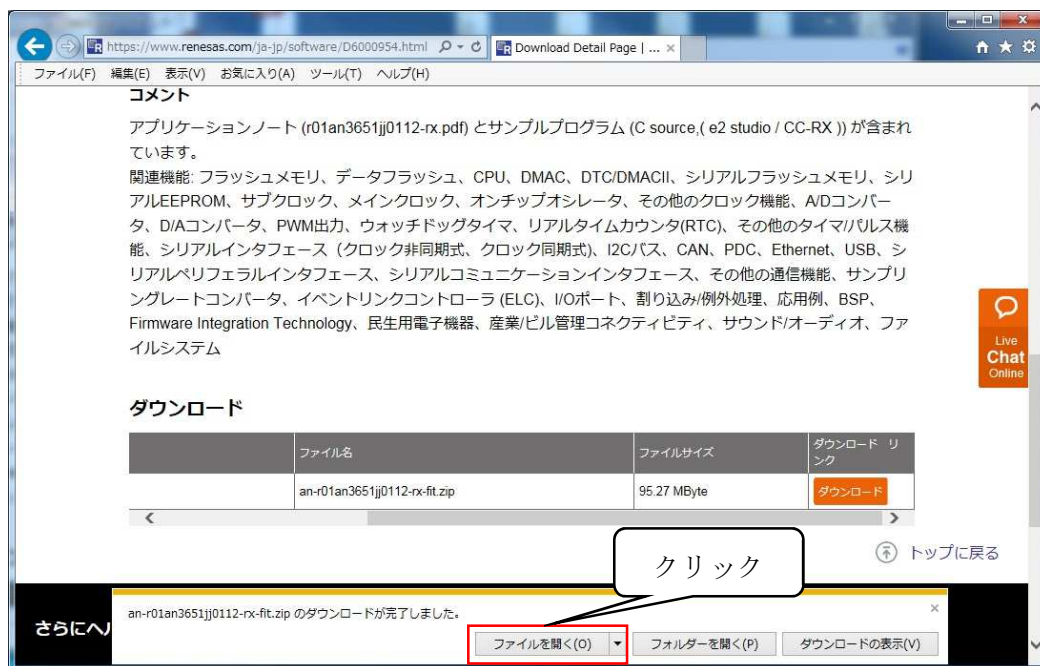


「ダウンロード」ページに移動されますので、下にスクロールして、ダウンロードをクリックします。

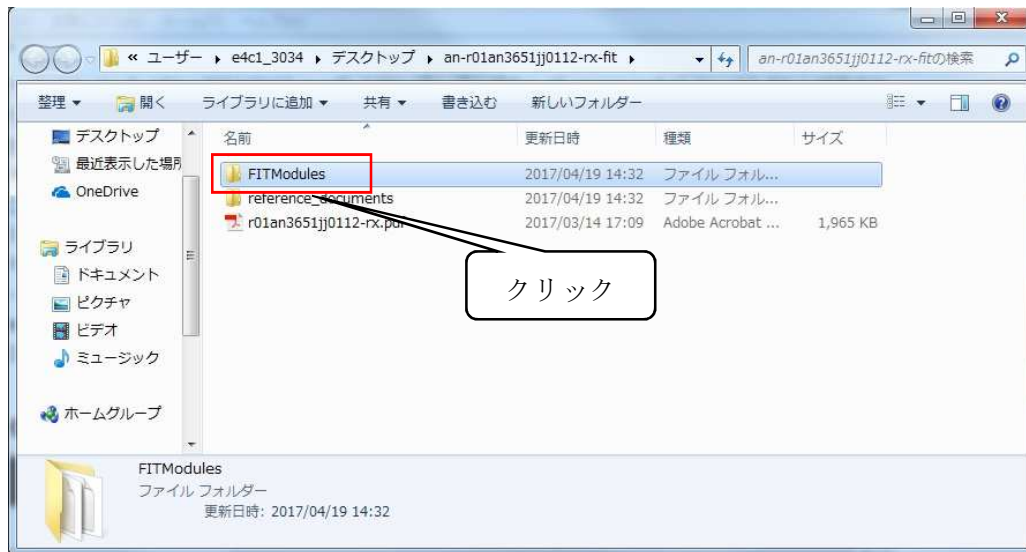
クリックすると、ダイアログが表示されますので、「保存」をクリックします。



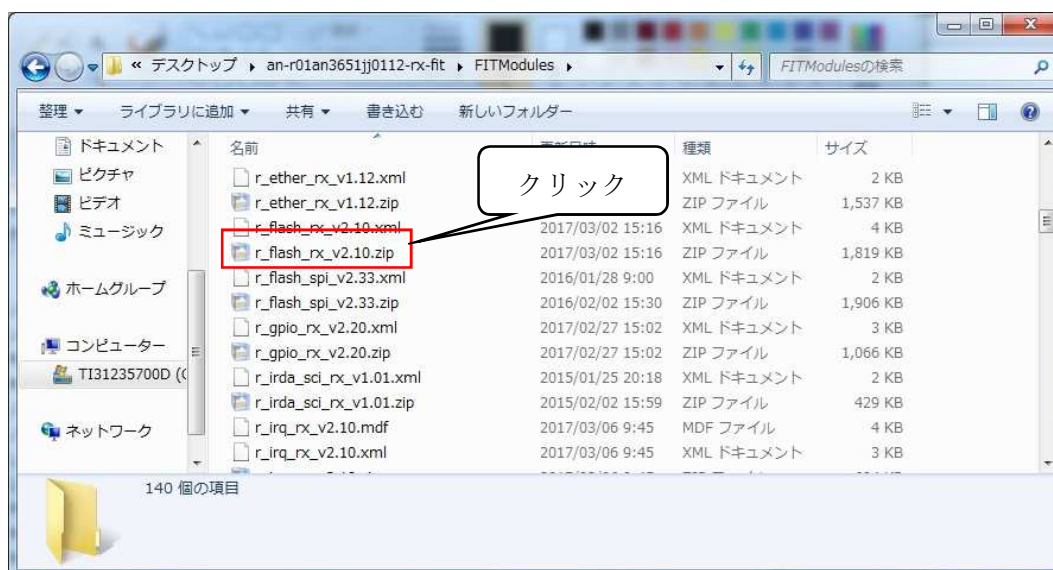
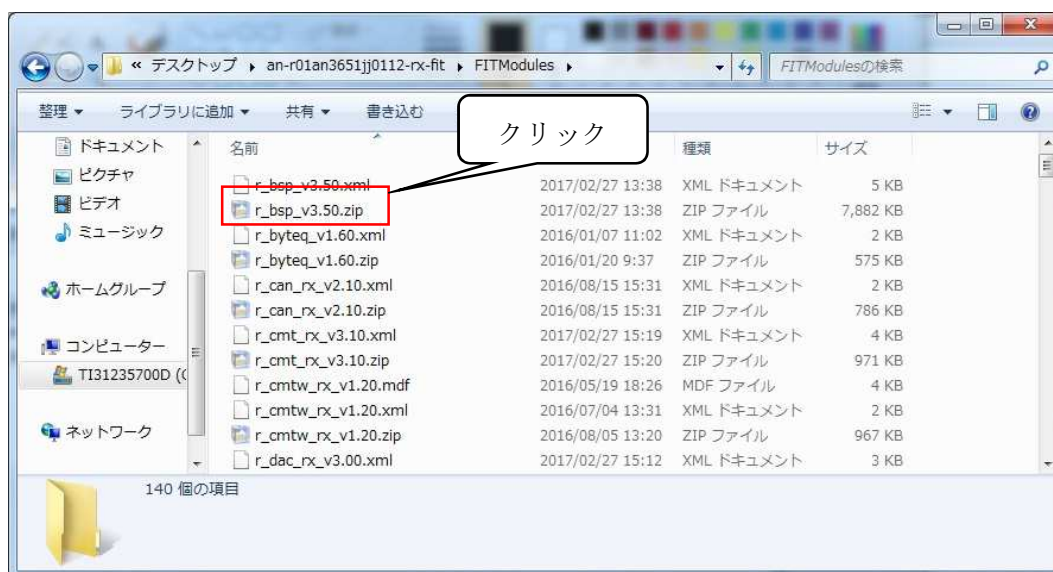
「an-r01an3651jj0112-rx-fit.zip」のダウンロードが完了すると、ダイアログが表示されますので、「ファイルを開く」をクリックし、zip ファイルを解凍します。



解凍した「an-r01an3651jj0112-rx-fit」フォルダを開き、「FITModules」フォルダに移動します。



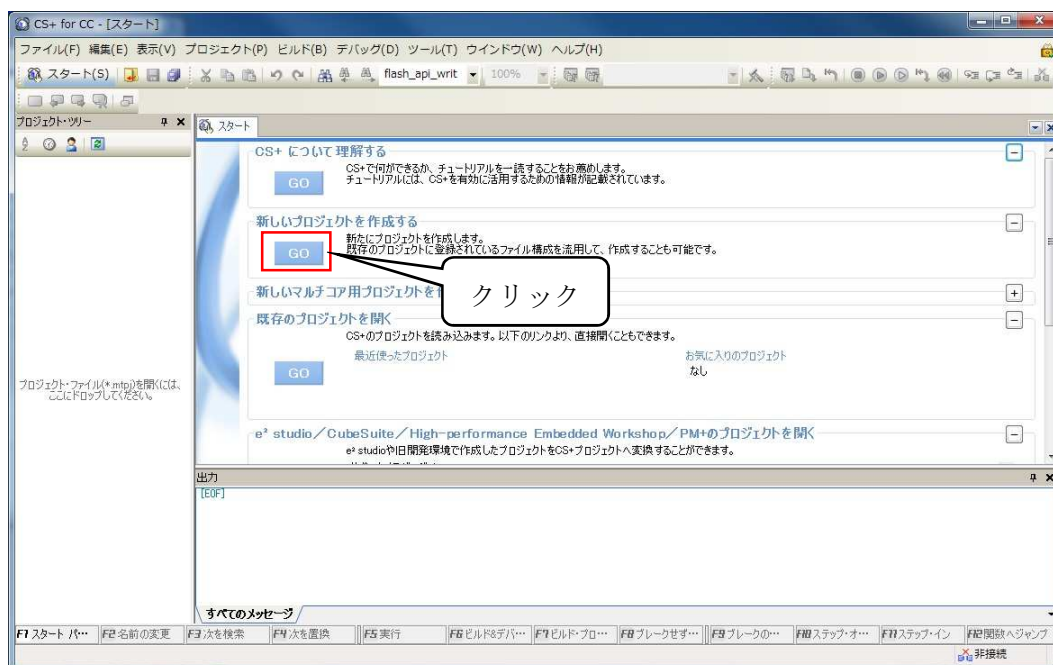
「FITModules」フォルダにある「r_bsp_v3.50.zip」と「r_flash_rx_v2.10.zip」をクリックし、zip ファイルを解凍します。



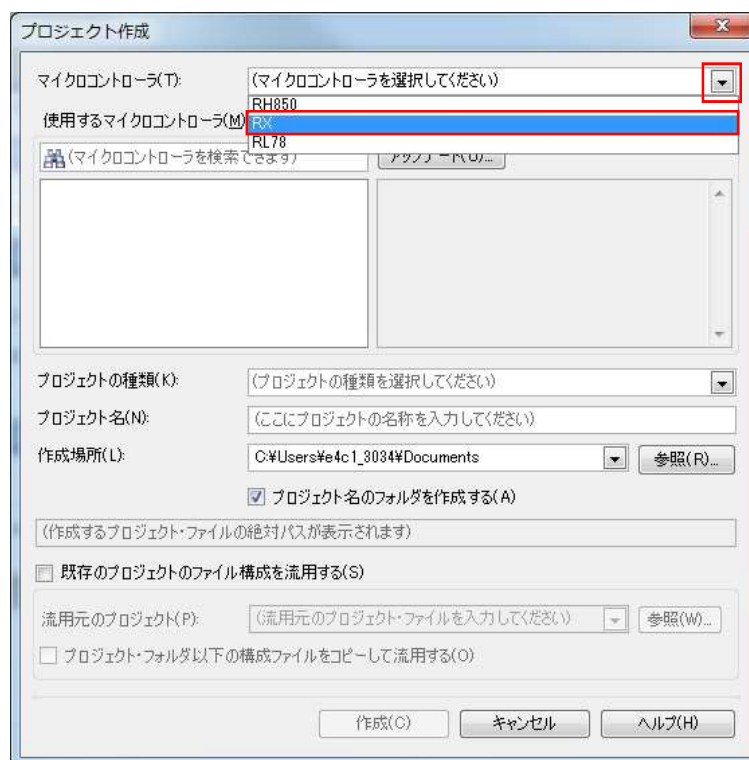
6. FIT モジュールの組み込み

ダウンロードした FIT モジュールを CS+ のプロジェクトに追加する手順を説明します。

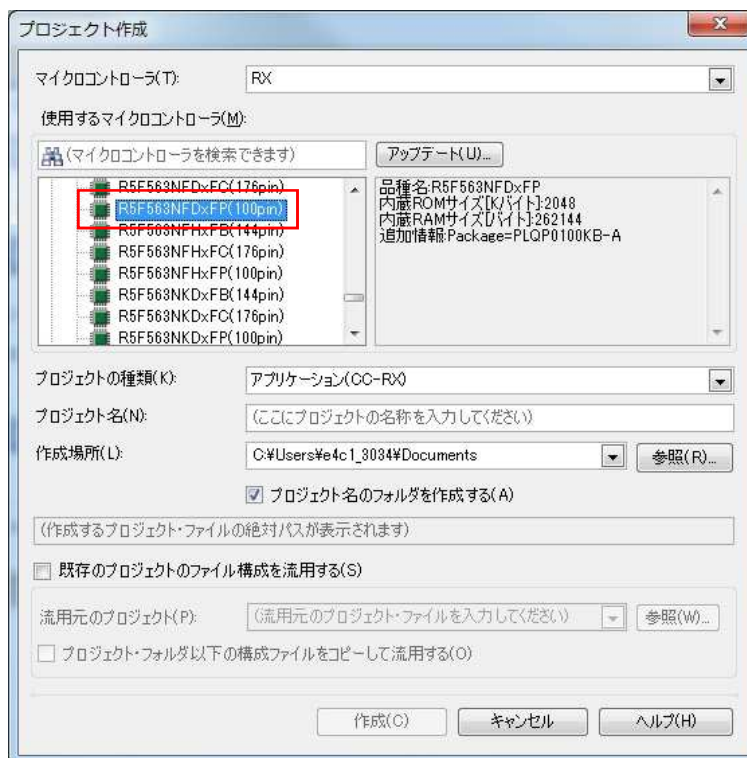
CS+ for CC を起動し、新しいプロジェクトを作成するの下にある「GO」をクリックし、新規プロジェクトを作成します。



