RX23W グループ

OTA ファームウェア更新 サンプルプログラム

要旨

本アプリケーションノートは、RX23W 上で動作し、Bluetooth® Low Energy 無線通信機能による OTA(Over The Air)ファームウェア更新を実現したサンプルプログラムについて解説します。

動作確認デバイス

Target Board for RX23W

関連ドキュメント

- RX23W グループ Target Board for RX23W クイックスタートガイド(R20QS0014)
- RX23W グループ ユーザーズマニュアル ハードウェア編 (R01UH0823)
- Bluetooth Low Energy プロトコルスタック 基本パッケージ ユーザーズマニュアル (R01UW0205)
- RX23W グループ BLE Module Firmware Integration Technology (R01AN4860)
- RX23W グループ Bluetooth Low Energy プロファイル開発者ガイド (R01AN4553)
- RX23W グループ Bluetooth Low Energy アプリケーション開発者ガイド(R01AN5504)
- RX23W グループ 高速通信用サンプルプログラム (R01AN5437)

Bluetooth® のワードマークおよびロゴは、Bluetooth SIG,Inc. が所有する登録商標であり、ルネサス エレクトロニクス株式会社はこれらのマークをライセンスに基づいて使用しています。その他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。

目次

1. はじめに 5
1.1 パッケージ構成 5

2. サンプルプログラムの実行 7
2.1 動作確認環境 7
2.2 OTA Server のセットアップ 9
2.2.1 Target Board for RX23W へのファームウェア書き込み 9
2.3 Android 版 FWU_Client による更新 13
2.3.1 アプリケーションのインストール 13
2.3.2 ファームウェアファイルの転送 13
2.3.3 ファームウェアの更新手順 14
2.3.4 動作確認済みデバイス 20
2.4 iOS 版 FWU_Client による更新 21
2.4.1 アプリケーションのインストール 21
2.4.2 ファームウェアファイルの転送 21
2.4.3 ファームウェアの更新手順 24
2.4.4 動作確認済みデバイス 24
2.5 Python 版 FWU_Client による更新 25
2.5.1 対向の RX23W のセットアップ 25
2.5.2 Python Application のセットアップ 25
2.5.3 ファームウェアの更新手順 26

3. OTA Server の OTA によるファームウェア更新機能 29
3.1 システム構成 32
3.2 セクションレイアウト 33
3.3 ファームウェア更新動作 34
3.3.1 アプリケーション更新モード 34
3.3.2 BLE Protocol Stack 更新モード 35
3.4 プロファイル仕様 35
3.4.1 Renesas OTA Reset Service 36
3.4.2 Renesas OTA Service 36
3.4.3 データフォーマット 37
3.5 OTA の通信シーケンス 39
3.6 起動プログラムの切り替え 43
3.7 ブートローダ 44
3.8 リロケータ 45
3.8.1 プログラムの動作 45
3.8.2 電源遮断時の動作 46
3.9 ダウンローダ 47
3.9.1 Bluetooth LE の動作 47
3.9.2 ボンディング情報管理 47
3.9.3 Code Flash への書き込み 47
3.9.4 電源遮断時の動作 47
3.10 ユーザアプリケーション 48
3.10.1 ダウンローダへの切り替え動作 48

3.10.2 サンプルプログラムのユーザアプリケーション 48

4. OTA 更新対応プロジェクトの作成 49
4.1 ソースコード(r_ble_ota)の追加 49
4.2 セクション分割の設定 50
4.3 標準ライブラリの設定 54
4.4 更新用ファームウェアの出力設定 56
4.4.1 ファームウェア情報 JSON ファイルのプロパティ 58
4.5 Flash FIT モジュールの設定 59
4.6 Renesas OTA Reset Service の追加 60
4.7 ソースコードの各セクションへの割り当て 62
4.8 dbsct.c の編集 65
4.9 r_ble_ota コンフィグレーション設定 66
4.10 ユーザアプリケーションの設定 68
4.10.1 メイン関数の設定 68
4.10.2 ユーザアプリケーションの main 関数の登録と初期化セクションの設定 69
4.11 ユーザアプリケーションの機能追加 70
4.11.1 Renesas OTA Reset Service とダウンローダへの切り替え処理 70
4.11.2 ボンディングの実施 72
4.11.3 GATT Service の Service Changed Characteristic の Indication 処理 72

5. 更新用ファームウェアの確認 74

6. Android 版 FWU_Client 75
6.1 ビルド手順 75
6.2 デバッグ手順 75
6.3 Android 10.0 の API 仕様変更への対応 76

7. iOS 版 FWU_Client 77
7.1 動作確認環境 77
7.2 デバッグ手順 77
7.3 iOS の GATT データベースのキャッシュ機能 77

8. Python 版 FWU_Client 79
8.1 システム構成 79
8.2 制御コマンド 79
8.3 各コマンドの動作シーケンス 81
8.3.1 scan コマンド 81
8.3.2 conn コマンド 81
8.3.3 disconn コマンド 83
8.3.4 update コマンド 84
8.3.5 exit コマンド 85
8.4 OTA Client 86
8.4.1 ブロック図 86
8.4.2 アプリケーションパケット 87
8.4.3 コマンドラインインタフェースでの通信 92
8.4.4 app_lib の修正箇所 93

8.4.5 高速通信サンプルプログラムの使用 93
8.4.6 注意事項 93
8.5 Python Application 94
8.5.1 ブロック図 94
8.5.2 ログ出力 94

9. Version1.00 からの変更点 96
9.1 ユーザアプリケーションに Service Changed Characteristic を実装しました。 96
9.2 FIT のバージョン更新 96
9.3 OTA Server のコード変更 96

10. 制限事項/注意事項 97

11. 更新に失敗する場合の確認項目 98
11.1 Error Response が返る場合 98
11.2 更新開始直後に停止してしまう場合 98
11.3 Smart Configurator のコード生成機能を使用する場合 99

12. Appendix 103
12.1 OTA ファームウェア更新時間（参考値） 103

改訂記録 104

1. はじめに

OTA ファームウェア更新サンプルプログラムは、Bluetooth® Low Energy 無線通信機能を利用して、下記のいずれかの構成で RX23W のファームウェアを更新します。詳細は 2 章を参照してください。

- Android スマートフォンから更新対象の RX23W にファームウェアを更新
- iOS スマートフォンから更新審対象の RX23W にファームウェアを更新
- PC と接続した対向の RX23W から更新対象の RX23W にファームウェアを更新

RX23W のファームウェアは、下記のいずれかのプログラムを更新できます。OTA ファームウェア更新機能の詳細は 3 章を参照してください。

- ユーザアプリケーションのみ
- ユーザアプリケーションと Bluetooth Low Energy 通信部の両方

OTA ファームウェア更新に対応する RX23W 向けファームウェアの作成方法は、4 章を参照してください。

1.1 パッケージ構成

OTA ファームウェアサンプルプログラムのパッケージ構成を表 1-1 に示します。

表 1-1　OTA ファームウェアサンプルプログラムのパッケージ構成

r01an5910xx0111-rx23w-otafwup	
│　ble_sample_tbrx23w_ota_client.zip	OTA Client プログラム
│　ble_sample_tbrx23w_ota_server.zip	OTA Server プログラム
│　r01an5910ej0110-rx23w-otafwup.pdf	英文ドキュメント
│　r01an5910jj0110-rx23w-otafwup.pdf	日文ドキュメント
│	
├──Android	
│　　FWU_Client.apk	Android 版 FWU_Client のアプリケーションファイル
│　　FWU_Client.zip	Android 版 FWU_Client のプロジェクトファイル
│	
├──iOS	
│　　FWU_Client.zip	iOS 版 FWU_Client のプロジェクトファイル
│	
├──ProjectSetting	
│　│　section.esi	OTA Server のセクション情報のエクスポートファイル
│　│	
│　└──service	
│　　├──json	
│　　│　Renesas_OTA_Reset_Service_Client.json	QE for BLE の Renesas OTA Reset Service のクライアント用サービスファイル
│　　│　Renesas_OTA_Reset_Service_Server.json	QE for BLE の Renesas OTA Reset Service のサーバー用サービスファイル
│　　│　Renesas_OTA_Service_Client.json	QE for BLE の Renesas OTA Service のクライアント用サービスファイル
│　　│　Renesas_OTA_Service_Server.json	QE for BLE の Renesas OTA Service のサーバー用サービスファイル
│　　│	
│　　└──service api	
│　　　　r_ble_otac.c	QE for BLE の Renesas OTA Service のクライアント用サービス API
│　　　　r_ble_otac.h	
│　　　　r_ble_ota_resetc.c	QE for BLE の Renesas OTA Reset Service のクライアント用サービス API
│　　　　r_ble_ota_resetc.h	
│　　　　r_ble_ota_resets.c	QE for BLE の Renesas OTA Reset Service のサーバー用サービス API
│　　　　r_ble_ota_resets.h	
│	
├──PythonApplication	
│　　BinaryFileReader.py	バイナリ読み出しスクリプト
│　　ClientDeviceCommunication.py	OTA Client のパケット解析スクリプト
│　　FirmwareInfoReader.py	ファームウェア情報読み出しスクリプト

│　　CommandExecuter.py	コマンドを処理するスクリプト
│　　main.py	PythonApplication のメインスクリプト
│　　port.json	コムポートの設定ファイル
│　　SerialCommunication.py	シリアル通信スクリプト
│　　UserInterface.py	入出力スクリプト
│	
├──SampleFirmware	
│　├──Difference	
│　│　├──ota_sample_Version1.11_src_diff	更新用サンプルファームウェアのソースコード差分
│　│　│　├──r_ble_ota	
│　│　│　│　　r_ble_ota_config.h	
│　│　│　└──smc_gen	
│　│　│　　└──Config_BLE_PROFILE	
│　│　│　　　　app_main.c	
│　│　│　　　　gatt_db.c	
│　│　│　　　　gatt_db.h	
│　│　│　　　　r_ble_bas.c	
│　│　│　　　　r_ble_bas.h	
│　│　│	
│　│　└──ota_sample_Version1.12_src_diff	更新用サンプルファームウェアのソースコード差分
│　│　　└──r_ble_ota	
│　│　　　　r_ble_ota_config.h	
│　│　　　│	
│　│　　　└──downloader	
│　│　　　　　r_ble_ota_dl_cmt_driver.c	
│　│	
│　├──FWU Client	サンプルプログラム実行のための更新用ファームウェア
│　│　├──ota_sample_Version1.10	
│　│　│　　firmware.bin	
│　│　│　　firmware_information.json	
│　│　│	
│　│　├──ota_sample_Version1.11	
│　│　│　　firmware.bin	
│　│　│　　firmware_information.json	
│　│　│	
│　│　└──ota_sample_Version1.12	
│　│　　　firmware.bin	
│　│　　　firmware_information.json	
│　│	
│　└──mot	
│　　　ble_sample_tbrx23w_ota_client.mot	サンプルプログラムの実行ファイル
│　　　ble_sample_tbrx23w_ota_server.mot	サンプルプログラムの実行ファイル
│	
└──Tool	
CompareFirmware.py	ファームウェア情報比較スクリプト

2. サンプルプログラムの実行

本章は OTA ファームウェア更新サンプルプログラムを用いたファームウェアの更新手順について示します。本サンプルでは OTA によるファームウェアアップデートで更新される RX23W 用プログラム(OTA Server)と、その更新を行う Client プログラム(FWU_Client)を提供します。FWU_Client は、Android 版、iOS 版、PC ともう一台の Target Board を使用する Python 版の 3 つを用意しています。ファームウェア更新は下記のいずれかの構成で実行できます。

- Android スマートフォンから更新対象の RX23W にファームウェアを転送
- iOS スマートフォンから更新対象の RX23W にファームウェアを転送
- PC と接続した対向の RX23W から更新対象の RX23W にファームウェアを転送

本サンプルでは更新用ファームウェアとして、ファームウェアファイル(firmware.bin)と、JSON 形式のファームウェア更新情報ファイル(firmware_information.json)を使用します。これらのファイルは[プロジェクト名]_Version[アプリケーションバージョン]フォルダに格納されます。

以下に、更新用ファームウェアフォルダの構成(プロジェクト名:ota_sample,アプリケーションバージョン:1.10)を示します。

```
FWU Client
 ├──ota_sample_Version1.10
 │      firmware.bin
 │      firmware_information.json
```

図 2.1 更新用ファームウェアフォルダの構成

OTA ファームウェア更新のサンプルとして以下の 3 つのファームウェアを同梱しています。

表 2-1　OTA ファームウェア更新サンプル

ファームウェア	説明
ota_sample_Version1.10	更新を行うファームウェアです。 アプリケーションは LED Switch Service(LSS)*を持ちます。
ota_sample_Version1.11	ユーザアプリケーション更新用ファームウェア。 GATT データベースに Battery Information Service が追加されます。
ota_sample_Version1.12	Bluetooth LE 通信部分を更新するファームウェア。 ファームウェア受信中の LED 点滅機能が追加されます。

*LED Switch Service は、BLE FIT Module に同梱されているデモ用のサービスです。

本サンプルプログラムの実行では、ユーザアプリケーションのみの更新手順を示します。ユーザアプリケーションと Bluetooth LE 通信部分の更新も同様の手順で行えます。

本サンプルプログラムの実行で、Version1.10 から Version1.11 に更新します。GATT データベースに Battery Information Service(BAS)が追加されます。

2.1 動作確認環境

OTA ファームウェア更新サンプルプログラムの動作確認環境を示します。

表 2-2　サンプルプログラムの動作確認環境

環境	条件
RX23W 評価ボード	Target Board for RX23W 2 台 ※1
コンパイラ	C/C++ Compiler for RX Family V2.08.01
RX23W 開発環境	Renesas e^2 studio 2021-07
ホストマシン	Windows 10
スクリプトインタプリタ	Python 3.7.3
スマートフォン	Android 8.0.0 iOS14.0 以上
スマートフォン開発環境	Android Studio 4.1.1 Xcode 12.5.1

※1　対向 RX23W からファームウェアを転送する場合は 2 台、スマートフォンからファームウェアを転送する場合は 1 台

2.2　OTA Server のセットアップ

本節では、本サンプルを実行するための OTA Server のセットアップ方法を示します。Version1.10 のファームウェアを Renesas Flash Programmer を用いて Target Board for RX23W に書き込みます。

2.2.1　Target Board for RX23W へのファームウェア書き込み

LSS を含むファームウェアファイル" ble_sample_tbrx23w_ota_server.mot"を Target Board for RX23W に書き込みます。ファームウェアを RX23W に書き込むには、Renesas Flash Programmer (RFP)を使用します。

ファームウェアを書き込む際は以下に示すように、Target Board for RX23W 上の ESW1-2 を ON に切り替えて、micro-B コネクタの ECN と PC を接続します。

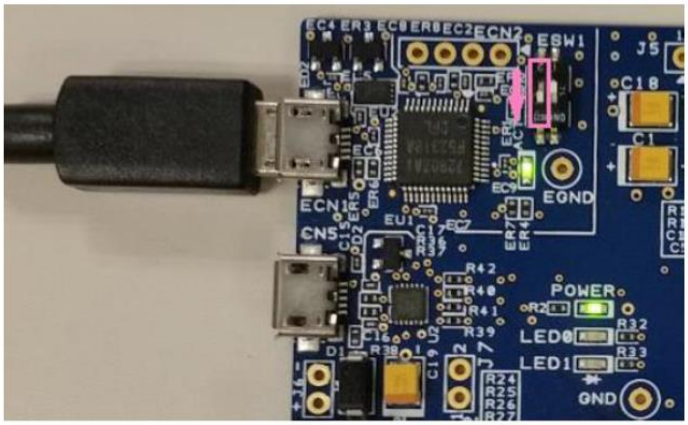

図 2.2　Target Board for RX23W の ESW1-2 を ON

RFP を起動して、メニューの"ファイル"→“新しいプロジェクトを作成…"を選択します。

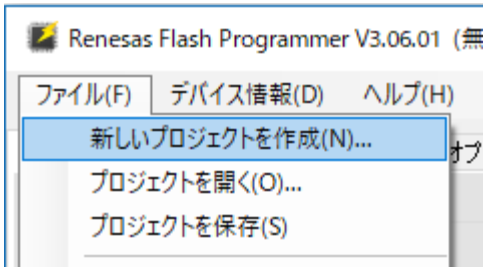

図 2.3　RFP の新しいプロジェクトを作成

新しいプロジェクトの作成ダイアログで、次を設定して"接続"ボタンをクリックします。

- マイクロコントローラ: RX200
- プロジェクト名 任意の名前
- 作成場所: 任意の場所
- 通信ツール: E2 emulator Lite
- 通信インタフェース: FINE
- 電源: 供給しない

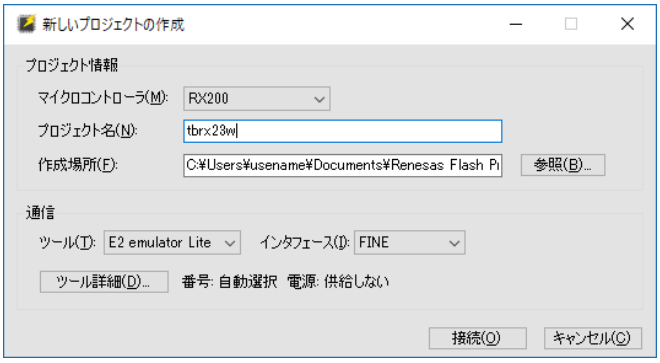

図 2.4　新しいプロジェクトの作成

OTA Server プログラム(ble_sample_tbrx23w_ota_server)はデータフラッシュを使用します。予期しない動作を防止するため、データフラッシュ上のデータを削除する必要があります。データフラッシュとコードフラッシュの全てのデータを消去するための設定を以下に示します。

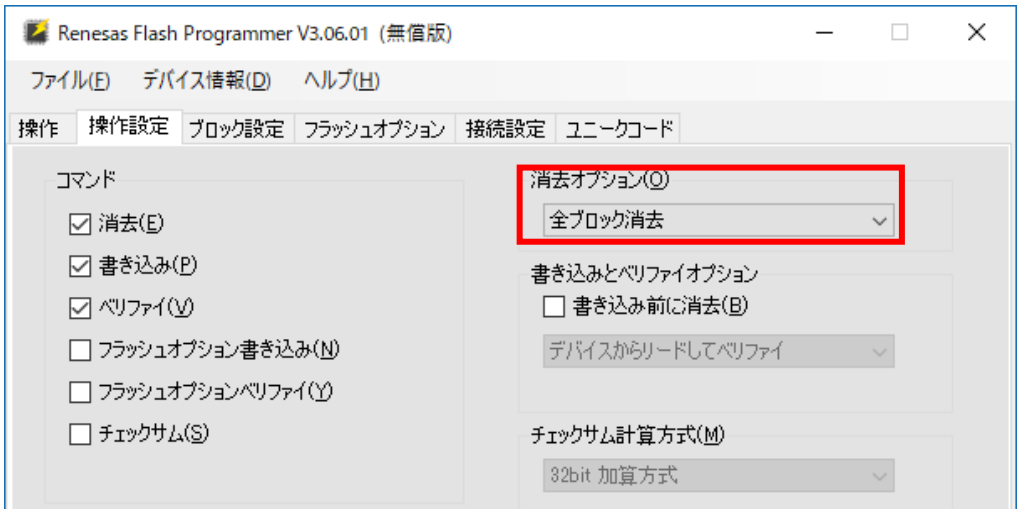

図 2.5　全てのデータフラッシュとコードフラッシュを消去する設定(操作設定タブ)

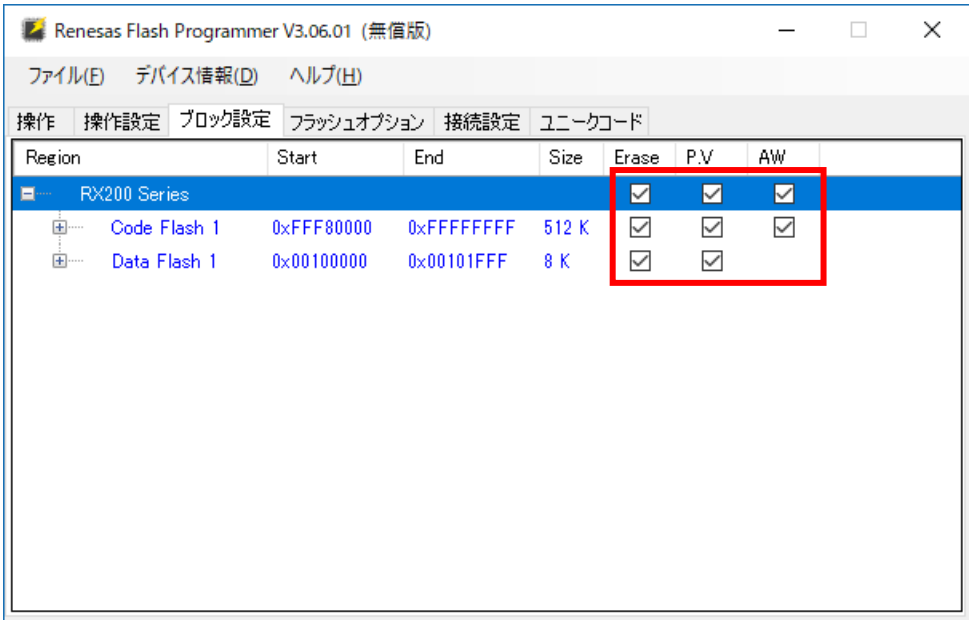

図 2.6　全てのデータフラッシュとコードフラッシュを消去する設定(ブロック設定タブ)

操作タブに移動して、ファームウェアファイル(ble_sample_tbrx23w_ota_server.mot)を選択し、スタートボタンをクリックしてファームウェアを書き込みます。

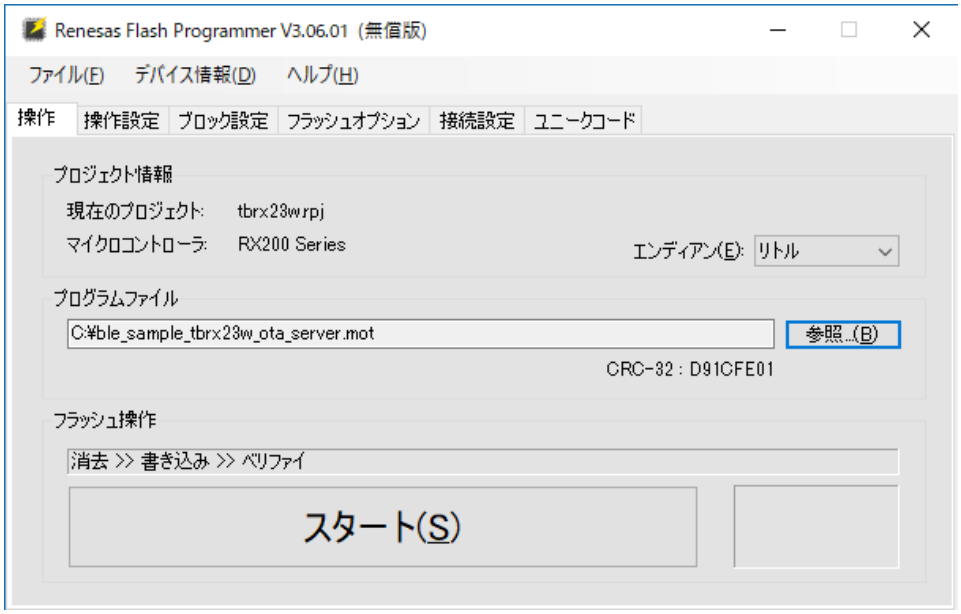

図 2.7　ファームウェアの書き込み画面

書き込みの完了後は、以下に示すように ESW1-2 を OFF に切り替えて、micro B コネクタの CN5 と PC に接続します。

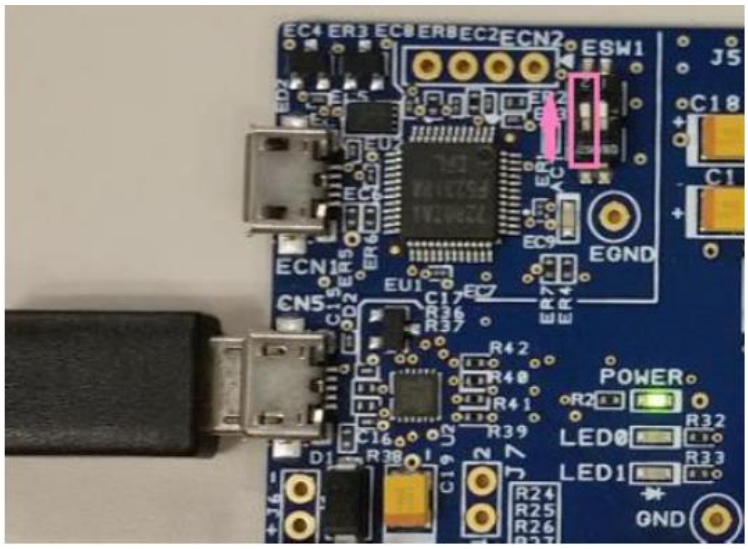

図 2.8　ターゲットボードを実行モードに切り替え

スマートフォンアプリの GATTBrowser を使用して OTA Server プログラムに接続すると、以下に示すように LED Switch Service が動作していることを確認できます。

