RX65N グループ
産業機器向けセキュアアップデートソリューション
ソリューション実装ガイド

R01AN4507JJ0100
Rev.1.0.0
2018.12.19

要旨

本資料は、お客様のプログラムに産業機器向けセキュアアップデートソリューションを実装する方法について説明します。
本ソリューションを導入する際に必要な情報や使用するツール類の操作方法については、スタートアップガイド（R01AN4506JJ0100）を参照してください。
本ソリューションのプログラムを含む提供パッケージは、ルネサスマイコンをご採用/ご採用予定のお客様に提供させていただいています。お取引のあるルネサスエレクトロニクス営業窓口か、下記窓口にお問い合わせください。（https://www.renesas.com/jp/ja/support/contact.html）

対象デバイス

RX65N グループ

目次

1. 概要

1.1 はじめに

本書は産業機器向けセキュアアップデートソリューション（以降、本ソリューション）に含まれる、RX65N でのファームウェアアップデートプログラムをお客様のプログラムに組み込む方法を説明します。ファームウェアアップデートプログラムを組み込むことにより、安全にファームウェアのアップデートを行うことができます。
本ソリューションの特徴については、“産業機器向けセキュアアップデートソリューション　スタートアップガイド”（R01AN4506JJ0100）を参照してください。

1.2 提供パッケージ

本ソリューションで提供するパッケージの一覧を表 1.1 に、パッケージの構成を表 1.2 に示します。

表 1.1　パッケージ一覧

名称	説明
産業機器向けセキュアアップデートソリューションパッケージ	本パッケージには RX65N 用プログラムおよび Windows GUI ツール、ドキュメントが含まれます。
鍵生成ツール	ファームウェアアップデートで使用する鍵生成を行う Windows PC 用アプリケーション
ファームウェア管理ツール	ファームウェアの管理および量産用ファームウェアの生成を行う Windows PC 用アプリケーション
アップデート管理ツール	ファームウェア管理ツールとの連携およびデバイスへのファームウェアアップデートを行う Windows PC 用アプリケーション
ユニーク ID 読み出しツール	デバイス毎のユニーク ID 情報を読み出す Windows PC 用アプリケーション
フラッシュプログラマ制御用マクロ	フラッシュプログラマを制御するためのターミナルソフト用のマクロ
RX65N 用アプリケーションプログラム	ファームウェアアップデートプログラムとサンプルプログラムを含むプロジェクト一式 RX ファミリ用リアルタイム OS[RI600V4]（以降、RTOS）版および NonOS 版
RX65N 用セキュアブートプログラム	RTOS 版または Non OS 版の RX65N 用セキュアブートプログラム（モトローラ S レコードフォーマットファイル）
スタートアップガイド	本ソリューションの概要および操作方法に関するドキュメント
ソリューション実装ガイド	本ソリューションの実装方法に関するドキュメント

表 1.2 パッケージの構成

ファイル名	説明
¥¥Secure_firmware_Update¥Key_generator	
Secure_Firmware_Update_Key_Generator.exe	鍵生成ツール
Secure_Firmware_Update.mdb	ファームウェア管理用データベース
¥¥Secure_firmware_Update¥Firmware_Manager	
Secure_Firmware_Update_Firmware_Manager.exe	ファームウェア管理ツール
Secure_Firmware_Update.mdb	ファームウェア管理用データベース
¥¥Secure_firmware_Update¥Update_Manager	
Secure_Firmware_Update_Update_Manager.exe	アップデート管理ツール
Secure_Firmware_Update_Update_Manager.mdb	アップデート管理ツール用データベース
¥¥Secure_firmware_Update¥UID_Reader	
Secure_Firmware_Update_UID_Reader.exe	ユニーク ID 読み出しツール
¥¥Secure_firmware_Update¥FlashProgramer	

rx65n_write.ttl	フラッシュプログラマを制御するターミナルソフト用のマクロ
¥¥Secure_firmware_Update¥RX65N¥nonOS_rx65n_secure_boot	
nonOS_rx65n_secure_boot.mot	Non OS 版セキュアブートプログラム（ファームウェア生成用）
nonOS_rx65n_secure_boot_debug.mot	Non OS 版セキュアブートプログラム（デバッグ用）
¥¥Secure_firmware_Update¥RX65N¥rx65n_secure_boot	
rx65n_secure_boot.mot	RTOS 版セキュアブートプログラム（ファームウェア生成用）
rx65n_secure_boot_debug.mot	RTOS 版セキュアブートプログラム（デバッグ用）
¥¥Secure_firmware_Update¥RX65N¥nonOS_rx65n_app_prog	
Non OS 版 プロジェクト一式	Non OS 版 RX65N 用ファームウェアアップデートプログラム
¥¥Secure_firmware_Update¥RX65N¥rx65n_app_prog	
RTOS 版 プロジェクト一式	RTOS 版 RX65N 用ファームウェアアップデートプログラム
¥¥Secure_firmware_Update¥document	
スタートアップガイド_セキュア FW アップデートソリューション開発.docx	スタートアップガイド
ソリューション実装ガイド_セキュア FW アップデートソリューション開発.docx	ソリューション実装ガイド

1.3 動作環境

ファームウェアアップデートプログラムを組み込むときに必要である環境を表 1.3 に示します。

表 1.3 組み込み時に必要な動作環境

項目	内容
使用マイコン	RX65N（セキュア対応版、ROM 容量 2MB、デュアルバンク）
通信	Ethernet もしくは SCI

ファームウェアアップデートプログラムで必要なハードウェアリソースを表 1.4 に示します。

表 1.4 必要なハードウェアリソース

ハードウェアリソース	使用チャネル数	概要
CMT	RTOS : 1	チャネル 0 を RTOS 管理用に占有
	NonOS : 2	Ethernet と USB（シリアル）（※1）で使用 チャネル番号はルネサスが提供している Firmware Integration Technology（以降、FIT）の CMT モジュールで動的に管理します。
ETHERC	1	Ethernet 通信に使用 ポート 10100（UDP）、10101（TCP）をファームウェアアップデートで使用します。
SCI	1	USB（シリアル）（※1）通信に使用 任意のチャネルをファームウェアアップデート用に占有

※1 非同期シリアル通信を指します。以降、非同期シリアルと表現します。

本ソリューションのパッケージに含まれるサンプルプログラムの動作確認を行った環境を表 1.5 に示します。

表 1.5 動作確認環境

項目	内容
使用マイコン	RX65N（セキュア対応版、ROM 容量 2MB、デュアルバンク）

使用ボード	Renesas Starter Kit+ for RX65N-2MB (Trusted Secure IP 搭載) （以降、RX65N RSK）
動作周波数	120MHz
動作電圧	3.3v
動作モード	シングルチップモード
通信	Ethernet、SCI8
統合開発環境	e2studio Version 6.00
エミュレータ	E2 エミュレータ Lite
リアルタイム OS	RI600V4 Trial 版 Version 1.04.00

1.4 参考資料

表 1.6 に参考資料の一覧を示します。

表 1.6 参考資料

ドキュメント名	資料番号	リビジョン
産業機器向けセキュアアップデートソリューション スタートアップガイド	R01AN4506JJ0100	1.00
RX65N グループ、RX651 グループ ユーザーズマニュアル ハードウェア編	R01UH0590JJ020	2.10
Renesas Starter Kit+ for RX65N-2MB ユーザーズマニュアル	R20UT3888JG0100	1.00
RI600V4 リアルタイムオペレーティングシステム ユーザーズマニュアルコーディング編	R20UT0711JJ0104	1.04
CC-RX コンパイラ ユーザーズマニュアル	R20UT3248JJ0105	1.05
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	R01AN1685JJ0360	3.60
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	R01AN1683EJ0160	1.60
RX ファミリ CMT モジュール Firmware Integration Technology	R01AN1856JJ0321	3.21
RX ファミリ イーサネットモジュール Firmware Integration Technology	R01AN2009JJ0115	1.15
RX ファミリ フラッシュモジュール Firmware Integration Technology	R01AN2184JU0330	3.30
RX ファミリ GPIO モジュール Firmware Integration Technology	R01AN1721JJ0231	2.31
RX ファミリ 簡易 I2C モジュール Firmware Integration Technology	R01AN1691JJ0220	2.20
RX ファミリ SCI モジュール Firmware Integration Technology	R01AN1815JJ0201	2.01
RX ファミリ システムタイマモジュール Firmware Integration Technology	R20AN0431JJ0100	1.00
RX ファミリ Ethernet ドライバと組み込み用 TCP/IP M3S-T4-Tiny のインタフェース変換モジュール	R20AN0311JJ0106	1.06
RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny 導入ガイド Firmware Integration Technology	R20AN0051JJ0207	2.07
組み込み用 TCP/IP M3S-T4-Tiny ユーザーズマニュアル	R20UW0031JJ0109	1.09
RX ファミリ TSIP モジュール Firmware Integration Technology	R20AN0371JJ0106	1.06
RX ファミリ AES ライブラリ	R20UW0068JJ0108	1.08

1.5 サンプルプログラム RX65N で動作するプログラムの概要

RX65N で動作するプログラムは大きく分けるとセキュアブートプログラムとアプリケーションプログラムに分かれます。セキュアブートプログラムはデバイスのパワーオンリセット時にアプリケーションプログラムの検証を行い、問題がなければ起動します。

アプリケーションプログラムはユーザプログラムとファームウェアアップデートプログラムとドライバを合わせたものになります。RX65N で動作するプログラムの構成を図 1.1、表 1.7 に示します。各プログラムの概要については 1.5.1 章～1.5.5 章を参照してください。

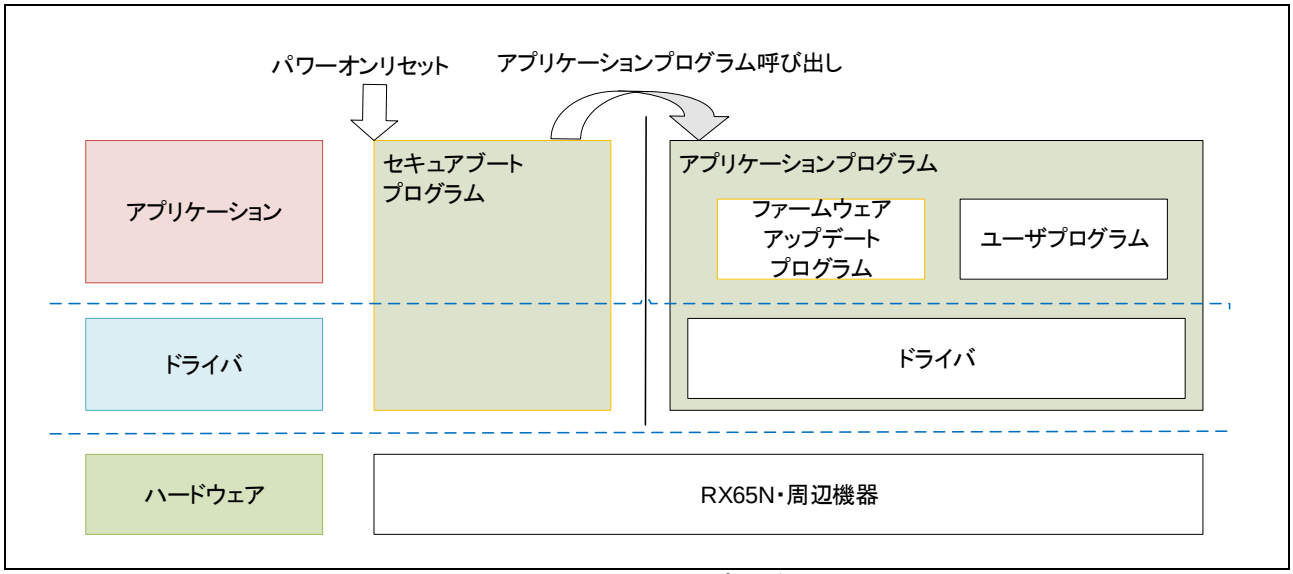

図 1.1　RX65N で動作するプログラムの構成

表 1.7　RX65N で動作するプログラムの構成

プログラム種別	説明	参照
アプリケーション プログラム	ユーザプログラムとファームウェアアップデートプログラムとドライバを合わせたものをアプリケーションプログラムと呼びます。 RTOS 版と NonOS 版のサンプルプログラムを用意しています。	1.5.1
ユーザプログラム	お客様のプログラムです。	1.5.2
ファームウェア アップデート プログラム	仮想サーバと Ethernet もしくは非同期シリアル通信を用いて安全なファームウェアアップデートを行います。認証後にファームウェアアップデートを行い、アップデート結果を仮想サーバへ返信します。 ファームウェアアップデートプログラムはサンプルプログラムに含まれています。	1.5.3
ドライバ	周辺機器・機能のドライバです。 本ソリューションのサンプルプログラムは FIT を使用しています。	1.5.4
セキュアブート プログラム	デバイス起動時にアプリケーションプログラムの検証を行い、問題がなければ起動します。セキュアブートプログラムはモトローラ S レコードフォーマットファイル（以降、mot ファイル）形式の提供となります。 セキュアブートプログラムは指定の領域に割り当てる必要があります。	1.5.5

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を 2 領域に分割して使用するデュアルバンク機能を使用しており、各領域にセキュアブートプログラムとアプリケーションプログラムの両方が書き込まれています。デュアルバンク機能を使用することにより、アプリケーションプログラムを動作させた状態でファームウェアアップデートをすることが可能になっています。図 1.2 にデュアルバンク機能での動作を示します。

① ファームウェアアップデートで非起動バンクに新しいアプリケーションプログラムを書きこみます
② 起動バンクと非起動バンクを入れ替え、次回起動時に新しいアプリケーションプログラムが起動するようにします
③ デバイスをリセットします
④ ファームウェアアップデートで書き込まれた新しいアプリケーションプログラムが起動します

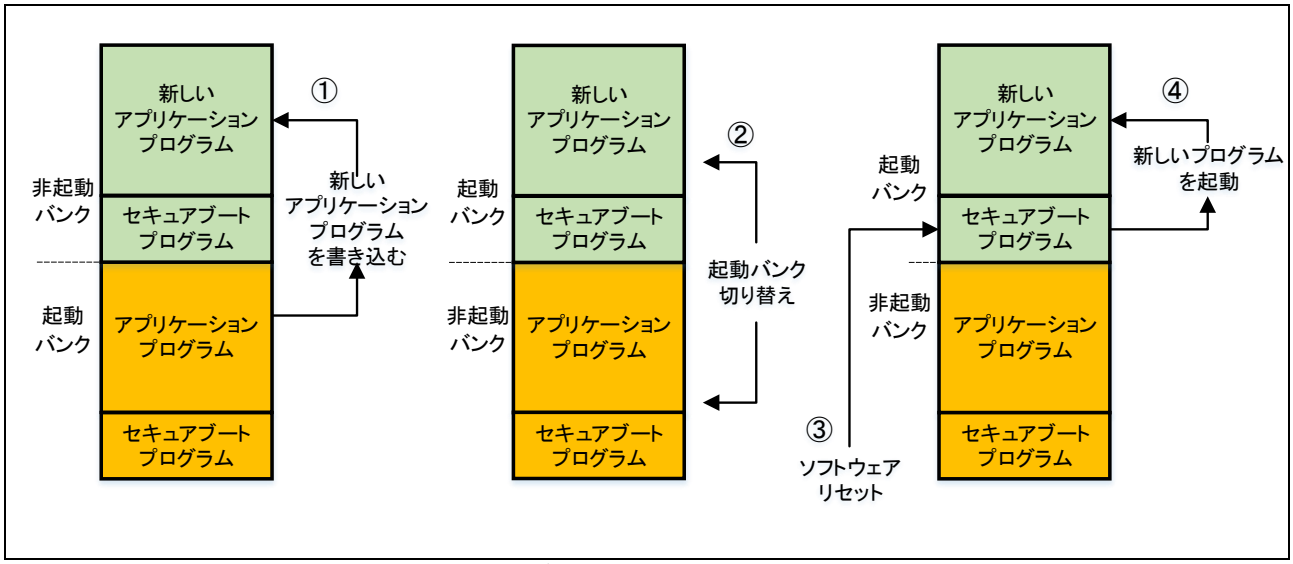

図 1.2　デュアルバンク機能での動作

本ソリューションを使用したプログラムでは固定アドレスに配置する必要のあるデータがあります。
表 1.8 に固定アドレスに配置する必要のあるデータを示します。

表 1.8　固定アドレスに配置する必要のあるデータ

アドレス	説明	参照
0x00100000 - 0x00100400	セキュアブート用データ	3.4、4.4
0xFFFDFBFC	アプリケーションプログラムのエントリアドレス	3.4、4.4
0xFFFFFF80	固定割り込みベクタ領域（RTOS のみ）	3.4
0xFFEE0000 - 0xFFEFFFFF	セキュアブートプログラム（非起動バンク）	1.5.5
0xFFFE0000 - 0xFFFFFFFF	セキュアブートプログラム（起動バンク）	1.5.5

1.5.1 アプリケーションプログラム

アプリケーションプログラムには、ユーザプログラムとファームウェアアップデートプログラムとドライバが含まれています。（図 1.3）ユーザプログラムはお客様が作成したプログラムとなり、ファームウェアアップデートプログラムはアプリケーションプログラムを更新するためのプログラムになります。ドライバは、ユーザプログラムとファームウェアアップデートプログラムで使用するドライバです。
アプリケーションプログラムでは、セキュアブートプログラムから呼び出すためのエントリアドレス設定が必要になります。エントリアドレス設定についての詳細は 3.7 章、4.7 章を参照してください。

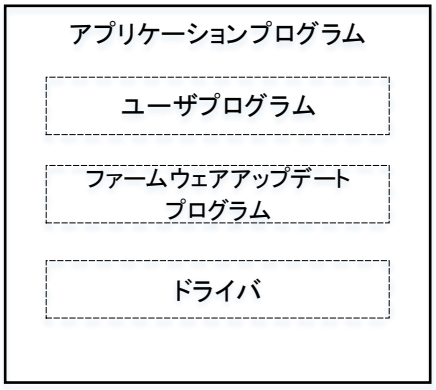

図 1.3 アプリケーションプログラムの構成

1.5.2 ユーザプログラム

ユーザプログラムはアプリケーションプログラム内のお客様のプログラムとなります。お客様のプログラムとファームウェアアップデートプログラム、ドライバ、セキュアブートプログラム（128Kbyte）の合計サイズが 1Mbyte 以下になるように作成してください。（図 1.4）ファームウェアアップデートプログラムのサイズについては 3.9 章（RTOS）、4.9 章（NonOS）を参照してください。

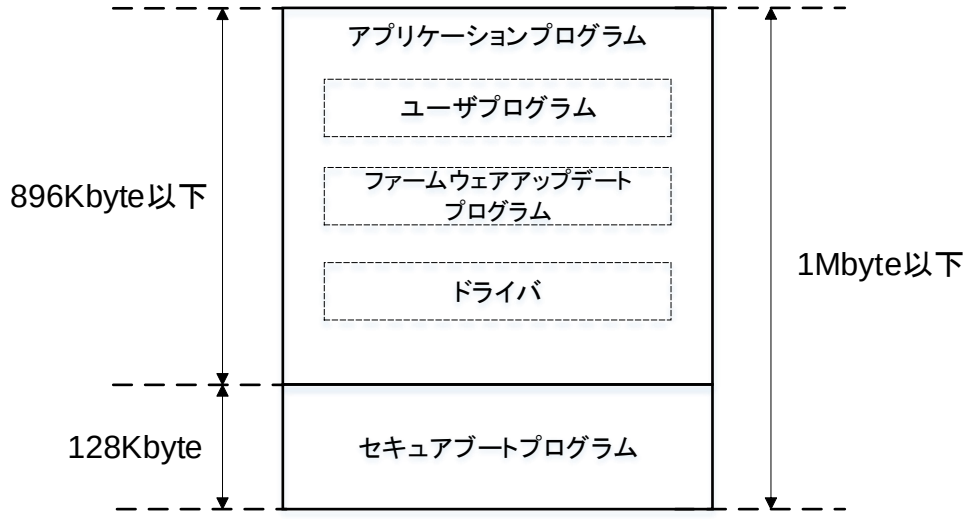

図 1.4 アプリケーションプログラムサイズ

1.5.3 ファームウェアアップデートプログラム

ファームウェアアップデートプログラムは仮想サーバと Ethernet もしくは非同期シリアルを介して通信を行いファームウェアの認証および更新を行います。（図 1.5）ファームウェアアップデートプログラムは通信処理やファームウェア認証を行うための暗号処理などが含まれています。

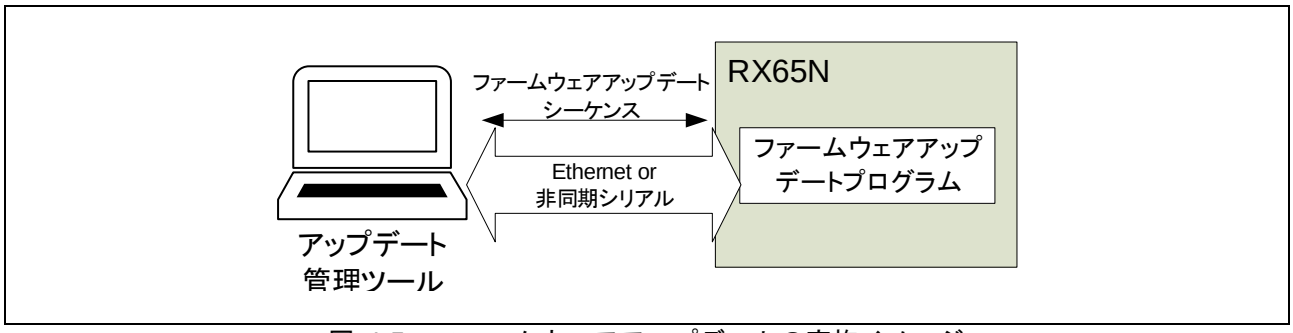

図 1.5 ファームウェアアップデートの実施イメージ

ファームウェアアップデートプログラムの処理の流れを図 1.6 に示します。
仮想サーバとファームウェアアップデートコマンド（以降、アップデートコマンド）を介してファームウェアの認証およびアップデートを行います。

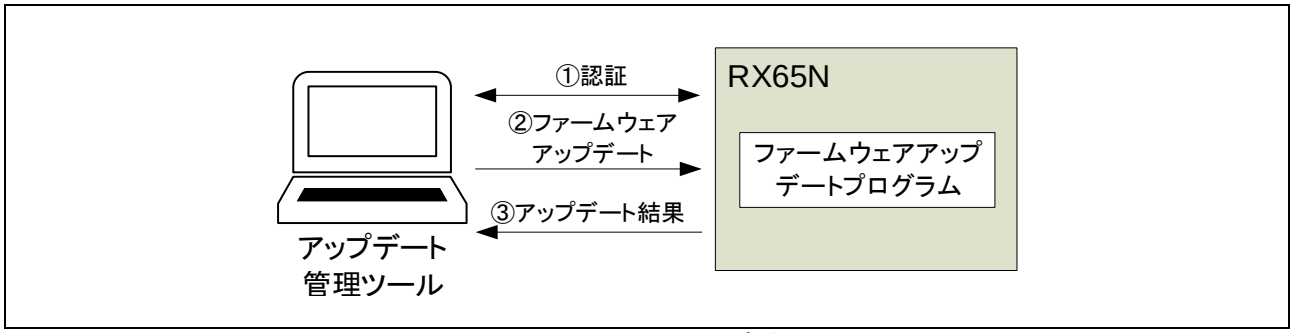

図 1.6 ファームウェアアップデートの流れ

ファームウェアアップデートプログラムについての詳細は 3 章、4 章を参照してください。

1.5.4 ドライバ

ファームウェアアップデートプログラムで使用する周辺機能を図 1.7 と表 1.9 に示します。各機能の詳細については RX65N グループ、RX651 グループ ユーザーズマニュアル ハードウェア編（R01UH0590JJ020）を参照してください。

