

RX100/RX200 シリーズ

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

要旨

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

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

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

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

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

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

対象デバイス

- ・RX230 グループ、RX231 グループ、RX260 グループ、RX261 グループ
- ・RX110 グループ、RX111 グループ、RX113 グループ、RX130 グループ、RX140 グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様に合わせて変更し、十分 評価してください。

関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
- RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)
- RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
- RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- RX ファミリ フラッシュモジュール Firmware Integration Technology (R01AN2184)
- RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)
- RX ファミリ バイト型キューバッファ (BYTEQ) モジュール Firmware Integration Technology (R01AN1683)



目次

1. 概要	4
1.1 本アプリケーションノートについて	4
1.2 動作環境	5
1.3 モジュール構成	6
1.4 ファイル構成	7
1.5 プロジェクトについて	9
2. 開発環境の入手	10
2.1 e² studio の入手方法	10
2.2 CS+の入手方法	10
2.3 コンパイラパッケージの入手方法	10
2.4 Renesas Flash Programmer の入手方法	10
3. フロジェクトの構築	11
3.1 ワークスペースの作成	11
3.2 プロジェクトの作成	13
3.3 プロジェクトのインポート	16
3.4 変更情報	19
3.4.1 コンフィギュレーションオプション	19
3.4.2 プロジェクトの設定変更	21
	00
4.1 ノロンェクトのヒルト	
4.2 テハックの準備	
4.2.1	
4.2.2 ホスト PC の設定	
4.3 フロジェクトのデバック	
5. アプリケーション概要	
5.1 ファームウェアアップデートプログラムの構成	
5.2 動作概要	
5.2.1 ファームウェアアップデートプログラムの書き込み	
5.2.2 ファームウェアプログラムの書き込み	
5.2.3 ファームウェアプログラムの更新	
5.2.4 ファームウェアアップデートプログラムの更新	
5.2.5 ファームウェアプログラムの動作	60
5.2.6 書き込みが失敗した場合の復旧方法	64
5.2.7 更新用ファームウェアアップデートプログラムの作成方法	65
5.3 ファームウェアアップデートプログラムの概略フローと画面出力	
5.3.1 メイン処理	
5.3.2 ファームウェアアップデート処理	
5.3.3 ファームウェア起動処理	71
5.4 ファームウェアプログラムの概略フローと画面出力	73
5.5 ファームウェアアップデートプログラム詳細	75



5.5.1 ファイル構成7	5
5.5.2 定数一覧7	6
5.5.3 型定義一覧8	2
5.5.4 変数一覧	5
5.5.5 関数一覧	9
5.6 ファームウェアプログラム詳細9	2
5.6.1 ファイル構成	2
5.6.2 定数一覧	3
5.6.3 型定義一覧	3
5.6.4 変数一覧	4
5.6.5 関数一覧	5
6. プロジェクトをインポートする方法	7
6.1 CS+での手順9	7
7. 参考ドキュメント	8



1. 概要

1.1 本アプリケーションノートについて

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

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

表 1.1 に使用する周辺機能と用途を、図 1.1 に動作概要を示します。

表 1.1 使用する周辺機能と用途

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



図 1.1 動作概要



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

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

1.2 動作環境

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

項目	内容
使用マイコン	R5F51305ADFN(RX130 グループ)
	R5F51406BDFN (RX140 グループ)
	R5F52318ADFP(RX231 グループ)
	R5F52618BGFP(RX261 グループ)
使用ボード	Renesas Starter Kit for RX130(製品型名: RTK5005130C00000BE)
	Renesas Starter Kit for RX140(製品型名: RTK551406BS00000BE)
	Renesas Starter Kit for RX231(製品型名: R0K505231C000BE)
	RX261 MCU グループ評価キット(製品型名: RTK5EK2610S00001BE)
統合開発環境	ルネサスエレクトロニクス製
	e ² studio Version 2024-07
	ルネサスエレクトロニクス製
	CS+ V.8.12.00
Cコンパイラ	ルネサスエレクトロニクス製
	C/C++ Compiler Package for RX Family V.3.06.00
	コンパイラオプション
	-lang = C99
フラッシュプログラマ	Renesas Flash Programmer V.3.16.00
エミュレータ	E2 Lite
エンディアン	リトルエンディアン

表 1.2 動作環境



1.3 モジュール構成

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



図 1.2 モジュール構成

表	1.3	モジュー	-ルー覧
---	-----	------	------

種類	アプリケーションノート名(型名)	FIT モジュール名
BSP	RX ファミリ ボードサポートパッケージモジュール Firmware	r_bsp
	Integration Technology	
	(R01AN1685)	
デバイスドライバ	RX ファミリ フラッシュモジュール Firmware Integration	r_flash_rx
	Technology	
	(R01AN2184)	
デバイスドライバ	RX ファミリ SCI モジュール Firmware Integration Technology	r_sci_rx
	(R01AN1815)	
デバイスドライバ	RX ファミリ バイト型キューバッファ(BYTEQ)モジュール	r_byteq
	Firmware Integration Technology	
	(R01AN1683)	
アプリケーション	メインプログラム	src



1.4 ファイル構成

図 1.3 に本アプリケーションノートのファイル構成を示します。



図 1.3 ファイル構成



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

「r01an3740_rx130/r01an3740_rx140/r01an3740_rx231/r01an3740_rx261」フォルダの下にある 「ファームウェアアップデートプロジェクト(update_firmware_rx130/update_firmware_rx140/ update_firmware_rx231/update_firmware_rx261)」および「ファームウェアプロジェクト (firmware_rx130/firmware_rx140/firmware_rx231/firmware_rx261)」は、本アプリケーションノートを 構築したプロジェクトです。e² studio のワークスペースにインポートすることでアプリケーションノートの 動作確認が可能です。



1.5 プロジェクトについて

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

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

表 1.4 プロジェクト設定

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

表	1.5	ター	ゲッ	ト固有の設定
---	-----	----	----	--------

項目	設定
ツールチェーン・バージョン	V3.04.00
デバッグ・ハードウェア	E2 Lite (RX)
データ・エンディアン	Little-endian data
ターゲットの選択(RX130 グループ)	R5F51305ADFN (RX130 LFQFP 80pin)
ターゲットの選択(RX140 グループ)	R5F51406BDFN (RX140 LFQFP 80pin)
ターゲットの選択(RX231 グループ)	R5F52318ADFP(RX231 LQFP 100pin)
ターゲットの選択(RX261 グループ)	R5F52618BDFP(RX261 LFQFP 100pin)
Renesas RTOS サポート	None



2. 開発環境の入手

2.1 e² studio の入手方法

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

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

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

2.2 CS+の入手方法

以下の URL にアクセスし、CS+をダウンロードしてください。 https://www.renesas.com/jp/ja/software-tool/cs.html

2.3 コンパイラパッケージの入手方法

以下の URL にアクセスし、RX ファミリ用 C/C++コンパイラパッケージをダウンロードしてください。 https://www.renesas.com/jp/ja/software-tool/cc-compiler-package-rx-family.html

2.4 Renesas Flash Programmer の入手方法

以下の URL にアクセスし、Renesas Flash Programmer をダウンロードしてください。 https://www.renesas.com/jp/ja/software-tool/renesas-flash-programmer-programming-gui.html



3. プロジェクトの構築

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

3.1 ワークスペースの作成

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

📴 e² studio ランチャー	×
ディレクトリーをワークスペースとして選択	
e ² studio は、ワークスペースディレクトリを使用して、環境設定と開	発成果物を保存します。
ワークスペース(<u>W</u>): C:¥workspace	~ 参照(<u>B</u>)
ワークスペースフォルダを入力し	、ます。 -
□この選択をデフォルトとして使用し、今後この質問を表示しない	[[] IJ] 「起動(L)」をクリックします。
▶ 最近のワークスペース(<u>R</u>)	
	起動(L) キャンセル



3. 以下の画面が表示されたら、「隠す」をクリックします。

1	workspace - e	e ² studio		((())	- 🗆 X
	71ル(E) 編集(E)	9-X(<u>S</u>)	リブアクダリンク(1) デビケート(N) 複索(A) ブロシエクト(P) Renesas <u>V</u> iev	/s 実行(<u>R</u>)	777F7(W) <\↓/(H) \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
Ц	RENE	SAS	Welcome to e ² studio		D D D D D D D D D D D D D D D D D D D
					「隠す」をクリックします。
İ		0	Create a new C/C++ project Create a new e ² studio C/C++ project		機要 フィーチャーの概要
		*	Import existing projects Import existing e ² studio projects from the filesystem or archive		<mark>チュートリアル</mark> チュートリアルの実行
		1	Import sample projects Download and import sample projects from Renesas website	1	サンプル サンブルの試行
		¢	Review IDE configuration settings Review the IDE's most fiercely contested preferences		<mark>新機能</mark> 新機能について
			Open an existing file Open a file from the filesystem	1	Quick Start Guides Quickly getting familiar with the tool
					□始動時にようこそを常に表示



3.2 プロジェクトの作成

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

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

	新規(N)	Alt+シフト+N >		Renesas C/C++ Proje	iect		>
0	ファイルを開く(.) ファイル・システムからプロジェクトを開く 最近のファイル	>		Synergy C/C++ Proje Makefile Project with C/C++ Project	ect th Existing Code		
	閉じる(C) すべて閉じる(L)	Ctrl+W Ctrl+シフト+W		フロジェクト(R) C/C++ プロジェクトに多	&換 (C/C++ ネ−ヲ	チャーを追加)	-
000	保存(S) 別名保存(A) すべて保管(E) 前回保管した状態に戻す(T)	Ctrl+S Ctrl+シフト+S		ソース・フォルタ フォルダー ソース・ファイル ヘッダー・ファイル テンプレートからファイル	c/C++ Project」	をクリックしま	ਰ ,
Ð	移動(V) 名前を変更(M) 更新(F) 行区切り文字の変換(D)	F2 F5	3 11 18 0	クラス コード生成 サンプル(X)		CtriaN	_
5	印刷(P)	Ctrl+P	F	CONE(0)		Curry	_
	インボート(I) エクスポート(O)						
	プロパティ(R)	Alt+Enter					
	ワークスペースの切り替え(W) 再開 終了/HID(X)	>					



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

新規 C/C++ ノ	コジェクトのテンプレート
All CMake Make Renesas Debug Renesas RX Renesas Synergy	GCC for Renesas RX C/C++ Executable Project A C/C++ Executable Project for Renesas RX using the GCC for Renesas RX Toolchain. GCC for Renesas RX C/C++ Library Project A C/C++ Library Project for Renesas RX using the GCC for Renesas RX Toolchain.
	Renesas CC-RX C/C++ Executable Project A C/C++ Project for Renesas RX using the Renesas CCRX toolchain. Renesas CC-RX C/C++ Library Project A C/C++ Library Project for Renesas RX using the Renesas CCRX toolchain.
「Renesas F 「Renesas C	X」をクリックし、 :C-RX C/C++ Executable Project」を選択します。
	「次へ(N)」をクリックします。

任意のプロジェクト名を入力します。[次へ(N)>]をクリックしてください。

New Renesas C	-RX Even	- x
New Renesas CC-	任意のプロジェクト名 X Executable Project	を入力します。
プロジェクト名(P)	Imple	
☑ デフォルト・ロケー	ションの使用(<u>D</u>)	
ロケーション(L):	C:¥workspace¥sample	参照(<u>R</u>)
ファイル・システムを通	 Create Directory for Project 択(い): デフォルト 	
ワーキング・セット		
ロワーキング・セッ	にプロジェクトを追加(工)	新規(<u>W</u>)
ワーキング・セット(C	н.	· 選択(E)
	「次へ(N)」	をクリックします。
(2)	< 戻る(<u>B</u>) 次へ(<u>N</u>) >	峰了(E) キャンセル



 「ターゲット・デバイス:」を設定します。RX130 グループの場合は「R5F51305AxFN」、RX140 グ ループの場合は「R5F51406BxFN」、RX231 グループの場合は「R5F52318AxFP」、RX261 グループ の場合は「R5F52618BxFP」としてください。その他の項目は任意の設定で構いません。設定が完了 してから[終了(F)]をクリックします。※下図は RX231 グループを使用する場合の例です。