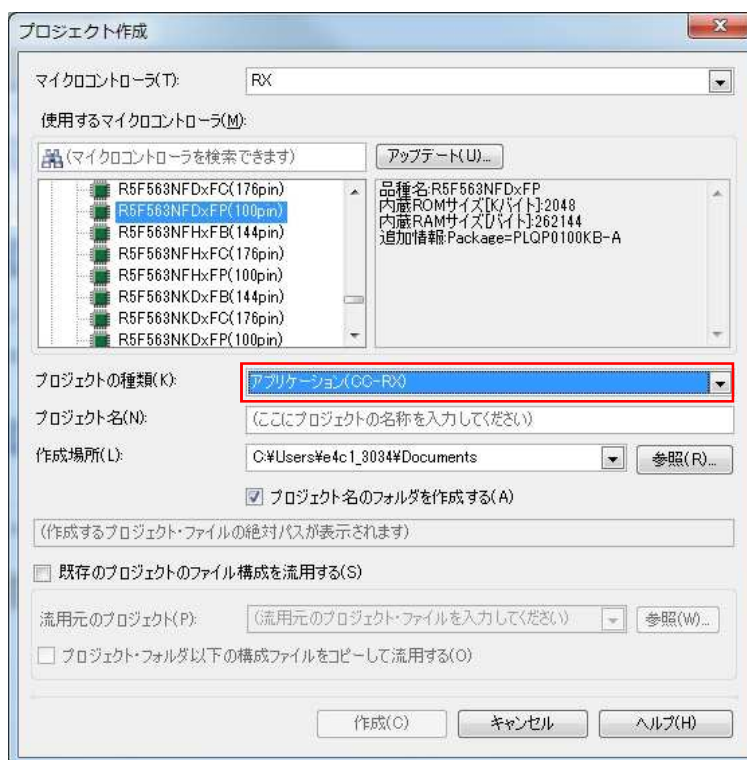
「マイクロコントローラ(T)」のプルダウンメニューから“RX”を選択します。



選択後、RX シリーズのマイコンが表示されますので、スクロールして、“RX63N” を選択します。“RX63N” をダブルクリックすると、RX63N の型名一覧が表示されますので、今回使用するマイコンである「R5F563NFDxFP」をクリックします。



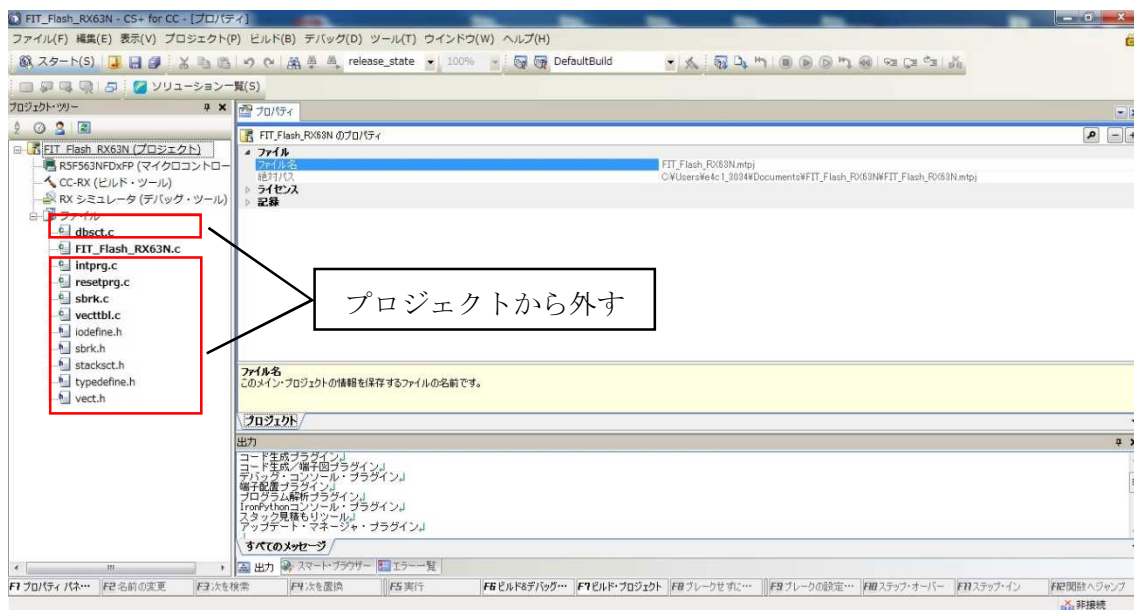
「プロジェクトの種類(K)」は“アプリケーション(CC-RX)”を選択します。



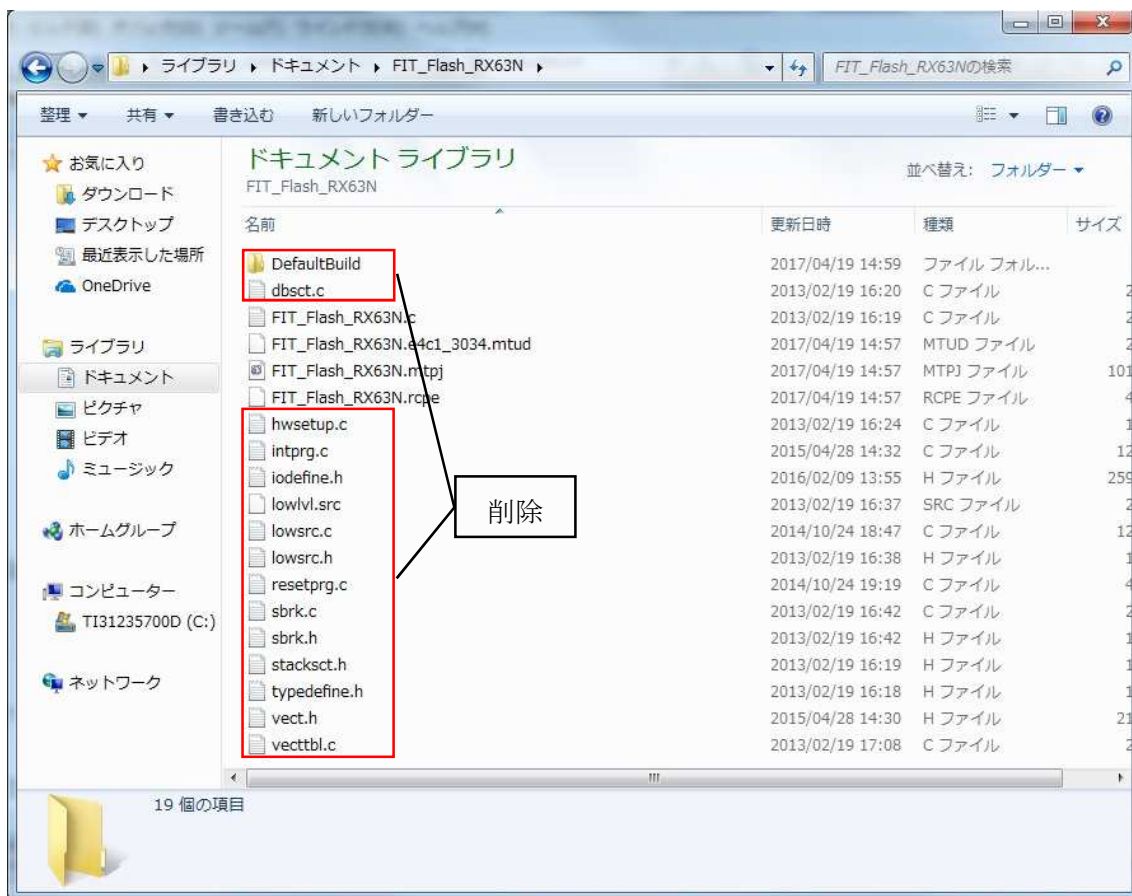
「プロジェクト名(N)」に新規作成するプロジェクト名を入力し、「作成(C)」ボタンを押してプロジェクトを作成します。ここでは、プロジェクト名を“FIT_Flash_RX63N”としています。

不要なファイルを削除します。

「プロジェクト・ツリー」画面で“FIT_Flash_RX63N.c”以外を全て選択します。ファイルを選択したまま、右クリックし「プロジェクトから外す(R)」を選択します。

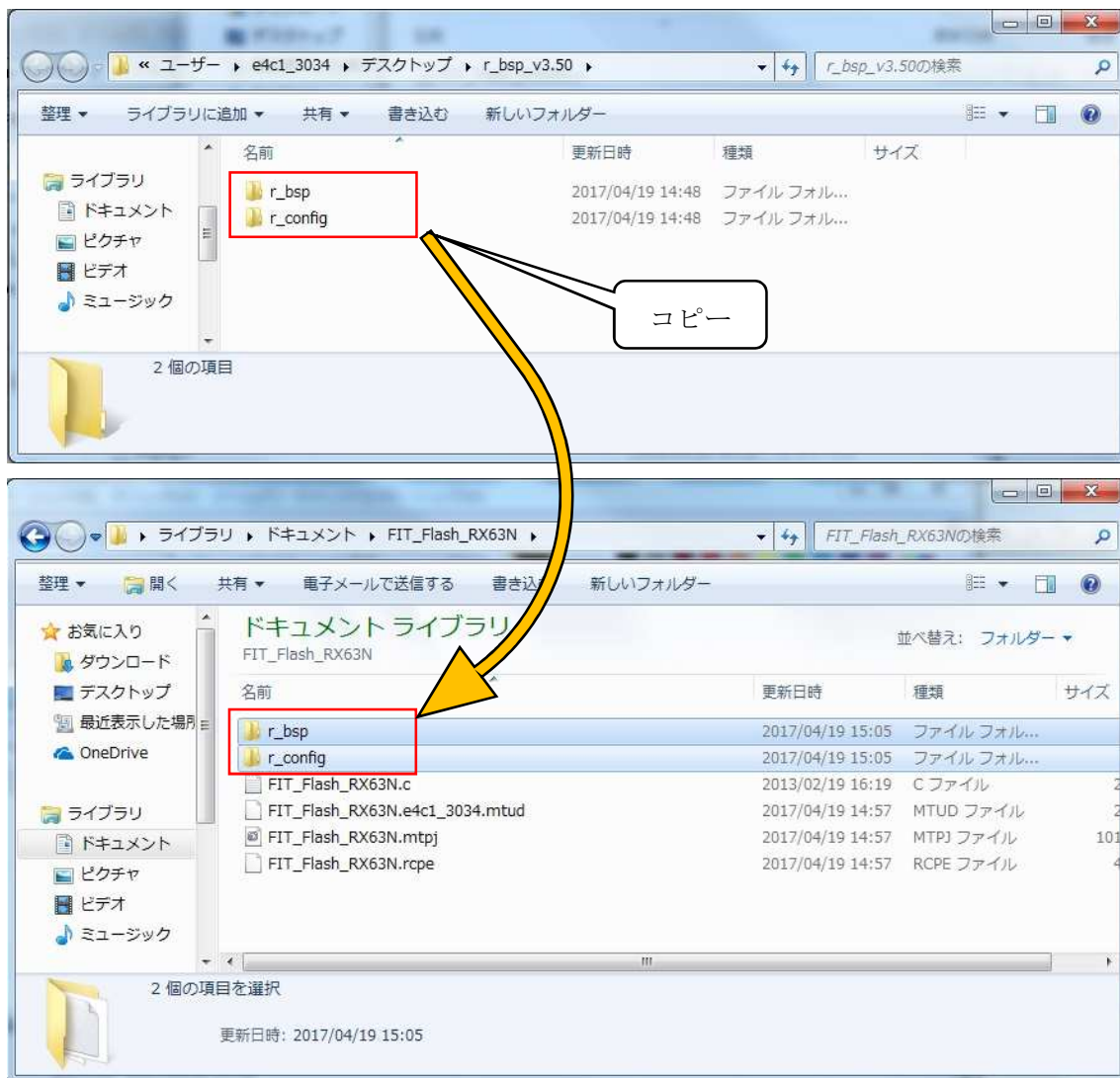


Windows のエクスプローラ側で、“FIT_Flash_RX63N.c”、“FIT_Flash_RX63N.mtud”、“FIT_Flash_RX63N.mtpj”、“FIT_Flash_RX63N.rcpe” 以外のファイルを削除します。