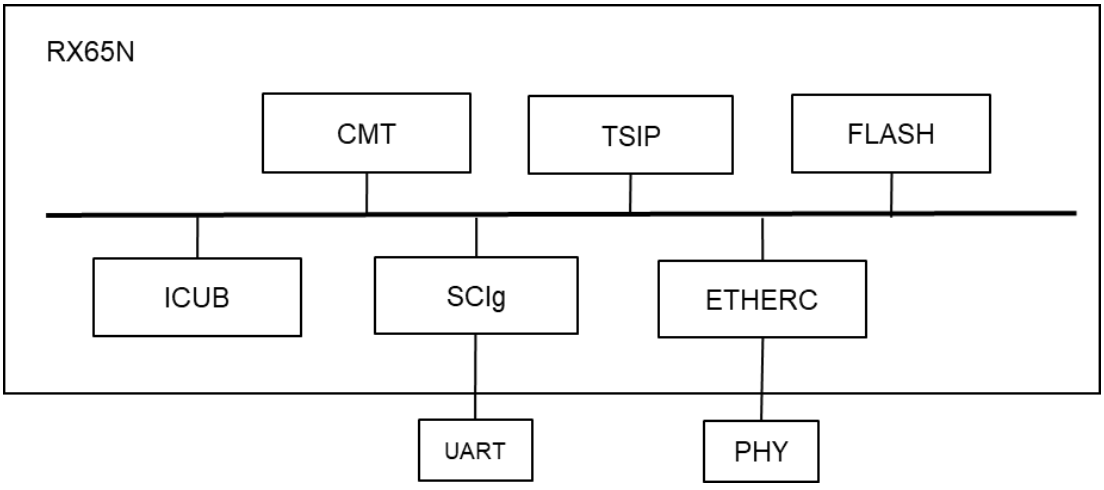

図 1.7 周辺機器一覧

表 1.9 周辺機能概要

周辺機能名	説明
CMT	コンペアマッチタイマ 周期タイマ処理を実施する
TSIP	Trusted Secure IP 暗号処理を実施する
FLASH	フラッシュメモリ コードフラッシュ・データフラッシュへの書き込みを実施する
ICUB	割り込みコントローラ 各周辺機能の割り込み制御を実施する
SCIg	シリアルコミュニケーションインタフェース 非同期シリアル通信を実施する。詳細については 5.6.2 章を参照
ETHERC	イーサネットコントローラ Ethernet 通信を実施する。詳細については 5.6.1 章を参照

本ソリューションで提供しているサンプルプログラムではドライバとして FIT を使用しています。使用している FIT の一覧を表 1.10 に示します。詳細についてはそれぞれのアプリケーションノートを参照してください。

表 1.10 使用している FIT の一覧

周辺機能名	FIT
CMT	RX ファミリ CMT モジュール
TSIP	RX ファミリ TSIP モジュール
FLASH	RX ファミリ フラッシュモジュール
ICUB	RX ファミリ ボードサポートパッケージモジュール
SCIg	RX ファミリ バイト型キューバッファ（BYTEQ） RX ファミリ SCI モジュール
ETHERC	RX ファミリ イーサネットモジュール RX ファミリ システムタイマモジュール RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny

1.5.5 セキュアブートプログラム

セキュアブートプログラムはデバイスのパワーオンリセット時にアプリケーションプログラムの検証を行い、問題がなければアプリケーションプログラムを起動します。
アプリケーションプログラムの検証について以下に示します。

① 起動バンクの検証
　成功：エントリポイントに登録されているアプリケーションプログラムを起動
　失敗：②を実施
② 非起動バンクの検証
　成功：③を実施
　失敗：起動不能状態
③ 起動バンクを切り替えてソフトウェアリセットをし、①を実施

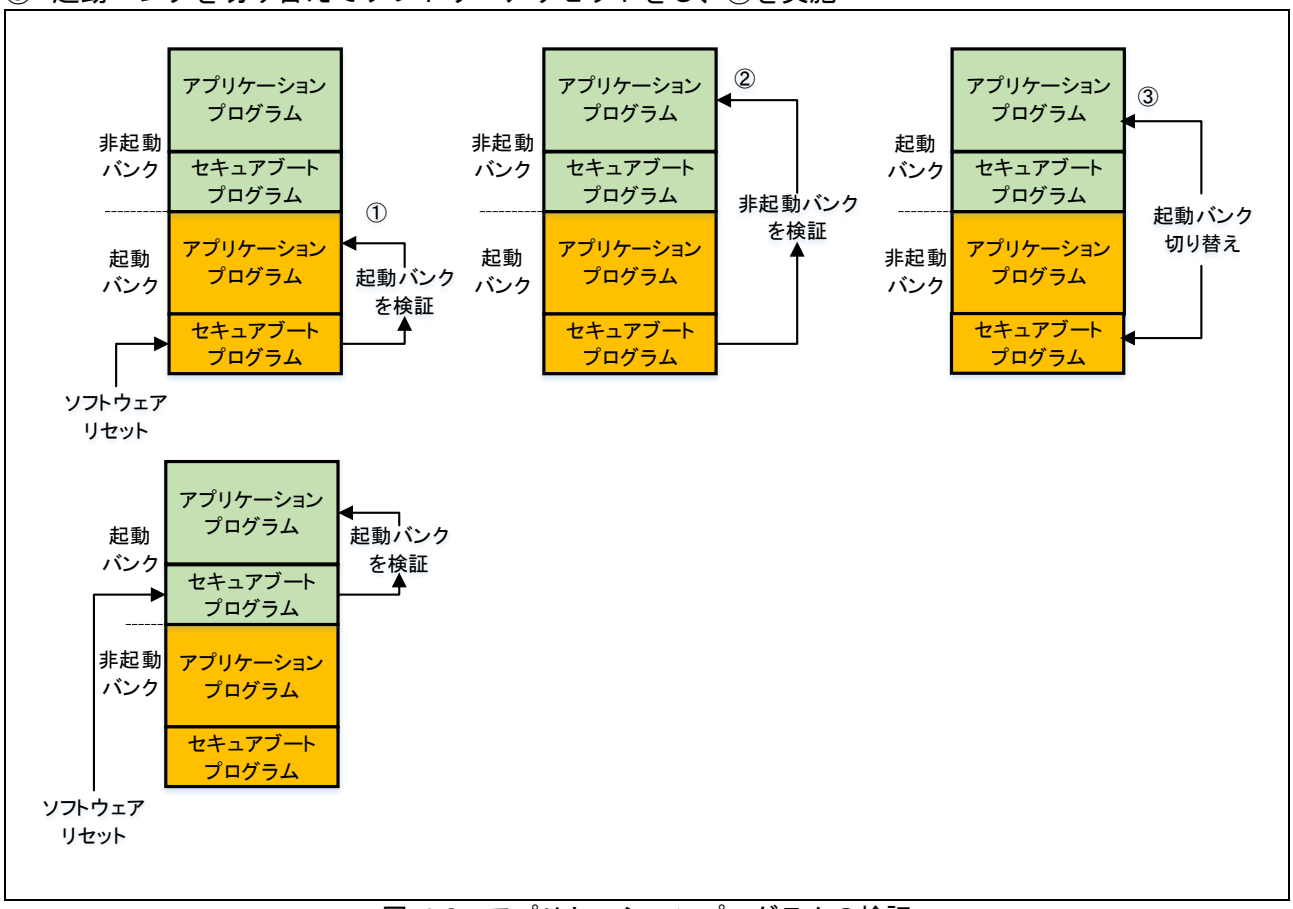

図 1.8　アプリケーションプログラムの検証

起動バンク・非起動バンクの両側の検証に失敗した場合は、起動不能状態になります。起動不能状態から復旧させるときは、フラッシュプログラマ等によるプログラムの書き換えを行う必要があります。

図 1.9 にセキュアブートプログラムとアプリケーションプログラムのメモリ配置を示します。セキュアブートプログラムは固定アドレスに書き込まれている必要があります。
セキュアブートプログラムの詳細については 2 章を参照してください。

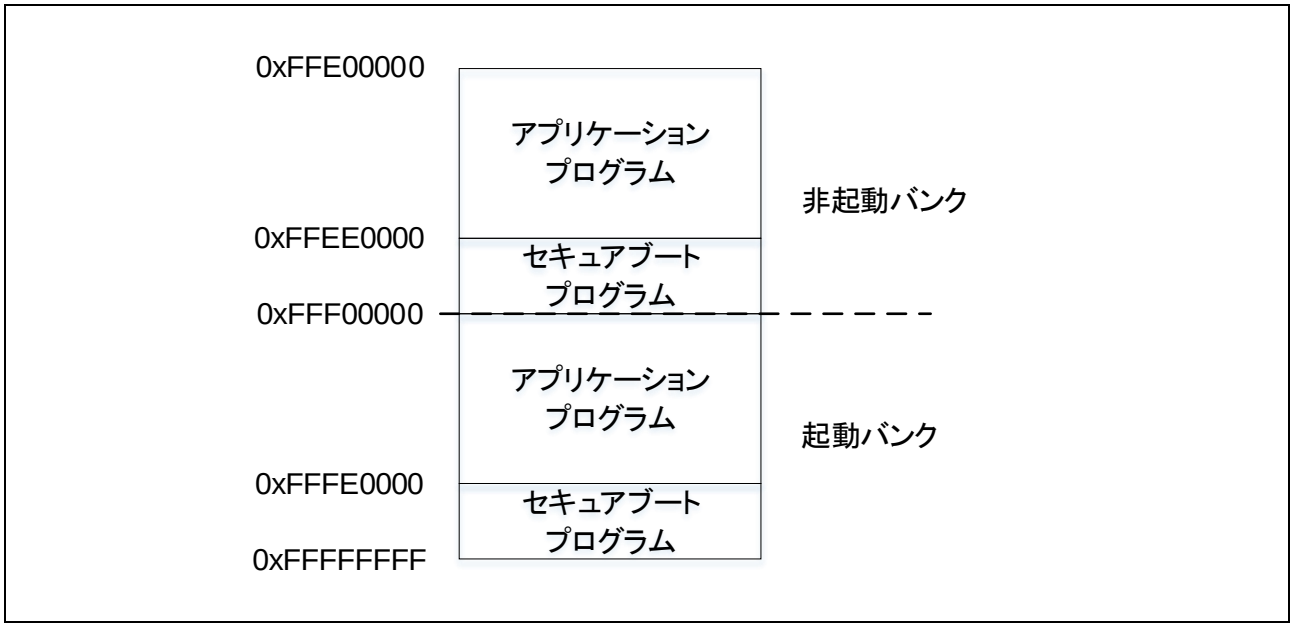

図 1.9 セキュアブートプログラムとアプリケーションプログラムのメモリ配置

2. セキュアブートプログラム

2.1 動作概要

セキュアブートプログラムは起動時にハッシュ関数を使用した管理領域の改ざん確認と、CMAC によるアプリケーションプログラムの検証を行います。管理領域の改ざん確認で異常が検出された場合は起動不能状態となります。管理領域の改ざん確認結果が問題ない場合、起動バンクのアプリケーションプログラムの検証を行います。検証結果が問題ない場合はアプリケーションプログラムを起動します。検証結果が異常な場合は非起動バンクの認証を行います。非起動バンクの検証結果に問題がない場合は起動バンクを切り替えてソフトウェアリセットを行います。
起動バンク、非起動バンクとも検証結果が異常であった場合は起動不能状態になります。
起動不能状態から復旧させるときは、フラッシュプログラマでプログラムを書き換える必要があります。
図 2.1 にセキュアブートプログラムの概略動作フローを示します。

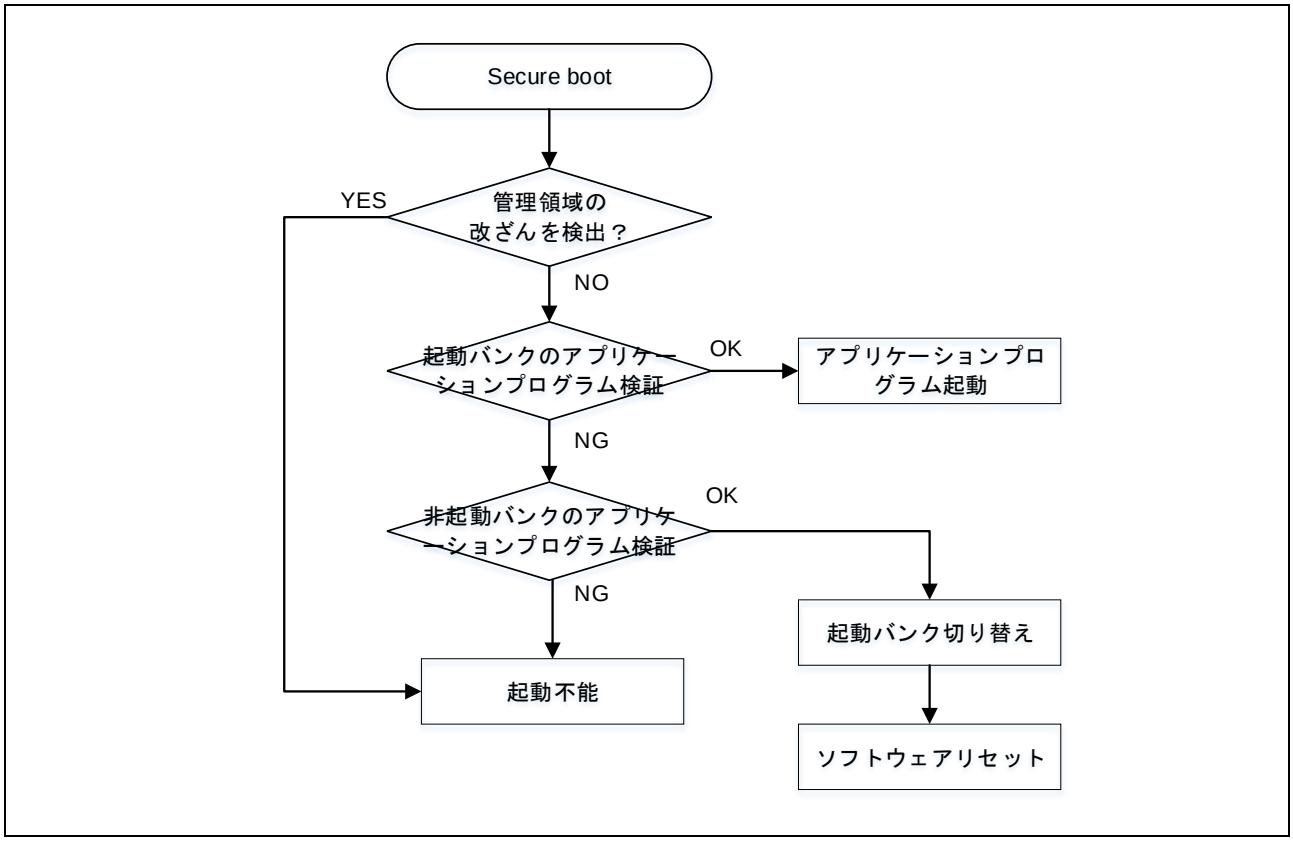

図 2.1 セキュアブートプログラムの概略動作フロー

2.2 ディレクトリ構成

セキュアブートはプログラムの提供ではなく、mot ファイルでの提供となります。表 2.1 にディレクトリ構成を示します。

表 2.1 ディレクトリ構成

フォルダ名 / ファイル名	説明
nonOS_rx65n_secure_boot¥	
nonOS_rx65n_secure_boot.mot	NonOS 版セキュアブート mot ファイル
rx65n_secure_boot¥	
rx65n_secure_boot.mot	RTOS 版セキュアブート mot ファイル

2.3 割り込み

表 2.2 にセキュアブートプログラムで使用する割り込み要因一覧を示します。優先度は 1～15 までとなっており、数値が高いほど優先度が高くなります。

表 2.2 割り込み要因一覧

セキュアブートプログラムファイル名	機能	名称	ベクタ番号	優先度
rx65n_secure_boot.mot	FLASH	FIFERR	21	13
		FRDYI	23	13
nonOS_rx65n_secure_boot.mot	FLASH	FIFERR	21	2
		FRDYI	23	2

2.4 最大スタックサイズ

表 2.3、にセキュアブートプログラムで使用する最大スタックサイズを示します。

表 2.3 最大スタックサイズ

セキュアブートプログラムファイル名	セクション名	サイズ
rx65n_secure_boot.mot	ユーザスタック：SURI_STACK	404byte
	割り込みスタック：SI	116byte
nonOS_rx65n_secure_boot.mot	ユーザスタック：SU	404byte
	割り込みスタック：SI	52byte

3. ファームウェアアップデートプログラム（RTOS 版）

3.1 動作概要

アプリケーションプログラム起動時に、RTOS カーネルを起動し、アプリケーションの初期設定後にファームウェアアップデートプログラムのタスクとユーザプログラムのタスクを起動します。図 3.1 に概略動作フローを示します。
ファームウェアアップデートプログラムのタスクを起動すると、仮想サーバからアップデートコマンドが送信されるのを待ち、受信した場合にファームウェアアップデート処理を開始します。

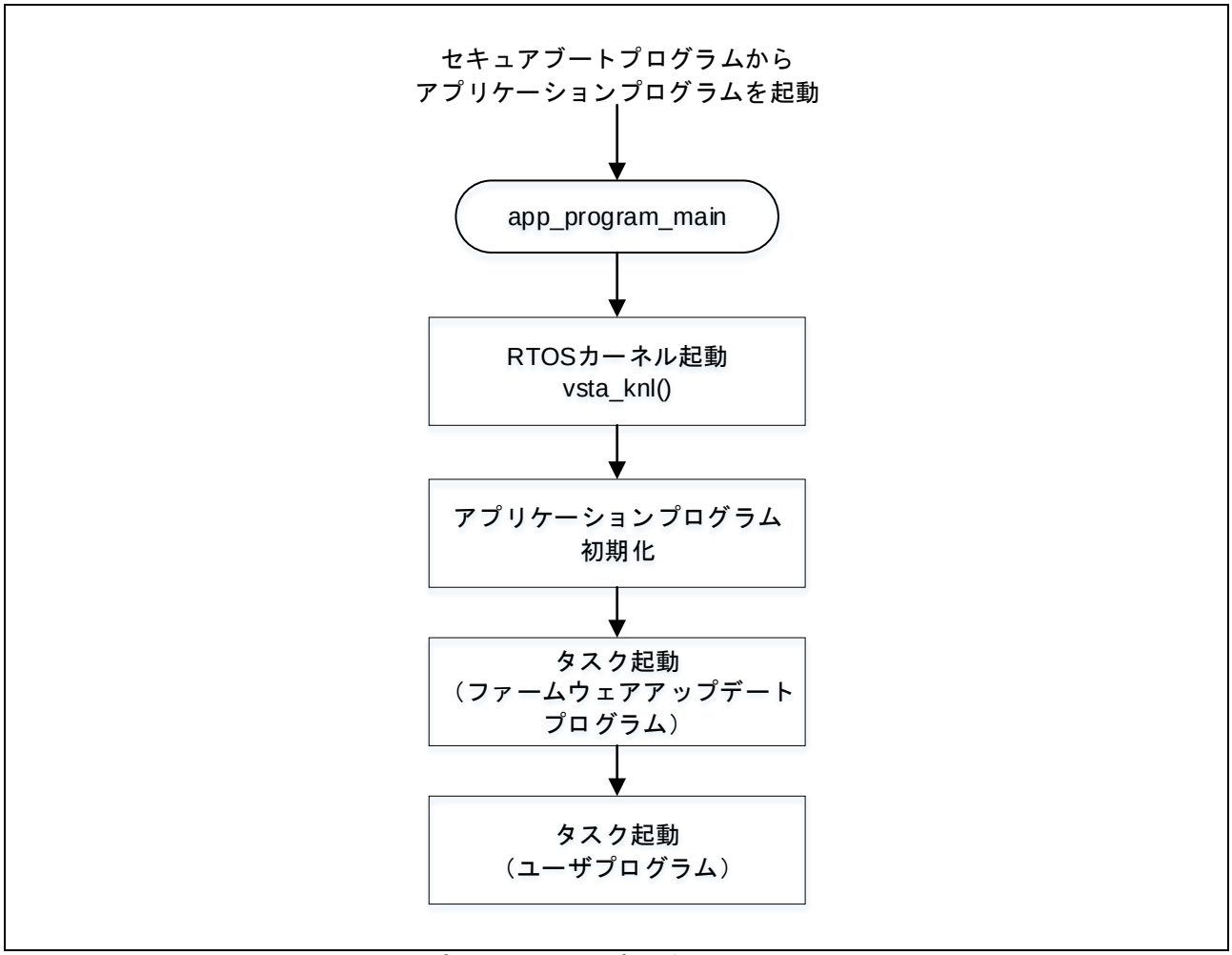

図 3.1 アプリケーションプログラムの概略動作フロー

3.2 ソフトウェア構成

本ソリューションで提供しているサンプルプログラムのソフトウェア構成を図 3.2 に示します。

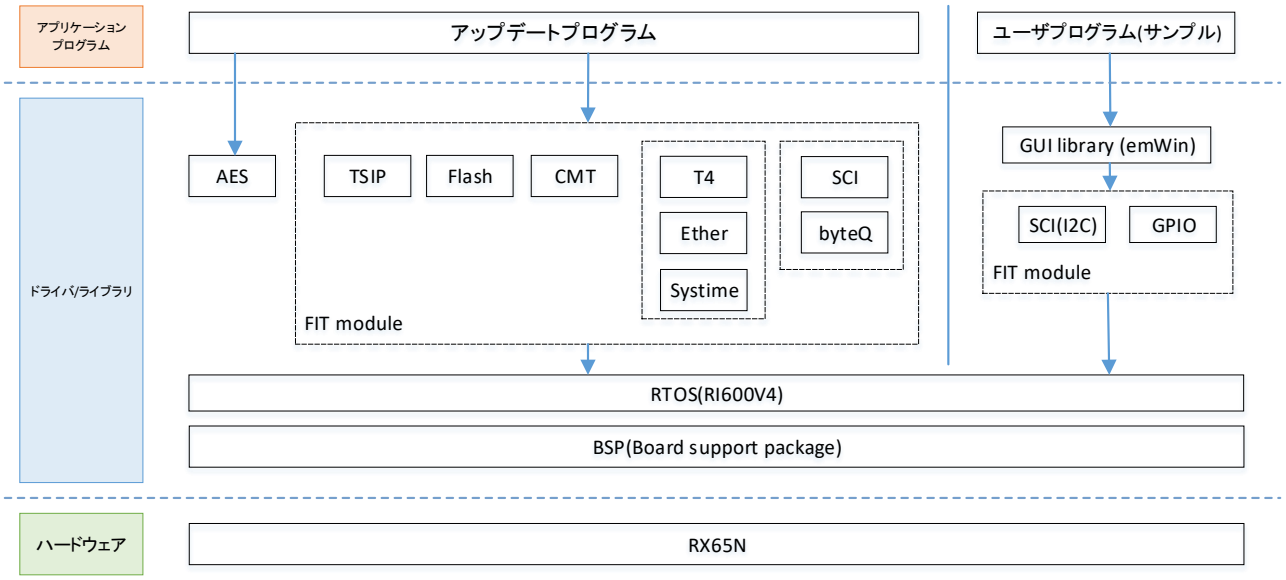

図 3.2 ソフトウェア構成図

3.3 ディレクトリ構成

本ソリューションで提供しているサンプルプログラムのディレクトリ構成を表 3.1 を示します。

表 3.1 ディレクトリ構成

フォルダ名 / ファイル名	説明
rx65n_app_prog	
¥cd_data_aes_library_rx_v104r00	RX ファミリ AES ライブラリ（バイナリ提供）
¥cfg_output	RTOS ヘッダファイル（自動生成）
¥generate	RTOS コンフィグ
¥r_bsp	FIT（BSP）
¥r_byteq	FIT（BYTEQ）
¥r_cmt_rx	FIT（CMT）
¥r_config¥	FIT（コンフィグレーション）
¥r_ether_rx	FIT（Ethernet）
¥r_flash_rx	FIT（Flash）
¥r_gpio_rx	FIT（GPIO）
¥r_pincfg	PIN コンフィグレーション
r_pinset.c	PIN 設定
r_pinset.h	PIN 設定 ヘッダ
¥r_sci_iic_rx	FIT（簡易 I2C）
¥r_sci_rx	FIT（SCI）
¥r_sys_time	FIT（SystemTime）
¥r_t4_driver_rx	FIT（T4 インタフェース変換）
¥r_t4_rx	FIT（T4 プロトコルスタック）
¥r_tsip_rx	FIT（TSIP）
¥src	アプリケーションプログラムのフォルダ
¥firm_update	ファームウェアアップデートプログラムのフォルダ
command.c	アップデートコマンド
command.h	アップデートコマンド ヘッダ
firm_update.c	ファームウェア更新
firm_update.h	ファームウェア更新 ヘッダ
uart.c	非同期シリアル
uart.h	非同期シリアル ヘッダ
ether.c	Ethernet（UDP、TCP）通信
ether.h	Ethernet（UDP、TCP）通信 ヘッダ
flash_api_bgo_wrapper.c	Flash
flash_api_bgo_wrapper.h	Flash ヘッダ
base64_decode.c	Base64 デコード
base64_decode.h	Base64 デコード ヘッダ
¥uitron	RTOS のフォルダ
hw_control.h	ハードウェア制御用定義
rtos_sample_config.h	サンプルコンフィグレーション
sysdwn.c	システムダウン
task.c	メインタスク
app_prog_main.c	アプリケーションプログラムメイン
app_prog_main.h	アプリケーションプログラムメイン ヘッダ
user_task.c	ユーザタスク
user_task.h	ユーザタスク ヘッダ
¥user	ユーザプログラムのフォルダ
¥Config	emWin コンフィグ

