

RXファミリ

R01AN3294JJ0106

Rev.1.06

May 1, 2021

USB CDC経由内蔵FlashROM書き換えプログラム

要旨

このアプリケーションノートでは、USB ペリフェラル・コントローラを使用した FlashROM 書き換えプログラムについて説明します。

対象デバイス

RX111, RX113, RX231, RX23W
 RX62N/RX621, RX630, RX63N/RX631, RX63T
 RX65N/RX651, RX64M, RX71M, RX66T/RX72T
 RX72M, RX72N, RX66N, RX671

目次

1. 資料概要	2
2. USB CDC経由内蔵FlashROM書き換えプログラム概要	4
3. USB CDC経由内蔵FlashROM書き換えプログラムのセットアップ	8
4. USB CDC経由内蔵FlashROM書き換えプログラムの実行	10
5. ユーザプログラムの作成時の注意事項	26
6. USB CDC経由内蔵FlashROM書き換えプログラムとユーザプログラムに対する設定	28
7. USB CDC経由内蔵FlashROM書き換えプログラムの解説	35
8. ファイル転送アプリケーション (RX USB Firmware Updater) の解説	56
9. データ通信仕様	68
10. e ² studio用プロジェクトをCS+で使用する場合	74

1. 資料概要

本書は、USB パリフェラル・コントローラを使用した内蔵 FlashROM 書き換えプログラムのアプリケーションノートです。本書は「1.2 関連ドキュメント」と併用してご利用ください。

1.1 機能

本プログラムは Universal Serial Bus Specification (以降、USB と記述)の Communication Device Class を用いてユーザプログラムの内蔵 FlashROM 書き換えを行うことができます。

1.2 関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
2. RX Family Flash Module Using Firmware Integration Technology アプリケーションノート
3. RX ファミリ ボードサポートパッケージモジュール アプリケーションノート
4. 各 MCU ユーザーズ・マニュアル ハードウェア編

ルネサス エレクトロニクスホームページより入手できます。

ルネサス エレクトロニクスホームページ

【<http://japan.renesas.com/>】

USB デバイスページ

【<http://japan.renesas.com/usb/>】

1.3 注意事項

- a. 本アプリケーションノートは、動作を保証するものではありません。本アプリケーションノートをシステムに適用される場合は、お客様における動作検証は十分に実施いただきますようお願いいたします。
- b. 本プログラムは、Little Endian 設定になっています。ユーザプログラムが Big Endian の場合は、本プログラムの Endian も Big Endian に変更して下さい。Endian 設定については、「6.2 USB CDC経由内蔵FlashROM書き換えプログラムの設定」を参照してください。
- c. 本プログラムをお客様のシステムに実装する場合は、必ず「6 USB CDC経由内蔵FlashROM書き換えプログラムとユーザプログラムに対する設定」、「7.4 注意事項」を参照してください。
- d. USB CDC経由内蔵FlashROM書き換えプログラムは、ユーザプログラム(mot/hex ファイル)の解析を行いません。PC 上で動作するファイル転送アプリケーションプログラム(GUI ツール)を新たに開発する場合は、ユーザプログラムの解析処理を行う必要があります。また、RX デバイスとの USB データ通信仕様については、「9. データ通信仕様」を参照してください。
- e. 本プログラムは、USB Command Verifier(CV)をサポートしていません。
- f. "r_config"フォルダ下にある r_usb_fwupdater_config.h ファイル以外のファイルを変更した場合の動作確認は行っていません。
- g. 本プログラムは、各 FIT モジュールを使用しています。本プログラムでは Web 公開中の FIT モジュールのソースコードに対し変更を行っています。
- h. Dual モードを使用する場合、「FW_CODE」セクションを 0xFFFFFFFF7C 番地に配置してください。
- i. RSSK(RX23W)をご使用の場合、RSSK に対し以下の抵抗の移動が必要です。

R89	-->	R90
R96	-->	R97

R112 → R113

- j. 本書に記載された「USB0 モジュール」および「USB1 モジュール」という用語は、MCU ごとに示すモジュールが異なりますので、以下を参照してください。

用語	MCU	USB モジュール名
USB0 モジュール (開始アドレス:0xA0000)	RX62N/RX621	USB モジュール
	RX63N/RX631	USBa モジュール
	RX630	USBa モジュール
	RX63T	USBa モジュール
	RX64M	USBb モジュール
	RX71M	USBb モジュール
	RX65N/RX651	USBb モジュール
	RX66T/RX72T	USBb モジュール
	RX72M	USBb モジュール
	RX72N	USBb モジュール
	RX66N	USBb モジュール
	RX671	USBb モジュール
	RX111	USBc モジュール
	RX113	USBc モジュール
	RX231	USBd モジュール
RX23W	USBc モジュール	
USB1 モジュール (開始アドレス:0xA0200 / 0xD0400)	RX62N/RX621	USB モジュール
	RX63N/RX631	USBa モジュール
	RX64M	USBA モジュール
	RX71M	USBAa モジュール
	RX671	USBb モジュール

1.4 用語一覧

本書で使用される用語と略語は以下のとおりです。

API	: Application Program Interface
BSP	: Renesas Board support package module
CDC	: Communication Device Class
e ² studio	: Eclipse embedded studio (RX対応)
H/W	: Renesas USB device
MCU	: Micro control Unit
P/E	: Program / Erase
RSK	: Renesas Starter Kit
RSSK	: Renesas Solution Starter Kit
USB	: Universal Serial Bus

2. USB CDC経由内蔵FlashROM書き換えプログラム概要

本プログラムは、ホスト・マシン（以降、PCと記します）上のファイル転送アプリケーションによって指定されたユーザプログラムをUSB経由で評価ボードに転送します。転送されたユーザプログラムは、Flash Self programmingライブラリを使用してROM上の任意の場所に書き込まれます。

本プログラムの構成は、次の通りです。

1. USB CDC経由内蔵FlashROM書き換えプログラム

評価ボードに実装されるプログラムです。USBでのシリアル通信、およびセルフ書き換えを行います。

2. ファイル転送アプリケーション

ホスト・マシン（PC）で動作し、指定ファイルをUSB通信で評価ボードへ転送します。

3. ユーザプログラム

動作確認のためのファイルです。USB CDC経由内蔵FlashROM書き換えプログラムで書き込みを行います。

Program1：RSK/RSSKボード上のLEDが順番に点灯します。

Program2：RSK/RSSKボード上のLEDが同時に点滅します。

次に、プログラムのデータの流れを表します。

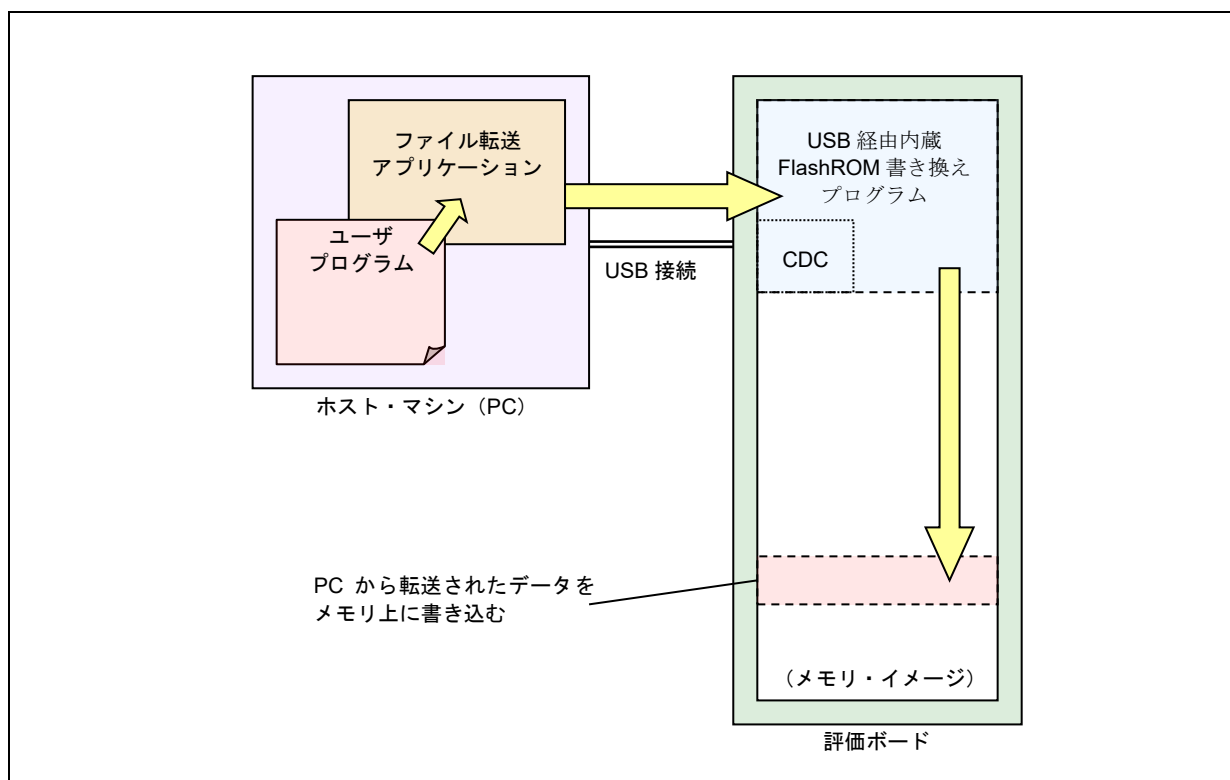


Figure 2-1 RX USB F/W Update のデータの流れ

特定条件下で評価ボードを起動することによりUSB CDC経由内蔵FlashROM書き換えプログラムが動作し、その他は、ユーザプログラムが動作します。

2.1 特長

本プログラムの特長を以下に示します。

1. 本プログラムは CDC(Communication Device Class)を使って USB ホストとのデータ通信 (Full-speed 転送)を行います。
2. 本プログラムは、内蔵フラッシュ・メモリを一部専有します。ご使用の MCU がユーザブート領域をサポートしている場合、ユーザブート領域に配置することも可能です。
3. 本プログラムがサポートしているユーザプログラムのフォーマットは、モトローラ S 形式とインテル HEX 形式です。(mot / hex ファイル)
4. 本プログラムは FlashROM に対する Write & Verify をサポートしています。
5. デュアルモードをサポートしています。(デュアルモードについては、デュアルモードをサポートする MCU のハードウェアマニュアルを参照してください。)
6. バックアップ機能をサポートしています。バックアップ機能の詳細については、「7.2 バックアップ機能」を参照してください。
7. ユーザプログラムは、すべての割り込みを使用できます。

2.2 ROM サイズ

本プログラムが使用する ROM サイズを以下に示します。

ROMサイズ : 8Kバイト

Note:

上記のサイズは、コンパイラ CC-RX V.3.01/V.3.03を使用し、最適化オプションにDefaultオプションを指定した測定結果です。

2.3 対象デバイスと FLASH TYPE

RX の FLASH Type は 4 タイプ存在します。ご使用の MCU が、どの Flash Type かは下記の表を参照下さい。詳しくは RX Family Flash Module Using Firmware Integration Technology アプリケーションノートをご参照下さい。

Table 2-1 MCU の Flash 書き込みタイプ

Flash 書き込みタイプ	対応デバイス
Flash Type1	RX111, RX113, RX231, RX23W
Flash Type2	RX62N/RX621, RX630, RX63N/RX631, RX63T
Flash Type3	RX64M, RX71M, RX72T/RX66T
Flash Type4	RX65N/RX651, RX72M, RX72N, RX66N, RX671

2.4 動作確認環境

本プログラムは、下記環境にて動作確認を行っております。

1. ハードウェア環境

- | | |
|------------|--|
| a. 評価ボード | RSK/RSSKボード |
| b. MCU | RX71M,RX64M,RX63N, RX651, RX62N, RX63T, RX630, RX111, RX113,RX231,RX72T
RX72M, RX72N, RX66N, RX23W, RX671 |
| c. エミュレータ | E2 Lite |
| d. USBケーブル | 評価ボードとPC間でUSB通信 |
| e. PC | Window [®] 8.1/ Window [®] 10 (32bit/64bit) 搭載PC |

Note:

RX23W では、RSSK ボードを使用しています。

2. ソフトウェア環境

- | | |
|------------------|---------------------------------------|
| a. 統合開発環境 | e ² studio |
| b. コンパイラ | RX ファミリ用 C/C++コンパイラパッケージ CC-RX V.3.01 |
| c. Flash 書き込みツール | Renesas Flash Programmer V.3.03.00 |
| d. サンプル・プログラム一式 | |

USB CDC経由内蔵FlashROM書き換えプログラム
ファイル転送アプリケーション
サンプル・ユーザプログラム

Note:

- (a). RX62N では、USB1 モジュールを使った動作確認を行っていません。
- (b). RX671 では、RX ファミリ用 C/C++コンパイラパッケージ CC-RX V.3.03 を使って動作確認を行いました。

2.5 フォルダ構成

本プログラムのフォルダ構成を示します。

```
(Top Directroy)
+-reference
|   +-cdc_inf
|   |   CDCドライバ用サンプルinfファイル(CDC_Demo.inf)
|   +-FirmwareUpdateGUI
|   |   |   ファイル転送アプリケーション(UsbfUpdater.exe / UsbfUpdater.ini)
|   |   +-source
|   |   |   ファイル転送アプリケーションソース一式
|   +-SampleProgram (動作確認用サンプルプログラム)
|   |   +- (MCU名)
|   |   |   +-src (サンプルプログラムソース一式)
|   |   |   +-mot (サンプル・ユーザプログラム)
+-workspace (USB CDC経由内蔵FlashROM書き換えプログラムサンプルプロジェクト一式)
|   +- (MCU名_FirmwareUpdater)
```

次に、各フォルダの説明を示します。

(1). reference¥cdc_inf

Windows®用CDCドライバをインストールするためのINFファイルが格納されているフォルダです。

CDC_Demo.inf : Windows®用のCDCドライバ(Windows® 32bit/64bit共通)

(2). reference¥FirmwareUpdateGUI

ファイル転送アプリケーションが格納されているフォルダです。

UsbfUpdater.exe : ファイル転送アプリケーションの実行ファイル

UsbfUpdater.ini : ファイル転送アプリケーションの設定ファイル

(3). reference¥FirmwareUpdateGUI¥source

ファイル転送アプリケーションのソース・プログラムが格納されているフォルダです。「8.ファイル転送アプリケーション (RX USB Firmware Updater) の解説」を参照してください。

(4). reference¥SampleProgram

サンプル・ユーザプログラムが格納されているフォルダです。

sample1.mot : LED が順番に点灯します

sample2.mot : LED が点滅します

(5). workspace

各MCUのUSB CDC経由内蔵FlashROM書き換えプログラムが格納されているフォルダです。「7 USB CDC経由内蔵FlashROM書き換えプログラムの解説」を参照してください。

3. USB CDC経由内蔵FlashROM書き換えプログラムのセットアップ

この章では、本プログラムのセットアップ手順を説明します。

3.1 プロジェクトのセットアップ

workspace フォルダ下からご使用の MCU と同じ名前のフォルダ名を選択し、以下の手順に沿ってプロジェクトのセットアップを行ってください。なお、以下の手順は e² studio を使ったセットアップ手順になります。

(1). e² studio を起動してください。

※ はじめてe² studio を起動する場合、Workspace Launcher ダイアログが表示されますので、プロジェクトを格納するためのフォルダを指定してください。

(2). [ファイル] → [インポート]を選択してください。インポートの選択ダイアログが表示されます。

(3). インポートの選択画面で、[既存プロジェクトをワークスペースへ]を選択してください。

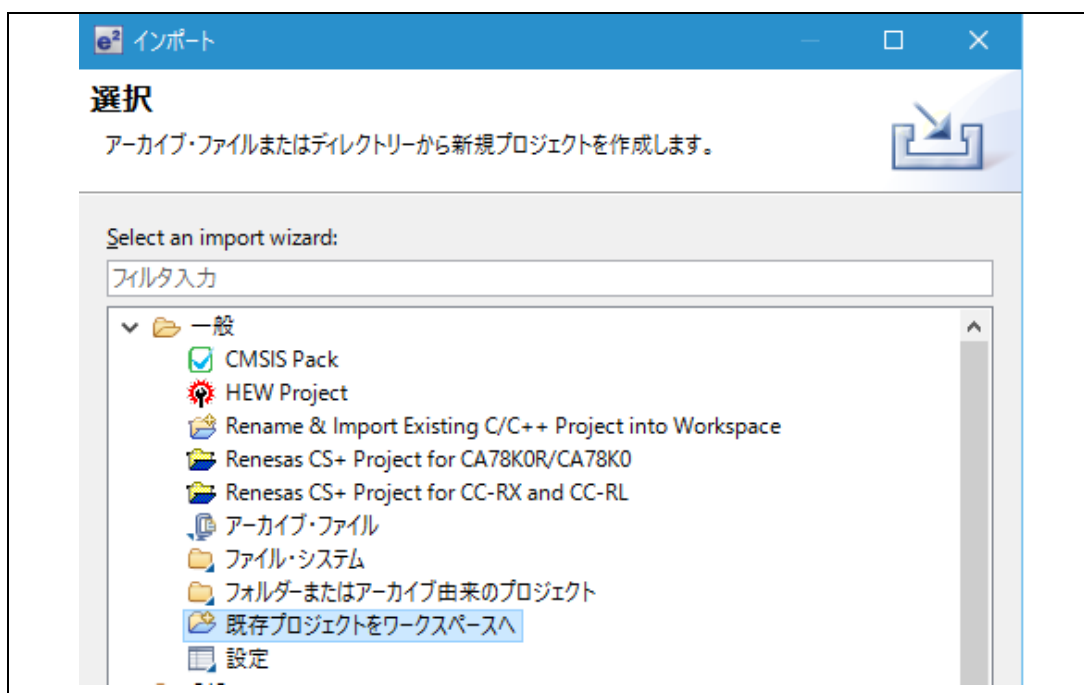


Figure 3-1 インポートの選択

(4). [ルートディレクトリの選択] の [参照] ボタンを押下して、「.cproject」(プロジェクトファイル) が格納されたフォルダを選択して下さい。

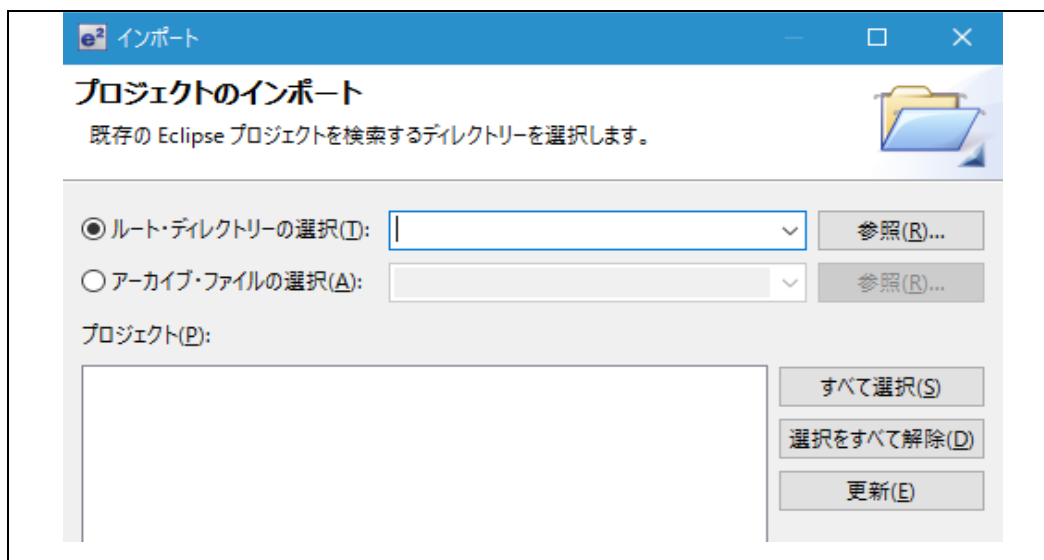


Figure 3-2 プロジェクトのインポート画面

- (5). [終了]をクリック して下さい。

プロジェクトのワークスペースへのインポートが完了します。

Note:

デュアルモードをサポートする MCU をリニアモードでご使用になる場合、**Figure 3-3**内の"デバイスの変更"(赤枠部分)よりリニアモード用のデバイスへ変更してください。たとえば、R5F565NEHxFBのデバイスをご使用の場合、R5F565NEHxFB_DUAL(デュアルモード)から R5F565NEHxFB(リニアモード)へデバイス変更してください。

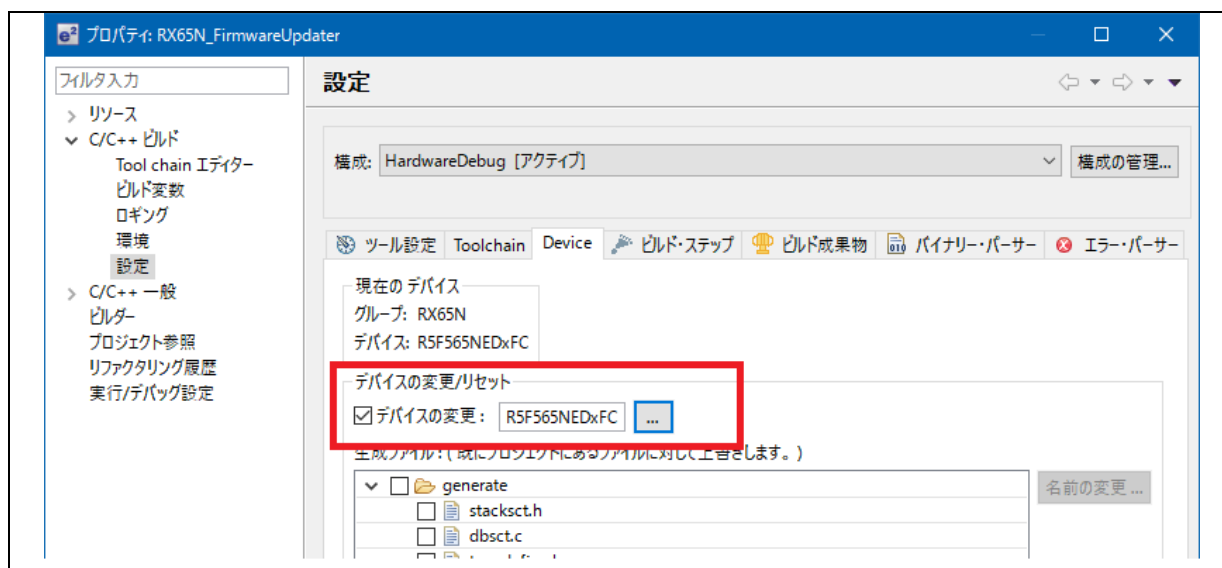


Figure 3-3 デバイスの変更画面

4. USB CDC経由内蔵FlashROM書き換えプログラムの実行

本プログラムの実行方法について説明します。

ここでは RSK/RSSKボードを用い、2つの異なるユーザプログラムが動作することを確認します。

4.1 ファイル転送アプリケーション(RX USB Function Firmware Updater)の起動

フォルダ FirmupdateGUI 内の UsbfUpdater.exe を起動すると、ユーザプログラムを送信するファイル転送アプリケーション(Windows 用 GUI ソフト)が立ち上がります。

ファイル転送アプリケーションに対する設定については、Figure 4-1を参照してください。

Note:

ファイル転送アプリケーションが起動しない場合、.exe ファイルと同じフォルダに UsbfUpdater.ini ファイルがあるかどうかを確認して下さい。

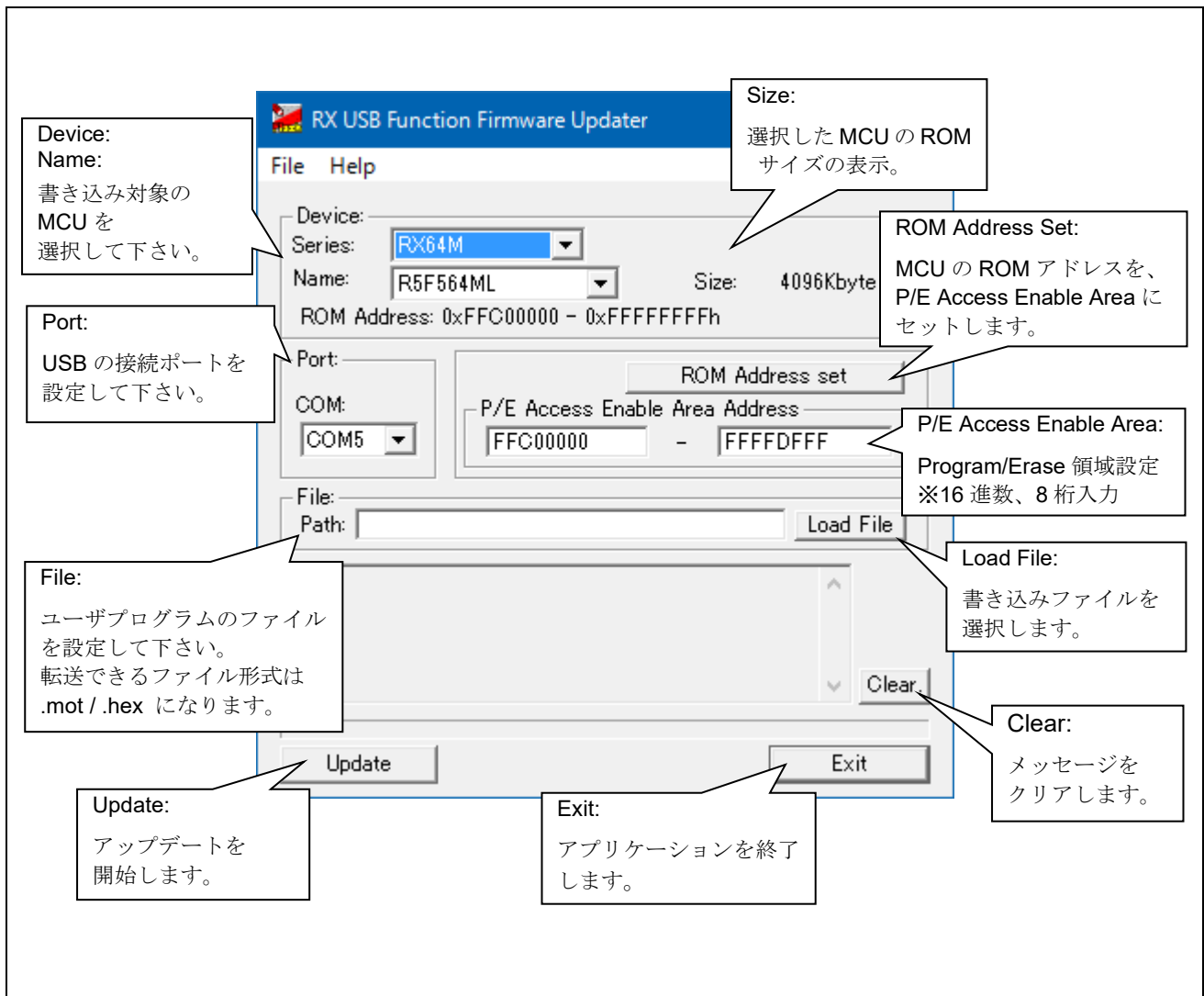


Figure 4-1 RX USB Firmware Updater GUI ソフト

4.1.1 P/E Access Enable Area Address

ユーザプログラム書き込み時において、本プログラム領域が上書きされないように Program/Erase 可能領域を設定してください。ファイル転送アプリケーションは、ここで設定された領域に対し Erase → Program を行います。

なお、本プログラムでは、リセットベクタを含む ROM ブロック (RX シリーズでは Block 0 に該当) に対するアクセスを禁止しています。そのため、P/E Access Enable Area Address に対しては、Table 4-1 で示す範囲で設定ください。

Table 4-1 P/E Access Enable Area Address 設定

バックアップ機能	P/E address Setting		
OFF	プログラム ROM 領域先頭アドレス	-	0xFFFFDFFF
ON	プログラム実行領域先頭アドレス	-	0xFFFFDFFF

Note:

1. Erase 時、指定されたアドレスを含むブロックが消去されます。ROM のブロックサイズにご注意下さい。ブロックサイズについては、各 MCU 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
2. アドレスを含むブロックが消去されます。ROM のブロックサイズにご注意下さい。ブロックサイズについては、各 MCU 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。
3. デュアルモード選択時は、起動バンクの領域を指定してください(更新対象領域を指定しないでください)。
4. P/E Access Enable Address には、ユーザプログラムの先頭アドレス(先頭 Flash ROM ブロックの先頭アドレス)と終了アドレス(終了 Flash ROM ブロックの終了アドレス)を指定してください。
5. バックアップ機能およびプログラム実行領域については、「7.2 バックアップ機能」を参照してください。

4.2 USB CDC経由内蔵FlashROM書き換えプログラムのROM書き込みおよび実行

この章では、本プログラムを実行し、書き換え処理を行うときの手順を説明します。

4.2.1 USB CDC経由内蔵FlashROM書き換えプログラムのROM書き込み

(1). ハードウェアのセットアップ

USB CDC経由内蔵FlashROM書き換えプログラムをご使用のMCUに書き込む場合の接続図を以下に示します。

a. エミュレータを使用する場合

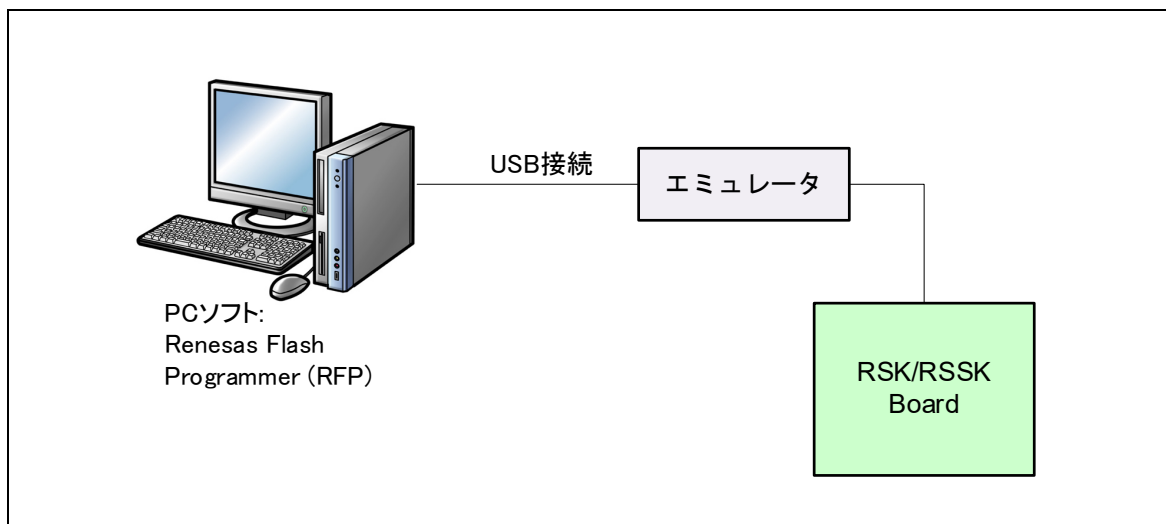


Figure 4-2 エミュレータを使用する場合の接続図

b. エミュレータを使用しない場合

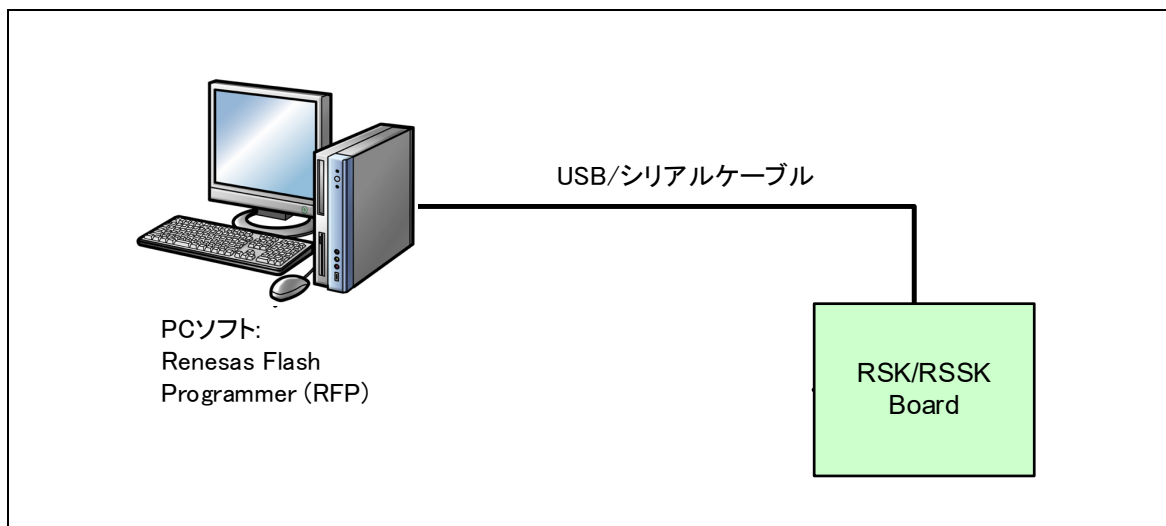


Figure 4-3 エミュレータを使用しない場合の接続図

Note:

- USBブートモードによりユーザブート領域に本プログラムを書き込んだ場合、すでにユーザブート領域にProgramされているUSBブートモードのプログラムが上書きされますのでご注意ください。
- エミュレータを使用しない場合(Figure 4-3参照)で、ユーザブート領域へ本プログラムの書き込み行うときは、ブートモードによるROM書き込みを行ってください。USBブートモードではユーザブート領域への書き込み

を行うことはできません。

- c) エミュレータを使用する場合(**Figure 4-2**参照)は、本プログラムをユーザブート領域に書き込むことができません。
- d) USBブートモードにより本プログラムを書き込む場合は、ユーザブート領域以外の領域に書き込みを行ってください。
- e) ブートモードおよびUSBブートモードについては、MCUのユーザーズマニュアル ハードウェア編を参照ください。

(2). USB CDC経由内蔵FlashROM書き換えプログラムの書き込み

Renesas Flash Programmer(RFP)を起動し、「プログラムファイル」の参照ボタンから Workspace¥(MCU 名)フォルダにあるUSB CDC経由内蔵FlashROM書き換えプログラムの書き込みファイルを選択します。スタートを押下すると、ターゲットボードにプログラムがダウンロードされます。「正常終了」と表示されれば、書き込みは完了です。

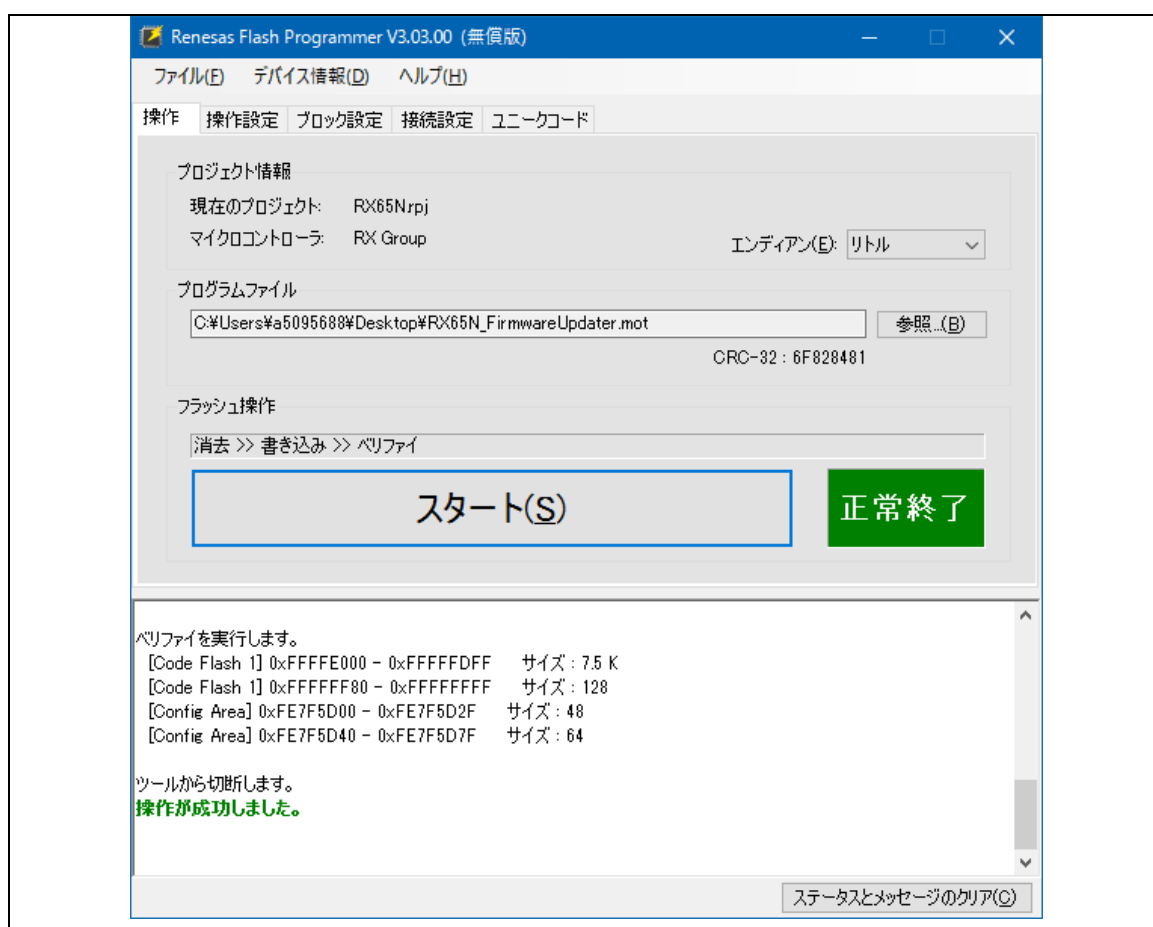


Figure 4-4 ファイルの指定

Note:

- a. Renesas Flash Programmerについての詳細は、以下のURLを参照してください

URL: (日本語)

<https://www.renesas.com/ja-jp/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html>

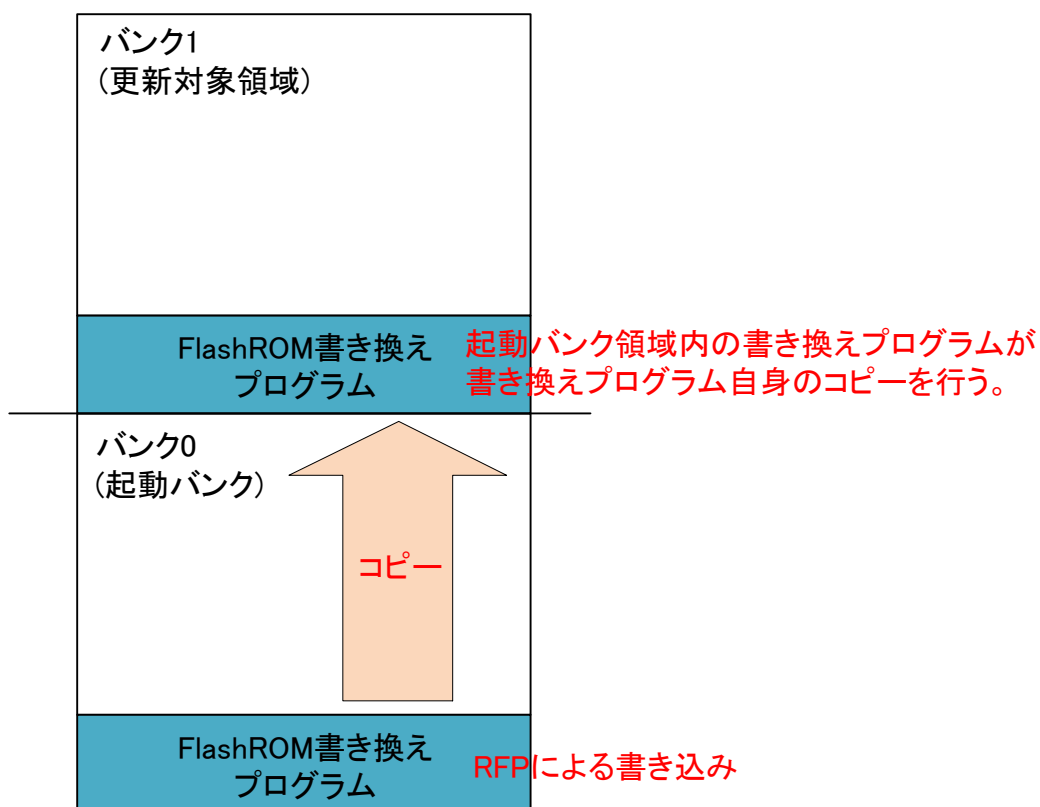
URL: (英語)

<https://www.renesas.com/en-us/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html>

- b. USB CDC経由内蔵FlashROM書き換えプログラムの配置については「4.2.2 USB CDC経由内蔵FlashROM書き換えプログラムの配置」を参照してください。

(3). 更新対象領域への FlashROM 書き換えプログラムのコピー (デュアルモード選択時)

上記(2)によりUSB CDC経由内蔵FlashROM書き換えプログラムの書き込みが完了した後、RSK/RSSK を電源投入/リセットすると、起動バンク内の**USB CDC経由内蔵FlashROM書き換えプログラム**が更新対象領域へ書き換えプログラム自身のコピーを行います。



Note:

更新対象領域への FlashROM 書き換えプログラムのコピーに失敗した場合、ユーザプログラム書き込み時に以下のメッセージがファイル転送アプリケーション(PC 側)に表示されます。

ERR: Copying of Flash ROM rewrite program failed.

4.2.2 USB CDC経由内蔵FlashROM書き換えプログラムの配置

本プログラムの配置アドレスについて説明します。

(1). ユーザブート領域以外の ROM 領域に配置する場合

USB CDC経由内蔵FlashROM書き換えプログラムは、以下の領域に配置してください。

USB CDC経由内蔵FlashROM書き換えプログラム 配置領域		
0xFFFFE000	-	0xFFFFFFFF

下記はRX63Nのメモリ・マップです。メモリ・マップの詳細に関しては、対象MCUのユーザーズ・マニュアル ハードウェア編 を参照してください。

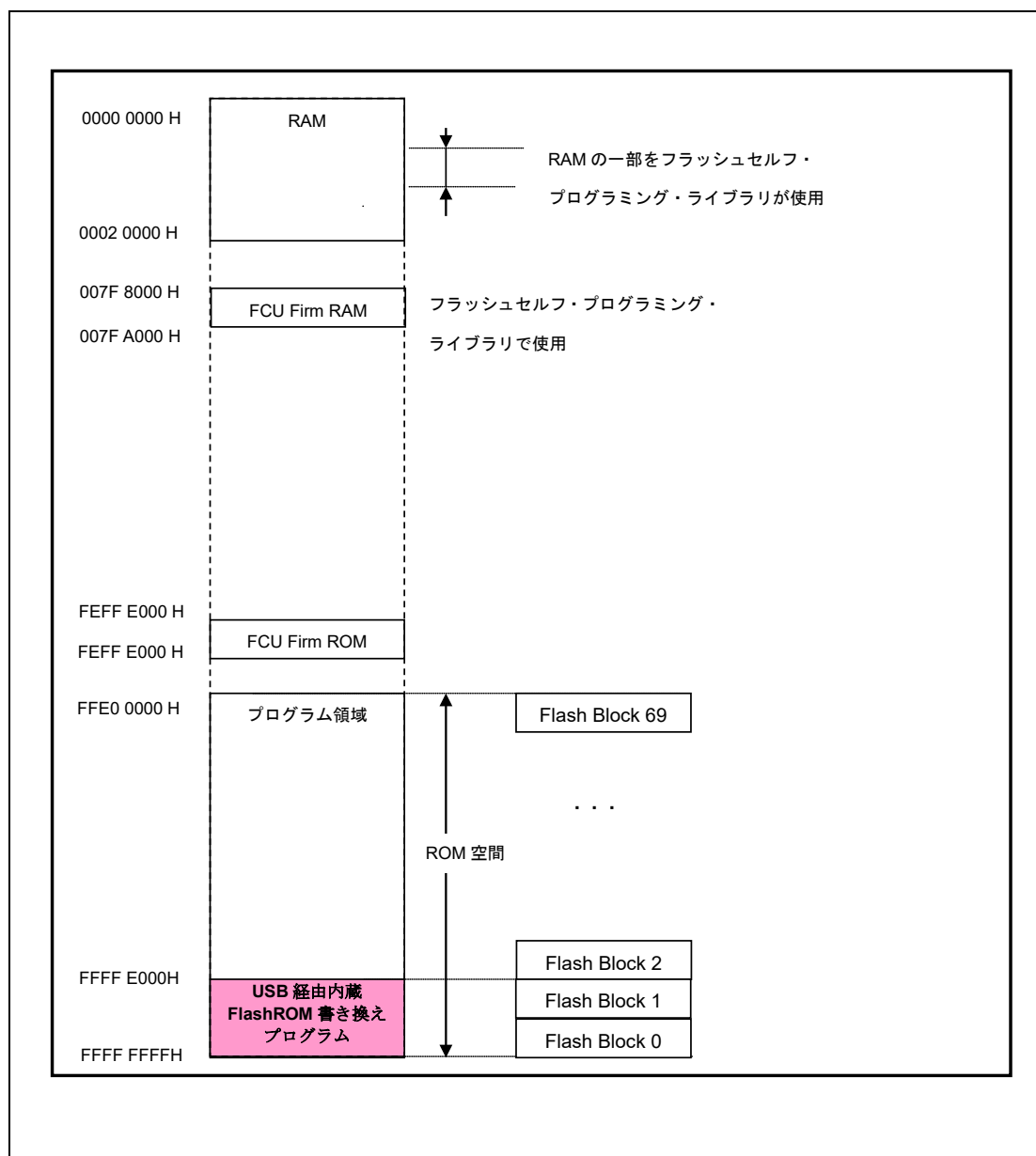


Figure 4-5 メモリ・マップ (ユーザブート領域非使用時)

Note:

USB CDC経由内蔵FlashROM書き換えプログラムをコンパイルする場合は、e² studio 上の[関数の分岐幅]に対し24bitを選択してください。なお、[関数の分岐幅]は、[ファイル]→[プロパティ]→[C/C+ビルド]→[設定]を選択後、[ツール設定]タブ→[CPU]→[拡張]で指定できます。

(2). ユーザブート領域に配置する場合

ご使用のMCUがユーザブート領域をサポートしている場合、USB CDC経由内蔵FlashROM書き換えプログラムをユーザブート領域に配置することができます。ユーザブート領域については、Table 4-2を参照してください。

Table 4-2 MCUのユーザブート領域情報

MCU	ユーザブート領域	user boot address	
RX71M	32KB	0xFF7F8000	- 0xFF7FFFFFFF
RX64M	32KB	0xFF7F8000	- 0xFF7FFFFFFF
RX66T/RX72T	32KB	0xFF7F8000	- 0xFF7FFFFFFF
RX63T	16KB	0xFF7FC000	- 0xFF7FFFFFFF
RX63N/RX631	16KB	0xFF7FC000	- 0xFF7FFFFFFF
RX630	16KB	0xFF7FC000	- 0xFF7FFFFFFF
RX62N/RX621	16KB	0xFF7FC000	- 0xFF7FFFFFFF

Note:

USB CDC経由内蔵FlashROM書き換えプログラムをユーザブート領域に配置してコンパイルする場合は、e² studio 上の[分岐幅]の設定に対し「指定しない」を選択してください。（デフォルトは「24ビット以内」になっています）なお、[分岐幅]は、[ファイル]→[プロパティ]→[C/C+ビルド]→[設定]を選択後、[Common]→[CPU]で指定できます。