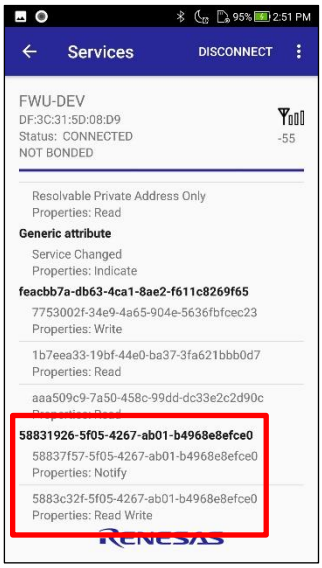

図 2.9　GATTBrowser との接続確認

スマートフォンアプリ GATTBrowser は Google Play で入手できます。

https://play.google.com/store/apps/details?id=com.renesas.ble.gattbrowser

GATTBrowser の詳細は以下を参照してください。

https://www.renesas.com/document/apn/gattbrowser-android-smartphone-application-instruction-manual

2.3　Android 版 FWU_Client による更新

2.3.1　アプリケーションのインストール

FWU_Client をスマートフォンにインストールします。同梱のアプリケーションパッケージファイル "FWU_Client.apk"をスマートフォンに転送しインストールします。

2.3.2　ファームウェアファイルの転送

図 2.10 に示すように ota_sample_Version1.11 フォルダをスマートフォンに転送します。スマートフォンの内部ストレージに FWU Client フォルダを作成し、本サンプルパッケージ内の"FWU Client"フォルダの ota_sample_Version1.11 をコピーしてください。

```
Internal Storage:¥FWU Client
 ├──ota_sample_Version1.11
 │          firmware.bin
 │          firmware_information.json
```

図 2.10　スマートフォンアプリのフォルダ構成

FWU_Client は、接続した OTA Server のプロジェクト名(BLE_OTA_PROJECT_NAME の設定値)と一致するフォルダのうちバージョン情報が最も新しいものを取得し更新を行います。ota_sample_Version1.12 のフォルダも転送するとユーザアプリケーションと Bluetooth LE 通信部分が更新されます。

2.3.3　ファームウェアの更新手順

2.3.3.1　スマートフォンアプリの操作手順

FWU_Client を起動すると、デバイスの検索画面が表示されます。

図 2.11　デバイス検索画面

OTA Server(デバイス名：FWU-DEV)をタップして接続します*。接続確立後、デバイス情報画面に接続した OTA Server のファームウェア情報が表示されます。画面下部に更新可能なモードとバージョンが表示されます。

*スマートフォンが保持している鍵情報と OTA Server の鍵が異なっていると接続できない場合があります。接続に失敗する場合は、スマートフォンデバイスに保持されている OTA Server の鍵情報を削除してやり直してください。

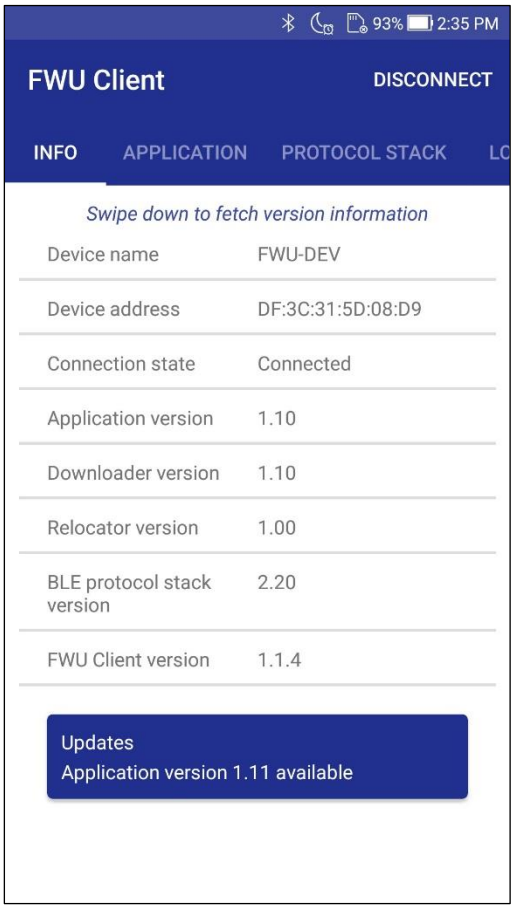

図 2.12　デバイス情報画面

画面をスワイプし、アプリケーション更新画面に移動します。更新画面では、接続した OTA Server と更新用ファームウェアのバージョン情報が表示されます。アプリケーション部分が更新可能な場合は、APPLICATION UPDATE ボタンが青色で表示されます。Bluetooth LE 通信部分の更新時は、PROTOCOL STACK タブを開きます。

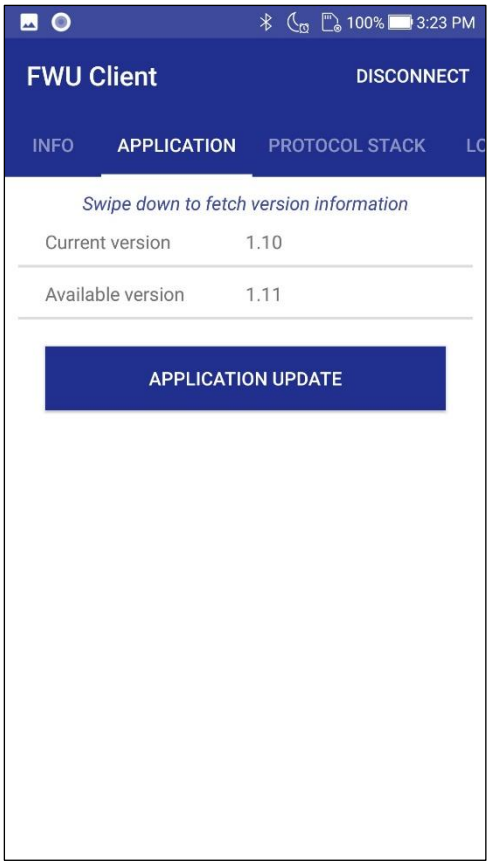

図 2.13　アプリケーション更新画面

APPLICATION UPDATE ボタンをタップしてファームウェアの更新を開始します。更新の進捗状況がプログレスバーに表示されます。更新の詳細は LOGS タブに表示されます。

更新時間は、通信状況にもよりますが、アプリケーションの更新には 2～5 分程度です。

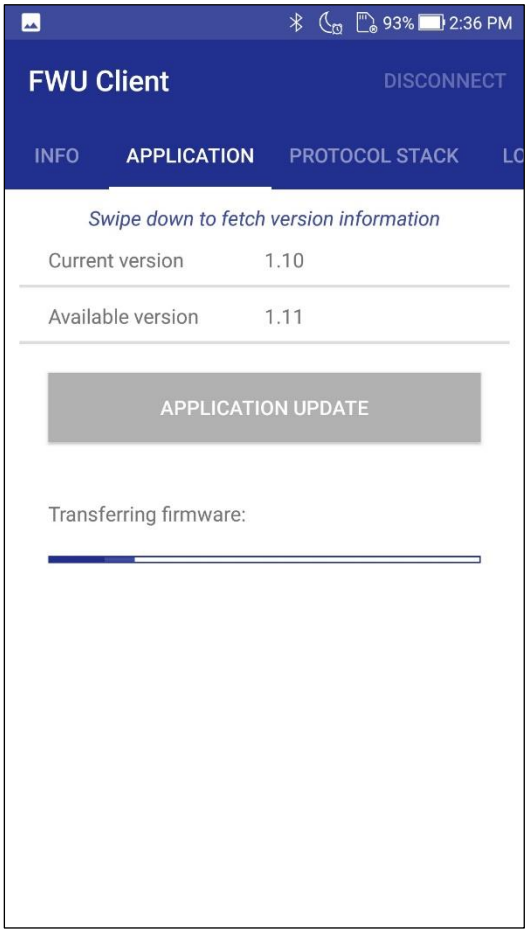

図 2.14　アップデート進行中画面

更新が成功すると OTA Server は再起動します。FWU_Client はデバイス検索画面に戻ります。更新中に問題が発生した場合は、原因がポップアップ表示され、FWU_Client はデバイス検索画面に戻ります。

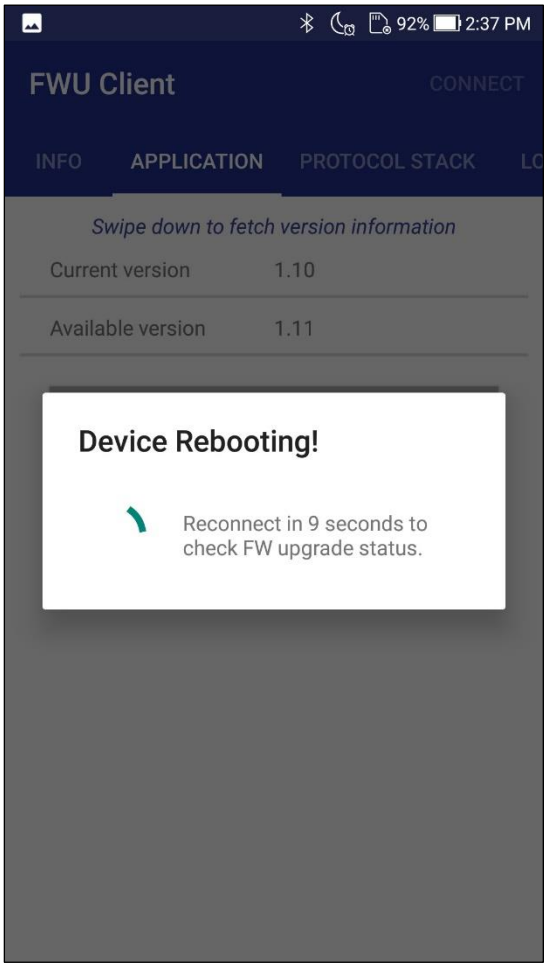

図 2.15　更新完了画面

2.3.3.2　アップデートの確認方法

デバイス検索画面からもう一度 OTA Server に接続し、デバイス情報画面で OTA Server のファームウェアのバージョン情報を確認します。

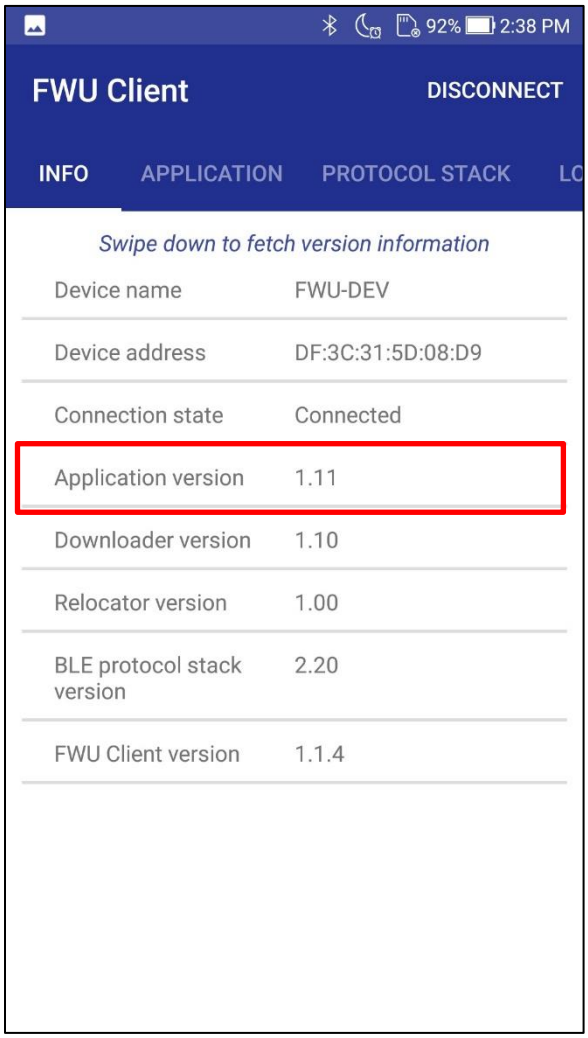

図 2.16　アップデート完了の確認

また、GATTBrowser から接続し Battery Service を確認します。

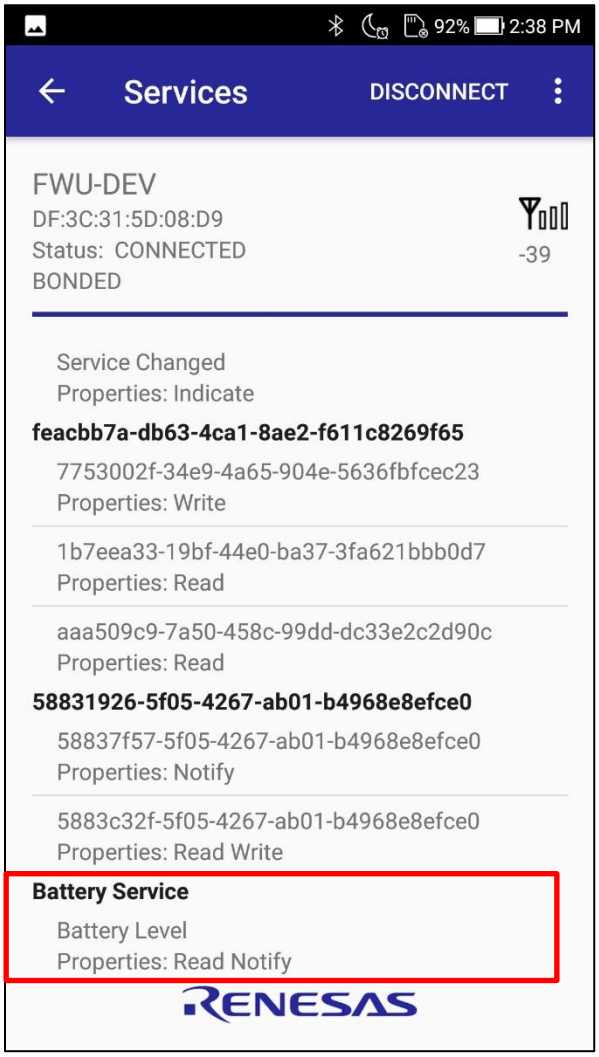

図 2.17　追加された Battery Information Service の確認

2.3.4　動作確認済みデバイス

動作確認済みの Android デバイスは以下の通りです。

表 2-3　動作確認デバイス

メーカ	Device name	Android Version
ASUS	ZenFone5	8.0.0
ASUS	ZenFone4	8.0.0
SHARP	SH-M05	8.0.0

2.4　iOS 版 FWU_Client による更新

FWU_Client のインストールには、Apple Developer Program への登録が必要です。Apple の公式サイト(https://developer.apple.com)をご覧ください。

2.4.1　アプリケーションのインストール

Xcode で FWU_Client プロジェクト(FWU_Client.xcodeproj)を開きます。

プロジェクトの Signing & capabilities タブから Team と Bundle identifier を環境に合わせて設定します。iOS デバイスを接続しプロジェクトスキームから iOS Device を選択します。Build ボタンを押してアプリケーションを起動します。

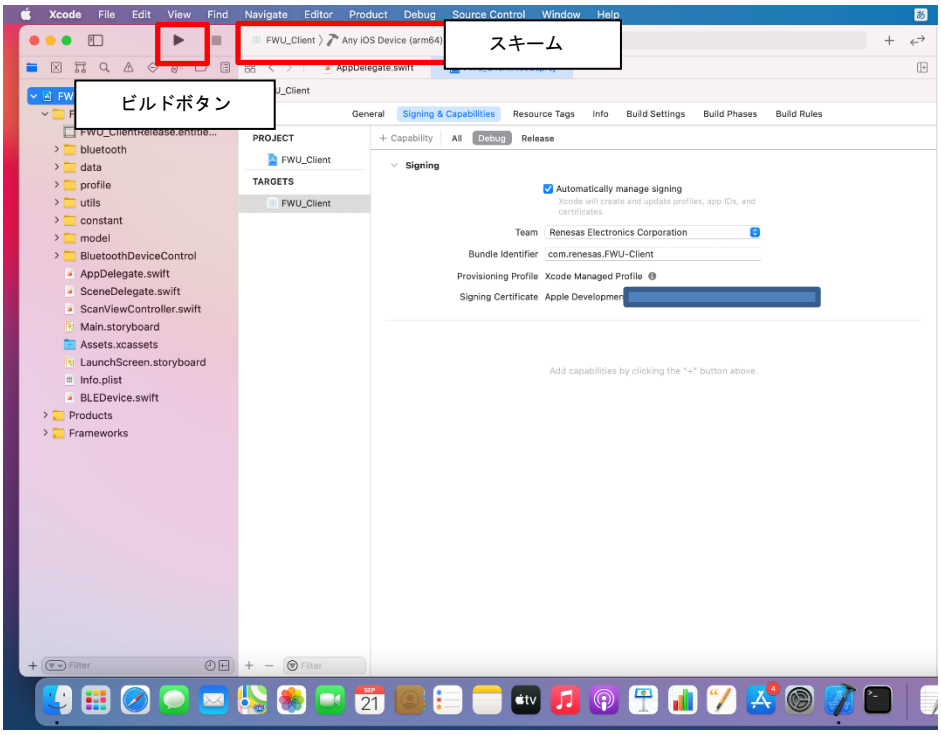

図 2.18 Xcode の設定画面

2.4.2　ファームウェアファイルの転送

iOS デバイスに更新用ファームウェアを転送する方法は以下の 2 つです。

- Finder または iTunes から転送
- iCloud から転送

FWU_Client に格納されているファームウェアは、iOS の「ファイル」アプリから確認できます。

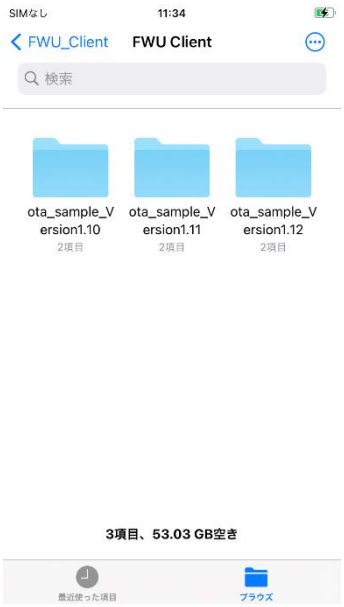

図 2.19 「ファイル」アプリによる更新ファイルの確認

2.4.2.1 Finder または iTunes から転送

PC と iOS デバイスを有線で接続します。Finder または iTunes の Files タブの FWU_Client アプリ領域に FWU Client フォルダをドラッグ&ドロップして転送します。FWU Client フォルダには更新用ファームウェアである firmware.bin と firmware_information.json が含まれるフォルダを格納します。

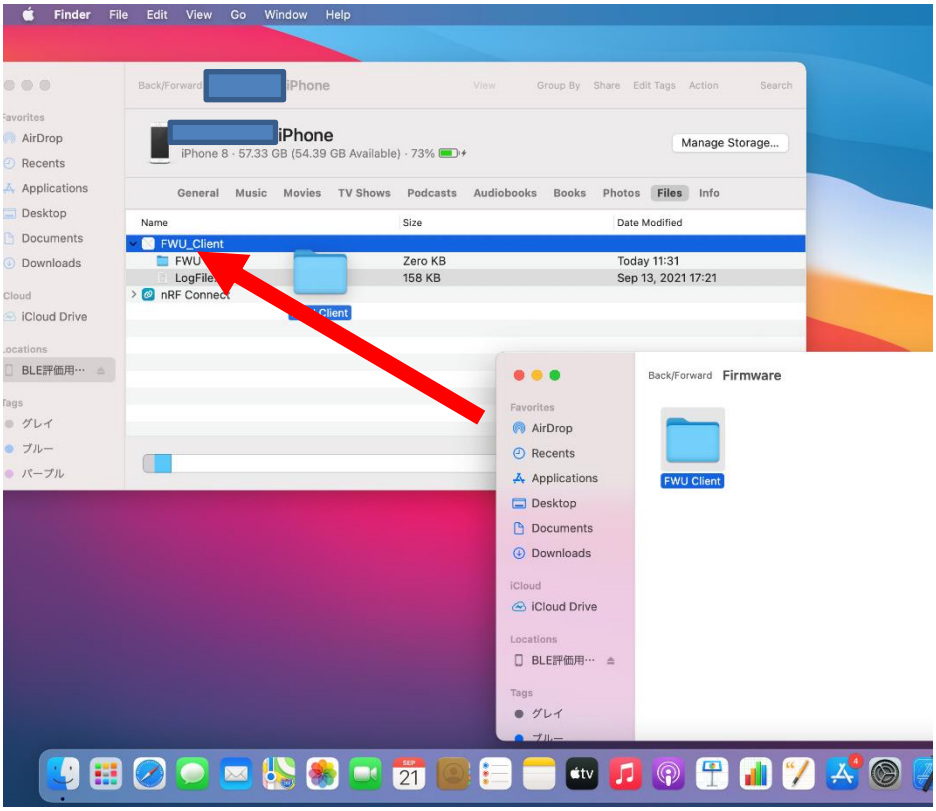

図 2.20 Finder を利用した更新用ファームウェアの転送

2.4.2.2 iCloud から転送

iOS デバイスに登録された Apple ID の iCloud にアクセスします。iCloud 上の任意のディレクトリに、FWU Client フォルダをアップロードします。