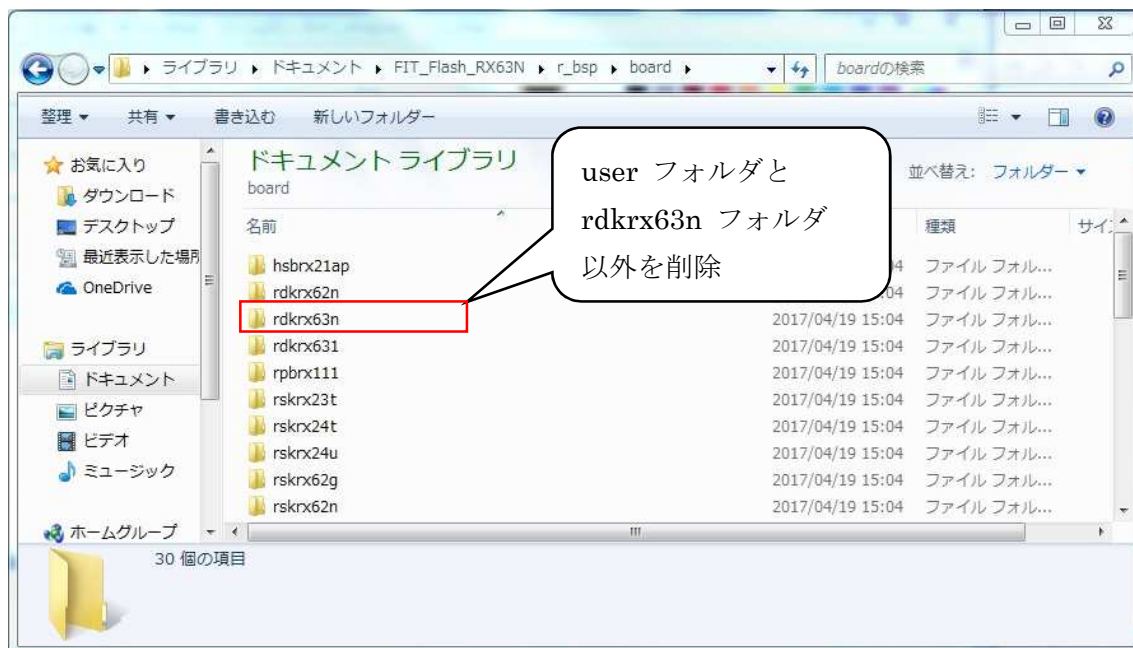


次にダウンロードした BSP の FIT モジュールを追加します。

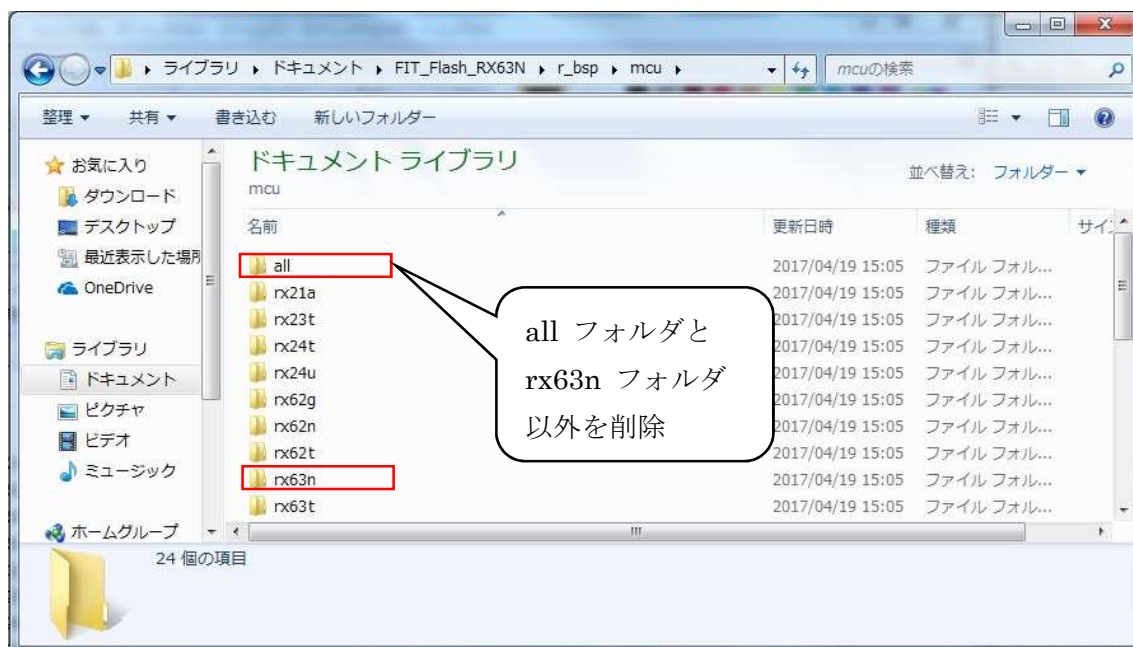
Windows のエクスプローラを使用して“FIT_Flash_RX63N”フォルダ内に解凍した r_bsp_v3.50 フォルダにある r_bsp、r_config フォルダをコピーします。



r_bsp¥board フォルダに移動し、user フォルダと rdkrx63n フォルダを除く残りのフォルダを削除します。

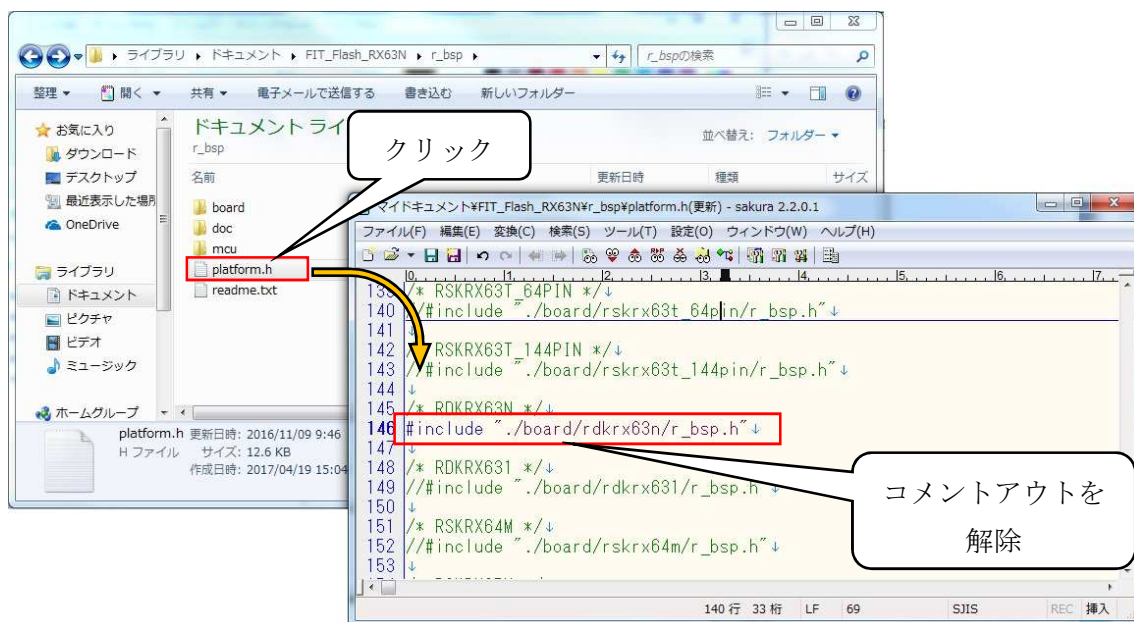


r_bsp¥mcu フォルダに移動し、all フォルダと rx63n フォルダを除く残りのフォルダを削除します。

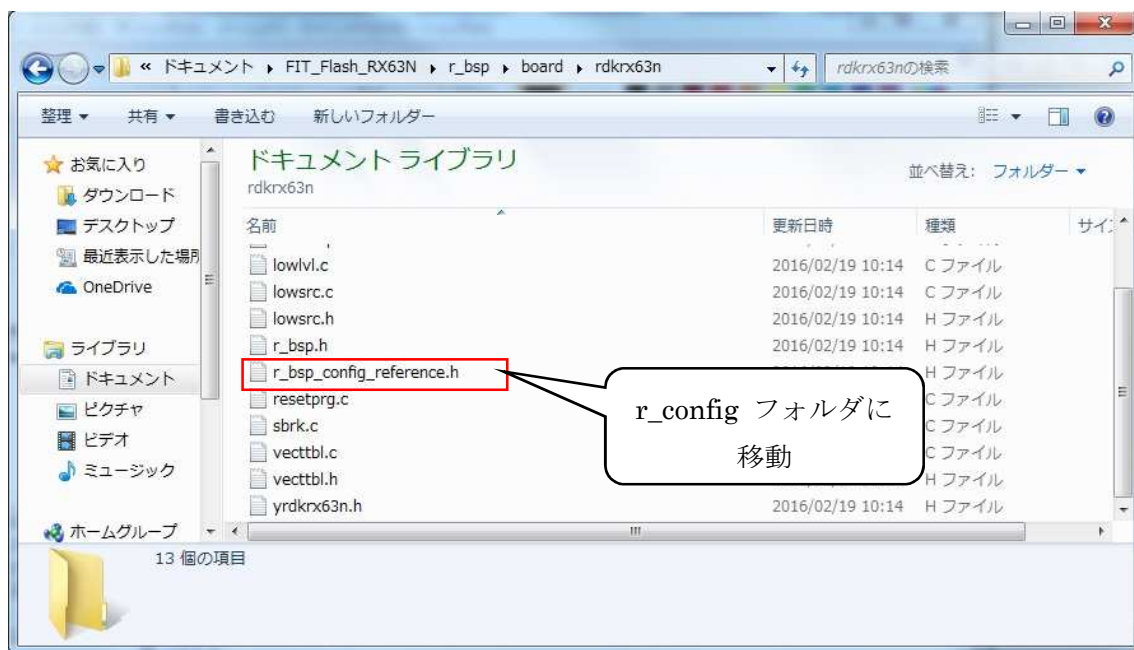


r_bsp フォルダ内にある platform.h ファイルを開きます。

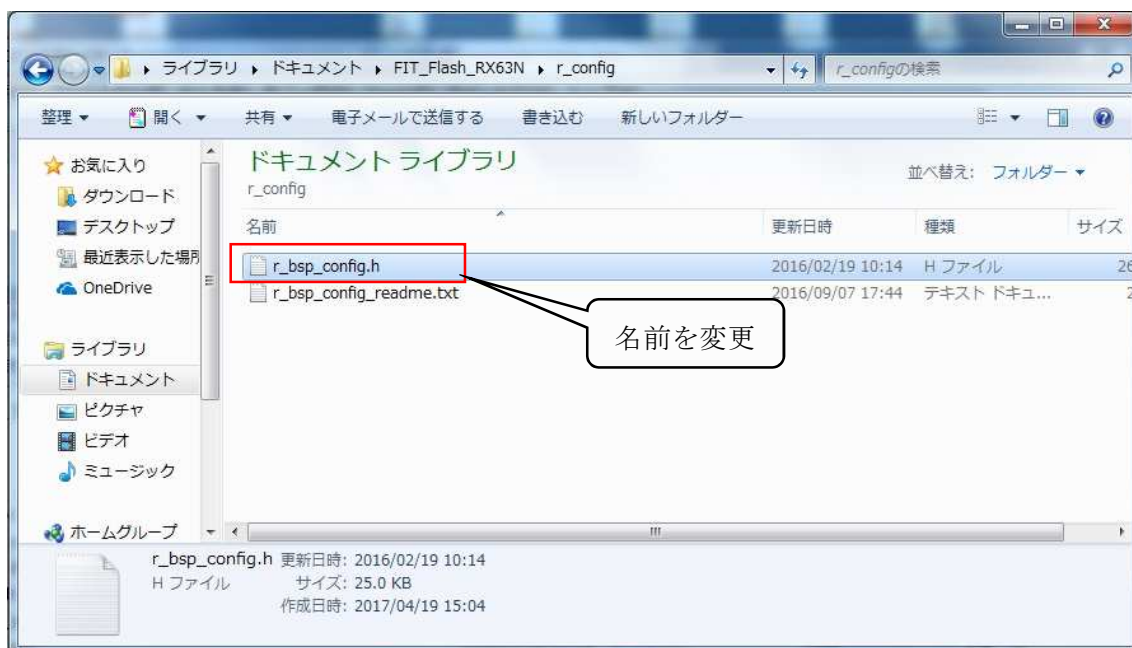
開いて、`//#include "../board/rdkrx63n/r_bsp.h"`のコメントを解除します。