下記はUSB CDC経由内蔵FlashROM書き換えプログラムをRX63Nのユーザブート領域に配置した場合のメモリ・マップになります。

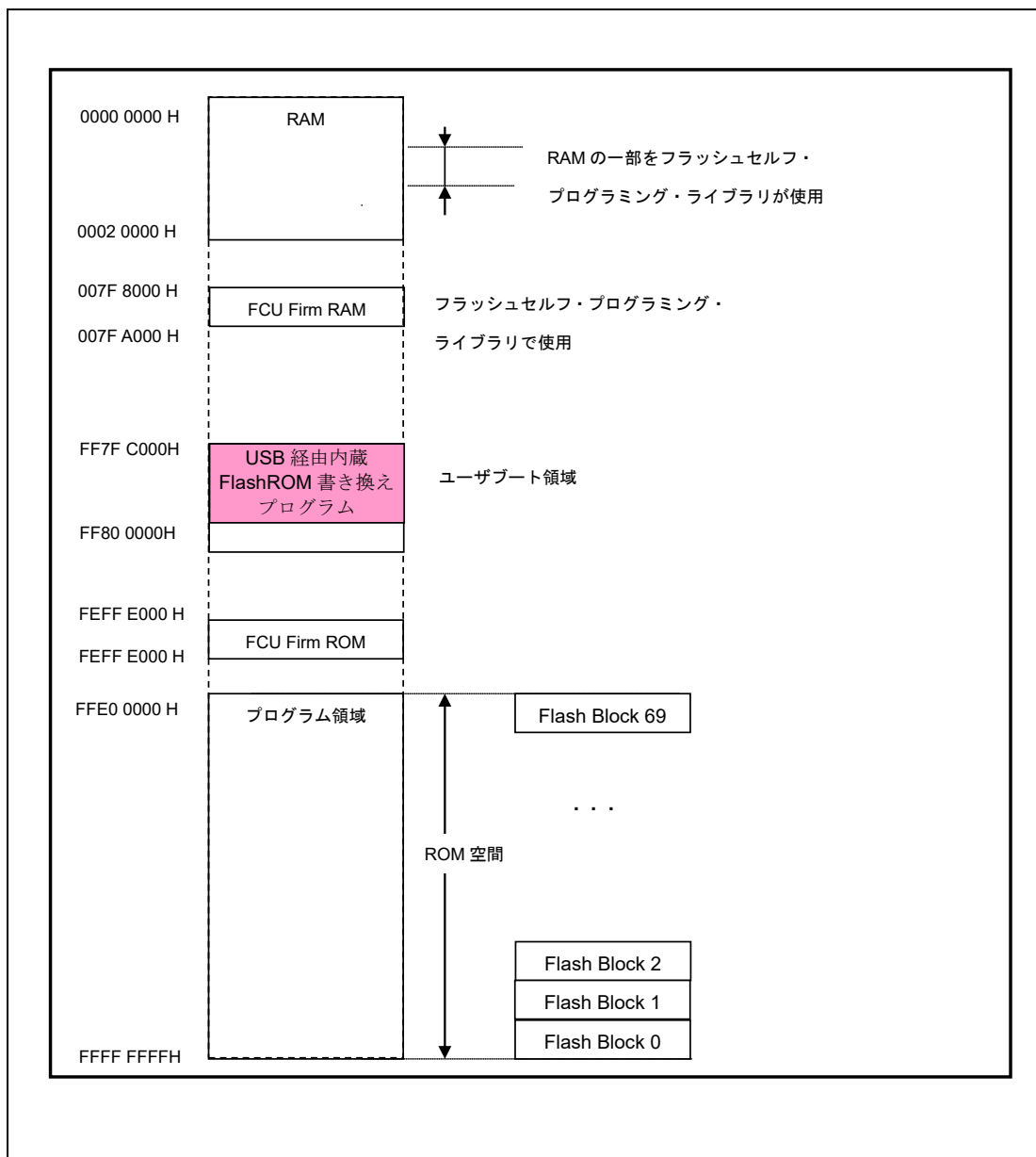


Figure 4-6 メモリ・マップ (ユーザブート領域使用時)

(3). デュアルモードを使用する場合

デュアルモード使用時のメモリ・マップは以下の通りです。

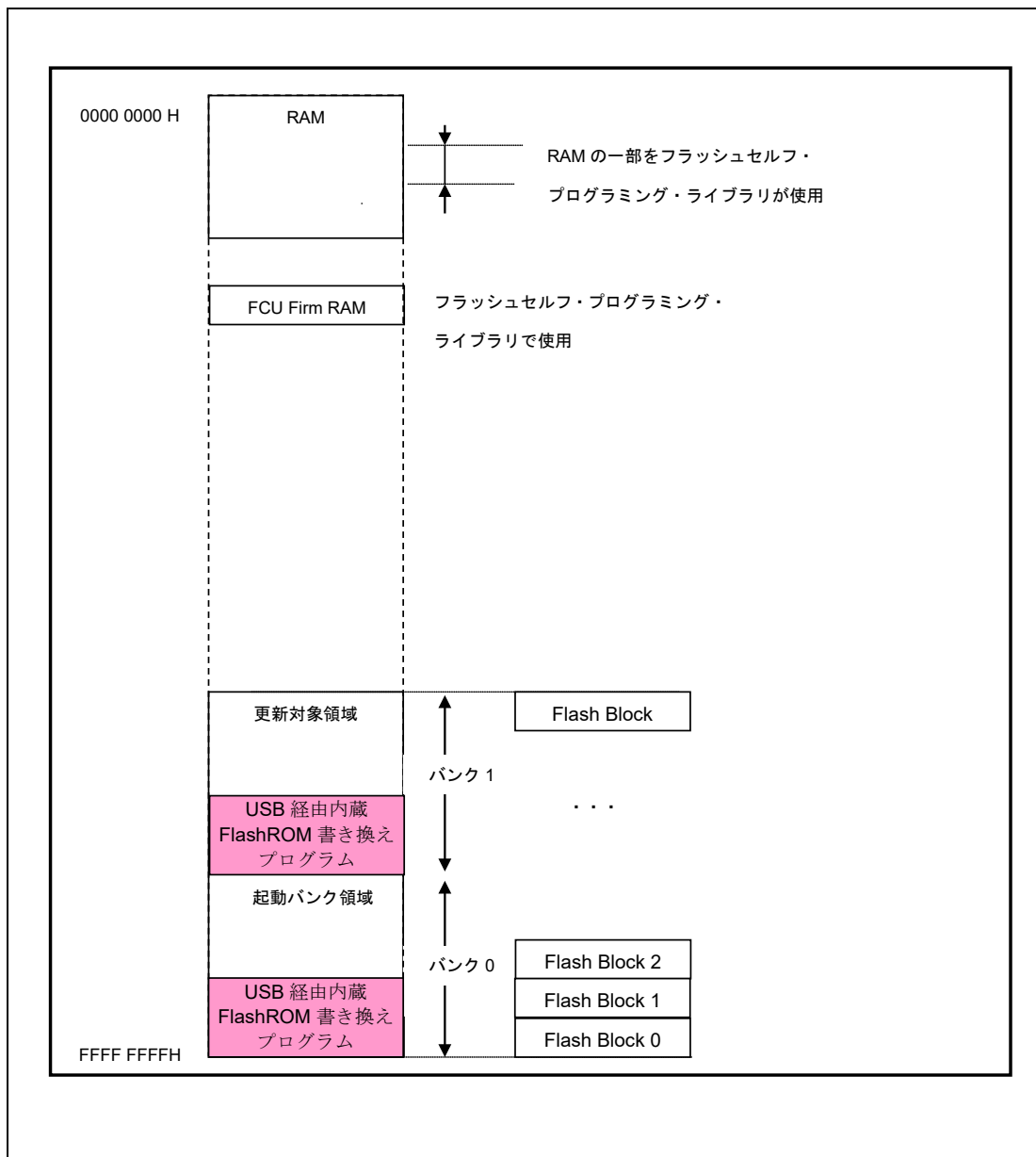


Figure 4-7 メモリ・マップ (デュアルモード使用時)

4.3 USB CDC経由内蔵FlashROM書き換えプログラムの実行(ユーザプログラム書き込み)

本章では、USB CDC経由内蔵FlashROM書き換えプログラムを実行し、ユーザプログラムの書き込み手順について説明します。

(1). ハードウェアの準備

書き換え処理を実行するために、エミュレータを外し、PC と評価ボードを USB ケーブルで接続します。

Figure 4-8に接続図を示します。

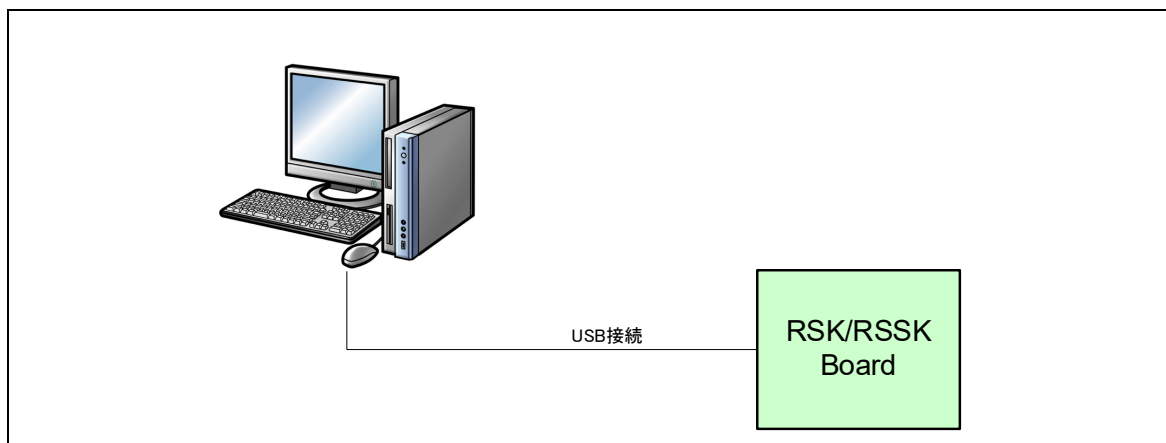


Figure 4-8 PC と評価ボードの接続図

(2). USB CDC経由内蔵FlashROM書き換えプログラムの起動

評価ボード上の SW3 を押下しながらリセット・ボタンを押してください。Program モードに遷移し、PC からの転送データを待つ状態になります。

Note:

ファイル転送アプリケーションが動作するPCには、CDCドライバをインストールする必要があります。詳細については、「4.5 CDCドライバのインストール」を参照してください。

(3). ファイル転送準備

ファイル転送アプリケーション(RX USB Function Firmware Updater :PC 側ソフト)を起動してください。(Figure 4-10参照)。

ファイル転送アプリケーション上の"COM:"には、Windows のデバイス・マネージャを確認し、割り当てられた COM 番号を選択してください。

Note:

COM番号は環境によって変わります。また、COM番号は1~9をご使用ください。

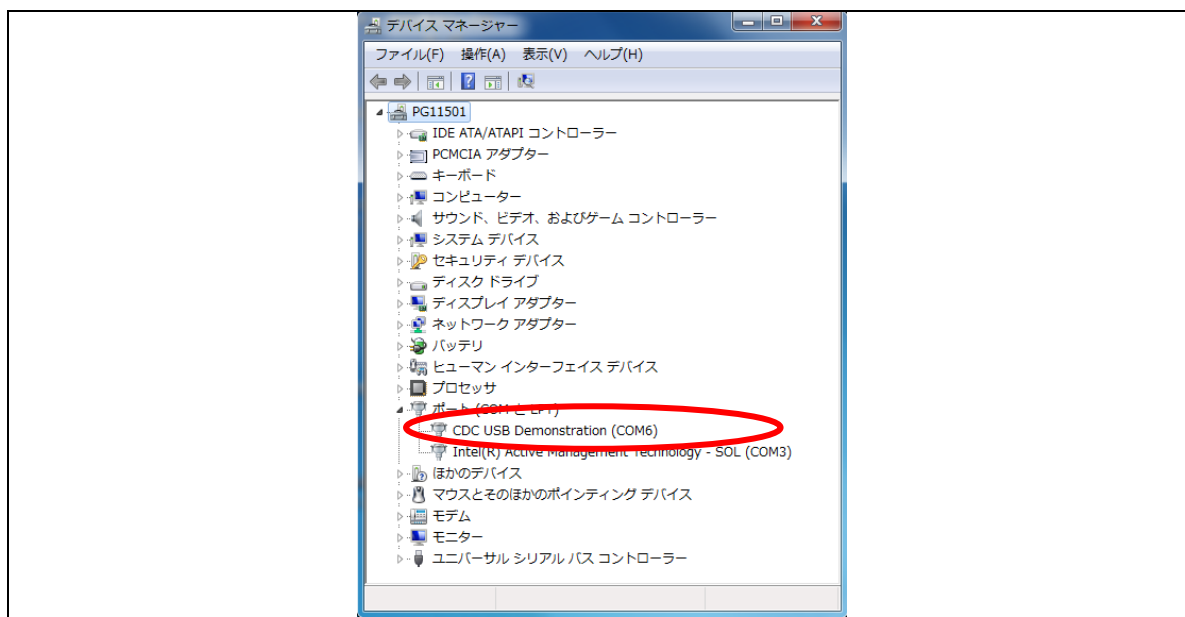


Figure 4-9 デバイス・マネージャのポート確認

(4). 転送ファイル選択

ファイル転送アプリケーション (RX USB Function Firmware Updater :PC側ソフト) 上の“Load File” ボタンをクリックして、ROM書き込み対象ファイル(ユーザプログラム)を選択します。また、“Device:”には、ご使用のMCUを選択して下さい。

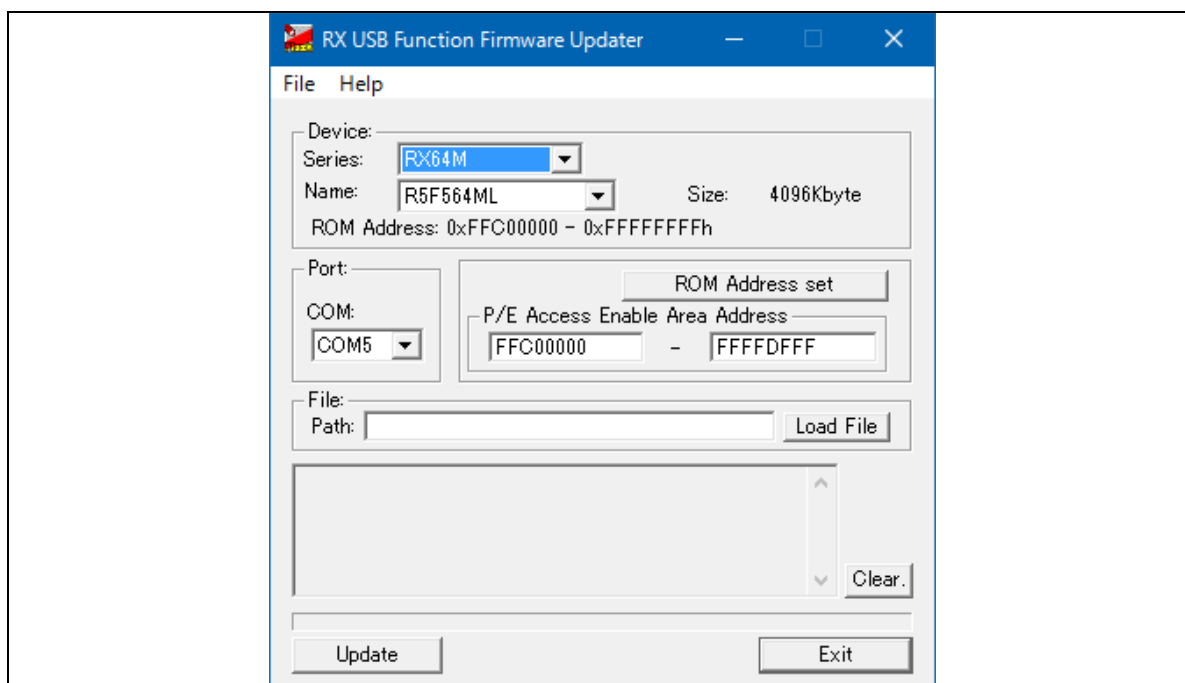


Figure 4-10 RX USB Firmware Updater GUI ソフト(ファイル転送アプリケーション)

ファイル転送アプリケーションの使用方法に関しては、「**4.1 ファイル転送アプリケーション(RX USB Function Firmware Updater)の起動**」をご参照下さい。

(5). P/E 制限領域の設定 (P/E Enable Address 設定)

ROM に対する Program/Erase 可能範囲を設定します。詳細は、「4.1.1 P/E Access Enable Area Address」を参照下さい。

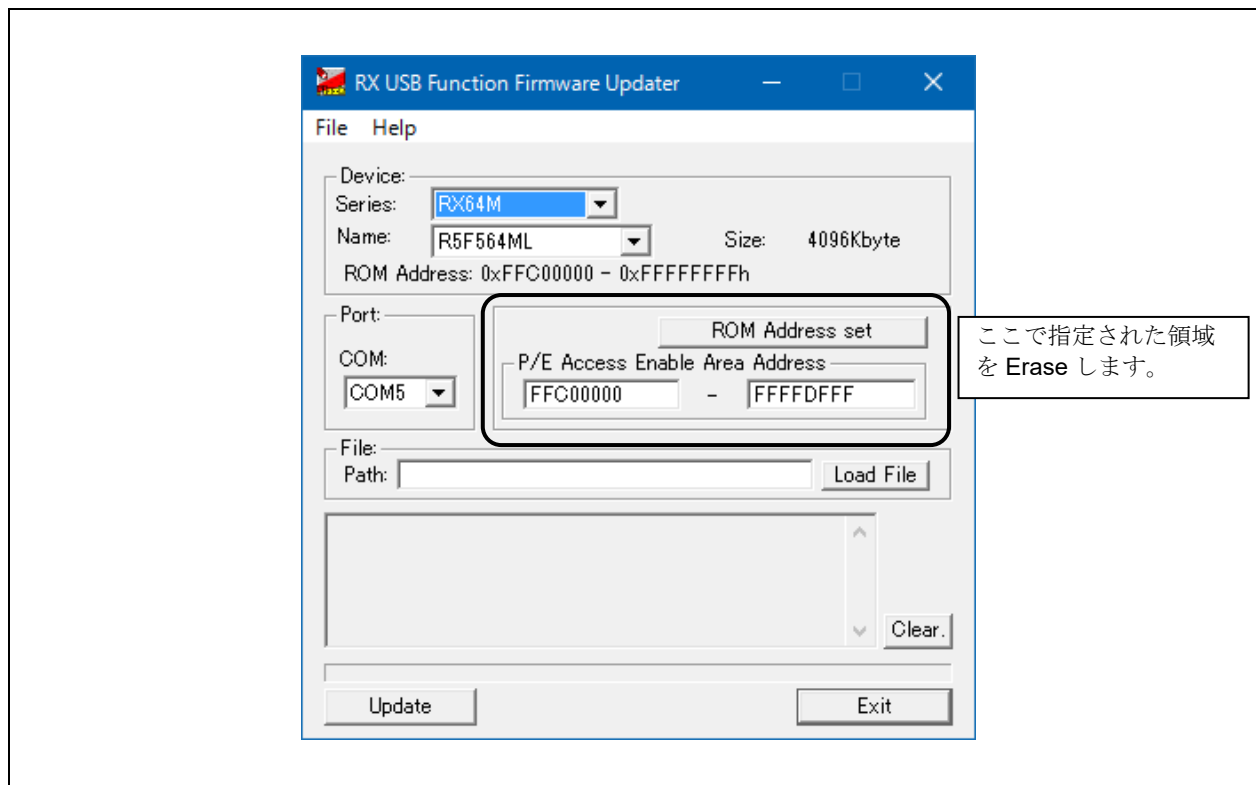


Figure 4-11 P/E 制限領域の設定

(6). ユーザプログラム転送実行

ファイル転送アプリケーション(GUI)の“Update” ボタンをクリックします。開始のメッセージが表示され、ファイルの転送処理、および書き換え処理が開始されます。

Note:

- ユーザプログラム書き込み中に USB ケーブルの Detach を行わないでください。書き込み中に USB ケーブルを Detach した場合、RX MCU をリセットする必要があります。
- ユーザプログラムの書き込みに失敗した場合は、ファイル転送アプリケーション(GUI)のメッセージ欄にメッセージが表示されます。各メッセージについては「8.4 メッセージ表示」を参照して下さい。
- デュアルモード使用時、「4.2.1 USB CDC経由内蔵FlashROM書き換えプログラムのROM書き込み」(3)に記載されている更新対象領域へのUSB CDC経由内蔵FlashROM書き換えプログラムのコピー処理に失敗した場合、ファイル転送アプリケーション(PC ツール)上にメッセージ"ERR:Flash ROM rewrite program does not exist in update target area."が表示されます。

(7). ユーザプログラム転送完了

転送処理, および書き換え処理が終了すると, ファイル転送アプリケーション(GUI)により, 終了メッセージが表示されます。これで一連の書き換え処理は終了です。なお、デュアルモード選択時、更新対象領域に対するユーザプログラム書き込みが正常終了すると、USB CDC経由内蔵FlashROM書き換えプログラムによってバンク切り替え処理が行われます。また、更新対象領域へのユーザプログラム書き込みに失敗した場合、USB CDC経由内蔵FlashROM書き換えプログラムによるバンク切り替え処理は行われません。

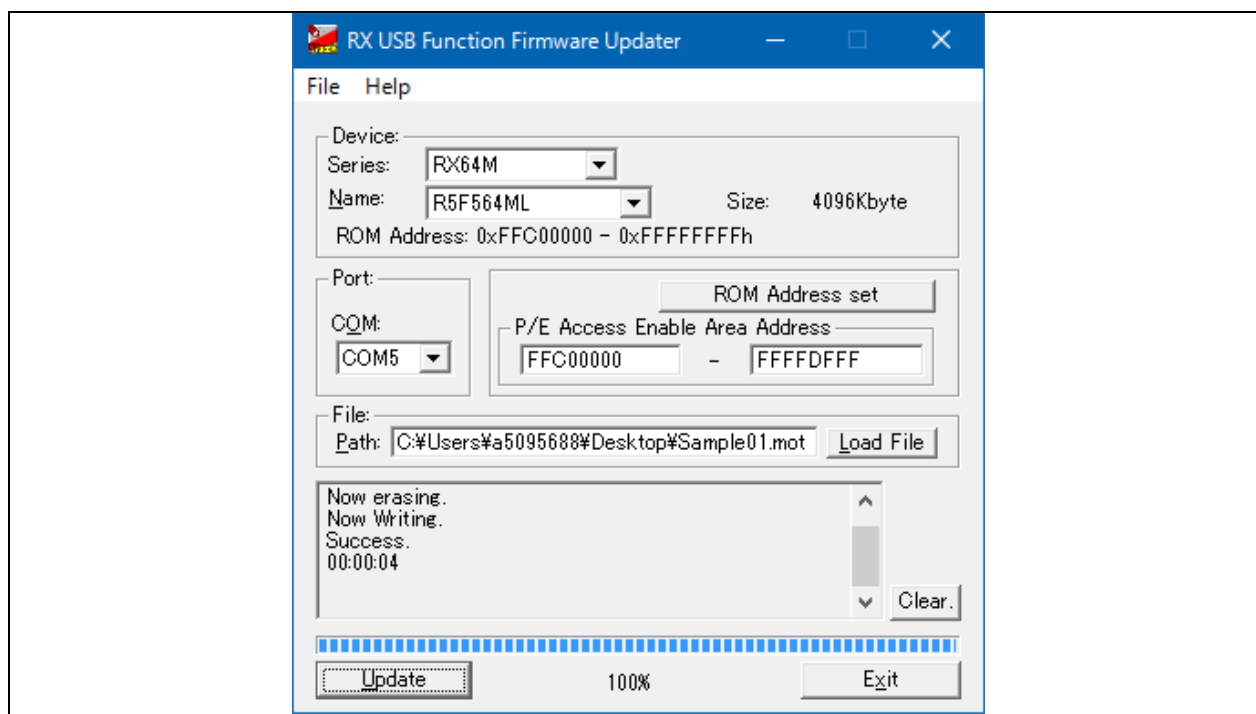


Figure 4-12 書き換え処理終了

(8). ユーザプログラムの起動

書き換えが完了すると自動的にソフトウェアリセットされ、書き込んだユーザプログラムが起動します。

サンプル・プログラム 1(ユーザプログラム)を書き込んだ場合、RSK/RSSK ボードの LED が順に点灯します。なお、デュアルモード選択時、上記(7)でのユーザプログラム書き込みが正常終了した場合、更新対象領域に書き込まれたユーザプログラムが起動します。また、ユーザプログラム書き込みに失敗した場合、起動バンク領域内にあった従来のユーザプログラムが起動します。

(9). ユーザプログラムの書き換え

ユーザプログラムを書き換えます。サンプル・プログラム 2(ユーザプログラム)を用意し、再度USB CDC経由内蔵FlashROM書き換えプログラムを起動して、(4)から同様の手順で書き換えを行って下さい。

(10). 書き換え完了

書き込みが完了すると評価ボードがリセットされ、新しく書き込んだユーザプログラムが起動します。

サンプル・プログラム 2(ユーザプログラム)を書き込んだ場合、RSK/RSSK ボードの LED が点滅します。

4.4 ユーザプログラム書き込み時の注意事項

1. USB CDC経由内蔵FlashROM書き換えプログラムが書き込まれた領域にユーザプログラムを上書きしてしまっ

た場合、USB CDC経由内蔵FlashROM書き換えプログラムの書き込みからやり直して下さい。

※ MCUによってROMのEraseブロック単位が異なりますので、ご注意下さい。

- リセットベクタを含むブロックをEraseしないようにご注意下さい。リセットベクタが消去されるとUSB CDC経由内蔵FlashROM書き換えプログラムが起動できなくなります。