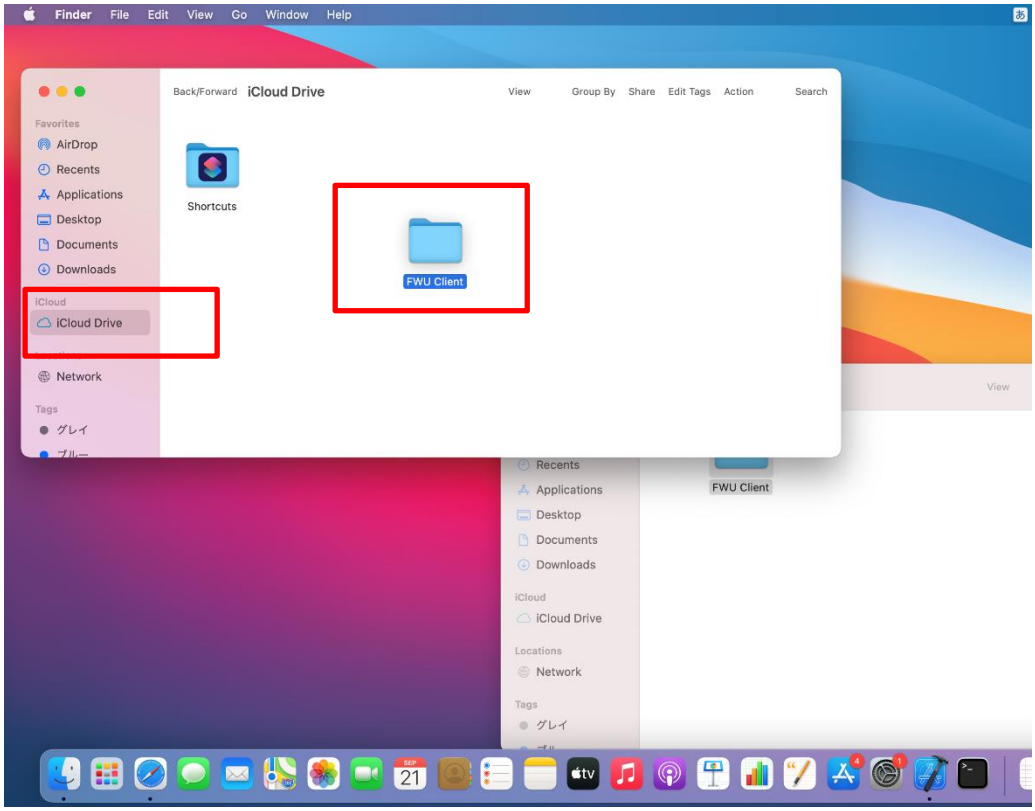

図 2.21 iCloud へのファイルの転送

iOS デバイスの「ファイル」アプリから、iCloud にアクセスし更新用のファイルを確認します。ダウンロードボタンをタップし、デバイスに保持します。

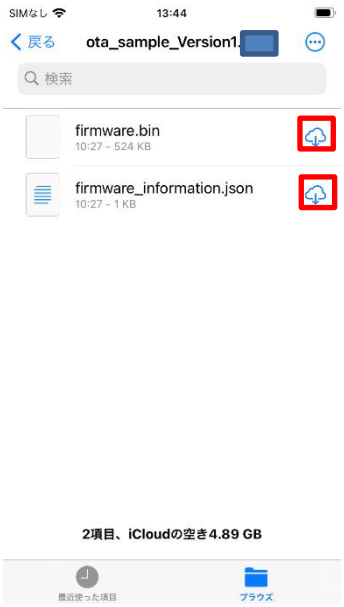

図 2.22 iCloud からデバイスへファイルのダウンロード

FWU_Client を起動します。＋ボタンから iCloud Update をタップし、転送した更新用ファームウェアフォルダを選択します。

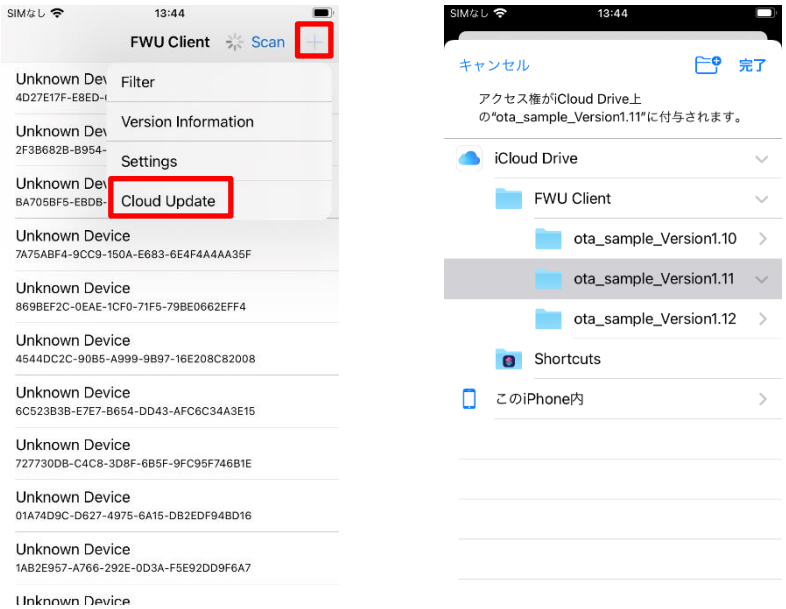

図 2.23 iCloud から FWU_Client の領域へのコピー

選択したファームウェアフォルダが、FWU_Client のアプリ領域にコピーされます。この操作は接続後デバイス情報画面から行えます。

2.4.3　ファームウェアの更新手順

更新の実行方法については、Android 版と同様です。2.3 節をご覧ください。iOS 版 FWU_Client によるアプリケーション部分の更新時間は、30 秒程度です。

本バージョンの OTA Server は、iOS の GATT データベースのキャッシュを削除するために、接続後に Service Changed Characteristic の Indication を行います。iOS 版 GATTBrowser は、Service Changed Characteristic の Indication を受けると切断するため、更新の確認には Android 版 GATTBrowser をお使いください。

2.4.4　動作確認済みデバイス

表 2-4　動作確認デバイス

Device name	iOS Version
iPhone 8	14.7.1

2.5 Python 版 FWU_Client による更新

Python 版 FWU_Client は、PC と Target Board for RX23W を使用して OTA Server のファームウェアアップデートを実現します。本節では、本サンプルパッケージ同梱の

- Python Application プログラム (Python Application)
- OTA Client プログラム(ble_sample_tbrx23w_ota_client)

を使用します。

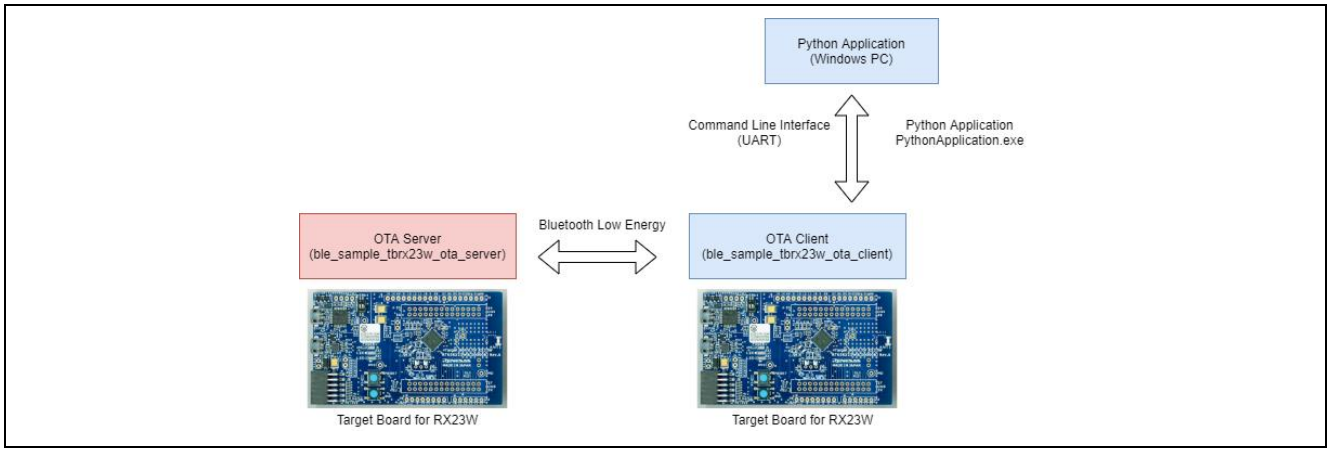

図 2.24 Python 版 FWU_Client のシステム構成

2.5.1 対向の RX23W のセットアップ

2.2 節の手順を参照し、OTA Client のファームウェア"ble_sample_tbrx23w_ota_client.mot"を OTA Server とは別の Target Board for RX23W に書き込み実行します。

OTA Client は、PC で動作する Python Application から更新用ファームウェアを受け取ります。USB ケーブルで OTA Client が書き込まれた Target Board for RX23W の CN5 を PC に接続します。

2.5.2 Python Application のセットアップ

Python Application の実行には PySerial モジュールが必要です。pip などを用いてインストールします。

Python Application フォルダ内の”port.json”を編集し OTA Client の serial port 番号を設定します。OTA Client の baud rate は 115200 に設定されています。

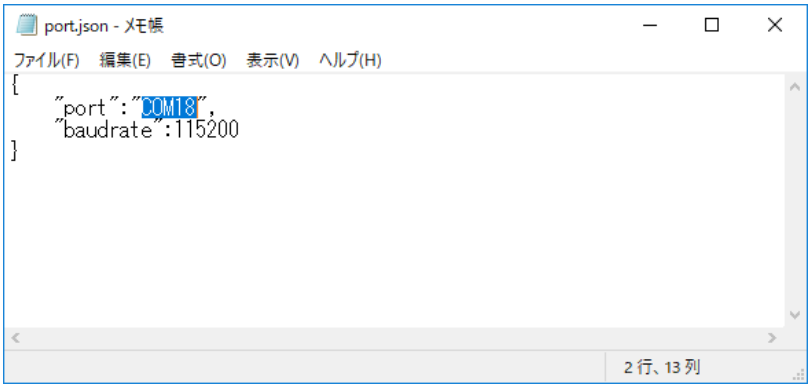

図 2.25 port.json の port 番号設定

2.5.3　ファームウェアの更新手順

コマンドプロンプトから main.py を実行します。

赤枠で示したコマンドを順に実行します。conn コマンドの青枠”df:3c:31:5d:08:d9 1”の部分は OTA Server の BD アドレスを入力します。スキャン後の BD ADDRESS 及び BD Address Type に表示されます。

```
C:¥r01an5910xx0110-rx23w-otafwup¥PythonApplication>python main.py
Enter Command
scan 1 1 9 FWU-DEV

BD ADDRESS : df:3c:31:5d:08:d9
BD Address Type : 1
Advertise Raw Data : 02,01,06,08,09,46,57,55,2d,44,45,56

BD ADDRESS : df:3c:31:5d:08:d9
BD Address Type : 1
Advertise Raw Data : 08,09,46,57,55,2d,44,45,56

SCAN END
Enter Command
conn 5 df:3c:31:5d:08:d9 1

ConnectionSuccess
BD ADDRESS : df:3c:31:5d:08:d9
ProjectName : ota_sample
ApplicationVersion : 01.10
DownloaderVersion : 01.10
RelocatorVersion : 01.00
BLE Protocol Stack version : 02.20
Server Activate Mode : User Application Activate

Enter Command
update app C:¥r01an5910xx0110-rx23w-otafwup¥SampleFirmware¥FWU Client¥ota_sample_Version1.11
address: fff90800 progress : 100.0% 017/017
UPDATE SUCCESS
Update Time: 17.73 sec
Enter Command
```

図 2.26 Python Application の実行画面

2.5.3.1 アップデート完了の確認方法

conn コマンドを再度実行し、OTA Server に接続します。Application Version が更新されていることを確認します。確認後、disconn コマンドと exit コマンドを実行し Python Application を終了します。

```
Update Time: 17.73 sec
Enter Command
conn 5 df:3c:31:5d:08:d9 1

ConnectionSuccess
BD ADDRESS : df:3c:31:5d:08:d9
ProjectName : ota_sample
ApplicationVersion : 01.11
DownloaderVersion : 01.10
RelocatorVersion : 01.00
BLE Protocol Stack version : 02.20
Server Activate Mode : User Application Activate

Enter Command
disconn
Disconnected
Enter Command
exit
Serial Port Closing...
```

図 2.27 アップデート完了の確認

また、GATTBrowser から接続し Battery Service の追加を確認します。

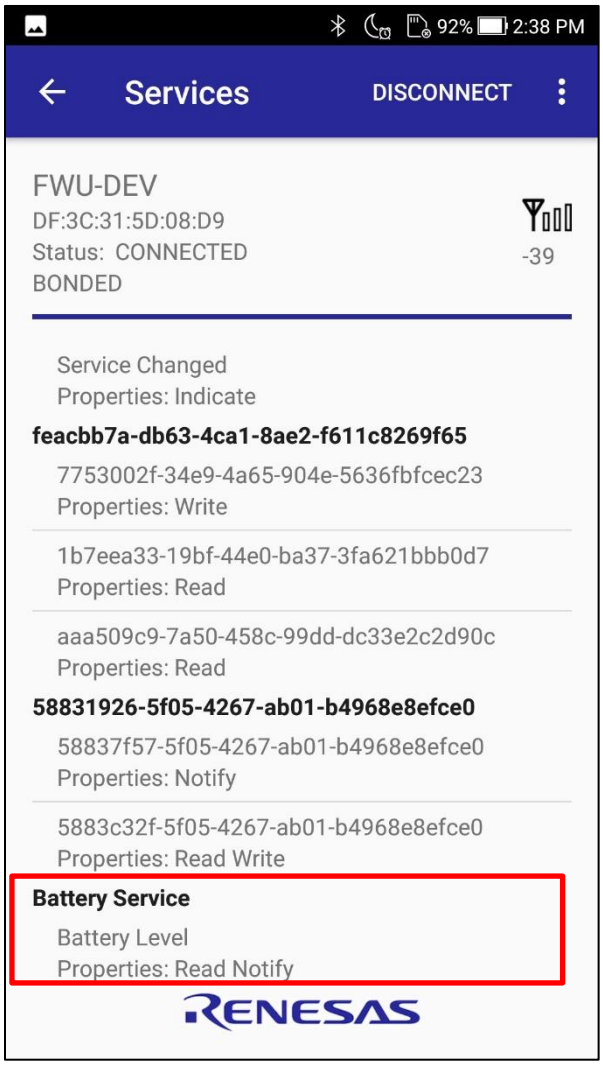

図 2.28　追加した Battery Information Service の確認

3. OTA Server の OTA によるファームウェア更新機能

本章はサンプルプログラムで提供する OTA Server の OTA ファームウェア更新機能の詳細について示します。本サンプルは、OTA によるファームウェア更新を、独立した 3 つのプログラムとブートローダを用いて実現します。BLE プロトコルスタックは、ユーザアプリケーションとファームウェア受信用プログラム(ダウンローダ)で共有します。

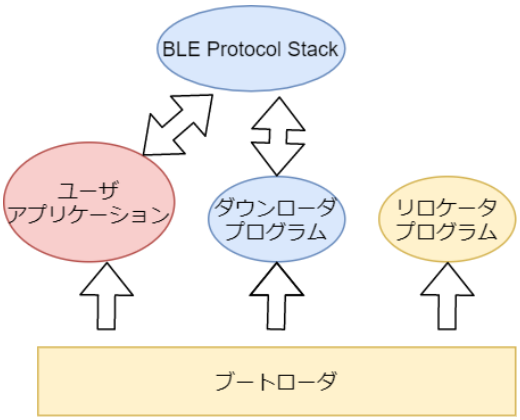

図 3.1 OTA ファームウェア更新機能ソフトウェアの概念図

- ユーザアプリケーション

  ユーザアプリケーションは通常運用時の Bluetooth LE アプリケーションです。

- ダウンローダ

  ダウンローダは BLE プロトコルスタックを利用して Client から更新用ファームウェアを受信します。ダウンローダはコードフラッシュに受信したファームウェアを保持します。

- リロケータ

  リロケータは、ダウンローダが受信したファームウェアを実行するための正しい領域に書き込みます。

- ブートローダ

  ブートローダはデータフラッシュに書き込まれた起動プログラムのフラグ情報に読み出し、プログラムを実行します。

本サンプルのファームウェア更新機能はアプリケーションのみの更新と Bluetooth LE 通信部とアプリケーションの更新の 2 つの更新モードがあります。OTA によるファームウェアの更新の簡略図を示します。

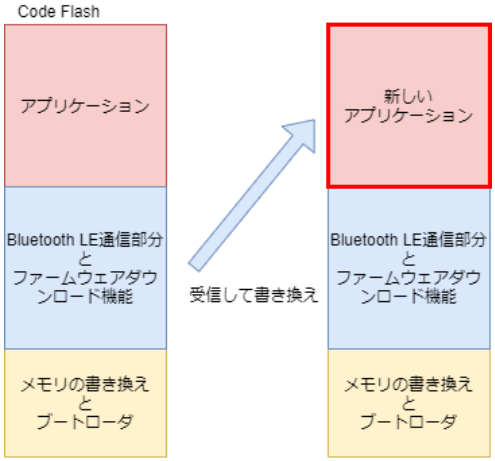

図 3.2 アプリケーション部分のみの更新

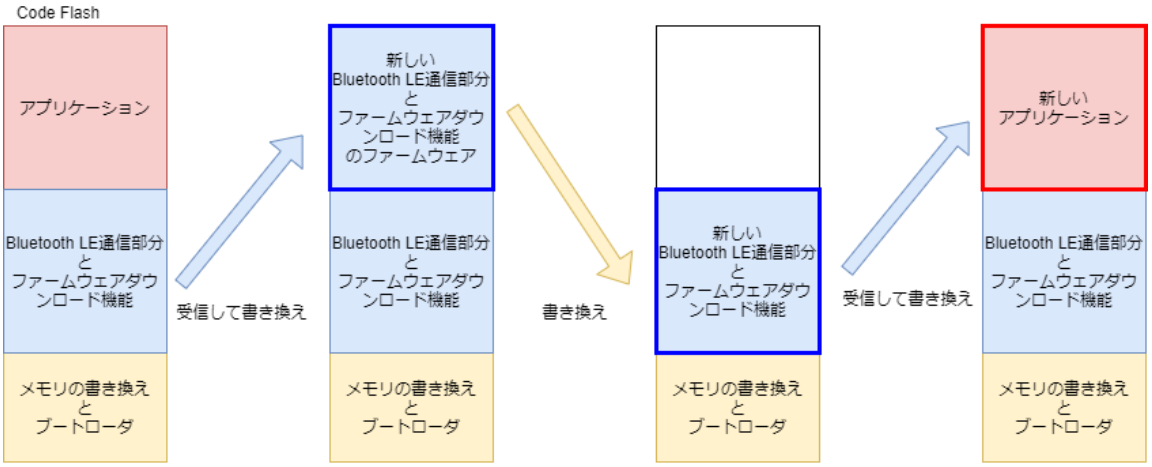

図 3.3 Bluetooth LE 通信部分とアプリケーションの更新

アプリケーション領域と更新用ファームウェアの保持領域を共通化する事により、内蔵 Flash のみで、Bluetooth Low Energy 通信部分を更新が可能です。

次に、OTA によるファームウェアの更新を行う際の 3 つのプログラムの遷移図を示します。ユーザアプリケーションは、GATT の Renesas OTA Reset Service を利用して、ユーザから更新開始されます。ダウンローダは、Renesas OTA Service を利用してクライアントから更新用のファームウェアを受け取ります。ダウンローダはリロケータを起動し、ファームウェアをコードフラッシュの正しい領域に再配置します。アプリケーションのみの更新の場合はユーザアプリケーションを起動します。Bluetooth LE 通信部分を更新する場合には、もう一度ダウンローダを起動しアプリケーション部分を受け取ります。

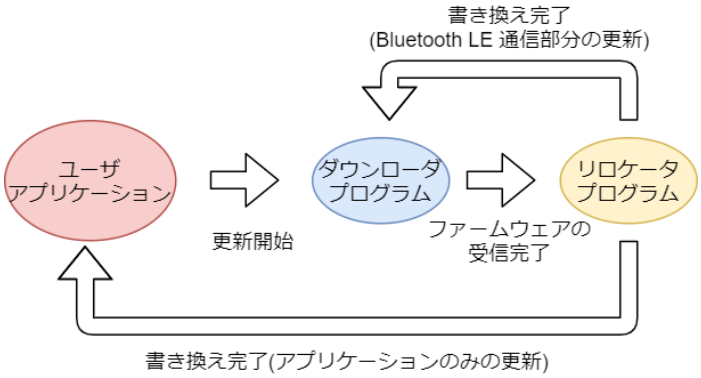

図 3.4 ファームウェア更新中のプログラムの遷移

3.1　システム構成

OTA ファームウェア更新サンプルプログラムのシステム構成を示します。

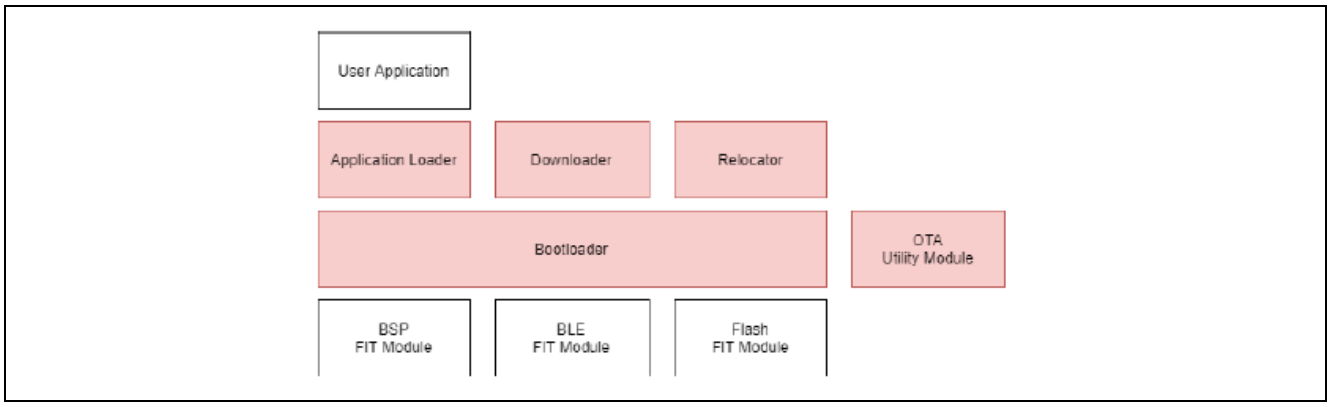

図 3.5　OTA ファームウェア更新サンプルプログラムのシステム構成

OTA ファームウェア更新機能が使用する FIT モジュールの詳細については下記を参照してください。

- RX ファミリ ボードサポートパッケージ(BSP) FIT モジュール

  https://www.renesas.com/document/apn/rx-family-board-support-package-module-using-firmware-integration-technology
- RX23W グループ BLE FIT モジュール

  https://www.renesas.com/document/apn/rx23w-group-ble-module-firmware-integration-technology-application-note
- RX23W グループ Flash FIT モジュール

  https://www.renesas.com/document/apn/rx-family-flash-module-using-firmware-integration-technology

表 3-1　使用する周辺機能

周辺機能	詳細	用途
Bluetooth Low Energy	-	OTA ファームウェア更新シーケンスの実行
Data Flash	ブロック 1	実行プログラムのフラグ保持
CMT	チャネル 0	OTA ファームウェア更新シーケンス中のタイムアウト監視 ダウンローダプログラムを実行に使用します。ユーザアプリケーションでの使用を制限しません。

3.2　セクションレイアウト

OTA ファームウェア更新サンプルプログラムの ROM/RAM セクション構成を図 3.6 に示します。セルフプログラミング機能に対応するため、各プログラムはコードフラッシュ上で分割されたセクションに配置されます。RAM はセクションオーバーレイ機能を使用し、0x00001404~0x0000BA00 の領域を各プログラムで共有します。

