Renesas USB MCU

USB ペリフェラル・ファームウエア・アップデータ

R01AN1680JJ0110
Rev.1.10
Sep 30, 2016

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

RL78/G1C, RL78/L1C

目次

1. 資料概要

本書は、USB ペリフェラル･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. RL78 Family Flash Self Programming Library Type01 application note
3. 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

1. 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します。
2. 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 ペリフェラル･ファームウエア･アップデータとユーザプログラムに対する設定**」、「7.3　**注意事項**」を参照してください。

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
H/W	:	Renesas USB device
MCU	:	Micro control Unit
P/E	:	Program / Erase
RSK	:	Renesas Starter Kit
USB	:	Universal Serial Bus

2. USB ペリフェラル・ファームウエア・アップデータ概要

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 ペリフェラル・ファームウエア・アップデータ
   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 ペリフェラル･ファームウエア･アップデータで書き込みを行います。

Program1　：　RSK ボード上の LCD に文字列が表示されます。
Program2　：　RSK ボード上の LCD に文字列が表示されます。

次に，プログラムのデータの流れを表します。

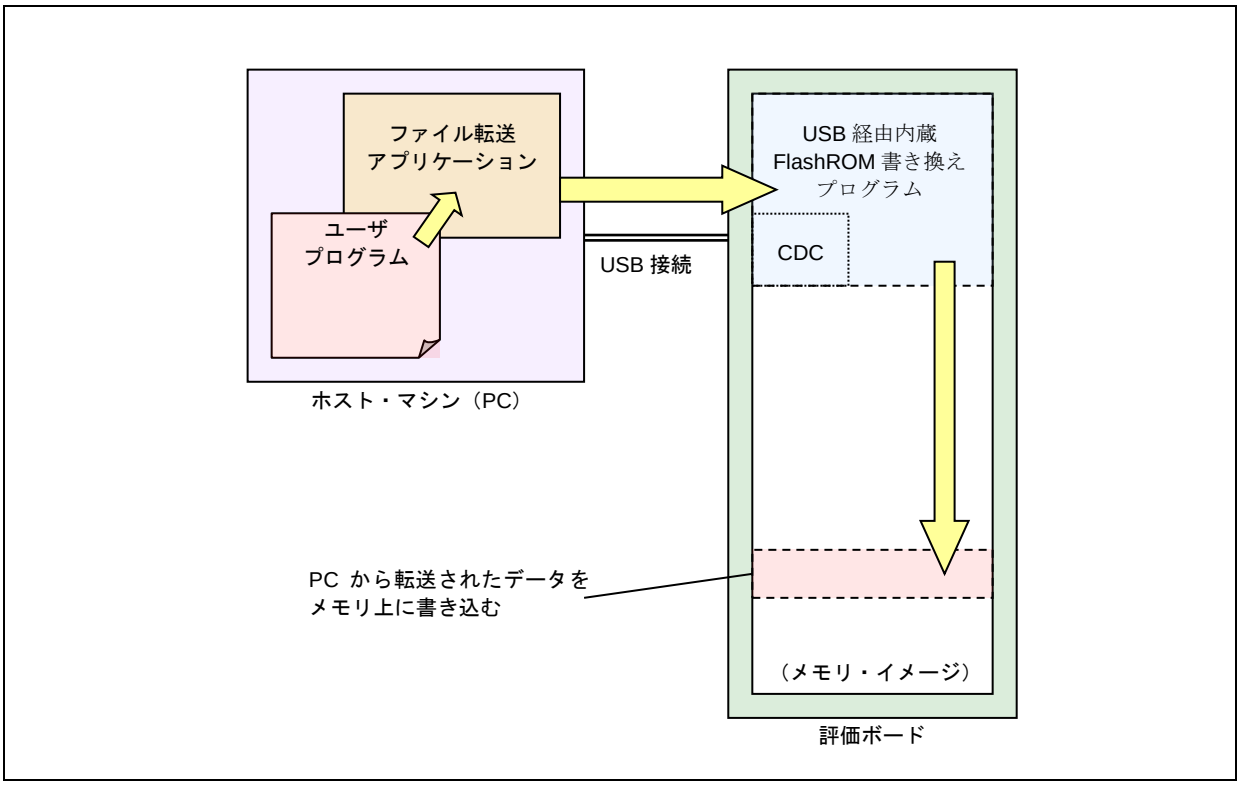

Figure 2-1 USB F/W Update のデータの流れ

特定条件下で評価ボードを起動することによりUSBペリフェラル・ファームウエア・アップデータが動作し、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 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。(Table 2-1参照)　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を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 フォーマットとインテル拡張フォーマットです。（.mot/.HEX ファイル）
4. ROM 上のアドレスを指定して，任意の領域に書き込み可能です。
5. 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/RAM サイズ

本プログラムが使用する ROM/RAM をTable 2-1に示します。

Table 2-1 USB ペリフェラル・ファームウエア・アップデータ**の ROM/RAM サイズ**

コンパイラ	ROM/RAMサイズ（バイト）	
	ROM	RAM
CC-RL	6.2K	0.5K
CA78K0R	7.3K	0.7K

※1： 最適化オプションはDefaultオプションが使用されています。

2.3　動作確認済環境

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	Renesas Starter Kit
b.	MCU	RL78/G1C, RL78/L1C
c.	エミュレータ	E1 エミュレータ
d.	USB ケーブル	評価ボードと PC 間で USB 通信
e.	PC	Windows® 7/ Window® 8.1/ Window® 10 搭載 PC

2. ソフトウェア環境

a.	コンパイラ	CA78K0R コンパイラ V.1.71 (CS+) CC-RL コンパイラ V.1.01 (e^2 studio)
b.	Flash 書き込みツール	Renesas Flash Programmer
c.	サンプル・プログラム一式	USB ペリフェラル・ファームウエア・アップデータ 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