4.5 CDC ドライバのインストール

ファイル転送アプリケーションが動作するPCには、CDCドライバをインストールする必要があります。USB CDC経由内蔵FlashROM書き換えプログラムの書き込みを行ったターゲットボードをPCに接続すると、Figure 4-13に示すウィザードが表示され、CDCドライバのインストールが行われます。

- デバイス・マネージャより、ドライバーソフトウェアの更新を選択します。
- 《コンピューターを参照してドライバーソフトウェアを検索します (R) 》を選択します。

Note:

- PCのOSがWindows® 10の場合、CDCドライバのインストール作業は不要です。
- PCのOSがWindows® 8.1の場合、INFファイルのほかデジタル署名済のカタログファイルが必要になります。デジタル署名済のカタログファイルはお客様により作成いただく必要があります。

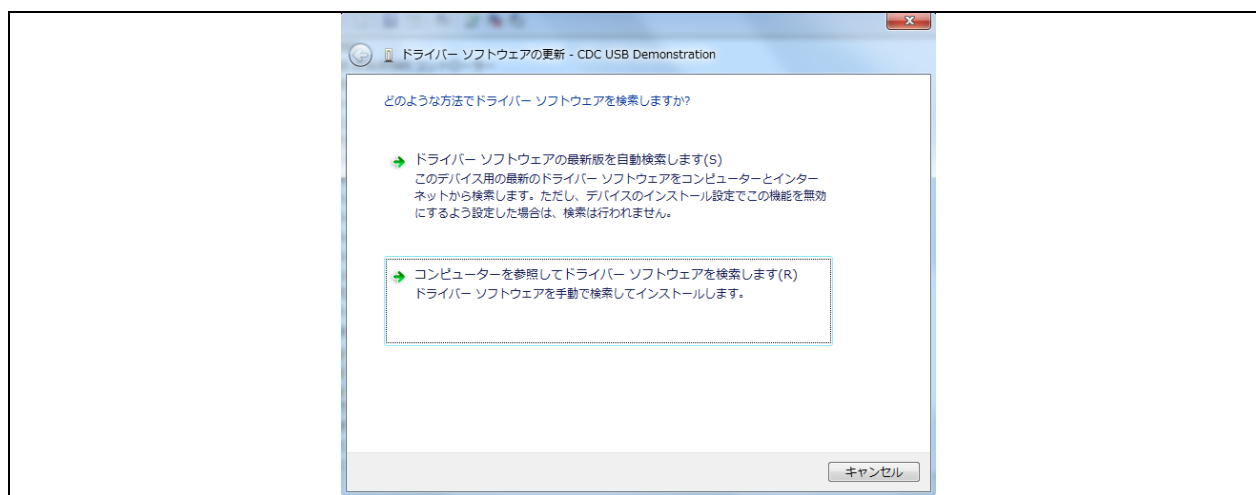


Figure 4-13 新しいハードウェアの検索ウィザード

- 《次の場所で最適のドライバーソフトウェアを検索します》を選択します。

“参照 (R) ” をクリックして“CDC_Demo.inf”の存在するフォルダを指定し、“次へ (N) ” をクリックしてください。

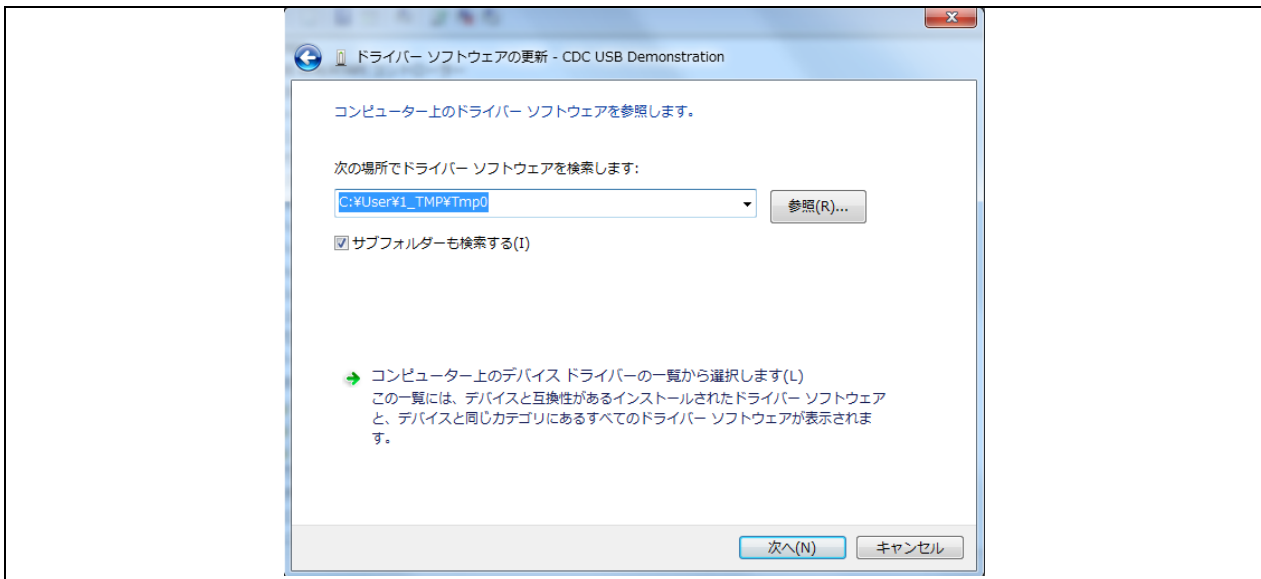


Figure 4-14 ドライバの場所の選択

Note:

CDC_Demo.inf ファイルは、パッケージ内の"reference¥cdc_inf"に格納されています。

- (4). 次のインストール確認画面が表示される場合は、“このドライバーソフトウェアをインストールします (I)”をクリックしてください。

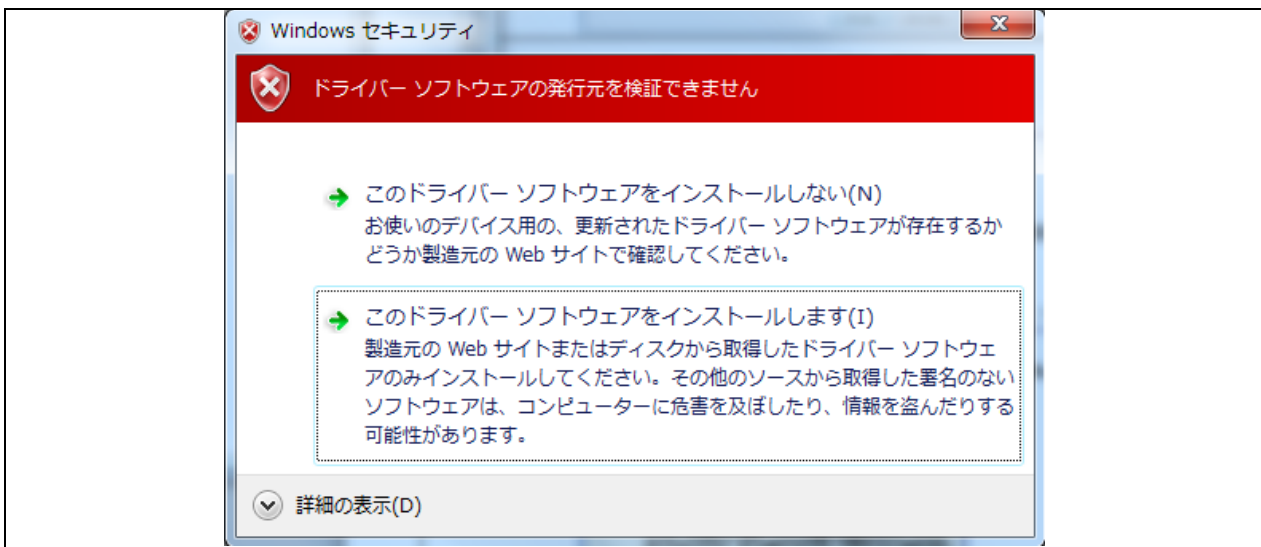


Figure 4-15 インストール確認

- (5). 次のウインドウが表示されたら、CDC ドライバのインストールは完了です。“閉じる”をクリックしてください。

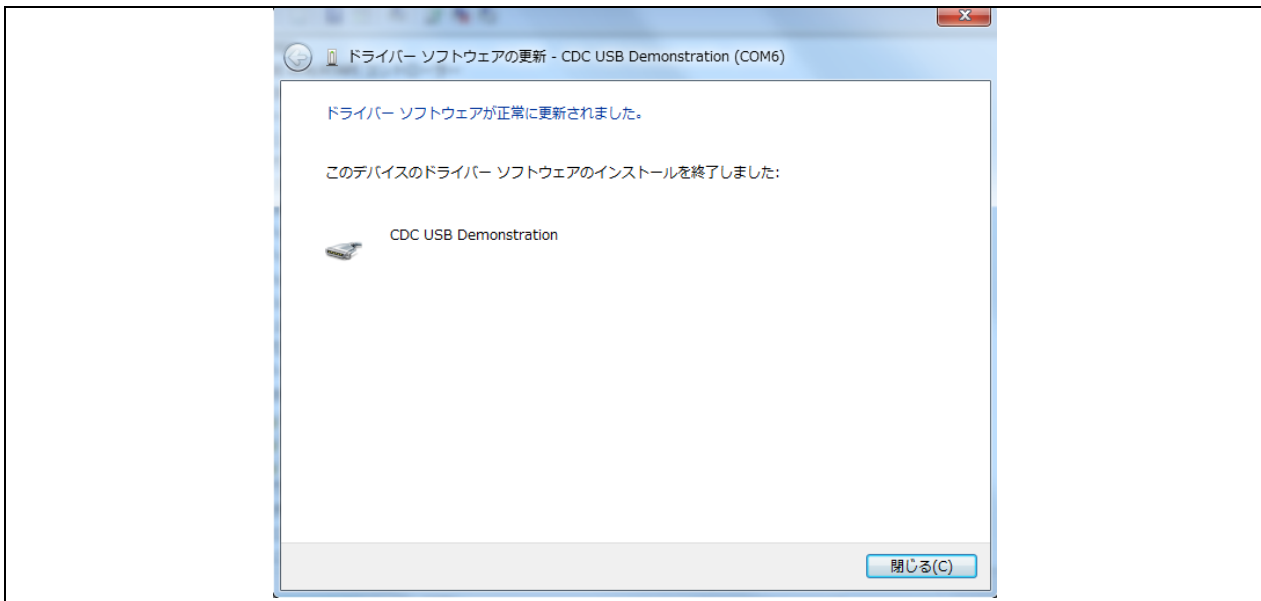


Figure 4-16 インストール完了

※ Windows 8.1 環境でドライバをインストールする場合、インストール確認が出ずにエラーになる場合があります。

5. ユーザプログラムの作成時の注意事項

この章では、ユーザプログラムを作成するうえで注意すべき事項について説明します。

5.1 ファイルフォーマット

USB CDC経由内蔵FlashROM書き換えプログラムがサポートしているファイル形式は以下の通りです。

- ・モトローラS形式
- ・インテルHEX形式

Note:

本プログラムはロードアドレスが昇順のファイルのみをサポートしています。ロードアドレスが、降順または前後するファイルについてはサポートしていません。

5.2 UserApp Header 領域 (ユーザ・アプリケーション・ヘッダ)

USB CDC経由内蔵FlashROM書き換えプログラムを使用してユーザプログラムを書き込む場合、そのユーザプログラムにはUserApp Header(ユーザ・アプリケーション・ヘッダ)領域が必要になります。UserApp Header領域のサイズは、ユーザプログラムのスタートアドレス格納領域(4バイト)とセキュリティコード格納領域(4バイト)の計8バイトです。

(Figure 5-1参照)

UserApp Header領域の作成については、「6.1 ユーザプログラムに対する設定」を参照してください。

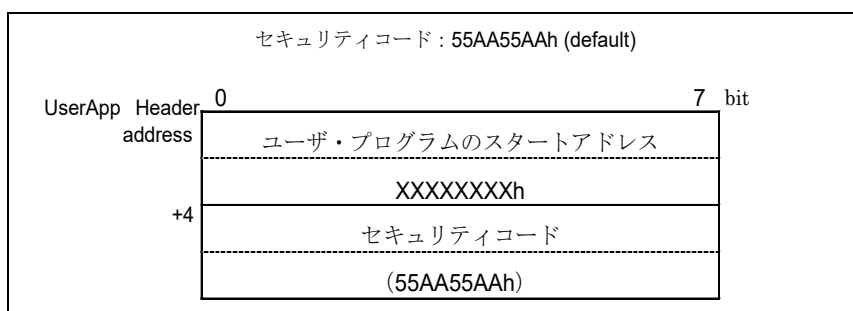


Figure 5-1 UserApp Header 領域

このヘッダ情報をUSB CDC経由内蔵FlashROM書き換えプログラムが起動時に読み取ることにより、UserAppの起動シーケンスに移行します。詳しくは、「7.3.1 電源投入時/Reset時の動作フロー」を参照して下さい。

5.3 固定ベクタ領域

ユーザプログラム(mot/hexファイル)には固定ベクタ領域を含めないでください。

Note:

固定ベクタは、USB CDC経由内蔵FlashROM書き換えプログラム側の固定ベクタが使用されます。

5.4 オプション設定メモリ

オプション設定メモリをサポートするMCUをご使用の場合、ユーザプログラム側のオプション設定メモリに対する設定は行わないでください。ユーザプログラム(mot/hexファイル)内にオプション設定メモリの設定がある場合、USB CDC経由内蔵FlashROM書き換えプログラムは、正常に動作しません。

Note:

オプション設定メモリの指定は、USB CDC経由内蔵FlashROM書き換えプログラムに対して行ってください。詳細については、「6.2 USB CDC経由内蔵FlashROM書き換えプログラムの設定」を参照してください。

5.5 バックアップ機能使用時のセクション設定

ユーザプログラムはArea1領域内で実行されますので、ユーザプログラムのcode属性およびromdata属性のセクション設定はArea1の領域を指定し、ビルドを行ってください。

- code属性 : 実行命令を格納します。
- romdata属性 : ROMデータを格納します。

Note:

バックアップ機能については、「7.2 バックアップ機能」を参照してください。

6. USB CDC経由内蔵FlashROM書き換えプログラムとユーザプログラムに対する設定

USB CDC経由内蔵FlashROM書き換えプログラムとユーザプログラムに対して必要な設定内容を以下に示します。

6.1 ユーザプログラムに対する設定

1. 設定内容 1

ユーザプログラムでは、Figure 6-1を参考にして UserApp Header 領域を作成してください。UserApp Header 領域については、「5.2 UserApp Header領域 (ユーザ・アプリケーション・ヘッダ)」を参照してください。

2. 設定内容 2

上記1で作成した UserApp Header 領域に対しセクションを設定し、そのセクションを必ずユーザプログラムの先頭に配置してください。

```

/*****
APPLICATION INTERFACE HEADER
The purpose of the header is for an external application to be able to read
certain values from known addresses.
- Start address of UserApp.
- Security code must match what PCDC Flashloader expects.
- For revision purposes of applications etc.
- Do not change the order of these variables!
*****/
#pragma section C UserApp_Head_Sect

/* START ADDRESS of user application header data - Appheader address + 0x00. */
const uint32_t userapp_entry_addr = (uint32_t) PowerON_Reset_PC;

/* - Appheader address + 0x04. */
const uint32_t userapp_sec_code = (uint32_t) USERAPP_SECURITY_CODE;

```

Figure 6-1 UserApp Header コード例

手順：

[プロパティ] → [C/C+ビルド] → [設定] を選択後、ツール設定タブを選び、 [Linker] → [セクション]を選択する。

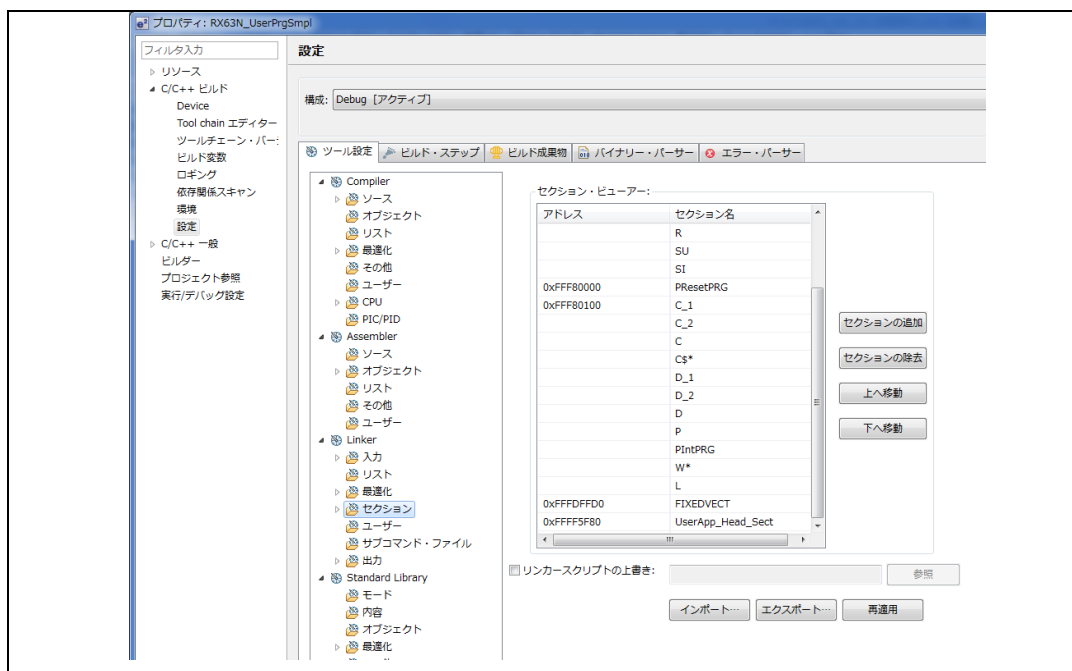


Figure 6-2 サンプル・プログラムのセクション設定例

6.2 USB CDC経由内蔵FlashROM書き換えプログラムの設定

1. 設定内容 1

"r_config\r_usb_fwupdater_config.h"ファイル内の以下の定義に対し、お客様のシステムに応じた設定を行ってください。

(1). USB モジュール指定

USB_CFG_USE_USBIP 定義に対し、使用する USB モジュール番号を指定してください。USB0 モジュールを使用する場合は USB_CFG_USE_USBIP 定義に対し、USB_CFG_IP0 を設定し、USB1 モジュールを使用する場合は、USB_CFG_IP1 を設定してください。

```
#define USB_CFG_USE_USBIP USB_CFG_IP0 // USB0モジュール使用設定
#define USB_CFG_USE_USBIP USB_CFG_IP1 // USB1モジュール使用設定
```

Note:

ご使用のMCUがUSBモジュールを1つしかサポートしていない場合には、USB_CFG_USE_USBIP 定義に対し、USB_CFG_IP0 を設定してください。

(2). Vendor ID, Product ID 指定

USB_CFG_VENDOR_ID 定義と USB_CFG_PRODUCT_ID 定義に対しお客様の Vendor ID と Product ID を指定してください。

```
#define USB_CFG_VENDOR_ID 0x0000 // Vendor ID値設定
#define USB_CFG_PRODUCT_ID 0x0002 // Product ID値設定
```

Note:

- a. USB_CFG_VENDOR_ID 定義には必ずお客様の Vendor ID を設定してください。
- b. 上記のマクロに設定した値を PC 側の INF ファイルにも設定してください。

(3). バックアップ機能設定

USB_CFG_BACKUP 定義に対し、バックアップ機能を使用するかどうかを指定してください。バックアップ機能を使用する場合は、USB_CFG_ON を指定し、バックアップ機能を使用しない場合は、USB_CFG_OFF を指定してください。

```
#define USB_CFG_BACKUP USB_CFG_ON // バックアップ機能使用設定
#define USB_CFG_BACKUP USB_CFG_OFF // バックアップ機能非使用設定
```

Note:

バックアップ機能については、「7.2 バックアップ機能」を参照してください。

(4). 使用パイプ設定

データ転送で使用するパイプ番号を設定してください。

a. Bulk IN, Bulk OUT 転送

Bulk IN, Bulk OUT 転送で使用するパイプ番号(USB_PIPE1 から USB_PIPE5)を指定してください。なお、USB_CFG_BULK_IN と USB_CFG_BULK_OUT に対し、同じパイプ番号は指定しないでください。

```
#define USB_CFG_BULK_IN パイプ番号 (USB_PIPE1からUSB_PIPE5)
#define USB_CFG_BULK_OUT パイプ番号 (USB_PIPE1からUSB_PIPE5)
```

b. Interrupt IN 転送

Interrupt IN 転送で使用するパイプ番号(PIPE6 から PIPE9)を指定してください。

```
#define USB_CFG_INTERRUPT_IN   パイプ番号   (USB_PIPE6からUSB_PIPE9)
```

(5). USB パワー設定

以下の定義に対し SelfPower / BusPower のいずれかを指定してください。

```
#define USB_CFG_POWER          USB_CFG_BUS           // Bus Power設定
#define USB_CFG_POWER          USB_CFG_SELF         // Self Power設定
```

(6). PLL クロック周波数設定

USBAa/USBA モジュールをご使用の場合、PHYSET レジスタの PLL クロックソース周波数設定ビット(CLKSEL) に対し、20MHz または 24MHz のいずれかを選択してください。

```
#define USB_CFG_CLKSEL         USB_CFG_24MHZ        // 24MHz 設定
#define USB_CFG_CLKSEL         USB_CFG_20MHZ        // 20MHz 設定
```

Note:

- USBAa/USBA モジュール以外の USB モジュールの場合、この定義は無視されます。
- USBAa/USBA モジュールは、RX71M/RX64M で使用されている USB モジュールです。

(7). CPU バスウェイト設定

USBAa/USBA モジュール内にある BUSWAIT レジスタに設定する数値を USB_CFG_BUSWAIT に対し指定してください。

```
#define USB_CFG_BUSWAIT       7                    // 7 ウェイト設定
```

Note:

- USB_CFG_BUSWAIT に指定する数値については、RX71M/RX64M のハードウェアマニュアル内の BUSWAIT レジスタの章を参照してください。
- USBAa/USBA モジュール以外の USB モジュールの場合、この定義は無視されます。
- USBAa/USBA モジュールは、RX71M/RX64M で使用されている USB モジュールです。

(8). USB レギュレータ設定

RX231 がサポートしている USB レギュレータ機能を使用するか、または使用しないかの設定を下記の定義に対し行ってください。

```
#define USB_CFG_REGULATOR     USB_CFG_OFF          // 非使用
#define USB_CFG_REGULATOR     USB_CFG_ON           // 使用
```

Note:

RX231 以外の MCU をご使用の場合、この定義は無視されます。

(9). その他の指定

USB CDC経由内蔵FlashROM書き換えプログラムは、ユーザプログラム内の UserApp Header 領域の内容を参照しますので、UserApp Header 領域の配置アドレスを変更した場合は、変更後の UserApp Header 領域を参照するよう本プログラムを変更する必要があります。また、セキュリティコードの値を変更した場合も、本プログラムの変更が必要になります。UserApp Header 領域の詳細については、「5.2 UserApp Header領域 (ユーザ・アプリケーション・ヘッダ)」を参照してください。

a. UserApp Header 領域の配置アドレス設定

USB_CFG_USERAPP_HEADER_ADDR 定義に対し、UserApp Header 領域の配置アドレスを設定してください。

```
#define USB_CFG_USERAPP_HEADER_ADDR   UserApp Header領域の配置アドレス
```

b. セキュリティコード設定

USB_CFG_USERAPP_SECURITY_CODE 定義に対し、UserApp Header 領域に設定したセキュリティコードを設定してください。

```
#define USB_CFG_USERAPP_SECURITY_CODE セキュリティコード
```

Note:

セキュリティコードには、0xFFFFFFFF 以外の値を設定してください。

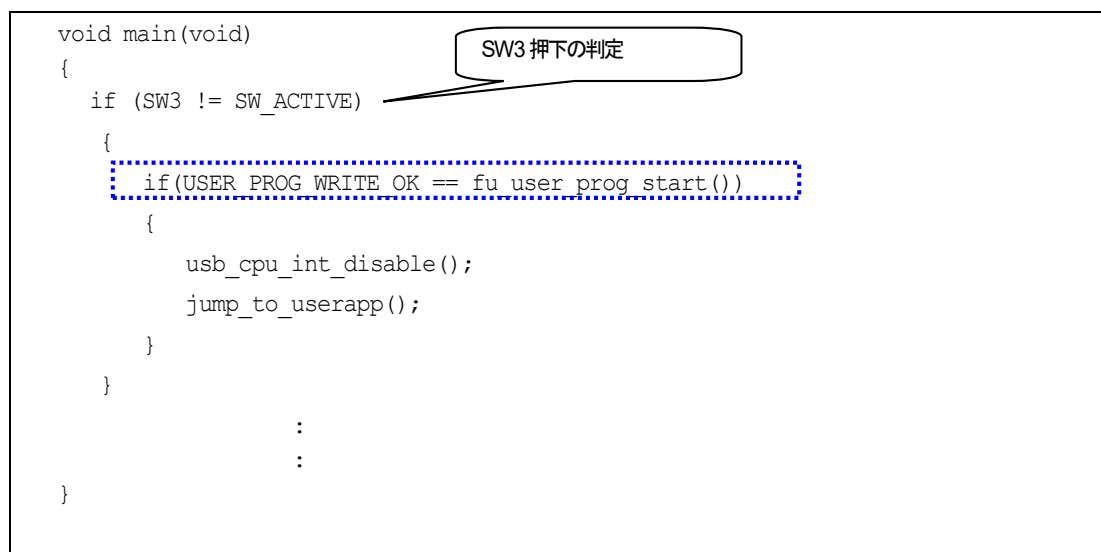
2. 設定内容 2

デュアルモードをサポートする MCU をご使用の場合、"r_config¥r_bsp_config.h"ファイル内の BSP_CFG_CODE_FLASH_BANK_MODE 定義に対し、0 (デュアルモード)または1(リニアモード)のいずれかを指定してください。

```
#define BSP_CFG_CODE_FLASH_BANK_MODE 0 // デュアルモード
#define BSP_CFG_CODE_FLASH_BANK_MODE 1 // リニアモード
```

3. 設定内容 3

USB CDC経由内蔵FlashROM書き換えプログラムは、評価ボード上の SW(スイッチ)の状態により、ROM 書き換え処理にジャンプするか、ユーザプログラムにジャンプするかの判定を行っています。この判定処理は、ボードの仕様に依存しますので、ご使用のボードに合わせた判定処理に変更いただきますようお願いいたします。判定処理は、main 関数内で行っています。



```
void main(void)
{
    if (SW3 != SW_ACTIVE)
    {
        if (USER PROG WRITE OK == fu user_prog start())
        {
            usb_cpu_int_disable();
            jump_to_userapp();
        }
    }
    :
    :
}
```

Figure 6-3 main()関数処理

4. 設定内容 4 (USB 端子設定)

お客様のシステムに応じた USB 端子の設定を行ってください。USB 端子設定は以下の関数で行われています。

```
ファイル名      :    demo_src¥main.c
関数名          :    usb_pin_setting()
```

5. 設定内容 5(オプション設定メモリ)

オプション設定メモリの設定は、下記の項目に対してのみ行うことが可能です。下記以外はすべてデフォルト値を設定してください。

(1). FASTSTUP ビット

- (2). LVDAS / STUPLVD1REN ビット
- (3). VDSEL / STUPLVD1LVL ビット
- (4). MDE ビット

なお、USB CDC経由内蔵FlashROM書き換えプログラムは、ユーザプログラム内にあるオプション設定メモリのROM書き込みを行いません。本プログラムのオプション設定メモリが、ユーザプログラムでも使用されますので、オプション設定メモリに対する設定は、本プログラムに対して行ってください。

Note:

- a. USB CDC経由内蔵FlashROM書き換えプログラムのオプション設定メモリの初期値は、すべてデフォルト値となっています。
 - b. RX62N は、オプション設定メモリをサポートしていません。
 - c. オプション設定メモリの詳細については、MCU の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. 設定内容 6 (コンパイルオプション)

上記1から4の設定後に行うコンパイルでは、以下に示すコンパイルオプションを指定してください。

- (1). USB CDC経由内蔵FlashROM書き換えプログラムをユーザブート領域外のROM領域に配置する場合
e² studio 上の[分岐幅]に対し[24 ビット以内]を選択してください。
- (2). USB CDC経由内蔵FlashROM書き換えプログラムをユーザブート領域に配置する場合
e² studio 上の[分岐幅]に対し[設定しない]を選択してください。

Note:

[分岐幅]は、[ファイル]->[プロパティ]->[C/C+ビルド]->[設定] を選択後、[Common]->[CPU] で指定できます。

7. 設定内容 7 (デュアルモード選択時)

上記*設定内容 2*でデュアルモードを指定した場合、アドレス 0xFFFFF7C に対し FW_CODE セクションを追加してください。

	W*
	L
	P*
0xFFFFF7C	FW_CODE
0xFFFFF80	EXCEPTVECT
0xFFFFFFC	RESETVECT

6.3 ユーザプログラムの配置

ユーザプログラムは、USB CDC経由内蔵FlashROM書き換えプログラムが書き込まれたROM領域と重ならない領域に配置してください。ユーザプログラムの配置設定は、セクション設定により行ってください。

Note:

1. ユーザプログラムが以下のROM領域に配置されるように配置設定を行ってください。なお、デュアルモード選択時は、起動バンク領域にユーザプログラムが配置されるよう設定を行ってください。

バックアップ機能	ユーザプログラム配置可能領域
----------	----------------

OFF	プログラム ROM 領域先頭アドレス	-	0xFFFFDFFB
ON	プログラム実行領域先頭アドレス	-	0xFFFFDFFB

Note:

バックアップ機能およびプログラム実行領域については、「7.2 バックアップ機能」を参照してください。

2. 0xFFFFDFFC – 0xFFFFDFFFの領域(4バイト)は、USB CDC経由内蔵FlashROM書き換えプログラムが管理領域として使用します。
3. Flash Self programmingライブラリはRAM領域の一部を使用しますが、USB CDC経由内蔵FlashROM書き換えプログラム実行時にのみ使用するため、ユーザプログラムの動作には影響がありません。

7. USB CDC経由内蔵FlashROM書き換えプログラムの解説

この章では、USB CDC経由内蔵FlashROM書き換えプログラムで使用している各ファイルについて説明します。

7.1 ファイル・フォルダ構成

USB CDC経由内蔵FlashROM書き換えプログラムのソース・ファイルとフォルダ構成を示します。

(MCU 名)		ビルド結果
+HardwareDebug		
+src		
+----r_config	[API 設定ファイル]	
+----r_flash_rx	[Simple Flash API]	
	+---- src	[FlashAPI ドライバ]
	+---- flash_type_1	Flash 書き込みタイプ 1 API
	+---- flash_type_2	Flash 書き込みタイプ 2 API
	+---- flash_type_3	Flash 書き込みタイプ 3 API
	+---- flash_type_4	Flash 書き込みタイプ 4API
	+---- targetets	各種 MCU ROM 情報
+----r_bsp	[Renesas Board Support Package]	
	+---- board	BSP 各 RSK/RSSK 用設定
	+---- mcu	BSP 各 MCU 情報
+----demo_src	[サンプルアプリケーション]	
	+---- inc	サンプルアプリケーションヘッダファイル
+----USB	[USB driver]	
	+---- inc	USB ドライバ共通ヘッダファイル
	+---- src	USB ドライバ

Figure 7-1 USB CDC経由内蔵FlashROM書き換えプログラムのフォルダ構成

なお、本プログラムは以下のパッケージを使用しています。

- r_bsp (Renesas board support package)
- r_flash_rx (RX family simple flash module)

7.1.1 src¥r_config フォルダ

対象 MCU の設定などの設定ファイルが格納されているフォルダです。

Table 7-1 API ヘッダファイル

ファイル名	説明
r_bsp_config.h	BSPの設定ヘッダファイル
r_flash_rx_config.h	Flash書き込み設定ファイル
r_usb_fwupdater_config.h	Flash ROM書き換えプログラム設定ファイル

7.1.2 src¥r_flash_rx フォルダ

シンプルフラッシュAPIのソース・ファイル, およびヘッダファイルが格納されているフォルダです。詳細は、Flash Module Using Firmware Integration Technology のアプリケーションノートを参照してください。

ボードサポートパッケージ(r_bsp)でMCUを選択すると、フラッシュ書き込みタイプが自動的に選択されます。

7.1.3 src¥r_bsp フォルダ

Renesas Board support package moduleのソース・ファイルおよびヘッダファイルが格納されているフォルダです。詳細は、RXファミリ ボードサポートパッケージモジュール アプリケーションノートを参照してください。

7.1.4 src¥demo_src フォルダ

USB CDC経由内蔵FlashROM書き換えプログラムのソース・ファイルが格納されているフォルダです。

Table 7-2 USB CDC経由内蔵FlashROM書き換えプログラムのソース・ファイル

ファイル名	説明
main.c	C言語メイン関数記述ファイル
r_usb_pcdc_apl.c	USB送受信処理ファイル
r_fwupdater_apl.c	ROM書き換えプログラム処理ファイル (ROM書き込みデータ作成)
r_flash_apl.c	フラッシュAPI呼び出し処理ファイル (ROM書き換え処理)
r_usb_descriptor.c	ディスクリプタ定義
inc¥r_usb_pcdc_apl.h	USB送受信処理ヘッダファイル
inc¥r_fwupdater_apl.h	ROM書き換えプログラムヘッダファイル
inc¥r_flash_apl.h	フラッシュ呼び出し処理ヘッダファイル

7.1.5 src¥USB フォルダ

CDC(USB)のソース・ファイル, およびヘッダファイルが格納されているフォルダです。

Table 7-3 USB CDC経由内蔵FlashROM書き換えプログラムのソース・ファイル

ファイル名	説明
inc¥r_usb_reg.h	USBレジスタの初期化、設定用定義
inc¥r_usb_define.h	USB用定義
inc¥r_usb_extern.h	関数Extern
src¥r_usb_api.c	USB 送受信、初期化処理ファイル
src¥r_usb_driver.c	USBドライバ処理
src¥r_usb_classcdc.c	USB CDC処理
src¥r_usb_rx_mcu.c	USB割り込み初期化、ポート設定ファイル
src¥r_usb_reg.c	USB レジスタ設定など

7.1.6 HardwareDebug フォルダ

ビルド時にUSB CDC経由内蔵FlashROM書き換えプログラムの実行可能なオブジェクト・ファイルとmotファイルが格納されるフォルダです。

Table 7-4 CDCプログラムのオブジェクト・ファイル

ファイル名	説明
(MCU名)_FirmwareUpdater.mot	motフォーマットの実行可能オブジェクト・ファイル

7.2 バックアップ機能

USB CDC経由内蔵FlashROM書き換えプログラムは、FlashROM 書き換え中、USB 通信失敗等により FlashROM 書き換えに失敗しても、すでに特定領域に保持されているユーザプログラムが起動するバックアップ機能をサポートしています。

バックアップ機能の FlashROM 書き換え処理概要は以下の通りです。

1. USB CDC経由内蔵FlashROM書き換えプログラムは、内蔵 FlashROM(プログラム ROM 領域)を2つの領域に分け、1つの領域をプログラム実行領域(Area1)、もう1つの領域をユーザプログラム格納領域(Area2)として使用します。Area1 と Area2 は内蔵 FlashROM 領域の中心から2つに分けた領域です。これらの2つの ROM 領域のサイズは同一です。なお、Area2 の領域には未使用領域が存在します。これは、Area1 領域内のUSB CDC経由内蔵FlashROM書き換えプログラムが Area2 の領域には存在しないためです。



Figure 7-2 バックアップ機能使用時の Flash ROM 領域

2. バックアップ機能を有効にした場合、USB CDC経由内蔵FlashROM書き換えプログラムは、必ず Area2 の領域にユーザプログラム(mot1)の書き込みを行います。

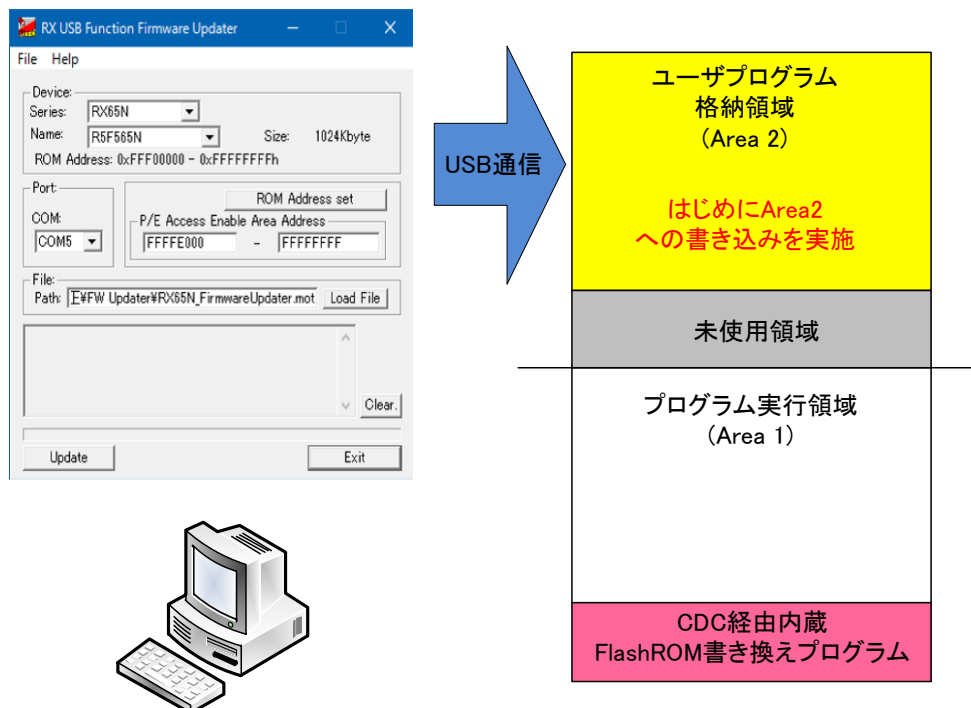


Figure 7-3 内蔵 FlashROM(Area2)へのユーザプログラムの書き込み

3. 書き込み正常終了後、USB CDC経由内蔵FlashROM書き換えプログラムが Area2 から Area1 へコピー処理を行います。Area1 へのコピー完了後、Area1 内にあるユーザプログラムが実行されます。

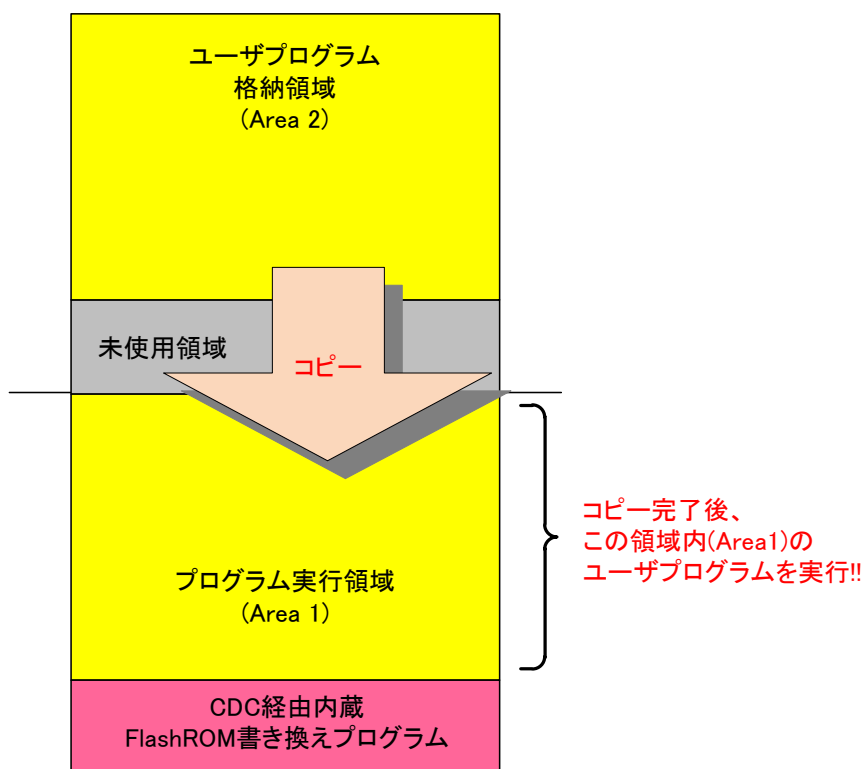


Figure 7-4 ユーザプログラムのコピー処理

4. USB CDC経由内蔵FlashROM書き換えプログラムによりユーザプログラムを更新する場合、Area2 の領域を消去した後、Area2 への FlashROM 書き込みが行われ、書き込み完了後、Area1 へのコピー処理が行われます。

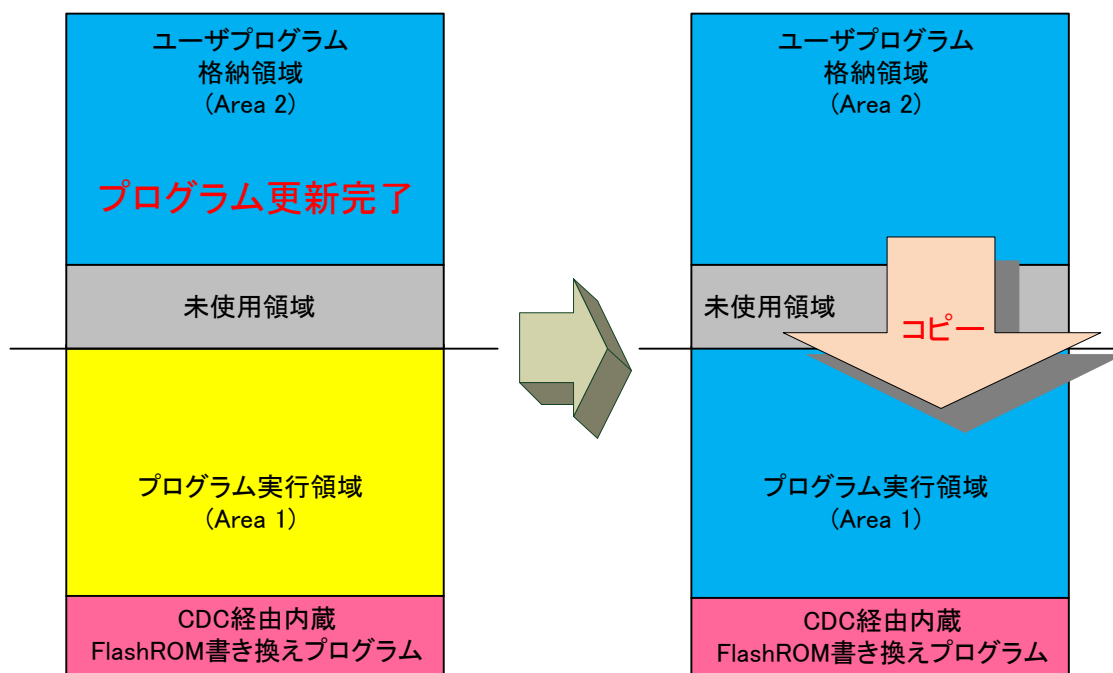


Figure 7-5 ユーザプログラムの更新

Note:

Area2 への書き込みが正常完了した後、なんらかの理由により Area1 を消去できなかった場合、Area1 内のユーザプログラムは更新されず、以前、Area1 に書き込んだユーザプログラムが再度起動します。Area1 を消去できない現象が発生した場合は、Area2 への書き込み処理を再度実行してください(上記2参照)。Area1 を消去することができなかった場合、ファイル転送アプリケーション(PC ツール)上のメッセージ"Now Copying"の次の行にメッセージ"ERR: Writing process stop."または"ERR: Data reception error."が表示されます。

5. Area2 への書き込み中、USB 通信失敗等により FlashROM 書き込みが失敗しても、Area1 領域には、上記4で書き込まれたユーザプログラムは保持されたままです。FlashROM 書き込み失敗前のユーザプログラムが起動されます。

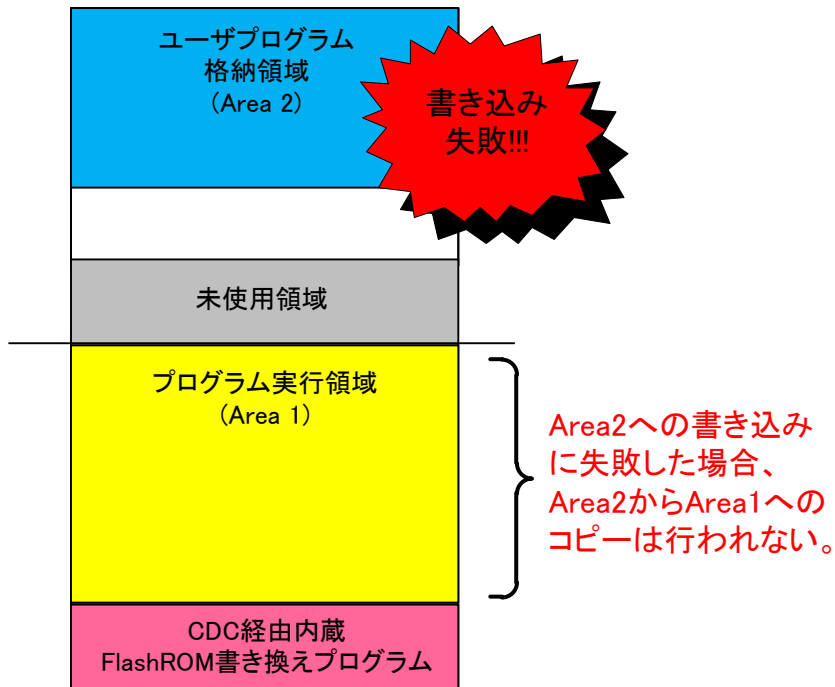


Figure 7-6 ユーザプログラムの更新失敗

Note:

1. Area2 から Area1 へのコピー中、何らかの理由によりそのコピーに失敗した場合、RSK/RSSK のリセット/電源投入を行ってください。再度USB CDC経由内蔵FlashROM書き換えプログラムが Area2 から Area1 へのコピー処理を行います。コピー処理が正常に終了するとユーザプログラムが起動されます。このコピー処理には RSK/RSSK のリセット/電源投入から最大 10 秒程度の時間を要します。

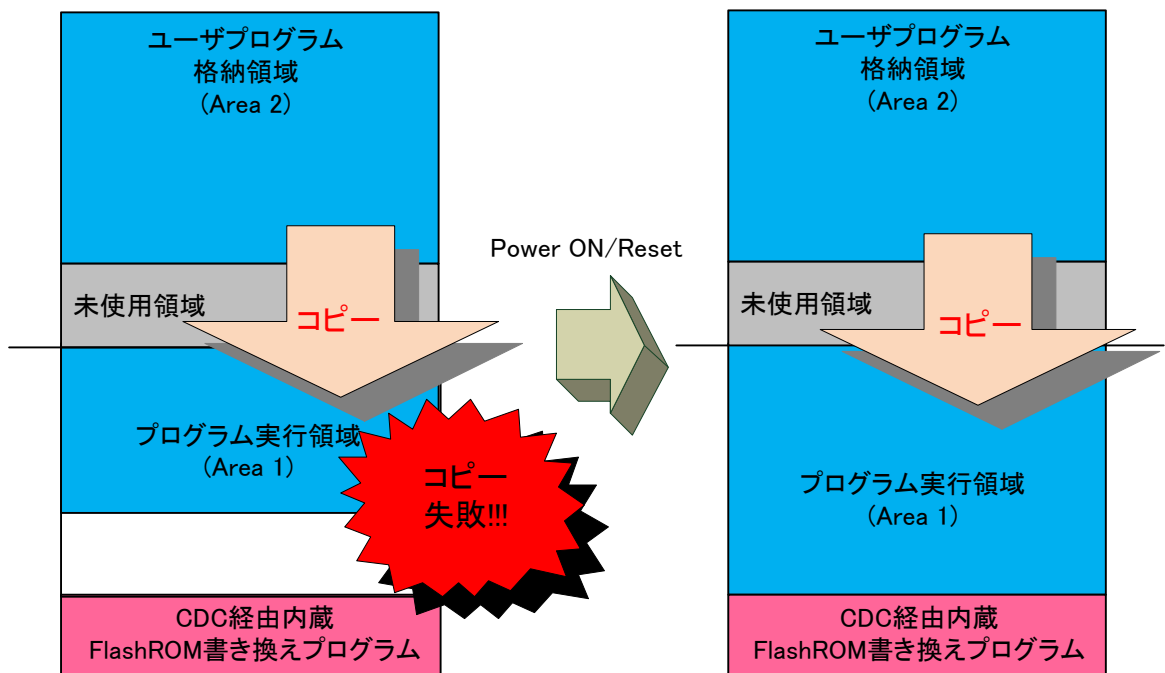


Figure 7-7 Area2 から Area1 へのコピー失敗

2. ユーザプログラムは Area1 領域内で実行されますので、ユーザプログラムの code 属性および romdata 属性のセクション設定は Area1 の領域を指定し、ビルドを行ってください。

code属性 : 実行命令を格納します。
romdata属性 : 固定データを格納します。

3. バックアップ機能のサポート/非サポートは r_usb_fwupdater_config.h 内のマクロ定義に対する設定により行います。この設定の詳細については、「6.2 USB CDC経由内蔵FlashROM書き換えプログラムの設定」を参照してください。
4. デュアルモードをサポートする MCU をご使用の場合、デュアルモードをご使用ください。