¥GUI	emWin ライブラリヘッダ
¥GUI_APP	emWin アプリケーション
WindowDLG2.c	GUI 表示

表 3.2 にプリプロセッサ・マクロ一覧を示します。

表 3.2　プリプロセッサ・マクロ一覧

プリプロセッサ・マクロ名	値	説明
__RX	1 固定	製品ラインナップ＝RX ファミリ

3.4 セクション配置

本ソリューションで提供しているサンプルプログラムのセクション配置を表 3.3 に示します。

注意事項
・※は移動することができません。
・0xFFFE0000 - 0xFFFFFFFF にアプリケーションプログラムを配置することはできません。

表 3.3 アプリケーションプログラムのセクション配置（RTOS 版）

アドレス	※移動不可	セクション名	説明
RAM	0x00000100	B_GUI_WORK	GUI ライブラリワーク領域
データ FLASH	0x00100000※	B_PRODUCTS_1	プロダクト情報
	0x00100020※	B_UID	ユニーク ID
	0x00100030※	B_INSTALL_KEY	暗号化されたインストール鍵束
	0x00100230※	B_MSK1	MSK1 用 index 格納領域
	0x00100274※	B_MSK2	MSK2 用 index 格納領域
	0x001002C0※	D_FIRMWARE_UPDATE_CONTROL_BLOCK	管理領域（メイン）
	0x00100340※	D_FIRMWARE_UPDATE_CONTROL_BLOCK_MIRROR	管理領域（ミラー）
	0x001003C0※	B_VERSION_1	プログラムバージョン（起動バンク）
	0x001003C4※	B_PREV_VERSION_1	プログラムバージョン（非起動バンク）
	0x00100400	D_MAC_ADDR_1	MAC アドレス
拡張 RAM	0x00840000	SI	割り込みスタック領域
		SURI_STACK_1	RTOS スタック領域
		B_ETHERNET_BUFFERS_1	ETHER 送信バッファおよび受信バッファ領域
		B_RX_DESC_1	ETHER 受信ディスクリプタ領域
		B_TX_DESC_1	ETHER 送信ディスクリプタ領域
		B_1	1byte 未初期化データ領域
		R_1	1byte 未初期化データ領域（変数）
		B_2	2byte 未初期化データ領域
		R_2	2byte 未初期化データ領域（変数）
		B	4byte 未初期化データ領域
		R	4byte 未初期化データ領域（変数）
		R_STDLIB*	標準ライブラリ 初期値なしデータ
		B_STDLIB*	標準ライブラリ 未初期化データ
		RPFRAM*	FLASH FIT 初期化データ領域
		BRI_RAM	RTOS 1byte 未初期化データ領域
		BRI_RAM_1	RTOS 2byte 未初期化データ領域
		BSECURE_FW_UPDATE*	ファームウェアアップデートプログラム未初期化データ領域
		RSECURE_FW_UPDATE*	ファームウェアアップデートプログラム未初期化データ領域（変数）
コード FLASH	0xFFF00000	C_1	1byte 定数領域
		C_2	2byte 定数領域
		C	4byte 定数領域
		D	4byte 初期化データ領域
		D_1	1byte 初期化データ領域
		D_2	2byte 初期化データ領域
		P	プログラム領域
		CRI_ROM_2	RTOS 4byte 定数領域

		CRI_ROM_1	RTOS　2byte 定数領域
		CRI_ROM	RTOS　1byte 定数領域
		PRI_KERNEL	RTOS　プログラム領域
		W*	switch 文分岐テーブル領域
		L	リテラル領域
		PFRAM*	FLASH FIT　プログラム領域
		P_STDLIB*	標準ライブラリ　プログラム領域
		C_STDLIB*	標準ライブラリ　定数領域
		L_STDLIB*	標準ライブラリ　リテラル領域
		D_STDLIB*	標準ライブラリ　初期化データ領域
		C$*	C$DEC、C$BSEC、C$VECT の定数領域
	0xFFFDD000	PSECURE_FW_UPDATE*	ファームウェアアップデートプログラム プログラム領域
		CSECURE_FW_UPDATE*	ファームウェアアップデートプログラム 定数領域
		DSECURE_FW_UPDATE*	ファームウェアアップデートプログラム 初期化データ用域
	0xFFFDFBFC※	APP_RESETVECTOR	アプリケーションプログラムのエントリアドレス
	0xFFFDFC00	INTERRUPT_VECTOR	可変割り込みベクタ領域
	0xFFFFFF80※	FIX_INTERRUPT_VECTOR	固定割り込みベクタ領域

3.5　使用端子一覧

ファームウェアアップデートプログラムで使用する端子一覧を表 3.4 に示します。

表 3.4　端子一覧

モジュール/機能	端子名	入出力	内容
SCI8	TXD8	出力	SCI8 の送信データ出力端子
	RXD8	入力	SCI8 の受信データ入力端子
ETHERC （MII）	ET0_TX_CLK	入力	送信クロック ET0_TX_EN、ET0_ETXD3～ET0_ETXD0、 ET0_TX_ER 信号出力時のタイミング基準信号
	ET0_RX_CLK	入力	受信クロック ET0_RX_DV、ET0_ERXD3～ET0_ERXD0、 ET0_RX_ER 信号入力時のタイミング基準信号
	ET0_TX_EN	出力	送信データ有効 ET0_ETXD3～ET0_ETXD0 上に有効な送信データが出力されていることを示す信号
	ET0_ETXD3～ ET0_ETXD0	出力	4 ビットの送信データ
	ET0_TX_ER	出力	送信エラー 送信中のエラーを PHY-LSI に通知するための信号
	ET0_RX_DV	入力	受信データ有効 ET0_ERXD3～ET0_ERXD0 上に有効な受信データがあることを示す信号
	ET0_ERXD3～ ET0_ERXD0	入力	4 ビットの受信データ
	ET0_RX_ER	入力	受信エラー PHY-LSI から ETHERC へ転送中のフレームにエラーがあることを示す信号
	ET0_CRS	入力	キャリア感知

	ET0_COL	入力	衝突検出
	ET0_MDC	出力	マネジメントデータクロック ET0_MDIO による情報転送用の基準クロック信号
	ET0_MDIO	入出力	マネジメントデータ I/O PHY-LSI との間で管理情報を交換するための双方向データ信号
	ET0_LINKSTA	入力	PHY-LSI からのリンクステータス入力

3.6 割り込み

表 3.5 にファームウェアアップデートプログラムで使用している割り込み一覧を以下に示します。

表 3.5 割り込み一覧

割り込み要求発生元	名称	ベクタ番号	優先度
CMT	CMI0	28	13
FLASH	FIFERR	21	13
	FRDYI	23	13
SCI8	RXI8	100	14
	TXI8	101	14
	BL1.TEI8	111	3
Ethernet	AL1.EINT0	113	2

3.7 エントリアドレス設定

セキュアブートプログラムから呼び出す関数をセクション（APP_RESETVECTOR）に登録します。
図 3.3 にエントリアドレスの設定箇所を示します。

```
rx65n_app_prog/r_bsp/board/generic_rx65n/vecttbl.c
---
#pragma section C USER_RESETVECTOR
void (* const Reset_Vector[])(void) =
{
  app_prog_main     ★セキュアブートプログラムから呼び出す関数を登録します。
};
```

図 3.3 アプリケーションプログラムのエントリアドレス設定

3.8 固定割り込みベクタテーブル

表 3.6 に固定割り込みベクタテーブルの定義を示します。

表 3.6 固定割り込みベクタテーブル

ベクタアドレス	ベクタ番号	要因	省略時の扱い
0xFFFFFF80	0	エンディアン選択レジスタ	コンパイラの endian オプションに応じて，以下が設定されます。 - “-endian=little” の場合 0xFFFFFFFF - “-endian=big” の場合 0xFFFFFFF8
0xFFFFFF84	1	（予約領域）	0xFFFFFFFF
0xFFFFFF88	2	オプション機能選択レジスタ 1	
0xFFFFFF8C	3	オプション機能選択レジスタ 0	
0xFFFFFF90	4	（予約領域）	
0xFFFFFF94	5	（予約領域）	
0xFFFFFF98	6	（予約領域）	
0xFFFFFF9C	7	ROM コード・プロテクト（フラッシュメモリ）	
0xFFFFFFA0	8	オンチップ・デバッガ ID コード・プロテクト（フラッシュメモリ）	
0xFFFFFFA4	9		
0xFFFFFFA8	10		
0xFFFFFFAC	11		
0xFFFFFFB0	12	（予約領域）	
0xFFFFFFB4	13	（予約領域）	
0xFFFFFFB8	14	（予約領域）	

0xFFFFFFBC	15	（予約領域）	
0xFFFFFFC0	16	（予約領域）	
0xFFFFFFC4	17	（予約領域）	
0xFFFFFFC8	18	（予約領域）	
0xFFFFFFCC	19	（予約領域）	
0xFFFFFFD0	20	特権命令例外	
0xFFFFFFD4	21	アクセス例外	
0xFFFFFFD8	22	（予約領域）	
0xFFFFFFDC	23	未定義命令例外	
0xFFFFFFE0	24	（予約領域）	
0xFFFFFFE4	25	浮動小数点例外	
0xFFFFFFE8	26	（予約領域）	
0xFFFFFFEC	27	（予約領域）	
0xFFFFFFF0	28	（予約領域）	
0xFFFFFFF4	29	（予約領域）	
0xFFFFFFF8	30	ノンマスカブル割り込み	
0xFFFFFFFC	31	リセット	PowerON_Reset_PC()

詳細については RI600V4 リアルタイム・オペレーティング・システム　ユーザーズマニュアル　コーディング編（R20UT0711JJ0104）を参照してください。

3.9　プログラムサイズ

表 3.7 にファームウェアアップデートプログラムサイズを示します。ユーザプログラムは含みません。

表 3.7　ファームウェアアップデートのプログラムサイズ

データ種類	サイズ
RAMDATA	72,589 byte
ROMDATA	24,219 byte
PROGRAM	209,351 byte

3.10　最大スタックサイズ

表 3.8 に最大スタックサイズを示します。

表 3.8　最大スタックサイズ

スタック種別	最大スタックサイズ
ユーザスタック：SURI_STACK	1004byte
割り込みスタック：SI	116byte

3.11　RTOS

本章は RX ファミリ用リアルタイム OS[RI600V4]使用を前提としています。その他の OS を利用する場合は変更点を十分に確認して適用してください。

3.11.1 タスク一覧

ファームウェアアップデートプログラムのタスク一覧を表 3.9 に示します。
全てのタスクについて同じ優先度を設定してください。サンプルプログラムは「3」となっています。

表 3.9 ファームウェアアップデートプログラムのタスク一覧

タスク名	概要	優先度
tsk_update_command()	Ethernet、非同期シリアルでアップデートコマンド受信しアップデート処理を行います。	3
tsk_send_res3tsk2_on_sci()	アップデート結果を非同期シリアルで送信します。	3

3.11.2 タスク管理

ファームウェアアップデートプログラムのタスクはイベントフラグを用いてタスク管理を行っています。アップデートコマンドの受信契機でイベントフラグをセットしタスクをアクティブにします。
図 3.4 に Ethernet でアップデートコマンドを受信したときの動作、図 3.5 に非同期シリアルでアップデートコマンドを受信したときの動作を示します。

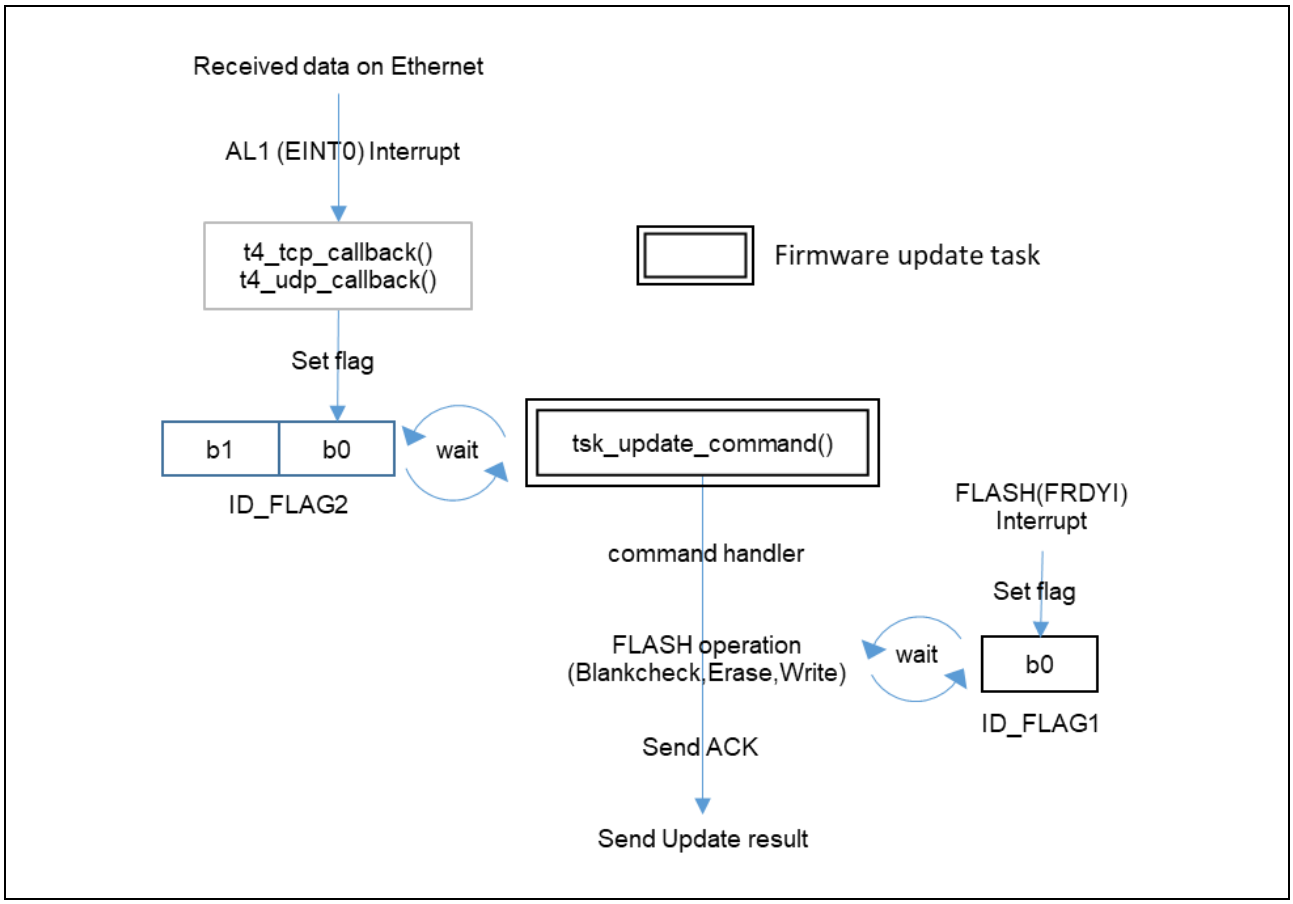

図 3.4 イベントフラグとタスクの関係について（Ethernet）

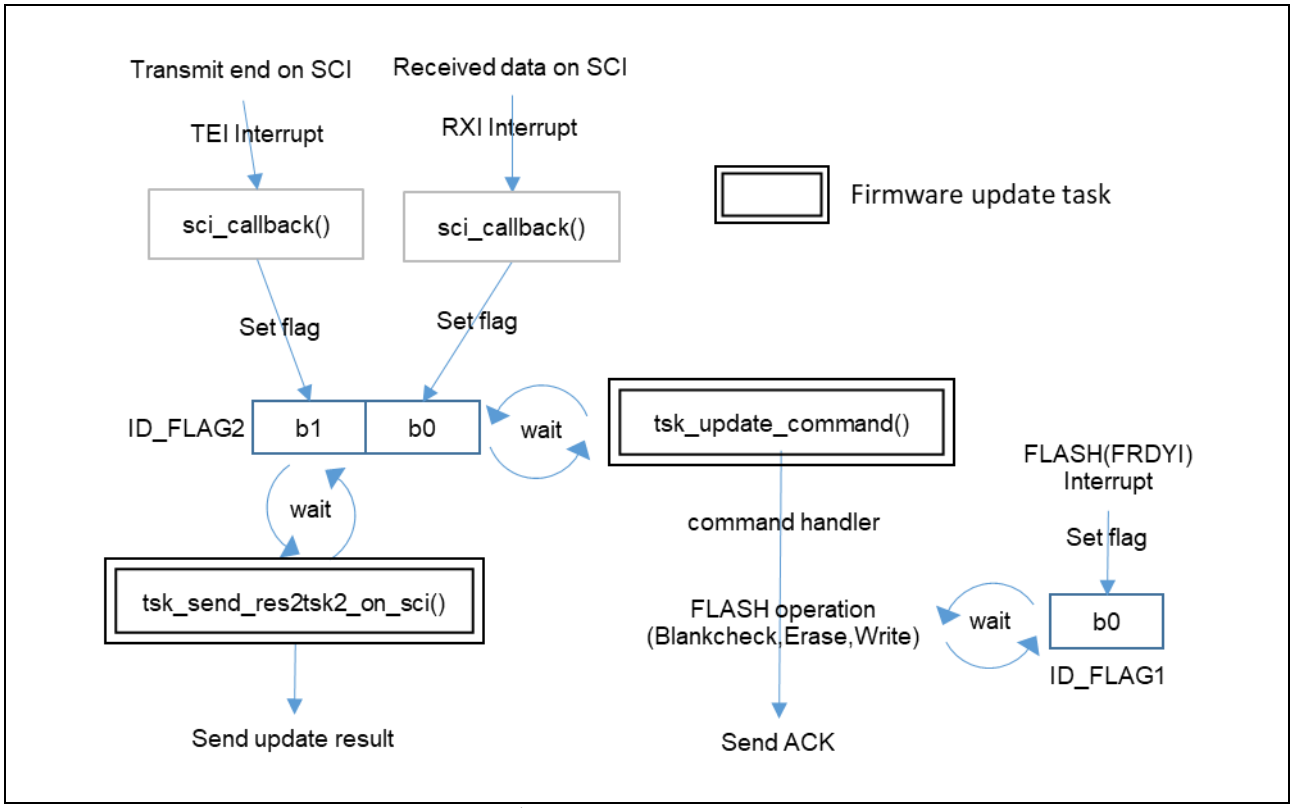

図 3.5　イベントフラグとタスクの関係について（非同期シリアル）

3.11.3　周期タイマ

表 3.10 にファームウェアアップデートプログラムで使用する周期タイマの一覧を示します。

表 3.10　周期タイマー覧

ID	値（ms）	説明
ID_CYC_LAN_STATUS	10	Ethernet のステータス取得
ID_CYC_WAKEUP_TSK_TCP	100	TCP 通信端点の監視

3.11.4　アラーム

表 3.11 にファームウェアアップデートプログラムで使用するアラームの一覧を示します。

表 3.11　アラーム一覧

ID	値（ms）	説明
ID_ALM1	5000	非同期シリアルでアップデートコマンドを受信した場合のガードタイマ

4. ファームウェアアップデートプログラム（NonOS 版）

4.1 動作概要

ファームウェアアップデートプログラムを起動するとアップデートコマンド受信待ちのメインループに入ります。Ethernet、非同期シリアルでアップデートコマンドを受信すると割り込みハンドラ内で受信フラグを設定します。メインループはこの受信フラグを常に監視し、受信フラグが設定されたときにディスパッチャを呼び出します。

ファームウェアアップデートプログラムを起動するとメインループから抜けないため、ユーザプログラムを同時に動作させることはできません。したがってアプリケーションプログラムの起動時にユーザ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いずれかを選択する必要があります。
サンプルプログラムでは SW1 を用いて起動プログラムを選択しています。
図 4.1 にアプリケーションプログラムの概略動作フローを、図 4.2 にファームウェアアップデートプログラムのメインループ動作概要フローを示します。