2.4　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)
 │　　＋―FirmupdateGUI
 │　　│　　│　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)
 ＋―workspace (USBペリフェラル・ファームウエア・アップデータ プロジェクト一式)
　　＋―(CA78K0R)
　　＋―(CCRL)
```

次に，各フォルダの説明を示します。

(1). reference¥cdc_inf

Windows ®用のCDCドライバ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¥FirmupdaterGUI

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¥FirmupdaterGUI¥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（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。

このサンプルプログラムはLCDに文字列を表示します。

(5). workspace

各MCUのUSBペリフェラル・ファームウエア・アップデータが格納されているフォルダです。「7　**USBペリフェラル・ファームウエア・アップデータの解説**」を参照してください。

3. USB ペリフェラル・ファームウエア・アップデータのセットアップ

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 (e² studio)

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。

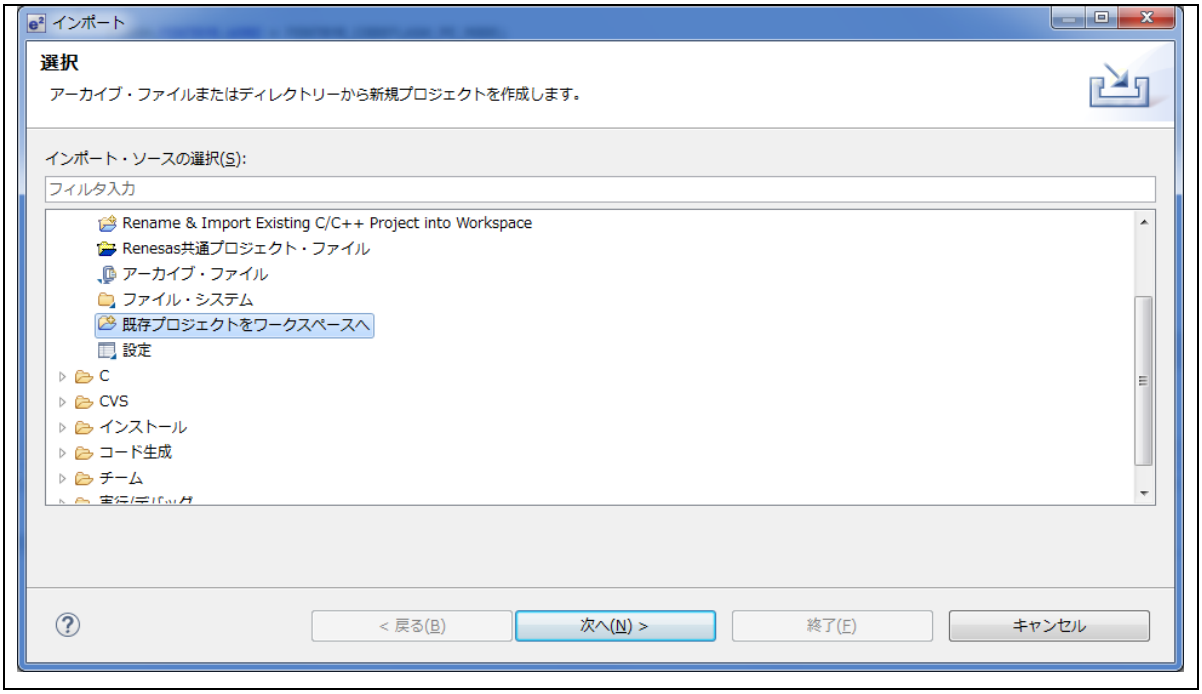

Figure 3-1　インポートの選択

(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。

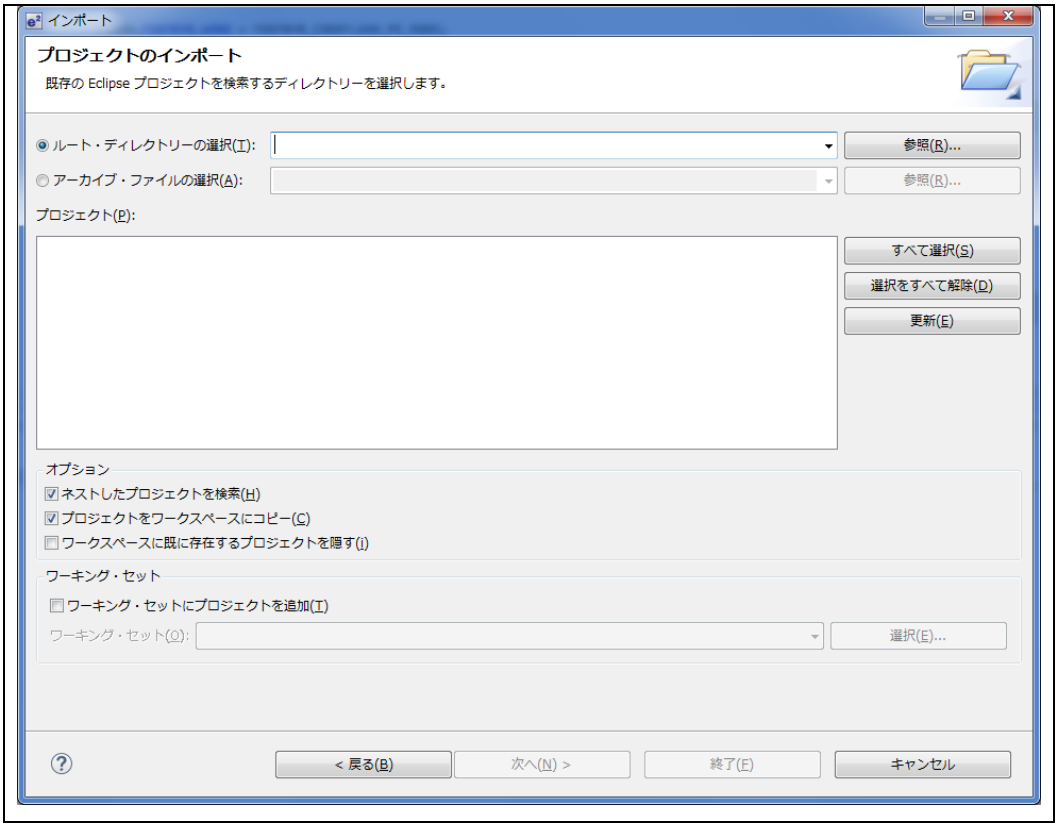

Figure 3-2　プロジェクトのインポート画面

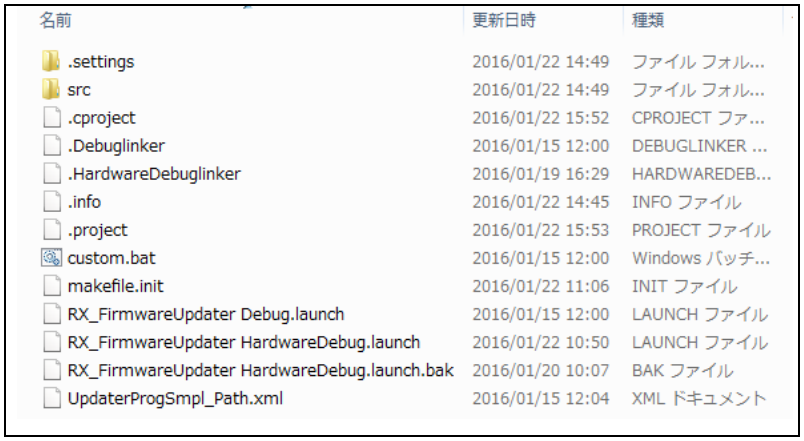

Figure 3-3　.project ファイルを含むフォルダ例

(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。

3.2 プロジェクトのセットアップ (CS+)

mtpj ファイルをダブルクリックしてください。

4.　USB ペリフェラル・ファームウエア・アップデータの実行

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(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。

以下に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。

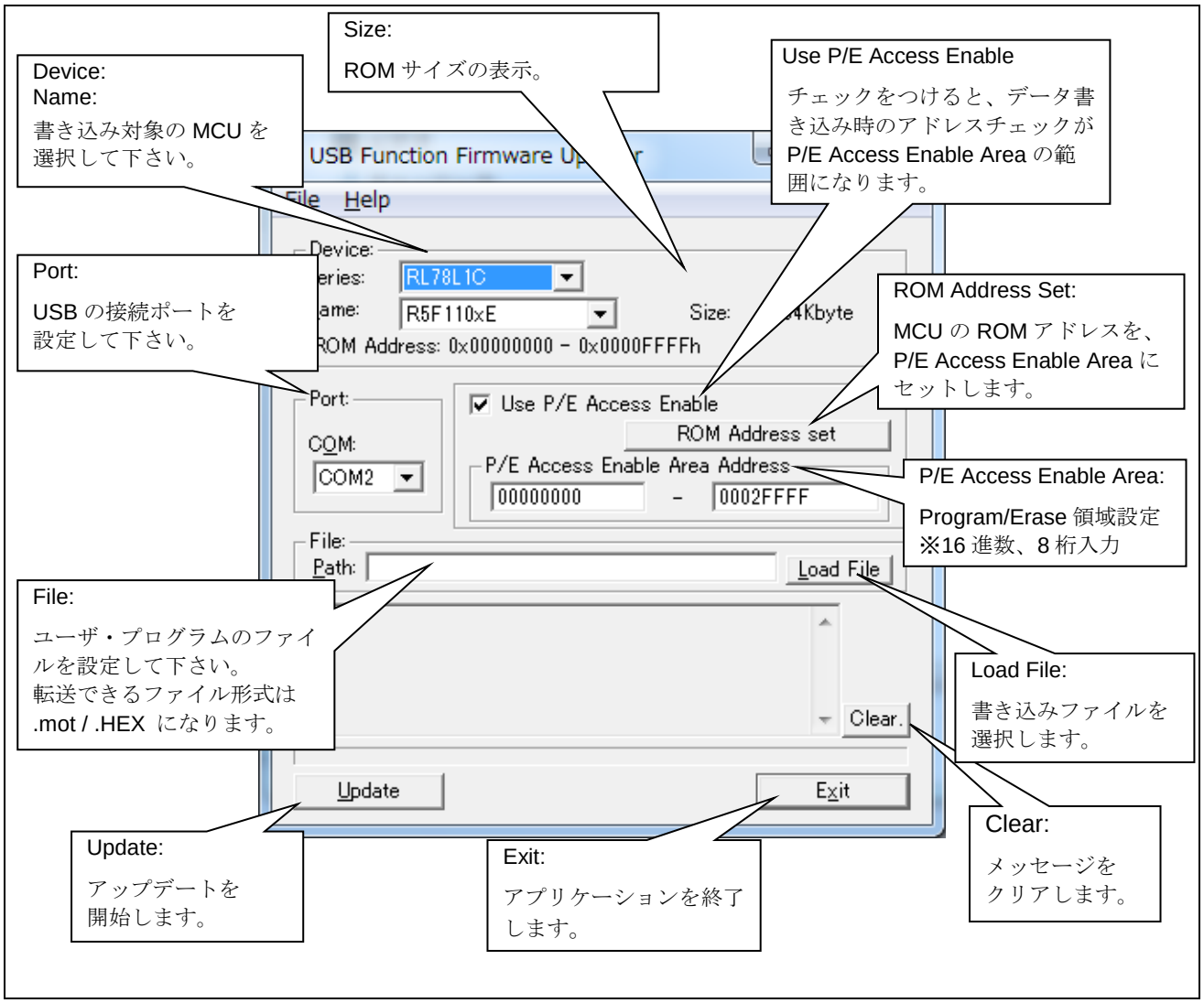

Figure 4-1　USB Firmware Updater GUI ソフト

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、ファームウェアアップデータ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。

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。(開始アドレスには、必ず 0x00000000 を指定してください。)

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

MCU	P/E address Setting		
RL78/G1C	0x00000000	-	任意
RL78/L1C	0x00000000	-	任意

[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。