7.3 ブート処理

ブート処理とは、マイコンをリセット後、メイン関数（C言語記述：main()）が実行される前に実行する処理を指します。

RXマイコンでは、リセット後の初期化処理として、主に次のことを行います。

- スタック領域の確保とスタック・ポインタの設定
- main 関数の引数領域の確保
- data 領域、スタック領域の初期化
- hdwinit 関数でのユーザプログラムへの分岐及び MCU 周辺デバイスの初期化
- main 関数への分岐

リセット後、USB CDC経由内蔵FlashROM書き換えプログラムからユーザプログラムへジャンプしてくるため、必ずUSB CDC経由内蔵FlashROM書き換えプログラムが処理され、上記に記載したマイコンの初期化処理等が行われます。

7.3.1 電源投入時/Reset 時の動作フロー

次に、USB CDC経由内蔵FlashROM書き換えプログラムの電源投入時の動作フローについて説明します。

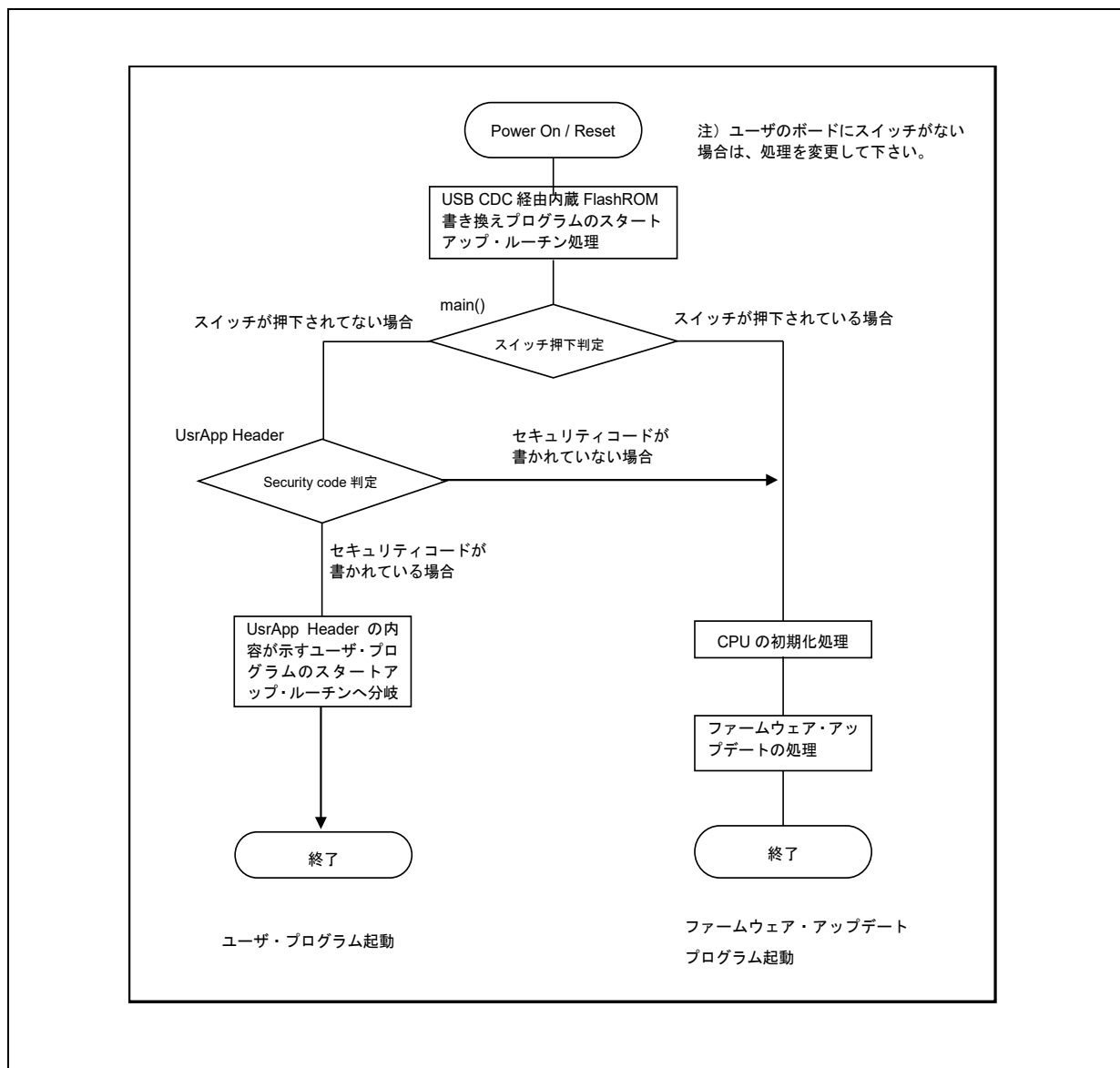


Figure 7-8 電源投入/Reset 時の動作フロー

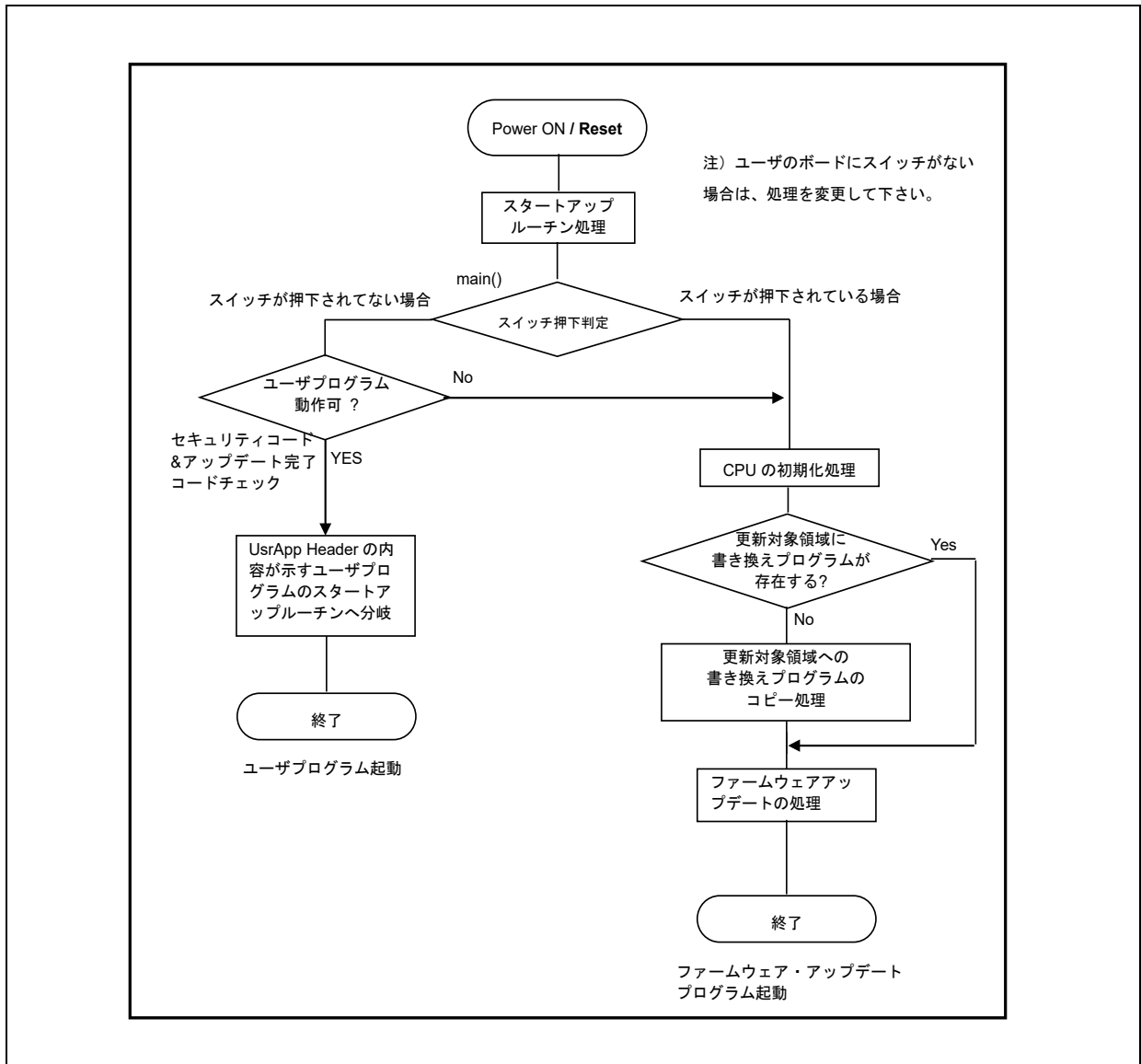


Figure 7-9 電源投入時/リセット時の動作フロー (デュアルモード使用時)

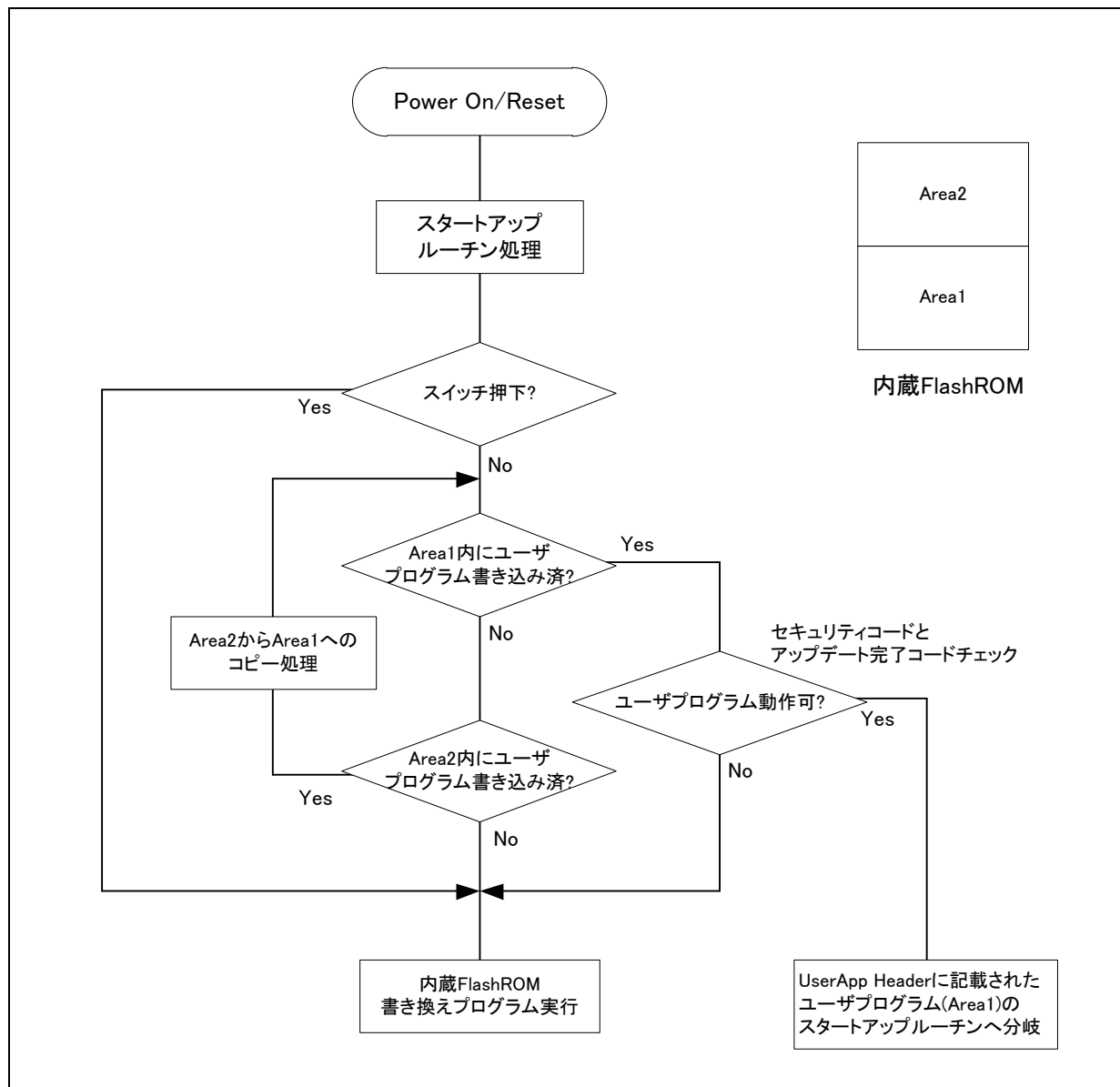


Figure 7-10 電源投入時/リセット時の動作フロー (バックアップ機能使用時)

セキュリティコードおよびユーザプログラムへの分岐アドレス情報に関しては、「5.2 UserApp Header領域 (ユーザ・アプリケーション・ヘッダ)」を参照して下さい。

なお、UserApp Header領域にセキュリティコードが正常に設定されていても、ユーザプログラムのスタートアドレスが正しくない場合、ユーザプログラムは動作しません。

7.3.2 ユーザプログラムの起動条件

以下の条件をすべて満たした場合、UsrApp Header 領域に設定されたユーザプログラムが起動します。

1. 正しいセキュリティコードが設定されている。
2. 正しいユーザプログラムの先頭アドレスが設定されている。
3. アップデート完了コードが正常に書かれている。

正常に書き込みが完了した時に、当該書き換えプログラムが自動的にアップデート完了コードの書き込みを行います。

なお、セキュリティコードおよびアップデート完了コードが不一致(正しくない)の場合、USB CDC経由内蔵FlashROM書き換えプログラムが起動し、ユーザプログラムは起動しません。

7.3.3 USB CDC経由内蔵FlashROM書き換えプログラムの起動条件

1. ユーザプログラムが ROM に書かれている場合

評価ボード上のスイッチ(RSK:スイッチ 3, RSSK:スイッチ 2)を押下した状態でリセット起動するとUSB CDC経由内蔵FlashROM書き換えプログラムが起動します。

2. ユーザプログラムが ROM に書かれていない場合

USB CDC経由内蔵FlashROM書き換えプログラムが起動します。

7.4 注意事項

1. USB CDC経由内蔵FlashROM書き換えプログラムは、評価ボード上のスイッチ(RSK:スイッチ 3, RSSK:スイッチ 2)の状態により、USB CDC経由内蔵FlashROM書き換えプログラムにジャンプするか、またはユーザプログラムにジャンプするかの判定を行っています。この判定処理は、ボードの仕様に依存しますので、ご使用のボードに合わせた判定処理に変更いただきますようお願いいたします。判定処理を行っている部分は、USB CDC経由内蔵FlashROM書き換えプログラムの main 関数です。
2. USB CDC経由内蔵FlashROM書き換えプログラムは、書き込み先アドレスがFlashROM範囲内かどうかのチェックを行っていませんのでご注意ください。

7.5 USB CDC 経由内蔵 FlashROM 書き換え用関数

ここでは、USB CDC経由内蔵FlashROM書き換えプログラム内で使用する関数を示します。
ただし、BSP、シンプルフラッシュAPI関連の関数を除きます。

7.5.1 データ・タイプ

USB CDC経由内蔵FlashROM書き換えプログラムにおける、データ・タイプを下表に示します。

Table 7-5 データ・タイプ

データ・タイプ	指定子	有効範囲
int8_t	signed char	符号つき8ビット整数
int16_t	signed short	符号つき16ビット整数
int32_t	signed long	符号つき32ビット整数
uint8_t	unsigned char	符号なし8ビット整数
uint16_t	unsigned short	符号なし16ビット整数
uint32_t	unsigned long	符号なし32ビット整数

7.5.2 構造体

Table 7-6 response_record_t 構造体

データ・タイプ	変数名	内容
uint32_t	record_type	レコード種別
uint8_t	record_len	レコード長
uint8_t	response_type	応答種別 ACK/NAK
uint8_t	err_field	エラー・コード
uint8_t	checksum	チェック・サム

Table 7-7 rom_write_buf_t 構造体定義

データ・タイプ	変数名	内容
uint8_t	data[ROM_WRITE_SIZE]	ROM書き込み用バッファ
uint32_t	dest_addr	書き込み先アドレス
uint16_t	data_flag	データ格納フラグ 0: なし 1: データあり

Table 7-8 rom_erase_addr_t 構造体定義

データ・タイプ	変数名	内容
uint32_t	start_addr	ROM消去開始アドレス
uint32_t	end_addr	ROM消去終了アドレス

7.5.3 Flash 書き込みメイン処理関数

Table 7-9 メイン処理関数一覧

ファイル名	関数名	処理概要
main.c	main	USB 端子設定、FlashROM 書き換えプログラム/ユーザプログラムの分岐処理
r_usb_pcdc_apl.c	usb_main	初期化、メイン処理
r_usb_pcdc_apl.c	fu_cdc_read	USB CDC データ受信要求処理
r_usb_pcdc_apl.c	fu_main	Flash ROM 書き換えメイン処理
r_usb_pcdc_apl.c	usb_send_response_record	USB Host(GUI ツール)へのデータ応答処理
r_usb_pcdc_apl.c	jump_to_userapp	ユーザプログラムへのジャンプ処理
r_usb_pcdc_apl.c	usb_transfer_complete	USB データ送受信完了フラグ切り替え処理
r_fwupdater_apl.c	fl_write_data_init	フラッシュ書き込み用変数の初期化
r_fwupdater_apl.c	fl_erase_area	フラッシュイレース処理呼び出し
r_fwupdater_apl.c	fl_write_data	フラッシュ書き込み判定、書き込み処理呼び出し
r_fwupdater_apl.c	fu_check_security_code	セキュリティコードチェック
r_fwupdater_apl.c	fu_byte2num	4 バイトデータ(アドレス)を unsigned long 型に変換
r_flash_apl.c	fl_rom_write	Flash ROM 書き込み処理
r_flash_apl.c	fl_rom_erase	Flash ROM 消去処理
r_flash_apl.c	fl_set_access_window	Flash ROM アクセス許可設定処理。Flash Type1 のみ。
r_flash_apl.c	fl_get_blk_num	Flash ROM 開始アドレス、終了アドレスからブロック数、ブロック位置情報を取得
r_flash_apl.c	fl_get_blk_addr	Flash ROM アドレスから、該当アドレスが属する ROM ブロックの開始アドレスを取得

Table 7-10 main 関数

関数名	main	
記述形式	void main (void)	
機能	初期化処理とUSB CDC経由内蔵FlashROM書き換えプログラムとユーザプログラムの分岐を行う	
入出力	入力	なし
	出力	なし
備考	動作の詳細は「7.5.5USB CDC経由内蔵FlashROM書き換えプログラムへの分岐」をご参照下さい。	

Table 7-11 usb_main 関数

関数名	usb_main
記述形式	void usb_main (void)
機能	初期化処理、メイン処理を行う

入出力	入力	なし
	出力	なし
備考		動作の詳細は「7.5.5USB CDC経由内蔵FlashROM書き換えプログラムへの分岐」をご参照下さい。

Table 7-12 fu_cdc_read 関数

関数名		fu_cdc_read
記述形式		static uint16_t fu_cdc_read(void)
機能		USBデータ受信要求処理
入出力	入力	なし
	出力	uint16_t : read結果
備考		CDC_BLK_OUT_OK : 読み込み完了 CDC_NO_CONFIGURED : CDC未接続 CDC_DETCH : CDC接続エラー CDC_BLK_OUT_ERR : 読み込み異常

Table 7-13 fu_main 関数

関数名		fu_main
記述形式		void fu_main (void)
機能		内蔵FlashROM書き換えプログラムメイン処理
入出力	入力	なし
	出力	なし
備考		--

Table 7-14 usb_send_response_record 関数

関数名		usb_send_response_record
記述形式		static void usb_send_response_record (uint8_t response_type, uint8_t response_field)
機能		ホスト側にデータを送信する
入出力	入力	なし
	出力	なし
備考		通信プロトコルに関しては、「9 データ通信仕様」をご参照下さい。

Table 7-15 jump_to_userapp 関数

関数名		jump_to_userapp
記述形式		static void jump_to_userapp (void)
機能		ユーザプログラムへジャンプする
入出力	入力	なし
	出力	なし
備考		ジャンプ先のアドレス情報に関しては、「5.2 UserApp Header領域 (ユー

「ザ・アプリケーション・ヘッダ」を参照して下さい。

Table 7-16 usb_transfer_complete 関数

関数名	usb_transfer_complete	
記述形式	void usb_transfer_complete (void)	
機能	送受信フラグを変更する	
入出力	入力	なし
	出力	なし
備考	なし	

Table 7-17 fl_write_data_init 関数

関数名	fl_write_data_init	
記述形式	void fl_write_data_init (void)	
機能	フラッシュ書き込み用変数初期化	
入出力	入力	なし
	出力	なし
備考	なし	

Table 7-18 fl_erase_area 関数

関数名	fl_erase_area	
記述形式	flash_err_t fl_erase_area(void)	
機能	フラッシュイレース処理	
入出力	入力	なし
	出力	flash_err_t : イレース結果
備考	なし	

Table 7-19 fl_write_data 関数

関数名	fl_write_data	
記述形式	flash_err_t fl_write_data(void)	
機能	フラッシュ書き込み処理	
入出力	入力	なし
	出力	flash_err_t : 書き込み結果判定
備考	なし	

Table 7-20 fu_check_security_code 関数

関数名	fu_check_security_code	
記述形式	flash_err_t fu_check_security_code(void)	
機能	セキュリティコードチェック	

入出力	入力	なし
	出力	flash_err_t: セキュリティコードチェック、ROMイレース結果
備考		なし

Table 7-21 fu_byte2num 関数

関数名		fu_byte2num
記述形式		static uint32_t fu_byte2num(uint8_t * dat, uint16_t size)
機能		4バイトデータ(アドレス)をunsigned long型に変換
入出力	入力	dat: byte列 size: 接続するサイズ
	出力	uint32_t: 算出結果を返却
備考		なし

Table 7-22 fl_rom_write 関数

関数名		fl_rom_write
記述形式		flash_err_t fl_rom_write (void)
機能		ROM書き込みAPI呼び出し窓口関数。Type1により処理分岐
入出力	入力	なし
	出力	flash_err_t: 処理結果
備考		なし

Table 7-23 fl_rom_erase 関数

関数名		fl_rom_erase
記述形式		flash_err_t fl_rom_erase (const uint32_t start_addr, const uint32_t end_addr)
機能		ROM消去API呼び出し窓口関数。Type1により処理分岐
入出力	入力	start_addr: イレース開始アドレス (アドレスを含むブロックを消去) end_addr: イレース終了アドレス (アドレスを含むブロックを消去)
	出力	flash_err_t: 処理結果
備考		Type1,3,4は一括で消去指定をしていますが、Type2はAPI側の処理でArea境界判定を行っているため1回の指定でAreaを超えるサイズを指定出来ません。よって都合上1blockずつの消去指定となっています。

Table 7-24 fl_set_access_window 関数

関数名		fl_set_access_window
記述形式		flash_err_t fl_set_access_window (const uint32_t start_addr, const uint32_t end_addr)
機能		ROMアクセス領域設定API呼び出し窓口関数。
入出力	入力	start_addr: ROMアクセス許可開始アドレス

力		end_addr : ROMアクセス許可終了アドレス
	出力	flash_err_t : 処理結果
備考	Flash Type1のみ行う処理となります。アクセス許可アドレスは10bitシフトした状態で保持されるため、終了アドレスは10bit切り捨てられることを考慮したアドレスが設定されます。アクセス許可領域の設定となるため、広めに設定する分には処理的に問題ありません。	