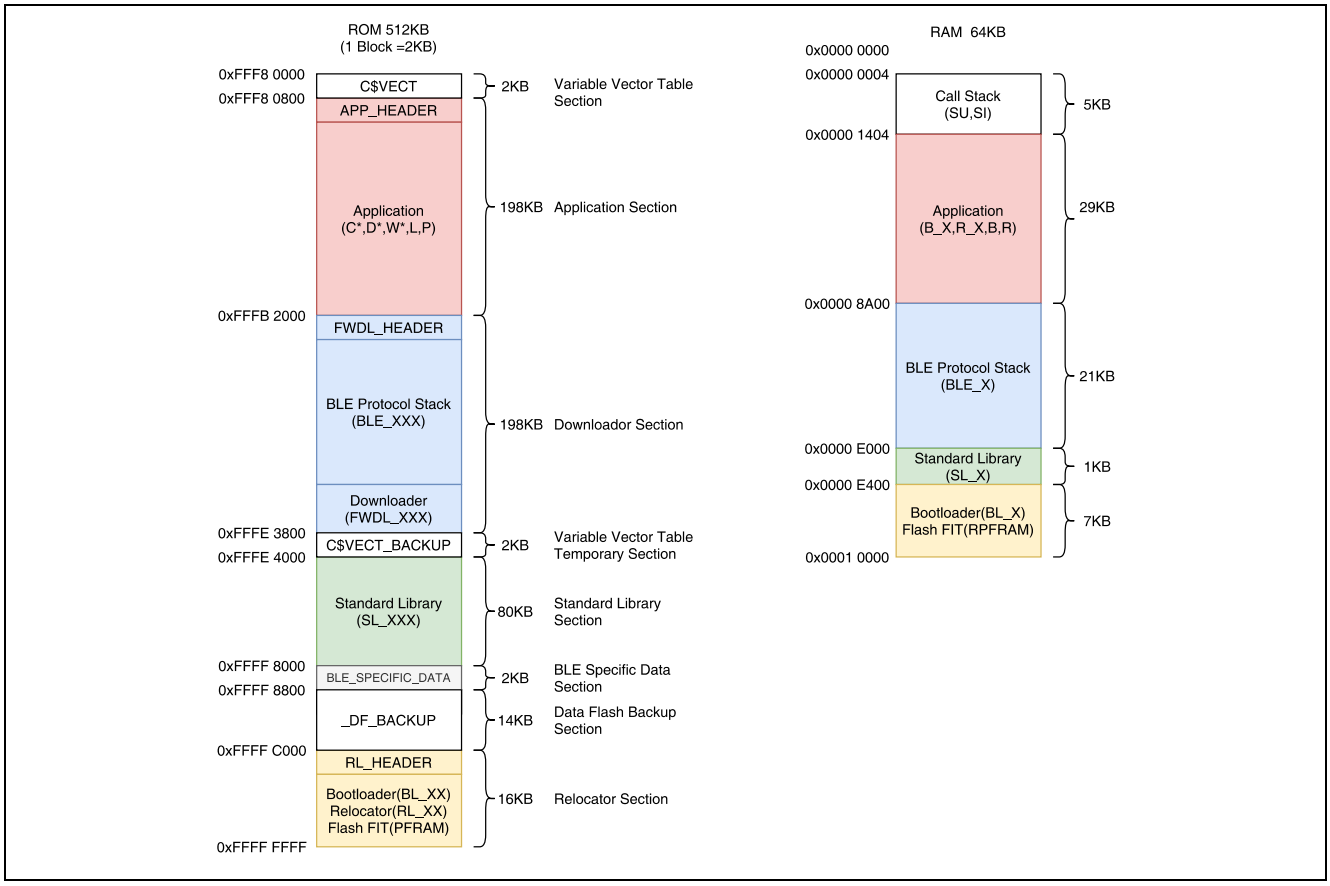

図 3.6　OTA ファームウェア更新サンプルプログラムの ROM/RAM セクションレイアウト

3.3 ファームウェア更新動作

OTA ファームウェア更新機能は、下記のいずれかの動作モードで更新することができます。

- アプリケーション更新モード：ユーザアプリケーションのみ
- BLE Protocol Stack 更新モード：ユーザアプリケーションと Bluetooth LE 通信部の両方

3.3.1 アプリケーション更新モード

アプリケーション更新モードの動作シーケンスを示します。本モードでは、アプリケーションセクションと可変ベクタテーブルセクションを更新します。ダウンローダは、更新ファームウェアのアプリケーションセクション部分をアプリケーションセクションに書き込み、可変ベクタテーブルを C$VECT_BACK_UP セクションに書き込みます。リロケータは C$VECT_BACK_UP セクションから C$VECT セクションに再配置します。その後リロケータはアプリケーションプログラムを起動します。

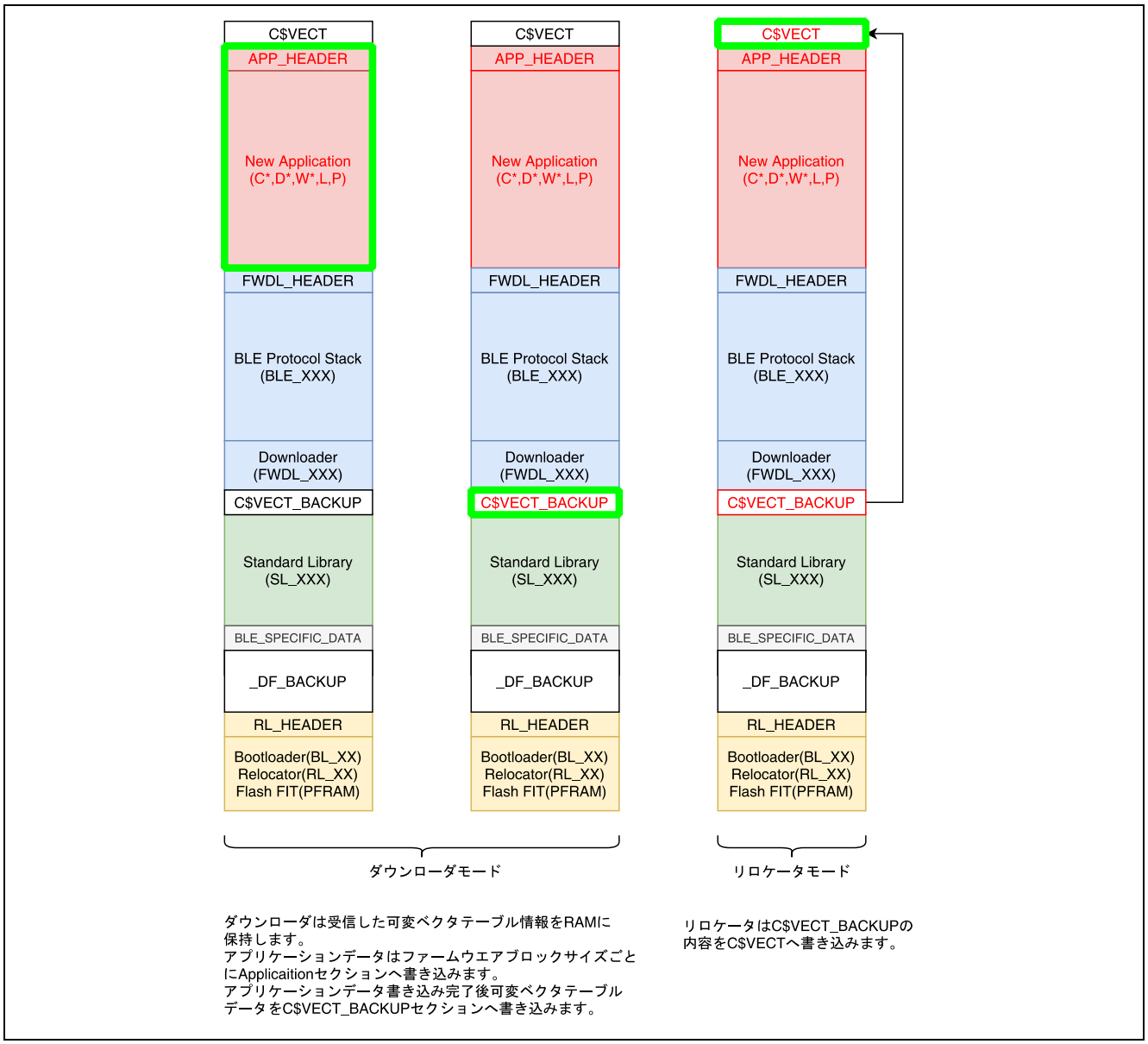

図 3.7 OTA ファームウェア更新の ROM セクション操作（ユーザアプリケーションのみ更新）

3.3.2 BLE Protocol Stack 更新モード

BLE Protocol Stack 更新モードの動作シーケンスを示します。ダウンローダは、Application セクションに BLE プロトコルスタックとダウンローダを、C$VECT_BACKUP セクションに可変ベクタテーブルを書き込み、リロケータがそれぞれ、Downloader セクション、C$VECT セクションに再配置します。その後、更新されたダウンローダを起動し、アプリケーション更新モードを実行します。

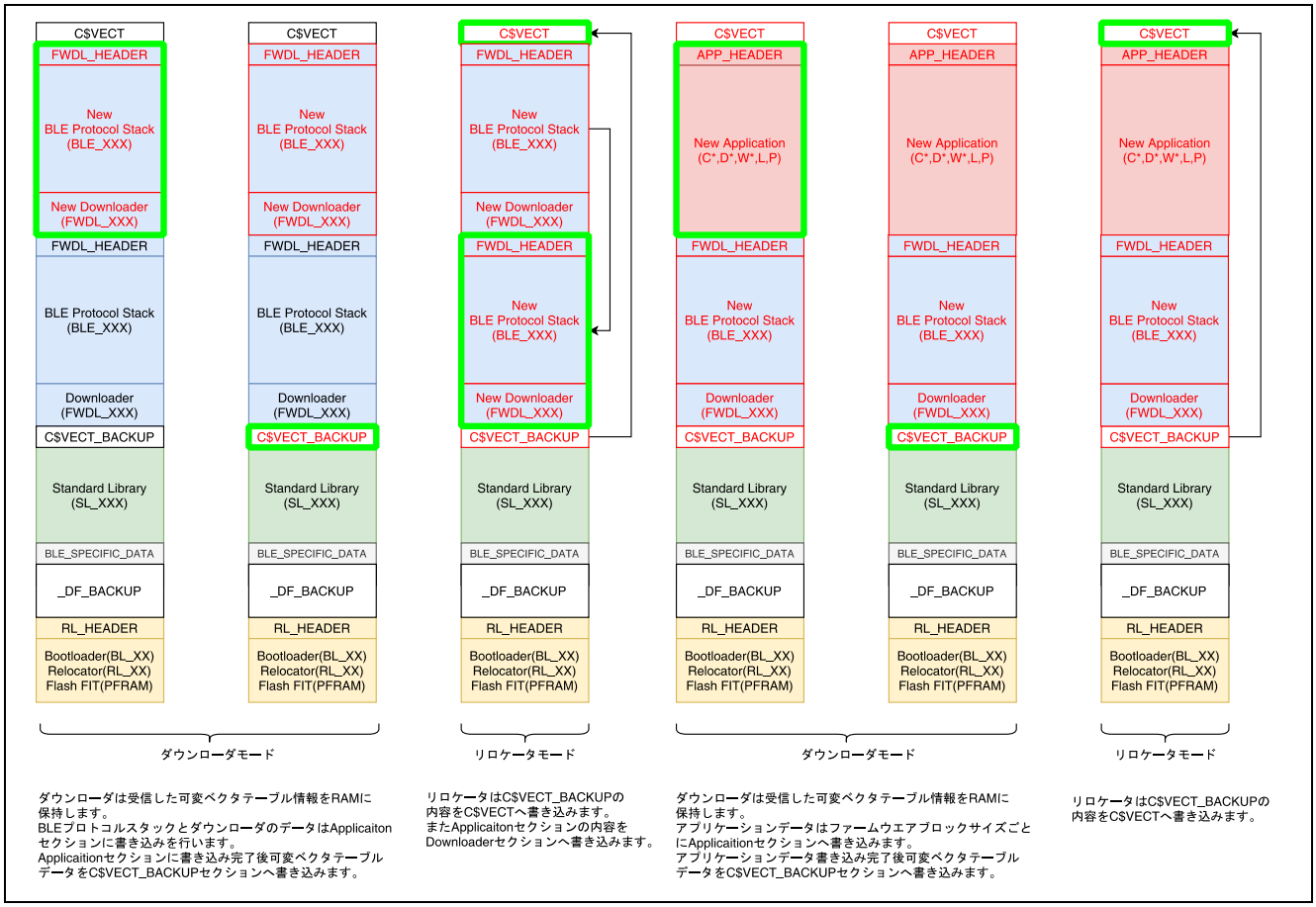

図 3.8　OTA ファームウェア更新の ROM セクション操作（BLE Protocol Stack 更新モード）

3.4 プロファイル仕様

OTA ファームウェア更新では、以下の二つの GATT サービスを利用して通信を行います。更新時間短縮のため、MTU サイズは 247 を前提としています。

- Renesas OTA Reset Service
- Renesas OTA Service

Renesas OTA Reset Service は、ユーザアプリケーションに組み込んで使用します。ファームウェアのバージョン情報やプロジェクト情報を OTA Client に提供します。OTA によるダウンローダへの切り替え機能を提供します。

Renesas OTA Service は、ダウンローダで使用されます。ファームウェア情報の展開と、更新用ファームウェアのダウンロード方法を定義します。

3.4.1 Renesas OTA Reset Service

表 3-2 サービスの概要

サービス	タイプ	UUID
Renesas OTA Reset Service	Primary Service	feacbb7a-db63-4ca1-8ae2-f611c8269f65

キャラクタリスティック	プロパティ	パーミッション	説明	UUID
Virtual Reset Button Characteristic	Write Only	Encryption	18byte, Reset Code (18-byte strings)	7753002f-34e9-4a65-904e-5636fbfcec23
Project Information Characteristic	Read Only	Encryption	variable length Project Name (20byte strings)	1b7eea33-19bf-44e0-ba37-3fa621bbb0d7
Version Information Characteristic	Read Only	Encryption	Version information (8byte)	aaa509c9-7a50-458c-99dd-dc33e2c2d90c

3.4.2 Renesas OTA Service

表 3-3　サービスの概要

サービス名	タイプ	UUID
Renesas OTA Service	Primary Service	9d5998f8-105b-4691-92be-4b1b4d3ee8bb

キャラクタリスティック	プロパティ	パーミッション	説明	UUID
Data Control Characteristic	Write, Indication	Encryption	受信するファームウェアのファームウェア情報や送受信のタイミングを制御するのに使用します。 0byte コマンド 1-19 byte データ	629c8ef7-aa42-4f1e-8330-fe832961b926
Data Transfer Characteristic	Write Without Response	Encryption	ファームウェアのバイナリデータの受信に使用します。 0-MTU-3 byte ファームウェアデータ	13561280-ecb3-4691-9ab0-33649c7e03db

3.4.3 データフォーマット

Renesas OTA Reset Service 及び Renesas OTA Service で使用するエラーコードを表 3-4 に示します。

Data Control Characteristic で実行する OTA コマンドを表 3-5 に示します。整数型は全て Little Endian で指定します。

表 3-4 Renesas OTA Service 及び Renesas OTA Reset Service で使用するエラーコード

エラーコード	説明
OTA ERROR INVALID ADDRESS (0x81)	Firmware Download Start コマンド実行時に、OTA Server と Client の送受信するセクションの開始アドレスが一致しなかった場合に送信されます。
OTA ERROR INVALID LENGTH (0x82)	セクションの更新コマンド実行時に、指定したサイズがセクションを超えているか、更新開始時に指定したサイズと異なる場合に送信されます。
OTA ERROR CHECK SUM (0x83)	Firmware Download End,及び各 Update End コマンド実行時に、Client が送信したチェックサムと OTA Server の計算したチェックサムが異なる場合に送信されます。
OTA ERROR RESET CODE (0x84)	Virtual Reset Button Characteristic に書き込まれた Reset Code が OTA Server のものと一致しない場合に送信されます。
OTA ERROR PROJECT INFORMATION (0x85)	Variable Vector Update Start コマンド実行時に、Project Information Request で送信されたプロジェクト名を比較します。この時、プロジェクト名が一致しない場合に送信されます。
OTA ERROR INVALID SECTION (0x86)	各 Update Start 若しくは Update End コマンドで、セクションの開始アドレスが OTA Server のものと一致しない場合に送信されます。
OTA ERROR INVALID VERSION (0x87)	Application Update Start 及び BLE Downloader Start コマンド実行時に、バージョン情報の確認を行います。Version Information Request で指定されたバージョン情報が更新可能でない場合に送信されます。
OTA ERROR INVALID CMD (0x90)	OTA Server が、実行されたコマンドを受け付けない状態にあるときに送信されます。

表 3-5 Data Control キャラクタリスティックのコマンド

コマンド名	データ構造	役割
VariableVectorTableUpdateStart (0x01)	0-3byte 可変ベクタテーブルセクションの開始アドレス 4-7byte 可変ベクタテーブルセクションのサイズ	可変ベクタテーブルのファームウェアデータの送信を開始します。
VariableVectorTableUpdateEnd (0x02)	0-3byte 可変ベクタテーブルセクションの開始アドレス 4-7byte 可変ベクタテーブルセクションのサイズ 8-11byte 更新領域に対するチェックサム	可変ベクタテーブルのファームウェアデータの送信が終了した事を伝えます。ファームウェアの妥当性確認のため、送信したファームウェアデータのチェックサムを送信します。
ApplicationUpdateStart (0x03)	0-3byte アプリケーションセクションの開始アドレス 4-7byte アプリケーションセクションのサイズ	アプリケーションセクションのセクション情報の確認とファームウェアデータの送信を開始します。ファームウェアデータは、Firmware Download Start コマンド後に送信されます。
ApplicationUpdateEnd (0x04)	0-3byte アプリケーションセクションの開始アドレス 4-7byte アプリケーションセクションのサイズ 8-11byte 更新領域に対するチェックサム	アプリケーションセクションのファームウェアデータの送信が終了した事を伝えます。ファームウェアの妥当性確認のため、送信したファームウェアデータのチェックサムを送信します。
BLEDownloaderUpdateStart (0x05)	0-3byte ダウンローダセクションの開始アドレス 4-7byte ダウンローダセクションのサイズ	ダウンローダセクションのセクション情報の確認とファームウェアデータの送信を開始します。ファームウェアデータは、Firmware Download Start コマンド後に送信されます。
BLEDownloaderUpdateEnd (0x06)	0-3byte ダウンローダセクションの開始アドレス 4-7byte ダウンローダセクションのサイズ 8-11byte 更新領域に対するチェックサム	ダウンローダセクションのファームウェアデータの送信が終了した事を伝えます。ファームウェアの妥当性確認のため、送信したファームウェアデータのチェックサムを送信します。
FirmwareDownloadStart (0x07)	0-3byte ダウンロード対象セクションの開始アドレス 4-7byte ダウンロード対象セクションのサイズ	各 Update Start コマンドで指定したセクションのファームウェアデータを、本コマンドして指定したサイズ(ファームウェアブロック)毎に区切って送信を行います。
FirmwareDownloadEnd (0x08)	0-3byte ダウンロード対象セクションの開始アドレス 4-7byte ダウンロード対象セクションのサイズ 8-11byte 更新領域に対するチェックサム	Firmware Download Start コマンドで指定したファームウェアデータの送信が完了したことを通知します。
ProjectInformationRequest (0x09)	1 - 18byte 更新するプロジェクト名	Client が更新するファームウェアのプロジェクト名を送信します。OTA Server は、本コマンド受信後に Project Information Response コマンドを送信します。
ProjectInformationResponse (0x0A)	1 - 18byte OTA Server のプロジェクト名	OTA Server に書き込まれているファームウェアのプロジェクト名を送信します。
VersionInformationRequest (0x0B)	0-1byte アプリケーションセクションのバージョン 2-3byte ダウンローダセクションのバージョン 4-5byte リロケータセクションのバージョン 6-7byte BLE ProtocolStack のバージョン	Client が更新するファームウェアのバージョン情報を送信します。OTA Server は、本コマンド受信後に Project Information Response コマンドを送信します。
VersionInformationResponse (0x0C)	0-1byte アプリケーションセクションのバージョン 2-3byte ダウンローダセクションのバージョン 4-5byte リロケータセクションのバージョン 6-7byte BLE ProtocolStack のバージョン	OTA Server に書き込まれているファームウェアのバージョン情報を送信します。

3.5 OTA の通信シーケンス

OTA Server は以下のシーケンスに従ってファームウェアを受信し更新を行います。OTA Client 及びスマートフォンアプリ(FWU_Client)はこのシーケンスを実行します。

- ユーザアプリケーション
  ユーザアプリケーションでは、バージョン情報とプロジェクト名の確認とダウンローダプログラムへの切り替えを行います。以下に通信シーケンスを示します。

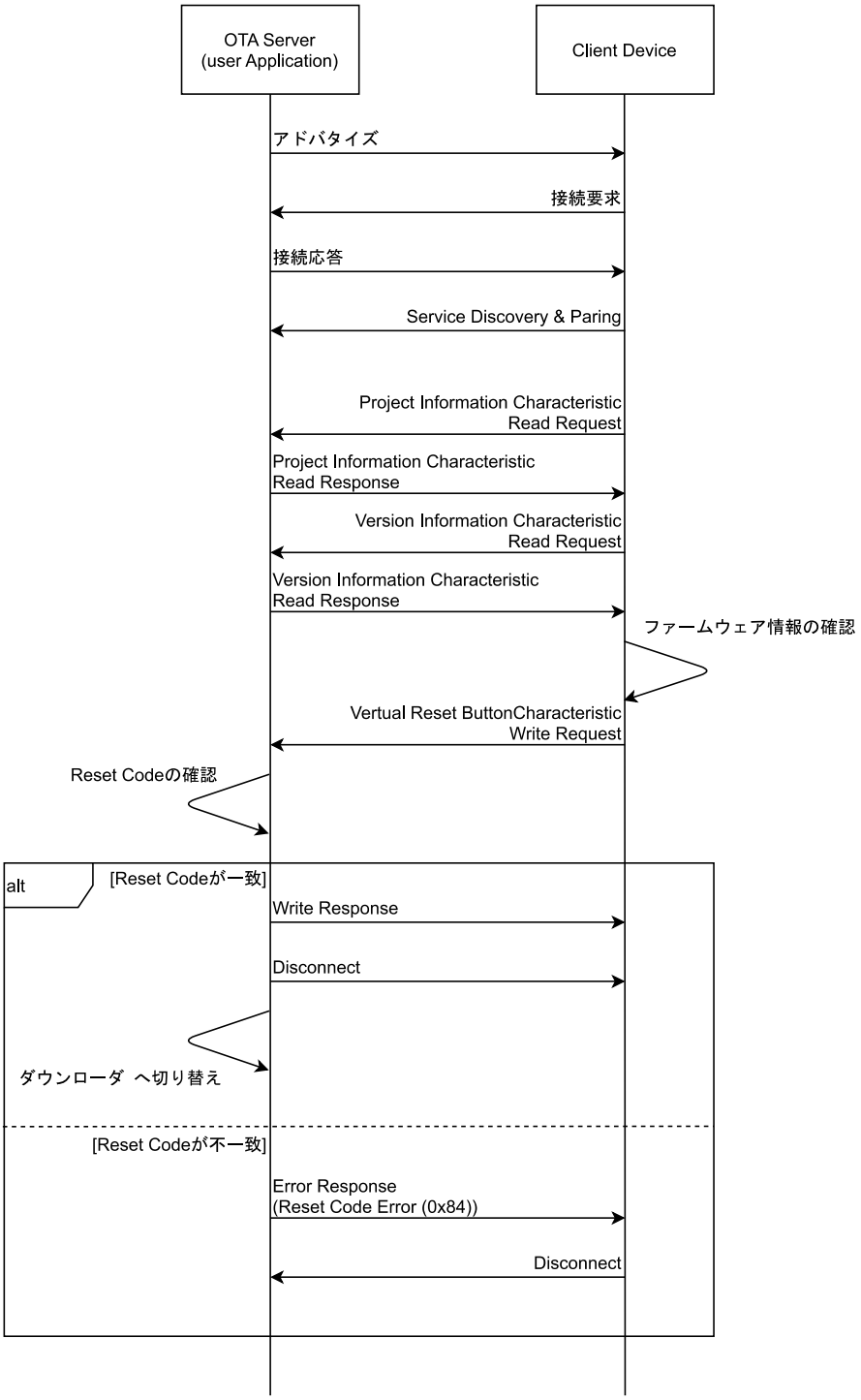

図 3.9 ユーザアプリケーションでの OTA Reset Service を利用したプログラム切り替えシーケンス

- ダウンローダ
  ダウンローダ実行中は、アプリケーションセクションの更新若しくはダウンローダセクションの更新のためのファームウェアを受信します。どちらの更新についても、可変ベクタテーブルを先にダウンロードし、その後に指定されたセクションの受信を行います。以下に通信シーケンスを示します。