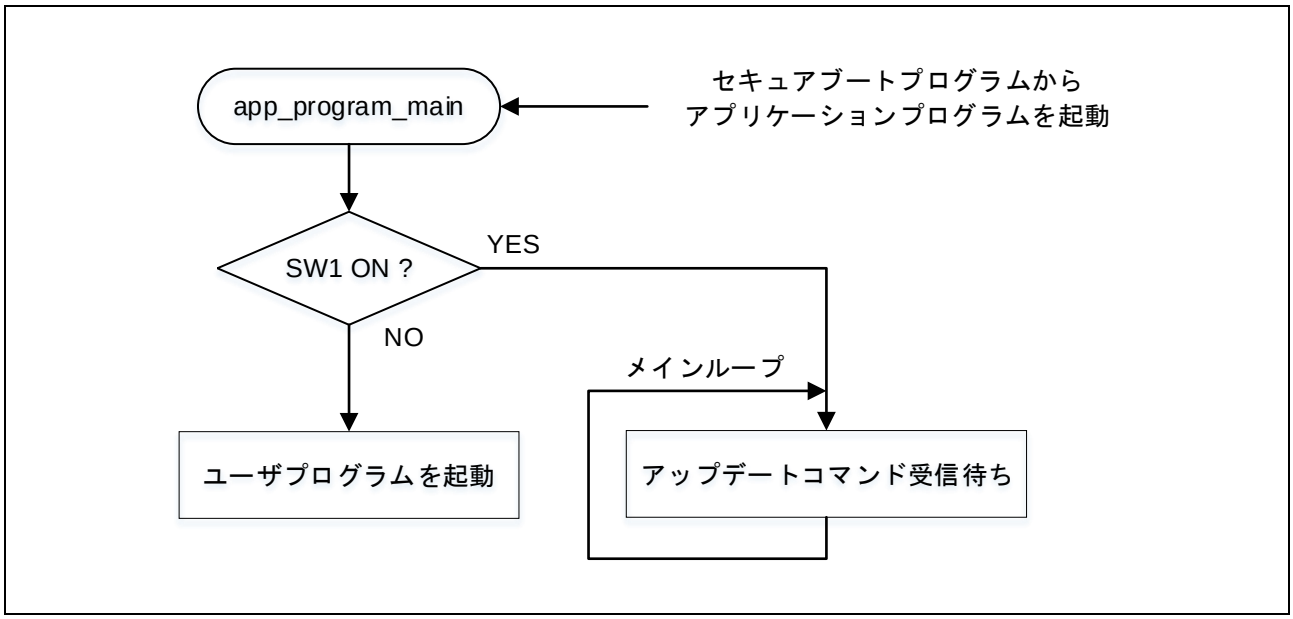

図 4.1 アプリケーションプログラムの動作概略フロー

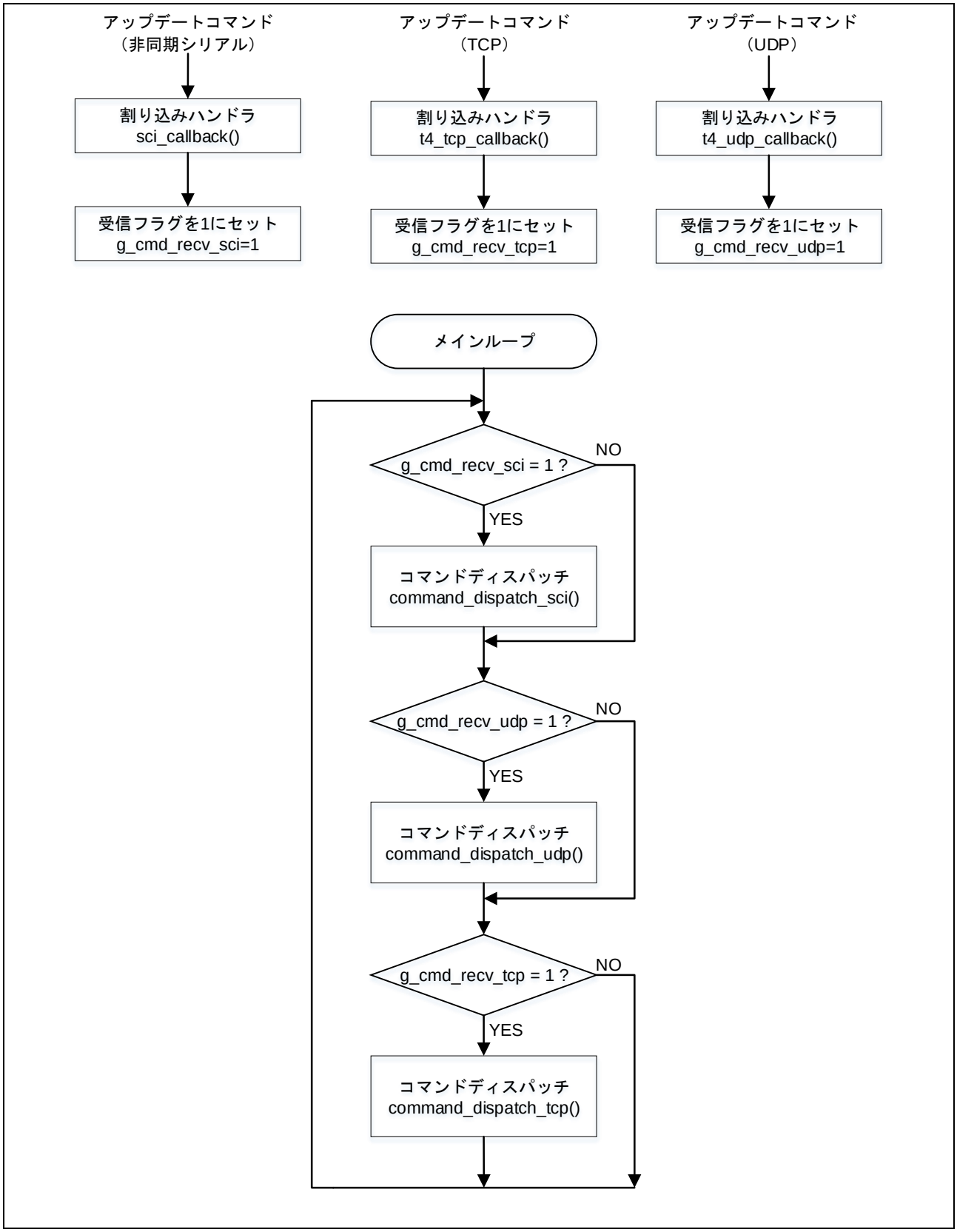

図 4.2　ファームウェアアップデートプログラムのメインループ動作概要フロー

4.2 ソフトウェア構成

本ソリューションで提供しているサンプルプログラムのソフトウェア構成を図 4.3 に示します。

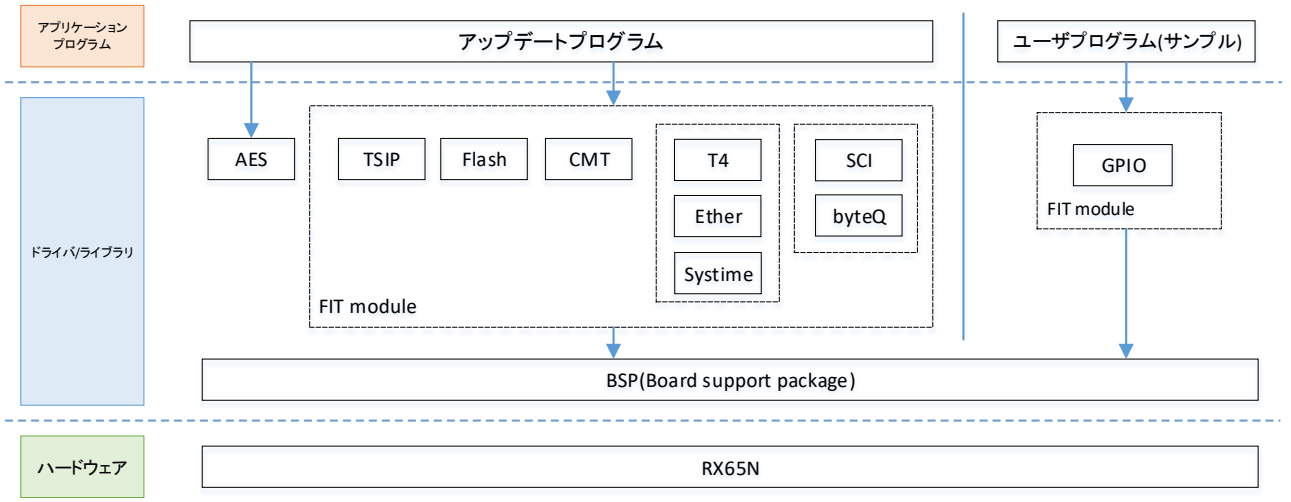

図 4.3 ソフトウェア構成図

4.3 ディレクトリ構成

本ソリューションで提供しているサンプルプログラムのディレクトリ構成を表 4.1 に示します。

表 4.1 ディレクトリ構成

フォルダ名 / ファイル名	説明
nonOS_rx65n_app_prog¥	
cd_data_aes_library_rx_v104r00¥	RX ファミリ AES ライブラリ（バイナリ提供）
r_bsp¥	FIT（BSP）
r_byteq¥	FIT（BYTEQ）
r_cmt_rx¥	FIT（CMT）
r_config¥	FIT（コンフィグレーション）
r_ether_rx¥	FIT（Ethernet）
r_flash_rx¥	FIT（Flash）
r_gpio_rx¥	FIT（GPIO）
r_pincfg¥	PIN コンフィグレーション設定
r_pinset.c	PIN 設定
r_pinset.h	PIN 設定 ヘッダ
r_sci_rx¥	FIT（SCI）
r_sys_time¥	FIT（SystemTime）
r_t4_driver_rx¥	FIT（T4 インタフェース変換）
r_t4_rx¥	FIT（T4 プロトコルスタック）
r_tsip_rx¥	FIT（TSIP）
src¥	アプリケーションプログラムのフォルダ
firm_update¥	ファームウェアアップデートプログラムのフォルダ
base64_decode.c	Base64 デコード
base64_decode.h	Base64 デコード ヘッダ
command.c	アップデートコマンド
command.h	アップデートコマンド ヘッダ
ether.c	Ethernet（UDP、TCP）通信
ether.h	Ethernet（UDP、TCP）通信 ヘッダ
firm_update.c	ファームウェア更新
firm_update.h	ファームウェア更新 ヘッダ
flash_api_bgo_wrapper.c	Flash
flash_api_bgo_wrapper.h	Flash ヘッダ
lcd_conf.c	LCD コントローラ設定
r_ascii.c	ASCII フォントデータ
r_ascii.h	ASCII フォントデータ ヘッダ
r_simple_graphic.c	LCD 表示
r_simple_graphic_if.h	LCD 表示 ヘッダ
uart.c	非同期シリアル
uart.h	非同期シリアル ヘッダ
user¥	ユーザプログラムのフォルダ
user_main.c	ユーザプログラムメイン
app_prog_main.c	アプリケーションプログラムメイン

表 4.2 にプリプロセッサ・マクロ一覧を示します。

表 4.2 プリプロセッサ・マクロ一覧

プリプロセッサ・マクロ名	値	説明
__RX	1 固定	製品ラインナップ＝RX ファミリ

4.4 セクション配置

本ソリューションで提供しているサンプルプログラムのセクション配置を表 4.3 に示します。
注意事項
・※は移動することができません。
・0xFFFE0000 - 0xFFFFFFFF にアプリケーションプログラムを配置することはできません。

表 4.3 アプリケーションプログラムのセクション配置（NonOS 版）

アドレス ※移動不可		セクション名	説明
データ FLASH	0x00100000※	B_PRODUCTS_1	プロダクト情報
	0x00100020※	B_UID	ユニーク ID
	0x00100030※	B_INSTALL_KEY	暗号化されたインストール鍵束
	0x00100230※	B_MSK1	MSK1 用 index 格納領域
	0x00100274※	B_MSK2	MSK2 用 index 格納領域
	0x001002C0※	D_FIRMWARE_UPDATE_CONTROL_BLOCK	管理領域（メイン）
	0x00100340※	D_FIRMWARE_UPDATE_CONTROL_BLOCK_MIRROR	管理領域（ミラー）
	0x001003C0※	B_VERSION_1	プログラムバージョン（起動バンク）
	0x001003C4※	B_PREV_VERSION_1	プログラムバージョン（非起動バンク）
	0x00100400	D_MAC_ADDR_1	MAC アドレス
拡張 RAM	0x00800000	B_FRAME2_1	LCD 1byte 未初期化データ領域
	0x00840000	SI	割り込みスタック領域
		SU	ユーザスタック領域
		B_ETHERNET_BUFFERS_1	ETHER 送信バッファおよび受信バッファ領域
		B_RX_DESC_1	ETHER 受信ディスクリプタ領域
		B_TX_DESC_1	ETHER 送信ディスクリプタ領域
		B_1	1byte 未初期化データ領域
		R_1	1byte 未初期化データ領域（変数）
		B_2	2byte 未初期化データ領域
		R_2	2byte 未初期化データ領域（変数）
		B	4byte 未初期化データ領域
		R	4byte 未初期化データ領域（変数）
		R_STDLIB*	標準ライブラリ 初期値なしデータ
		B_STDLIB*	標準ライブラリ 未初期化データ
		RPFRAM*	FLASH FIT 初期化データ領域
		BSECURE_FW_UPDATE*	ファームウェアアップデートプログラム 未初期化データ領域
		RSECURE_FW_UPDATE*	ファームウェアアップデートプログラム 未初期化データ領域（変数）
コード FLASH	0xFFF00000	C_1	1byte 定数領域
		C_2	2byte 定数領域
		C	4byte 定数領域
		D	4byte 初期化データ領域
		D_1	1byte 初期化データ領域
		D_2	2byte 初期化データ領域
		P	プログラム領域
		W*	switch 文分岐テーブル領域
		L	リテラル領域
		PFRAM*	FLASH FIT プログラム領域
		P_STDLIB*	標準ライブラリ プログラム領域

		C_STDLIB*	標準ライブラリ　定数領域
		L_STDLIB*	標準ライブラリ　リテラル領域
		D_STDLIB*	標準ライブラリ　初期化データ領域
		C$*	C$DEC、C$BSEC、C$VECT の定数領域
	0xFFFDD000	PSECURE_FW_UPDATE*	ファームウェアアップデートプログラム プログラム領域
		CSECURE_FW_UPDATE*	ファームウェアアップデートプログラム 定数領域
		DSECURE_FW_UPDATE*	ファームウェアアップデートプログラム 初期化データ領域
	0xFFFDFBFC※	APP_RESETVECTOR	アプリケーションプログラムのエントリアドレス
	0xFFFDFC00	EXCEPTVECT	割り込みベクタ領域

4.5　使用端子一覧

ファームウェアアップデートプログラムで使用する端子一覧を表 4.4 に示します。

表 4.4　使用端子一覧

モジュール/機能	端子名	入出力	内容
SCI8	TXD8	出力	SCI8 の送信データ出力端子
	RXD8	入力	SCI8 の受信データ入力端子
ETHERC （MII）	ET0_TX_CLK	入力	送信クロック ET0_TX_EN、ET0_ETXD3～ET0_ETXD0、 ET0_TX_ER 信号出力時のタイミング基準信号
	ET0_RX_CLK	入力	受信クロック ET0_RX_DV、ET0_ERXD3～ET0_ERXD0、 ET0_RX_ER 信号入力時のタイミング基準信号
	ET0_TX_EN	出力	送信データ有効 ET0_ETXD3～ET0_ETXD0 上に有効な送信データが出力されていることを示す信号
	ET0_ETXD3～ ET0_ETXD0	出力	4 ビットの送信データ
	ET0_TX_ER	出力	送信エラー 送信中のエラーを PHY-LSI に通知するための信号
	ET0_RX_DV	入力	受信データ有効 ET0_ERXD3～ET0_ERXD0 上に有効な受信データがあることを示す信号
	ET0_ERXD3～ ET0_ERXD0	入力	4 ビットの受信データ
	ET0_RX_ER	入力	受信エラー PHY-LSI から ETHERC へ転送中のフレームにエラーがあることを示す信号
	ET0_CRS	入力	キャリア感知
	ET0_COL	入力	衝突検出
	ET0_MDC	出力	マネジメントデータクロック ET0_MDIO による情報転送用の基準クロック信号
	ET0_MDIO	入出力	マネジメントデータ I/O PHY-LSI との間で管理情報を交換するための双方向データ信号
	ET0_LINKSTA	入力	PHY-LSI からのリンクステータス入力

4.6 割り込み

表 4.5 にファームウェアアップデートプログラムで使用している割り込み一覧を以下に示します。

表 4.5 割り込み一覧

割り込み要求発生元	名称	ベクタ番号	優先度
CMT	CMI0	28	1
	CMI1	29	1
FLASH	FIFERR	21	2
	FRDYI	23	2
SCI8	RXI8	100	14
	TXI8	101	14
	BL1.TEI8	111	3
Ethernet	AL1.EINT0	113	2

4.7 エントリアドレス設定

セキュアブートプログラムから呼び出す関数をセクション（APP_RESETVECTOR）に登録します。
図 4.4 にエントリアドレスの設定箇所を示します。

```
nonOS_rx65n_app_prog/r_bsp/board/generic_rx65n/vecttbl.c

#pragma section C USER_RESETVECTOR
void (* const Reset_Vector[])(void) =
{
  app_prog_main     ★セキュアブートプログラムから呼び出す関数を登録します。
};
```

図 4.4 アプリケーションプログラムのエントリアドレス設定

4.8 例外ベクタテーブル

表 4.6 に例外ベクタテーブルの定義を示します。

表 4.6 例外ベクタテーブル

ベクタアドレス	要因
EXCEPTVECT+0x00	（予約領域）
EXCEPTVECT+0x04	（予約領域）
EXCEPTVECT+0x08	（予約領域）
EXCEPTVECT+0x0C	（予約領域）
EXCEPTVECT+0x10	（予約領域）
EXCEPTVECT+0x14	（予約領域）
EXCEPTVECT+0x18	（予約領域）
EXCEPTVECT+0x1C	（予約領域）
EXCEPTVECT+0x20	（予約領域）
EXCEPTVECT+0x24	（予約領域）
EXCEPTVECT+0x28	（予約領域）
EXCEPTVECT+0x2C	（予約領域）
EXCEPTVECT+0x30	（予約領域）
EXCEPTVECT+0x34	（予約領域）
EXCEPTVECT+0x38	（予約領域）
EXCEPTVECT+0x3C	（予約領域）
EXCEPTVECT+0x40	（予約領域）
EXCEPTVECT+0x44	（予約領域）
EXCEPTVECT+0x48	（予約領域）

EXCEPTVECT+0x4C	（予約領域）
EXCEPTVECT+0x50	特権命令例外
EXCEPTVECT+0x54	アクセス例外
EXCEPTVECT+0x58	未定義命令例外
EXCEPTVECT+0x5C	（予約領域）
EXCEPTVECT+0x60	（予約領域）
EXCEPTVECT+0x64	浮動小数点例外
EXCEPTVECT+0x68	（予約領域）
EXCEPTVECT+0x6C	（予約領域）
EXCEPTVECT+0x70	（予約領域）
EXCEPTVECT+0x74	（予約領域）
EXCEPTVECT+0x78	ノンマスカブル割り込み
0xFFFFFFFC	リセット

詳細については RX65N グループ、RX651 グループ　ユーザーズマニュアル　ハードウェア編（R01UH0590JJ020）を参照してください。

4.9　プログラムサイズ

表 4.7 にファームウェアアップデートプログラムのサイズを示します。ユーザプログラムは含みません。

表 4.7　ファームウェアアップデートプログラムのサイズ

データ種類	サイズ
RAMDATA	71,219 byte
ROMDATA	23,469 byte
PROGRAM	191,646 byte

4.10　最大スタックサイズ

表 4.8 に最大スタックサイズを示します。

表 4.8　最大スタックサイズ

スタック種別	最大スタックサイズ
ユーザスタック：SU	1020byte
割り込みスタック：SI	68byte

5. ファームウェアアップデートプログラムの実装方法について

お客様のプログラムにファームウェアアップデートプログラムを組み込む手順を説明します。ファームウェアアップデートプログラムの処理が OS の有り無しで異なるため RTOS 版と NonOS 版のサンプルプログラムをそれぞれ提供しています。

5.1 ファームウェアアップデートプログラムの実装の流れ

図 5.1 にユーザプログラムにファームウェアアップデートプログラムを実装する流れを示します。

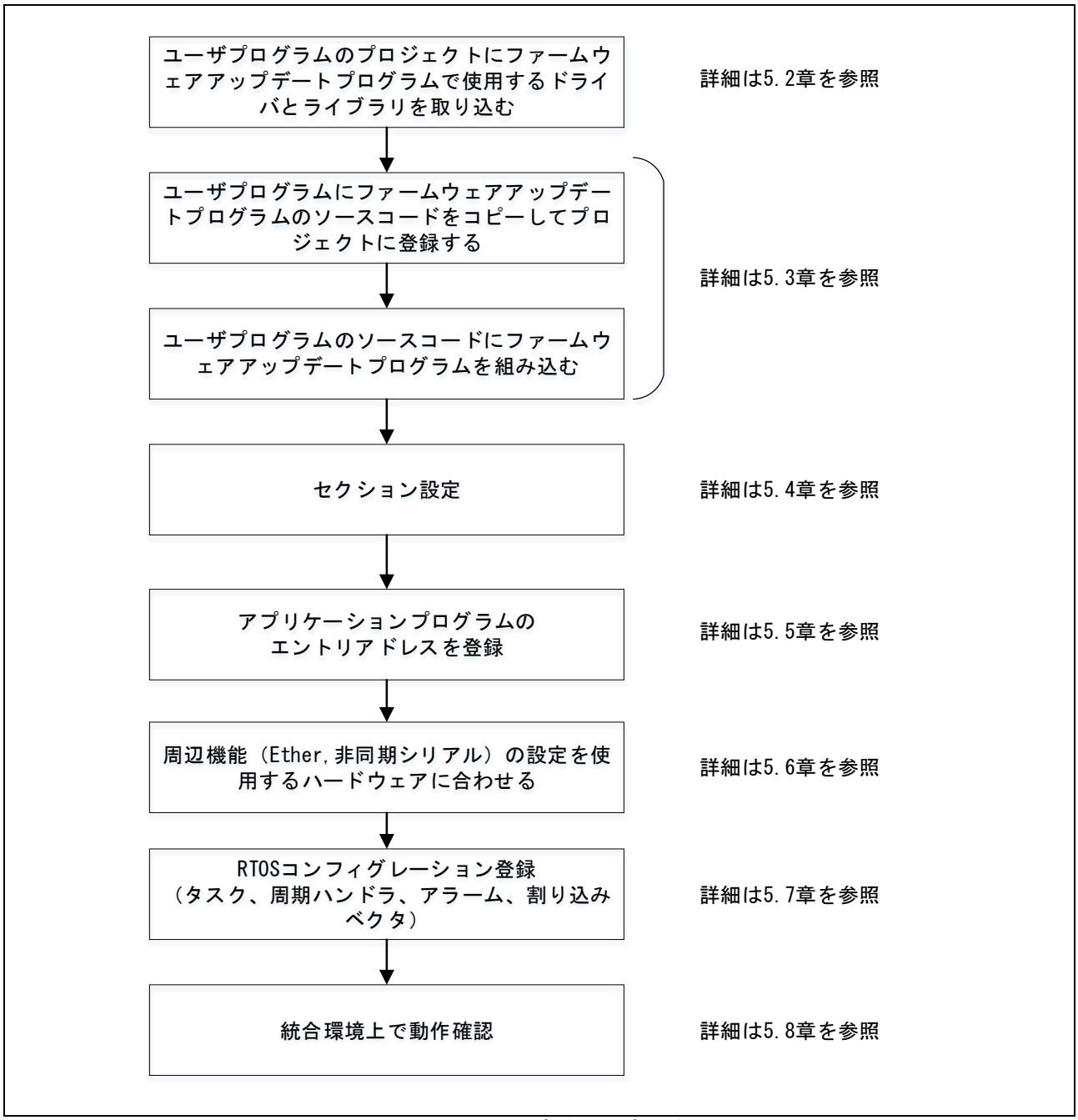

図 5.1 ファームウェアアップデートプログラム実装の流れ

5.2 ドライバとライブラリの取り込み

ユーザプログラムにファームウェアアップデートプログラムで使用するドライバとライブラリをコピーしてプロジェクトに登録します。表 5.1、表 5.2 に追加するドライバとライブラリの一覧を示します。

表 5.1 取り込み対象の FIT・ライブラリー覧（RTOS 版）

ファイル名	説明
rx65n_app_prog	
¥cd_data_aes_library_rx_v104r00	RX ファミリ AES ライブラリ（バイナリ提供）
¥r_bsp	FIT（BSP）
¥r_byteq	FIT（BYTEQ）
¥r_cmt_rx	FIT（CMT）
¥r_config¥	FIT（コンフィグレーション）
¥r_ether_rx	FIT（Ethernet）
¥r_flash_rx	FIT（Flash）
¥r_pincfg	PIN コンフィグレーション
¥r_sci_rx	FIT（SCI）
¥r_t4_driver_rx	FIT（T4 インタフェース変換）
¥r_t4_rx	FIT（T4 プロトコルスタック）
¥r_tsip_rx	FIT（TSIP）