3.3 プロジェクトのインポート

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

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



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



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

📴 リージョン設定	×	地域を選択します
リージョンを選択してください。	_/	地域で送訳しよう。
Japan	_ ~	クリックします。
ОК	7	



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



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

図 My Renesas ログイ My Renesasアカウン	ン の情報を入力してください。	М	× Renesas
メール・アドレス: [<u>/ᡭᡘᡃᠵ᠆ᡟ</u>	をお忘れの方?
パスワード: [<u>₽</u>	Proxy Settings
<u>My Renesas</u> へ	登録いただくと、ツール製品などのダウ:	ノロードサービスやメールニュースな	などの各種サー
ビスをご利用いた	どけるようになります。	ログイン	キャンセル

7. [Accept]をクリックします。

お客様が「同意します」ボタンもしくはDisclaimer002-JPN(以下、「本契約」といいま す)の電子コピーの契約条件に同意することを確認するために設計されたその他のボタンも しくはメカニズムをクリックし、または本契約のライセンス許諾対象のソフトウェア(以 下、「本ソフトウェア」といいます。)の全部もしくは一部をダウンロード、インストー ル、アクセスもしくはその他の手段により複製もしくは使用することで、(a)お客様は、お客 様が権限を有する被許諾者(以下、「ライセンシー」といいます。)を代理または代表して 本契約を締結し、それによりライセンシーが本契約に法的に拘束されることを承諾の上、本 契約を締結する意思表示を行ったこととなり、また、(b)お客様はライセンシーを代理また は代表し、ライセンシーを拘束する権利、権能および権限を有することを表明しかつ保証し たことになります。	▶ 免责事項 (サンプル・コード)	×
	6客様が「同意します」ボタンもしくは <u>Disclaimer002-JPN</u> (以下、「本契約」といいま す)の電子コピーの契約条件に同意することを確認するために設計されたその他のボタンも しくはメカニズムをクリックし、または本契約のライセンス許諾対象のソフトウェア(以 下、「本ソフトウェア」といいます。)の全部もしくは一部をダウンロード、インストー し、アクセスもしくはその他の手段により複製もしくは使用することで、(a)お客様は、お客 様が権限を有する被許諾者(以下、「ライセンシー」といいます。)を代理または代表して 体契約を締結し、それによりライセンシーが本契約に法的に拘束されることを承諾の上、本 契約を締結する意思表示を行ったこととなり、また、(b)お客様はライセンシーを代理また は代表し、ライセンシーを拘束する権利、権能および権限を有することを表明しかつ保証し たことになります。	~



- 8. 本アプリケーションノートを保存します。
- 9. RX130 グループの場合:「プロジェクト(P):」に表示されている「update_firmware_rx130」および 「firmware_rx130」の両方を選択し、「終了(F)」をクリックします。

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

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

RX261 グループの場合:「プロジェクト(P):」に表示されている「update_firmware_rx261」および 「firmware_rx261」の両方を選択し、「終了(F)」をクリックします。

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



3.4 変更情報

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

3.4.1 コンフィギュレーションオプション

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

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

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

(1) フラッシュ FIT モジュールの設定変更

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

プロパティ	值
✓	
# Parameter check	Enable parameter checks
# Enable code flash programming	Includes code to program ROM area
# Enable BGO/Non-blocking data flash operations	Forces data flash API function to block until completed.
# Enable BGO/Non-blocking code flash operations	Forces ROM API function to block until completed.
# Enable code flash self-programming	Programming code flash while executing in RAM.

(2) SCI FIT モジュールの設定変更

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

プロパティ	値
# Transmit end interrupt	Enable

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

プロパティ	値
~ 👜 SCI	
✓	
~ SCK1端子	回 使用する
∼ RXD1/SMISO1/SSCL1端子	☑ 使用する
~ TXD1/SMOSI1/SSDA1端子	使用する
~ CTS1#/RTS1#/SS1#端子	回 使用する



(3) SCI FIT モジュールの設定変更(RX231 グループ、RX261 グループ)

コンポーネント設定画面において、SCI FIT モジュールで使用するチャネルを CH1 から RX231 グループ では CH5 に、RX261 グループでは CH6 に変更しています。※下図は RX231 グループを使用する場合の例 です。

プロパティ	值
# Include software support for channel 0	Not
# Include software support for channel 1	Not
# Include software support for channel 2	Not
# Include software support for channel 3	Not
# Include software support for channel 4	Not
# Include software support for channel 5	Include

コンポーネント設定画面において、RX231 グループでは RXD5 と TXD5、RX261 グループでは RXD6 と TXD6 の端子を有効にしています。※下図は RX231 グループを使用する場合の例です。

プロパティ	值
V 🐵 SCI	
✓	
➤ SCK0端子	回 使用する
▲ RXD0/SMISO0/SSCL0端子	□ 使用する
► TXD0/SMOSI0/SSDA0端子	[] 使用する
▲ CTS0#/RTS0#/SS0#端子	□ 使用する
✓	
▲ 5CK1端子	回 使用する
▲ RXD1/SMISO1/SSCL1 端子	□ 使用する
▲ TXD1/SMOSI1/SSDA1端子	自使用する
▲ CTS1#/RTS1#/SS1#端子	[] 使用する
✓	
~ SCK5端子	🗐 使用する
∼ RXD5/SMISO5/SSCL5端子	 使用する
➤ TXD5/SMOSI5/SSDA5端子	 使用する
~ CTS5#/RTS5#/SS5#端子	Ⅲ 使用する

端子設定画面において、端子割り当てを RX231 グループでは RXD5 は PA3 に変更し、TXD5 は PA4 に 変更しています。RX261 グループでは RXD6 は PB0 に変更し、TXD6 は PB1 に変更しています。※下図は RX231 グループを使用する場合の例です。

使用	機能	端子割り当て	端子番号	方向	備考
	CTS5#	/ 設定されていません	/ 設定されていません	なし	
	RTS5#	✓ 設定されていません	/ 設定されていません	なし	
\checkmark	RXD5	PA3/MTIOC0D/MTCLKD/TIOCD0/TCLKB/R	/ 67	1	
	SCK5	✓ 設定されていません	/ 設定されていません	なし	
	SMISO5	/ 設定されていません	/ 設定されていません	なし	
	SMOS15	/ 設定されていません	/ 設定されていません	なし	
	SS5#	/ 設定されていません	/ 設定されていません	なし	
	SSCL5	/ 設定されていません	/ 設定されていません	なし	
	SSDA5	/ 設定されていません	/ 設定されていません	なし	
	TXD5	PA4/MTIC5U/MTCLKA/TMRI0/TIOCA1/TXI	/ 66	0	



3.4.2 プロジェクトの設定変更

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

プロジェクト設定を確認する場合、以下の手順で行ってください。

 e² studioの対象プロジェクト、RX130 グループの場合(update_firmware_rx130 または firmware_rx130)を、RX140の場合(update_firmware_rx140 または firmware_rx140)を、RX231 グ ループの場合(update_firmware_rx231 または firmware_rx231)を、RX261 グループの場合 (update_firmware_rx261 または firmware_rx261)を選択して右クリックします。その後「プロパティ (R)」をクリックします。※下図は RX231 グループを使用する場合の例です。

workspace - e ² studi	io		
ファイル(F) 編集(E) ソ-	-ス(S)	リファクタリング(T) ナビゲート(N) 核 (B) マロロロ update firmware	8索(A) プロジェクト(P) Re nx231 Har > 🌼 🛛 🔫 👻
■ プロジェクト・エクスプロー	5- ×		E % 7 % - C
> 😂 firmware_rx231			
> 🥴 update_firmware	_rx23	1 [HardwareDebug]	
		新規(N) 次へジャンプ(I)	>
		新規ウィンドウで開く(N) 表示方法(W)	Alt+シフト+W >
		コピー(C) 貼り付け(P) 削除(D) ソース 移動(V) 名前を変更(M)	Ctrl+C Ctrl+V 削除 >
	28	インボート(I) エクスポート(O)	
	£	プロジェクトのビルド(B) プロジェクトをクリーンにする 更新(F) プロジェクトを閉じる(S) 無関係なプロジェクトを閉じる(U)	F5
		ビルド・ターゲット インデックス ビルド構成	>
	0 *	実行(R) デパッグ(D) チーム(E) 比較対象(A) ローカル履歴から復元(Y) MISRA-C C/C++ Project Settings	> > > Ctrl+Alt+P
	*	Renesas C/C++ Project Settings C/C++ コード解析を実行 System Explorer Command Prompt Validate クリック	します。
		構成 Source	>
		プロパティ(R)	Alt+Enter



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



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

± ^ /	- 赤 西 レ チ ー ピー パーノ	ちし のじょ じ 肌 白	$(\neg $			(1/0)
77 . S 1	一変 史 し に フロンチィ	ノトリアルト設定	(ノアームリエアア	ッフテート	N J U V T /J N J	(1/2)
1.0.1	父父したくにくてく			/ / /	/ / / / / / /	\ ·/ _/

項目	変更内容		説明
Compiler	「インクルード・ファイル・ディレ	各 FIT モ 注	ジュールで設定が必要なインクルードパス
- ソース	クトリ」にインクルードパスを追加	を追加しる	ます。
	する。	「スマー	トコンフィグレータ」を使用して、各 FIT
		エジュール	- ニシントシン ション にのからてい ロー・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
		<u> </u>	
		サンフル.	ノロクラムのイングルートハスを追加しま
		す。本フロ	ロジェクトでは、
		src/src_up	pdate_firmware,
		src/src_up	pdate_firmware/r_fw_up_rx,
		src/src_up	pdate_firmware/r_xmodem を追加してい
		ます。	
		<設定例>	
		インクルード・ファ	アイルを検索するフォルダ (-include) 🛛 🖓 🖓 🖏 😓
		"\${workspace	loc:/\$(ProiName)/src/src update firmware)"
		"\${workspace_l	loc:/\${ProjName}/src/src_update_firmware/r_fw_up_rx}"
		"\${workspace_l	loc:/\${ProjName}/src/src_update_firmware/r_xmodem}"
Linker	RAM 領域に、RPFRAM セクショ	サンプルこ	プログラムが使用する RAM 領域を設定し
- セクション	ン、RPFW_UP_RAM セクションを	ます。	
	追加する。	<設定例>	
		セクション・ビューア	P-:
		アドレス	セクション名
		0x0000004	SU
			SI
			8_1
			B_2
			R_2
			B
			RDERAM
			RPFW_UP_RAM



項目	変更内容		説明
Linker	ROM 領域に、FW_UP_VER セクショ	サンプル	プログラムを配置する ROM 領域を設
- セクション	ン、FW_UP_COMPLETE セクション	定します。	。定数データをスタートアッププログ
	を追加する。	ラム保護	機能の代替領域より上に配置して、P*
		セクショ	ンをスタートアッププログラム保護機
	FW LIP VER セクションの先頭アドレ	能のデフ	オルト領域に配置します。
	スを、	<設定例>	
	RX130 グループの場合 0xFFFF6800	アドレス	セクション名
	RX140 グループ、RX231 グループお	0xFFFF6000	FW_UP_VER
	よび RX261 グループの場合	UXFFFF6008	C 2
	0vFFFF6000 に設定する		c
			C\$*
			D*
	C_1 セクションの先頭アドレスを、		L
	RX130 グループの場合 0xFFFF6808	0xFFFF6FF0	FW_UP_COMPLETE
	RX140 グループ、RX231 グループお	0xFFFFC000	P*
	よび RX261 グループの場合	0xFFFFFF80 0xFFFFFFFC	EXCEPTVECT
	0vFFFF6008 に設定する		
	FW_UP_COMPLETE セクションの先		
	頭アドレスを、		
	RX130 グループの場合 0xFFFF73F0		
	RX140 グループ、RX231 グループお		
	よび RX261 グループの場合		
	0xFFFF6FF0に設定する。		
	D*セクションの生産マドレスを		
	UXFFFFCUUU に設定する。		
Linker	ROM から RAM ヘマップするセクショ	サンプル	ブログラムは、コードフラッシュメモ
- セクション	ンに PFRAM=RPFRAM,	リの書き	換えを RAM 上で実行します。そのた
- シンボル・	PFW_UP_RAM=RPFW_UP_RAM を	め ROM カ	から RAM にマップする設定を追加しま
ファイル	追加する。	す。	
		-	