Table 7-25 fl_get_blk_num 関数

関数名	fl_get_blk_num	
記述形式	static uint32_t fl_get_blk_num(const uint32_t iaddr_start, const uint32_t iaddr_end, uint16_t *start_blk, uint16_t *end_blk)	
機能	ROM開始アドレス、終了アドレスからブロック数、ブロック番号情報を算出	
入出力	入力	iaddr_start : 開始アドレス指定 iaddr_end : 終了アドレス指定 start_blk : 開始ブロック番号 sta_end : 終了ブロック番号
	出力	uint32_t : 開始～終了アドレス間のブロック数
備考	使用する定義はFlash APIのROM情報の定義に依存します。ブロック番号はROMの後ろから割り当てられるので、StartAddress=EndBlock, EndAddress=StartBlockとなることに注意してください。	

Table 7-26 fl_get_blk_addr 関数

関数名	fl_get_blk_addr	
記述形式	static flash_block_address_t fl_get_blk_addr(const uint32_t iaddr)	
機能	ROMアドレスから、該当アドレスが属するROMブロック開始アドレスを算出	
入出力	入力	iaddr : ブロック開始アドレス算出用ROMアドレス
	出力	flash_block_address_t : ブロック開始アドレス
備考	使用する定義はFlash APIのROM情報の定義に依存します。	

7.5.4 USB ドライバ関数

Table 7-27は、USB ドライバの関数一覧です。

Table 7-27 USB モジュール関数一覧

ファイル名	関数名	処理概要
r_usb_api.c	usb_bulk_in_start	Bulk データ送信要求
r_usb_api.c	usb_bulk_out_start	Bulk データ受信要求
r_usb_api.c	usb_driver_init	USB 初期化处理
r_usb_driver.c	usb_int_isr	USB 割り込み処理
r_usb_driver.c	usb_save_request	リクエスト情報の取得
r_usb_driver.c	usb_ctrl_read_data_stage	Control リードデータステージ処理
r_usb_driver.c	usb_ctrl_write_nodata_stage	ノーデータ Control ステータスステージ処理
r_usb_driver.c	usb_intr_int_pipe0	PIPE0 用 USB BRDY 割り込み処理
r_usb_driver.c	usb_bemp_int_pipe0	PIPE0 用 USB Empty 割り込み処理
r_usb_driver.c	usb_intr_int	Bulk データ送受信
r_usb_driver.c	usb_intr_int_read	Bulk データ受信
r_usb_driver.c	usb_intr_int_write	Bulk データ送信
r_usb_driver.c	usb_ctr_read_start	Control データ送信要求
r_usb_driver.c	usb_ctr_write_start	Control データ受信要求
r_usb_driver.c	usb_write_fifo	USB FIFO へのデータ書き込み
r_usb_driver.c	usb_read_fifo	USB FIFO からのデータ読み出し
r_usb_driver.c	usb_chk_frdy	FRDY bit(USB モジュール) チェック
r_usb_driver.c	usb_chg_port	USB Pipe 切り替え
r_usb_driver.c	usb_req_get_descriptor	標準リクエスト処理
r_usb_driver.c	usb_req_set_configuration	標準リクエスト処理
r_usb_classcdc.c	usb_reset_ep	USB Pipe Configuration 処理
r_usb_classcdc.c	usb_cdc_init	Serial initialize
r_usb_classcdc.c	usb_class_write_data_stage	クラスリクエストライトデータステージ処理
r_usb_classcdc.c	usb_class_read_data_stage	クラスリクエストリードデータステージ処理
r_usb_classcdc.c	usb_class_write_nodata_stage	クラスリクエストノーデータステータスステージ処理
r_usb_rx_mcu.c	usb_cpu_mcu_initialize	MCU 初期化
r_usb_rx_mcu.c	usb_int_init	USB 割り込み初期化
r_usb_rx_mcu.c	usb_cpu_delay_1us	ウェイト関数(us)
r_usb_rx_mcu.c	usb_cpu_delay_1ms	ウェイト関数(ms)
r_usb_rx_mcu.c	usb_cpu_int_disable	USB 割り込み禁止
r_usb_rx_mcu.c	usb_cpu_usbint_init	USB 割り込み初期化

7.5.5 USB CDC経由内蔵FlashROM書き換えプログラムへの分岐

USB CDC経由内蔵FlashROM書き換えプログラムのmain()関数内でユーザプログラムにジャンプするかUSB CDC経由内蔵FlashROM書き換えプログラムを継続するかの分岐判定を行います。

条件分岐を経て、CPU内蔵機能・周辺回路の初期化を行ったあと、USB CDC経由内蔵FlashROM書き換えプログラムを実行します。

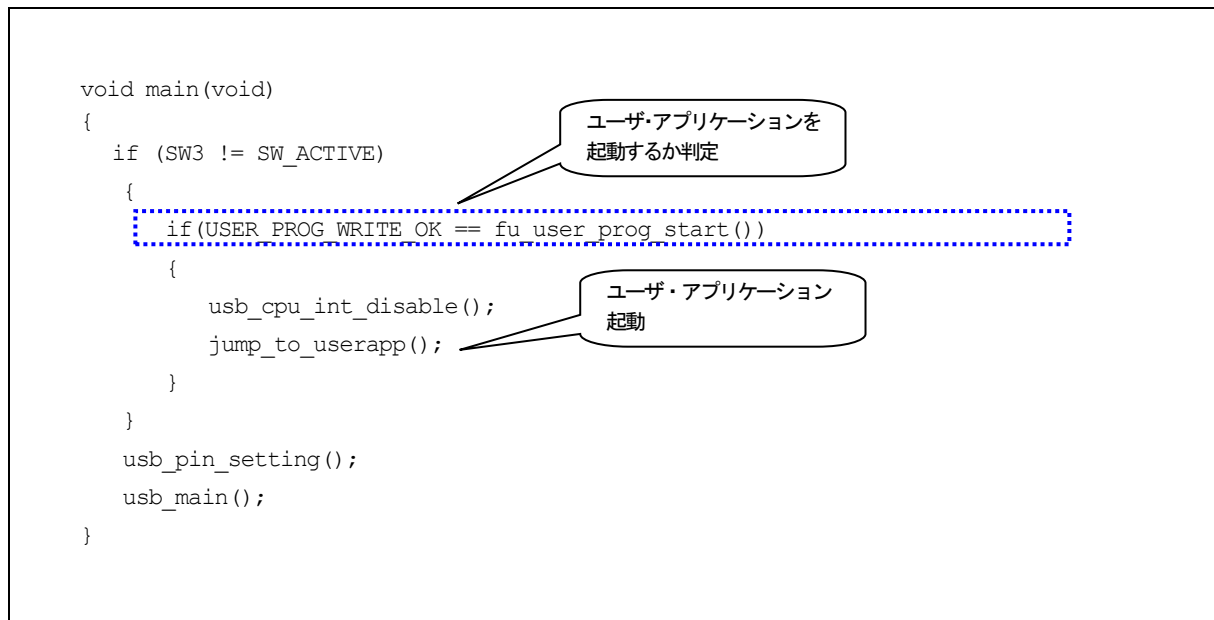


Figure 7-11 main()関数

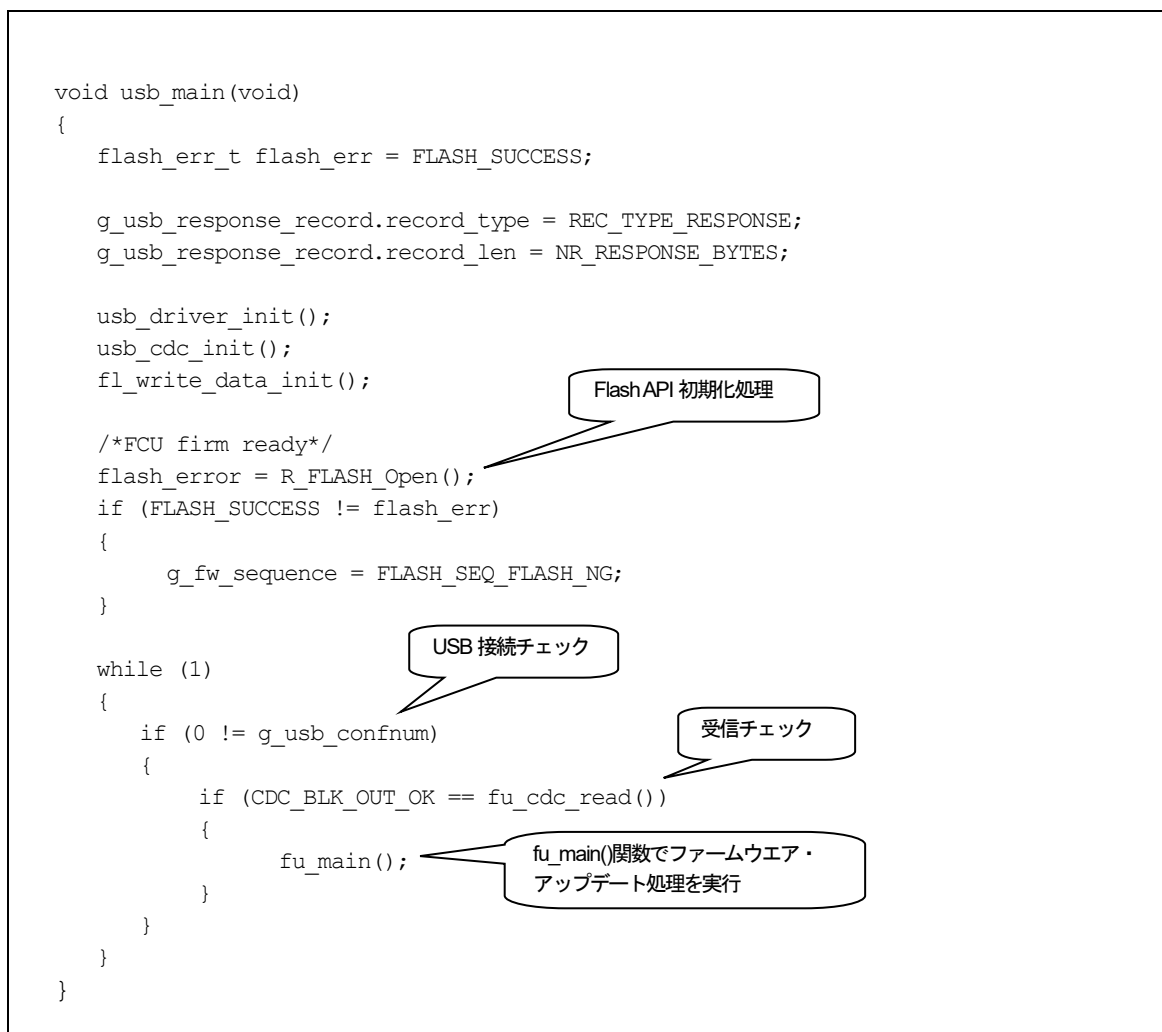


Figure 7-12 usb_main()関数

7.5.6 ユーザ・アプリケーションへのジャンプ

ユーザプログラムへのジャンプ処理は、`jump_to_userapp()` 関数によって行われます。なお、ジャンプ先であるユーザプログラムの先頭アドレス指定については、「5.2 UserApp Header領域 (ユーザ・アプリケーション・ヘッダ)」を参照してください。

8. ファイル転送アプリケーション（RX USB Firmware Updater）の解説

この章では、PC上で動作するファイル転送アプリケーションについて記載します。

8.1 開発環境

ファイル転送アプリケーションは、次に示す環境で構築されています。

OS : Windows 8.1, Windows 10

開発環境 : Visual Studio 2017

8.2 動作概要

ファイル転送アプリケーションは、起動時に引数として、書き換え対象のファイル名（およびオプション）を受け取ると、直接書き換え処理に移行します。ファイルの指定がない場合は、設定画面を表示します。

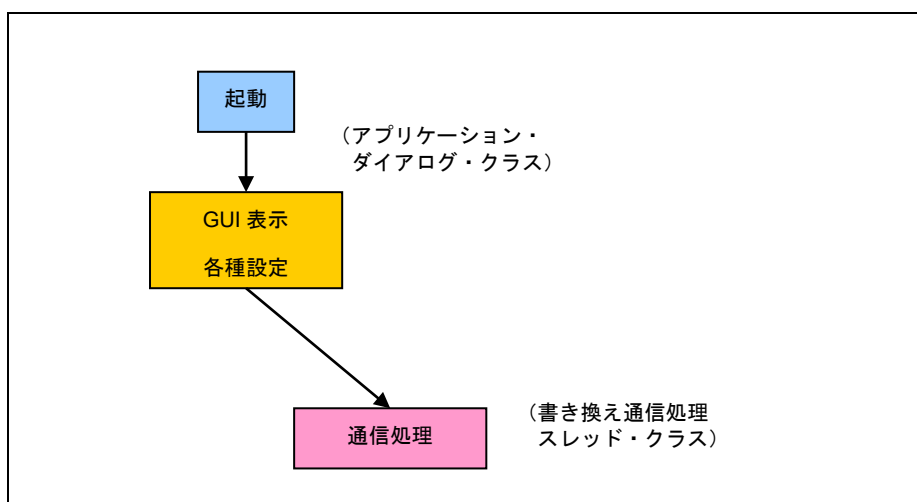


Figure 8-1 ファイル転送アプリケーション動作概要

8.3 ファイル構成

ファイル転送アプリケーションのファイル一覧を次に示します（主要なファイルのみを記載しています）。

Table 8-1 ファイル転送アプリケーションのファイル一覧

ファイル名	説明
FlashSelfRewriteGUI.sln	ソリューション・ファイル
FlashSelfRewriteGUI.rc	リソース・ファイル
FlashSelfRewriteGUI.cpp	アプリケーション・クラスの処理ファイル
FlashSelfRewriteGUI.h	アプリケーション・クラスの定義ファイル
FlashSelfRewriteGUIDlg.cpp	アプリケーションのダイアログ・クラスの処理ファイル
FlashSelfRewriteGUIDlg.h	アプリケーションのダイアログ・クラスの定義ファイル
CommandThread.cpp	書き換え通信処理スレッド・クラス処理ファイル
CommandThread.h	書き換え通信処理スレッド・クラス定義ファイル
CommonProc.cpp	共通処理クラス処理ファイル
CommonProc.h	共通処理クラス定義ファイル
SerialPort.cpp	シリアルCOMポート通信クラスの処理ファイル
SerialPort.h	シリアルCOMポート通信クラスの定義ファイル
Resource.h	リソース・ヘッダ・ファイル
UsbfUpdater.ini	アプリケーション動作設定ファイル

8.3.1 アプリケーション・クラス (FlashSelfRewriteGUI)

初回起動時に引数（オプション）をチェックし、ダイアログ・クラスを呼び出します。

次にアプリケーションの起動オプションを示します。

Table 8-2 アプリケーション起動オプション

オプション	説明
/S nnnnnn	書き込み開始アドレスを16進数で指定
/C nn	接続COMポート番号の指定
Filename	書き換え対象のファイル・パス

8.3.2 アプリケーション・ダイアログ・クラス (FlashSelfRewriteGUIDlg)

書き換え指定のダイアログ画面を表示します（「4. USB CDC経由内蔵FlashROM書き換えプログラムの実行」参照）。この画面で動作モード、書き換えアドレス、書き換えファイル、接続COMポートを指定します。また、画面表示の際に、アプリケーション動作設定ファイルを読み込み、先の指定がしてあればデフォルト値として画面に反映されます。

“Update” ボタンをクリックすると、書き換え通信処理スレッド・クラスが呼び出されます。

追加したメンバ変数を示します。

Table 8-3 アプリケーション・ダイアログ・クラスのメンバ変数の一覧

メンバ変数		説明
型	メンバ名	
Int	m_nCOM	接続するCOMポート番号
TCHAR	m_tcAppDir[_MAX_PATH]	アプリケーションの実行ディレクトリ
CString	m_strCurTargetSeries	現在のターゲット種別
CString	m_strCurTarget	現在のターゲット名
CString	m_strCurDevice	現在のデバイス
CStringArray	m_arDeviceSeries	デバイス種別のリスト
CStringArray	m_arDeviceVal	デバイスのリスト
CStringArray	m_arDeviceText	デバイス名のリスト
Int	m_nDevSize	現在のデバイスのROMサイズ
CWinThread*	m_pCommandThread	スレッド・クラスのポインタ
BOOL	m_bExistThread	スレッドの動作状態
BOOL	m_bStartUp	初回起動を表す
DWORD	m_dwROMStartAddress	ROM領域開始アドレス
DWORD	m_dwROMEndAddress	ROM領域終了アドレス
DWORD	m_dwEnROMStartAddress	ROM P/Eアクセス許可開始アドレス
DWORD	m_dwEnROMEndAddress	ROM P/Eアクセス許可終了アドレス

メンバ関数を次に示します。

Table 8-4 Read_DeviceInfo 関数

関数名	Read_DeviceInfo	
記述形式	bool Read_DeviceInfo (void)	
機能	アプリケーション動作設定ファイルから情報を取得	
入出力	入力	なし
	出力	TRUE(成功)/FALSE(失敗)

Table 8-5 Write_DeviceInfo 関数

関数名	Write_DeviceInfo
-----	------------------

記述形式		bool Write_DeviceInfo (void)
機能		アプリケーション動作設定ファイルを更新する
入出力	入力	なし
	出力	TRUE(成功)/FALSE(失敗)

Table 8-6 Update_Message 関数

関数名		Update_Message
記述形式		void Update_Message (LPCTSTR)
機能		メッセージ表示欄にメッセージを表示する
入出力	入力	メッセージ文字列のポインタ
	出力	なし

Table 8-7 Initialize_Device 関数

関数名		Initialize_Device
記述形式		void Initialize_Device(void)
機能		初期化处理
入出力	入力	なし
	出力	なし

Table 8-8 DeviceListRefresh 関数

関数名		DeviceListRefresh
記述形式		void DeviceListRefresh (void)
機能		Deviceリストの作成を行う
入出力	入力	なし
	出力	なし

Table 8-9 DeviceInfoRefresh 関数

関数名		DeviceInfoRefresh
記述形式		void DeviceInfoRefresh (void)
機能		Deviceのコンボボックス更新
入出力	入力	なし
	出力	なし

Table 8-10 AppStatus 関数

関数名		AppStatus
記述形式		void AppStatus(bool stu)
機能		書き換え動作時の状態を設定する

入出力	入力	stu : TRUE(画面操作を有効にする) FALSE(画面操作を無効にする)
	出力	なし

8.3.3 書き換え通信処理スレッド・クラス (CommandThread)

シリアルCOMポート通信クラスを使用して、ターゲットとなる評価ボードに接続し、指定のファイルをインターフェース仕様に沿って送受信を行います。ファイルがHEXファイルの場合は、その解析も行います。

追加したメンバ変数を示します (アプリケーション・ダイアログ・クラスと同じ部分は、省略しています)。

Table 8-11 書き換え通信処理スレッド・クラスのメンバ変数一覧

メンバ変数		説明
型	メンバ名	
CDialog*	m_pAppDlg	呼び出し元のダイアログ・クラスのポインタ
CString	m_strAppDir	アプリケーションのあるディレクトリ
BOOL*	m_pbExistThread	スレッドの動作状況のポインタ
CSerialPort	m_Serial	シリアルCOMポート通信クラス
int	m_nCOM	接続するCOMポート番号
CString	m_strFileName	対象とするファイル・パス
EnMode	m_enMode	書き換えモード
DWORD	m_dwStartAddress	書き換え開始アドレス
DWORD	m_dwROMStartAddress	ROMの先頭アドレス
DWORD	m_dwROMEndAddress	ROMの末尾アドレス
int	m_addrIgnoreCheck	オプション設定メモリ、固定ベクタアドレス用フラグ

追加したメンバ関数を次に示します。

Table 8-12 Cal_CheckSum 関数

関数名	Cal_CheckSum	
記述形式	BYTE Cal_CheckSum(LPBYTE bytes, LONG size)	
機能	チェック・サムを算出します	
入出力	入力	bytes : データ列のポインタ size : データ列の長さ
	出力	チェック・サム算出値

Table 8-13 Change_strHex2Binary 関数

関数名	Change_strHex2Binary	
記述形式	VOID Change_strHex2Binary(LPCSTR strHex, LPBYTE pbytes, LONG size)	
機能	16進数であらわされた文字列をバイナリのデータ列に変換します	
入出力	入力	strHex : 16進数で表された文字列のポインタ pbyte : データ列の先頭ポインタ size : 変換するデータの数
	出力	なし

Table 8-14 Upsets_DWORD 関数

関数名	Upsets_DWORD	
記述形式	DWORD Upsets_DWORD(DWORD dwVal)	
機能	DWORD型の値をバイトごとに反転する (ex.) 0xaabbccdd → 0xddccbbaa	
入出力	入力	dwVal : 反転するDWORDの値
	出力	反転された値

Table 8-15 SET_StartRecord 関数

関数名	SET_StartRecord	
記述形式	VOID SET_StartRecord (LPVOID lpRecord)	
機能	書き換え開始レコードを作製する	
入出力	入力	lpRecord : レコード格納ポインタ
	出力	なし

Table 8-16 SET_EndRecord 関数

関数名	SET_EndRecord	
記述形式	VOID SET_EndRecord (LPVOID lpRecord)	
機能	書き換え終了レコードを作製する	
入出力	入力	lpRecord : レコード格納ポインタ

	出力	なし
--	----	----

8.3.4 共通処理クラス (CommonProc)

共通で使用される処理を定義しています。追加したメンバ関数を次に示します。

Table 8-17 GetAppDir 関数

関数名		GetAppDir
記述形式		static VOID GetAppDir(LPTSTR path, int sw = 0)
機能		アプリケーションの実行アドレスを取得します。
入出力	入力	path : 取得する文字列のポインタ sw : 0 パスをそのまま取得 1 ショート・パスに変更したパスを取得
	出力	なし

Table 8-18 Change_Hex2Val 関数

関数名		Change_Hex2Val
記述形式		static DWORD Change_Hex2Val(LPCSTR pHex)
機能		1バイト (16進数2桁) で表された文字列を数値に変換する
入出力	入力	pHex : 16進数2桁で表された文字列のポインタ
	出力	変換された値

Table 8-19 IsNumeric 関数

関数名		IsNumeric
記述形式		static BOOL IsNumeric(LPCTSTR lpNum, LONG size, int type)
機能		数値チェック処理
入出力	入力	lpNum : 数値を表した文字列のポインタ size : チェックする数値の桁数 type : 10 10進数としてチェックする 16 16進数としてチェックする
	出力	TRUE(数値であることを表す)/FALSE(数値ではないことを表す)

Table 8-20 IsExistFile 関数

関数名		IsExistFile
記述形式		static BOOL IsExistFile(LPCTSTR lpszFileName, BOOL bDirectory = FALSE)
機能		ファイルの存在チェック
入出力	入力	lpszFileName : 確認するファイル・パス bDirectory : FALSE(ファイルをチェックする) TRUE(ディレクトリをチェックする)
	出力	TRUE(存在する)/FALSE(存在しない)

8.3.5 シリアルCOMポート通信クラス (SerialPort)

このクラスを使用して、COMポートでのシリアル通信を行います。

追加したメンバ変数を次に示します。

Table 8-21 シリアルCOMポート通信クラスのメンバ変数一覧

メンバ変数		説明
型	メンバ名	
HANDLE	m_hCom	接続時に取得するハンドル
DCB	m_Dcb	デバイス制御ブロック構造体
COMMTIMEOUTS	m_TimeoutSts	タイムアウト設定用の構造体
INT	m_nCOM	接続するポート番号

メンバ関数を次に示します。

Table 8-22 Port_Open 関数

関数名	Port_Open	
記述形式	LONG Port_Open(INT com)	
機能	指定のCOMポートに接続します	
入出力	入力	com : COMポート番号
	出力	0 接続成功 -1 接続失敗

Table 8-23 Port_Close 関数

関数名	Port_Close	
記述形式	VOID Port_Close(VOID)	
機能	接続中のポートを切断します。	
入出力	入力	なし
	出力	なし

Table 8-24 Port_Write 関数

関数名	Port_Write	
記述形式	LONG Port_Write(LPCVOID buf, LONG cnt)	
機能	シリアル通信によるデータの送信を行う。	
入出力	入力	buf : 送信データの列のポインタ cnt : 送信データの長さ (バイト)
	出力	送信したバイト数。-1で送信の失敗。

Table 8-25 Port_Read 関数

関数名	Port_Read	
記述形式	LONG Port_Read(LPVOID buf, LONG cnt)	
機能	シリアル通信によるデータの受信を行う。	
入出力	入力	buf : 受信データを格納するデータ列のポインタ cnt : 受信するデータの長さ (バイト)
	出力	受信したバイト数。-1で受信の失敗を表す。