r_bsp¥board¥rdkrx63n フォルダにある r_bsp_config_reference.h ファイルを r_config フォルダに移動します。

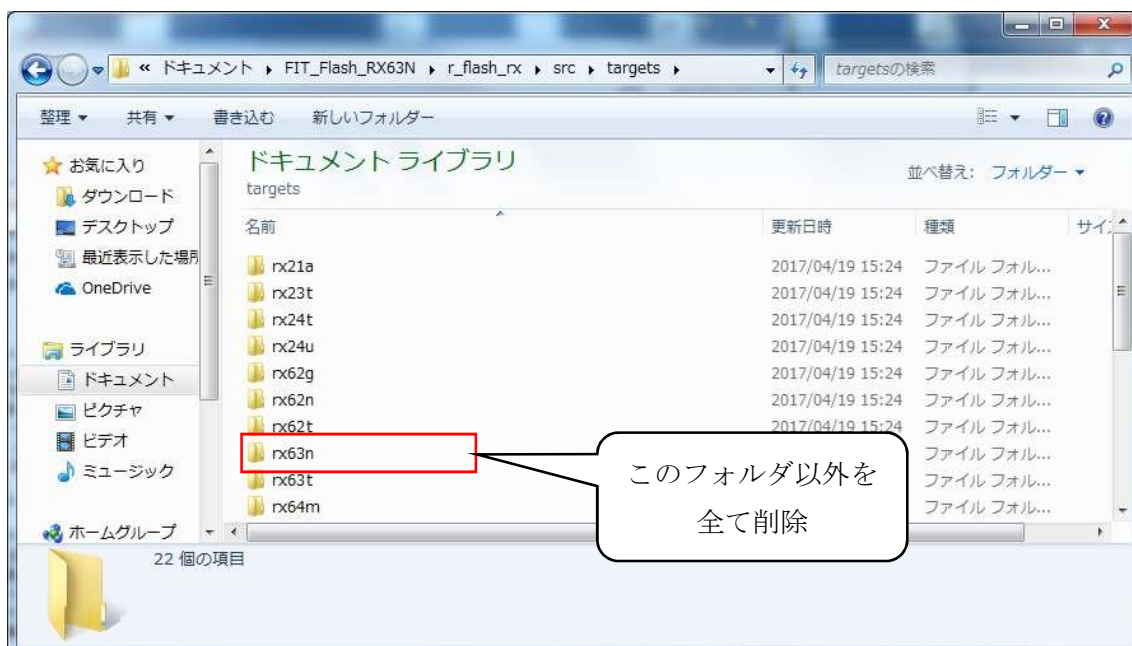


移動した `r_bsp_config_reference.h` ファイルの名前を `r_bsp_config.h` に変更します。



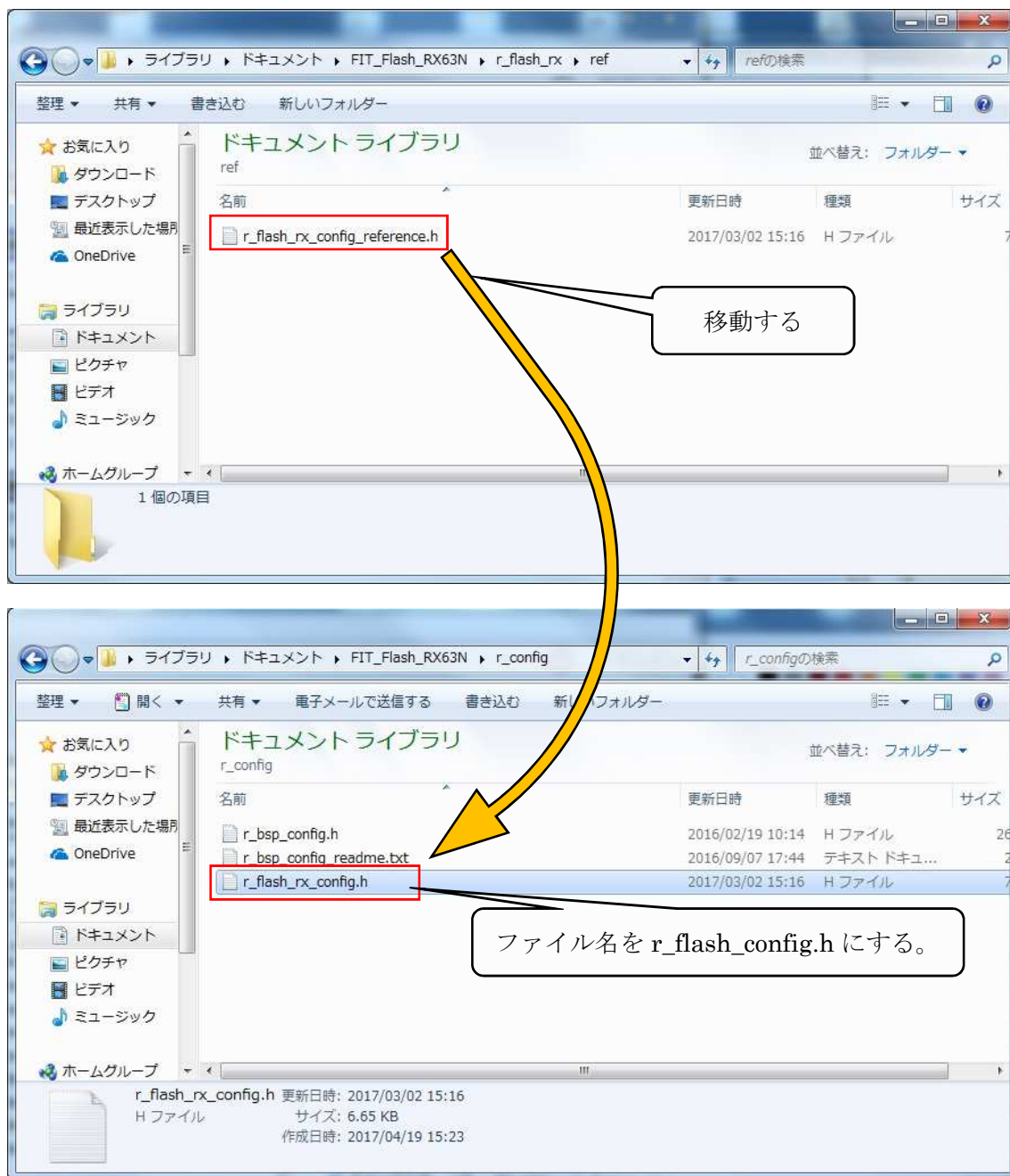
ダウンロードしたフラッシュメモリの FIT モジュールを追加します。

“FIT_Flash_RX63N” フォルダ内に FIT モジュールの `r_flash_rx` フォルダをコピーします。 `r_flash_rx\src\targets` フォルダに移動し、`rx63n` フォルダを除く残りのフォルダを削除します。



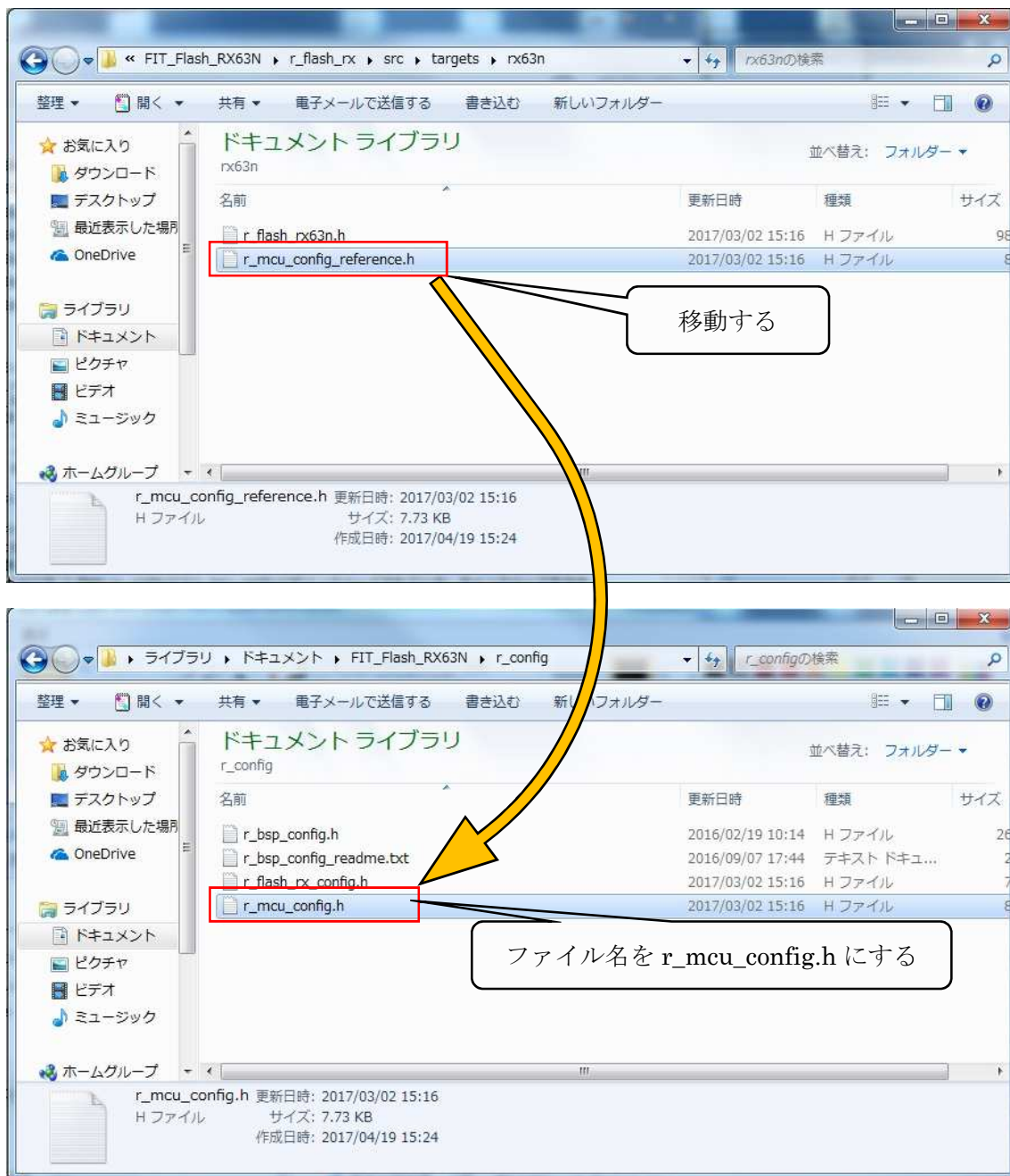
r_flash_rx¥ref¥rx63n フォルダにある r_flash_rx_config_reference.h ファイルを r_config フォルダに移動します。

移動した r_flash_rx_config_reference.h ファイルの名前を r_flash_rx_config.h に変更します。

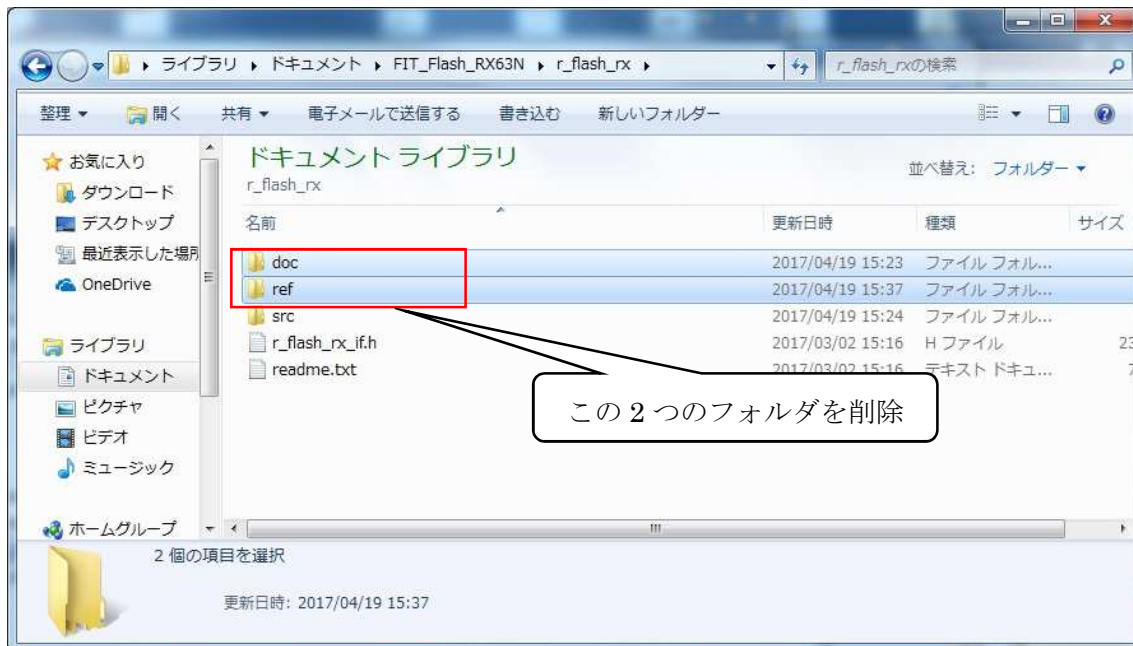


r_flash_rx¥src¥targets¥rx63n フォルダにある r_mcu_config_reference.h ファイルを r_config フォルダに移動します。