表 5.2 取り込み対象の FIT・ライブラリー覧（NonOS 版）

ファイル名	説明
nonOS_rx65n_app_prog	
¥cd_data_aes_library_rx_v104r00	RX ファミリ AES ライブラリ（バイナリ提供）
¥r_bsp	FIT（BSP）
¥r_byteq	FIT（BYTEQ）
¥r_cmt_rx	FIT（CMT）
¥r_config¥	FIT（コンフィグレーション）
¥r_ether_rx	FIT（Ethernet）
¥r_flash_rx	FIT（Flash）
¥r_pincfg	PIN コンフィグレーション
¥r_sci_rx	FIT（SCI）
¥r_sys_time	FIT（SystemTime）
¥r_t4_driver_rx	FIT（T4 インタフェース変換）
¥r_t4_rx	FIT（T4 プロトコルスタック）
¥r_tsip_rx	FIT（TSIP）

5.3 ファームウェアアップデートプログラムの組み込み

ユーザプログラムのコードにファームウェアアップデートプログラムを追加する手順を示します。

5.3.1 ファームウェアアップデートプログラムのソースコードを組み込む

ユーザプログラムにファームウェアアップデートプログラムのソースコードをコピーしてプロジェクトに登録します。表 5.3、表 5.4 にソースコードの一覧を示します。

表 5.3 ファームウェアアップデートプログラムのソースコード一覧（RTOS 版）

ファイル名	説明
rx65n_app_prog	
¥src	アプリケーションプログラムのフォルダ
¥firm_update	ファームウェアアップデートプログラムのフォルダ
base64_decode.c	Base64 デコード
base64_decode.h	Base64 デコード　ヘッダ
command.c	アップデートコマンド
command.h	アップデートコマンド　ヘッダ
ether.c	Ethernet（UDP、TCP）通信
ether.h	Ethernet（UDP、TCP）通信　ヘッダ
firm_update.c	ファームウェア更新
firm_update.h	ファームウェア更新　ヘッダ
flash_api_bgo_wrapper.c	Flash
flash_api_bgo_wrapper.h	Flash　ヘッダ
uart.c	非同期シリアル
uart.h	非同期シリアル　ヘッダ
¥uitron	RTOS のフォルダ
app_prog_main.c	アプリケーションプログラムメイン
app_prog_main.h	アプリケーションプログラムメイン　ヘッダ
task.c	メインタスク

表 5.4 ファームウェアアップデートプログラムのソースコード一覧（NonOS 版）

ファイル名	説明
nonOS_rx65n_app_prog	
¥src	アプリケーションプログラムのフォルダ
¥firm_update	ファームウェアアップデートプログラムのフォルダ
base64_decode.c	Base64 デコード
base64_decode.h	Base64 デコード　ヘッダ
command.c	アップデートコマンド
command.h	アップデートコマンド　ヘッダ
ether.c	Ethernet（UDP、TCP）通信
ether.h	Ethernet（UDP、TCP）通信　ヘッダ
firm_update.c	ファームウェア更新
firm_update.h	ファームウェア更新　ヘッダ
flash_api_bgo_wrapper.c	Flash
flash_api_bgo_wrapper.h	Flash　ヘッダ
uart.c	非同期シリアル
uart.h	非同期シリアル　ヘッダ
app_prog_main.c	アプリケーションプログラムメイン

5.3.2 RTOS 版のユーザプログラムにアップデート処理を実装する

5.3.2.1 アップデート処理の実装の流れ

図 5.2 に RTOS 版のユーザプログラムにアップデート処理を実装する流れを示します。

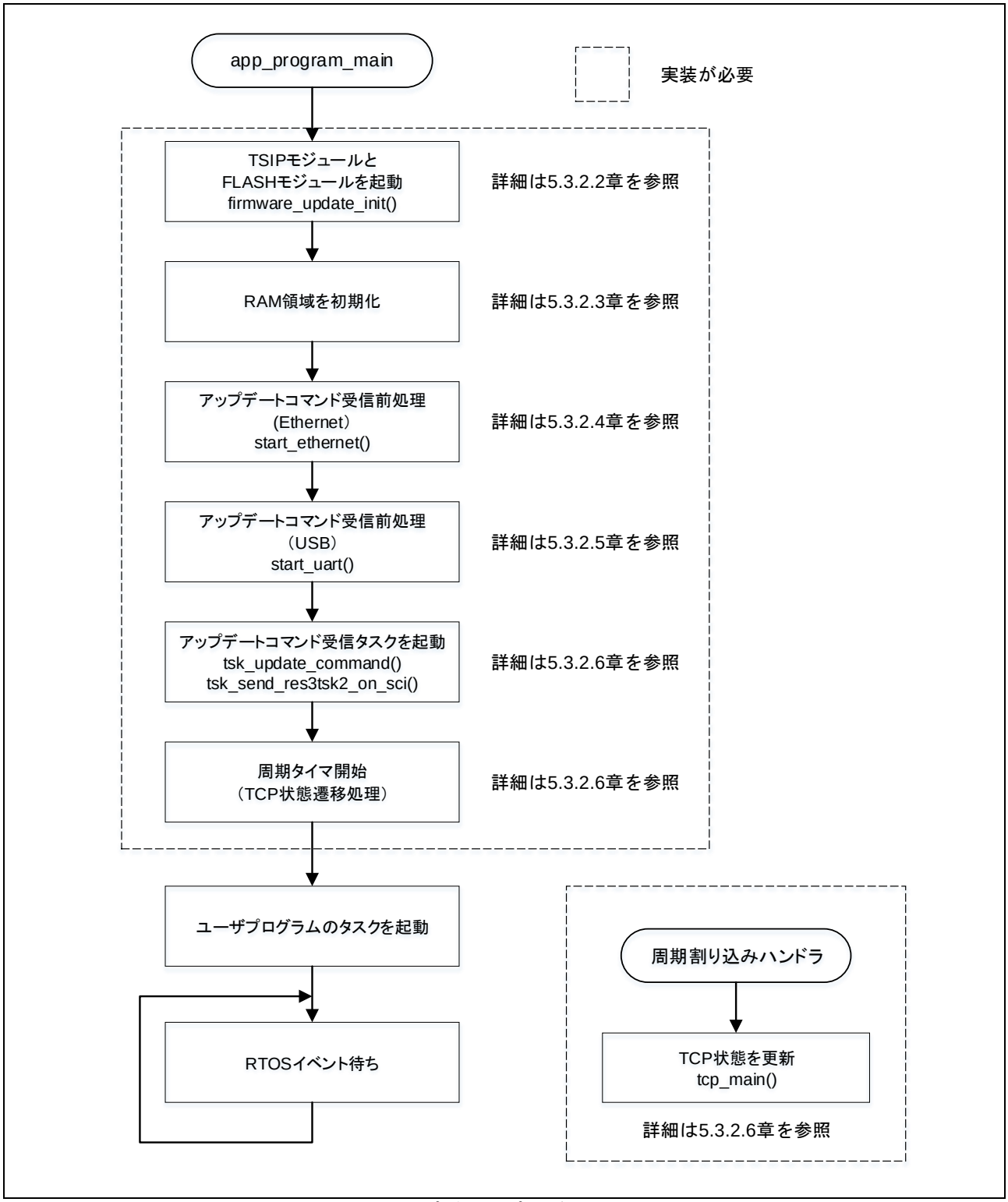

図 5.2 ファームウェアアップデートプログラムの実装の流れ（RTOS 版）

5.3.2.2 TSIP モジュールと FLASH モジュールを起動

firmware_update_init()をコールしてください。

5.3.2.3 RAM 領域を初期化

図 5.3 のコードを追加してください。

```
memset(&g_firmware_update_control_block_image, 0,
sizeof(FIRMWARE_UPDATE_CONTROL_BLOCK));

memcpy(&g_firmware_update_control_block_image,
&g_firmware_update_control_block_data,
sizeof(FIRMWARE_UPDATE_CONTROL_BLOCK));
```

図 5.3 RAM 領域の初期化コード

5.3.2.4 アップデートコマンド処理プログラムを実装（Ethernet）

1）TCP および UDP でアップデートコマンドを処理できるようにします。
start_ethrnet()をコールしてください。

2）通信方式、チャネル番号、端子（ピン配置）を設定します。
設定方法は 5.6.1 章を参照してください。

5.3.2.5 アップデートコマンド処理プログラムを実装（非同期シリアル）

1）非同期シリアルでアップデートコマンドを処理できるようにします。
start_uart()をコールしてください。

2）チャネル番号、端子（ピン配置）を設定します。
設定方法は 5.6.2 章を参照してください。

5.3.2.6 アップデートコマンド受信処理を実装

ユーザプログラムにアップデートコマンド受信処理を実装する方法を示します。

1） ファームウェアアップデートプログラムのタスクを起動します。
表 5.5 に起動対象のタスク ID とタスク名を示します。

表 5.5 ファームウェアアップデートプログラムのタスク一覧

タスク ID	タスク名
ID_TASK_UPDATE_CMD	tsk_update_command()
ID_TASK_SEND_RES3TSK2_ON_SCI	tsk_send_res3tsk2_on_sci()

2）TCP の状態を常に確認し、変化があれば状態更新処理を行います。
周期タイマ契機で tcp_main()をコールしてください。図 5.4 にコーディング例を示します。

```
void start_task(VP_INT exinf)
{
  /* Firmware update task initial processing */
  firmware_update_init();

  /* Update firmware update control area */
  memset(&g_firmware_update_control_block_image, 0,
sizeof(FIRMWARE_UPDATE_CONTROL_BLOCK));
  memcpy(&g_firmware_update_control_block_image,
&g_firmware_update_control_block_data,
sizeof(FIRMWARE_UPDATE_CONTROL_BLOCK));

  /* Start communication device */
  start_ethernet();   /* Ethernet */
  start_uart();    /* UART */

  /*
   * Start task
   */
  act_tsk(ID_TASK_UPDATE_CMD); /* Firmware update task */
  act_tsk(ID_TASK_SEND_RES3TSK2_ON_SCI); /* RES3TSK2 send task */

  sta_cyc(ID_CYC_WAKEUP_TSK_TCP);

    /* User task */
  start_user_task();

  /* Command waiting loop */
  while(1)
  {
    slp_tsk();
  }
}

void cyh_wakeup_tsk_tcp(VP_INT exinf)
{
  tcp_main();
}
```

図 5.4 アップデートコマンド受信処理のコーディング例

5.3.3 NonOS 版のユーザプログラムにアップデート処理を実装する

5.3.3.1 アップデート処理の実装の流れ

図 5.5 に NonOS 版のユーザプログラムにアップデート処理を実装する流れを示します。

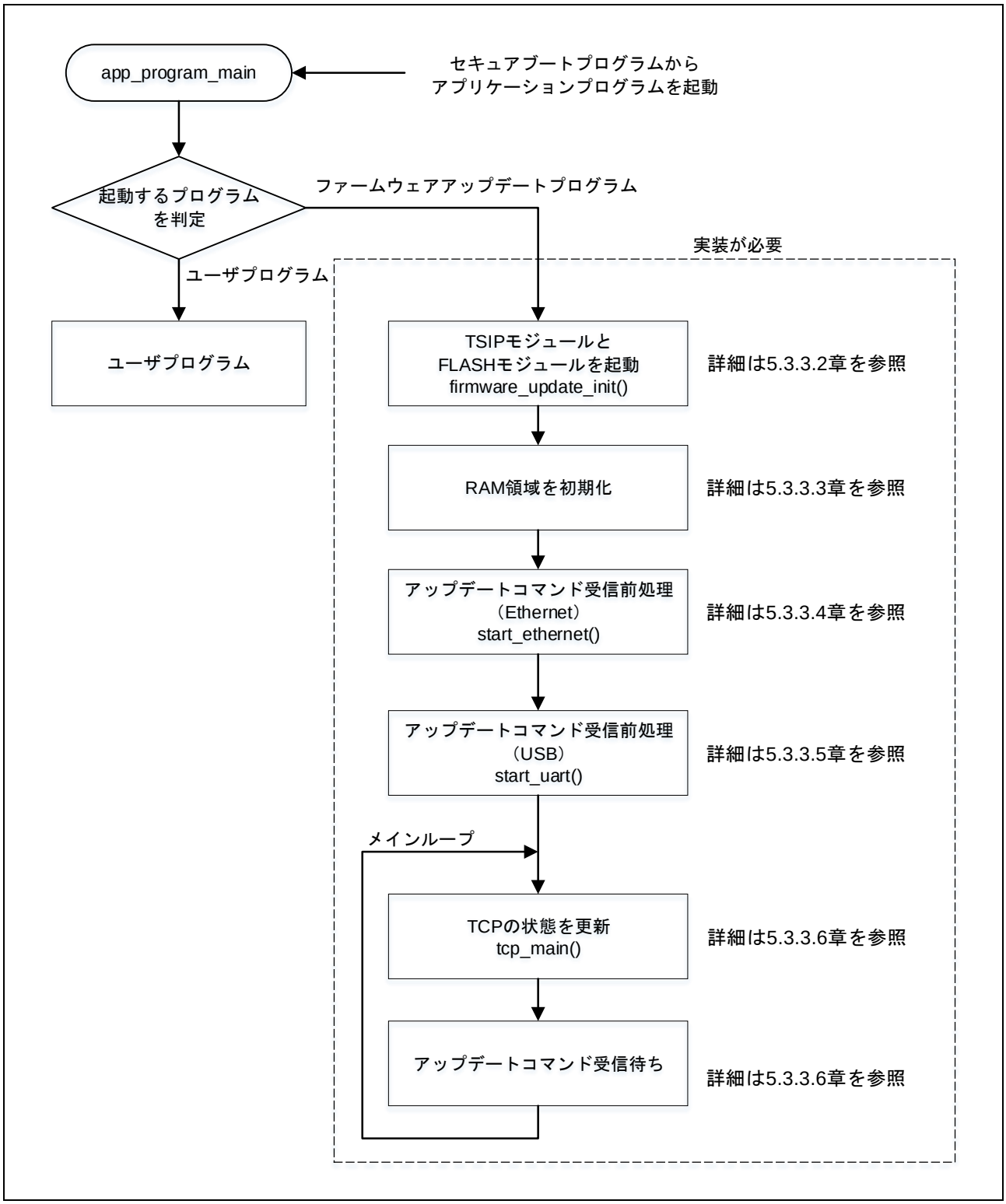

図 5.5 ファームウェアアップデートプログラムの実装の流れ（NonOS 版）

5.3.3.2 TSIP モジュールと FLASH モジュールを起動

RTOS 版と同じ手順です。5.3.2.2 章を参照してください。

5.3.3.3 RAM 領域を初期化

RTOS 版と同じ手順です。5.3.2.3 章を参照してください。

5.3.3.4 アップデートコマンド処理プログラムを実装（Ethernet）

RTOS 版と同じ手順です。5.3.2.4 章を参照してください。

5.3.3.5 アップデートコマンド処理プログラムを実装（非同期シリアル）

RTOS 版と同じ手順です。5.3.2.5 章を参照してください

5.3.3.6 アップデートコマンド受信処理を実装

ユーザプログラムにアップデートコマンドの受信メインループを実装します。以下にメインループで行う処理を示します。

1）受信フラグを常時確認し、アップデートコマンドを受信したときにディスパッチャを実行します。表 5.6 に受信フラグとディスパッチャの対応を示します。

表 5.6 受信フラグとディスパッチャの対応

アップデートコマンド受信経路	受信フラグ名	対応するディスパッチャ
非同期シリアル	g_cmd_recv_sci	command_dispatch_sci()
UDP	g_cmd_recv_udp	command_dispatch_udp()
TCP	g_cmd_recv_tcp	command_dispatch_tcp()

2）TCP の状態を常に確認し、変化があれば状態更新処理を行います。
メインループ内で tcp_main()をコールしてください。図 5.6 にコーディング例を示します。

```
void firmware_update_main(void)
{
  /* Initialize firmware update */
  firmware_update_init();

  /* Initialize control block */
  memset(&g_firmware_update_control_block_image, 0,
sizeof(FIRMWARE_UPDATE_CONTROL_BLOCK));
  memcpy(&g_firmware_update_control_block_image,
&g_firmware_update_control_block_data,
sizeof(FIRMWARE_UPDATE_CONTROL_BLOCK));

  /* Start communication device */
  start_ethernet();   /* Ethernet */
  start_uart();    /* UART */

  /* Wait for update commands */
  while(1)
  {
    tcp_main();

    if (CMD_RECEIVED_SCI == g_cmd_recv_sci)
    {
          command_dispatch_sci();
    }
    if (CMD_RECEIVED_UDP == g_cmd_recv_udp)
    {
       command_dispatch_udp(g_cepid);
    }
    if (CMD_RECEIVED_TCP == g_cmd_recv_tcp)
    {
       command_dispatch_tcp(g_cepid);
    }
  }
}
```

図 5.6 アップデートコマンド受信処理のコーディング例

5.4 セクション設定

アプリケーションプログラムのセクション設定を行います。
詳細については 3.4 章、4.4 章を参照してください。

5.5 アプリケーションプログラムのエントリアドレスを登録

セキュアブートプログラムから呼び出すアプリケーションプログラムのエントリアドレスを設定します。詳細については 3.7 章を参照してください。

5.6 周辺機能設定

周辺機能（Ethernet、非同期シリアル）のコンフィグレーション、端子設定を使用するハードウェアに合わせます。図 5.7 の RX65N RSK のハードウェア構成をもとに説明します。

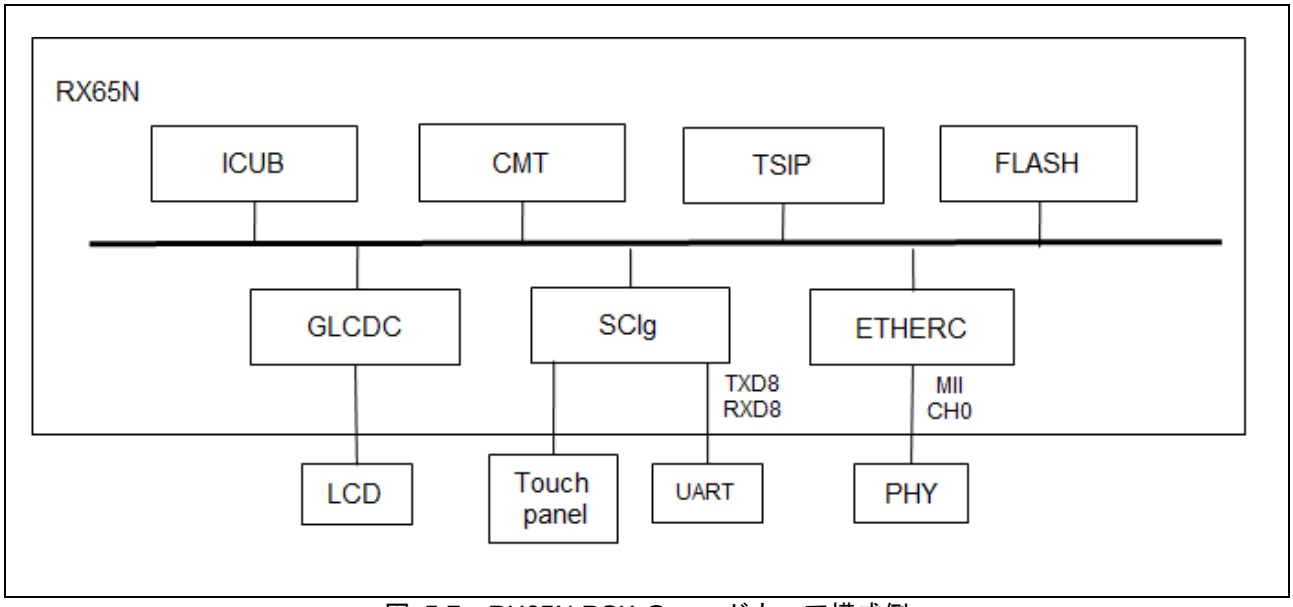

図 5.7 RX65N RSK のハードウェア構成例

5.6.1 Ethernet

インタフェース種別、チャネル番号、PHY-LSI アドレス、端子（ピン配置）を設定します。
表 5.7 に RX65N RSK の設定例を示します。

表 5.7 ハードウェア構成（Ethernet）

項目名	設定値
インタフェース種別	MII
チャネル番号	CH0（CH1 未実装）
PHY-LSI アドレス	30
端子（ピン配置）	ET0_TX_CLK（PC4）
	ET0_RX_CLK（P76）
	ET0_TX_EN（P80）
	ET0_ETXD3（PC6）
	ET0_ETXD2（PC5）
	ET0_ETXD1（P82）
	ET0_ETXD0（P81）
	ET0_TX_ER（PC3）

	ET0_RX_DV（PC2）
	ET0_ERXD3（PC0）
	ET0_ERXD2（PC1）
	ET0_ERXD1（P74）
	ET0_ERXD0（P75）
	ET0_RX_ER（P77）
	ET0_CRS（P83）
	ET0_COL（PC7）
	ET0_MDC（P72）
	ET0_MDIO（P71）
	ET0_LINKSTA（P54）

図 5.8 にインタフェース種別とチャネル番号の設定例を、図 5.9 に端子設定の設定例を示します。

```
/r_config/r_ether_rx_config.h
---
/* Ethernet interface select.
 0 = MII  (Media Independent Interface)
 1 = RMII (Reduced Media Independent Interface)
 */
   #define ETHER_CFG_MODE_SEL                          (0)★インタフェース＝MII


   #define ETHER_CFG_CH0_PHY_ADDRESS                    (30)     /* Please define
the PHY-LSI address in the range of 0-31. */  ★CH0 のアドレス=30
   #define ETHER_CFG_CH1_PHY_ADDRESS                    (1)     /* Please define
the PHY-LSI address in the range of 0-31. */  ★CH1 のアドレス=0（未実装）


/* EINT interrupt priority level. This definition is not used when EINT
interrupt is assigned to Group interrupt. */
   #define ETHER_CFG_EINT_INT_PRIORITY                 (2)     /* Please define
the interruption level within the range of 1-15. */

/* Group AL1 interrupt priority level. This definition is not used when EINT
interrupt is assigned to Peripheral
 interrupt. */
   #define ETHER_CFG_AL1_INT_PRIORTY                   (2)     /* Please define
the interruption level within the range of 1-15. */
```

図 5.8　インタフェース種別とチャネル設定（Ethernet）

```
r_pincfg/r_pinset.c
---
void R_ETHER_PinSet_ETHERC0_MII()
{
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC);

    /* Set ET0_TX_CLK pin */
    MPC.PC4PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B4 = 1U;

    /* Set ET0_RX_CLK pin */
    MPC.P76PFS.BYTE = 0x11U;
    PORT7.PMR.BIT.B6 = 1U;

    /* Set ET0_TX_EN pin */
    MPC.P80PFS.BYTE = 0x11U;
    PORT8.PMR.BIT.B0 = 1U;

    /* Set ET0_ETXD3 pin */
    MPC.PC6PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B6 = 1U;

    /* Set ET0_ETXD2 pin */
    MPC.PC5PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B5 = 1U;

    /* Set ET0_ETXD1 pin */
    MPC.P82PFS.BYTE = 0x11U;
    PORT8.PMR.BIT.B2 = 1U;

    /* Set ET0_ETXD0 pin */
    MPC.P81PFS.BYTE = 0x11U;
    PORT8.PMR.BIT.B1 = 1U;

    /* Set ET0_TX_ER pin */
    MPC.PC3PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B3 = 1U;

    /* Set ET0_RX_DV pin */
    MPC.PC2PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B2 = 1U;

    /* Set ET0_ERXD3 pin */
    MPC.PC0PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B0 = 1U;

    /* Set ET0_ERXD2 pin */
    MPC.PC1PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B1 = 1U;

    /* Set ET0_ERXD1 pin */
    MPC.P74PFS.BYTE = 0x11U;
    PORT7.PMR.BIT.B4 = 1U;

    /* Set ET0_ERXD0 pin */
    MPC.P75PFS.BYTE = 0x11U;
    PORT7.PMR.BIT.B5 = 1U;

    /* Set ET0_RX_ER pin */
```

```
    MPC.P77PFS.BYTE = 0x11U;
    PORT7.PMR.BIT.B7 = 1U;

    /* Set ET0_CRS pin */
    MPC.P83PFS.BYTE = 0x11U;
    PORT8.PMR.BIT.B3 = 1U;

    /* Set ET0_COL pin */
    MPC.PC7PFS.BYTE = 0x11U;
    PORTC.PMR.BIT.B7 = 1U;

    /* Set ET0_MDC pin */
    MPC.P72PFS.BYTE = 0x11U;
    PORT7.PMR.BIT.B2 = 1U;

    /* Set ET0_MDIO pin */
    MPC.P71PFS.BYTE = 0x11U;
    PORT7.PMR.BIT.B1 = 1U;

    /* Set ET0_LINKSTA pin */
    MPC.P54PFS.BYTE = 0x11U;
    PORT5.PMR.BIT.B4 = 1U;

    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_MPC);
}
```