4.2 USB ペリフェラル・ファームウエア・アップデータの 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 ペリフェラル・ファームウエア・アップデータの ROM 書き込み

(1) ハードウェアのセットアップ

USB ペリフェラル・ファームウエア・アップデータをご使用の MCU に書き込む場合の接続図を以下に示します。

a. E1/E20 エミュレータを使用する場合

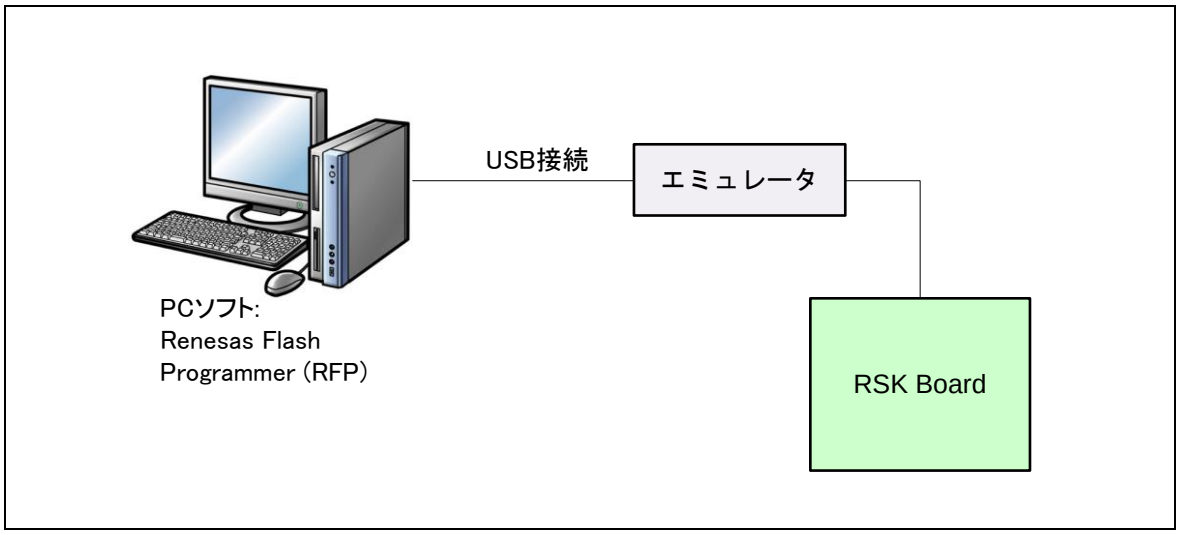

Figure 4-2 エミュレータを使用する場合の接続図

b. E1/E20 エミュレータを使用しない場合

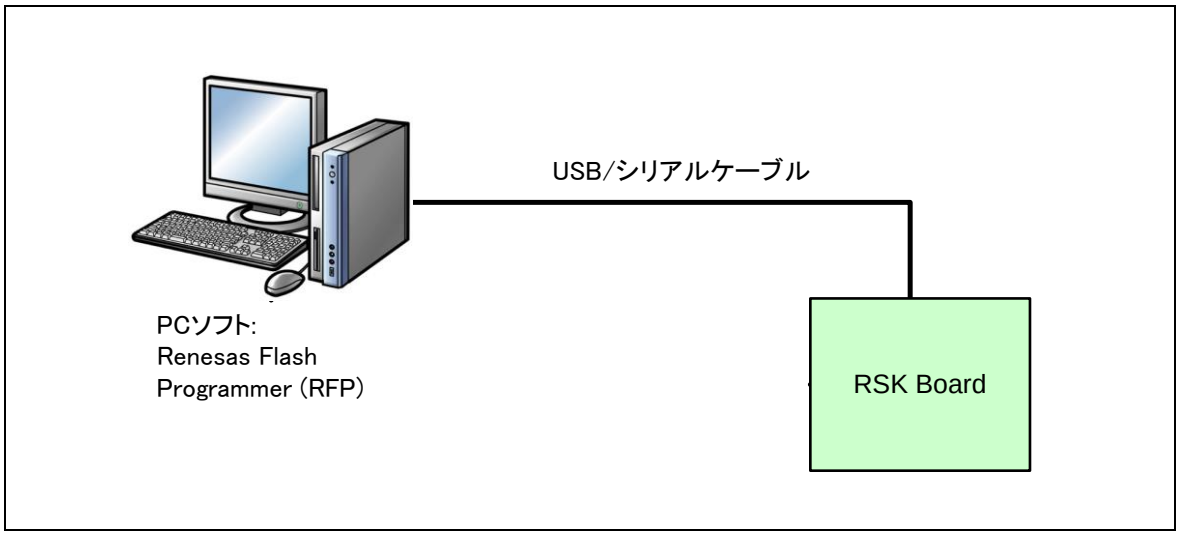

Figure 4-3 エミュレータを使用しない場合の接続図

(2) USB ペリフェラル・ファームウエア・アップデータの書き込み

Renesas Flash Programmer(RFP)を起動し，「ユーザ/データエリア」の参照ボタンから Workspace¥(MCU 名)フォルダにあるUSB ペリフェラル・ファームウエア・アップデータ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。出力パネルに PASS と表示され、パネル上部に緑地で「正常終了」と表示されれば、書き込みは完了です。

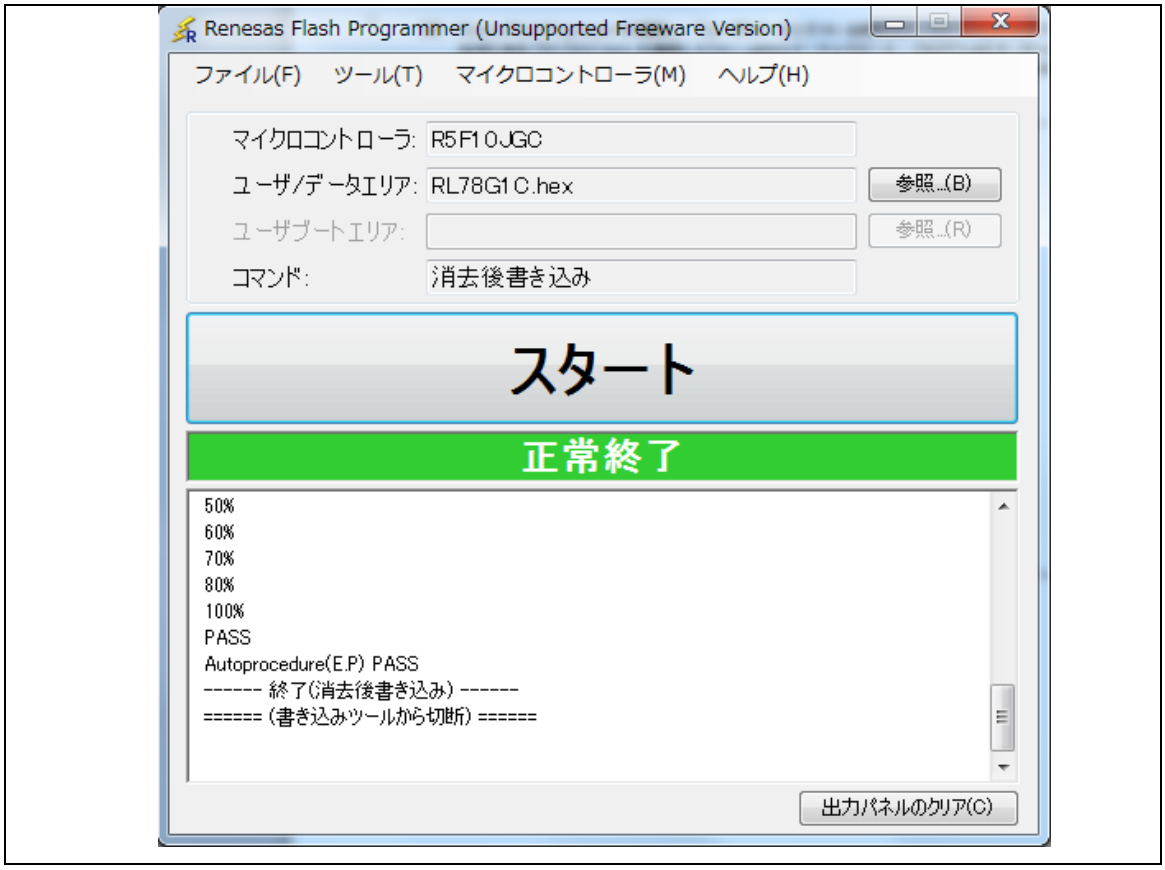

Figure 4-4　ファイルの指定

[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ペリフェラル・ファームウエア・アップデータの配置については「4.2.2 **USBペリフェラル・ファームウエア・アップデータの配置**」を参照してください。

4.2.2　USB ペリフェラル・ファームウエア・アップデータの配置

本プログラムの配置アドレスについて説明します。

USBペリフェラル・ファームウエア・アップデータは、0x2000番地以降の任意の領域に配置することができます。

下記はRL78/G1C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。

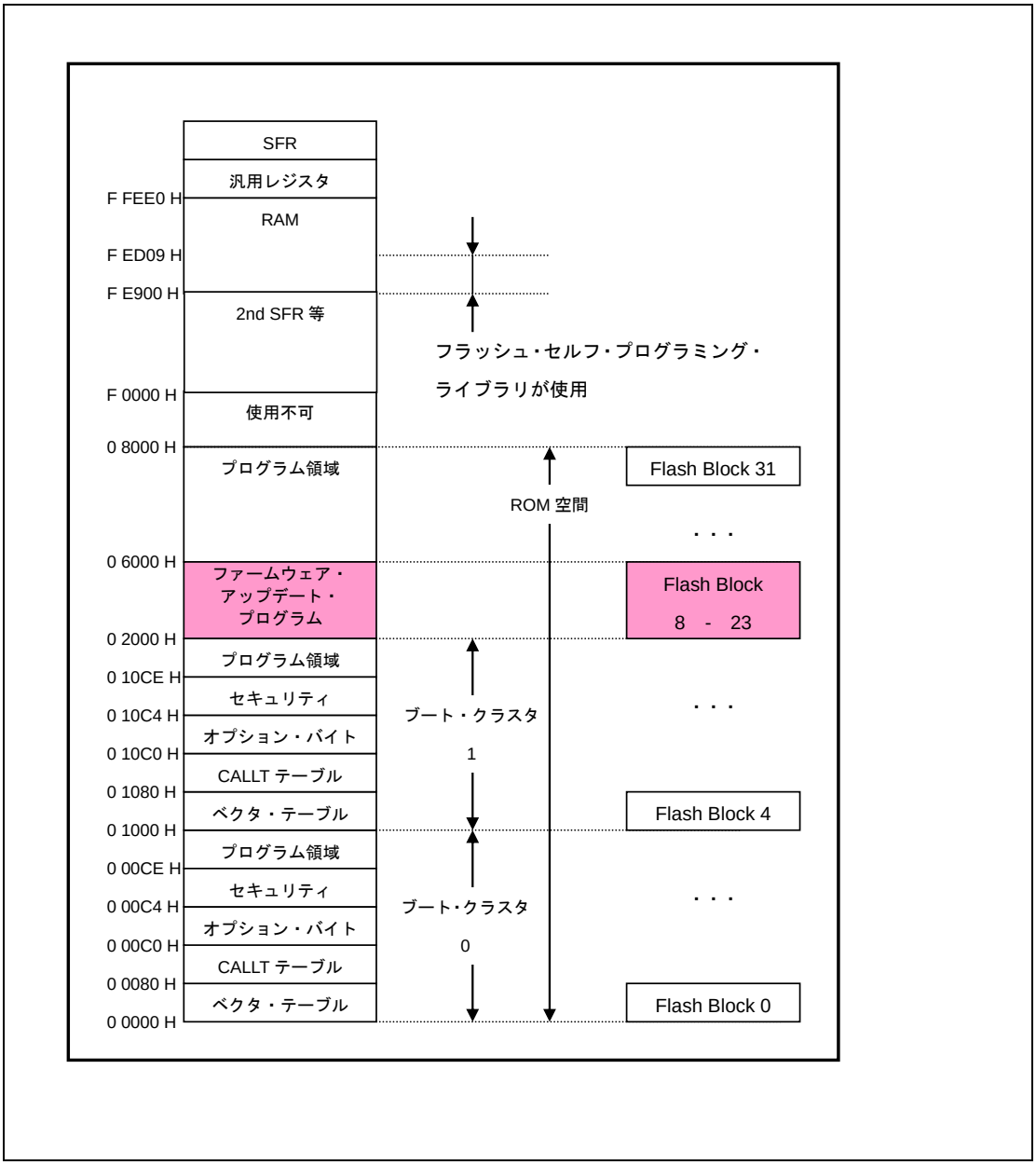

Figure 4-5　メモリ・マップ

4.3 USB ペリフェラル・ファームウエア・アップデータの実行(ユーザプログラム書き込み)

本章では、USB ペリフェラル・ファームウエア・アップデータを実行し、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 4-6に接続図を示します。

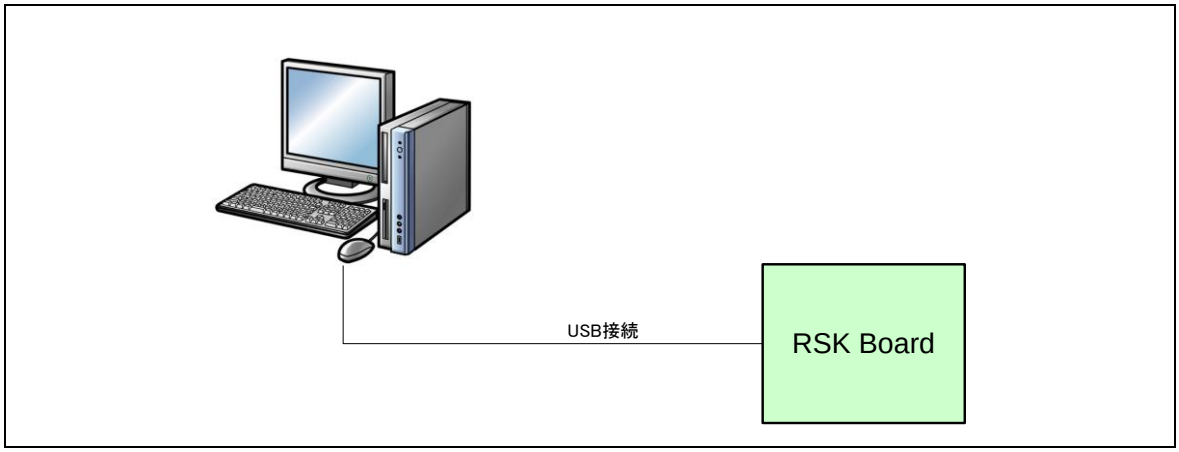

Figure 4-6 PC と評価ボードの接続図

(2). USB ペリフェラル・ファームウエア・アップデータの起動

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(USB Function Firmware Updater :PC 側ソフト)を起動してください。(Figure 4-8参照）。

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。

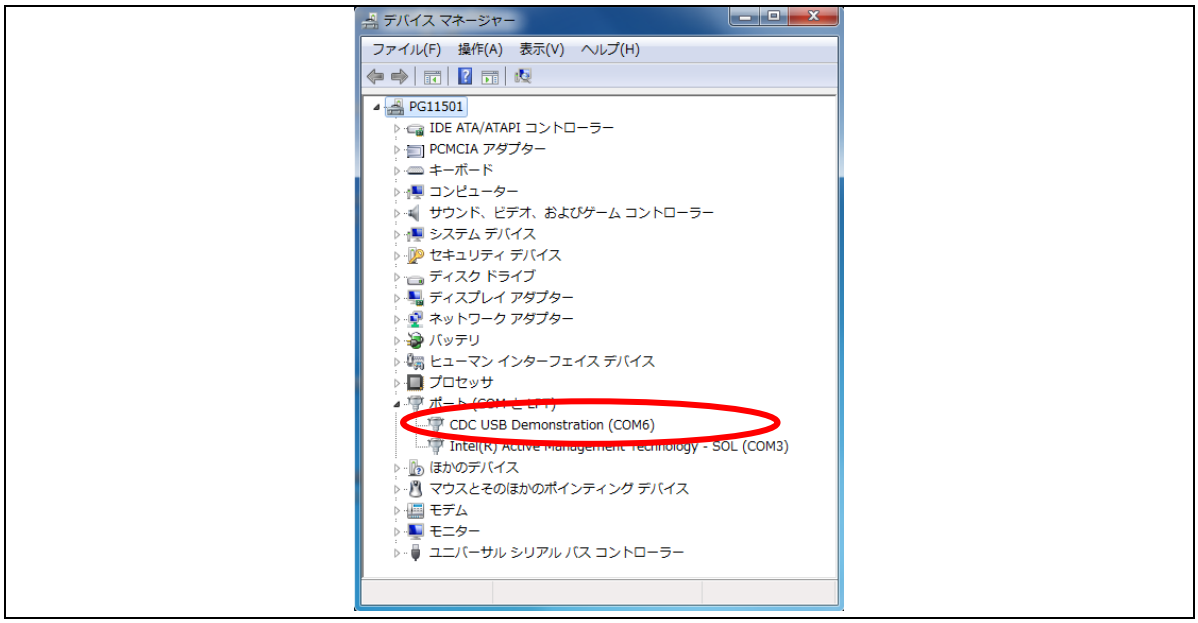

Figure 4-7　デバイス・マネージャのポート確認

(4). 転送ファイル選択

ファイル転送アプリケーション（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。

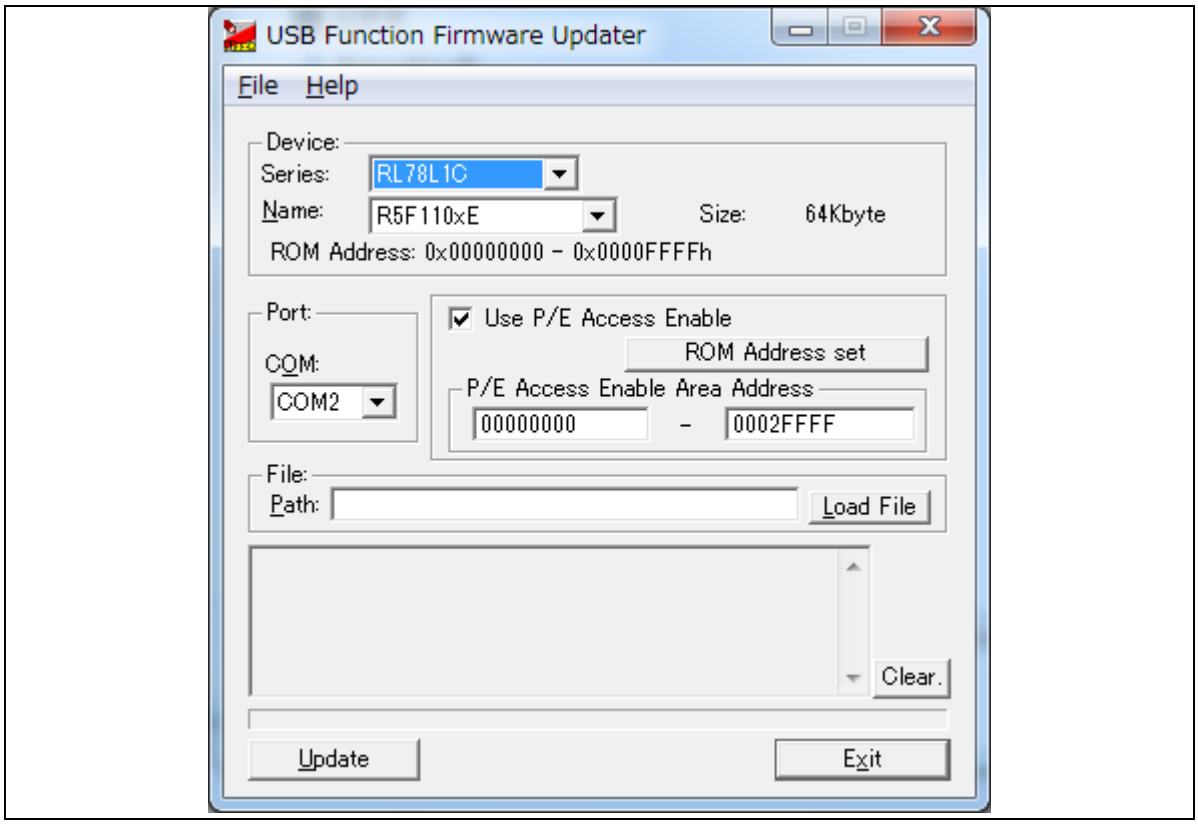

Figure 4-8　USB Firmware Updater GUI ソフト(ファイル転送アプリケーション)

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(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。

手順：

Use P/E Access Enable をチェックし、領域を設定して下さい。

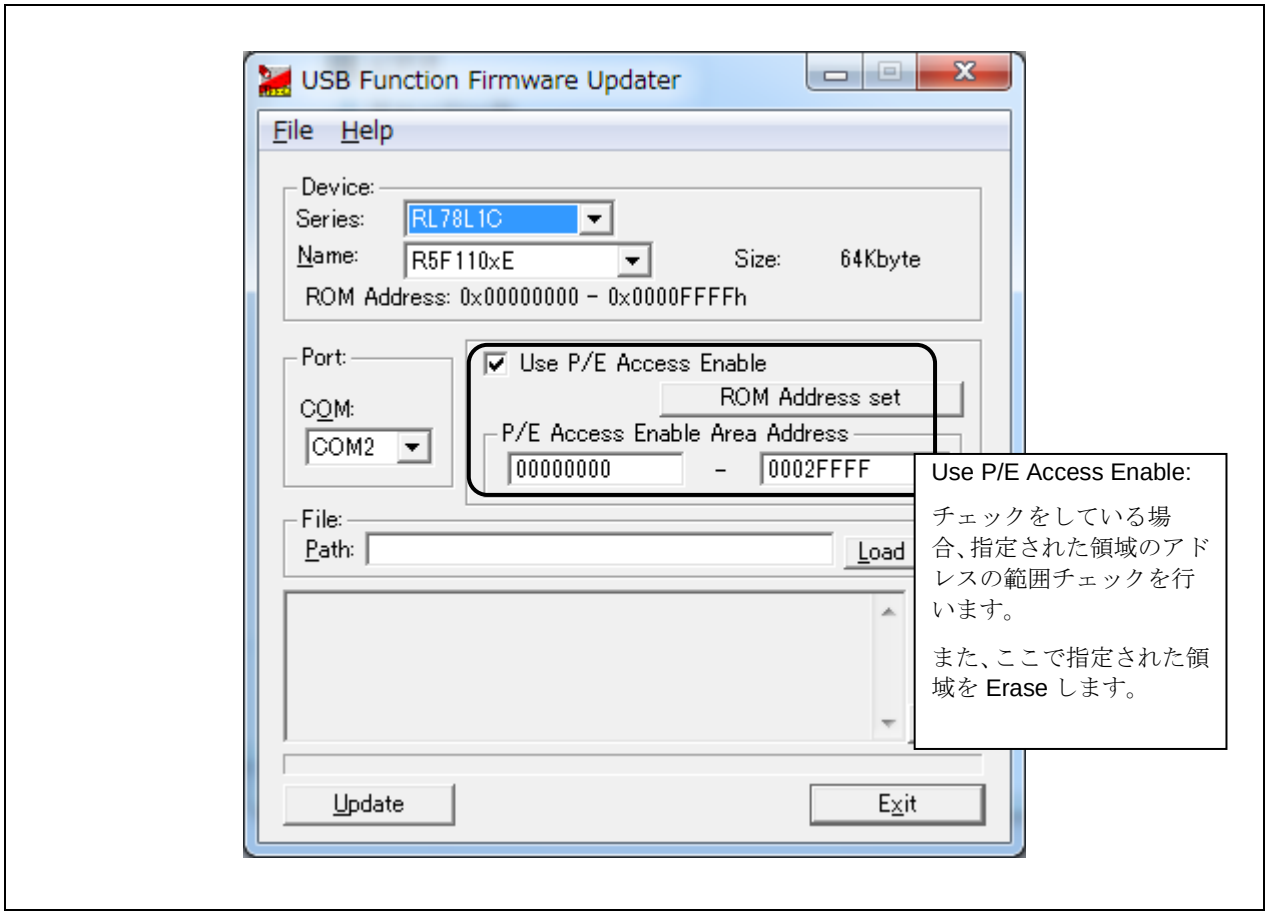

Figure 4-9　P/E 制限領域の設定