移動した r_mcu_config_reference.h ファイルの名前を r_mcu_config.h に変更します。

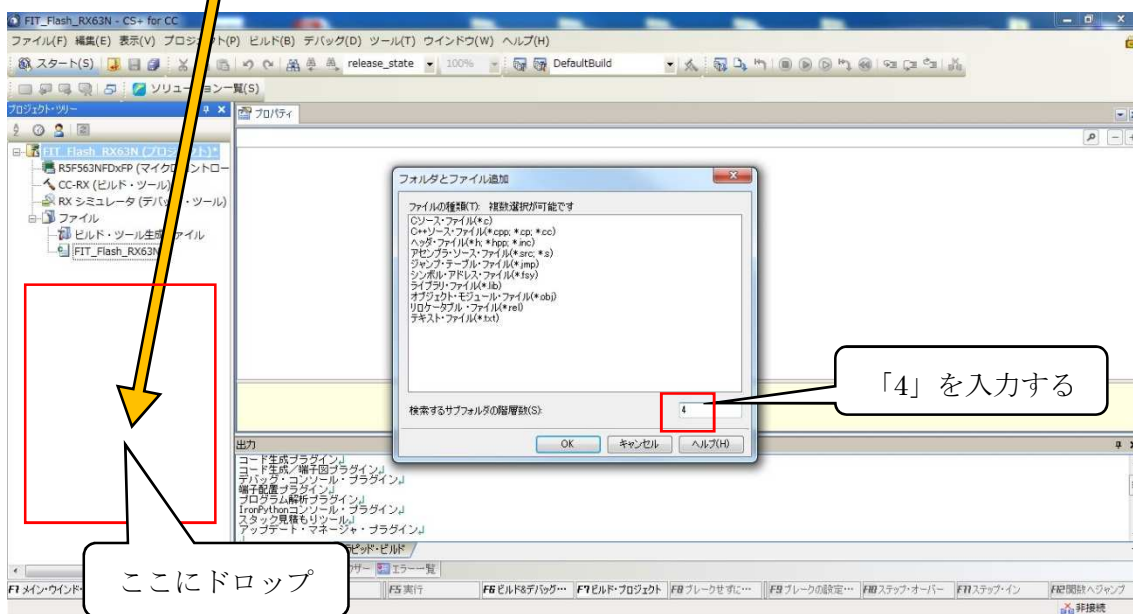
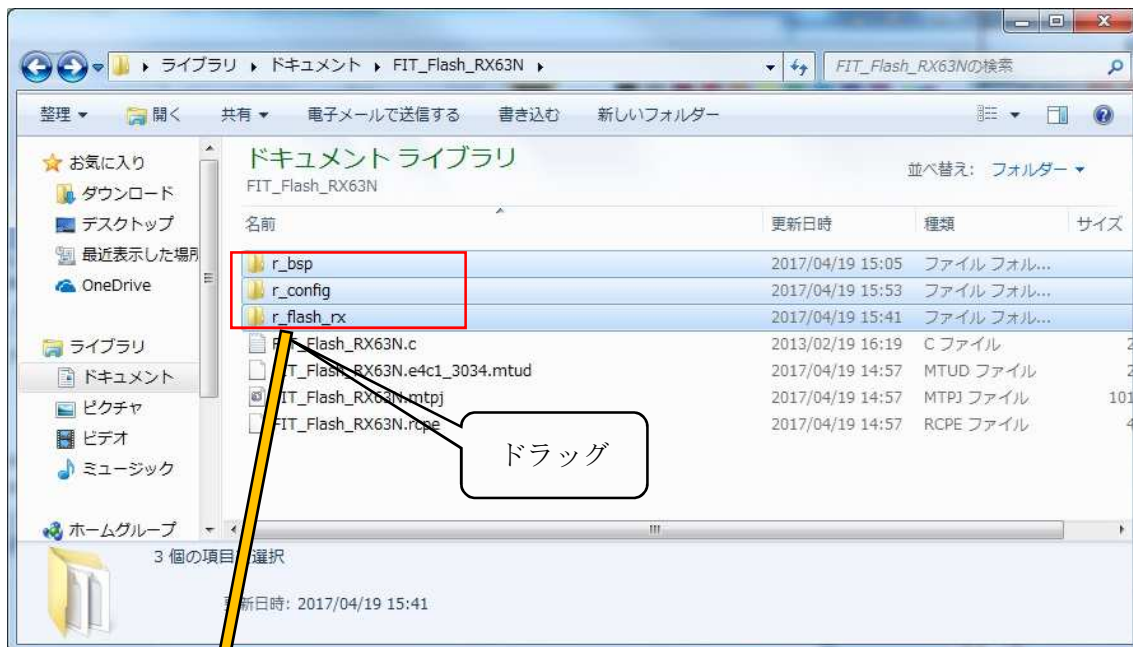


FIT_Flash_63N フォルダに戻り、r_flash_rx にある doc フォルダと ref フォルダを削除します。

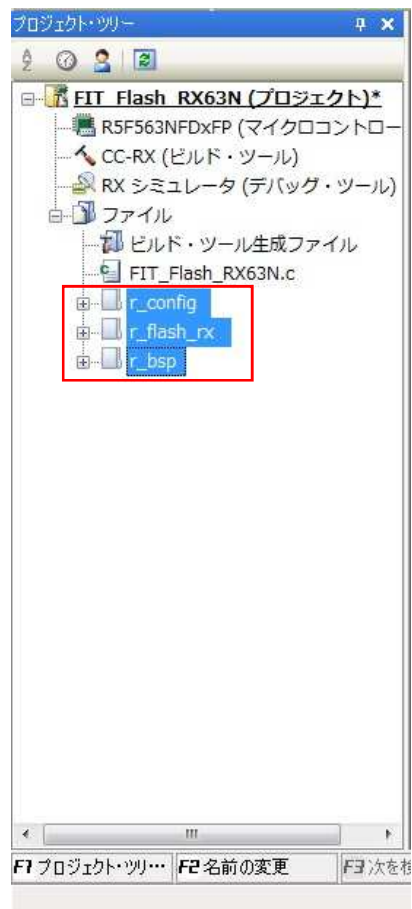


作成した CS+プロジェクトのプロジェクト・ツリー・パネルに“FIT”フォルダをドラッグ&ドロップします。

ドロップすると「フォルダとファイル追加」のダイアログが表示されます。「検索するサブフォルダの階層数(S)」に「4」を入力し、「OK」を押してください。



すると、FIT モジュールが CS+ のプロジェクト・ツリーに追加されます。



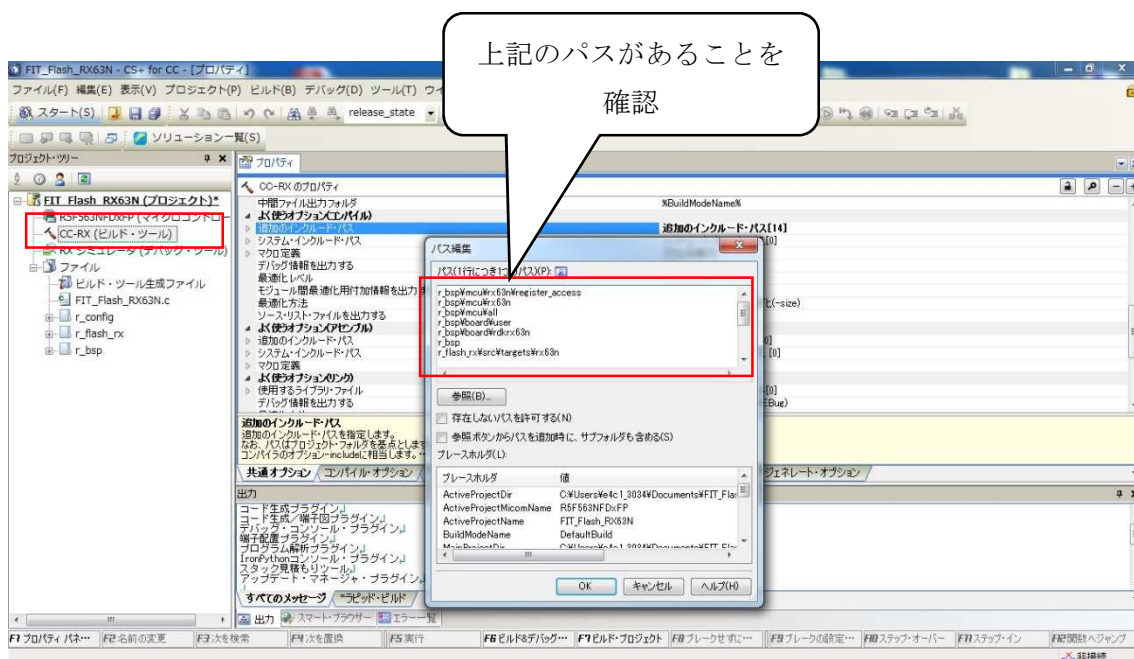
7. CC-RX の設定

7.1 インクルード・パスの確認

CS+のプロジェクト・ツリー・パネルで「CC-RX (ビルド・ツール)」をダブルクリックしてプロパティを開きます。

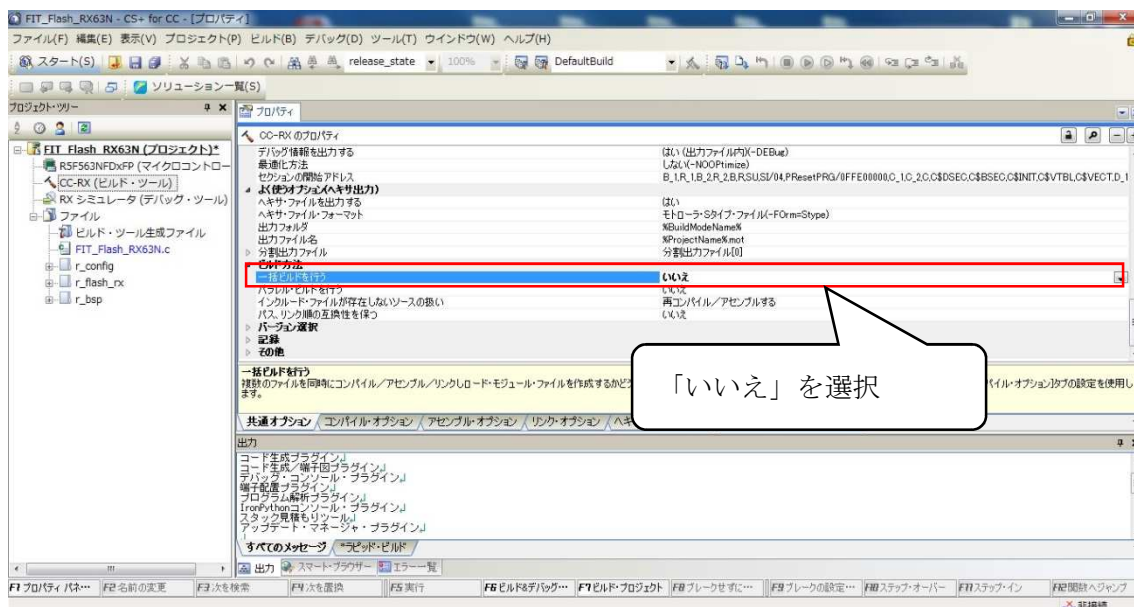
「共通オプション」タブで「追加のインクルード・パス」を選択し、「パス編集」ダイアログで、以下のパスがあることを確認します。

- ・ `r_bsp¥mcu¥rx63n¥register_access`
- ・ `r_bsp¥mcu¥rx63n`
- ・ `r_bsp¥mcu¥all`
- ・ `r_bsp¥board¥user`
- ・ `r_bsp¥board¥rdkrx63n`
- ・ `r_bsp`
- ・ `r_flash_rx¥src¥targets¥rx63n`
- ・ `r_flash_rx¥src¥flash_type_2`
- ・ `r_flash_rx¥src`
- ・ `r_flash_rx`
- ・ `r_config`