図 5.9　端子設定（Ethernet）

5.6.2 非同期シリアル

通信方式、チャネル番号、端子（ピン配置）を設定します。
表 5.8 に RX65N RSK の構成例を示します

表 5.8 ハードウェア構成（非同期シリアル）

項目名	設定値
通信方式	調歩同期式
チャネル番号	SCI8
端子（ピン配置）	RXI8（PJ1）
	TXI8（PJ2）

図 5.10～図 5.12 に設定例を示します。

```
r_config/r_sci_rx_config.h
---

#define SCI_CFG_ASYNC_INCLUDED  (1)★通信方式　調歩同期式
#define SCI_CFG_SYNC_INCLUDED   (0)
#define SCI_CFG_SSPI_INCLUDED   (0)
・・・
#define SCI_CFG_CH0_INCLUDED    (0)
#define SCI_CFG_CH1_INCLUDED    (0)
#define SCI_CFG_CH2_INCLUDED    (1)
#define SCI_CFG_CH3_INCLUDED    (0)
#define SCI_CFG_CH4_INCLUDED    (0)
#define SCI_CFG_CH5_INCLUDED    (1)
#define SCI_CFG_CH6_INCLUDED    (0)
#define SCI_CFG_CH7_INCLUDED    (0)
#define SCI_CFG_CH8_INCLUDED    (1) ★チャネル番号　SCI8
#define SCI_CFG_CH9_INCLUDED    (0)
#define SCI_CFG_CH10_INCLUDED   (0)
#define SCI_CFG_CH11_INCLUDED   (0)
#define SCI_CFG_CH12_INCLUDED   (0)
```

図 5.10 通信方式とチャネル番号の設定例（非同期シリアル）

```
r_pincfg/r_pinset.c
--
void R_SCI_PinSet_SCI8()
{
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC);

    /* Set RXD8 pin */
    MPC.PJ1PFS.BYTE = 0x0AU;
    PORTJ.PDR.BIT.B1 = 0;      // Set PJ1(Pin59) direction to output (for use
as RXD)
    PORTJ.PMR.BIT.B1 = 1U;
    PORTJ.PCR.BIT.B1 = 1U; /* Pull up */

    /* Set TXD8 pin */
    MPC.PJ2PFS.BYTE = 0x0AU;
    PORTJ.PDR.BIT.B2 = 1;      // Set PJ2(Pin58) direction to input (for use
as TXD)
    PORTJ.PMR.BIT.B2 = 1U;

    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_MPC);
```

```
}
```

図 5.11　端子設定（非同期シリアル）

```
src/firm_update/uart.c
---
void start_uart(void)
{
  sci_cfg_t   sci;

  /* Set up the configuration data structure for asynchronous (UART)
operation */
  sci.async.baud_rate    = 921600u;
  sci.async.clk_src     = SCI_CLK_INT;
  sci.async.data_size    = SCI_DATA_8BIT;
  sci.async.parity_en    = SCI_PARITY_OFF;
  sci.async.parity_type  = SCI_EVEN_PARITY;
  sci.async.stop_bits    = SCI_STOPBITS_1;
  sci.async.int_priority = 14u;    /* 1=Lowest, 15=Highest */

  /* Pin setting TXD8,RXD8 */
    R_SCI_PinSet_SCI8();

    /* SCI8 open */
    R_SCI_Open(SCI_CH8, SCI_MODE_ASYNC, &sci, sci_callback, &sci_handle);

    /* TEI interrupt enable */
    R_SCI_Control(sci_handle, SCI_CMD_EN_TEI, (void *)NULL);
}
```

図 5.12　起動処理（非同期シリアル）

5.7 RTOS コンフィグレーション設定

ファームウェアアップデートプログラムで使用するタスク・周期ハンドラ・アラームハンドラ・イベントフラグ・割り込みを RTOS コンフィグレーションファイル（以降、RTOS コンフィグ）に登録します。詳細については 5.7.1 章～5.7.5 章を参照してください。

5.7.1 タスクを登録

ファームウェアアップデートプログラムで使用するタスクを RTOS に登録します。
表 5.9 に登録が必要なタスクを、図 5.13 に登録例を示します。

表 5.9 登録が必要なタスク

タスク名	優先度	スタックサイズ	スタックのセクション
tsk_update_command()	3	2048	SURI_STACK
tsk_send_res3tsk2_on_sci()	3	2048	SURI_STACK

```
rx65n_app_prog/generate/rx65n_app_prog.cfg
--
task[]{ // Command handler
    name          = ID_TASK_UPDATE_CMD;
    entry_address   = tsk_update_command();
    initial_start   = OFF;
    stack_size     = 2048;
    priority       = 3;
    stack_section   = SURI_STACK;
    exinf         = 2;
};

task[]{ // Send "Res3,tsk2" on sci
    name          = ID_TASK_SEND_RES3TSK2_ON_SCI;
    entry_address   = tsk_send_res3tsk2_on_sci();
    initial_start   = OFF;
    stack_size     = 2048;
    priority       = 3;
    stack_section   = SURI_STACK;
    exinf         = 3;
};
```

図 5.13 RTOS コンフィグ登録例（タスク）

5.7.2 周期ハンドラを登録

ファームウェアアップデートプログラムで使用する周期ハンドラを RTOS に登録します。
周期ハンドラとは一定の時間ごとに周期的に起動される周期処理専用ルーチンです。
表 5.10 に登録が必要な周期ハンドラを、図 5.14 に登録例を示します。

表 5.10 登録が必要な周期ハンドラ

ハンドラ名	周期（ms）	概要
cyh_lan_status()	10	LAN ステータス取得
cyh_wakeup_tsk_tcp()	100	TCP 状態更新処理

```
rx65n_app_prog/generate/rx65n_app_prog.cfg
--

// Cyclic Handler Definition (TCP 100ms)
cyclic_hand[] {
    name          = ID_CYC_WAKEUP_TSK_TCP;
```

```
    entry_address   = cyh_wakeup_tsk_tcp();
    interval_counter = 100;
    start           = OFF;
    phsatr          = OFF;
    phs_counter     = 100;
    exinf           = 5;
};

// Cyclic Handler Definition
cyclic_hand[] {
    name            = ID_CYC_LAN_STATUS;
    entry_address   = cyh_lan_status();
    interval_counter = 10;
    start           = OFF;
    phsatr          = OFF;
    phs_counter     = 0;
    exinf           = 10;
};
```

図 5.14　RTOS コンフィグ登録例（周期ハンドラ）

5.7.3　アラームハンドラを登録

ファームウェアアップデートプログラムで使用するアラームハンドラを RTOS コンフィグに登録します。アラームハンドラとは指定した時間が経過したときに起動されるルーチンです。
表 5.11 に登録が必要なアラームハンドラを、図 5.15 に登録例を示します。

表 5.11　登録が必要なアラームハンドラ

ハンドラ名	概要
alh_command_rcv_timeout()	非同期シリアルでアップデートコマンドを受信したときのガードタイマ

```
rx65n_app_prog/generate/rx65n_app_prog.cfg
--

// Alarm Handler (dummy) Definition
alarm_hand[] {
    name            = ID_ALM1;
    entry_address   = alh_command_rcv_timeout();
    exinf           = 1;
};
```

図 5.15　RTOS コンフィグ登録例（アラームハンドラ）

5.7.4　イベントフラグを登録

ファームウェアアップデートプログラムで使用するイベントフラグを RTOS コンフィグに登録します。表 5.12 に登録が必要なイベントフラグを、図 5.16 に登録例を示します。

表 5.12　登録が必要なイベントフラグ

フラグ名	概要
ID_FLAG1	FLASH 割り込みの待ち合わせ
ID_FLAG2	アップデートコマンド受信の待ち合わせ

```
rx65n_app_prog/generate/rx65n_app_prog.cfg
--
flag[]{
```

```
  name           = ID_FLAG1;
  initial_pattern = 0x00000000;
  wait_multi      = TA_WMUL;
  clear_attribute = YES;
  wait_queue      = TA_TFIFO;
};

flag[]{
  name           = ID_FLAG2;
  initial_pattern = 0x00000000;
  wait_multi      = TA_WMUL;
  clear_attribute = YES;
  wait_queue      = TA_TFIFO;
};
```

図 5.16　RTOS コンフィグ登録例（イベントフラグ）

5.7.5　割り込みベクタを登録

ファームウェアアップデートプログラムで使用する割り込みベクタを RTOS に登録します。図 5.5 に登録の流れを示します。

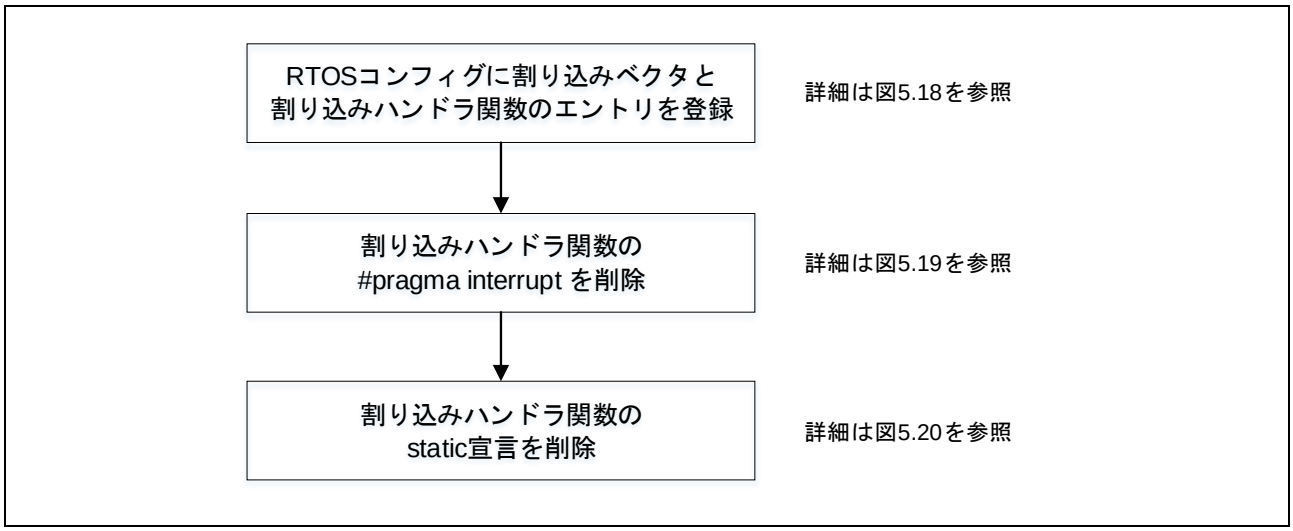

図 5.17　割り込みベクタ登録の流れ

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割り込みの一覧を示します。

表 5.13　割り込みベクター覧

割り込み名	ベクタ番号	割り込みハンドラ関数名
FLASH(FIFERR)	21	Excep_FCU_FIFERR
FLASH(FRDYI)	23	Excep_FCU_FRDYI
SCI8(RXI8)	100	sci8_rxi8_isr
SCI8(TXI8)	101	sci8_txi8_isr
GROUPBL1（TEI8）	111	group_bl1_handler_isr
GROUPAL1（EINT0）	113	group_al1_handler_isr

例として図 5.18～図 5.20 に SCI8（RXI8, TXI8）の登録手順を示します。
1） RTOS コンフィグに割り込みベクタと割り込みハンドラ関数を追加します。
図 5.18 に登録例を示します。

```
  rx65n_app_prog/generate/rx65n_app_prog.cfg
```

```
--
// Interrupt Handler (SCI8 RXI8) Definition
interrupt_vector[100]{  ★割り込みベクタ 100
    os_int         = YES;
    entry_address   = sci8_rxi8_isr();  ★RXI8 の割り込みハンドラ関数
  pragma_switch = ;
};

// Interrupt Handler (SCI8 TXI8) Definition
interrupt_vector[101]{  ★割り込みベクタ 101
    os_int         = YES;
    entry_address   = sci8_txi8_isr();  ★TXI8 の割り込みハンドラ関数
  pragma_switch = ;
};
```

図 5.18　割り込みベクタ登録（1）

2）1）で登録した割り込みハンドラ関数の#pragma interrupt 定義 を削除します。
図 5.19 にコーディング例を示します。

```
r_sci_rx/src/targets/rx65n/r_sci_rx65n.c
---
#pragma interrupt sci8_txi8_isr(vect=VECT(SCI8,TXI8))  ★削除
static void sci8_txi8_isr(void);
#pragma interrupt sci8_rxi8_isr(vect=VECT(SCI8,RXI8))  ★削除
static void sci8_rxi8_isr(void);
↓
static void sci8_txi8_isr(void);
static void sci8_rxi8_isr(void);
```

図 5.19　割り込みベクタ登録（2）

3）1）で登録した割り込みハンドラ関数を外部から参照できるようにします。
図 5.20 にコーディング例を示します。

```
r_sci_rx/src/targets/rx65n/r_sci_rx65n.c
---
static void sci8_txi8_isr(void);
static void sci8_rxi8_isr(void);

static void sci8_txi8_isr(void)
{
・・・
}

static void sci8_rxi8_isr(void)
{
・・・
}

↓  ★static 宣言を削除します。

void sci8_txi8_isr(void);
void sci8_rxi8_isr(void);

void sci8_txi8_isr(void)
{
```

```
・・・
}

void sci8_rxi8_isr(void)
{
・・・
}
```

図 5.20　割り込みベクタ登録（3）

5.8 動作確認

アプリケーションプログラムを統合環境（以降、e2studio）で動作させる手順について説明します。

5.8.1 エミュレータを接続

E2 エミュレータ Lite を用いて説明します。詳細については使用するエミュレータのマニュアルを参照してください。

1） E2 エミュレータ Lite を RX65N RSK の E1/E2 Lite connector に接続します。
図 5.21 に RX65N RSK のレイアウトを示します。

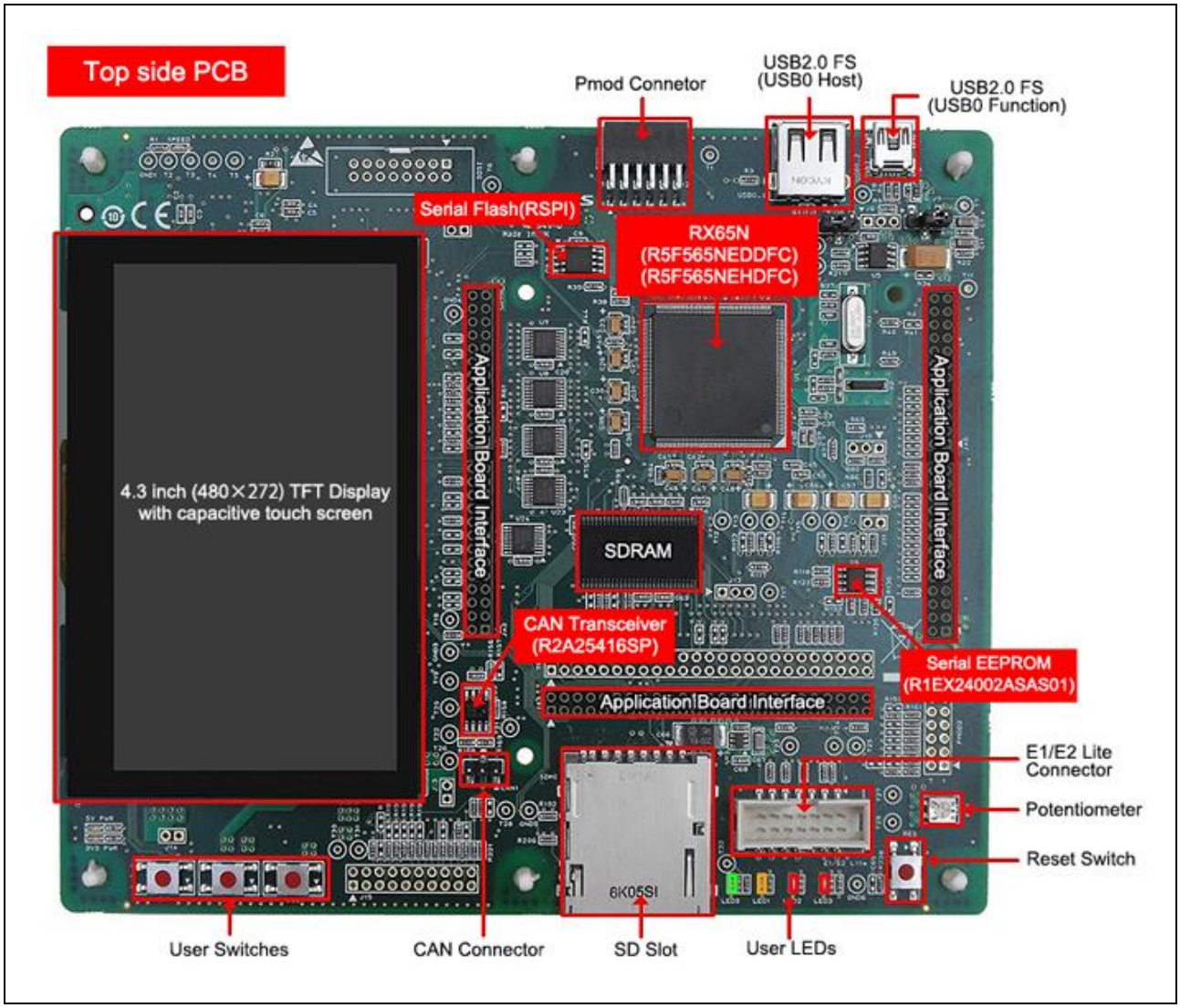

図 5.21 RX65N RSK のレイアウト

2）付属の AC アダプタを接続し、ボードに電源を供給します。
図 5.22 に RX65N RSK と E2 エミュレータ Lite の接続図を示します。

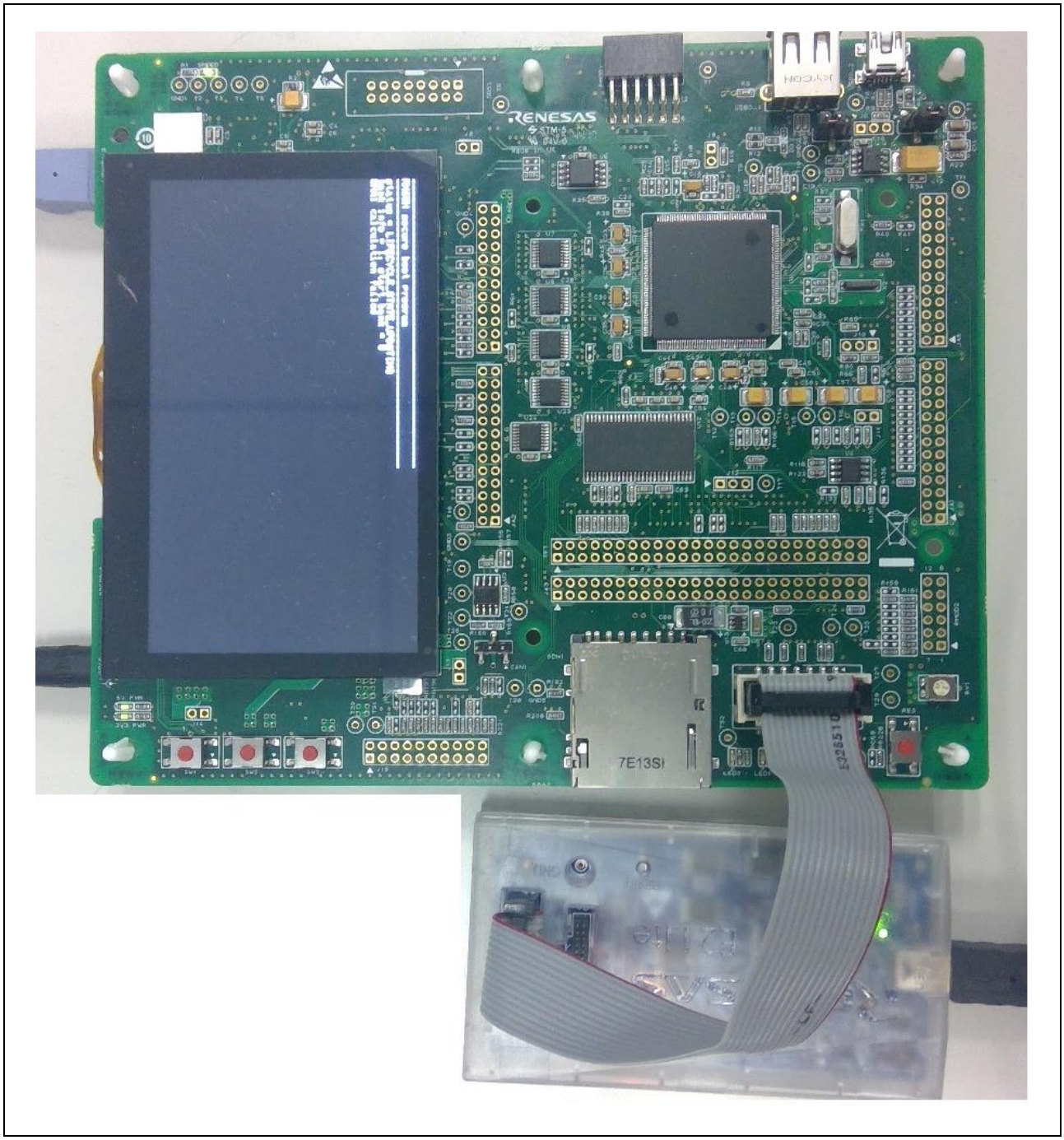

図 5.22　RX65N RSK と E2 エミュレータ Lite を接続

5.8.2 e2studio のデバッグ設定

本ソリューションはデュアルバンク機能を使用するためデバッグ設定の変更が必要です。
図 5.23、図 5.24 に変更点を示します。

Debugger＞接続設定＞CPU 動作モード
起動バンクを変更する：いいえ→はい

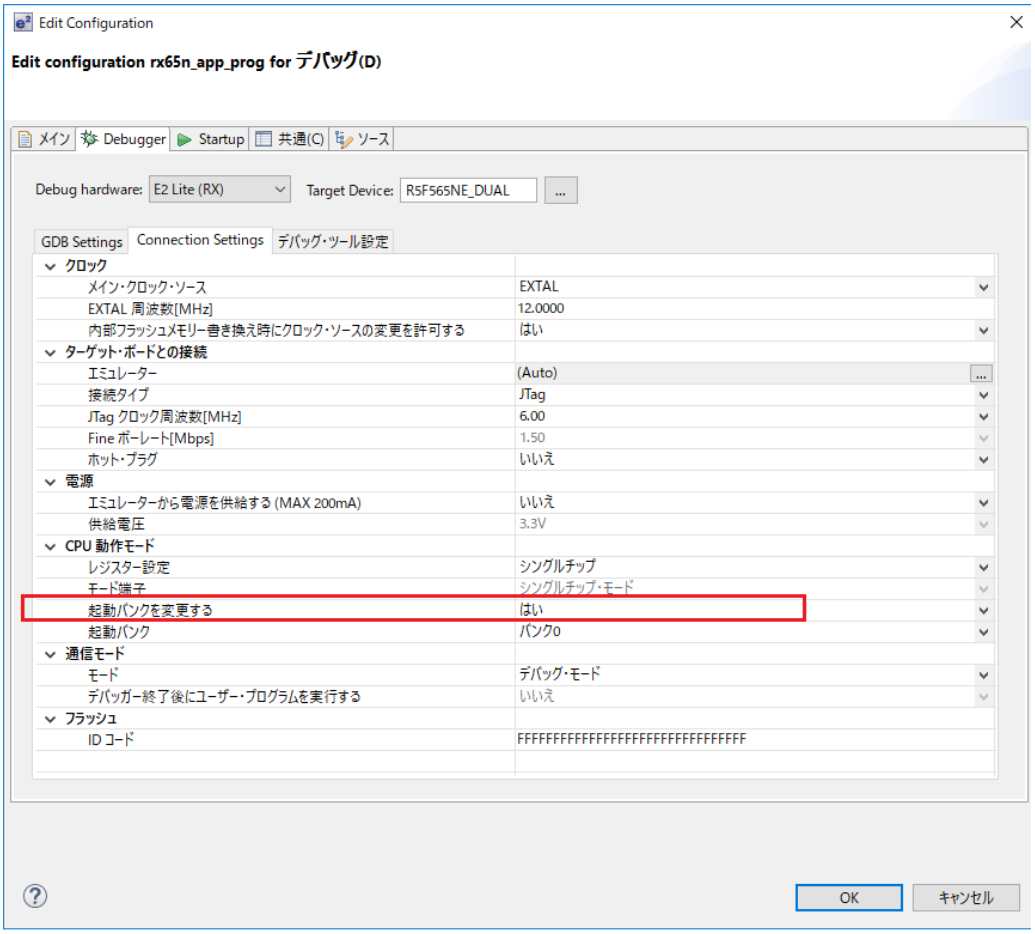

図 5.23 デバッグ設定（1）

Debugger＞デバッグ・ツール設定＞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：いいえ→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→はい