Table 8-26 Get_PortNumber 関数

関数名	Get_PortNumber	
記述形式	INT Get_PortNumber(VOID)	
機能	接続中のポート番号を取得	
入出力	入力	なし
	出力	接続中のポート番号

Table 8-27 AutoScanCom 関数

関数名	AutoScanCom	
記述形式	INT AutoScanCom (LPCTSTR pszService, LPCTSTR pszInterface, INT nNo = 0)	
機能	接続可能なCOMポートを検出	
入出力	入力	pszService : COMポートが動作しているサービス名 pszInterface : インターフェース名 nNo : この番号以降を検索する
	出力	検出したCOMポート番号。見つからなかったら0が返る。

8.3.6 アプリケーション動作設定ファイル (UsbfUpdater.ini)

アプリケーション動作設定ファイルはiniファイル形式で、設定値の保持、またはデバイスの情報を保持します。このファイルはexeファイルと同一のフォルダに置いて下さい。iniファイルがない場合、アプリケーションが正常に起動しません。

次にiniファイル内の定義を示します。

Table 8-28 アプリケーション動作設定ファイル説明 (セクション)

セクション	説明
Application	アプリケーションで設定中の値を表します。 アプリケーションが書き込む情報です。
SS_xxx	デバイスの前回表示情報を保持します。 アプリケーションが書き込む情報です。
Device. XXXXXXXXX	デバイスの情報を表します。(複数設定が可能) ユーザが追加可能な情報です。

Table 8-29 アプリケーション動作設定ファイル内項目一覧

セクション	キー	値	説明
Application	Series	XXX	指定中のターゲットのシリーズ種別
	COM	1~20	接続中または接続するCOMポート番号 注)Windows 10以降は設定できませんが使用できません
	EnableStartAddress	FFFFFFFF	書き込み許可開始アドレス
	EnableEndAddress	FFFFFFFF	書き込み許可終了アドレス
SS_XXX	Device	XXX	ターゲットで指定しているデバイス
Device. XXX	TargetSeries	XXX	このデバイスが属するシリーズ種別
	Name	XXX	このデバイスの名前
	Size	1~999	このデバイスのROMサイズ(Kbyte)
	StartAddress	FFFFFFFF	このデバイスのROM開始アドレス

デバイスの情報以外の項目は表示情報の保持となるため GUI ソフト終了時に自動で更新されます。

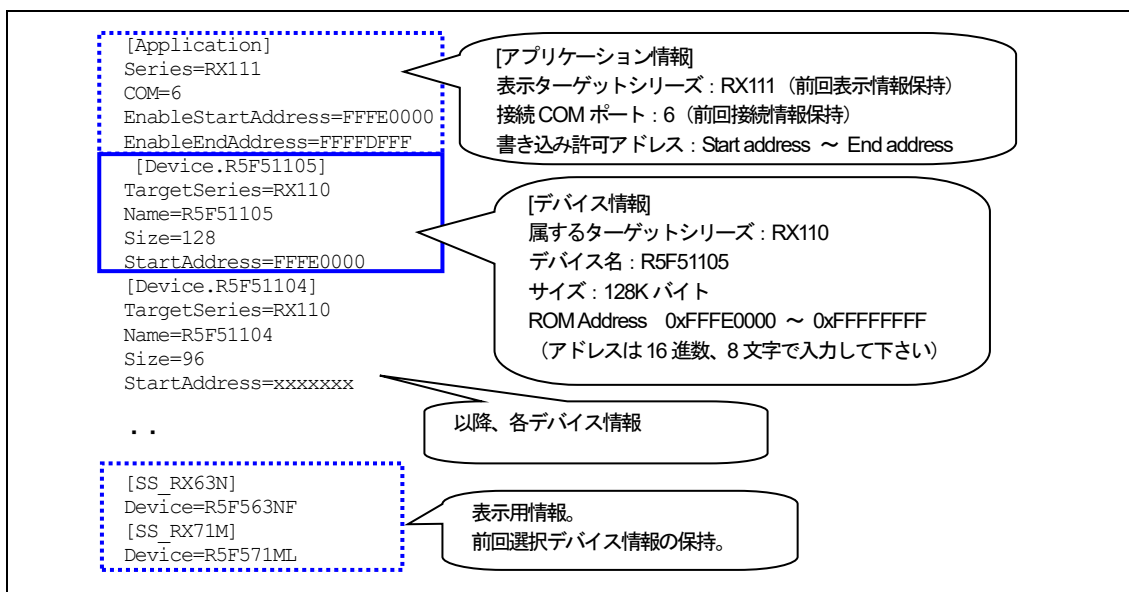


Figure 8-2 アプリケーション動作設定 ini ファイル

8.4 メッセージ表示

メッセージ表示欄に表示されるメッセージとその表示タイミングの一覧を次に示します。

Table 8-30 表示メッセージ一覧

メッセージ	表示タイミング
Start upload file.	書き換え処理開始時
Now erasing	Flash ROM消去中
Now writing	Flash ROM書き込み中
Now copying	Flash ROMコピー中(バックアップ機能使用時のみ)
Success.	書き換え処理正常終了時
Please input file.	書き換え処理時に指定のファイルが指定されていない。 または、指定ファイルが存在しない
Please set the address correctly.	アドレスが正常に指定されていない。
Please set COM port.	COMポートが正しく指定されていない
ERR: file open error.	ファイルのオープンに失敗した
ERR: file format error.	モトローラS形式、インテルHEX形式以外のファイルを指定した
ERR: Unable to connect to the COM port n.	COMポートnの接続に失敗した
ERR: Flash ROM initialization error	FlashROM初期化エラー
ERR: Data transmisson error.	データの送信に失敗した
ERR: Data reception error.	データの受信に失敗した (リトライ3回も同様)
ERR: Vefify error	ベリファイエラーが発生した。
ERR: Copying of Flash ROM rewrite program failed.	FlashROM書き換えプログラムのコピーに失敗した。(Dualモードのみ)
ERR: Unused area writing error	未使用領域書き込みエラー (バックアップ機能使用時のみ)
ERR: Option-Setting Memory writing error	オプション設定メモリ書き込みエラーが発生した。
ERR: Writing process stop.	ボード側から応答レコードでNAK (エラー) を受信した
ERR: Write Enable Area Address is ROM area over, or illegal value.	P/E Access Enable Area指定がROM領域を超えているか、異常値 (Use P/E Access Enable にチェックがある場合のみ)
ERR: Address is ROM area over. Process stop.	書き込みアドレスがROM領域を超えている
ERR: file size error.	ファイル・サイズ・チェックの際にデータのサイズがROM領域を超える場合
ERR: Security code of Updater and User program do not match.	Flash ROM書き換えプログラムとユーザプログラムのセキュリティコードが不一致
ERR: Get ROM Address Error. <Device: xxxx >	iniファイルのROM情報が間違っている場合
ERR: Get ROM Address Error. Update process stop.	iniファイルで読み込んだROM情報が正常ではないときに、書き込みを行おうとした場合

9. データ通信仕様

9.1 書き換え通信インターフェース仕様

ファイル転送アプリケーションが動作するPCと評価ボード間で通信する内容を示します。

9.1.1 通信データの構成

PCは最初に開始レコード，最後に終了レコードを送信します。フラッシュ・メモリに書き込むデータは，データ・レコードの形式で送信します。

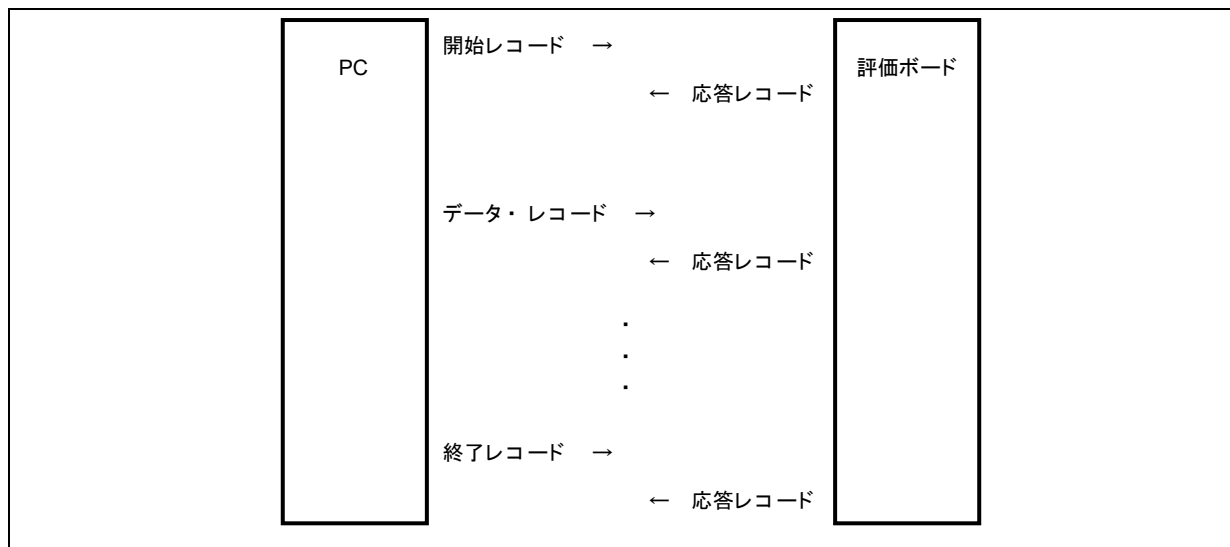


Figure 9-1 通信データ・シーケンス

9.1.2 PC 側送信データ

PC側は、開始レコード、データ・レコード、終了レコードを送信します。

各レコードは1レコードずつ送信し、応答レコードを受信するまで、次のレコードの送信は行いません。

(1). 開始レコード

書き換えの実行時に、最初に送信するレコードです。：14バイト

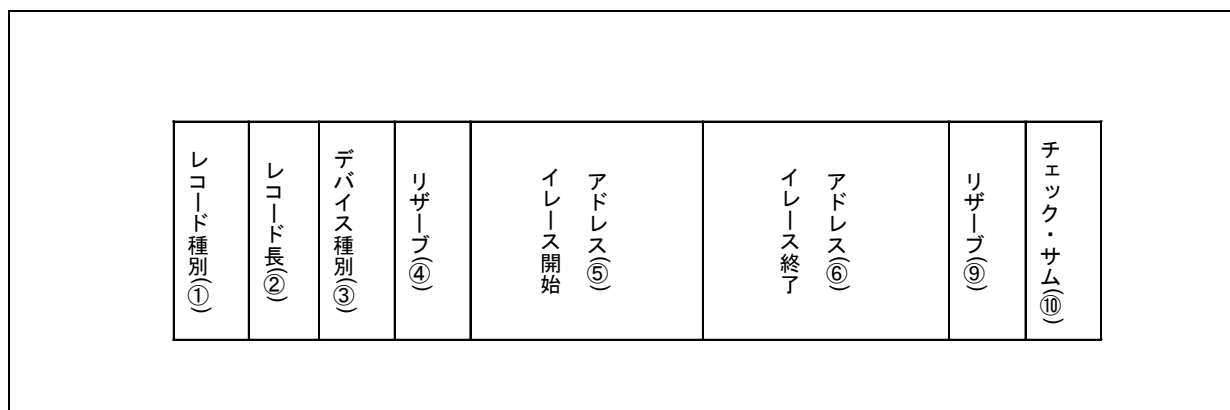


Figure 9-2 開始レコードの形式

- ① レコード種別：1バイト
レコードの種類
開始レコードのレコード種別は、0x00
- ② レコード長：1バイト
デバイス種別以降のバイト数
- ③ デバイス種別：1バイト
デバイスの種類（現在未使用のため0x00固定）
- ④ リザーブ：1バイト
0x00 固定
- ⑤ イレース開始アドレス：4バイト
ROMのイレース開始アドレス指定。アドレスは、32ビット数値でリトル・エンディアン形式
- ⑥ イレース終了アドレス：4バイト
ROMの終了アドレス指定。アドレスは、32ビット数値でリトル・エンディアン形式
- ⑦ リザーブ：1バイト
0x00 固定
- ⑧ チェック・サム：1バイト
レコードのチェック・サム
レコード長とデバイス種別と日付と時刻のチェック・サム
各バイトの値を加算した合計値の1の補数の下位8ビット

(2). データ・レコード

書き込むデータのレコードです。：(7+データ数)バイト (MAX 64バイト)

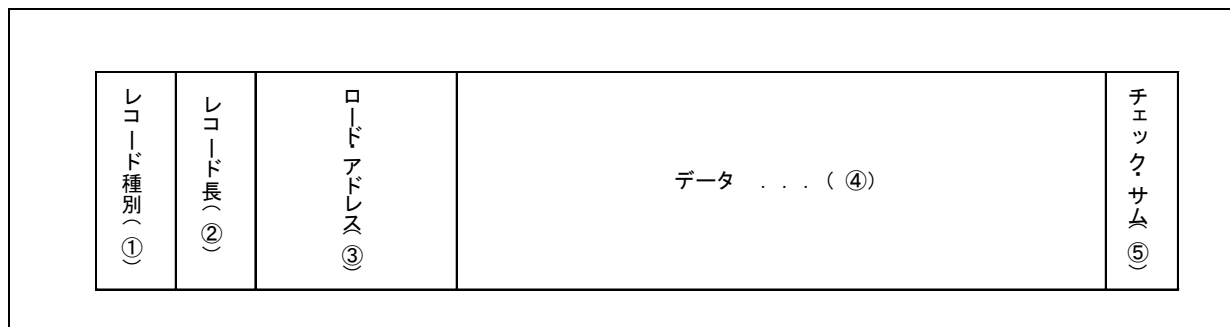


Figure 9-3 データ・レコードの形式

- ① レコード種別：1バイト
レコードの種類
データ・レコードのレコード種別は、0x0f
- ② レコード長：1バイト
ロード・アドレス以降のバイト数
- ③ ロード・アドレス：4バイト
フラッシュ・メモリのアドレス
このアドレスからデータが書き込まれる
ロード・アドレスは、32ビット数値でリトル・エンディアンの形式
- ④ データ：1～57バイト
フラッシュ・メモリに書き込むデータ
1レコードあたり最大で57バイト
- ⑤ チェック・サム：1バイト
レコードのチェック・サム
レコード長とロード・アドレスとデータのチェック・サム
各バイトの値を加算した合計値の1の補数の下位8ビット

(3). 終了レコード

すべてのデータを送信後、最後に送信するレコードです。 : 4バイト

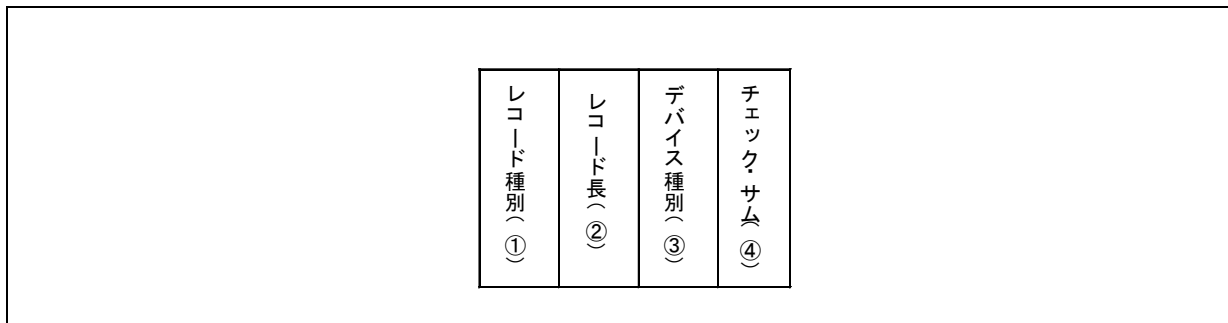


Figure 9-4 終了レコードの形式

- ① レコード種別 : 1 バイト
レコードの種類
終了レコードのレコード種別は、0xf0
- ② レコード長 : 1 バイト
デバイス種別以降のバイト数
- ③ デバイス種別 : 1 バイト
デバイスの種類 (現在未使用のため0x00固定)
- ④ チェック・サム : 1 バイト
レコードのチェック・サム
レコード長とデバイス種別のチェック・サム
各バイトの値を加算した合計値の1の補数の下位8ビット

9.1.3 評価ボード側送信データ

評価ボードは、PCからのレコードに対して、応答レコードを送信します。：5バイト

(1). 応答レコード

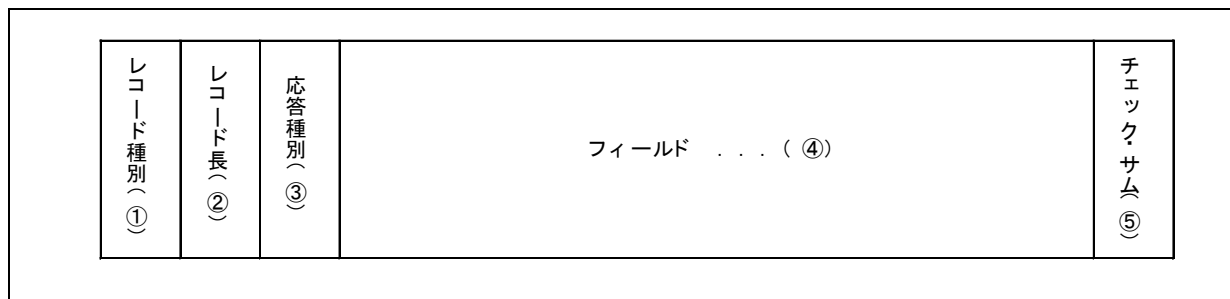


Figure 9-5 応答レコードの形式

- ① レコード種別：1バイト
レコードの種類
応答を返す対象のレコードのレコード種別
応答レコードのレコード種別は0xFF
- ② レコード長：1バイト
応答種別以降のバイト数
- ③ 応答種別：1バイト
応答種別
以下の3種類
- 0x00 : ACK
 - 0x0f : NAK (再送要求)
 - 0xf0 : NAK (エラー終了)
- ④ フィールド：1バイト
- a. 開始レコードの場合、バックアップ機能の有効/無効を示すコードを返します。
- | | | |
|------------|---|------|
| バックアップ機能無効 | : | 0xB0 |
| バックアップ機能有効 | : | 0xB1 |
- b. データ・レコード/終了レコードの場合、以下のステータスコードまたはエラーコードを返します。
- a) ステータスコード
- | | | |
|----------------|---|------|
| Flash ROMイレース中 | : | 0x01 |
| Flash ROM書き込み中 | : | 0x03 |
- b) エラー・コード
- | | | |
|---|---|------|
| フラッシュ初期化エラー | : | 0xE1 |
| セキュリティコード不一致エラー | : | 0xE2 |
| ROMイレースエラー | : | 0xE3 |
| パラメータエラー | : | 0xE4 |
| ベリファイエラー | : | 0xE5 |
| オプション設定メモリ書き換えエラー | : | 0xE6 |
| 更新対象領域へのFlash ROM書き換えプログラムコピー失敗
(デュアルモード使用時のみ) | : | 0xE7 |
| 未使用領域書き込みエラー
(バックアップ機能使用時のみ) | : | 0xE8 |

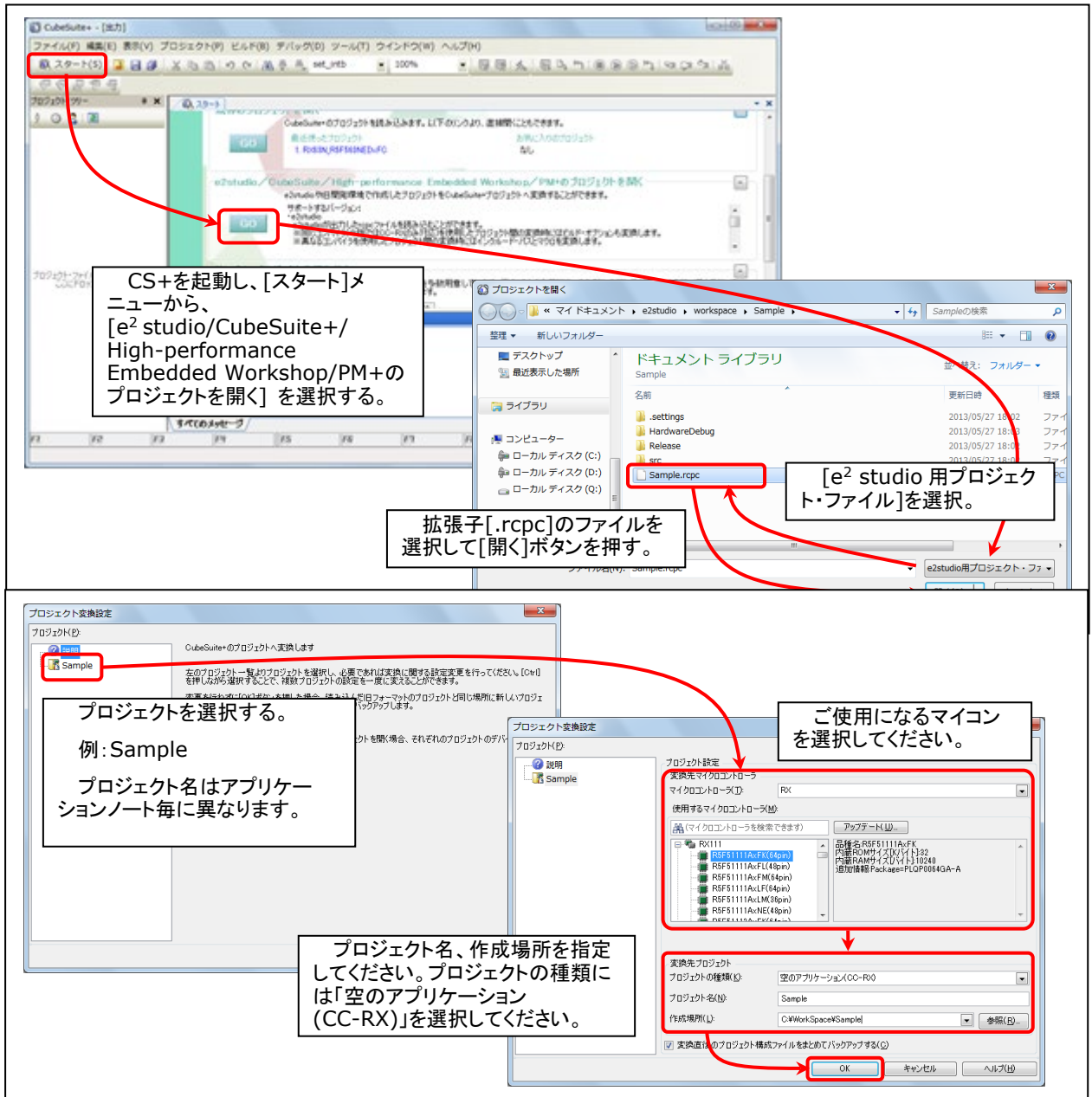
- ⑤ チェック・サム：1バイト
 - レコードのチェック・サム
 - レコード長と応答種別とフィールドのチェック・サム
 - 各バイトの値を加算した合計値の1の補数の下位8ビット

10. e² studio 用プロジェクトを CS+で使用する場合

USB CDC経由内蔵FlashROM書き換えプログラムのプロジェクトは、統合環境 e² studio で作成されています。USB CDC経由内蔵FlashROM書き換えプログラムを CS+で動作させる場合は、下記の手順でインポート処理を行ってください。

Note:

1. srcフォルダとrpcファイルを格納するフォルダの名前は、"MCU名_FirmwareUpdater"にしてください。
例えば、RX63Nの場合、フォルダ名は"RX63N_FirmwareUpdater"になります。
2. 「プロジェクト変換設定」ウィンドウ内の「変換直前のプロジェクト構成ファイルをまとめてバックアップする」のチェックを外してください。



ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<http://japan.renesas.com/>

お問い合わせ先

<http://japan.renesas.com/contact/>

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。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jun 30, 2016	—	初版発行
1.01	Jun 30, 2017	—	対象デバイスに RX65N/RX651 を追加
1.02	Sep 30, 2017	—	<ol style="list-style-type: none"> 1. RX65N/RX651-2M をサポート 2. デュアルモードをサポート 3. Write & Verify をサポート
1.03	Feb 16, 218	—	バックアップ機能をサポート
1.04	Apr 16, 2019	—	対象デバイスに RX66T/RX72T を追加
1.05	Mar 1, 2020	—	対象デバイスに RX72M/RX72N/RX66N/RX23W を追加
1.06	May 1, 2021	—	対象デバイスに RX671 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しないでください。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っていると、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、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

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。

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/