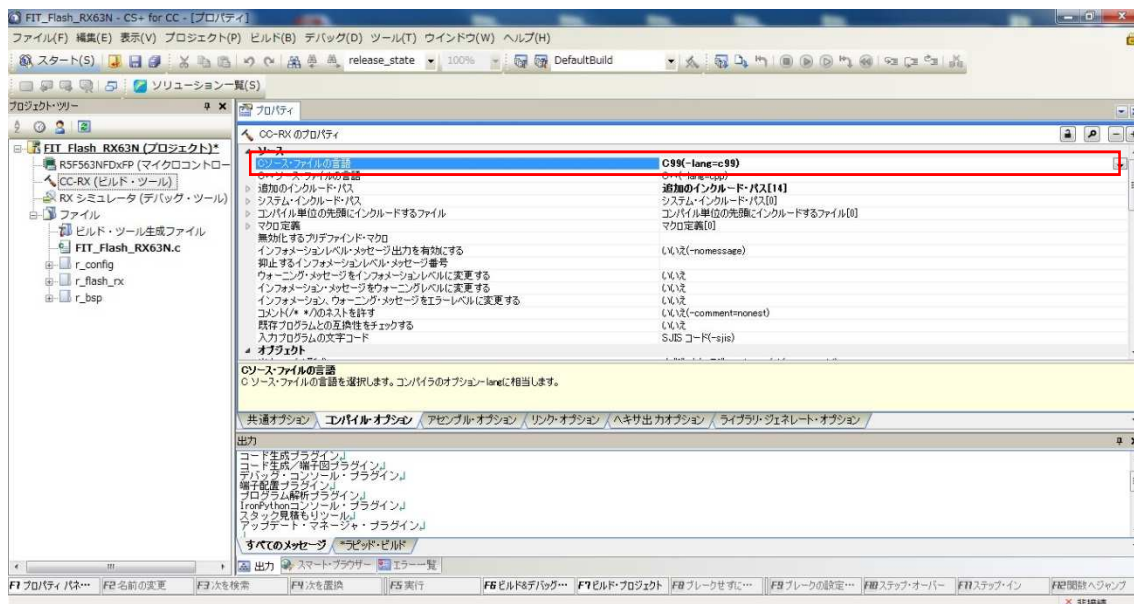
7.2 ビルド方法の変更

CS+のプロパティ・パネルで「共通オプション」タブを選択し、「ビルド方法」の「一括ビルドを行う」を“いいえ”に設定します。



7.3 C 言語規格の変更

CS+のプロパティ・パネルで「コンパイル・オプション」タブを選択し、「ソース」の「Cソース・ファイルの言語」を“C99(-lang=c99)”にします。

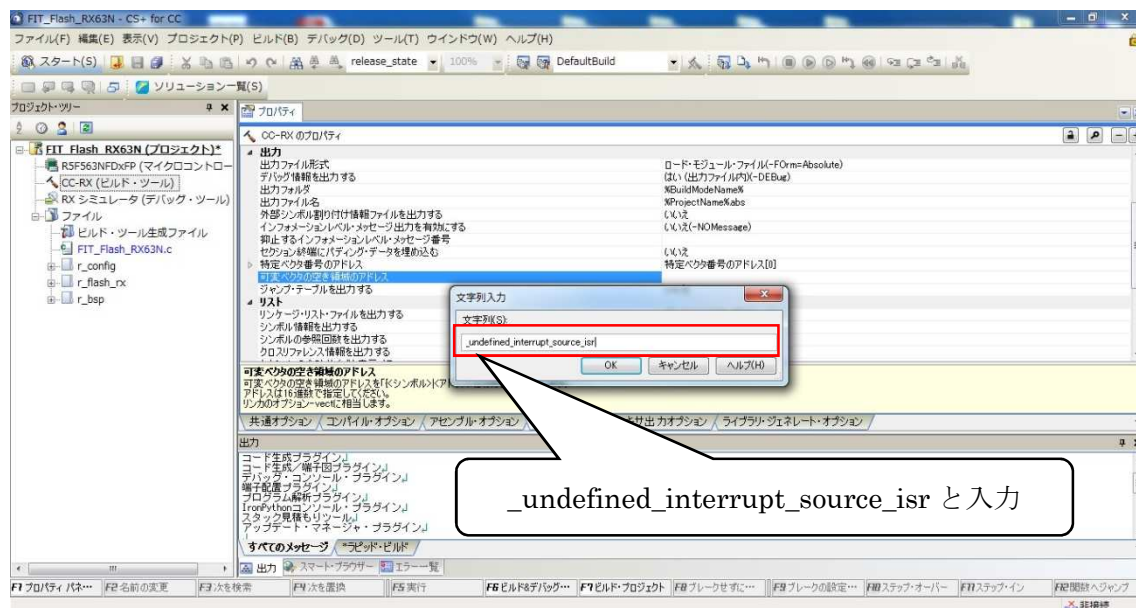


7.4 可変ベクタ空き領域の設定

使用しない割り込みの割り込みベクタ領域を `undefined_interrupt_source_isr()` 関数への割り込みベクタで埋めるためにリンクを設定します。

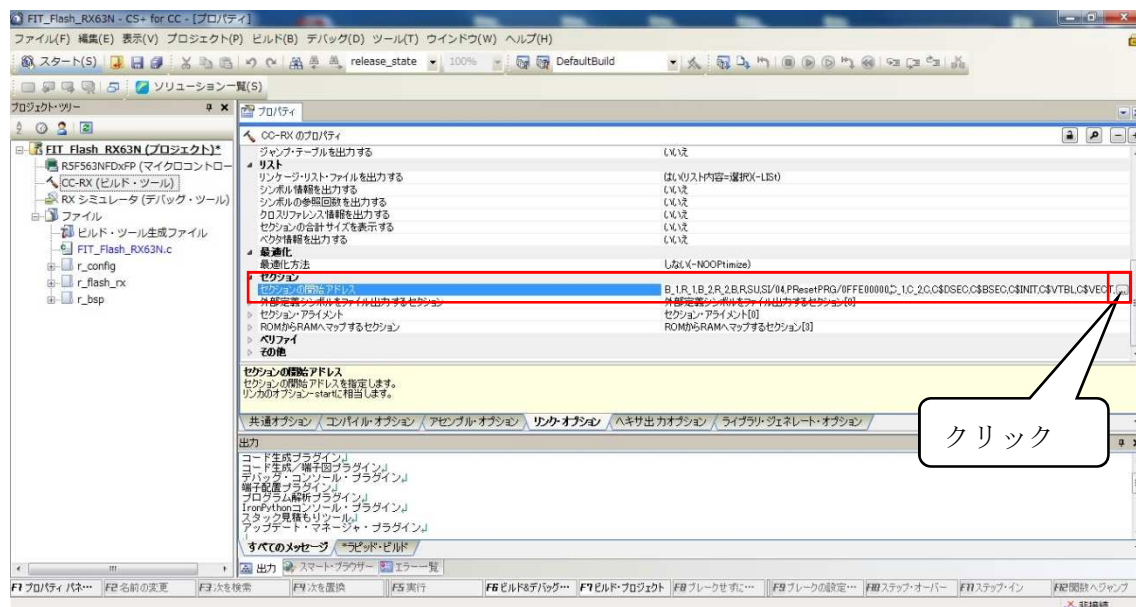
CS+のプロパティ・パネルで「リンク・オプション」タブを選択し、「出力」の「可変ベクタの空き領域のアドレス」を選択して「文字列入力」ダイアログを表示させます。

「文字列入力」ダイアログの「文字列(S):」に“`_undefined_interrupt_source_isr`”を入力し、OK をクリックします。



7.5 セクションアドレスの設定

CS+のプロパティ・パネルの「リンク・オプション」タブで、「セクション」の「セクションの開始アドレス」を選択して「セクション設定」ダイアログを表示させます。



セクションの一覧から FIXEDVECT のアドレス欄を選択した状態で、「変更」をクリックします。



FIXEDVECT のセクションアドレスを、“0xFFFF FF80” にします。



同様に、C_1 のセクションアドレスを、“0xFFE0 0000” にします。



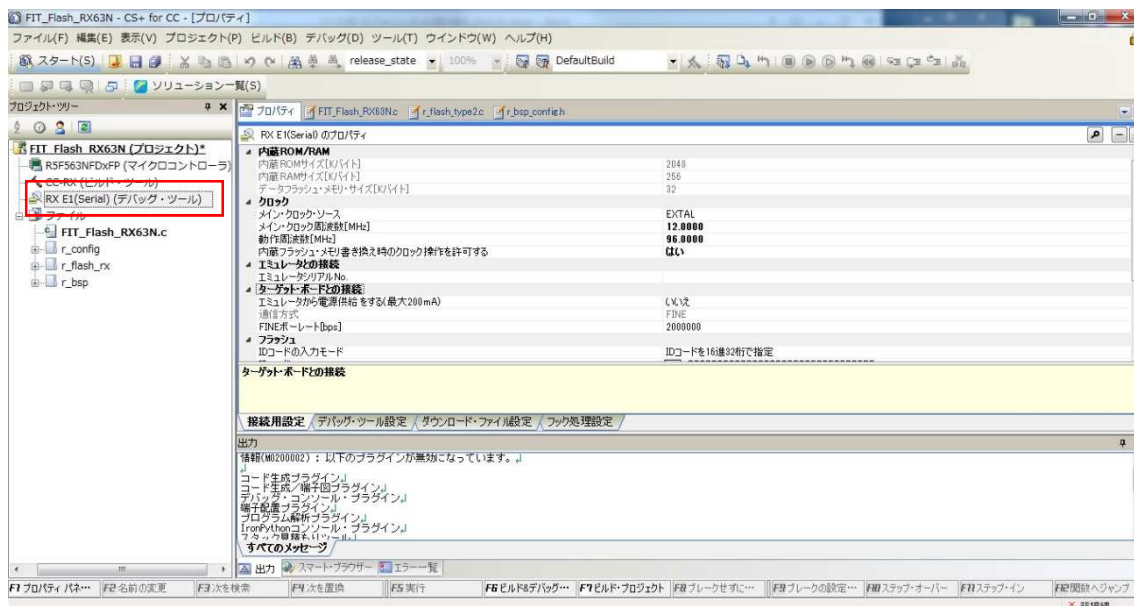
セクションの一覧から"PRresetPRG"、"C\$INIT"、"C\$VTBL"、"PIntPRG"を削除します。