(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。

(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。

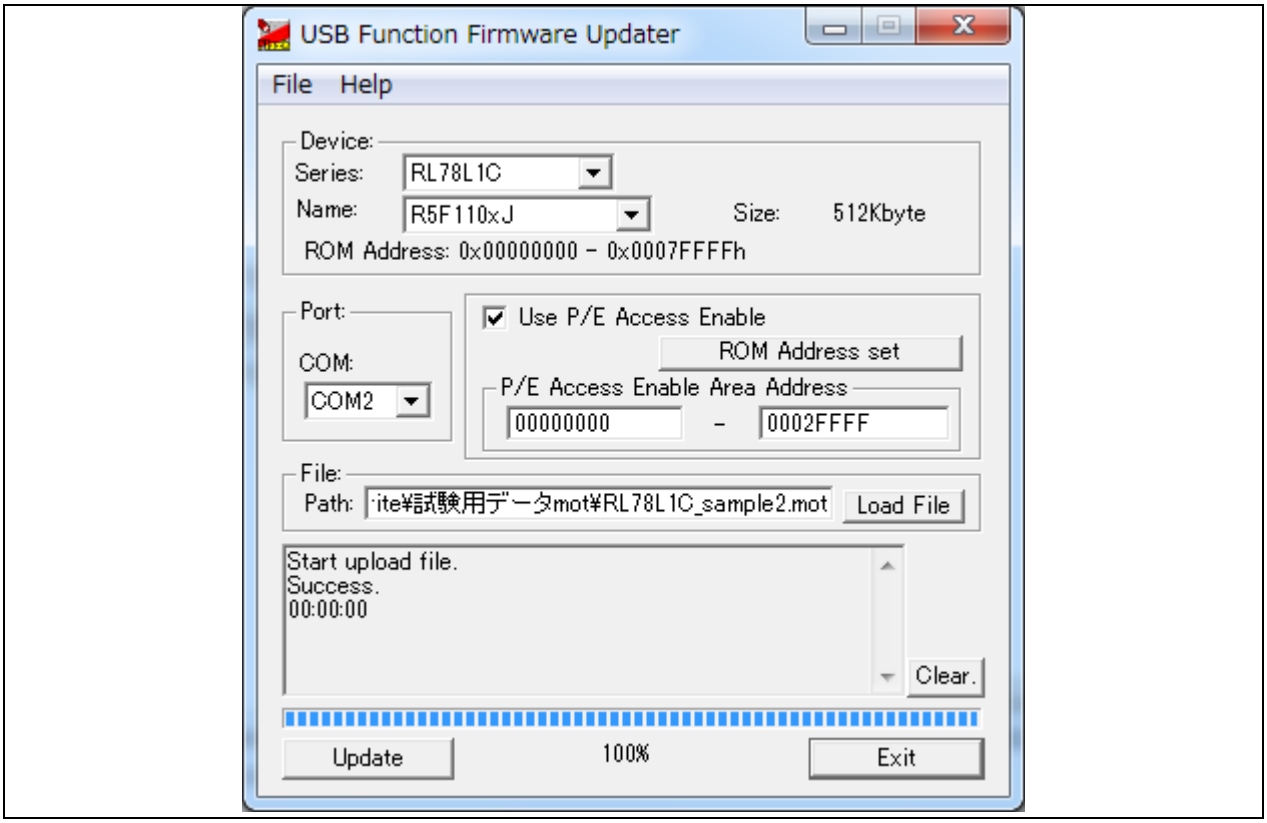

Figure 4-10　書き換え処理終了

(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 ボードの LED が順に点灯します。

(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 ペリフェラル・ファームウエア・アップデータを起動して、(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 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

USB ペリフェラル・ファームウエア・アップデータ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 ペリフェラル・ファームウエア・アップデータの書き込みからやり直して下さい。

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ペリフェラル・ファームウエア・アップデータ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 4-11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。

(1). 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。

(2). 《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。

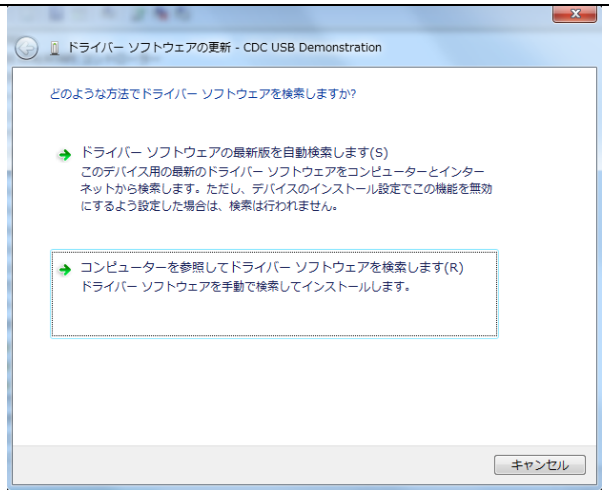

Figure 4-11　新しいハードウェアの検索ウィザード

(3). 《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。

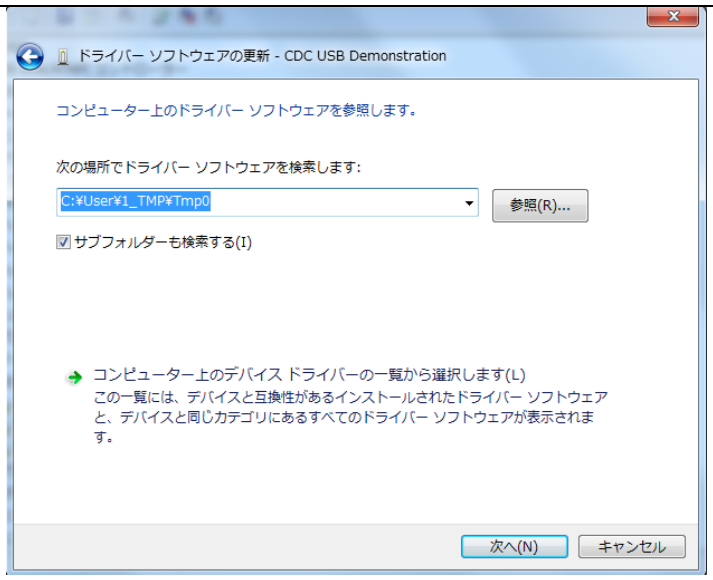

Figure 4-12　ドライバの場所の選択

(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。

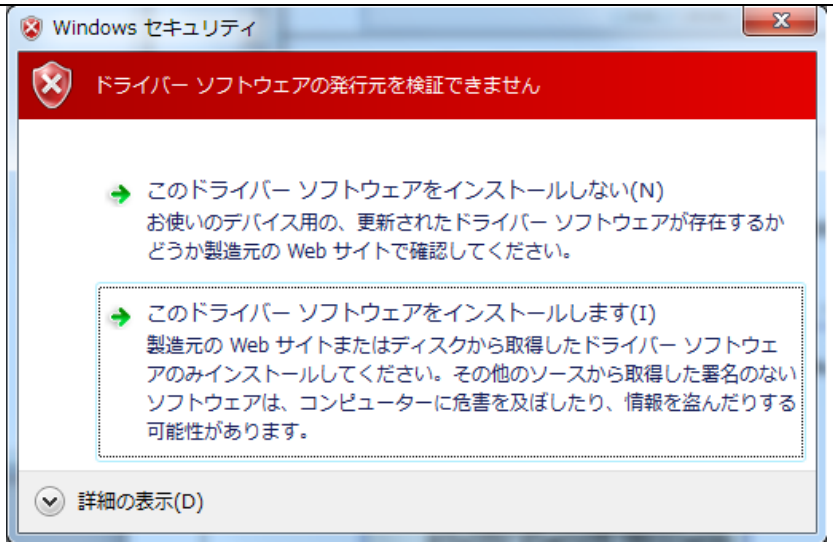

Figure 4-13　インストール確認

(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。

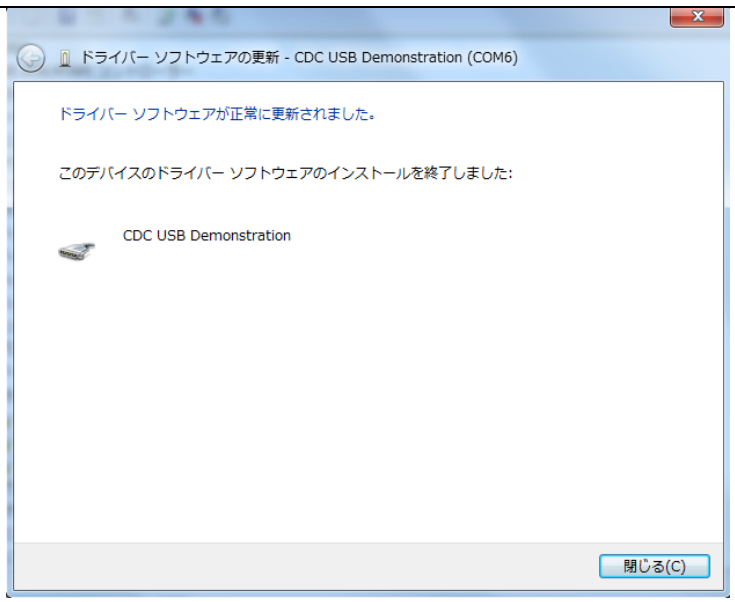

Figure 4-14　インストール完了

※　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ペリフェラル・ファームウエア・アップデータ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・タイプS（32ビット）

・モトローラ・タイプS（スタンダード）

・インテル拡張フォーマットファイル

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

USBペリフェラル・ファームウエア・アップデータ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。

UserApp Header

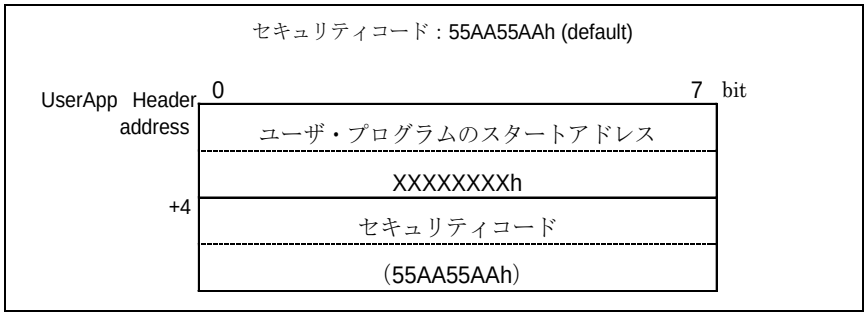

Figure 5-1 UserApp Header 領域

このヘッダ情報をUSBペリフェラル・ファームウエア・アップデータ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、「7.2.1 **電源投入時の動作フロー**」 を参照して下さい。

6. USB ペリフェラル・ファームウエア・アップデータとユーザプログラムに対する設定