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



項目	変更内容		説明	
Compiler	「インクルード・ファイル・ディレ	各 FIT モ	ジュールで設定が必要な	ぶインクルードパ
- ソース	クトリ」にインクルードパスを追加	スを追加し	します。	
	する。	「スマー	トコンフィグレータ」を	を使用して、各
		FIT モジョ	ıールを組み込む場合、	自動で設定され
		ます。		
		サンプル	プログラムのインクル-	- ドパスを追加し
		ます。本て	プロジェクトでは、src/	/src_firmware,
		src/src_fir	mware/r_fw_up_rx を追	追加しています。
		<設定例>		
		インクルード・ファイ	イルを検索するフォルダ (-include)	🗐 🜒 🗑 🖓 归
		"\${workspace_lo "\${workspace_lo	oc:/\${ProjName}/src/src_firmware}" oc:/\${ProjName}/src/src_firmware/r_fw_u	up_rx)"
Linker	RAM 領域に、RPFRAM セクショ	サンプル	プログラムが使用する F	RAM 領域を設定
- セクション	ン、RPFW_UP_RAM セクションを	します。		
	追加する。	<設定例>		
		セクション・ビューア	-:	-
		アドレス	セクション名	
		0x0000004	SI	
			B_1	
			R_1 B_2	
			R_2	
			B	
			RPFRAM	
			RPFW_UP_RAM	
- セクション	ROM 領域に配置するセクションの 先頭アドレスを、 RX130 グループの場合 0xFFFE0000 RX140 グループの場合 0xFFFC0000 RX231 グループおよび RX261 グ	ッフラル します。 ⁺ メモリの <設定例>	ウロックムを配置する「 サンプルプログラムを= 先頭アドレスに配置しま	KOM 領域を設定 コードフラッシュ ます。
	ループの場合 0xFFF80000 に設定す		C	
	る。		C\$*	
			W*	
			L D*	
		0xFFFFF80 0xFFFFFFC	EXCEPTVECT RESETVECT	
		【注】 フォ する 置す RX FFF RX グル FFF	ァームウェアプログラム ることはできないため、こ されないようにしてくださ 130 グループの場合 FF_6800h 番地~FFFF_B 140 グループ、RX231 グ, レープの場合 FF_6000h 番地~FFFF_B	は以下の領域を使用 の領域にデータが配 い。 FFFh 番地 ループおよび RX261 FFFh 番地
Linker	ROM から RAM ヘマップするセク	サンプル	プログラムは、スタート	トアッププログラ
- セクション	ションに PFRAM=RPFRAM,	ム保護機能	能の領域入れ替えを RA	M上で実行しま
- シンボル・	PFW_UP_RAM=RPFW_UP_RAM	す。その	ため ROM から RAM に	マップする設定を
ファイル	を追加する。	追加します	す。	

表 3.3 変更したプロジェクトのビルド設定(ファームウェアプロジェクト)



4. 動作確認

- 4.1 プロジェクトのビルド 以下の手順に従い、プロジェクトをビルドしてロードモジュールを作成します。
- ビルドするプロジェクト、RX130 グループの場合(update_firmware_rx130 または firmware_rx130) を、RX140 グループの場合(update_firmware_rx140 または firmware_rx140) を、RX231 グループの 場合(update_firmware_rx231 または firmware_rx231) を、RX261 グループの場合 (update_firmware_rx261 または firmware_rx261) をクリックします。※下図は RX231 グループを使 用する場合の例です。



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

プロ	ジェクト(P)	Renesas Views	実行(R)	ウィンドウ(W)	
	ブロジェクト プロジェクト Open Syr	〜を開く(E) ∽を閉じる(S) hergy Configurati	on	1	
	すべてビル ビルド構成	F(A) t	С	trl+Alt+B	
	プロジェクト	ーのビルド(B)		Ctrl+B	
	ワーキング・ クリーン(N 自動的に)	・セットのビルド(W)) ビルド(M)		>	- / / / / しより。
	ビルド・ター	-ゲット		>	
02	C/C++ イ すべての依 Change I Change T	ンデックス(i) 存関係を更新 Device foolchain Version	í.	Alt+D	
3	C/C++ Pr プロパティ(roject Settings P)	C	trl+Alt+P	

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

Renesas Optimizing Linker Completed 'Finished building target:'
'Build complete.'
13:11:24 ビルドが完了しました(所要時間 52s.846ms)



4.2 デバッグの準備

4.2.1 機器の準備

デバッグを開始する前に、評価ボードを準備します。表 4.1 に必要な機器の一覧を、図 4.1 (RX130/ RX140/RX231)、図 4.2 (RX261) にデバッグ構成図を示します。

No.	機器	補足
1	開発 PC	開発する PC です。
2	評価ボード	RX261 の場合は E2 Lite が内蔵されているため、
	(Renesas Starter Kit for RX130 / RX140 /	外付けの E2 Lite は必要ありません。
	RX231、RX261 MCU グループ評価キット)	
3	ホスト PC	開発 PC でも代用可能です。
	— XMODEM/SUM 転送プロトコルに対応	
	したシリアル通信ソフトウェア	
4	USB ケーブル(mini-B タイプ)	Renesas Starter Kit for RX130/RX140/RX231
		のシリアル人出力信号は USB シリアル変換され
		ており、ホスト PC と USB 接続することにより に担 COM ポートトーズ体田士スニトギズキナ
		仮想 COM ホートとして使用することができま す。
	USB ケーブル(Micro-B タイプ)	RX261 MCU グループ評価キットのシリアル入出
		カ信号は USB シリアル変換されており、ホスト
		PC と USB 接続することにより仮想 COM ポート
		として使用することができます。

表 4.1 機器一覧



図 4.1 デバッグ構成図 (RX130/RX140/RX231 の場合)





図 4.2 デバッグ構成図(RX261 の場合)

4.2.2 ホスト PC の設定

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

表 4.2 通信仕様

項目	内容
通信方式	調歩同期式通信
ビットレート	115200bps
データ長	8ビット
パリティ	なし
ストップビット	1ビット
フロー制御	なし



4.3 プロジェクトのデバッグ

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

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

00 the	Renesas Debug Tools 実行(R) デバッグ(D) 実行履歴(T)	> Ctrl+F11 F11 >
*	実行構成(N) 実行構成(N) デバッグ履歴(H) デバッグ(G) デバッグの構成(B)	·クリックします。
	ブレークポイント型(R)	>
0	ブレークポイントの切り替え(K) 行ブレークポイントの切り替え(I)	Ctrl+シフト+B
×× ∞ ∞ ∞	監視ポイントの切り替え(W) メソッド・ブレークボイントの切り替え(M) すべてのブレークポイントを初除(V) すべてのブレークポイントを削除(V)	



2. RX130 グループの場合:

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

RX140 グループの場合:

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

RX231 グループの場合:

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

RX261 グループの場合:

「Renesas GDB Hardware Debugging」の「update_firmware_rx261 HardwareDebug」をクリックします。

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





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

Aborda. 8t8, abdy5f3 Cfcrovback Cfc	■ デバッグ構成				
 ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	成の作成、管理、および実行	「デバ	ッグ・ツール設定」	」をクリックします。	Ŕ
ペルクスカ ロイビ+ アブリケーション ロイビ+ アブリケーション ロイビ+ アブリケーション ロイビ+ アブリケーション ロイビ+ アブリケーション ロイビ+ アブリケーション ロクロ ロク	2 🕫 🗎 🗶 🖻 🏹 🛨	名前(N): update firmware rx231 HardwareDebug			
□ C(C++ アブリケ-ション □ C(C++ リモ, ト・アブリケ-ション □ KASE Sript □ G08 OpenOCD Debugging □ G08 Jimulator Debugging (RH850) □ C08 A (+ Sr) 7 / T(x * Y / Z) □ Ava アブリケ-ション □ Java アブレット □ Ava アブリケ-ション □ Java アブレット □ Ava アブリケ-ション □ Ava PT Av	ィルタ入力	■ メイン Startup □ 共通(0 きょ)-7			
GDB OpenOCD Debugging GDB Simulator Debugging (RH85) GDB Simulator Debugging GDB Simulator Debugging GDB Approve F77ルト HO 27*/L4646使用 Java アプリケ・ション Java アプリケ・ション B Nava アプリケ・ション GDB Simulator Debugging C Infirmware_rx231 HardwareDebug GDB Fetrings Connection Settings Sfupport_area_Joc) C Dig Ate Firmware_rx231 HardwareDebug C Dig Ate Firmware_rx231 HardwareDebug C J UT-FikeLU 世やトする LUN S Reneass GDB Hardware Debug (RK, RL78) ABD 70-7 S Reneass GDB Hardware Debug LD 7*/L92 C J UT-FikeLU 世やトする LUN S Reneass GDB Hardware Debug LD 7*/L92 C J UT-FikeLU 世やトする LUN S Reneass GDB Hardware Debug LD 7*/L92 C J UT-FikeLU 世やトする LUN S Reneass GDB Hardware Debug LD 7*/L92 C J UT-FikeLU 世やトする LUN S Reneass GDB Hardware Debug LD 7*/L92 C D 7+ A MB BhD7PLX Contoout C D 7- P RAM BhD7PLX Contoout D 7- P RAM BhD7PLX Contoout D 7- P RAM BhD7PLX Contoout C 1- 7*- TD75/L92 Leting Bbb2gE/1975 LUN2 <td< td=""><td> C C/C++ アブリケーション C C/C++ リモート・アブリケーション EASE Script </td><td>Debug hardware: E2 Lite (RX) V Target Dev</td><td>rice: R5F52318</td><td></td><td></td></td<>	 C C/C++ アブリケーション C C/C++ リモート・アブリケーション EASE Script 	Debug hardware: E2 Lite (RX) V Target Dev	rice: R5F52318		
● IO アフル・ドウギンダ ③ Java アブレット デブルド IO ファイルをを使用 ばい ● IO デブルド IO ファイルを使用 ばい ● II ava アブレット デブルド IO ファイルを使用 ばい ● II movare ry231 Hardware Debugging ビリアイルを使用 ばい ・ ● II movare ry231 Hardware Debugging ビリディアン リトレ・エンディアン ・ ・ ③ Update firmware ry231 Hardware Debugging (RX, RL78) リブート Java アブリケーション ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ <td>GDB OpenOCD Debugging</td> <td>GDB Settings Connection Settings デバッグ・ツール設</td> <td>定</td> <td></td> <td></td>	GDB OpenOCD Debugging	GDB Settings Connection Settings デバッグ・ツール設	定		
GOB ハードウエア・デパッチング ゴ Java アブリケーション Java アブリケーション ジョンa アブリケーション B Reness GOB Hardware Debug レ C Introvace nx231 HardwareDebug レッチ化 C Introvace nx231 HardwareDebug レッチイアン D U T-1- Java アブリケーション メモリー C Introvace nx231 HardwareDebug レッチイアン D U T-1- Java アブリケーション メモリー A 都がガルーブ レいス ウ ワート Java アブリケーション メモリー A 都がガルーブ レいス ウ ロケラム RAM 戦闘がドレス 00 ウ ーク RAM 戦闘がドレス 00 ウ ーク RAM 戦闘がドレス 00 ウ ーグ・フランションを書き換えるブログラムをデパッグする レいス マーク RAM 戦闘がドレス 00 ウ ーグ・ブログラム保険戦艦設定 レいス マーグ・ブログラム保険地撃たしス 00 ローザ・ブログラム保険が野ドレス 00 ローザ・ブログラム保険が野ドレス 00 ローザ・ブログラム保険が野ドレス 00 ローザ・ブログラム保険が野ドレス 00 ワーク RAM 戦闘がアドレス 00 マーク RAM 戦闘がアドレス 00 ローザ・ブログラム レック 0230 マーク RAM 戦闘がアドレス 0230 マーク RAM 戦闘がアドレス 0230 アシアレードを有効にする 10.0 </td <td>GDB Simulator Debugging (RH850)</td> <td>~ IO</td> <td></td> <td></td> <td>^</td>	GDB Simulator Debugging (RH850)	~ IO			^
II Java アブリケ-ション IO ファイル名 S[support_area_loc) II Java アブリケ-ション III Java アブリケ-ション III Java アブリケ-ション Reneass GBB HardwareDebug III Java アブリケ-ション III Juva アブリケ-ション Reneass Simulator Debugging (RX, RL78) III Juva アブリケ-ション III Juva アブリケ-ション 電 起転ヴルーブ JII Juva アブリケーション III Juva アブリケーション 電 起転ヴルーブ JIII Juva アブリケーション IIII Juva アブリケーション マーク RAM 開始アドレス Ox1000 IIII Juva Privide Tili Juva マーク RAM 開始アドレス Ox1000 IIII Juva マーチーブログラム条げ形に関数を実行する IIVIス IIVIス マーゲーブログラム条げ形に関数を実行する IIVIス IIVIス マーク RAM 開始アドレス Ox1000 IIII Juva ローザー・ブログラム条げ形に関数を実行する IIVIス IIVIス マーゲーアログラム条げ市に関数を実行する IIVIス IIVIス マーク RAM 開始アドレス Ox100 IIVIス ワーク RAM 開始アドレス Ox100 IIVIス マーゲーアログラム集中に開数を実行する IIVIス IIVIス マーゲーアンクタン IIVIス IIVIス マーク RIMの開始アドレス Ox100 IIVIス マークシス IIVIII IIV	GDB ハードウェア・デバッギング	デフォルト IO ファイル名を使用	はい		~
I Java アブレット マー& ② Renesas GDB Hardware Debugging グウンロード後にリセットする は1、 ③ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ③ Renesas Simulator Debugging (RX, RL78) メモリー III. ソディアン ④ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ⑤ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ③ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ⑤ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ③ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ⑤ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ④ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ⑤ update_firmware_rx231 HardwareDebug III. ソディアン リトレ・エンディアン ⑤ update_firmware_rx231 HardwareDebug III. リー III. リー ○ up	🎞 Java アプリケーション	IO ファイル名	\${support_area_loc}		
	🗾 Java アプレット	✓ 一般			
● firmware_rx231 HardwareDebug > メモリー ● update_firmware_rx231 HardwareDebug レブイアン ● Renesas Simulator Debugging (RX, RL73) レブイアン ● リロート Java アブリケーション 小部スモリーの上書き ● レリー・ゴンダイアン 00 クーク RAM 開始アドレス 0x1000 ワーク RAM 開始アドレス 0x1000 ワーク RAM 開始アドレス 0x1000 マーク RAM 開始アドレス 0x500 「はい」に設定します。 ・ シスオム 小蔵式ログラムを打いグラムを打いグラムを打いグラムを打いグラムを打いグラムを打いグラムを打い使する 内蔵ブーク・フラッシュを書き換えるブログラムを打いグラムを打いグラムを打いグラムを打いグウォを ・ マーク・・プログラム案行前に関数を発行する いいえ マーク・・プログラム保護設定 - ユーザ・・プログラム条行前に関数を発行する いいえ シスオイ 0x0 ウーン RAM 開始アドレス 0x0 マーク RAM 開始2アドレス 0x0 マーク・・プログラム条行前に関数を発行する いいえ マーク RAM 開始2アドレス 0x0 ローク・アログラム保護設定 - ローク・アログラム保護設定 - ローク・アログラム条行前に関数を発行する いいえ マーク・アログラム条行前に関数を発行する いいえ ● サーク・アログラム条付前に目前の上述を有効にする いいえ ● ウンロードを有効にする いいえ ● ワンロードを有効にする いいえ ● ローク・クロクトと一致 ●	Renesas GDB Hardware Debugging	ダウンロード後にリセットする	はい		~
E update_firmware_nx231 HardwareDebug エンディアン リトル・エンディアン E Renesas Simulator Debugging (RX, RL78) メモリー書き込み時のペリファイ いいえ E リビー-1 Java アブリケーション 内部 ブラッシュメモリーの上書き (0) タ 部がラッシュメモリーの上書き (0) (0) フーク RAM 開始アドレス 0x1000 「はい」」に設定します。 マンデム の素面のグリクション (0) 内部 ブラッシュメモリーの上書き (0) (0) クーク RAM 開始アドレス 0x1000 「はい」」に設定します。 マンデム 小数ボーク・フラッシュを書き換えるブログラムをデバッグする にい、 (1) マンデム 小数ボーク・フラッシュを書き換えるブログラムをデバッグする にい、 (1) マンデム 小数ボーク・フラッシュを書き換えるブログラムをデバッグする にい、 (1) マンデム 小数ボーク・フラッシュを書き換えるブログラムをデバッグする にい、ス (1) マンク RAM 開始アドレス (1) (1) ローザー・ブログラム 保上時に開放を実行する (1) (1) マーク RAM 開始アドレス (1) (1) マーク RAM 開始アドレス (1) (1) (1) マーク RAM 開始アドレス (1) (1) (1) マーク RAM 開始アドレス (1) (1) (1) クリンロードを有効にする (1) (1) (1) タウンロードを有効にする (1) (1) (1) </td <td>firmware_rx231 HardwareDebug</td> <td>◇ メモリー</td> <td></td> <td></td> <td></td>	firmware_rx231 HardwareDebug	◇ メモリー			
CI Renesas Simulator Debugging (RX, RL78) CJ リニー・Java アブリケーション 本部ガルーブ 本部ガルーブ ウボスモリー病域 OI ウボスモリーの上書き OI ウボスモリーの上書き OI ウボスモリンカン ウボスログラムROMを書き換えるプログラムをデバッグする レンス ウボスログラムステレカボンス ウボスログラムステノカッシュを書き換えるプログラムをデバッグする レンス マンステム ウボスログラムステノカッシュ ロボーブログラム家に有能国数を実行する レンス シンスキン ロボーグの開設の開始のドレス ウボンログラムション ウボンログラム ロボンジョン ウボリンログシン レンス ウボンログシン ログリンログシン ログシン マン アン グウンロードを有効にする	update_firmware_rx231 HardwareDebug	エンディアン	リトル・エンディアン		~
PUT-1-Java アプリケーション 内部フラッシュメモリーの上書き [0] ハ部メモリー領域 [0] ワーク RAM 開始27ドレス の、1000 ワーク RAM サイズ (Bytes) マンステム 内蔵プログラムをアパッグする はい ハ蔵データ・フラッシュを書き換えるプログラムをデパッグする はい ハ蔵データ・フラッシュを書き換えるプログラムをデパッグする いいえ マンステム マンステム 内蔵プログラムROMを書き換えるプログラムをデパッグする はい マンステム マンス マ	Renesas Simulator Debugging (RX, RL78)	メモリー書き込み時のペリファイ	いいえ		~
◎ 起動グループ 今部メモリー領域 [0] ワーク RAM 開始アドレス 0x1000 ワーク RAM 開始アドレス 0x500 「はい」に設定します。 システム 内蔵プレクランROMを書き換えるプログラムをデバッグする 内蔵データ・フラッシュを書き換えるプログラムをデバッグする レいえ ンサー・プログラム実行前に関数を実行する レいえ ユーザー・プログラム実行前に関数を実行する レいえ マーク RAM 開始アドレス のx0 ユーザー・プログラム実行方前に関数を実行する いえ マーク RAM 開始アドレス のx0 マーク RAM 開始アドレス のx0 マーク RAM 開始アドレス のx10 マーク RAM 開始アドレス のx10 ローザー・プログラム集合制 リロードを有効にする 前回保管した状態に戻す(0)	🖳 リモート Java アプリケーション	内部フラッシュメモリーの上書き	[0]		
ワーク RAM 開始アドレス 0x1000 「はい」に設定します。 ワーク RAM サイズ (Bytes) 0x500 「はい」に設定します。 ク蔵プログラムをデバッグする しい ・ 内蔵プーク・フラッシュを書き換えるプログラムをデバッグする しい ・ ウボデーク・フラッシュを書き換えるプログラムをデバッグする いいえ ・ ン Start / Stop 機能設定 ・ ・ ユーザー・プログラム実行前に開数を実行する いいえ ・ まtart 開数の開始アドレス 0x0 ・ ワーク RAM 開始アドレス 0x0 ・ マーク RAM 開始アドレス 0x0 ・ ワーク RAM 開始アドレス 0x0 ・ マーク RAM 開始アドレス 0x0 ・ マウンロードを有効にする いいえ ・ 第回の含ち14 項目がフィルターに一致 100保管した状態に戻す(2) 連用の	■ 起動グループ	外部メモリー領域	[0]		
ワーク RAM サイズ (Bytes) 0x500 「はい」に設定します。 マジステム 内蔵プログラムROMを書き換えるプログラムをデバッグする い 内蔵プーク・フラッシュを書き換えるプログラムをデバッグする い マ Start / Stop 機能設定		ワーク RAM 開始アドレス	0x1000		
マシステム 内蔵プログラムROMを書き換えるプログラムをデバッグする【はい 内蔵データンフッシュを書き換えるプログラムをデバッグする【はい 内蔵データンフッシュを書き換えるプログラムをデバッグするしいえ マンStart / Stop 機能設定 ユーザー・プログラム保行前に関数を実行する レいえ start 関数の開始アドレス のx0 ユーザー・プログラム保行する リンステム マークークRAM 開始アドレス のx1 (パイト) のx230 マーク RAM 開始アドレス のがけない ワーク RAM 開始アドレス のx1 (パイト) のx230 マ 外部フラッシュ ダウンロードを有効にする 前回保管した状態に戻すい 適用()		ワーク RAM サイズ (Bytes)	0x500	「はい」に設定します。	
内蔵プログラムROMを書き換えるプログラムをデバッグする はい 内蔵データ・フラッシュを書き換えるプログラムをデバッグする いいえ × Start / Stop 機能設定 ユーザー・プログラム実行前に関数を実行する いいえ ***********************************		✓ システム			
内蔵データ・フラッシュを書き換えるプログラムをデバッグする いいえ ◆ Start / Stop 機能設定 ユーザー・ブログラム実行的に関数を実行する いいえ start 関数の開始アドレス 0x0 ユーザー・ブログラム停止時に関数を実行する いいえ いいえ いいえ いいえ の0 ワーク RAM 開始アドレス 0xfdd0 ワーク RAM サイズ (パイト) 0x230 ◆ 外部フラッシュ ダウンロードを有効にする いいえ 前回保管した状態に戻す() 適用()		内蔵ブログラムROMを書き換えるブログラムをデバック	げする はい		~
 		内蔵データ・フラッシュを書き換えるプログラムをデバック	グする いいえ		~
ユーザー・ブログラム実行前に関数を実行する いいえ start 関数の開始アドレス 0x0 ユーザー・ブログラム停止時に関数を実行する いいえ stop 関数の開始アドレス 0x0 ワーク RAM 開始アドレス 0x1 ワーク RAM 開始アドレス 0x1 マーク RAM 開始アドレス 0x230 マクラク FAM 開始アドレス 0x230 マクリンロードを有効にする いいえ		✓ Start / Stop 機能設定			
start 関数の開始アドレス 0x0 ユーザー・ブログラム停止時に関数を実行する いいえ stop 関数の開始アドレス 0x0 ワーク RAM 開始アドレス 0xfdd0 ワーク RAM 開始アドレス 0x230 V 外部フラッシュ ダウンロードを有効にする ダウンロードを有効にする いいえ		ユーザー・プログラム実行前に関数を実行する	いいえ		~
ユーザー・ブログラム停止時に関数を実行する いいえ stop 関数の開始アドレス 0x0 ワーク RAM 開始アドレス 0xfdd0 ワーク RAM 開始アドレス 0xfdd0 ワーク RAM サイズ (パイト) 0x230 ・ 外部フラッシュ		start 関数の開始アドレス	0x0		
stop 関数の開始アドレス 0x0 ワーク RAM 開始アドレス 0xfdd0 ワーク RAM 開始アドレス 0xfdd0 マーク RAM サイズ (パイト) 0x230 ・ 外部フラッシュ - ダウンロードを有効にする いいえ		ユーザー・プログラム停止時に関数を実行する	いいえ		~
ワーク RAM 開始アドレス Oxfdd0 ワーク RAM 開始アドレス Oxfdd0 ワーク RAM サイズ (パイト) Ox230 マ 外部フラッシュ レいえ		stop 関数の開始アドレス	0x0		
ワーク RAM サイズ (パイト) 0x230 マ 外部フラッシュ 0x230 ダウンロードを有効にする いいえ		ワーク RAM 開始アドレス	0xfdd0		
単 外部フラッシュ ダウンロードを有効にする いいえ 期目のうち 14 項目がフィルターに一致 前回保管した状態に戻す(V) 適用(V)		ワーク RAM サイズ (バイト)	0x230		
ダウンロードを有効にする いいえ 夏目のうち 14 項目がフィルターに一致 前回保管した状態に戻す() 適用()		◇ 外部フラッシュ			
頁目のうち 14 項目がフィルターに一致 前回保管した状態に戻す(<u>い</u>) 適用()		ダウンロードを有効にする	いいえ		~ 、
同日のうち14項目がフィルターに一致				前回保管した状態に同された	·演田/\/0
	頁目のうち 14 項目がフィルターに一致			別凹体質した状態に戻り(⊻)	週冊(1)
				デバッグ(D)	問じる