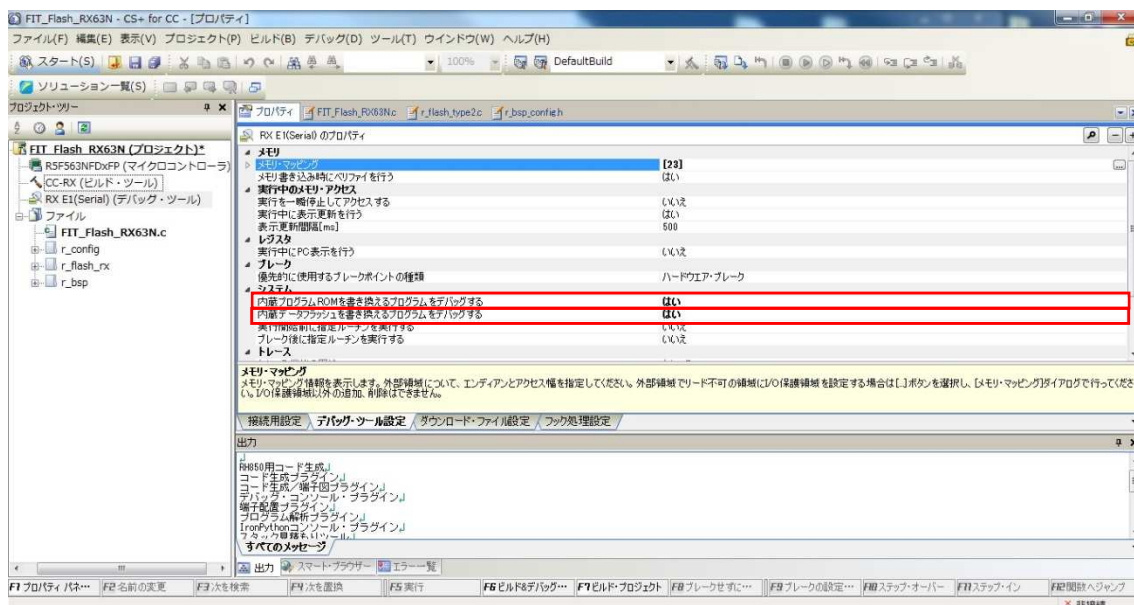
7.6 デバッグ・ツールの設定

本サンプルコードでは、内蔵フラッシュメモリの書き換えを実行し、メモリウィンドウにて対象領域のデータ変化を確認します。確認に必要な設定を以下に示します。

CS+のプロジェクト・ツリー・パネルで「RX E1(デバッグ・ツール)」をダブルクリックしてプロパティを開きます。

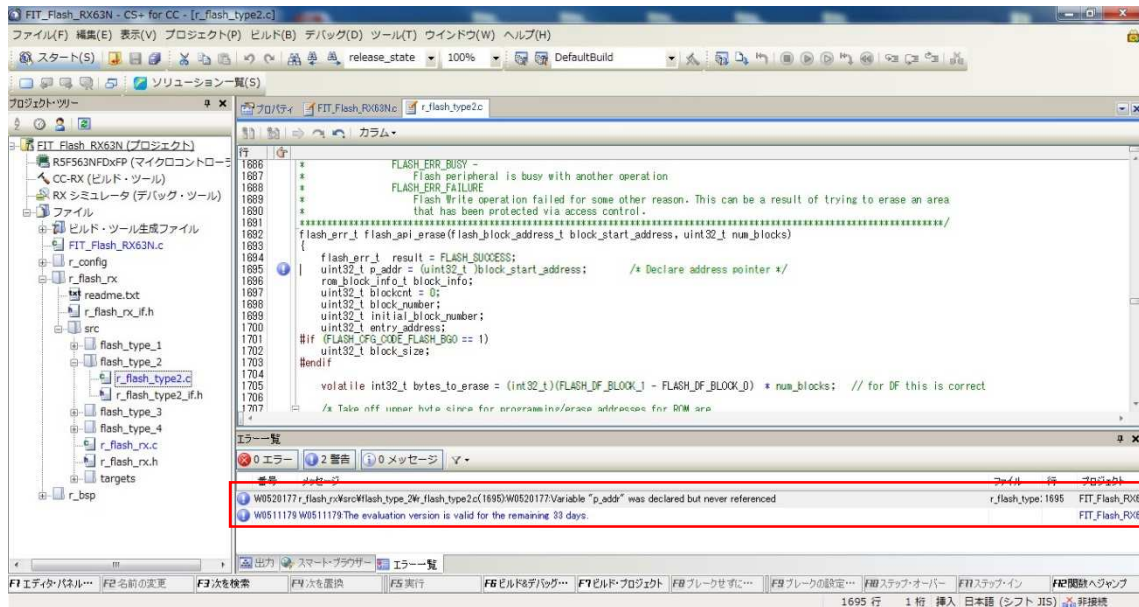


「デバッグ・ツール設定」タブで「システム」の「内蔵プログラム ROM を書き換えるプログラムをデバッグする」、および「内蔵データフラッシュを書き換えるプログラムをデバッグする」の設定を“はい”にします。



本サンプルコードでは、ビルド時に警告が発生します。

Flash API FIT モジュールで組み込んだファイルで、宣言された変数を使用していない旨の警告です。本サンプルコードでは、使用しないため、そのままにしています。

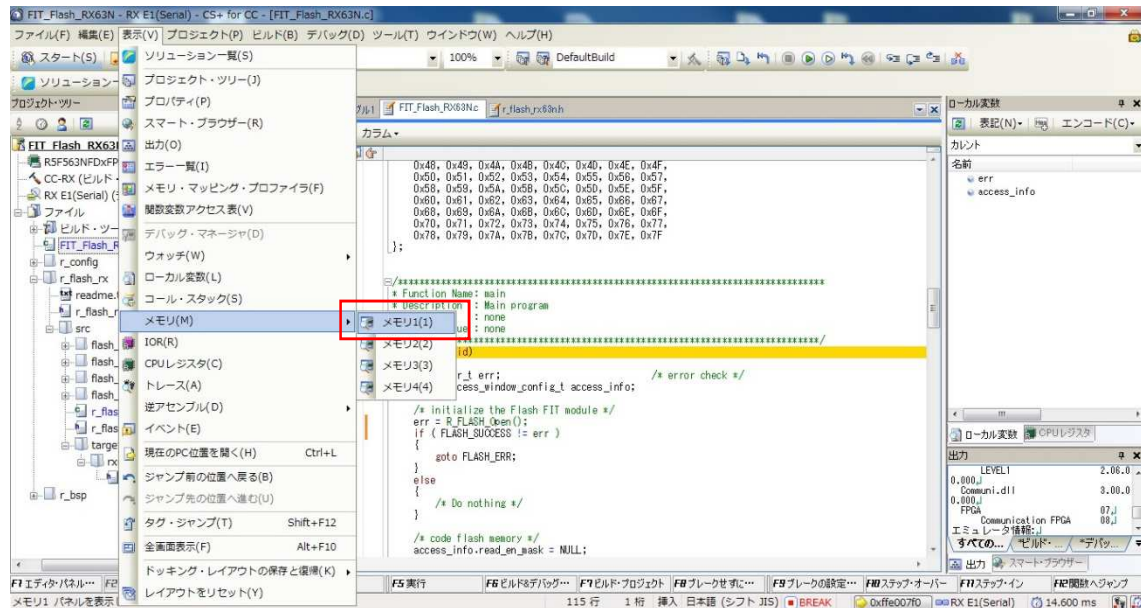


8. 動作確認方法

8.1 メモリ

フラッシュに書き込みができているかを確認する方法として、CS+に搭載されている機能のメモリを使用します。メモリから対象のフラッシュのデータを確認することができます。使用方法を以下に示します。

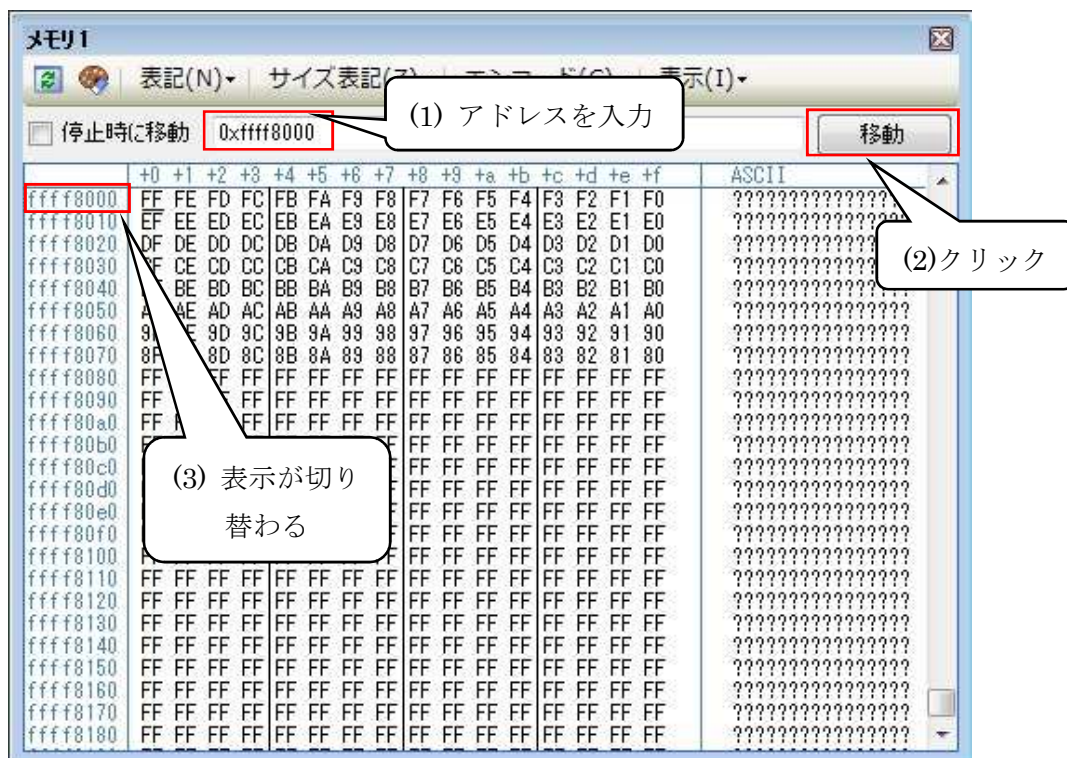
表示タブからメモリ(M)、そしてメモリ 1(1)を選択します。



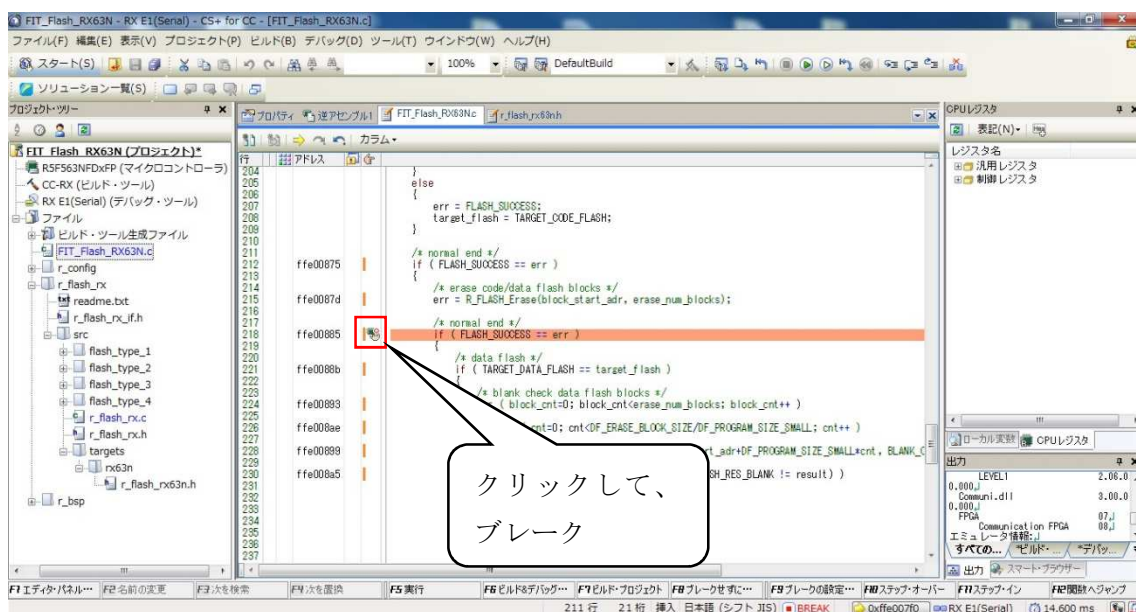
メモリ 1 パネルが表示されます。



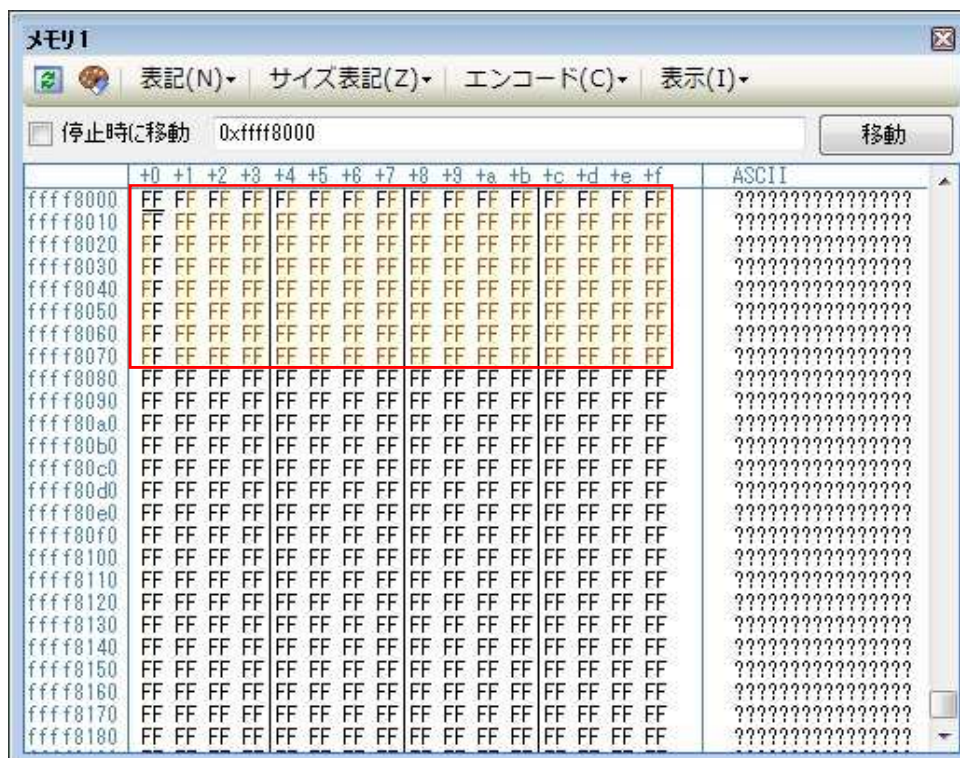
メモリ 1 パネル表示されたら、確認する ROM のアドレスである “0xffff8000”を入力し、移動をクリックします。すると、表示領域が “0xffff8000”に切り替わります。