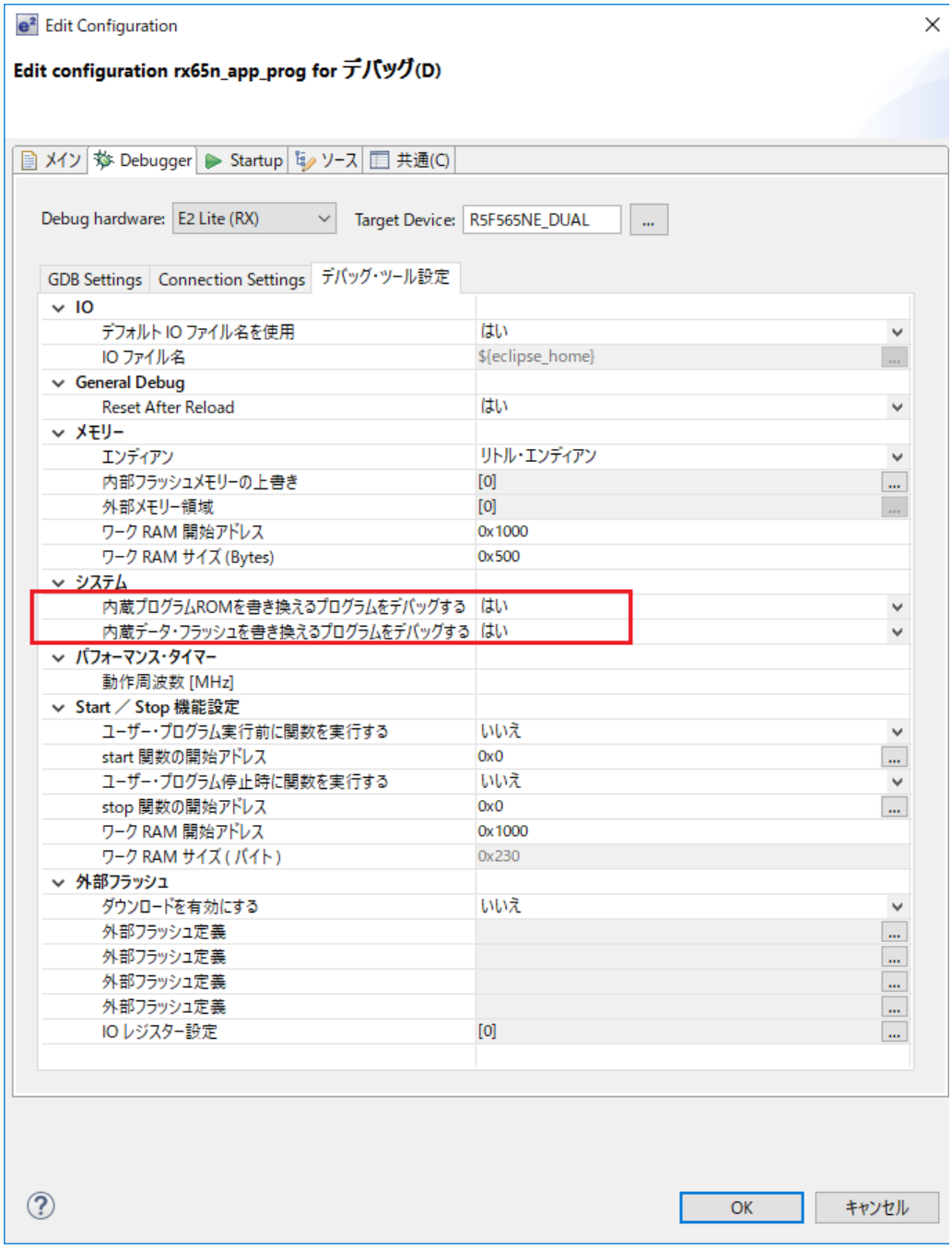

図 5.24　デバッグ設定（2）

5.8.3 アプリケーションプログラムを実行

アプリケーションプログラムを e2studio で動作させる手順を以下に示します。
1）アプリケーションプログラムを 0xFFE00000 番地に読み込む。
2）デバッグ用のセキュアブートプログラムを読み込む。
3）アプリケーションプログラム起動に必要なデータ（以降、オプションデータ）を作成する。
4）オプションデータを読み込む
5）アプリケーションプログラムを実行する。

次頁以降に詳細な手順を示します。
1）アプリケーションプログラムを 0xFFE00000 番地に読み込む
図 5.25 に読み込み手順を示します。

オフセット：FFF00000（FFF00000 は 2 の補数表現で-1MB を表します）

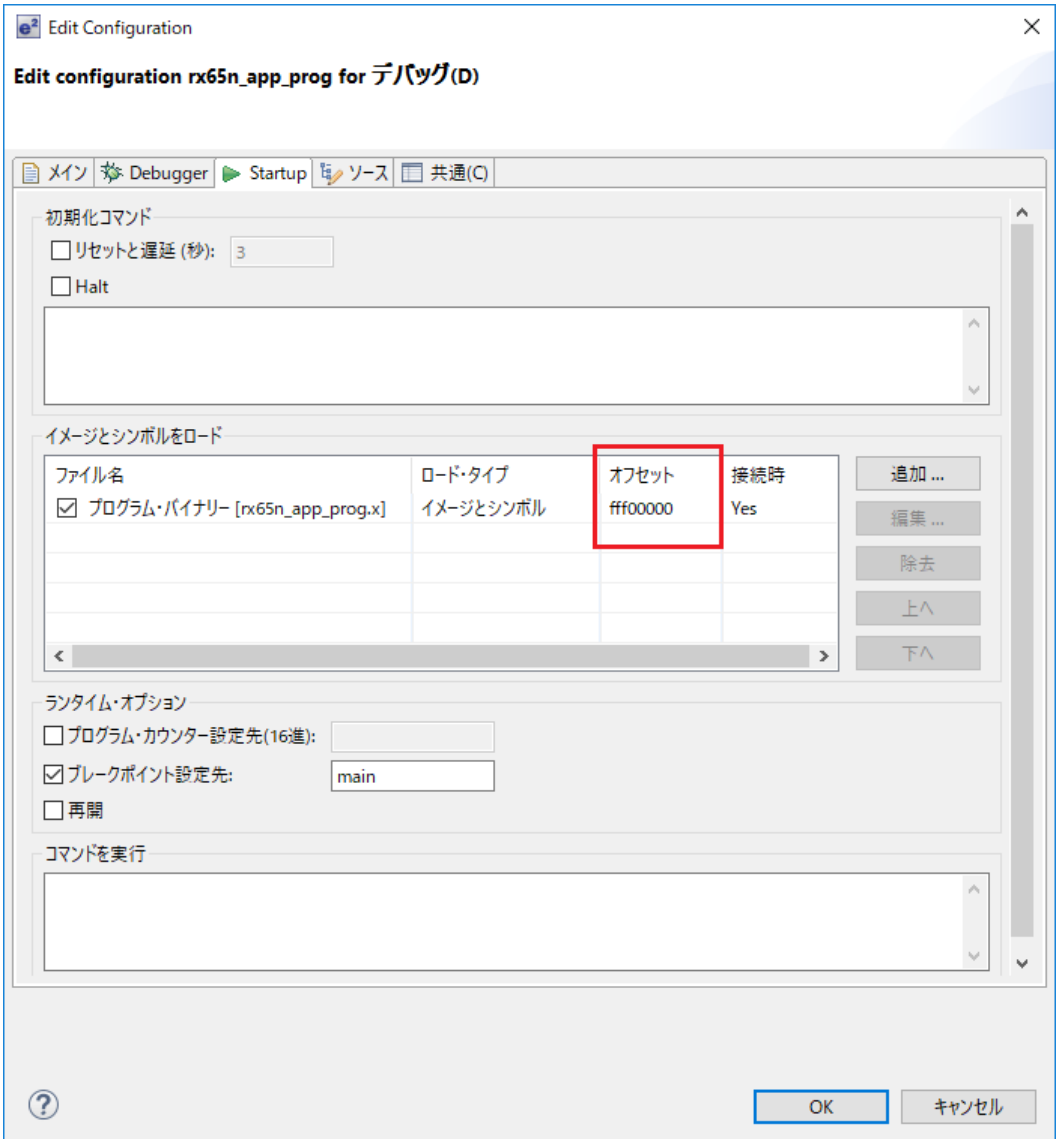

図 5.25 アプリケーションプログラムの読み込み

2）デバッグ用のセキュアブートプログラムを読み込む。
図 5.26 に読み込み手順を示します。

ファイル名：セキュアブートプログラム（デバッグ用）を指定
ロード・タイプ：イメージのみ
オフセット：0
ロード順序：リスト最上位

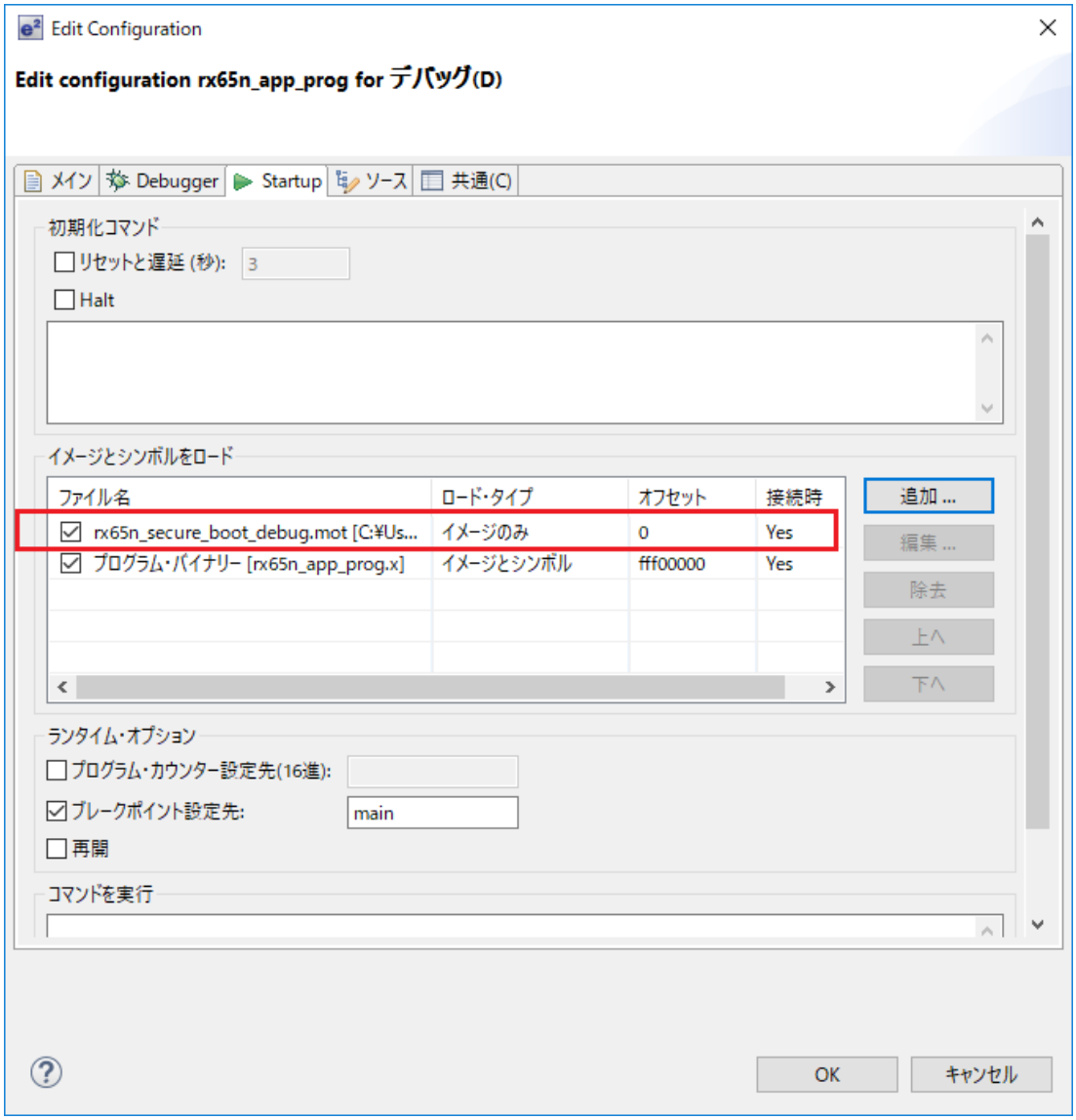

図 5.26　セキュアブートプログラムの読み込み

3）オプションデータを作成する。
量産用ファームウェアから以下を抽出しモトローラ S レコードフォーマット（*.mot）で保存します。
・データフラッシュ領域（0x00100000-0x00107FFF）
・オプション設定メモリ（0xFE7F5D00-0xFE7F5D7F）
・暗号関連のパラメータ（0xFFEFC000-0xFFEFC0DF）

図 5.27 にオプションデータの作成例を示します。

```
S022000025836354e5f3030303030303030315f524f4d5f52454c454153452e6d6f743F
--データフラッシュ領域--
S2141000025836354E0A0000000000000000006E
S2141000100000000000000000000000000000CB
S214100020FFFFFFFFFFFFFFFFFFFFFFFFFFFFCB
・・・
S214107FC0FFFFFFFFFFFFFFFFFFFFFFFFFFFFAC
S214107FD0FFFFFFFFFFFFFFFFFFFFFFFFFFFF9C
S214107FE0FFFFFFFFFFFFFFFFFFFFFFFFFFFF8C
S214107FF0FFFFFFFFFFFFFFFFFFFFFFFFFFFF7C

--オプション設定メモリ--
S315FE7F5D008FFFFFFFFFFFFFFFFFFFFFFFFF90
S315FE7F5D10FFFFFFFFFFFFFFFFFFFFFFFFFF10
S315FE7F5D30FFFFFFFFFFFFFFFFFFFFFFFFFF0
S315FE7F5D40FFFFFFFFFFFFFFFFFFFFFFFFFFE0
S315FE7F5D50FFFFFFFFFFFFFFFFFFFFFFFFFFD0
S315FE7F5D60FFFFFFFFFFFFFFFFFFFFFFFFFFC0
S315FE7F5D70FFFFFFFFFFFFFFFFFFFFFFFFFFB0

--暗号関連のパラメータ--
S315FFEFC000EB8322BD466B257DFA0A737CB3208DCA7F
S315FFEFC0102BB04E742533722A3A8E1A0BB8C73F539D
S315FFEFC020C32D932AD35179019D67C613ADFBA9E5BE
S315FFEFC03015E37CD6BF221191945C34A8D3B13DEAC8
S315FFEFC040B66515DC2488B5F92F83227CF359D367C0
S315FFEFC0509635F3CD9AD7B63FA9C454BFC3E5B4F827
S315FFEFC060B66515DC2488B5F92F83227CF359D367A0
S315FFEFC0709635F3CD9AD7B63FA9C454BFC3E5B4F807
S315FFEFC08055AA55AA55AA55AA55AA55AA55AA55AAC4
S315FFEFC09055AA55AA55AA55AA55AA55AA55AA55AAB4
S315FFEFC0A0B857F01CEFD8B86AFD8C8B802965F41E64
S315FFEFC0B0DF66C8E9EE63FFD73D5C512607ABD59246
S315FFEFC0C02A064E9AB13C13D8F8FDFC20C1B609B348
S315FFEFC0D0A8963373A7DB578BEFCC9B546C4081C885
```