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

📴 テバッグ構成		_	
構成の作成、管理、および実行			The second
			1
]
	名前(N): update_firmware_rx231 HardwareDebug		
7ィルタ入力	📄 メイン 🌾 Debugger 🕨 Startup 🔲 共通(C) 🦻 ソース		
 E C/C++ アプリケーション E C/C++ リモート・アプリケーション EASE Script 	Debug hardware: E2 Lite (RX) V Target Device:	R5F52318	
GDB OpenOCD Debugging	GDB Settings Connection Settings デバッグ・ツール設定		
GDB Simulator Debugging (RH850)	✓ 10		^
■ GDB ハードウェア・デバッギング	デフォルト IO ファイル名を使用	はい	~
💶 Java アプリケーション	IO ファイル名	\${support_area_loc}	
🖭 Java アプレット	~ 一般		
✓	ダウンロード後にリセットする		~
firmware_rx231 HardwareDebug	◇ メモリー	クリックします。	
update_firmware_rx231 HardwareDebug	エンディアン	リトル・エンディアン	~
Renesas Simulator Debugging (RX, RL78)	メモリー書き込み時のベリファイ	いいえ	~
🖳 リモート Java アプリケーション	内部フラッシュメモリーの上書き	[0]	
🖡 起動グループ	外部メモリー領域	[0]	
	ワーク RAM 開始アドレス	0x1000	
	ワーク RAM サイズ (Bytes)	0x500	
	~ システム		
	内蔵プログラムROMを書き換えるプログラムをデバッグする	はい	~
	内蔵データ・フラッシュを書き換えるプログラムをデバッグする	いいえ	~
	✓ Start / Stop 機能設定		
	ユーザー・プログラム実行前に関数を実行する	いいえ	~
	start 関数の開始アドレス	0x0	
	ユーザー・プログラム停止時に関数を実行する	いいえ	~
	stop 関数の開始アドレス	0x0	
	ワーク RAM 開始アドレス	0xfdd0	
	ワーク RAM サイズ (バイト)	0x230	
	◇ 外部フラッシュ		
	ダウンロードを有効にする	いいえ	<u> </u>
			·帝田(10)
16 項目のうち 14 項目がフィルターに一致		別四1本官した状態に戻す(⊻))西/刊(<u>Y</u>)
?		デバッグ(D)	閉じる
-			



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

番号	アドレス範囲	^
0	00100000 - 001003FF	
1	00100400 - 001007FF	
2	00100800 - 00100BFF	
3	00100C00 - 00100FFF	
4	00101000 - 001013FF	
5	00101400 - 001017FF	
6	00101800 - 00101BFF	
7	00101C00 - 00101FFF	
8	FFF80000 - FFF807FF	
9	FFF80800 - FFF80FFF	
10	FFF81000 - FFF クリックしま	す。 🗸
すべて選択	すべて選択解除 🗌 Show Only Se	lected
選択されたブ 未選択のプロ	ロックは消去せずに上書きします。 ックは、消去してか「OK」をクリックし	ます。



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

成の作成、管理、δよ 0実行			X	
1 🖻 🕫 🔛 🗶 🖻 🏹 🗸	名前(N): update_firmware_rx231 HardwareDebug			
ィルタ入力	◎ メイン 🏠 Debugger 🔈 Startup 🔲 共通(C) 転ソーフ			
 C C/C++ アブリケーション C C/C++ リモート・アブリケーション EASE Script 	Debug hardware: E2 Lite (RX) V Target Device:	R5F52318		
GDB OpenOCD Debugging	GDB Settings Connection Settings デバッグ・ツール設定			
GDB Simulator Debugging (RH850)	~ IO		^	
GDB ハードウェア・デバッギング	デフォルト IO ファイル名を使用	はい	~	
💶 Java アプリケーション	IO ファイル名	\${support_area_loc}		
🜌 Java アプレット	~ 一般			
Renesas GDB Hardware Debugging	ダウンロード後にリセットする	はい	~	
firmware_rx231 HardwareDebug	◇ メモリー			
update_firmware_rx231 HardwareDebug	エンディアン	リトル・エンディアン	~	
Renesas Simulator Debugging (RX, RL78)	メモリー書き込み時のベリファイ	いいえ	~	
🖳 リエート Java アプリケーション	内部フラッシュメモリーの上書き	[0]		
■ 起動グループ	外部メモリー領域	[0]		
	ワーク RAM 開始アドレス	0x1000		
	ワーク RAM サイズ (Bytes)	0x500		
	~ システム			
	内蔵プログラムROMを書き換えるプログラムをデバッグする	5 はい	~	
	内蔵データ・フラッシュを書き換えるプログラムをデバッグする	5 いいえ	~	
	✓ Start / Stop 機能設定			
	ユーザー・プログラム実行前に関数を実行する	いいえ	~	
	start 関数の開始アドレス	0x0		
	ユーザー・プログラム停止時に関数を実行する	いいえ	~	
	stop 関数の開始アドレス	0x0		
	ワーク RAM 開始アドレス	0xfdd0		
	ワーク RAM サイズ (パイト)	0x230		
	◇ 外部フラッシュ			
	ダウンロードを有効にする	いいえ	~ ~	
		七同原語しも小能に同一人の	*******	
項目のうち 14 項目がフィルターに一致	「デバッグ(D)」をク	リックします。	週用(<u>Y</u>)	
)		デバッグ(D)	閉じる	

以下のメッセージが表示されたら、「切り替え(S)」をクリックします。

2	This kind of launch is configured to ope	en the デバッグ perspective when it suspends.
	このデバッグ・パースペクティブは、アブリケーショ は、デバッグ・スタック、変数、およびブレークオ	ョンのデバッグをサポートするように設計されています。これに ポイント管理を表示するビューが組み込まれています。
	Switch to this perspective?	
		「切り替え(S)」をクリックします。
	にこの設定を使用する(R)	



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

		a a second s					101404	ictt win
☆× 日本 ▶Ⅰ**	C @ resetprg.c 11		** O	☆ 変数 № ブレーク 薬	モジュー 🏷 プロジェ	今日式 チイバント ミ	R Periph	🗂 IO Re 🦈
update_firmware_rx231 HardwareDebug [Renesas GDB Hardware Debugging]	141 ttttcc61	void PowerON_Reset_PC(void)	^			XXXB	B PC: 0/4	OA: 0/2 86
update_firmware_nzs1x[1] [cores 0]	143	/* Stack pointers are setup prior to calling this function - see comments above */		91プ 「ドレース開始	アドレス	データ	/12	917- 70
PowerON_Reset_PC() at resetprg.c141 0xfffccb1	145	/* Initialise the MCU processor word and Exception Table Register */		□ ■ トレース停止				
E Reserve CDR exerve (Head)	146	<pre>#ifRENESAS_VERSION >= 0x01010000</pre>	and 10	□ * トレース・レコ				
	147 ttttccbt	<pre>set_intb((void *)_sectop("SYLL1"));</pre>		□ 5 イベント・プレ				
	149	Noise		□ Ø 91マ-開始				
	150 151	<pre>set_intb((unsigned long)sectop("C\$VECT")); set_extb((unsigned long)_sectop("EXCEPTVECT"));</pre>		□ 🔊 91マー停止				
	152 153	#endif						
	154	/* Initialize FPSW for floating-point operations */						
	156 157 158	<pre>#storwuz #define FPU_ROUND 0x00000001 /* Let FPSW R#bits=01 (round to zero) */ #delse</pre>						
	159	<pre>#define FPU_ROUND 0x00000000 /* Let FPSW RMbits=00 (round to nearest) */ Mendif</pre>						
	161	##ifdefDOFF						
	162	<pre>#define FPU_DENOM 0x00000100 /* Let FPSW DNbit=1 (denormal as zero) */</pre>						
	163 164 165	<pre>#define FPU_DENOM 0x000000000 /* Let FPSW DMbit=0 (denormal as is) */ #endif</pre>						
	166 167 ffffcc85	<pre>set_fpsw(FPSM_init FPU_ROUND FPU_DENOM);</pre>						
	168	/* Switch to bish read exercise */						
	170 ffffcc88	operating requency set():						
	171 ffffcc8b 172	usb_lpc_clock_source_select();						
	173	/* If the warm start Pre C runtime callback is enabled, then call it. */						
	174 175 176	<pre>### 95P_CFG_USEM_WARMC_START_CALLBACK_PRE_INIT_EMABLED == 1 BSP_CFG_USEM_WARMC_START_PRE_C_FUNCTION(); #endif</pre>						
	177							
	178 179 ffffcc8e	<pre>/" Initialize C runtime environment "/ _INITSCT();</pre>						
	180	/* If the warm start Post C runtime callback is enabled, then call it. */		2				
	182	- #1# RSP CFG USER WARM START CALLRACK POST INITC FNARLED == 1	¥		ererfi			
		¢	,	JUJIJE MECHA	/////-r	-		
	1 3 y-1 1 1 1	レジスター 11 問題 W スマート・フラウザー M Debugger Console () デバッグ・シェル () メモリー				a se de los 2 3 de	व्यक्ता ल ५	
	EEWE28X d11	231 HardwareDeolog (keriesas Gub Hardware Deologging) 2 Ad . 06. 063						
	Communi.dll	3.02.00.000						
	Emulator Infor	nation: nd Revision EDITE Rev A						
	User Vcc	3.33545 V 5.47568 V						
	ターゲット接続終了 GDB: 58914	2.97.098.9				更新	が利用できます	,
	ターゲット接続状況	- 0K				97	ウェアで更新か	「利用できます。

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

ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) Re	nesas Views 実行(R)	ウィンドウ(W) ヘルプ(H)
🔏 💿 🏘 デバッグ(B) 🗸 🖬 update_firmware_rx231 Har Y 🔅 📑 🕶	🗟 🕼 🕲 🕶 🗞 🕶	8 1 🖞 🖉 🧶 🛠 🔹 🖉 📲 🔍 🕒 🛛 🖷 🕅 🕉 🖉 😵
★ デバッグ × □ ※ i+ 8 □ □	i resetprg.c ⊠	
 Cupdate_firmware_rx231 HardwareDebug [Renesas GDB Hardware Debugging] 20 update_firmware_rx231.x [1] [cores: 0] 21 Update_firmware_rx231.x [1] [cores: 0] 22 Thread #1 1 (single core) [core: 0] (Suspended : シグナル : SIGTRAP.Trace/bread 23 PowerON_Reset_PC() at resetprg.c:141 0xffffcc61 24 rx-elf-gdb -rx-force-v2 (7.8.2) 24 Renesas GDB server (Host) 	<pre>> 141 ffffcc61 142 143 144 145 146 147 ffffcc6f 148 ffffcc78 149</pre>	<pre>void PowerON Reset PC(vid) {</pre>



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



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

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

RX140 グループの場合:

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

RX231 グループの場合:

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

RX261 グループの場合:

```
RX261 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```


5. アプリケーション概要

5.1 ファームウェアアップデートプログラムの構成

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

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

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



図 5.1 ファームウェアアップデートプログラムのメモリマップ

衣 3.1 上奴ナーツ 限収の伸展	表 5.1	定数テータ管	遉域の構反
-------------------	-------	--------	-------

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



5.2 動作概要

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

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

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

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

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

表 5.2 サンプルプログラムの機能比較

5.2.1 ファームウェアアップデートプログラムの書き込み

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

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



5.2.2 ファームウェアプログラムの書き込み

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

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



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



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



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

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



図 5.4 コードフラッシュメモリ消去(RX231 グループの場合)

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



図 5.5 ファームウェアプログラムの送信(RX231 グループの場合)

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



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



- 6. 4と5の手順を繰り返し、ファームウェアプログラムの全データを書き込みます。
- ファームウェアプログラムの書き込みが終了したのち、ファームウェア起動処理に分岐します。ファームウェアアップデートプログラムは、エクストラ領域の設定によってスタートアッププログラム保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、ソフトウェアリセットを実行します。ファームウェアプログラムが起動します。



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



5.2.3 ファームウェアプログラムの更新

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

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



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



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



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



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



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

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



図 5.11 コードフラッシュメモリ消去(RX231 グループの場合)

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



図 5.12 ファームウェアプログラムの送信(RX231 グループの場合)

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



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



- 7. 5と6の手順を繰り返し、新しいファームウェアプログラムの全データを書き込みます。
- 新しいファームウェアプログラムの書き込みが終了したのち、ファームウェア起動処理に分岐します。 ファームウェアアップデートプログラムは、エクストラ領域の設定によってスタートアッププログラム 保護機能のデフォルト領域と代替領域を永続的に入れ替えたのち、ソフトウェアリセットを実行しま す。更新されたファームウェアプログラムが起動します。



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



5.2.4 ファームウェアアップデートプログラムの更新

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

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



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



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



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



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



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

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



図 5.18 コードフラッシュメモリ消去(RX231 グループの場合)

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



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



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

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



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



- 7. 5と6の手順を繰り返し、新しいファームウェアアップデートプログラムの全データを書き込みます。
- 8. 新しいファームウェアアップデートプログラムの書き込みが終了したのち、定数データ領域2のバージョン情報格納領域2から新しいファームウェアアップデートプログラムのバージョン情報を取得して、書き込み完了情報格納領域2に書き込みます。



図 5.21 書き込み完了情報の書き込み(RX231 グループの場合)



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



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



5.2.5 ファームウェアプログラムの動作

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

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



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



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



図 5.24 メッセージ出力(RX231 グループの場合)



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



図 5.25 コマンド受信(RX231 グループの場合)

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



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



5.2.6 書き込みが失敗した場合の復旧方法

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

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



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

(1) セクション設定を変更

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

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

デバイス	セクション		
	FW_UP_VER	C_1	FW_UP_COMPLETE
RX130 グループ	0xFFFF6800	0xFFFF6808	0xFFFF73F0
RX140 グループ	0xFFFF6000	0xFFFF6008	0xFFF6FF0
RX231 グループ			
RX261 グループ			

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

デバイス	セクション		
	FW_UP_VER	C_1	FW_UP_COMPLETE
RX130 グループ	0xFFFF7400	0xFFFF7408	0xFFFF7FF0
RX140 グループ	0xFFFF7000	0xFFFF7008	0xFFFF7FF0
RX231 グループ			
RX261 グループ			

(2) バージョン情報を変更

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

以下にバージョン情報の設定例を示します。

● Ver3.05 の場合

#define FW UP PROGRAM VERSION (0x0305u)

● Ver10.20 の場合

#define FW UP PROGRAM VERSION (0x1020u)



5.3 ファームウェアアップデートプログラムの概略フローと画面出力

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

5.3.1 メイン処理

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

(1) 概略フロー

図 5.27 にメイン処理の概略フロー示します。



図 5.27 メイン処理の概略フロー



(2) ターミナルソフトウェア画面出力

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

RX130 グループの場合:

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

図 5.28 メイン処理の画面出力(RX130 グループ)

RX140 グループの場合:

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

図 5.29 メイン処理の画面出力(RX140 グループ)

RX231 グループの場合:

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

図 5.30 メイン処理の画面出力(RX231 グループ)

RX261 グループの場合:

RX261 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>

図 5.31 メイン処理の画面出力(RX261 グループ)

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

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

'3'を送信するとファームウェア起動処理に分岐します。

改行コードを送信すると、メニューを再び表示します。

(3) アップデートモード

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

表	5.5	アン	ップデー	トモー	ドの概要
---	-----	----	------	-----	------

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

5.3.2 ファームウェアアップデート処理

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

(1) 概略フロー

図 5.32 にファームウェアアップデート処理の概略フローを示します。





図 5.32 ファームウェアアップデート処理の概略フロー



- (2) ターミナルソフトウェア画面出力
- 1. アップデート確認

ファームウェアアップデート処理は、アップデート確認のため図 5.33 のメッセージを出力します。

Erase flash memory and write firmware (Y/N)? 図 5.33 アップデート確認の画面出力

2. ファイルダウンロード開始

'Y'または'y'を送信すると、コードフラッシュメモリを消去したのち、図 5.34 のメッセージを出力して ファームウェア受信待ち状態になります。ターミナルソフトウェアから XMODEM/SUM プロトコルを使用 して.mot ファイルを送信してください。なお、ターミナルソフトウェアからの XMODEM/SUM プロトコル でのファイル送信については、ターミナルソフトウェアのマニュアルを参照してください。

Erasing has been done. Start Xmodem download...

図 5.34 ファイルダウンロード開始の画面出力

ファームウェアアップデート完了
 ファームウェアアップデートが完了すると、図 5.35 のメッセージを出力します。

Updating firmware has been done. >

4. エラー出力

ファームウェアアップデート中にエラーが発生した場合、その内容に応じて図 5.36 のメッセージを出力 します。

Initialize update error.	:ファームウェアアップデート処理の初期化失敗	
Finalize update error.	:ファームウェアアップデート処理の終了処理失敗	
Erasing error.	:コードフラッシュメモリ消去エラー	
Send error.	:送信処理失敗	
Receive error.	:受信処理失敗	
Timeout.	:XMODEM/SUM プロトコル転送のタイムアウト	
Data error.	:XMODEM/SUM プロトコル転送のデータエラー	
Block processing error.	:データ解析エラー、コードフラッシュメモリ書き込みエラー	
Set write complete information error. :書き込み完了情報の書き込み処理失敗		
図 5.36 エラー発生時の画面出力		

5. アップデートキャンセル

1.アップデート確認のとき、'Y'または'y'以外のコマンドを送信すると、図 5.37 のメッセージを出力し アップデートをキャンセルします。

Command canceled.

図 5.37 アップデートキャンセルの画面出力

図 5.35 ファームウェアアップデート完了の画面出力

5.3.3 ファームウェア起動処理

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

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

(1) 概略フロー

図 5.38 にファームウェア起動処理の概略フローを示します。



図 5.38 ファームウェア起動処理の概略フロー

- (2) ターミナルソフトウェア画面出力
- 1. ファームウェア起動確認