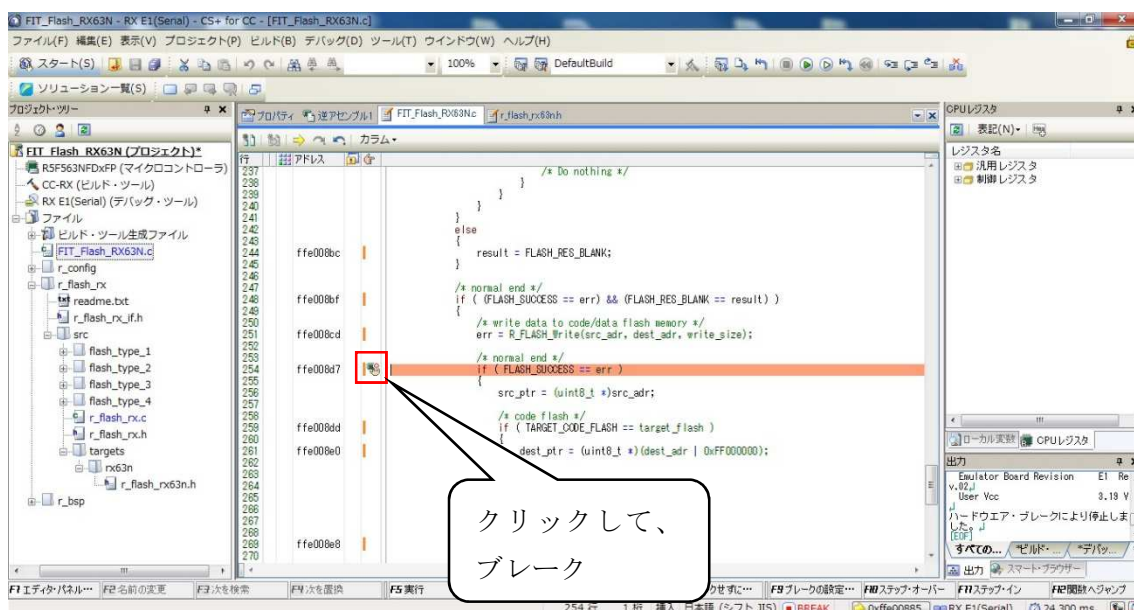
ROM の消去を確認します。R_FLASH_Erase 関数の実行後にブレークをかけて、実行ボタンをクリックします。 実行ボタンをクリック後、ブレークにより停止します。



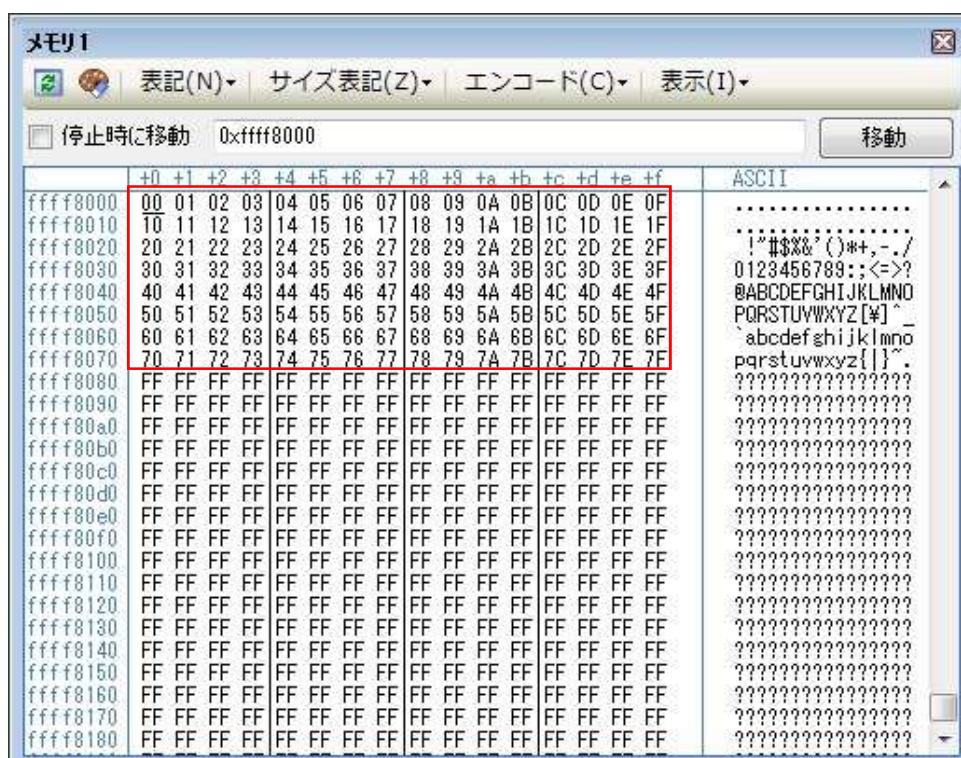
ROM の消去後の値は FFh となります。メモリ 1 パネルで、値を見ると FFh となっており、ROM の消去が確認できます。



ROM への書き込みを確認します。R_FLASH_Write 関数の実行後にブレークをかけて、実行ボタンをクリックします。 実行ボタンをクリック後、ブレークにより停止します。



メモリ 1 パネルで、値を見ると cf_write_data で設定したデータが表示されており、ROM への書き込みができたことが確認できます。

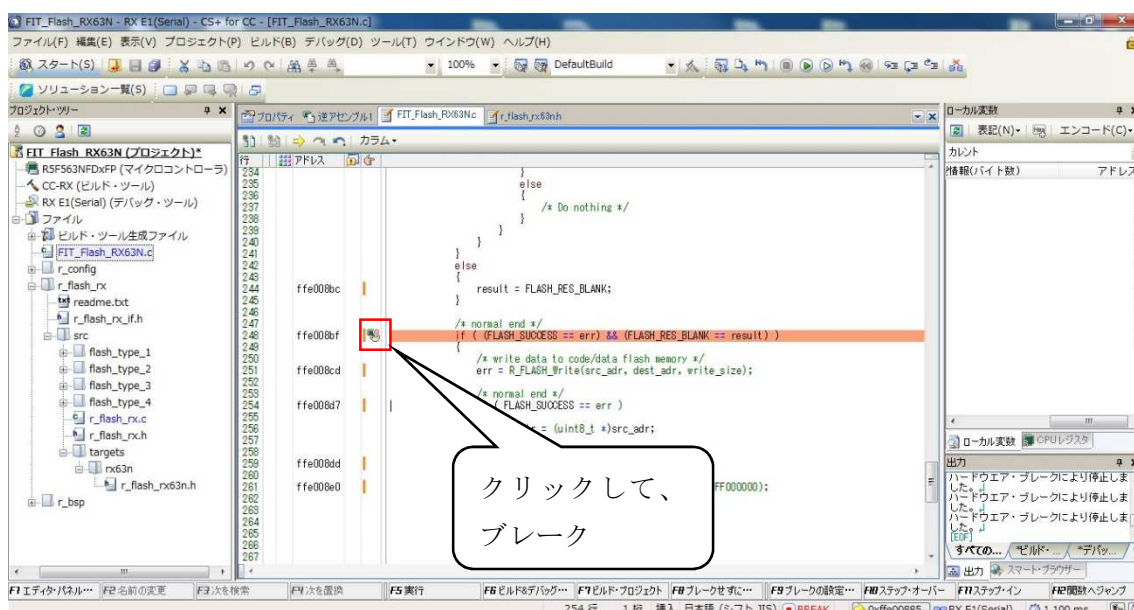


次に E2 データフラッシュへの書き込みを確認します。

メモリ 1 パネルに、確認する E2 データフラッシュのアドレスである “0x00100000” を入力し、移動をクリックします。すると、表示領域が “0x00100000” に切り替わります。

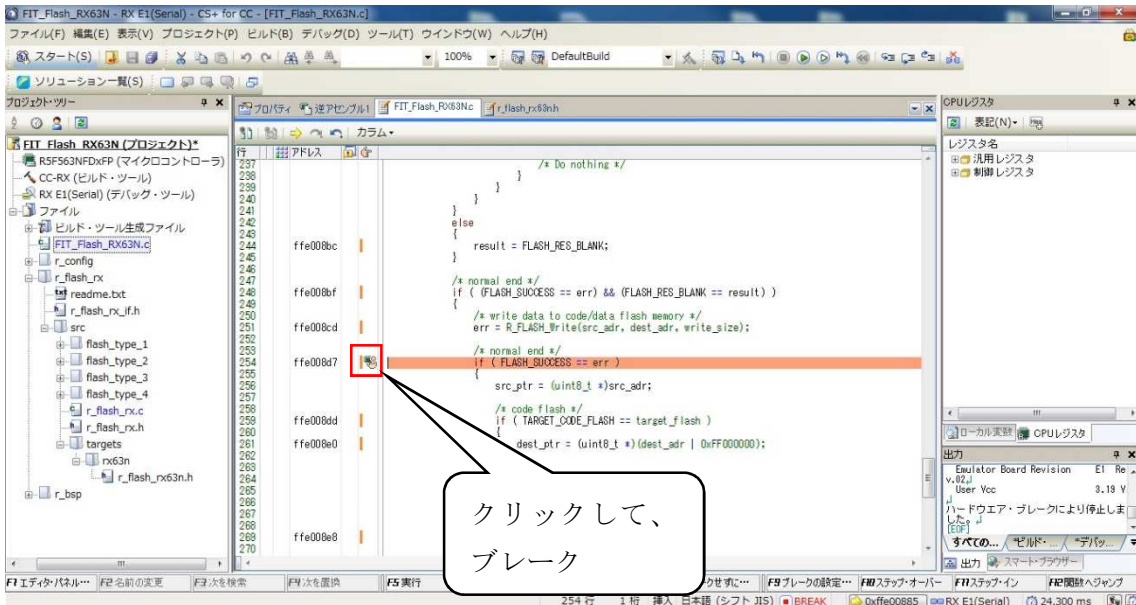


E2 データフラッシュの消去を確認します。R_FLASH_BlankCheck 関数の実行後にブレークをかけて、実行ボタンをクリックします。 実行ボタンをクリック後、ブレークにより停止します。

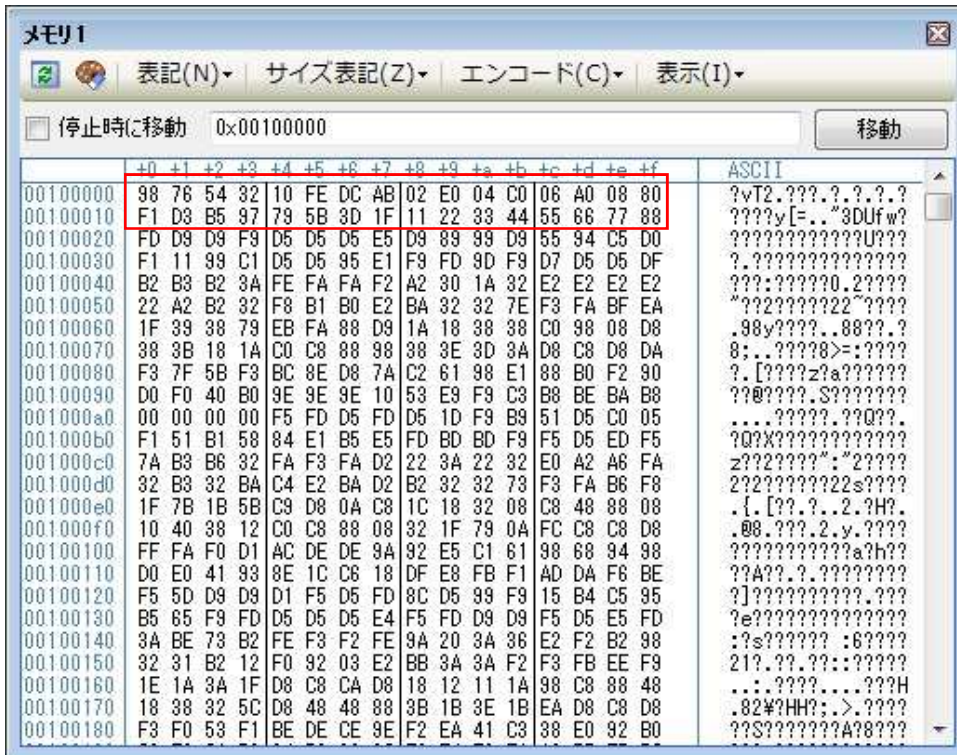


E2 データフラッシュの消去後のデータは不定値となるため、ブランクチェックで確認します。結果が“FLASH_RES_BLANK”の場合、ブランク（消去状態）でとなります。

E2 データフラッシュへの書き込みを確認します。R_FLASH_Write 関数の実行後にブレークをかけて、実行ボタンをクリックします。 実行ボタンをクリック後、ブレークにより停止します。



メモリ 1 パネルで、値を見ると df_write_data で設定したデータが表示されており、E2 データフラッシュへの書き込みができたことが確認できます。



9. 参考ドキュメント

RX63N グループ、RX631 グループ ユーザーズマニュアル ハードウェア編

Firmware Integration Technology ユーザーズマニュアル

RX ファミリ CS+に組み込む方法 Firmware Integration Technology

RX ファミリ フラッシュモジュール Firmware Integration Technology

以上