図 5.27　オプションデータ作成例

4）オプションデータを読み込む
図 5.28 に読み込み手順を示します。

ファイル名：オプションデータを指定
ロード・タイプ：イメージのみ
オフセット：0
ロード順序：リスト最下位

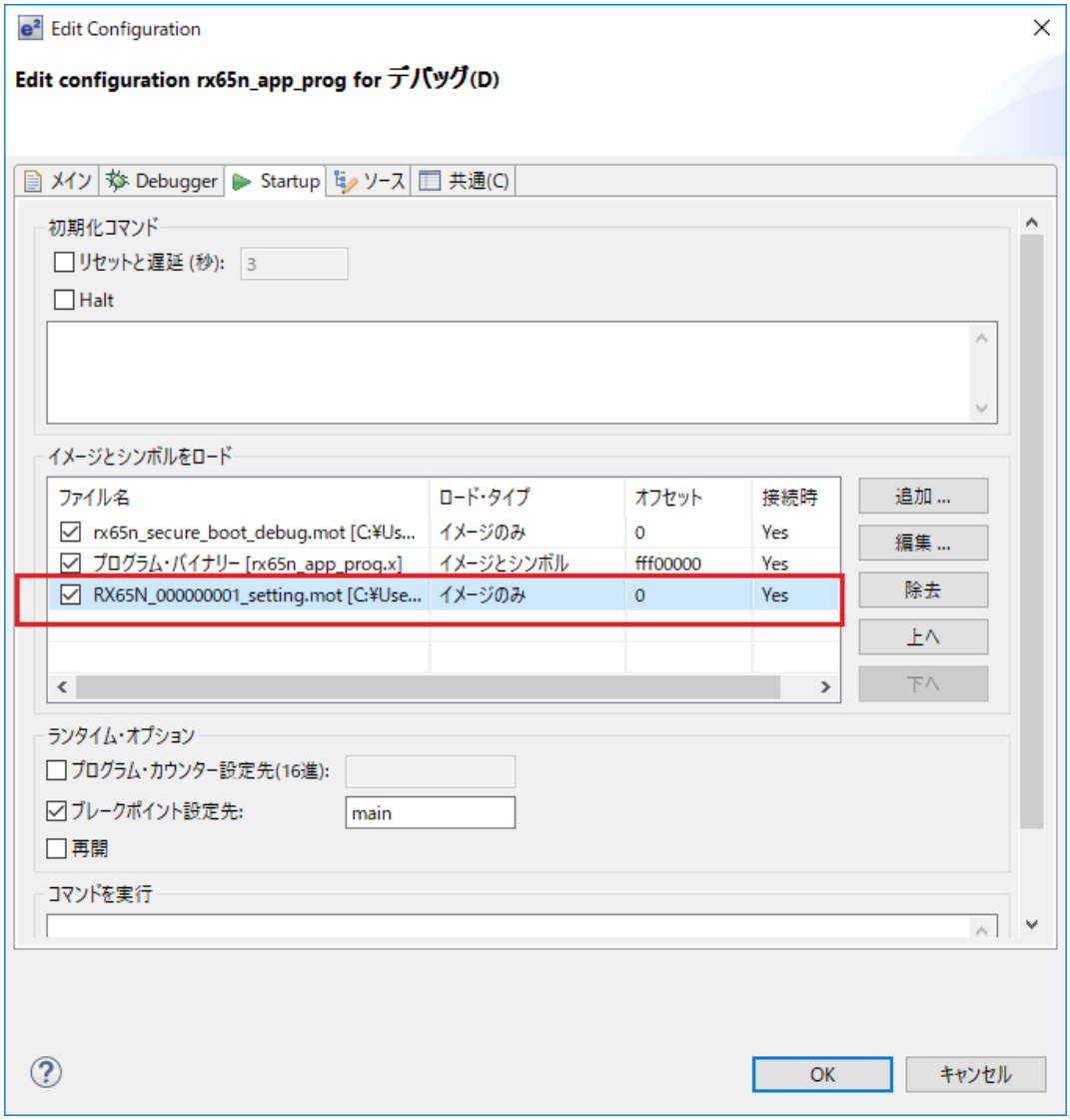

図 5.28　オプションデータの読み込み

5）アプリケーションプログラムを実行する
図 5.29～図 5.32 に手順を示します。

デバッグモードでプログラムを起動します。

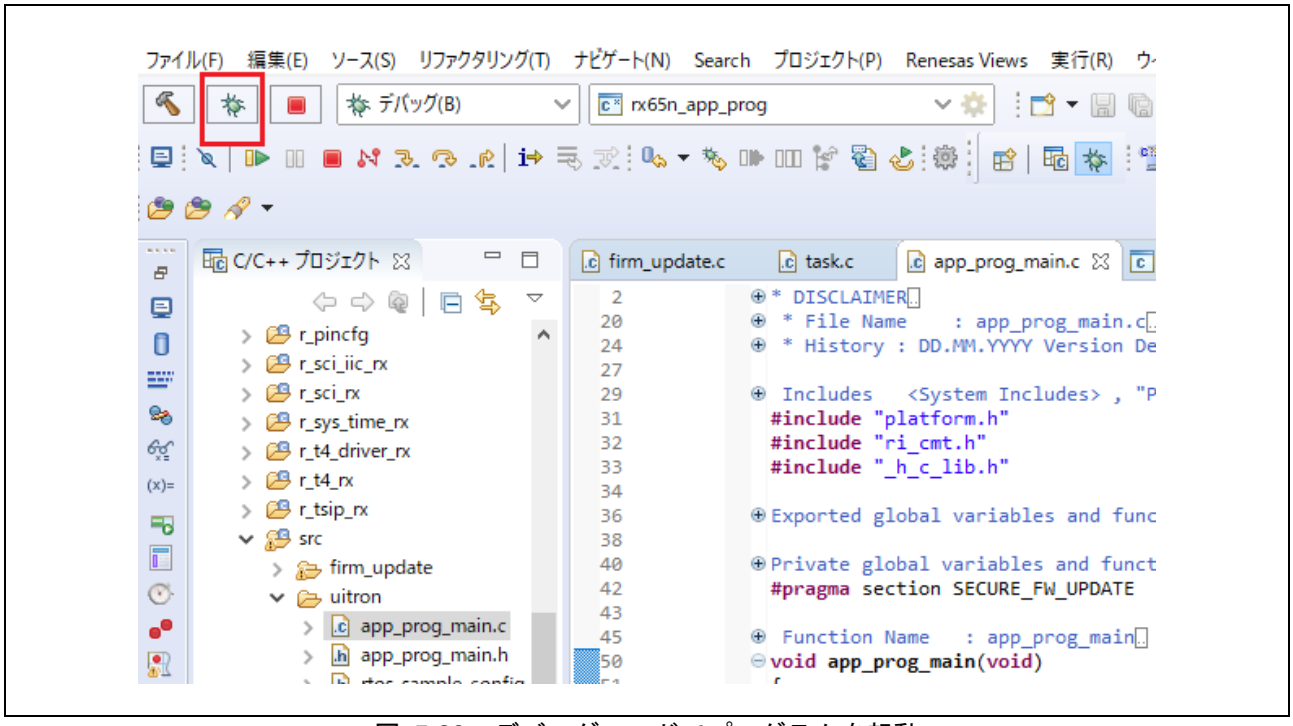

図 5.29　デバッグモードでプログラムを起動

アプリケーションプログラムのエントリ（app_prog_main）にブレークポイントを設定します。

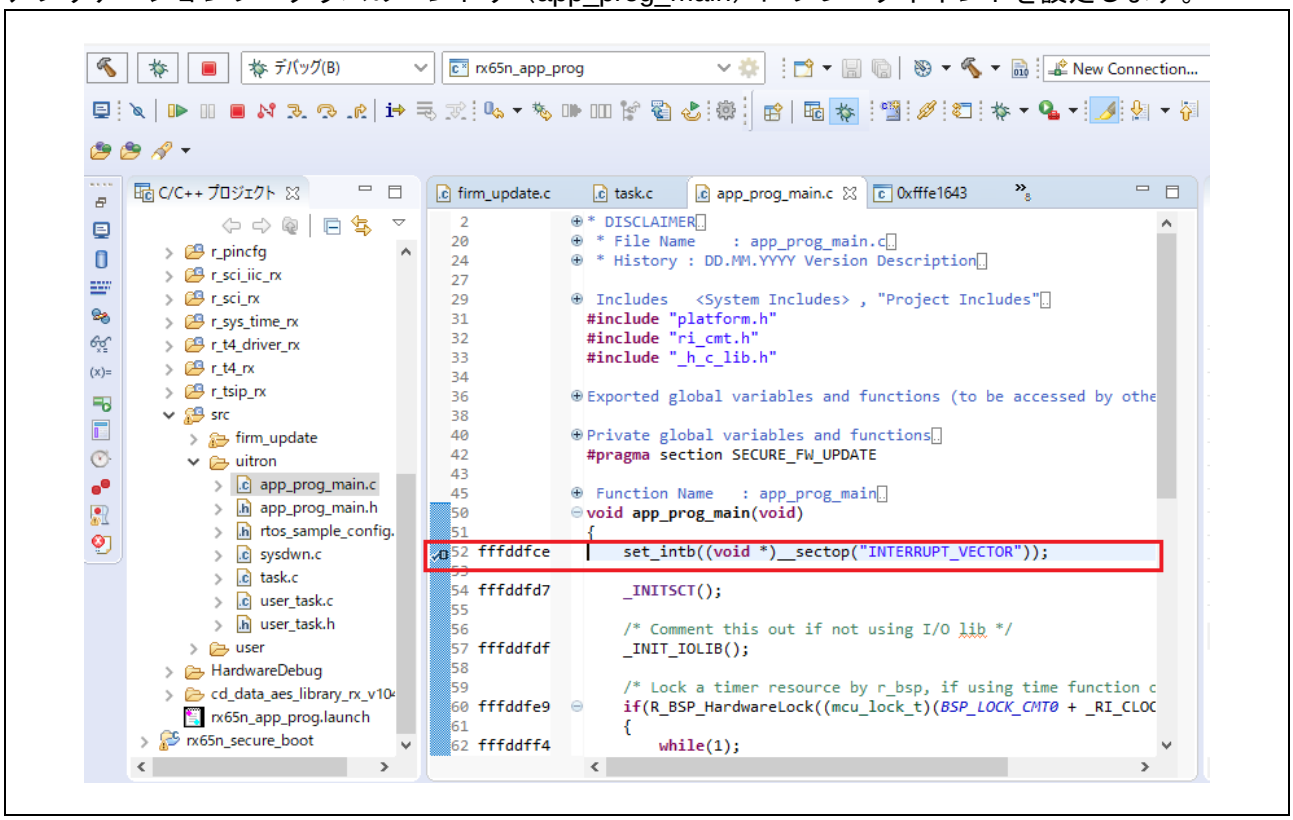

図 5.30　ブレークポイント設定

アプリケーションプログラムを開始します。

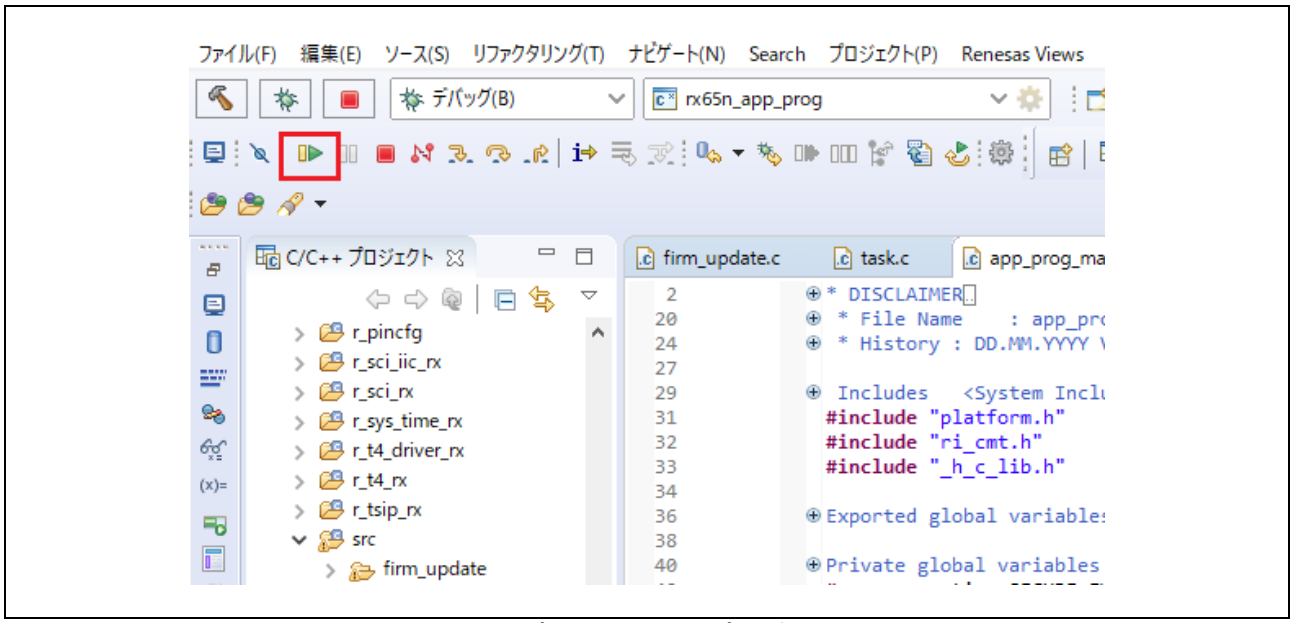

図 5.31 アプリケーションプログラムの開始

ブレークポイントで停止していることを確認してください。

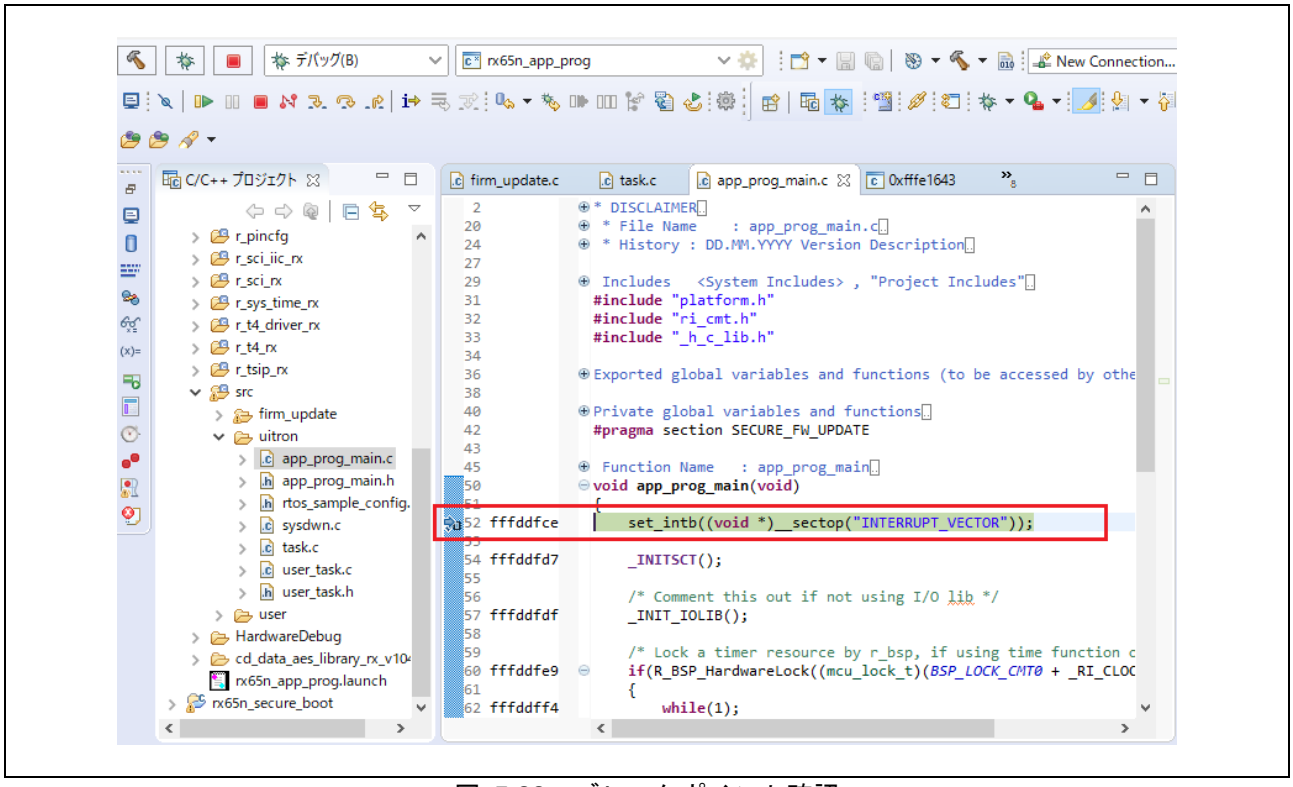

図 5.32 ブレークポイント確認

Appendix

A　サンプルプログラム

本ソリューションで提供しているアプリケーションプログラムに組み込まれているユーザプログラムの動作説明をします。

A.1　RTOS 版

RTOS 版のユーザプログラムを起動させたときの画面を図 A.0.1 に示します。
RTOS 版のアプリケーションプログラムでは、ユーザ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両方のタスクを起動しています。
RTOS 版のサンプルプログラムでは、周期的にカウンタを加算して、図 A.0.1 内①部分の画面表示を更新します。ファームウェアアップデートプログラムが実行された場合、図 A.0.1 内②部分にログを表示します。ログ表示エリアのスクロールバーはタッチ操作することができます。

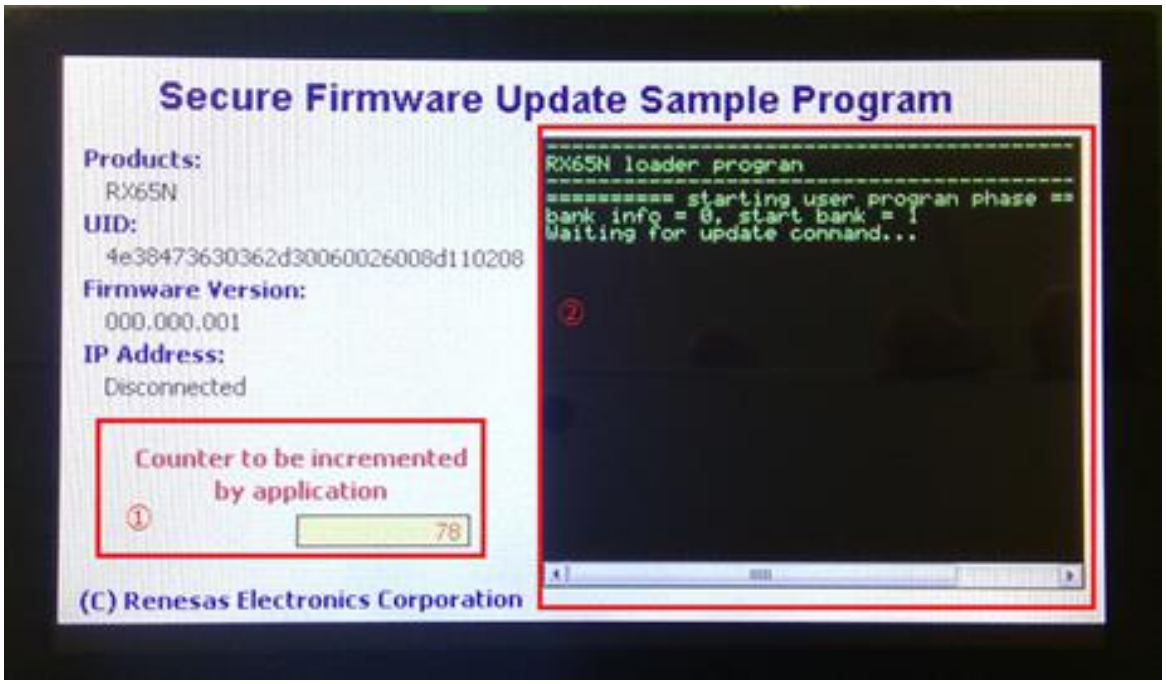

図 A.0.1 ユーザプログラム動作画面

A.2 NonOS 版

アプリケーションプログラム起動時に SW1（P03）の入力値を判定し、ユーザ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いずれかを起動します。選択方法は以下の通りです。
HI：（SW1 を押さずにリセット）→ユーザプログラムを起動します。
LOW：（SW1 を押しながらリセット）→ファームウェアアップデートプログラムを起動します。

図 A.0.2 に概略動作フローを示します。

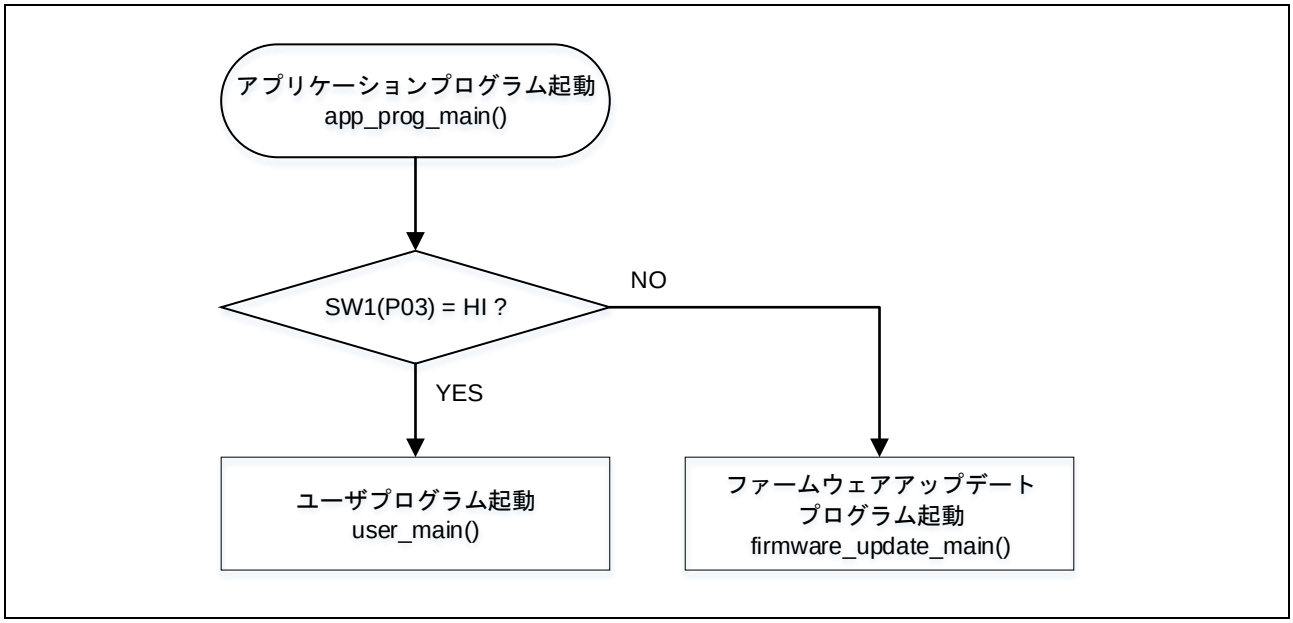

図 A.0.2 アプリケーションプログラムの概略動作フロー

図 A.0.3 に SW1 の配置を示します。

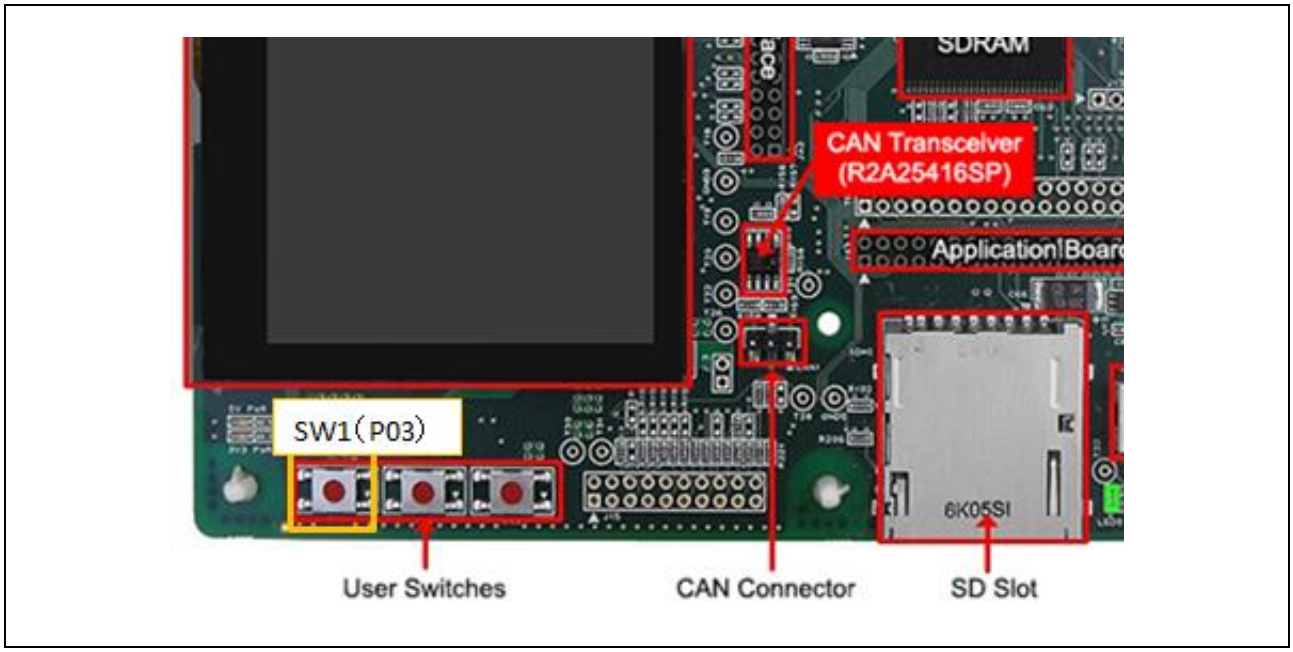

図 A.0.3 SW1 の配置

図 A.0.4 にユーザプログラムの起動画面を示します。サンプルプログラムでのユーザプログラムは図 A.0.4 のログを表示するのみのものとなっています。

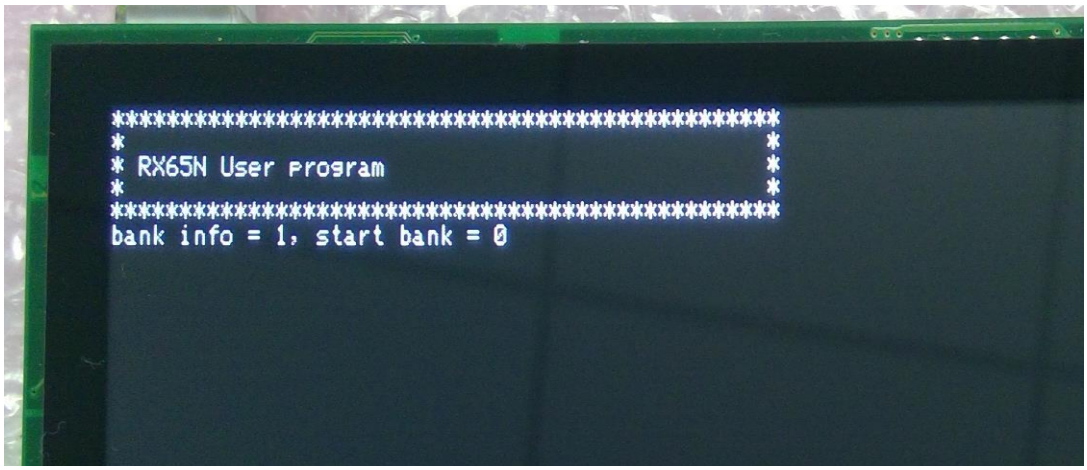

図 A.0.4　ユーザプログラムの起動画面

図 A.0.5 にファームウェアアップデートプログラムの起動画面を示します。アップデートが開始されるとログが表示されます。

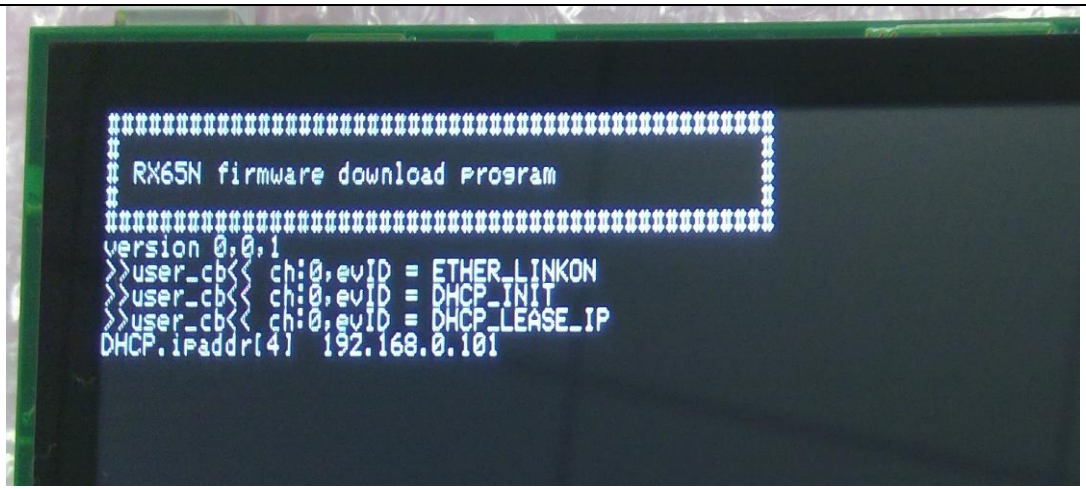

図 A.0.5　ファームウェアアップデートプログラムの起動画面

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	2018/12/19	68	初版発行

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. 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（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
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、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、内部ROM、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、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.4.0-1 2017.11)

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：https://www.renesas.com/contact/