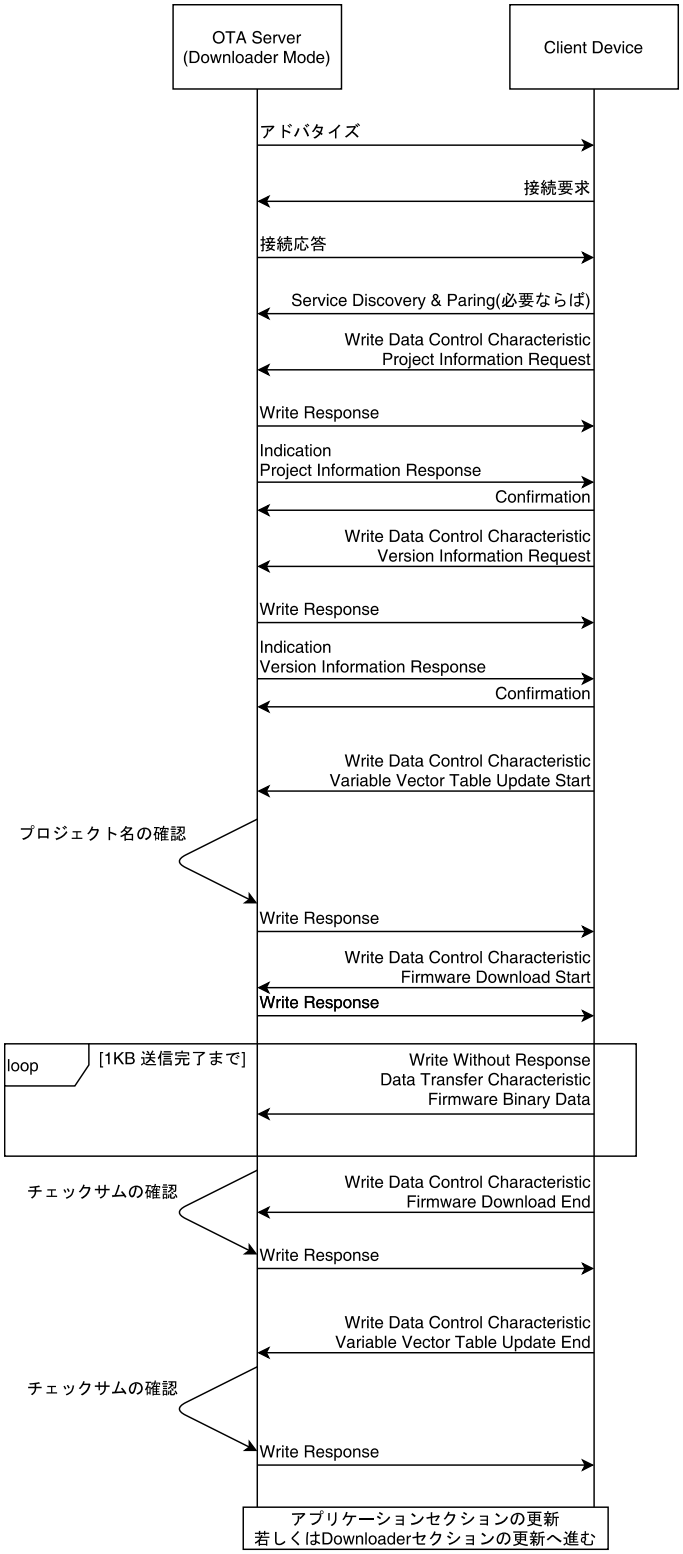

図 3.10　可変ベクタテーブルの受信

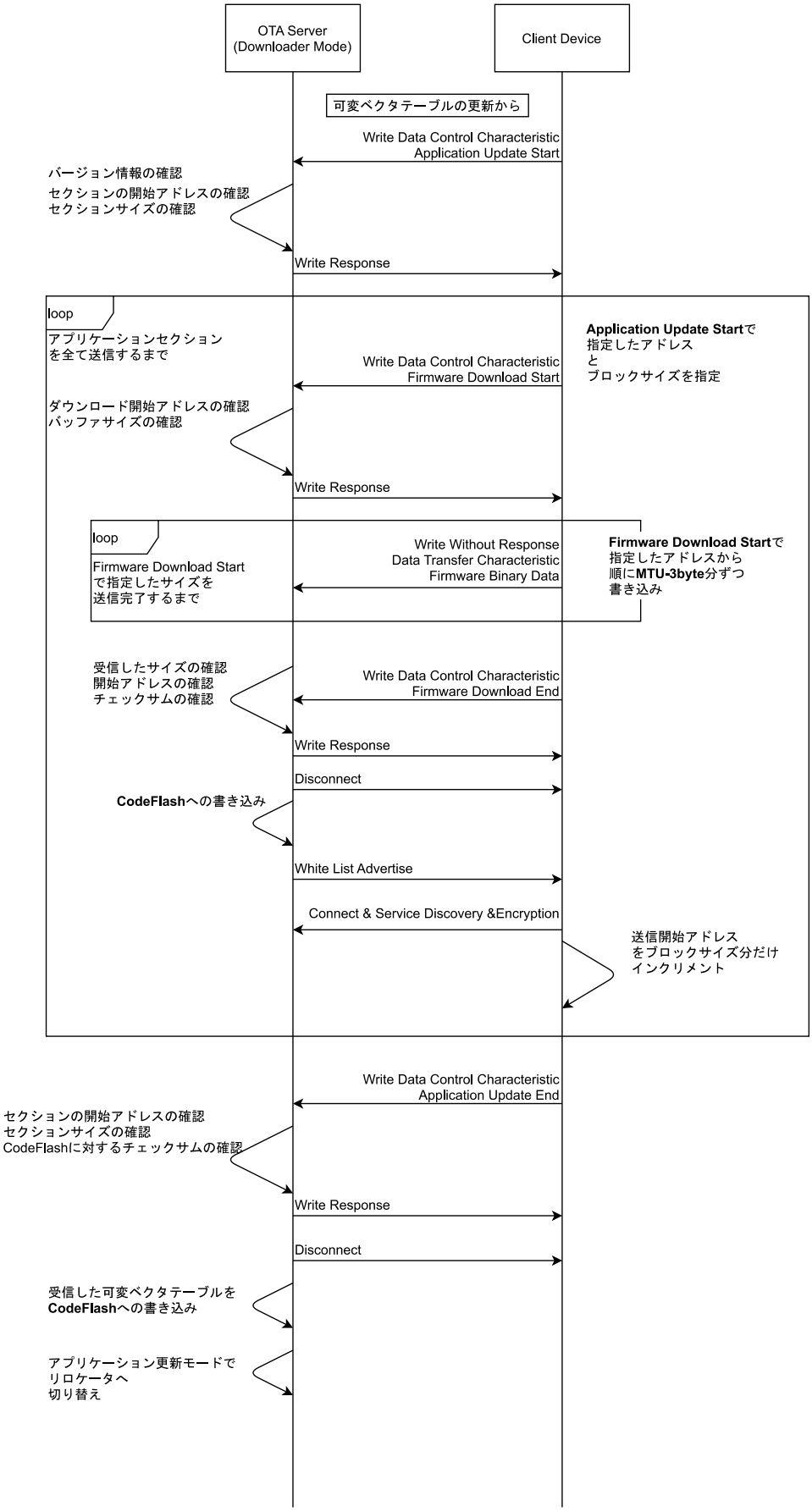

図 3.11　アプリケーションセクションの受信

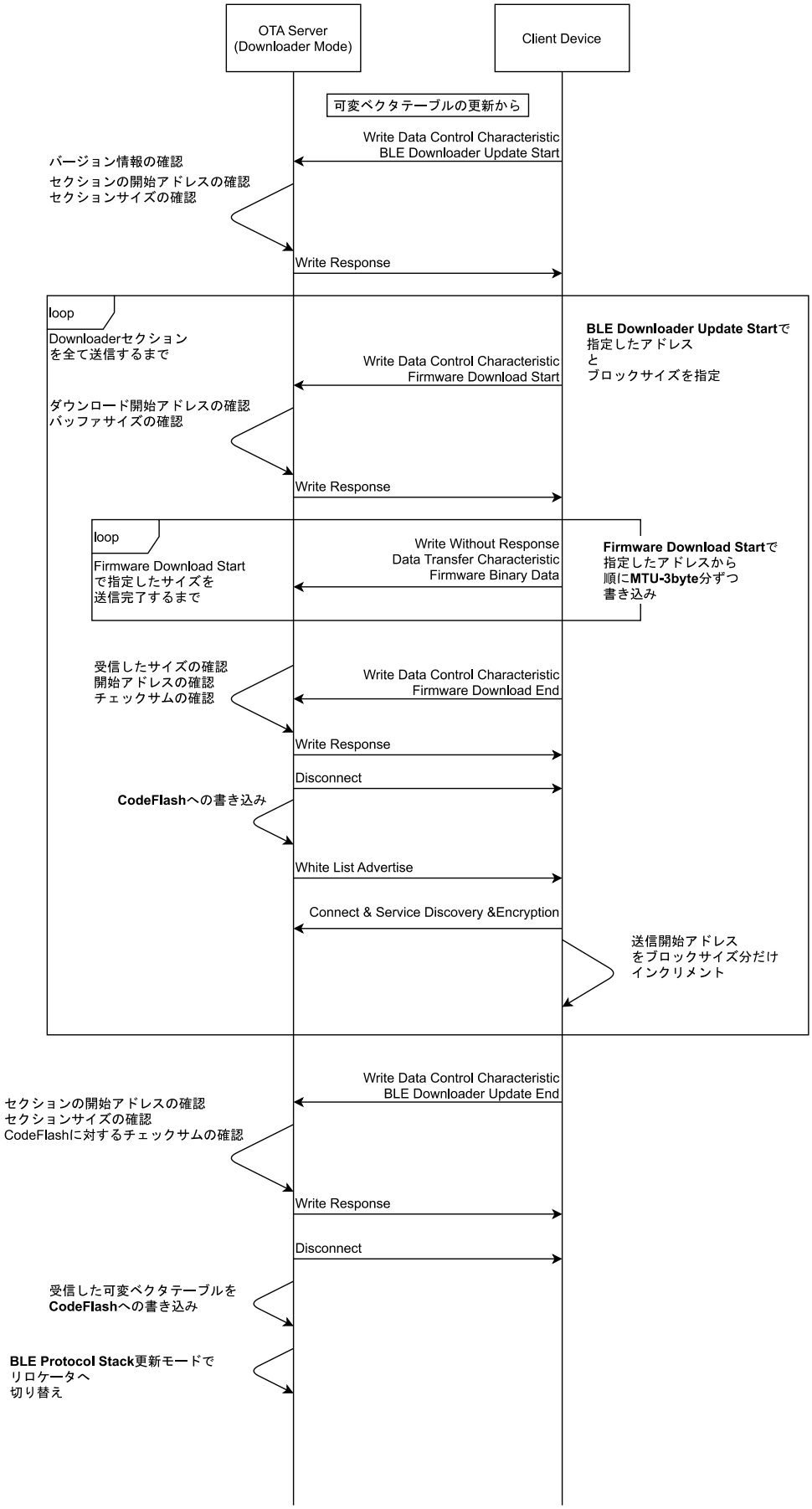

図 3.12　ダウンローダセクションの受信

Data Control Characteristic に不適な値が書き込まれた場合、OTA Server は Error Response を返します。Client Device が、Firmware Download End コマンドに対する Check sum Error を受け取った場合、もう一度 Firmware Download Start コマンドからやり直すことができます。それ以外の Error Response を受け取った場合、Client Device は OTA Server と切断し更新を終了します。

3.6 起動プログラムの切り替え

起動プログラムを切り替えは、Data Flash と Code Flash の二か所の領域に実行プログラムのフラグ情報を書き込み、MCU リセットを行います。独立した二か所の領域にフラグ情報を書き込む事で、電源遮断による動作停止から保護します。以下に起動プログラムの切り替えシーケンスを示します。

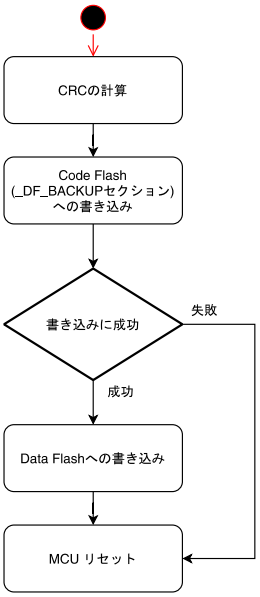

図 3.13 起動プログラムの切り替え動作

実行プログラムのフラグ情報は以下の通りです。

表 3-6 起動プログラムのフラグ情報

パラメータ	型	説明
crc	uint16_t	activate_mode から、unique_code の末尾までの領域に対して CRC 計算を行った結果を保持します。 フラグ情報の妥当性確認に使用されます。
activate_mode	e_ble_ota_activate_mode_t	起動プログラムを指定します。
status_info source code	st_ble_ota_dataflash_status_t - e_ble_ota_activate_mode_t - e_ble_ota_status_t	起動プログラムに渡す情報パラメータです。 source:実行されていたプログラム情報 code:起動プログラムに渡す情報
unique_code[12]	uint8_t	CRC 計算時にデータに冗長性を持たせるために定義されます。

3.7 ブートローダ

ブートローダは、Data Flash に保持している実行プログラムのフラグ情報をもとに、OTA Server の起動するプログラムを選択します。ブートローダは以下の二つの機能を実現します。

実行プログラムのフラグ情報の読み出しと妥当性確認

フラグ情報に基づいたプログラムの実行

以下にブートローダのフローチャートを示します。

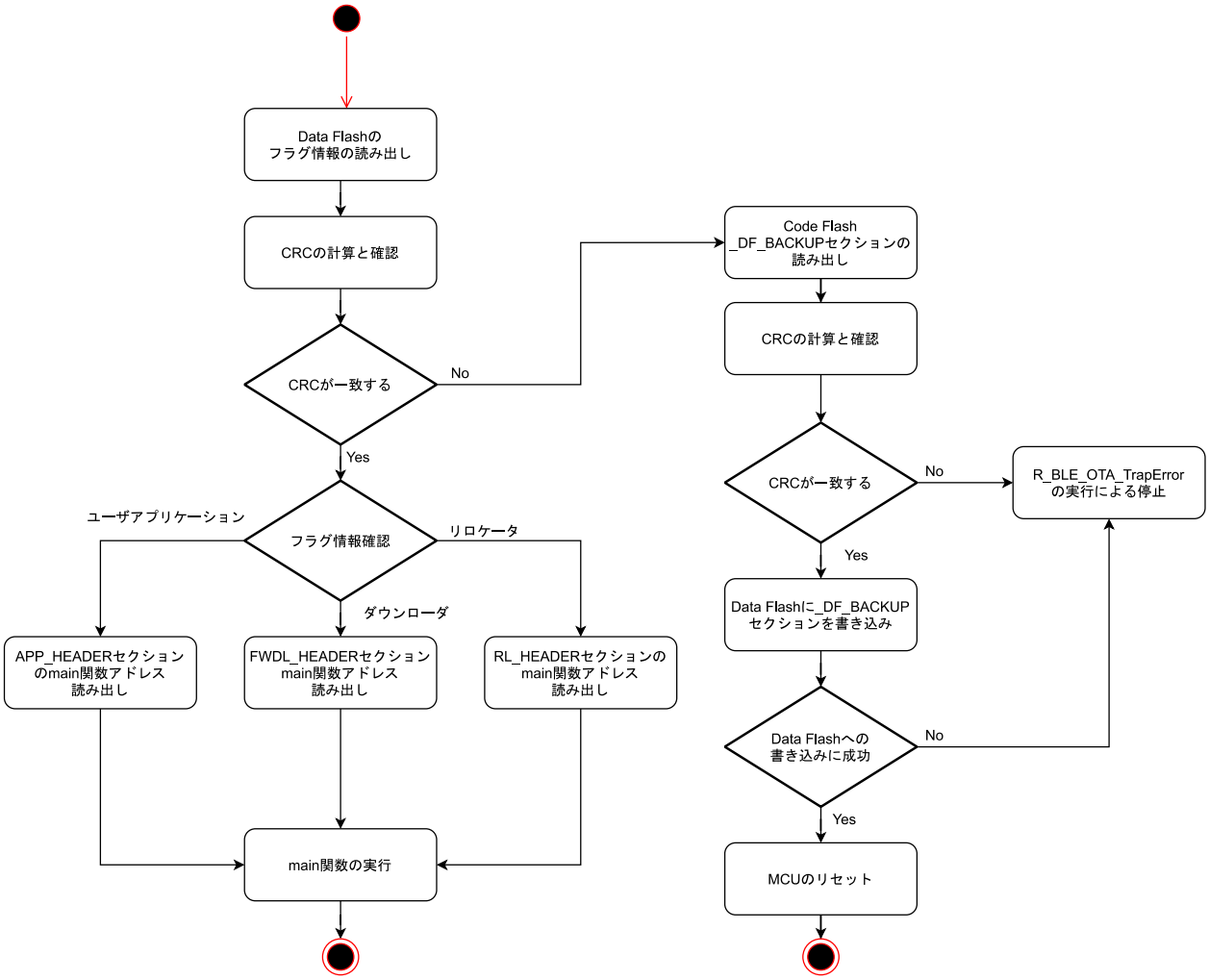

図 3.14 ブートローダの動作フローチャート

実行プログラムのフラグ情報は、書き込み中の電源遮断によって消えてしまわないように、Data Flash と Code Flash の_DF_BACKUP セクションの二か所に書き込まれます。

Data Flash のフラグ情報を読み出し CRC を計算しフラグ情報と一致するか確認します。CRC が一致しない場合には、_DF_BACKUP セクションを読み出し CRC による妥当性確認を行います。_DF_BACKUP セクションの CRC が一致する場合には、_DF_BACKUP セクションのフラグ情報を Data Flash に書き写し Reset します。_DF_BACKUP セクションの CRC が一致しない場合には、実行不可能となり、R_BLE_OTA_TrapError 関数を呼び出し、プログラムを停止します。

Data Flash のフラグ情報の妥当性確認後、起動情報を読み出し、各セクションの HEADER セクションに記載されている各プログラムの main 関数を実行します。ユーザアプリケーションは、r_ble_ota_app_loader.c ファイル内の application_loader 関数を経由して呼び出されます。

3.8 リロケータ

3.8.1 プログラムの動作

リロケータは、ダウンローダが受信したファームウェアをファームウェアー時保持セクションから正規のセクションにプログラムを再配置します。ダウンローダがファームウェアー時保持セクションに書き込んだファームウェアの種類は、Data Flash に書き込まれたフラグ情報に含まれます。

以下にリロケータの動作フローチャートを示します。

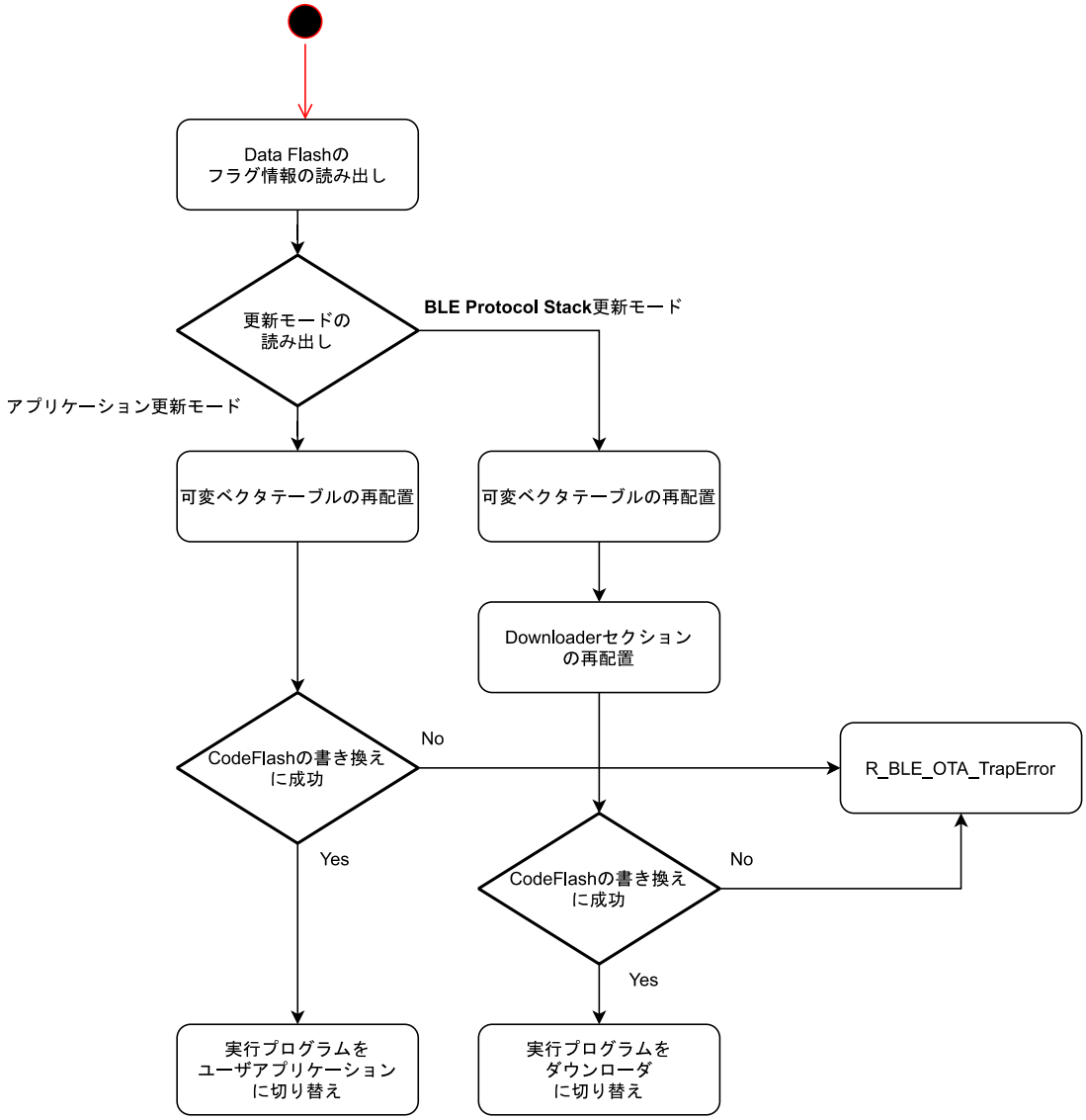

図 3.15 リロケータの動作フローチャート

アプリケーションセクションの更新の場合には、ファームウェアー時保持セクションとアプリケーションセクションが一致するため、可変ベクタテーブルの再配置のみを行います。ダウンローダセクションの更新の場合には、可変ベクタテーブル及び、ファームウェアー時保持セクションをダウンローダセクションへ再配置します。

ファームウェアの再配置後、アプリケーションの更新の場合には、次に実行するプログラムをユーザアプリケーションに、ダウンローダセクションの更新の場合には、次に実行するプログラムをダウンローダに設定します。

3.8.2　電源遮断時の動作

リロケータは、可変ベクタテーブル及び、ダウンローダセクションのプログラムの再配置を行います。リロケータは、書き換え前のセクションを変更しません。そのため、再配置中に電源遮断が発生した場合にも、図 3.15 のシーケンスが最初から実行されます。

3.9　ダウンローダ

ダウンローダでは、OTA Server は Client から更新用ファームウェアを受け取ります。ダウンローダは、”3.3 ファームウェア更新動作”と”3.5 OTA の通信シーケンス”を参照してください。

ユーザアプリケーションからダウンローダへ切り替わると、更新が完了するまでユーザアプリケーションは実行されません。

3.9.1　Bluetooth LE の動作

ダウンローダは GAP の Peripheral として動作し、Renesas OTA Service の UUID を含む Advertise を 30msec 間隔で行います。ダウンローダは、

ダウンローダの Advertise の設定は r_ble_ota_ble_interface.c ファイルに実装されています。

ダウンローダ動作中は、以下のタイミングで MCU リセットを行います。

- Advertise の開始から 10 秒経過
- Central からの接続後、Data Control Characteristic への書き込みが 10 秒間ない場合
- Client から切断された場合