USB ペリフェラル・ファームウエア・アップデータ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、そのセクションに対し任意のアドレスを設定して下さい。なお、UserApp Header 領域は、USB ペリフェラル・ファームウエア・アップデータ (固定ベクタを含む) と重ならない領域に配置して下さい。

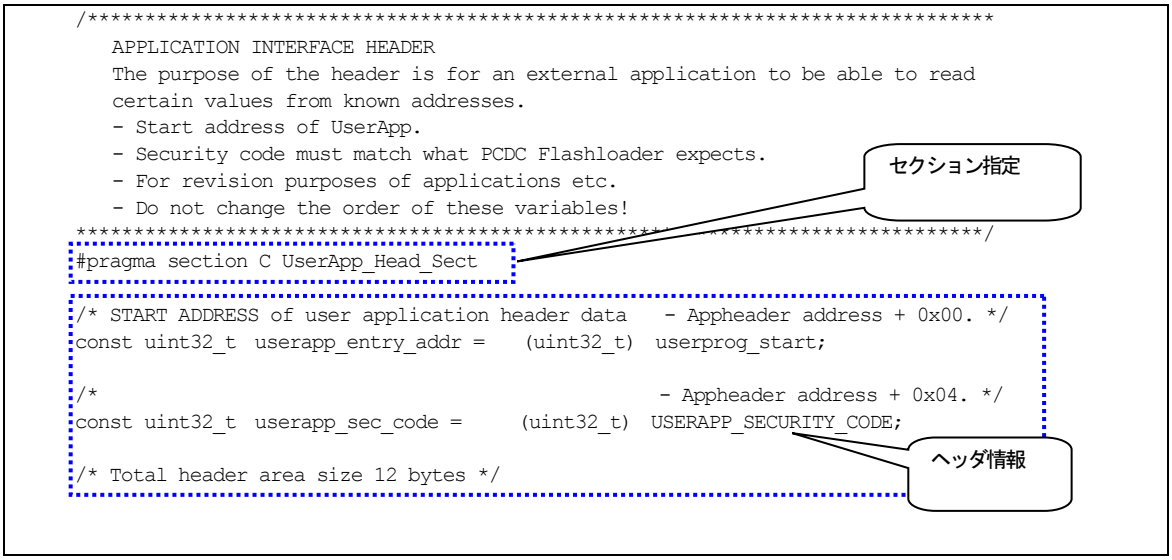

Figure 6-1　UserApp Header コード例

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。

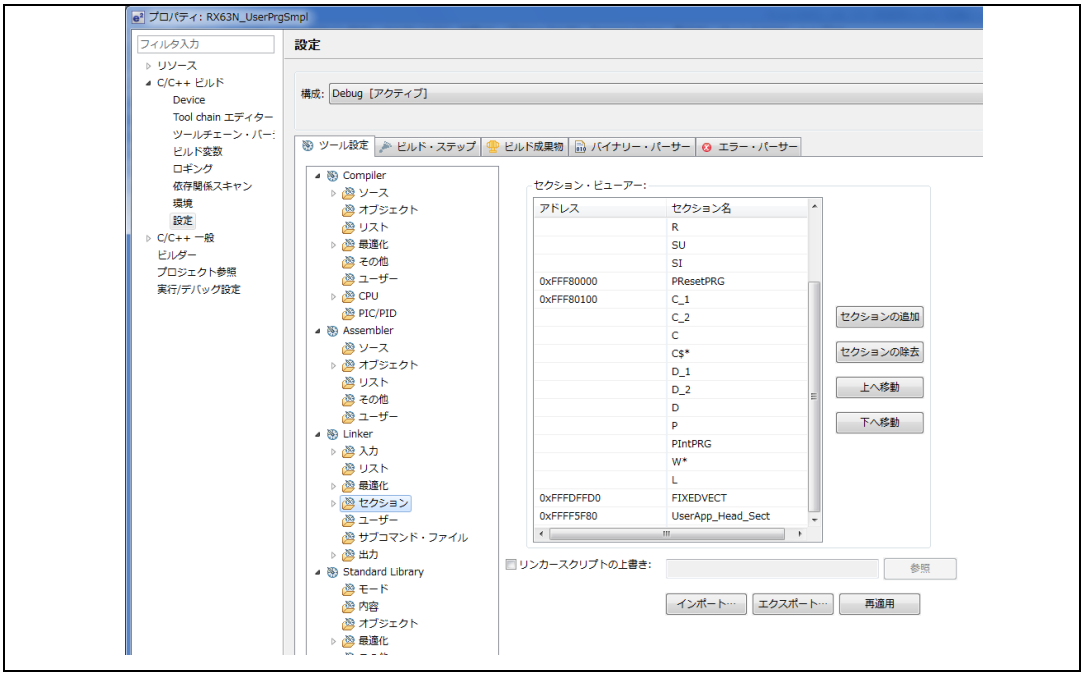

Figure 6-2　サンプル・プログラムのセクション設定例

6.2 USB ペリフェラル・ファームウエア・アップデータの設定

1. 設定内容 1 (r_config¥r_usb_updater_config.h)

USB ペリフェラル・ファームウエア・アップデータは、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 UseApp 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。

(1). 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

(2). 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

(3). FlashROM 書き換えプログラム配置アドレス指定

下記の定義に対し Flash ROM 書き換えプログラムの配置アドレスを指定してください。

#define	UPDATER_SECTION_START_ADDR	USBペリフェラル・ファームウエア・アップデータ開始アドレス

2. 設定内容 2

USB ペリフェラル・ファームウエア・アップデータは、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します。判定処理は、main 関数内で行っています。

3. 設定内容 3

USB ペリフェラル・ファームウエア・アップデータは、ユーザプログラム内にあるオプション・バイト領域の ROM 書き込みを行いません。本アップデータのオプション・バイトが、ユーザプログラムでも使用されますので、オプション・バイトに対する設定は、本アップデータに対して行ってください。

6.3 ユーザプログラムのリセットベクタ

USB ペリフェラル・ファームウエア・アップデータをユーザシステムに実装した場合、リセットベクタはUSB ペリフェラル・ファームウエア・アップデータ側のリセットベクタを使用するため、ユーザプログラム内にあるリセットベクタの書き込みは行われません。

6.4 ユーザプログラムの配置

ユーザプログラム(ベクタテーブル領域を除く)は、USBペリフェラル・ファームウエア・アップデータが書き込まれた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(ベクタテーブル領域を除く)は、以下のROM領域に配置されるようにユーザプログラムの配置設定を行ってください。

MCU	ユーザプログラム配置可能領域		
RL78/G1C	0x00002000	-	0x00007FFF
RL78/L1C	0x00002000	-	0x0003FFFF

2. ユーザプログラムをROMへ書き込む際に、P/Eアクセス制限を行うことでUSBペリフェラル・ファームウエア・アップデータ領域への上書きを防ぐことができます。詳細は「4.1.1 P/E Access Enable Area Address」を参照下さい。

3. Flash Self programmingライブラリはRAM領域の一部を使用しますが、USBペリフェラル・ファームウエア・アップデータ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 ペリフェラル・ファームウエア・アップデータの解説

この章では，USBペリフェラル・ファームウエア・アップデータ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ペリフェラル・ファームウエア・アップデータのソース・ファイルとフォルダ構成を示します。