 - ファームウェア起動処理は、ファームウェア起動確認のため図 5.39 のメッセージを出力します。

Execute firmware (Y/N)?

図 5.39 ファームウェア起動確認の画面出力

2. ファームウェアアップデートプログラム起動確認

ファームウェアアップデートプログラム更新後のファームウェア起動処理では、新しいファームウェア アップデートプログラムのバージョン確認のため図 5.40のメッセージを出力します。

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

図 5.40 新しいファームウェアアップデートプログラム起動確認の画面出力

3. ファームウェア起動

'Y'または'y'を送信すると、図 5.41 のメッセージを出力して、スタートアッププログラム保護機能の領域 を入れ替えたのちソフトウェアリセットを実行します。

Switch Start-Up area and do software reset.
図 5.41 スタートアッププログラム保護機能の領域入れ替えとソフトウェアリセットの画面出力

4. エラー出力

ファームウェア起動中にエラーが発生した場合、その内容に応じて図 5.42 のメッセージを出力します。

Initialize update error.	:ファームウェア起動処理の初期化失敗	
Finalize update error.	:ファームウェア起動処理の終了処理失敗	
Reset vector of the firmware is	invalid. :ファームウェアのリセットベクタが無効	
Switching Start-Up area error.	:スタートアッププログラム保護機能の領域入れ替えエラー	
図 5.42 エラー発生時の画面出力		

5. ファームウェア起動キャンセル

1.ファームウェア起動確認のとき、'Y'または'y'以外のコマンドを送信すると、図 5.43 のメッセージを出 カしアップデートをキャンセルします。

Command canceled.

図 5.43 ファームウェア起動キャンセルの画面出力


5.4 ファームウェアプログラムの概略フローと画面出力

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

(1) 概略フロー

図 5.44 にファームウェアプログラムの概略フローを示します。



図 5.44 ファームウェアプログラムの概略フロー



- (2) ターミナルソフトウェア画面出力
- 1. ファームウェアプログラム起動

ファームウェアプログラムは、ファームウェアプログラム起動の通知のため図 5.45 のメッセージを出力 します。

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

図 5.45 ファームウェアプログラム起動の画面出力

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

2. エラー出力

ファームウェアアップデートプログラム起動中にエラーが発生した場合、その内容に応じて図 5.46 の メッセージを出力します。

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

図 5.46 エラー発生時の画面出力



5.5 ファームウェアアップデートプログラム詳細

5.5.1 ファイル構成

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

表 5.6 ファームウェアアップデートプログラムで使用するファイル

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

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

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



5.5.2 定数一覧

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

定数名	設定値	内容
FW_UP_PROGRAM_VERSION	(0x0100u)	バージョン情報
DUMMY_DATA	(0xAA55AA55AA55AA55u)	書き込み完了情報格納領域の 前に格納するダミーデータ
MASK_NUM	(0x0Fu)	下位4ビット取得用マスク
ASCII_CODE_NUM	(0x30u)	「0」の文字コード
ASCII_CODE_POINT	(0x2Eu)	「.」の文字コード
DELAY_NUM	(1u)	R_BSP_SoftwareDelay 関数 に引数として渡す遅延時間
TIMEOUT_NUM	(10000u)	10 秒のタイムアウト時間判 定用カウント数
RECV_BYTE_SIZE	(1u)	受信用1バイトのサイズ
SEND_BYTE_SIZE	(1u)	送信用1バイトのサイズ
COMMAND_UPDATE_FIRM	('1')	入力コマンド用文字コード (ファームウェアプログラム アップデートコマンド)
COMMAND_UPDATE_FIRM_UPDATE	('2')	入力コマンド用文字コード (ファームウェアアップデー トプログラムアップデートコ マンド)
COMMAND_EXEC_PROGRAM	('3')	入力コマンド用文字コード (ファームウェア起動コマン ド)
COMMAND_YES_UPPER	('Y')	入力コマンド用文字コード ("Y")
COMMAND_YES_LOWER	('y')	入力コマンド用文字コード ("y")
COMMAND_CR	('¥r')	入力コマンド用文字コード (改行コード)
STRING_MAX_SIZE	RX130 グループ: (SCI_CFG_CH1_TX_BUFSIZ) RX140 グループ: (SCI_CFG_CH1_TX_BUFSIZ) RX231 グループ: (SCI_CFG_CH5_TX_BUFSIZ) RX261 グループ: (SCI_CFG_CH6_TX_BUFSIZ)	出力する文字列の最大サイズ

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

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

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



定数名	設定値	内容
FW_UP_BINARY_BUF_SIZE	(256u)	コードフラッシュメモリ書
		き込み用データのバッファ
		サイズ
FW_UP_BINARY_BUF_NUM	(2u)	コードフラッシュメモリ書
		き込み用データのバッファ
		数
FW_UP_BUF_NUM	(60u)	解析したモトローラSレ
		コードフォーマットデータ
		の内容を格納する配列の数
FW_UP_FIRM_ST_ADDRESS	RX130 グループ:	ファームウェアブログラム
	(FLASH_CF_BLOCK_127)	を書き込む領域の先頭アト
	RX140 グループ:	
	(FLASH_CF_BLOCK_127)	
	RX231 クルーフ:	
	(FLASH_CF_BLOCK_255)	
	RX261 クルーフ:	
	(FLASH_CF_BLOCK_255)	
FW_UP_FIRM_EN_ADDRESS		ファームワェアフロクラム
	$(FLASH_CF_BLOCK_37 - 1)$	を書ざ込む領域の取終アト
	$RX^{140} \mathcal{O} \mathcal{V} - \mathcal{I}:$	
	(FLASH_CF_BLOCK_19 - 1)	
	$RX231 \mathcal{O}\mathcal{W} = \mathcal{O}$:	
	(FLASH_CF_BLOCK_19 - 1)	
EW/ LIP CONST 1 ST ADDRESS	(FLASH_CF_BLOCK_19-1) RX130 ガループ・	
FW_OF_CONST_T_ST_ADDRESS	(ELASH CE BLOCK 37)	足数) フ 領域 の 開始)
	$RX140 $ $\pi \mu - \pi$	
	(FLASH OF BLOCK 19)	
	RX231 グループ・	
	(FLASH OF BLOCK 19)	
	RX261 グループ・	
	(FLASH CF BLOCK 19)	
FW UP CONST 1 FIN ST ADDRESS	RX130 グループ:	定数データ領域1の最終書
	(FLASH CF BLOCK 34 - 256)	き込み処理に使用する先頭
	RX140 グループ:	アドレス
	(FLASH_CF_BLOCK 17 - 256)	
	RX231 グループ:	
	(FLASH_CF_BLOCK_17 - 256)	
	RX261 グループ :	
	(FLASH_CF_BLOCK_17 - 256)	

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

定数名	設定値	内容
FW_UP_COMPLETE_1_ST_ADDRESS	RX130 グループ :	書き込み完了情報格納領域
	(FLASH_CF_BLOCK_34 - 8)	1の先頭アドレス
	RX140 グループ :	
	(FLASH_CF_BLOCK_17 - 8)	
	RX231 グループ :	
	(FLASH_CF_BLOCK_17 - 8)	
	RX261 グループ :	
	(FLASH_CF_BLOCK_17 - 8)	
FW_UP_CONST_1_EN_ADDRESS	RX130 グループ :	定数データ領域1の最終ア
	(FLASH_CF_BLOCK_34 - 1)	ドレス
	RX140 グループ :	
	(FLASH_CF_BLOCK_17 - 1)	
	RX231 グループ:	
	(FLASH_CF_BLOCK_17 - 1)	
	RX261 グループ:	
	(FLASH_CF_BLOCK_17 - 1)	
FW_UP_CONST_2_ST_ADDRESS	RX130 グループ:	定数データ領域2の先頭ア
	(FLASH_CF_BLOCK_34)	ドレス
	(FLASH_CF_BLOCK_17)	
	$RX231 \mathcal{I}\mathcal{I}\mathcal{I}\mathcal{I}\mathcal{I}\mathcal{I}\mathcal{I}$	
	(FLASH_CF_BLOCK_17)	
	(F,A,C,C,C,C,C,C,C,C	
EW LID CONST 2 FIN ST ADDRESS	(FLASH_CF_BLOCK_17)	
FW_0P_CONST_2_FIN_ST_ADDRESS	(FLASH CE PLOCK 21 256)	た 数 ナーダ
	(FLASH_CF_BLOCK_31 - 250) PX140 グループ・	アドレス
	(ELASH CE BLOCK 15 - 256)	
	(FLASH_CF_BLOCK_13 - 230) RX231 ガループ・	
	(FLASH CE BLOCK 15 - 256)	
	RX261 グループ・	
	(FLASH_CE_BLOCK_15 - 256)	
FW UP COMPLETE 2 ST ADDRESS	RX130 グループ・	まき込み完了情報格納領域
	(FLASH CF BLOCK 31 - 8)	2の開始アドレス
	RX140 グループ:	
	(FLASH CF BLOCK 15 - 8)	
	RX231 グループ :	
	(FLASH_CF_BLOCK_15 - 8)	
	RX261 グループ :	
	(FLASH_CF_BLOCK_15 - 8)	

RENESAS

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

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

定数名	設定値	内容
FW_UP_CONST_2_EN_ADDRESS	RX130 グループ :	定数データ領域2の最終ア
	(FLASH_CF_BLOCK_31 - 1)	ドレス
	RX140 グループ :	
	(FLASH_CF_BLOCK_15 - 1)	
	RX231 グループ:	
	(FLASH_CF_BLOCK_15 - 1)	
	(FLASH_CF_BLOCK_15 - 1)	フクーレマッププログラノ
FW_UP_STUP_ST_ADDRESS	(E A SH CE PLOCK 15)	スタートアッフラロクラム 保護機能のデフォルト領域
	$(FLASI_CF_BLOCK_IS)$	休暖機能のアンオル F 横域 の先頭アドレス
	(FLASH CE BLOCK 7)	
	RX231 グループ:	
	(FLASH CF BLOCK 7)	
	RX261 グループ :	
	(FLASH_CF_BLOCK_7)	
FW_UP_STUP_EN_ADDRESS	(FLASH_CF_BLOCK_END)	スタートアッププログラム
		保護機能のデフォルト領域
		の最終アドレス
FW_UP_FIRM_BLOCK_NUM	RX130 グループ : (90u)	ファームウェアプログラム
	RX140 グループ : (108u)	を書き込む領域のブロック
	RX231 グループ : (236u)	数
	RX261 グループ : (236u)	
FW_UP_CONST_BLOCK_NUM	RX130 グループ:(3u)	定数データ領域のブロック
	RX140 グループ:(2u)	数
	RX231 グループ:(2u)	
	RX261 グループ : (2u)	
FW_UP_STUP_BLOCK_NUM	RX130 グループ: (16u)	スタートアッププログラム
	RX140 グループ:(8u)	保護機能のテフォルト領域
	RX231 グループ:(8u)	のフロック致
	RX261 グループ:(8u)	
FW_UP_FIRM_RESETVECT	RX130 グループ:	ファームウェアプログラム
	(FLASH_CF_BLOCK_15 - 4)	のリセットヘクタのアトレ
		^
	(FLASH_CF_BLOCK_7 - 4) RX231 ガループ・	
	(FLASH CF BLOCK 7 - 4)	
	RX261 グループ・	
	(FLASH CF BLOCK 7 - 4)	
FW_UP_BLANK_VALUE	(0xFFFFFFFF)	コードフラッシュメモリが
		ブランク時の読み出し値
PRCR_KEY	(0xA500u)	PRCR レジスタのキーコー
		ド
SET_PRC1	(0x0002u)	PRCR レジスタの PRC1
		ビットを設定するための値
MCU_RESET	(0xA501u)	SWRR レジスタに設定して
		MCU をリセットさせる値



定数名	設定値	内容
MOT_S_CHECK_SUM_FIELD	(0x02u)	モトローラ S レコードフォーマットのチェックサム
		フィールドの文字数
ADDRESS_LENGTH_S1	(0x04u)	モトローラ S レコードフォーマットのアドレス
		フィールドの文字数(S1 タイプ)
ADDRESS_LENGTH_S2	(0x06u)	モトローラSレコードフォーマットのアドレス
		フィールドの文字数(S2 タイプ)
ADDRESS_LENGTH_S3	(0x08u)	モトローラSレコードフォーマットのアドレス
		フィールトの文字数(S3タイフ)
BUF_LOCK	(1u)	指定したモトローラSレコードフォーマットのパッ
	(0)	
BUF_UNLOCK	(0u)	指定したモトローフ 5 レコートノオーマットのハッ コッけ関切され エいそ
MOT RECORD SO	(0)	
	(00)	モドロー ノ S レコード ノオーマッドのレコードダイ プ (S0 タイプ)
MOT RECORD S1	(11)	チトローラらレコードフォーマットのレコードタイ
	(10)	プ(S1 タイプ)
MOT RECORD S2	(2u)	モトローラSレコードフォーマットのレコードタイ
		プ (S2 タイプ)
MOT_RECORD_S3	(3u)	モトローラ S レコードフォーマットのレコードタイ
		プ(S3 タイプ)
MOT_RECORD_S7	(7u)	モトローラ S レコードフォーマットのレコードタイ
		プ(S7 タイプ)
MOT_RECORD_S8	(8u)	モトローラSレコードフォーマットのレコードタイ
		ブ(S8 タイプ)
MOT_RECORD_S9	(9u)	モトローラ S レコードフォーマットのレコードタイ
		フ (S9 タイフ)
MASK_LOWER_BYTE	(0x000000FFu)	下位1パイト取得用マスク
ASCII_CODE_0	(0x30u)	10」の文字コード
ASCII_CODE_9	(0x39u)	19」の文字コード
ASCII_CODE_UPPER_A	(0x41u)	「A」の文字コード
ASCII_CODE_UPPER_F	(0x46u)	「F」の文字コード
ASCII_CODE_LOWER_A	(0x61u)	「a」の文字コード
ASCII_CODE_LOWER_F	(0x66u)	「f」の文字コード
CONVERT_HEX_NUM	(0x0Fu)	0~9の文字コードを 16 進数に変換するための値
CONVERT_HEX_UPPER_CHAR	(0x37u)	A~Fの文字コードを16進数に変換するための値
CONVERT_HEX_LOWER_CHAR	(0x57u)	a~f の文字コードを 16 進数に変換するための値

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



5.5.3 型定義一覧

図 5.47~図 5.50 にファームウェアアップデートプログラムで使用する型定義を示します。

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

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

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