3.9.2　ボンディング情報管理

ダウンローダは、ペアリング情報を不揮発領域に保持しません。app_lib のセキュリティライブラリが書き込む不揮発領域にあるボンディング情報を利用します。

ダウンローダは、不揮発領域にあるペアリング情報をもとに Resolving　List に登録しますが、ペアリング時には、対向デバイスの IRK とアドレスを Resolving　List に登録しません。RPA デバイスを使用する場合には、ユーザアプリケーションの実行中にボンディングするようにしてください。

3.9.3　Code Flash への書き込み

ダウンローダは、ファームウェアブロックサイズ毎に受け取ったファームウェアを Code Flash に書き込みます。可変ベクタテーブルは可変ベクタテーブルバックアップセクションに、アプリケーションセクション及びダウンローダセクションのファームウェアはファームウェア一時保持セクションに書き込まれます。

Code Flash の書き込み中は、Code Flash の読み出しができないため、Bluetooth LE 接続を切断し、書き込みを行います。

3.9.4　電源遮断時の動作

ダウンローダ実行中に電源遮断が発生した場合、OTA Server はもう一度ダウンローダを実行します。ファームウェアのダウンロード途中であっても、図 3.10 の最初から更新シーケンスを再開します。

3.10 ユーザアプリケーション

ユーザアプリケーションは、OTA Server が実現するアプリケーション部分です。OTA による更新後に起動されます。

3.10.1 ダウンローダへの切り替え動作

ユーザアプリケーションから OTA によるファームウェアアップデートを開始するには、実行するプログラムをダウンローダに切り替えます。

OTA によるダウンローダへの切り替えは、Renesas OTA Reset Service を利用します。ユーザアプリケーションからダウンローダへの切り替えシーケンスを以下に示します。

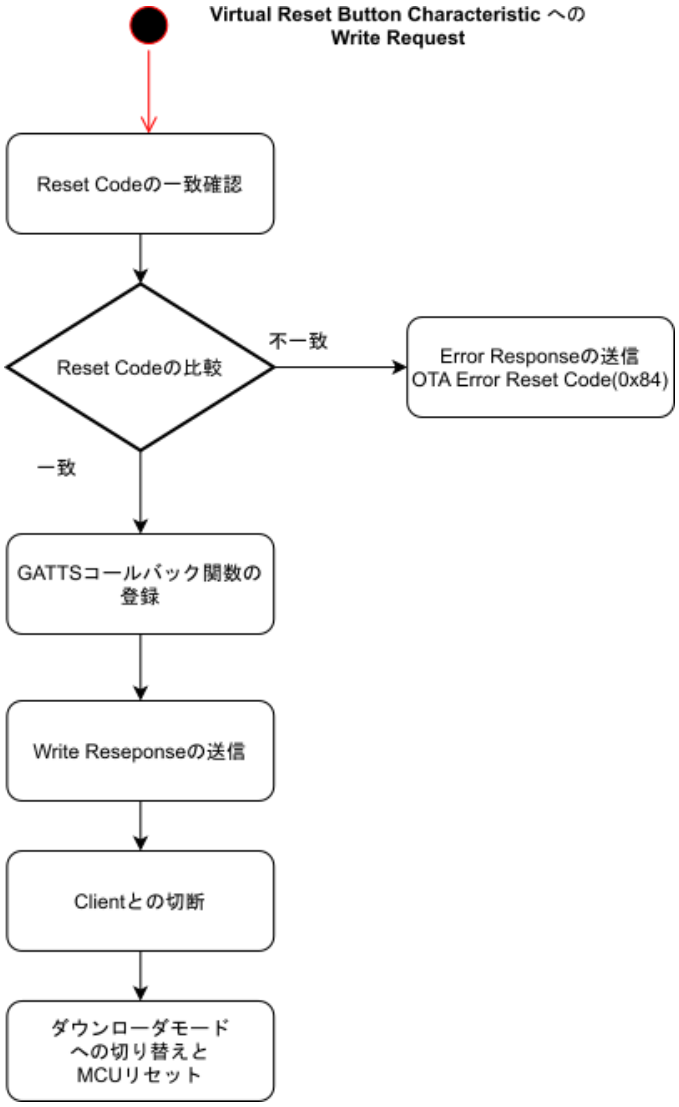

図 3.16 OTA Reset Service による実行プログラムの切り替え

3.10.2 サンプルプログラムのユーザアプリケーション

本パッケージに同梱の OTA Server はユーザアプリケーション部分に、”RX23W グループ Bluetooth Low Energy アプリケーション開発者ガイド(R01AN5504)”に同梱の Peripheral サンプルを実装しています。

4. OTA 更新対応プロジェクトの作成

本章は、既存のプロジェクトにファームウェア更新機能を追加する方法を示します。なお、本サンプルでは、BLE Protocol Stack の”Balance Library”のみをサポートします。既存プロジェクトには、BLE FIT モジュールが追加されていることとします。

4.1 ソースコード(r_ble_ota)の追加

本サンプルの OTA Server プロジェクトである ble_sample_tbrx23w_ota_server から r_ble_ota フォルダを既存のプロジェクトへコピーします。

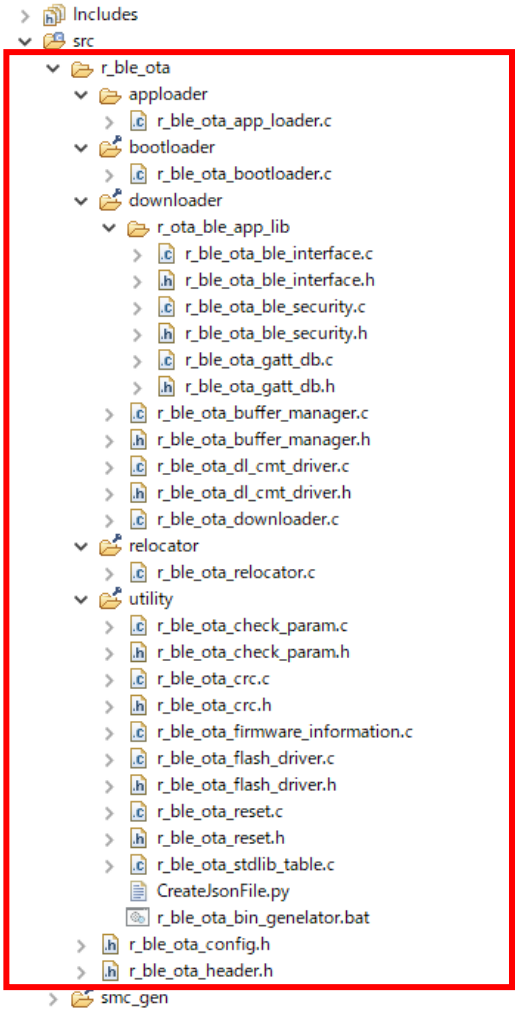

図 4.1 r_ble_ota フォルダ

コピーしたフォルダのパスを既存プロジェクトに登録します。プロジェクトを右クリックして、"C/C++ ビルド”-“設定"-“ツール設定”-“Compiler”-“ソース”-“インクルード・ファイルを検索するフォルダ (-include)に下記フォルダを追加してください。

"${workspace_loc:/${ProjName}/src/r_ble_ota}"

"${workspace_loc:/${ProjName}/src/r_ble_ota/downloader}"

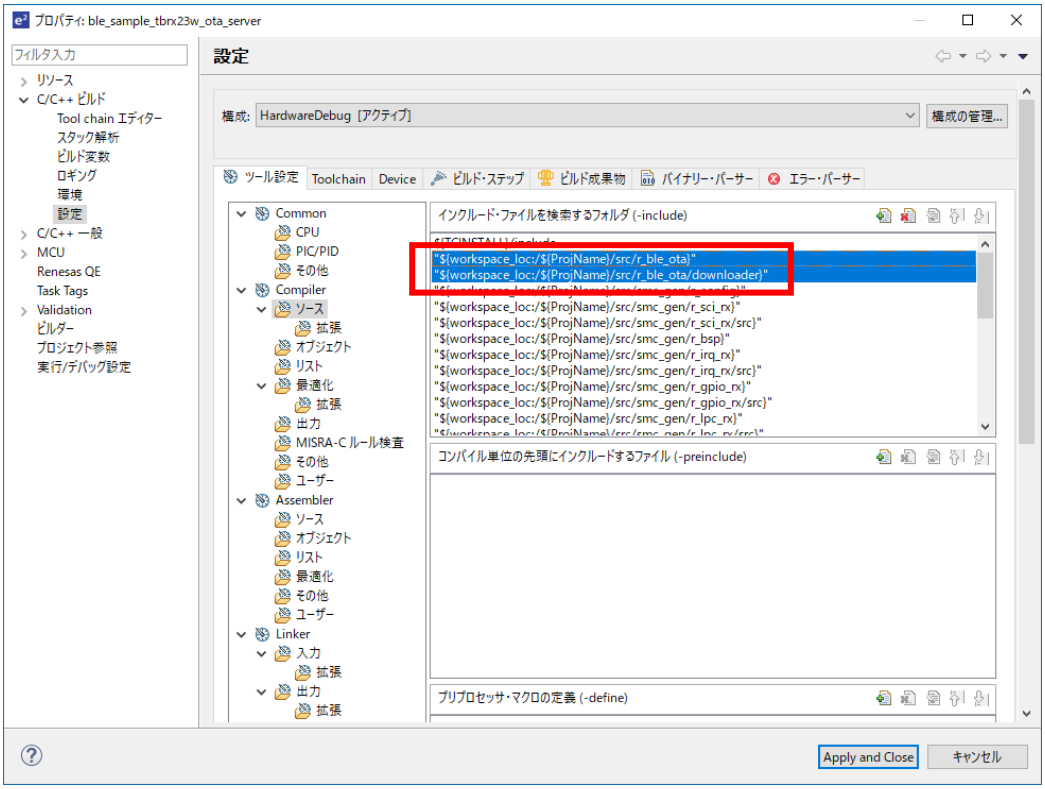

図 4.2　インクルードパスの追加

4.2　セクション分割の設定

プログラムを OTA ファームウェア更新機能に対応したセクションに分割します。e² studio で次の 2 つの項目を設定します。

1. Create section (-start)

   "C/C++ ビルド"-“設定”-“ツール設定”-“Linker”-“セクション”-“セクション(-start)”の設定を変更します。

   図 4.3 に示すようにセクション領域を分割します。本パッケージに同梱されている ProjectSetting¥section.esi ファイルをインポートすることで、OTA Server のサンプルプログラムと同様のセクションレイアウトを生成できます。

   アプリケーションセクションを拡張する場合には、FWDL_HEADER の開始アドレスを変更し、r_ble_ota_header.h の

   BLE_OTA_APPLICATION_SECTION_SIZE

   BLE_OTA_TEMPORALY_SECTION_SIZE

   BLE_OTA_DOWNLOADER_SECTION_SIZE

   を適切な値に変更します。

   ダウンローダプログラムとリロケータプログラムが使用する RAM は、ユーザアプリケーションが使用する RAM 領域にセクションオーバーレイされています。

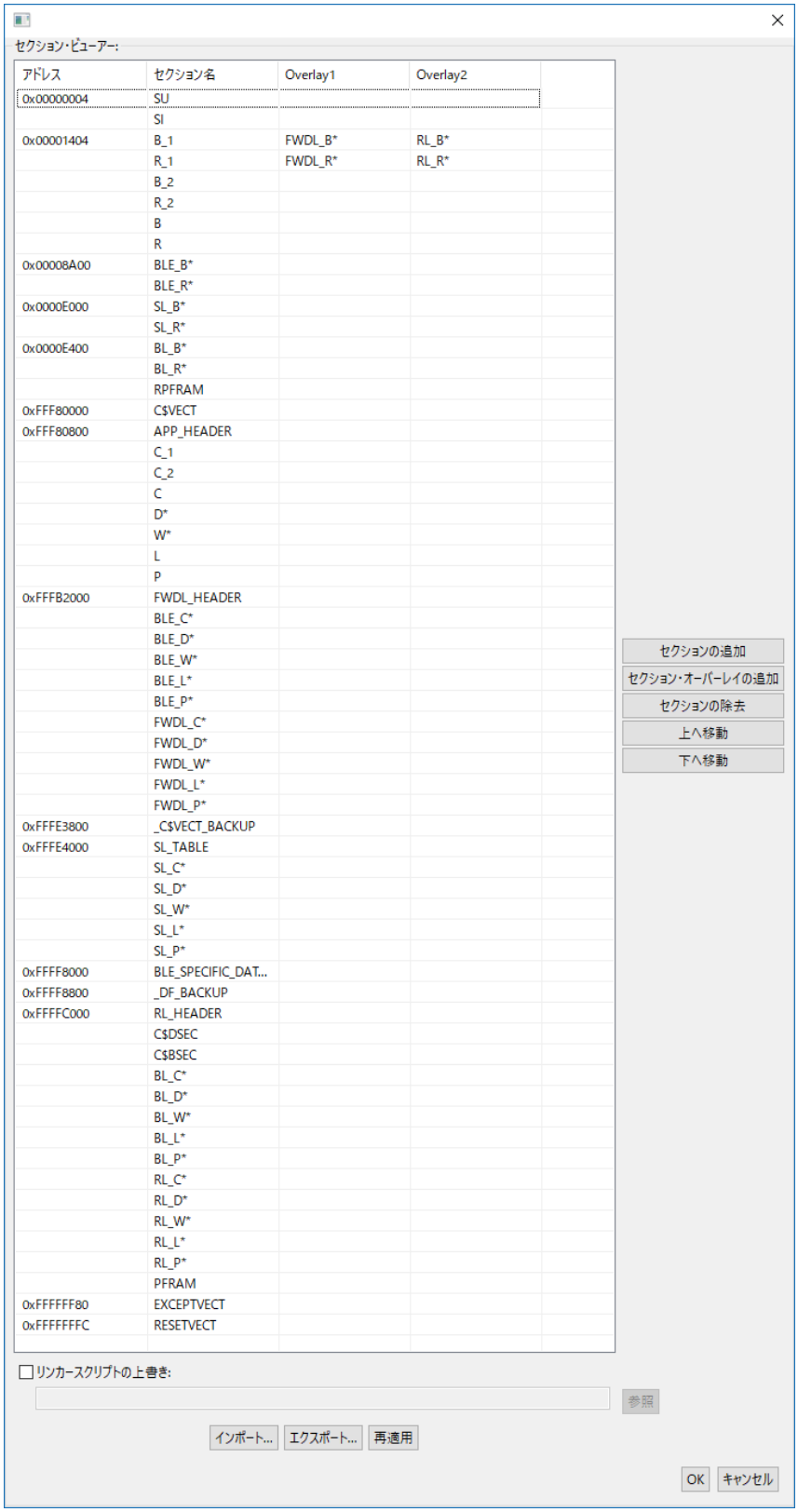

図 4.3　セクションの分割設定

```
24
25  /* OTA Configuration */
26
27  /* Application Main Function */
28  #define MAIN_FUNCTION                          app_main
29  /* Project Information */
30
31  /* BLE_OTA_PROJECT_NAME length < 18*/
32  #define BLE_OTA_PROJECT_NAME                "ota_sample"
33
34  /* BLE_OTA_PROJECT_NAME length = 18*/
35  #define BLE_OTA_RESET_CODE          "inge9ubled9hy4tljn"
36
37  /* Downloader Mode Device Name */
38  #define BLE_OTA_DOWNLOADER_DEVICE_NAME         "FWU-DEV"
39
40  /* Version Information */
41  #define BLE_OTA_APPLICATION_MAJOR_VERSION          (0x01)
42  #define BLE_OTA_APPLICATION_MINOR_VERSION          (0x0A)
43  #define BLE_OTA_DOWNLOADER_MAJOR_VERSION           (0x01)
44  #define BLE_OTA_DOWNLOADER_MINOR_VERSION           (0x0A)
45  #define BLE_OTA_RELOCATER_MAJOR_VERSION            (0x01)
46  #define BLE_OTA_RELOCATER_MINOR_VERSION            (0x00)
47
48  /* Section Information */
49  #define BLE_OTA_APPLICATION_SECTION_SIZE     (0x00031800)
50  #define BLE_OTA_TEMPORALY_SECTION_SIZE       (0x00031800)
51  #define BLE_OTA_DOWNLOADER_SECTION_SIZE      (0x00031800)
52  #define BLE_OTA_RELOCATER_SECTION_SIZE       (0x00004000)
53  #define BLE_OTA_STDLIB_SECTION_SIZE          (0x00014000)
54
55  /* Downloader Firmware Block Size */
56  #define BLE_OTA_DL_FW_BUFFER_SIZE                (0x4000)
57  #define BLE_OTA_DL_CVECT_BUFFER_SIZE               (1024)
58
59  /* Memory Configuration */
60  /* Data Flash*/
61  #define BLE_OTA_DATAFLASH_BLOCK_LENGTH             (1024)
62  #define BLE_OTA_USE_DATAFLASH_START_ADDRESS  (0x00100400)
63
```

図 4.4　r_ble_ota_config.h の変更

2.　ROM から RAM へマッピングするセクションの設定

"C/C++ ビルド"-"設定"-"ツール設定"-"Linker"-"セクション"-"シンボル・ファイル"-"ROM から RAM へマップするセクション (-rom)"の設定を変更します。図 4.5 の通り、各プログラムの ROM から RAM へのマッピングを追加します。

```
D=R
D_1=R_1
D_2=R_2
BLE_D=BLE_R
BLE_D_1=BLE_R_1
BLE_D_2=BLE_R_2
BL_D=BL_R
BL_D_1=BL_R_1
BL_D_2=BL_R_2
RL_D=RL_R
RL_D_1=RL_R_1
RL_D_2=RL_R_2
FWDL_D=FWDL_R
FWDL_D_1=FWDL_R_1
FWDL_D_2=FWDL_R_2
SL_D=SL_R
SL_D_1=SL_R_1
SL_D_2=SL_R_2
PFRAM=RPFRAM
```

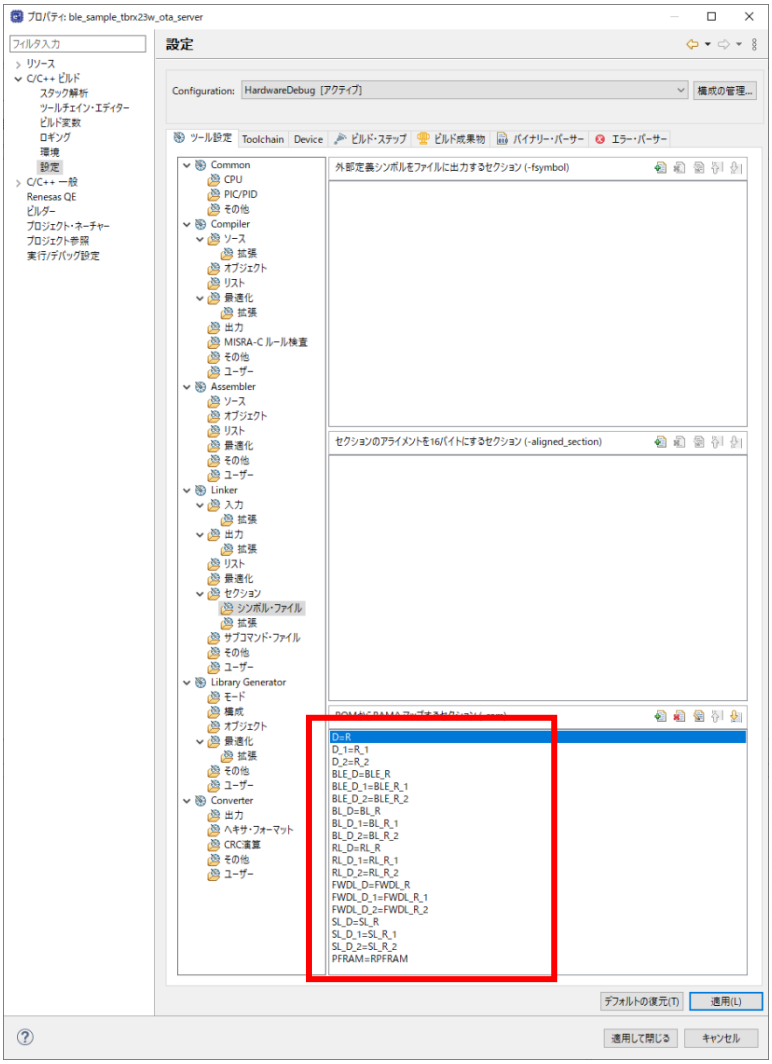

図 4.5　ROM から RAM にマップするセクション

4.3　標準ライブラリの設定

標準ライブラリは OTA による更新対象外になります。使用するライブラリは予めファームウェアに書き込みます。

“C/C++ ビルド”-“設定”-“ツール設定”-”Library Generator”-“構成”で使用するライブラリを選択します。

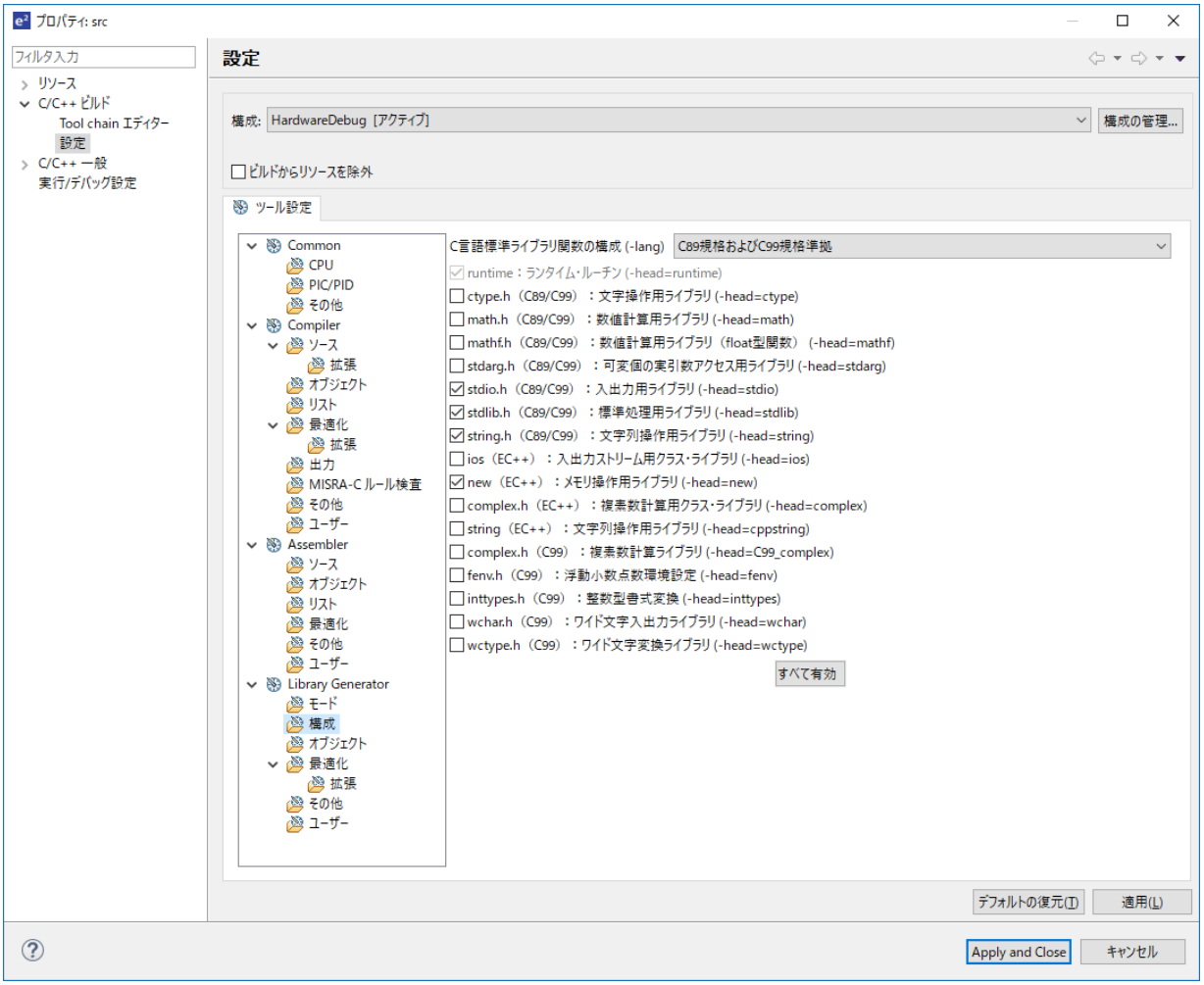

図 4.6　標準ライブラリの選択

OTA Server プログラムでは、標準ライブラリを配置するための専用のセクションを準備します。Library Generator のオブジェクト設定で、セクションを次のように設定します。