```
(Compiler name)
  +src
    +———APL          [FW Updater application]
    +———config       [FWUpdater configuration file]
    +———FSL          [Flash Self Library]
    +———HwResource   [Hardware Resource Setting]
    +———USB          [USB driver]
          +——— inc                USB driver common header file
          +——— src                USB driver
```

Figure 7-1 USB ペリフェラル・ファームウエア・アップデータのフォルダ構成

7.1.1 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_usb_fwupdater_config.h	USBペリフェラル・ファームウエア・アップデータ設定ファイル

7.1.2 FSL フォルダ

Flash Seif Libray関連のライブラリファイルおよびインクルードファイルをこのフォルダに格納してください。

7.1.3 APL フォルダ

USBペリフェラル・ファームウエア・アップデータのソース・ファイルが格納されているフォルダです。

Table 7-2 USB ペリフェラル・ファームウエア・アップデータのソース・ファイル

ファイル名	説　明
src¥main.c	C言語メイン関数記述ファイル
src¥r_usb_main_flashapi.c	ROM書き換えプログラム処理ファイル
inc¥r_usb_main_flashapi.h	ROM書き換えプログラムヘッダファイル

7.1.4 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 ペリフェラル・ファームウエア・アップデータのソース・ファイル

ファイル名	説　　明
inc¥r_usb_fwupdater_defreg.h	USBレジスタの初期化、設定用定義
inc¥r_usb_fwupdater_defusr.h	USBレジスタの指定。ユーザ設定箇所
inc¥r_usb_fwupdater_extern.h	関数Extern
inc¥r_usb_ctypedef.h	型定義
inc¥r_usb_defvalue.h	USB設定用定義
inc¥r_usb_usrconfig.h	USBのユーザ設定ファイル
inc¥r_usb_sysdef.h	USBシステム情報
src¥r_usb_fwupdater_api.c	USB 送受信、初期化処理ファイル
src¥r_usb_fwupdater_basic.c	USB メイン処理
src¥r_usb_fwupdater_classcdc.c	USB CDC処理
src¥r_usb_fwupdater_usbreg.c	USB レジスタ設定など

7.1.5 HwResource フォルダ

RSK がサポートしている周辺用のドライバが格納されているフォルダです。

Table 7-4　周辺ドライバファイル

ファイル名	説　　明
src¥keydriver.c	Switch用ドライバファイル
src¥lcddriver.c	LCD用ドライバファイル
src¥rl78usbmcu.c	RL78設定ファイル

7.2 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。

RL78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 ペリフェラル・ファームウエア・アップデータからユーザプログラムへジャンプしてくるため、必ずUSB ペリフェラル・ファームウエア・アップデータが処理され、上記に記載したマイコンの初期化処理等が行われます。

7.2.1　電源投入時の動作フロー

次に，USBペリフェラル・ファームウエア・アップデータの電源投入時の動作フローについて説明します。

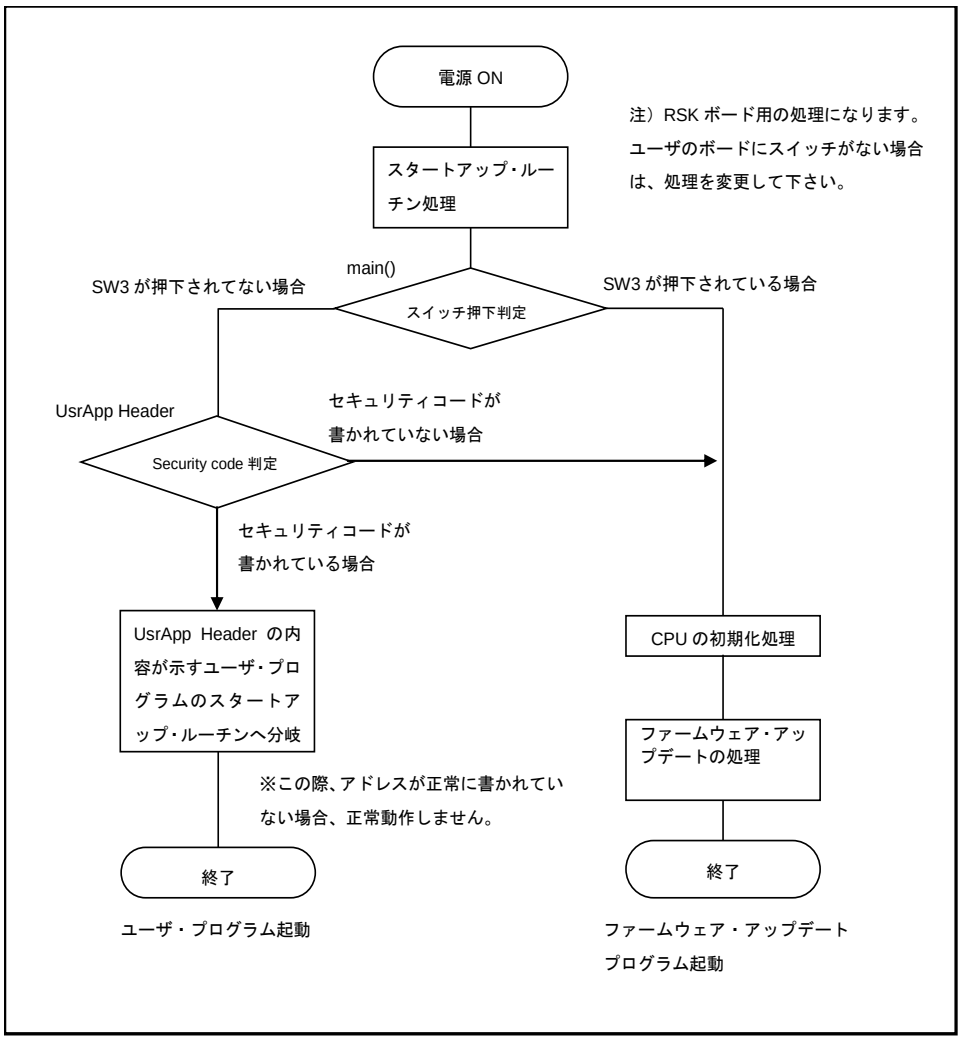

Figure 7-2　電源投入時の動作フロー

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、UsrApp 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。

7.2.2 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。

a. 正しいセキュリティコードが設定されている。

b. 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。

なお、セキュリティコードが不一致(正しくない)の場合、USB ペリフェラル・ファームウエア・アップデータのみが起動し、ユーザプログラムは起動しません。

7.2.3 USB ペリフェラル・ファームウエア・アップデータの起動条件

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 SW3 を押下した状態でリセット起動するとUSB ペリフェラル・ファームウエア・アップデータが起動します。

2. ユーザプログラムが ROM に書かれていない場合

   USB ペリフェラル・ファームウエア・アップデータが起動します。

7.3 注意事項

USB ペリフェラル・ファームウエア・アップデータは、評価ボード上の SW の状態により、USB ペリフェラル・ファームウエア・アップデータにジャンプするか、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します。判定処理を行っている部分は、USB ペリフェラル・ファームウエア・アップデータの main 関数です。

7.4　USB 経由内蔵 FlashROM 書き換え用関数

ここでは、USBペリフェラル・ファームウエア・アップデータ内で使用する関数を示します。
ただし、シンプルフラッシュAPI関連の関数を除きます。

7.4.1　データ・タイプ

USBペリフェラル・ファームウエア・アップデータにおける，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

7.4.2　構造体

Table 7-6　スタート・レコード定義

データ・タイプ	変数名	内容
uint8_t	device_type	デバイス種別（0x00固定）
uint8_t	rsv1[1]	リザーブ
uint8_t	rom_start_addr[4]	ROMイレース開始アドレス
uint8_t	rom_end_addr[4]	ROMイレース終了アドレス
uint8_t	rsv2[1]	リザーブ
uint8_t	checksum	チェック・サム

Table 7-7　エンド・レコード定義

データ・タイプ	変数名	内容
uint8_t	record_type	レコード種別
uint8_t	record_len	レコード長
uint8_t	dev_type	デバイス種別（0x00固定）
uint8_t	checksum	チェック・サム

Table 7-8　レスポンス・レコード定義

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 7-9 ROM 書き込み構造体定義

データ・タイプ	変数名	内容
uint8_t	recv_data[ROM_WRITE_SIZE]	ROM書き込み用バッファ
uint32_t	now_addr	今回書き込みアドレス
uint32_t	next_addr	次回書き込み予定アドレス
uint32_t	dest_addr	書き込み基準アドレス。 [dest_addr + wr_count * ROM_WRITE_SIZE] が書き込みアドレスになる。
uint16_t	wr_count	基準アドレスからの書き込み回数。
uint16_t	counter	現在のデータ数 <0~ROM_WRITE_SIZE-1>

FALSH書き込みサイズ(ROM_WRITE_SIZE)をタイプごとに固定で持っており、書き込み数分のデータがたまるまで受信データバッファからROM書き込み用バッファにコピーします。

書き込み基準アドレスは、データが連続している間は保持し続けます。書き込みアドレスが前回アドレスからROM_WRITE_SIZEを超えた位置になった場合、基準アドレスを取り直し、書き込み回数をクリアします。

7.4.3 Flash 書き込みメイン処理関数

Table 7-10 メイン処理関数一覧

ファイル名	関数名	処理概要
main.c	**main**	初期化、ユーザプログラムへのジャンプ判定
main.c	**cdc_read**	CDC の read 検知
main.c	**cdc_main**	受信データ処理分岐（イレース・書き込み・終了判定）
main.c	**cf_code_copy**	コードを RAM にコピー
main.c	**send_response_record**	アプリケーションへの返答
main.c	**sub_Byte2ddress**	4 バイトのアドレスを unsigned long でアドレス値に変換
main.c	**subproc_start_record**	スタート・レコード受信時の初期処理（ROM イレース等）
main.c	**sub_write_initial**	フラッシュ書き込み用変数の初期化
main.c	**sub_flashwrite_proc**	フラッシュ書き込み判定、書き込み処理