```
typedef enum e update mode t
{
   UPDATE_FW,
   UPDATE_FW_UP_1,
  UPDATE FW UP 2,
} update_mode_t;
typedef enum e_fw_up_return_t
{
   FW UP SUCCESS,
   FW UP ERR OPENED,
   FW UP ERR NOT OPEN,
   FW UP ERR NULL PTR,
   FW UP ERR INVALID RECORD,
   FW UP ERR BUF FULL,
   FW UP ERR BUF EMPTY,
   FW_UP_ERR_INITIALIZE,
   FW UP ERR ERASE,
   FW UP ERR WRITE,
   FW UP ERR VERIFY,
   FW UP ERR SWITCH AREA,
   FW UP ERR INVALID ADDRESS,
   FW UP ERR INVALID RESETVECT,
   FW_UP_ERR_INTERNAL
} fw up return t;
typedef struct st fw up fl data t
{
   uint32 t src addr;
   uint32 t dst addr;
   uint32 t len;
   uint16 t count;
} fw up fl data t;
```

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

```
typedef enum fw up mot s cnt t
{
   STATE_MOT_S_RECORD_MARK = 0,
   STATE_MOT_S_RECORD_TYPE,
   STATE MOT S LENGTH 1,
   STATE MOT S LENGTH 2,
   STATE_MOT_S_ADDRESS,
   STATE_MOT_S_DATA,
   STATE_MOT_S_CHKSUM_1,
   STATE_MOT_S_CHKSUM_2
} fw_up_mot_s_cnt_t;
typedef struct MotSBufS
{
  uint8_t addr_length;
uint8_t data_length;
uint8_t * p_address;
uint8_t * p_data;
   uint8_t type;
uint8_t act;
   struct MotSBufS * p_next;
} fw up mot s buf t;
typedef struct WriteDataS
{
   uint32_t addr;
   uint32_t len;
uint8_t data[FW_UP_BINARY_BUF_SIZE];
   struct WriteDataS * p next;
   struct WriteDataS * p prev;
} fw up write data t;
```

図 5.50 ファームウェアアップデートプログラムで使用する型定義(r_fw_up_buf.h)

5.5.4 変数一覧

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

表 5.14 、	ファームウェアア	ップデート	ヽプログラム [・]	で使用する	static 型変数	(main.c)
----------	----------	-------	---------------------	-------	------------	----------

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

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

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

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

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



刑	亦粉夕	内容	估田問 粉
±			医用因数
Static	"sp_app_put_mot_s_but		
Iw_up_mot_s_bui_t		処理で現在使用しているモト	fw_up_put_mot_s
		ローラSレコードデータバッ	
		ファへのポインタ	
static	*sp_app_get_mot_s_buf	コードフラッシュメモリ書き込	fw_up_buf_init
fw_up_mot_s_buf_t		み用データ作成処理で現在使用	fw_up_get_binary
		しているモトローラ S レコード	
		データバッファへのポインタ	
static	s_mot_s_buf[FW_UP_BUF	モトローラSレコードフォー	fw_up_buf_init
fw_up_mot_s_buf_t	_NUM]	マットデータの内容を格納する	fw_up_memory_init
		バッファ	
static	*sp_app_write_buf	現在使用しているコードフラッ	fw_up_buf_init
fw_up_write_data_t		シュメモリ書き込み用データ	fw_up_get_binary
		バッファへのポインタ	
static	s_write_buf[FW_UP_BINA	コードフラッシュメモリ書き込	fw_up_buf_init
fw_up_write_data_t	RY_BUF_NUM]	み用データを格納するバッファ	
static	s_mot_s_data_state	モトローラSレコードフォー	fw_up_buf_init
fw_up_mot_s_cnt_t		マットデータの解析状態	fw_up_put_mot_s
static uint32_t	s_write_current_address	現在のコードフラッシュメモリ	fw_up_buf_init
		書き込み先アドレス	fw_up_get_binary
static bool	s_detect_terminal_flag	終端レコード検出フラグ	fw_up_buf_init
			fw_up_put_mot_s
			fw_up_get_binary

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



型	変数名	内容	使用関数
const uint32_t	g_program_version	ファームウェアアップデート	show_menu_start_up
		プログラムのバージョン	
const uint64_t	g_dummy_data	書き込み完了情報格納領域の	-
		前に格納するダミーデータ	
static const uint8_t	s_string_menu0[]	RX130 グループ :	show_menu_start_up
		"RX130 firmware update using	
		Start-Up Program Protection	
		menu ver"	
		RX140 グルーノ:	
		"RX140 firmware update using	
		menu ver"	
		RX231 グループ・	
		"RX231 firmware undate using	
		Start-Up Program Protection	
		menu ver"	
		RX261 グループ :	
		"RX261 firmware update using	
		Start-Up Program Protection	
		menu ver"	
static const uint8_t	s_string_menu1[]	"1Update firmware	show_menu_start_up
static const vist0 t		program¥r¥n"	ahann ar ann at ant un
static const uint8_t	s_string_menu2[]	2Opdate Innware update	snow_menu_start_up
static const uint8 t	s string menu3[]	"3Execute firmware¥r¥n"	show menu start up
static const uint8 t	s string input[]	"> "	show menu start up
			update firmware
			exec_firmware
static const uint8_t	s_string_crlf[]	"¥r¥n"	main
			update_firmware
			exec_firmware
static const uint8_t	s_string_update[]	"Erase flash memory and write	update_firmware
		firmware (Y/N)?"	
static const unt8_t	s_string_erase_success[]	"Erasing has been done.¥r¥n"	update_firmware
static const uint8_t	s_string_download[]	"Start Xmodem	update_firmware
static const uint® t	s string finish ymodom[]		undato firmwara
	s_string_initsri_xtrioderit[]	done.¥r¥n"	upuale_iiiiiware
static const uint8 t	s string exec firm[]	"Execute firmware (Y/N)?"	exec firmware
static const uint8 t	s string reset[]	"Switch Start-Up area and do	exec firmware
	_ _ 	software reset.¥r¥n"	
static const_uin8_t	s_string_exec_firm_update[]	"Execute new firmware update	exec_firmware
		program Ver"	
static const uint8_t	s_string_y_n[]	" (Y/N)?"	exec_firmware
static const uint8_t	s_string_cancel[]	"Command canceled.¥r¥n"	update_firmware
			exec_firmware

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



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

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



5.5.5 関数一覧

表 5.20 にファームウェアアップデートプログラムの関数を、表 5.21 にファームウェアアップデートプロ グラムで使用する FIT モジュールの関数を、表 5.22~表 5.25 にファームウェアアップデートプログラムで 使用する e2 studio のスマートコンフィグレータにより生成される関数を示します。



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

表 5.20 ファームウェアアップデートプログラムの関数



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

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

表 5.22 ファームウェアアップデートプログラムで使用する e² studio のスマートコンフィグレータにより 生成される関数(RX130 グループ)

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

表 5.23 ファームウェアアップデートプログラムで使用する e² studio のスマートコンフィグレータにより 生成される関数(RX140 グループ)

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

表 5.24 ファームウェアアップデートプログラムで使用する e² studio のスマートコンフィグレータにより 生成される関数(RX231 グループ)

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

表 5.25 ファームウェアアップデートプログラムで使用する e² studio のスマートコンフィグレータにより 生成される関数(RX261 グループ)

関数名	対応する FIT モ ジュール	用途	使用している関数
R_SCI_PinSet_SCI6	SCI FIT モジュール	SCI 用端子設定	main



5.6 ファームウェアプログラム詳細

5.6.1 ファイル構成

表 5.26 にファームウェアプログラムで使用するファイル、表 5.27 にファームウェアプログラムで使用する標準インクルードファイルを示します。なお、FIT モジュールおよび統合開発環境で自動生成されるファ イルは除きます。

ファイル名	概要
main.c	メインソースファイル
main.h ^{*1}	メインインタフェースファイル
r_fw_up_rx.c ^{*1}	ファームウェアアップデートソースファイル
r_fw_up_rx_if.h ^{*1}	ファームウェアアップデートインタフェースファイル
r_fw_up_rx_private.h ^{*1}	ファームウェアアップデートヘッダファイル
r_fw_up_buf.c ^{*1}	ファームウェアデータ用バッファ処理ソースファイル
r_fw_up_buf.h ^{*1}	ファームウェアデータ用バッファ処理ヘッダファイル
【注】*1 ファームウェアアップラ	ートプログラムと同一のファイルを使用しています

表 5.26 ファームウェアプログラムで使用するファイル

【注】*1 ファームウェアアッフテートフロクラムと同一のファイルを使用しています。

表 5.27 ファームウェアプログラムで使用する標準インクルードファイル

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



5.6.2 定数一覧

表 5.28 にファームウェアプログラムで使用する定数(main.c)を示します。なお、ファームウェアアップデートプログラムと同一のファイルに記載されている定数は「5.5.2 定数一覧」を参照してください。

表 5.28 ファームウェアプログラムで使用する定数(main.c)

定数名	設定値	内容
RECV_BYTE_SIZE	(1)	受信用1バイトのサイズ
COMMAND_CR	('¥r')	入力コマンド用文字コード (改行コード)
STRING_MAX_SIZE	RX130 グループ: (SCI_CFG_CH1_TX_BUFSIZ) RX140 グループ: (SCI_CFG_CH1_TX_BUFSIZ) RX231 グループ: (SCI_CFG_CH5_TX_BUFSIZ) RX261 グループ: (SCI_CFG_CH6_TX_BUFSIZ)	出力する文字列の最大サイズ

5.6.3 型定義一覧

型定義は「5.5.3型定義一覧」を参照してください。



5.6.4 変数一覧

表 5.29 にファームウェアプログラムで使用する static 型変数(main.c)を、表 5.30 にファームウェアプ ログラムで使用する const 型変数(main.c)を示します。なお、ファームウェアアップデートプログラムと 同一のファイルに記載されている変数は「5.5.4 変数一覧」を参照してください。

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

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

表 5.30 ファームウェアプログラムで使用する const 型変数(main.c)

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



5.6.5 関数一覧

表 5.31 にファームウェアプログラムの関数を、表 5.32 にファームウェアプログラムで使用する FIT モジュールの関数を、表 5.33~表 5.35 にファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される関数を示します。なお、ファームウェアアップデートプログラムと同一のファイルに記載されており、ファームウェアプログラムでは使用していない関数は除きます。

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

表 5.31 ファームウェアプログラムの関数

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



表 5.33 ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される 関数 (RX130 グループ)

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

表 5.34 ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される 関数(RX140 グループ)

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

表 5.35 ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される 関数 (RX231 グループ)

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

表 5.36 ファームウェアプログラムで使用する e2 studio のスマートコンフィグレータにより生成される 関数 (RX261 グループ)

関数名	対応する FIT モ	用途	使用している関数
	ジュール		
R_SCI_PinSet_SCI6	SCI FIT モジュール	SCI 用端子設定	main



6. プロジェクトをインポートする方法

サンプルプログラムは e² studio のプロジェクト形式で提供しています。本章では、CS+ヘプロジェクト をインポートする方法を示します。インポート完了後、ビルドおよびデバッグの設定を確認してください。

6.1 CS+での手順

CS+でご使用になる際は、下記の手順で CS+にインポートしてください。

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

(使用する CS+のバージョンによっては画面が異なる場合があります。)



図 6.1 プロジェクトを CS+にインポートする方法

7. 参考ドキュメント

- RX110 グループ ユーザーズマニュアル ハードウェア編 (R01UH0421)
- RX111 グループ ユーザーズマニュアル ハードウェア編 (R01UH0365)
- RX113 グループ ユーザーズマニュアル ハードウェア編 (R01UH0448)
- RX130 グループ ユーザーズマニュアル ハードウェア編 (R01UH0560)
- RX140 グループ ユーザーズマニュアル ハードウェア編 (R01UH0905)
- RX230 グループ、RX231 グループ ユーザーズマニュアル ハードウェア編 (R01UH0496)
- RX260 グループ、RX261 グループ ユーザーズマニュアル ハードウェア編 (R01UH1045)
- (最新版をルネサスエレクトロニクスホームページから入手してください)
- テクニカルアップデート/テクニカルニュース
 (最新の情報をルネサスエレクトロニクスホームページから入手してください)
- Cコンパイラマニュアル
- RX ファミリ用 C/C++コンパイラパッケージ
 (最新版をルネサスエレクトロニクスホームページから入手してください)



改訂記録

		改訂内容	
Rev.	発行日	ページ	ポイント
1.00	Jul.01.17	-	初版発行
1.10	Sep.26.18	-	RX100 シリーズを追加
1.20	Mar.24.22	全体	対象デバイスに以下を追加
			・RX140 グループ
			 ・RX140 グループ 以下の理由でサンプルプログラムの仕様を変更 ファームウェアアップデートプログラム容量削減のため ・CMT モジュールを削除 ・プログラム部と定数部を分離 プログラム更新時の利便性を向上させるため ・書き換え完了後のファームウェア起動手順を変更 表 1.2 の内容を変更 表 1.5 の設定内容を変更 「2.2.コンパイラパッケージの入手方法」と「2.3.Renesas Flash Programmer の入手方法」の URL を変更 「3.プロジェクトの構築」と「4.動作確認」を更新 「5.1.ファームウェアアップデートプログラムの構成」を追加
			「5.2.動作概要」で使用する図を変更
			「5.2.7.更新用ファームウェアアップデートプログラムの作成 方法」を追加
			「5.3.ファームウェアアップデートプログラムの概略フロー と画面出力」と「5.4.ファームウェアプログラムの概略フ ローと画面出力」で使用する図を変更
			「5.5.ファームウェアアップデートプログラム詳細」と「5.6. ファームウェアプログラム詳細」を更新
			「6.プロジェクトをインポートする方法」の内容を変更
			 「7.参考ドキュメント」を追加
1.30	Sep.05.24	全体	対象デバイスに以下を追加
			・RX261 グループ

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―オン リセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入に より、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」について の記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した 後に切り替えてください。リセット時、外部発振子(または外部発振回路)を用いたクロックで動作を開始するシステムでは、クロックが十分安定し た後、リセットを解除してください。また、プログラムの途中で外部発振子(または外部発振回路)を用いたクロックに切り替える場合は、切り替え 先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、VIL(Max.)から VIH(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、VIL(Max.)から VIH(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。

リザーブアドレス(予約領域)のアクセス禁止
 リザーブアドレス(予約領域)のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられているリザーブアドレス(予約領域)があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合が あります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

- 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。
- 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、Harshenvironment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある 機器・システム(生命維持装置、人体に埋め込み使用するもの等)、もしくは多大な物的損害を発生させるおそれのある機器・システム(宇宙機器 と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等)に使用されることを意図しておらず、これらの 用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責 任を負いません。

- 7. あらゆる半導体製品は、外部攻撃からの安全性を100%保証されているわけではありません。当社ハードウェア/ソフトウェア製品にはセキュリ ティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害(当社製品または当社製品が使用されてい るシステムに対する不正アクセス・不正使用を含みますが、これに限りません。)から生じる責任を負うものではありません。当社は、当社製品ま たは当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行 為(「脆弱性問題」といいます。)によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害に ついて、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア/ソフトウェア製品について、商品 性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
- 8. 当社製品をご使用の際は、最新の製品情報(データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等)をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
- 9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする 場合があります。また、当社製品は、データシート等において高信頼性、Harshenvironment向け製品と定義しているものを除き、耐放射線設計を 行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客 様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を 行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行って ください。
- 10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用 を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことに より生じた損害に関して、当社は、一切その責任を負いません。
- 11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
- 12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたしま す。
- 13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
- 14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。
- 注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社が直接的、間接的に 支配する会社をいいます。
- 注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.5.0-12020.10)

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 ロ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/