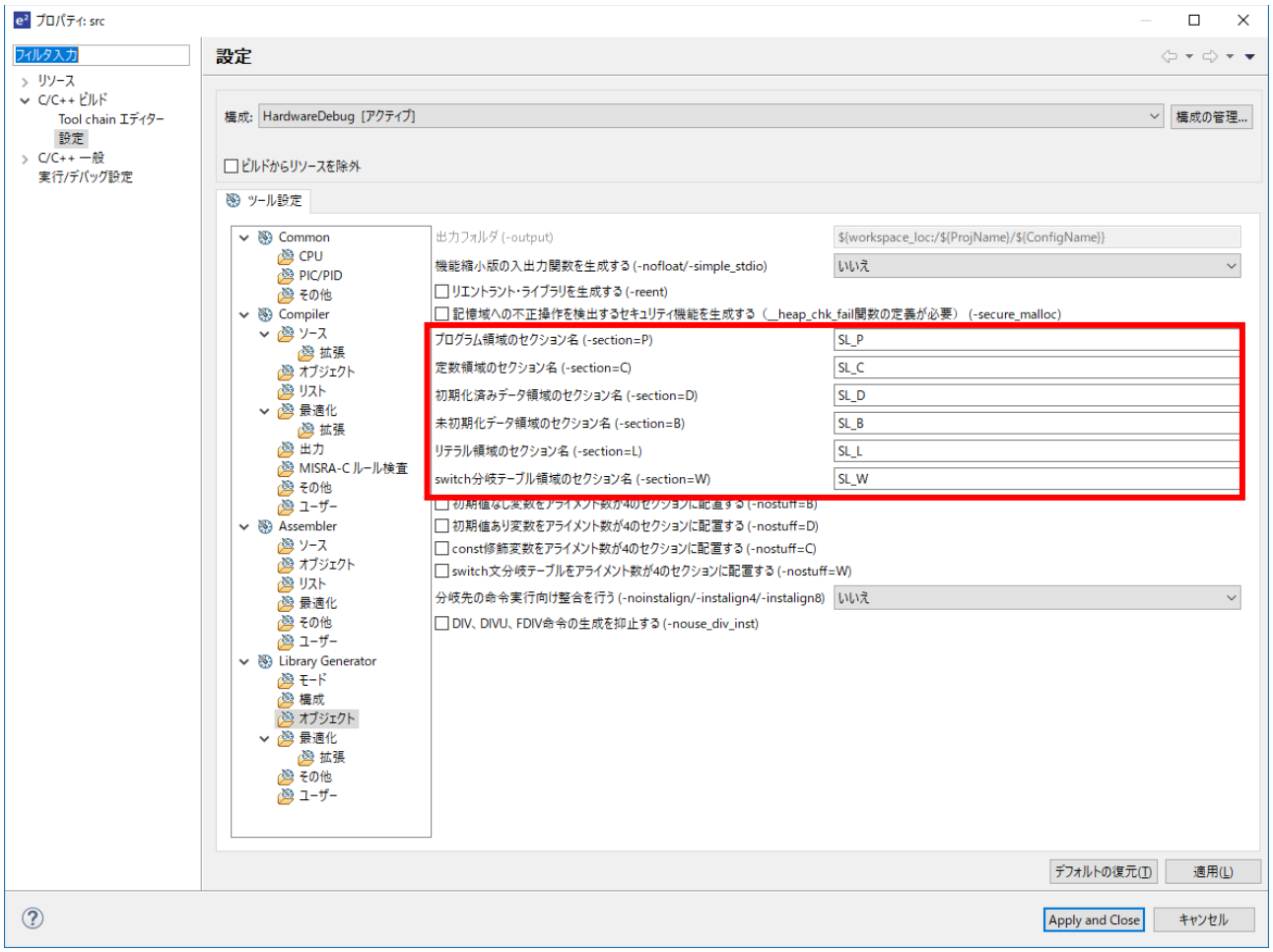

図 4.7　標準ライブラリのセクション名設定

OTA Server プログラムでは、プログラムの変更よって標準ライブラリのコードフラッシュへの配置が変更されないように、標準ライブラリで定義されるすべての関数を配列に格納しています。

r_ble_ota/utility/ r_ble_ota_stdlib_table.c で、使用する標準ライブラリに対応したマクロ定義を有効にします。

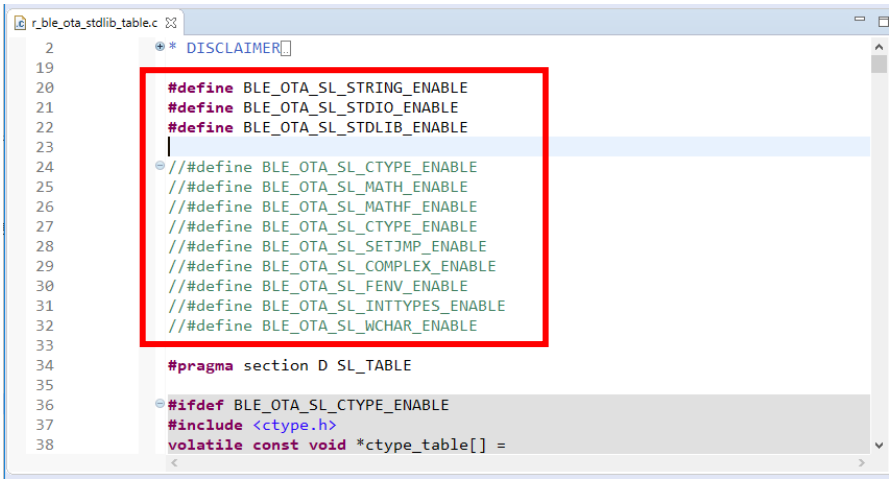

図 4.8 使用する標準ライブラリの有効化

4.4　更新用ファームウェアの出力設定

本サンプルプログラムで使用する更新用ファームウェアは、セクション情報等のファームウェア情報が記述された json ファイル及び、ファームウェアのバイナリファイル(.bin)の二つからなります。

"r_ble_ota_bin_genelator.bat"によって、CCRX コンパイラから生成されたロードモジュールファイル(.abs)ファイルからバイナリファイル(firmware.bin)を生成します。

また、"r_ble_ota_bin_genelator.bat"は、生成された実行バイナリファイル(.bin)と、リンケージリストファイル(.map)から、更新に必要なファームウェア情報を抽出し、”firmware_information.json”ファイルを作成します。バッチファイルから python が実行されますので、python の実行パスを通してください。

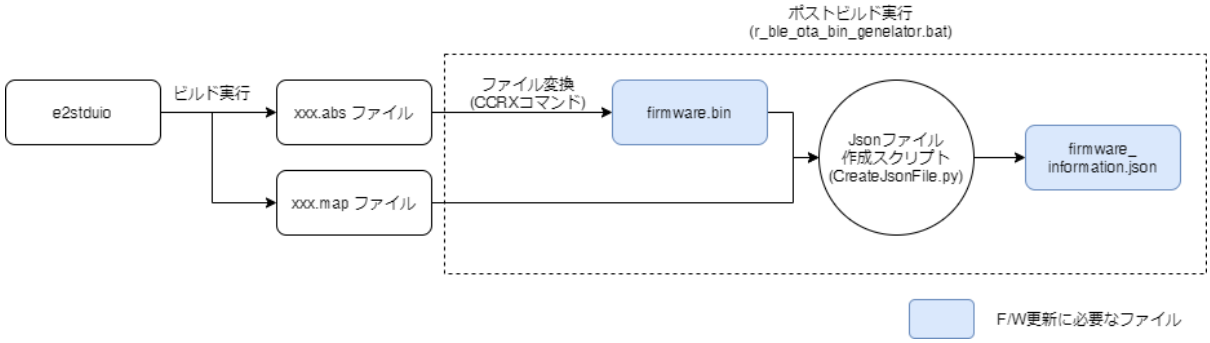

図 4.9 更新用ファームウェアの出力

ビルド後にバッチファイルを実行するように「ビルド後のステップ」設定を変更します。

..¥src¥r_ble_ota¥utility¥r_ble_ota_bin_genelator.bat

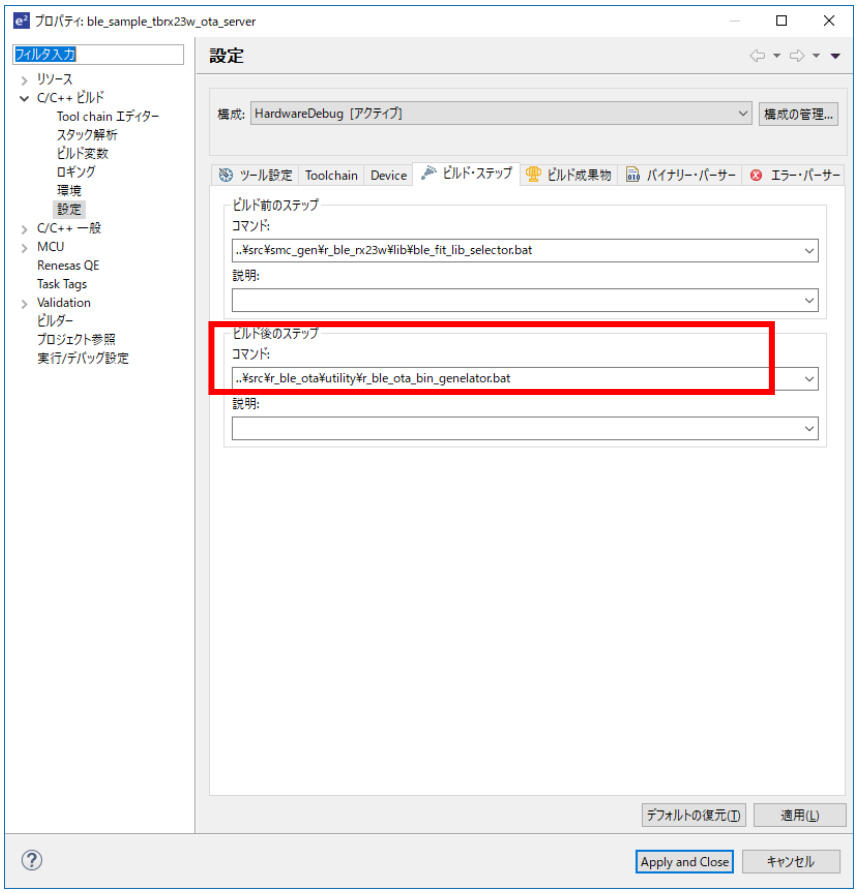

図 4.10　バッチファイルの登録

また、リンケージリストファイル(.map)に、オブジェクトのマップ情報を書き出すように、リンカーの設定を変更します。

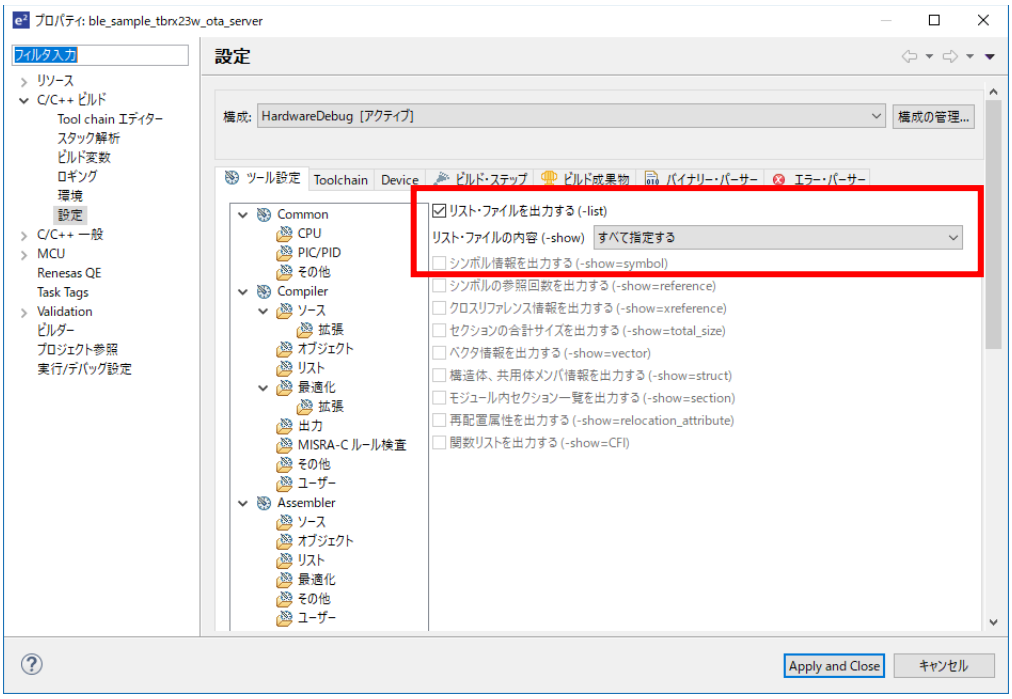

図 4.11 リンケージリストファイル(.map)生成の設定

作成された二つのファイルは、ビルド完了後にビルド構成名(例:HardwareDebug)のフォルダ内に”FWU Client”フォルダと、その中に[Projectname]_Version[Application Version]フォルダが生成されます。

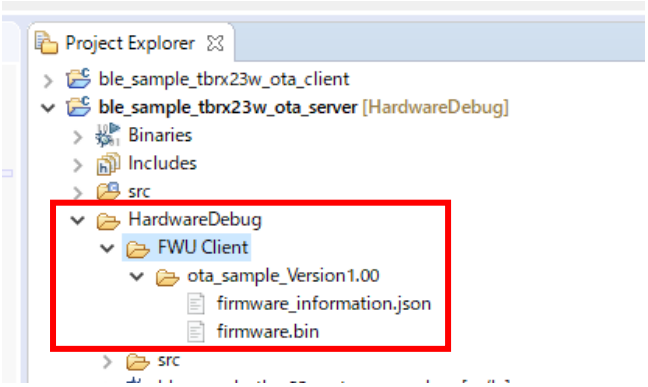

図 4.12　OTA 更新対応ファームウェアの作成

4.4.1　ファームウェア情報 JSON ファイルのプロパティ

OTA ファームウェア更新サンプルプログラムは RX23W 向けに生成されたファームウェアバイナリファイルに加え、ファームウェアのプロジェクト名、バージョン情報、セクション情報を含む JSON 形式のファームウェア更新情報ファイルを使用します。

JSON ファイルに含まれる情報を表 4-1 に示します。

OTA Server プログラムのバージョン情報とセクション情報は"r_ble_ota_firmware_information.c"に記述されています。JSON ファイルの記述がソースコードの内容と一致していることを確認してください。

表 4-1　ファームウェア更新情報ファイルの内容

プロパティ	初期値	説明
ProjectName	ota_sample	プロジェクト名です。OTA Client は、このプロパティに基づいて OTA Server を識別します。
ResetCode	inge9ubled9hy4tljn	OTA Reset Service の Virtual Reset Button に書き込む値です。
MajorApplicationVersion	0x01	アプリケーションのメジャーバージョンです。
MinorApplicationVersion	0x00	アプリケーションのマイナーバージョンです。
MajorDownloaderVersion	0x01	ダウンローダのメジャーバージョンです。
MinorDownloaderVersion	0x00,0x01	ダウンローダのマイナーバージョンです。
MajorRelocatorVersion	0x01	リロケータのメジャーバージョンです。
MinorRelocatorVersion	0x00	リロケータのマイナーバージョンです。
MajorBLEProtocolStackVersion	0x02	BLE ProtocolStack のメジャーバージョンです。
MinorBLEProtocolStackVersion	0x00	BLE ProtocolStack のマイナーバージョンです。
SectionInformation	---	各プログラムのメモリのセクション配置を記述します。 top は、各セクションの開始アドレス。 Size プロパティは、プログラムが実際に使用しているセクションサイズ max プロパティは、セクションサイズマクロで指定したサイズが設定されます。
VariableVectorTableSection	top-0xFFF80000 size-0x00000400 max-0x00000400	可変ベクタテーブルセクション配置情報です。
ApplicationSection	top-0xFFF80800 size-0x00020000 max-0x00031800	アプリケーションセクション配置情報です。
DownloderAndProtocolStackSection	top-0xFFFB2000 size-0x00026800 max-0x00031800	ダウンローダセクション配置情報です。
VariableVectorTableBackupSection	top-0xFFFE3800 size-0x00000800 max-0x00000800	可変ベクタテーブルの一時書き込みセクションの配置情報です。
StandardLibrarySection	top-0xFFFE4000 size- 0x00007800 max- 0x00014000	標準ライブラリのセクション配置情報です。
SwapTemporarySection	top-0xFFFF8000 size-0x00004000 max-0x00004000	未使用領域のセクション配置情報です。
SwapSection	top-0xFFFFC000 size-0x00004000 max-0x00004000	リロケータセクションの配置情報です。

4.5 Flash FIT モジュールの設定

コードフラッシュの書き換え中コードフラッシュへのアクセスが禁止されているため、コードフラッシュへの書き込みは RAM に命令コードを配置して実行します。

OTA Server プログラムはコードフラッシュの書き換えに Flash FIT モジュールを使用します。Flash FIT モジュールの下記設定を変更し、Flash FIT モジュールの RAM 実行を有効にします。

- Enable code flash programming: Includes code to program ROM area.

右上のコード生成ボタンを押して設定を反映します。

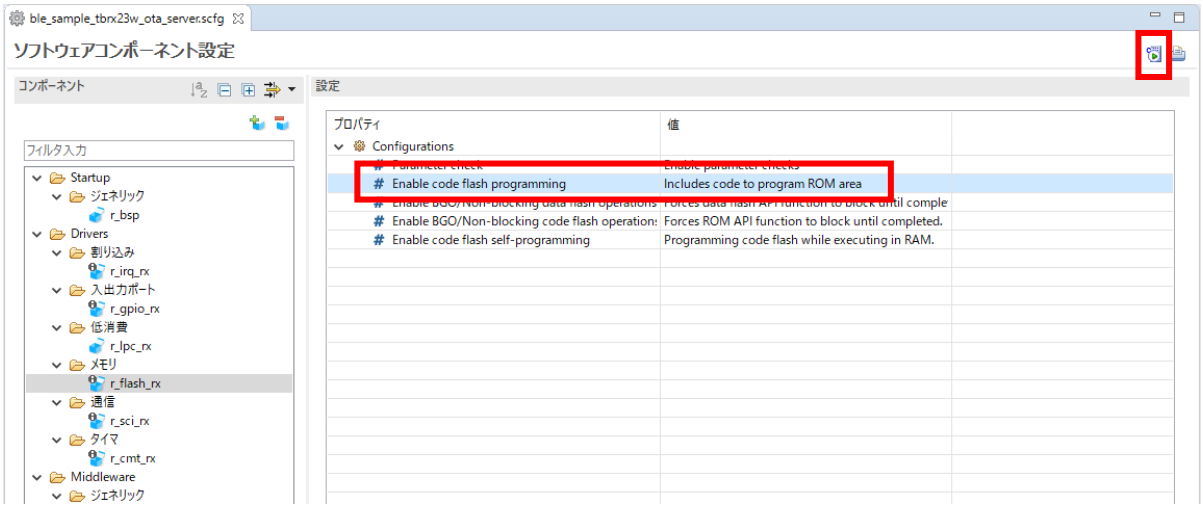

図 4.13 Flash FIT モジュールの設定

4.6 Renesas OTA Reset Service の追加

ユーザアプリケーション実行中に、ファームウェアのバージョン情報の確認や、ダウンローダへの切り替えを実現するために、GATT データベースに Renesas OTA Reset サービスを追加します。

QE for BLE から GATT データベースへのサービスを登録します。Import ボタンをクリックし、同梱の JSON ファイル(Renesas_OTA_Reset_Service_Server.json)を指定して追加します。

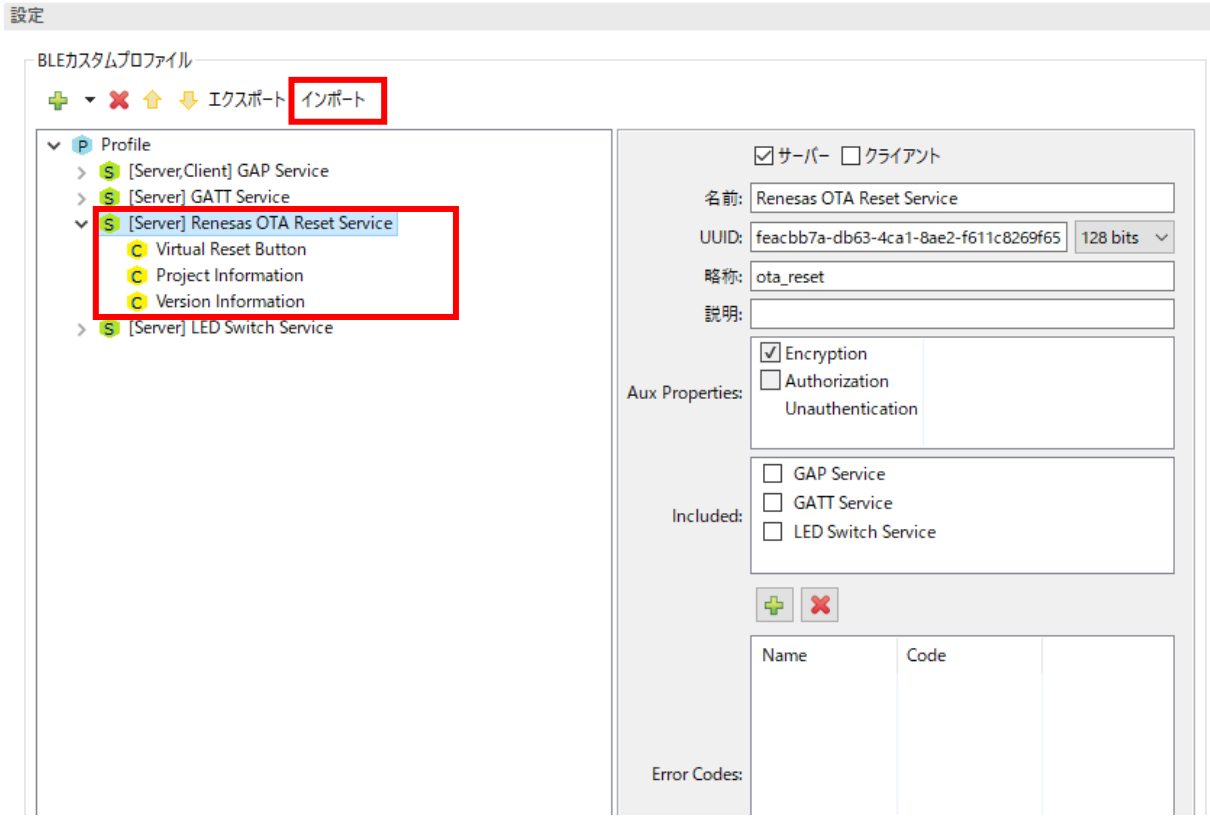

図 4.14 OTA Reset Service の追加画面

サービスを追加後、コード生成を実行します。

図 4.15 コード生成の実行

smc_gen¥Config_BLE_PROFILE フォルダ内のファイルが上書きされます。上書き前のファイルは、trash フォルダに保持されています。元のソースコードが必要な場合は、gatt_db.c / gatt_db.h 以外のファイルをコピーして復元します。