main.c	**jump_to_userapp**	ユーザプログラムへのジャンプコード
main_call_flashapi.c	**FSL_Func_Write**	ROM 書き込み API の呼び出し関数。Type により処理分岐
main_call_flashapi.c	**FSL_Func_Erase**	ROM 消去 API の呼び出し関数。Type により処理分岐
main_call_flashapi.c	**FSL_Func_SetAccessWindow**	ROM アクセス許可 API の呼び出し関数。Type1 のみ。

Table 7-11 main 関数

関数名		**main**
記述形式		void main (void)
機能		スタート時のエントリ関数。初期化処理とUSBペリフェラル・ファームウエア・アップデータ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「7.4.5USBペリフェラル・ファームウエア・アップデータへの分岐」をご参照下さい。**

Table 7-12 cdc_read 関数

関数名		cdc_read
記述形式		static uint16_t cdc_read(void)
機能		CDCのread検知
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_CONFIGUED：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 cdc_main 関数

関数名		cdc_main
記述形式		void cdc_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		動作の詳細は「7.4.7**USBペリフェラル・ファームウエア・アップデータのメイン処理**」をご参照下さい。

Table 7-14 cf_code_copy 関数

関数名		cf_code_copy
記述形式		static void cf_code_copy (void)
機能		指定のフラッシュ・メモリ・ブロックの全情報をRAM領域に退避させる
入 出力	入力	なし
	出力	なし
備考		Flash ROM書き込みモード（ROM P/E Mode）中はROMへのアクセスが制限されるため、ROM書き込み、ROM消去に使用するコードはRAMに退避させておく必要があります。

Table 7-15 subproc_start_record 関数

関数名		subproc_start_record
記述形式		static flash_err_t subproc_start_record(uint16_t flashseq_check)
機能		スタート・レコード受信時に指定のROM領域をイレースする。
入 出力	入力	uint16_t flashseq_check：Flash書き込み遷移状態。
	出力	なし
備考		なし

Table 7-16 send_response_record **関数**

関数名		**send_response_record**
記述形式		static void send_response_record (U8 response_type, U8 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 7-17 sub_Byte2num **関数**

関数名		**sub_Byte2num**
記述形式		static uint32_t sub_Byte2num(U8 * dat, uint16_t size)
機能		byte列をsize分接続して整数として返却するマクロ
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 7-18 sub_write_initial 関数

関数名		**sub_write_initial**
記述形式		static void sub_write_initial (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 7-19 sub_flashwrite_proc 関数

関数名		**sub_flashwrite_proc**
記述形式		static flash_err_t sub_flashwrite_proc(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　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 7-21　FSL_Func_Write 関数

関数名		FSL_Func_Write
記述形式		flash_err_t　FSL_Func_Write (void)
機能		ROM書き込みAPI呼び出し窓口関数
入　出力	入力	なし
	出力	flash_err_t：処理結果
備考		なし

Table 7-22　FSL_Func_Erase 関数

関数名		FSL_Func_Erase
記述形式		flash_err_t FSL_Func_Erase (const uint32_t start_addr, const uint32_t end_addr, uint16_t disable_check)
機能		ROM消去API呼び出し窓口関数。Type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） disable_check：Start/Endアドレスを含むブロックをアクセス不許可とし、イレース範囲から1ブロック分除外する(オプション設定メモリ対応用につき検討中)
	出力	flash_err_t：処理結果
備考		なし

Table 7-23　FSL_Func_SetAccessWindow 関数

関数名		FSL_Func_SetAccessWindow
記述形式		flash_err_t FSL_Func_SetAccessWindow (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 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。

7.4.4 USB ドライバ関数

Table 7-24は、USB ドライバの関数一覧です。

Table 7-24 USB モジュール関数一覧

ファイル名	関数名	処理概要
r_usb_fwupdater_api.c	**bulk_in_start**	USB バルク受信
r_usb_fwupdater_api.c	**bulk_out_start**	USB バルク送信
r_usb_fwupdater_api.c	**usb_init**	USB 初期化処理
r_usb_fwupdater_basic.c	**usbint**	USB 割り込み処理
r_usb_fwupdater_basic.c	**save_request**	リクエスト情報の取得
r_usb_fwupdater_basic.c	**ctrl_read_data_stage**	データ処理
r_usb_fwupdater_basic.c	**ctrl_write_nodata_stage**	データ以外の処理
r_usb_fwupdater_basic.c	**intr_int_pipe0**	パイプ 0 割り込み初期化
r_usb_fwupdater_basic.c	**bemp_int_pipe0**	Interrupt BEMP pipe0
r_usb_fwupdater_basic.c	**intr_int**	割り込み初期化
r_usb_fwupdater_basic.c	**intr_int_read**	受信割り込み初期化
r_usb_fwupdater_basic.c	**intr_int_write**	送信割り込み初期化
r_usb_fwupdater_basic.c	**ctr_read_start**	受信開始
r_usb_fwupdater_basic.c	**ctr_write_start**	送信開始
r_usb_fwupdater_basic.c	**write_fifo**	指定された USB FIFO に指定されたデータを書き込み
r_usb_fwupdater_basic.c	**read_fifo**	USB FIFO から指定されたデータバッファを読み込み
r_usb_fwupdater_basic.c	**chg_port**	Read specified buffer size from the USB FIFO
r_usb_fwupdater_basic.c	**req_get_descriptor**	ディスクリプタ取得
r_usb_fwupdater_basic.c	**req_set_configuration**	設定
r_usb_fwupdater_classcdc.c	**serial_init**	CDC 初期化
r_usb_fwupdater_classcdc.c	**reset_ep**	Pipe Configuration
r_usb_fwupdater_classcdc.c	**cdc_init**	Serial initialize
r_usb_fwupdater_classcdc.c	**class_write_data_stage**	Class request is "get line coding".
r_usb_fwupdater_classcdc.c	**class_read_data_stage**	Class request is "set line coding".
r_usb_fwupdater_classcdc.c	**class_write_nodata_stage**	Class request is "set control line state".
rl78usbmcu.c	**usb_cpu_mcu_initialize**	MCU 初期化
rl78usbmcu.c	**usb_int_init**	USB 割り込み初期化
rl78usbmcu.c	**delay_xus**	ウェイト
rl78usbmcu.c	**usb_cpu_int_disable**	USB 割り込み禁止
rl78usbmcu.c	**usb_cpu_usbint_init**	USB 割り込み初期化

7.4.5 USB ペリフェラル・ファームウエア・アップデータへの分岐

USBペリフェラル・ファームウエア・アップデータのmain()関数内でユーザプログラムにジャンプするかUSBペリフェラル・ファームウエア・アップデータ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ペリフェラル・ファームウエア・アップデータを実行します。

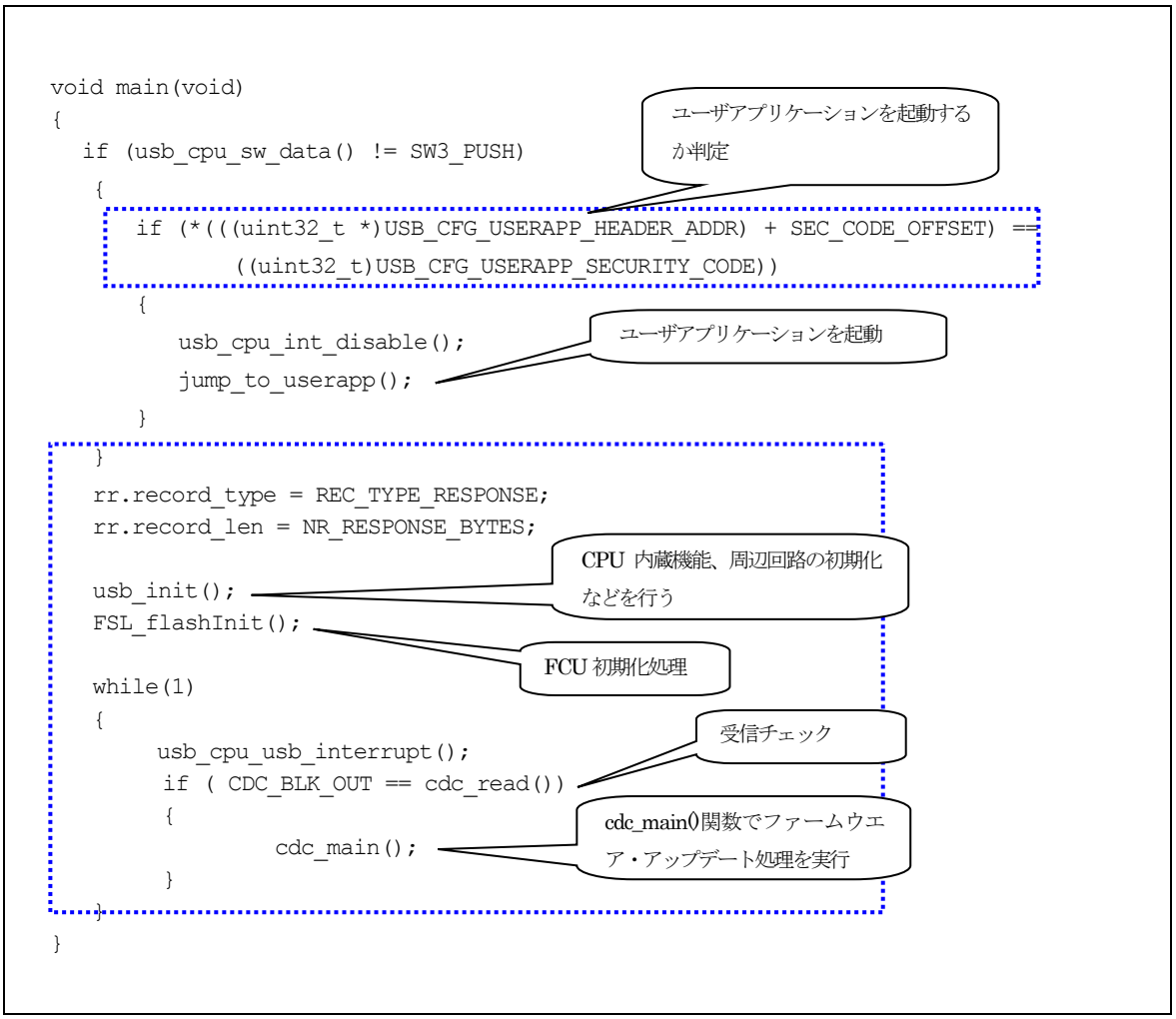

Figure 7-3 main()関数

7.4.6　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。

7.4.7　USB ペリフェラル・ファームウエア・アップデータのメイン処理

関数cdc_main()により、USBペリフェラル・ファームウエア・アップデータの処理が行われます。本関数はmain()関数から呼び出されます。

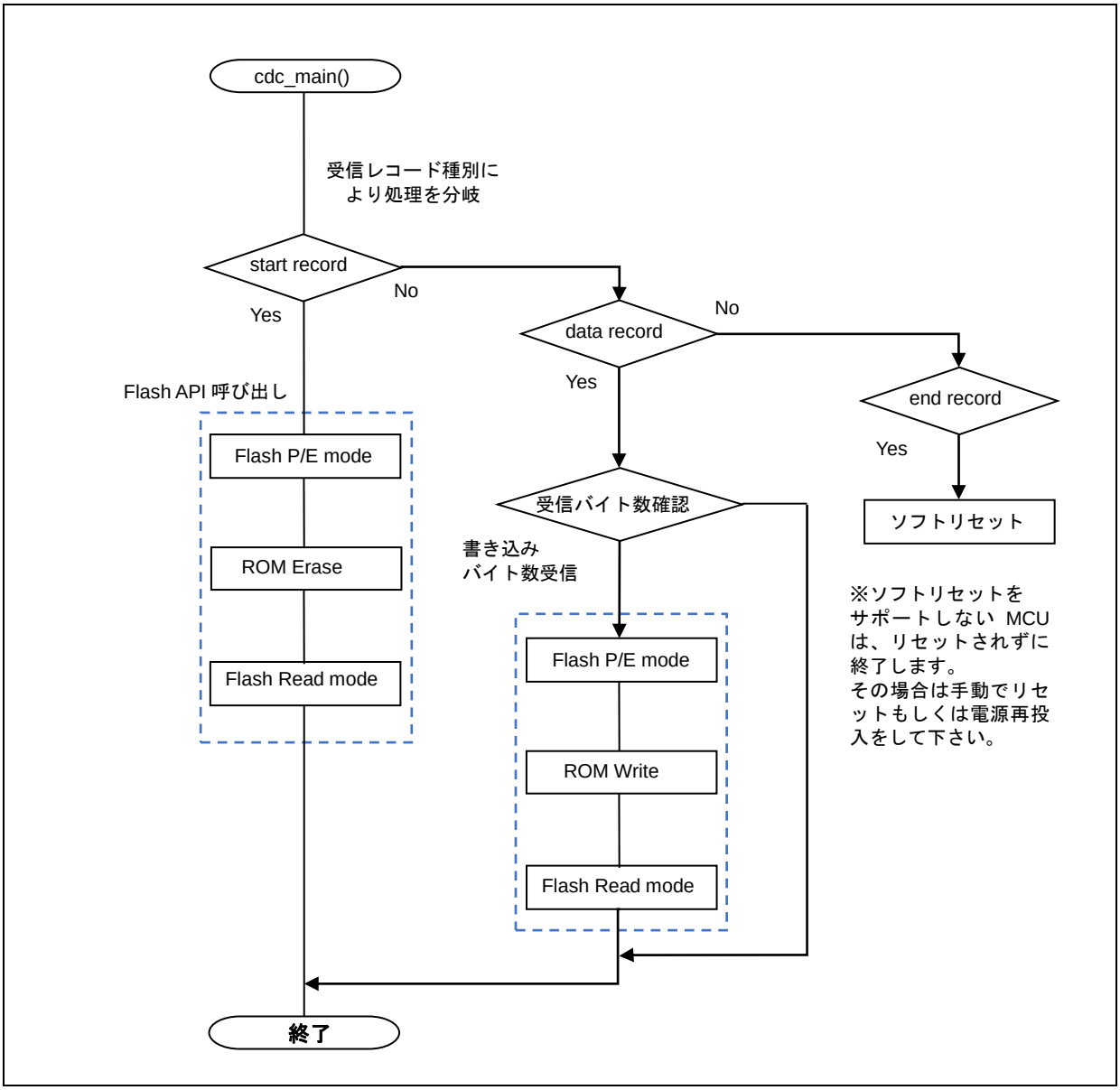

Figure 7-4　Flash 書き込み動作フロー

ROMイレース単位はブロック単位で行われます。書き込みサイズは、各ROMの仕様に依ります。

8.　ファイル転送アプリケーション（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 7, Windows 8.1, Windows 10
開発言語	：Microsoft Visual C++ 6.0（MFC）

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。

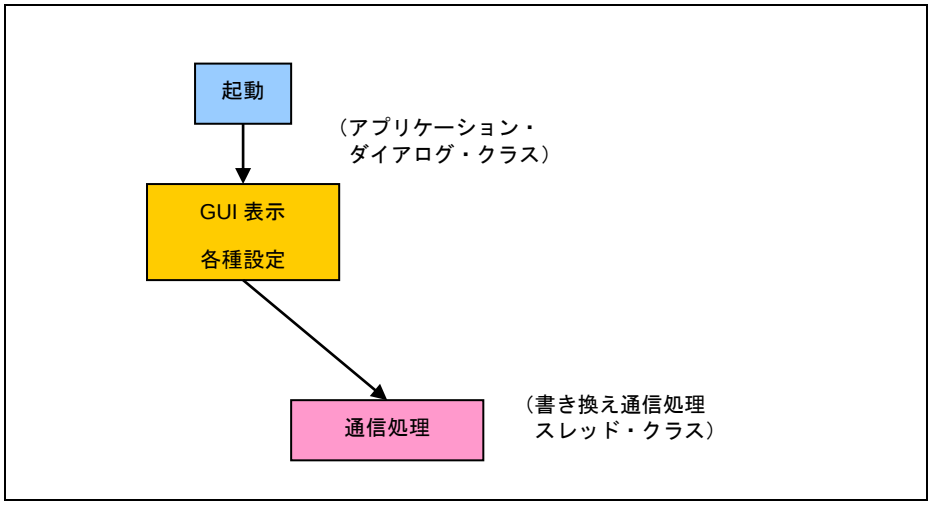

Figure 8-1　ファイル転送アプリケーション動作概要

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.dsw	ワークスペース・ファイル
FlashSelfRewriteGUI.dsp	プロジェクト・ファイル
FlashSelfRewriteGUI.clw	クラス・ウィザード用ファイル
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ペリフェラル・ファームウエア・アップデータの実行**」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
CWinThrread*	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
COleDateTime	m_dtStart	書き換え処理開始日付
COleDateTime	m_dtEnd	書き換え処理終了日付

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		初期化処理
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

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_strHex2Bibary 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. XXXXXXXX	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以降は設定できますが使用できません
	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。

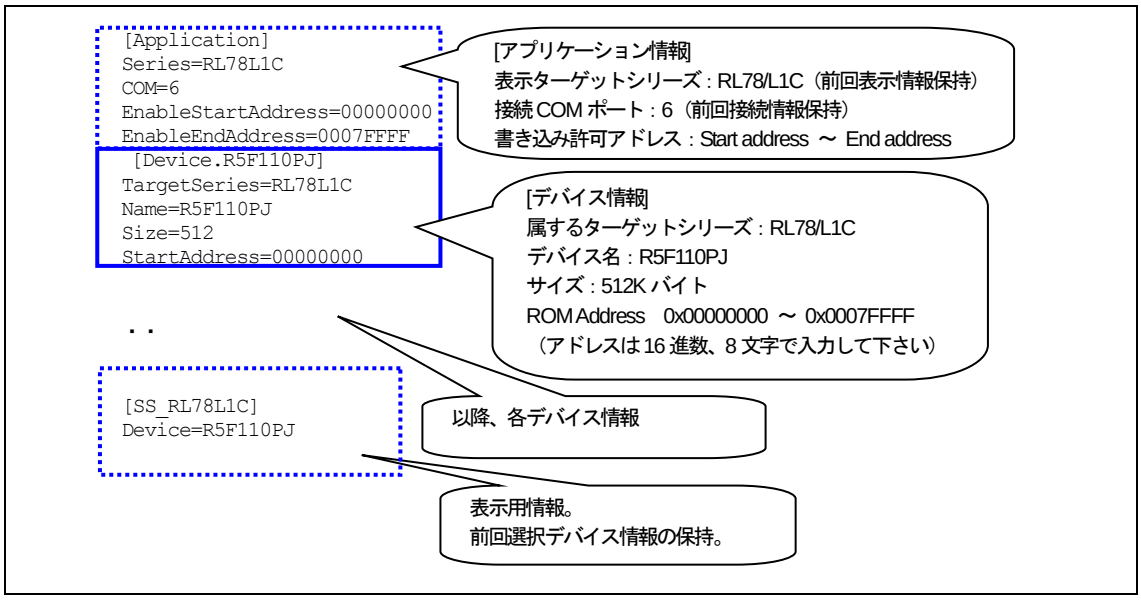

Figure 8-2　アプリケーション動作設定 ini ファイル

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
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フォーマット，インテル拡張フォーマット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: 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: 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 Acccess Enable Area指定がROM領域を超えているか、異常値の場合（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領域を超えている場合
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: 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。

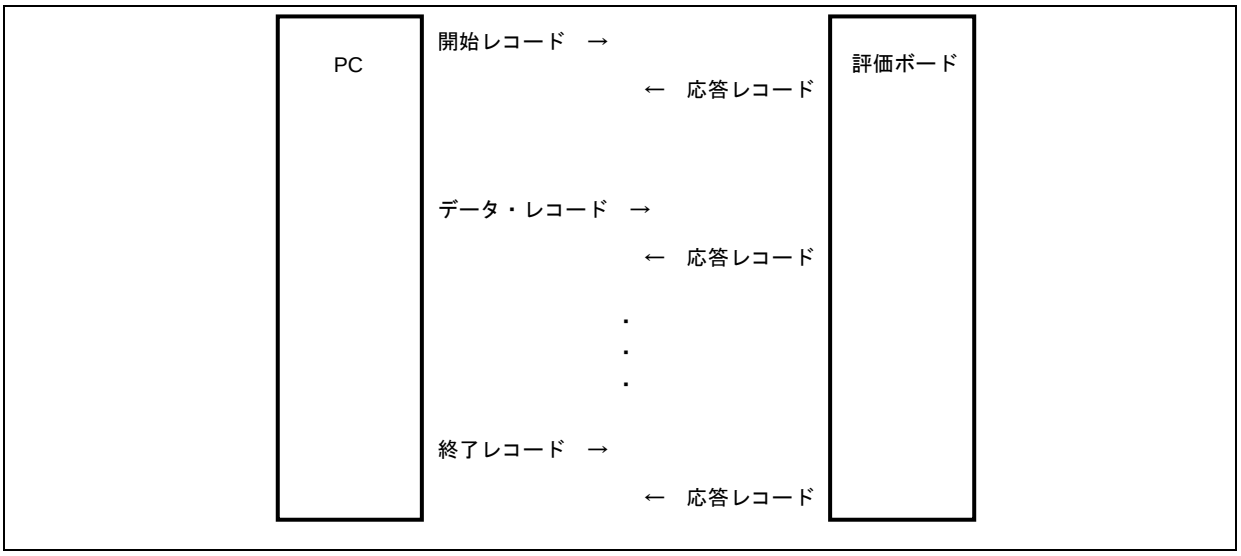

Figure 9-1　通信データ・シーケンス

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

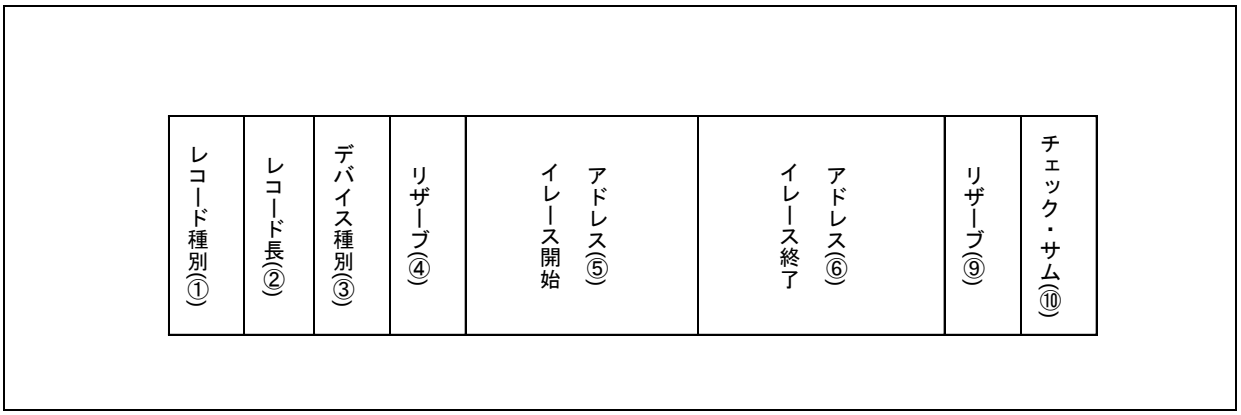

Figure 9-2　開始レコードの形式

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・エンディアンの形式

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・エンディアンの形式

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）

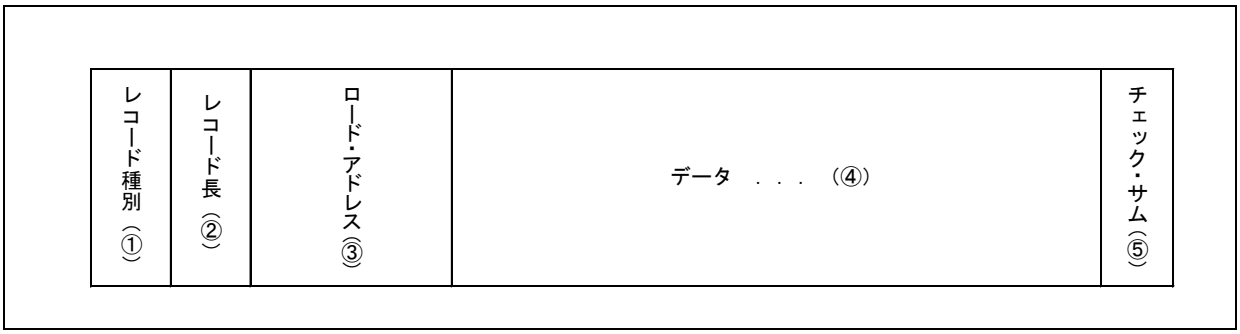

Figure 9-3　データ・レコードの形式

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

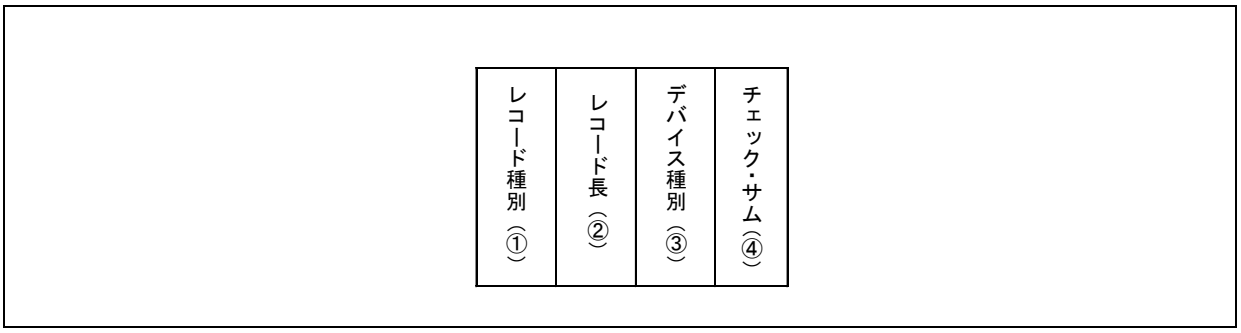

Figure 9-4　終了レコードの形式

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。：5～8バイト

(1).　応答レコード

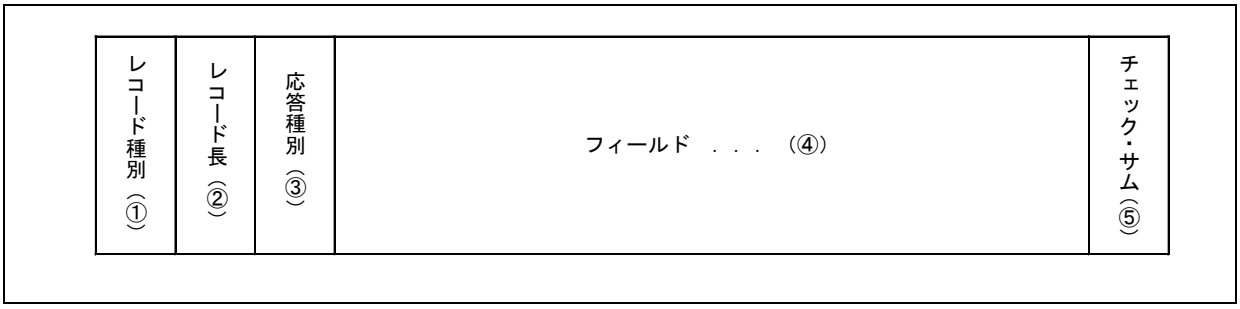

Figure 9-5　応答レコードの形式

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：1～4 バイト

エラーの場合は，エラー・コード1バイト

エラーでない場合は，レコード種別によって内容が異なる

－開始レコード　　：デバイス種別、Write/Eraseアクセス制限アドレス

－データ・レコード　　：ロード・アドレス

－終了レコード　　：デバイス種別

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

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/

お問合せ先
http://japan.renesas.com/contact/

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
1.10	Sep 30, 2016	－	USB ドライバ部が変更されました。

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、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

   【注意】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. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。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。

4. クロックについて

   【注意】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。

5. 製品間の相違について

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。お客様が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. 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

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その総株主の議決権の過半数を直接または間接に保有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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/