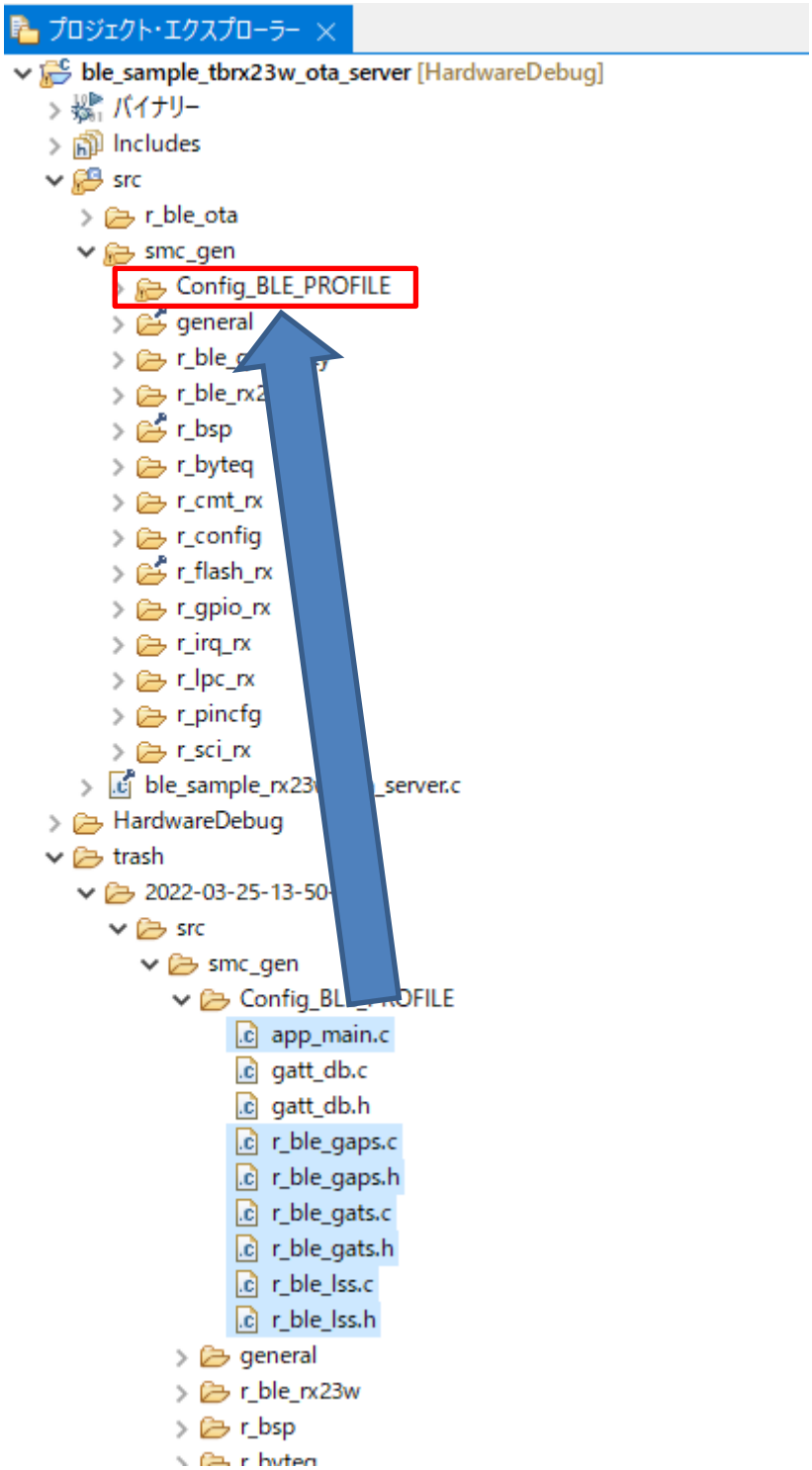

図 4.16 trash から既存プログラムをコピー

4.7　ソースコードの各セクションへの割り当て

各プログラムを 4.2 章で設定したセクションへ割り当てます。e² studio はソースコードのフォルダ毎に、プログラムを配置するセクションを指定できます。プロジェクトエクスプローラーからフォルダを右クリックし、プロパティを選択してメニュー画面を開き、”C/C++ ビルド”-“設定”-“ツール設定”-“Compiler”-“オブジェクト”で各プログラムのセクションを指定します。

次のフォルダおよびファイルはブートローダセクション(BL_*)に配置します。

r_ble_ota/bootloader

r_ble_ota/utility

smc_gen/general

smc_gen/r_flash_rx

smc_gen/r_bsp

<ProjectName>.c (BSP モジュールから呼び出される main 関数を BL_*に配置する)

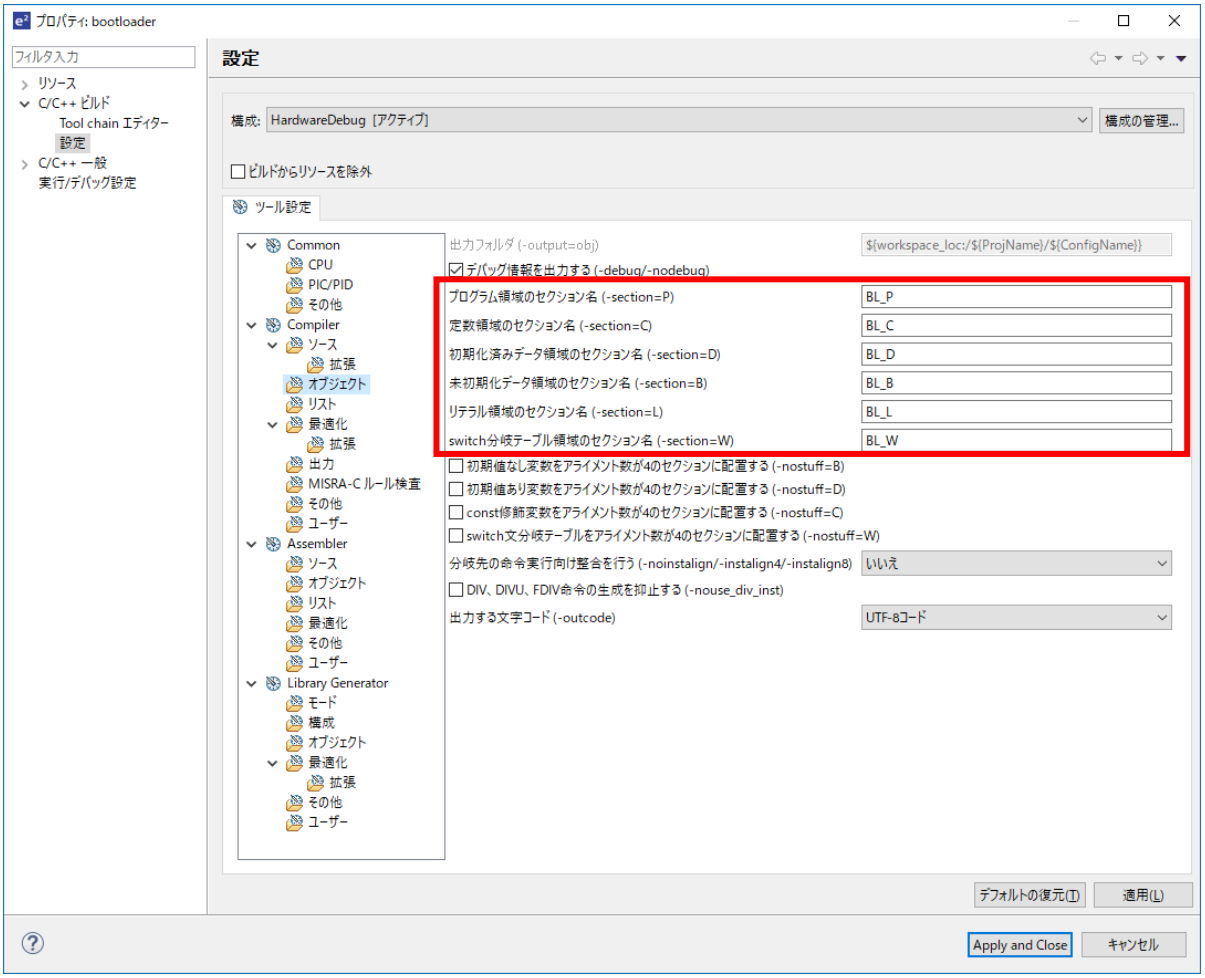

図 4.17　ブートローダセクションの指定

次のフォルダはダウンローダセクション(FWDL_*)に配置します。

r_ble_ota/downloader

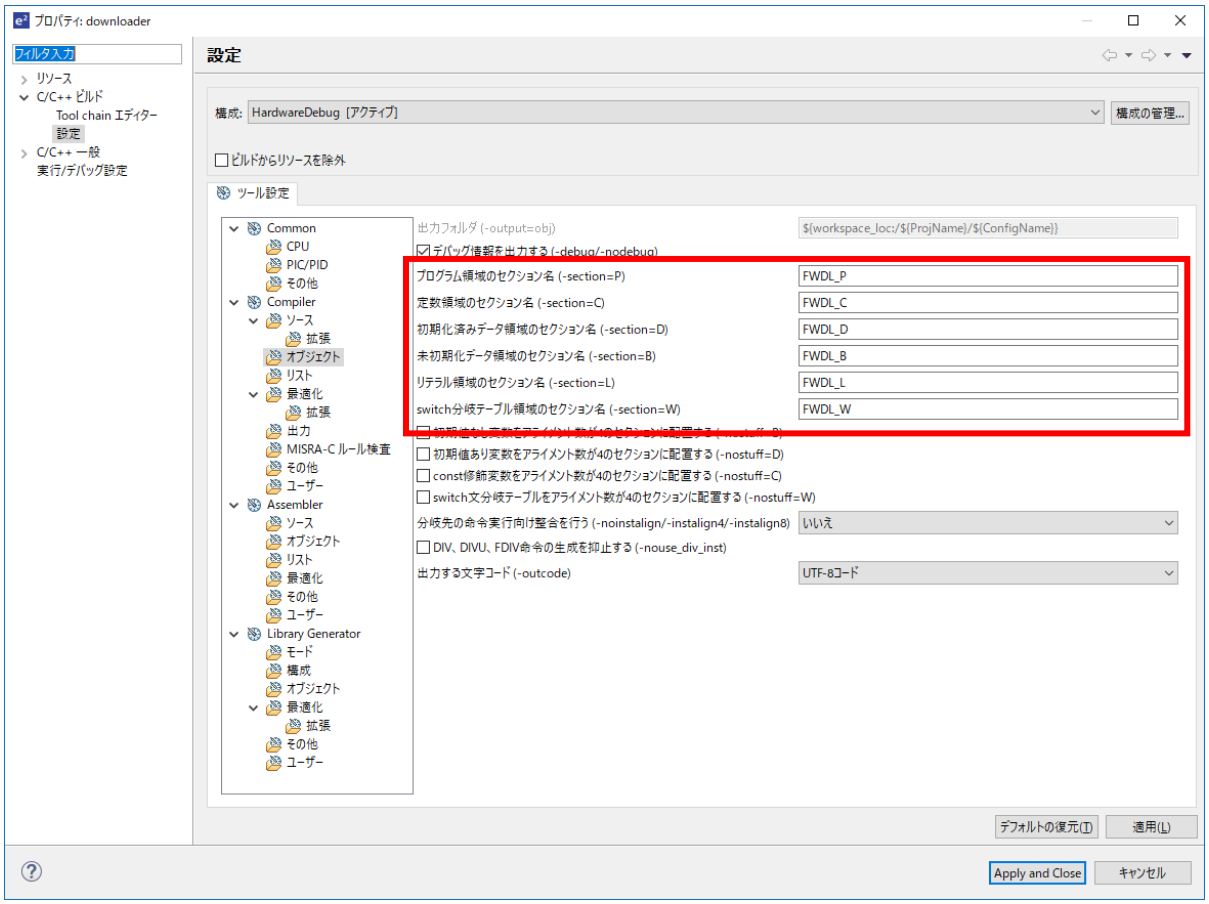

図 4.18　ダウンローダセクションの設定

次のフォルダはリロケータセクション(RL_*)に配置します。

r_ble_ota/relocator

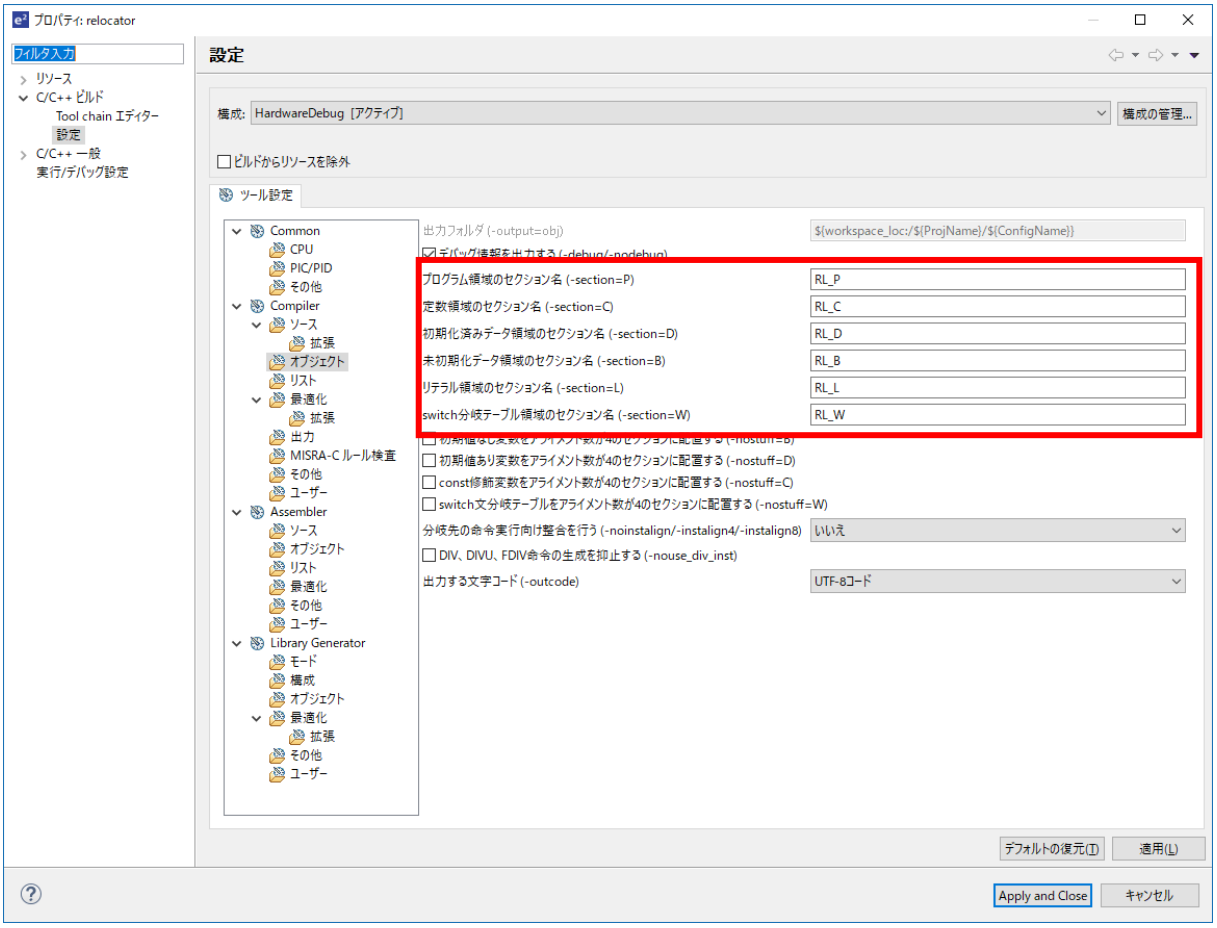

図 4.19　リロケータセクションの設定

4.8　dbsct.c の編集

RX23W を含む RX ファミリでは、マイコン起動時に__INITSCT 関数によって RAM が初期化されます。__INITSCT 関数は、_DTBL テーブル及び、BTBL を参照して RAM を初期化します。このテーブルを変更し、起動時にブートローダと標準ライブラリの RAM を初期化します。

これらの変数は BSP FIT モジュールの下記ファイルで定義されます。

src¥smc_gen¥r_bsp¥mcu¥all¥dbsct.c

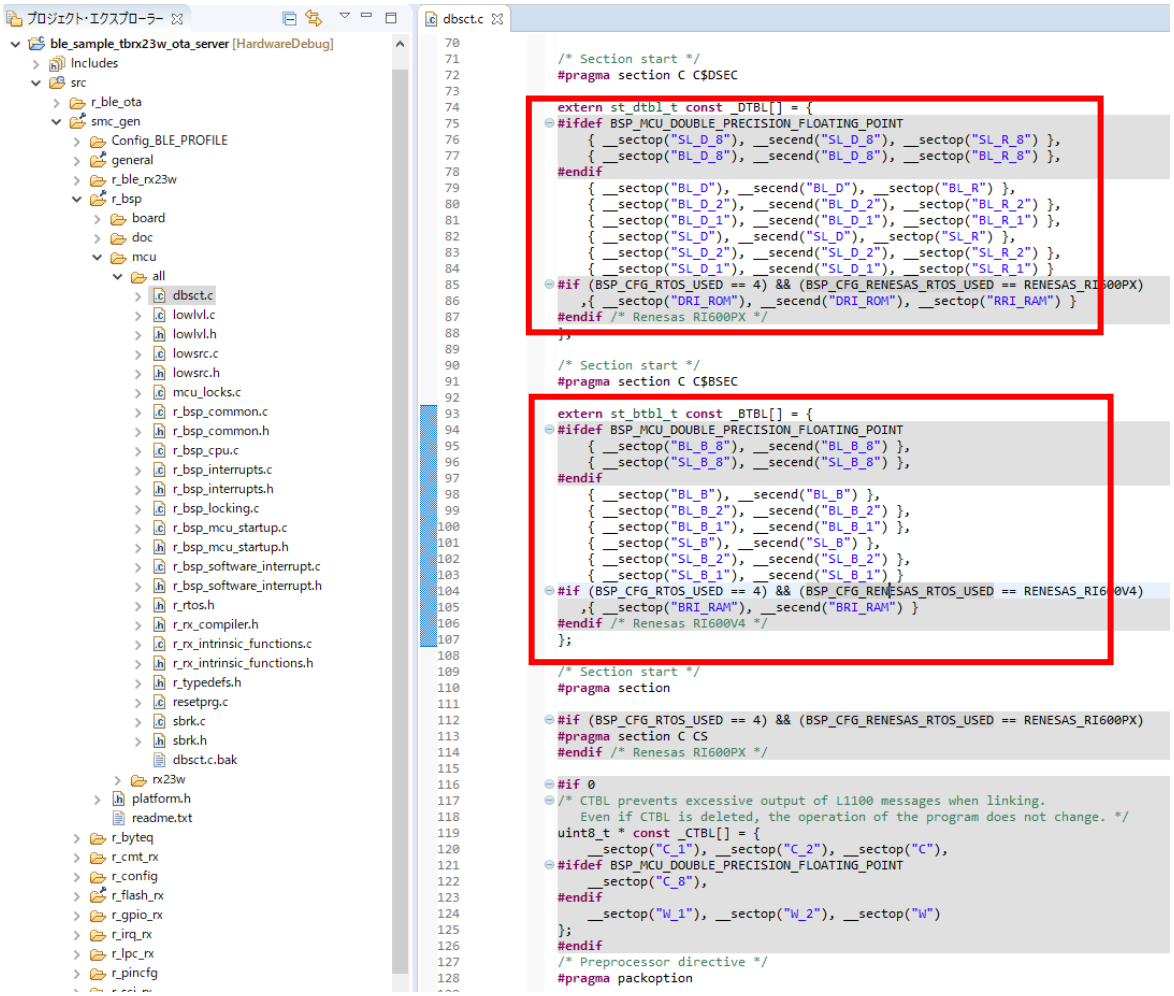

図 4.20　**dbsct.c ファイルの修正**

次に CTBT を無効化しリロケータセクションへの差分を除去します。このテーブルを削除した事による実行ファイルへの影響はありません。

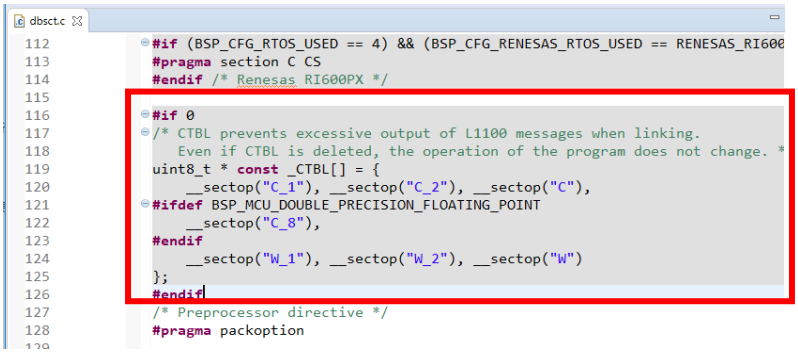

図 4.21 CBTL の無効化

4.9 r_ble_ota コンフィグレーション設定

OTA Server の動作、セクション情報とバージョン情報は、r_ble_ota_config.h で設定します。設定する項目を表 4-2 に示します。

表 4-2 r_ble_ota コンフィグレーションオプション

マクロ名	デフォルト値	説明
BLE_OTA_DEBUG_PUTS	空マクロ	ダウンローダ実行中の更新イベント情報のログ出力先関数名を指定します。空マクロの場合、ログ出力は行われません。
MAIN_FUNCTION	app_main	ユーザアプリケーションのメイン関数を指定します。引数と戻り値の型は void 型になります。
BLE_OTA_PROJECT_NAME	“ota_sample”	ファームウェアのプロジェクト情報を設定します。最大で 18 文字です。ファームウェアを区別するために使用されます。 **【注】このコンフィグはプロジェクト作成時に設定してください。ファームウェアの更新では変更しないでください。**
BLE_OTA_RESET_CODE	"inge9ubled9hy4tljn"	OTA Reset Service の Virtual Reset Button に書き込む値です。18 文字で指定してください。プロジェクト作成時に設定し、ファームウェアの更新を通して変更しないようにしてください。 **【注】このコンフィグはプロジェクト作成時に設定してください。ファームウェアの更新では変更しないでください。**
BLE_OTA_DOWNLOADER_DEVICE_NAME	“FWU-DEV”	ダウンローダ実行中のデバイス名を設定します。本マクロで設定した値は、GAP Service の DeviceName Characteristic とアドバタイズ時のスキャンレスポンスデータの Complete Name として使用されます。 **アプリケーションと同様の値を推奨します。**
BLE_OTA_APPLICATION_MAJOR_VERSION	0x01	アプリケーションのメジャーバージョンを設定します。
BLE_OTA_APPLICATION_MINOR_VERSION	0x0A	アプリケーションのマイナーバージョンを設定します。
BLE_OTA_DOWNLOADER_MAJOR_VERSION	0x01	ダウンローダのメジャーバージョンを設定します。BLE Protocol Stack の変更やダウンローダを変更し、ダウンローダセクションを変更した場合に変更してください
BLE_OTA_DOWNLOADER_MINOR_VERSION	0x0A	ダウンローダのマイナーバージョンを設定します。BLE Protocol Stack の変更やダウンローダを変更し、ダウンローダセクションを変更した場合に変更してください
BLE_OTA_RELOCATER_MAJOR_VERSION	0x01	リロケータのメジャーバージョンを設定します。リロケータ若しくはブートローダを変更した場合に変更してください。
BLE_OTA_RELOCATER_MINOR_VERSION	0x00	リロケータのマイナーバージョンを設定します。リロケータ若しくはブートローダを変更した場合に変更してください。
BLE_OTA_APPLICATION_SECTION_SIZE	0x31800 (198KB)	アプリケーションセクションのサイズを指定します。
BLE_OTA_TEMPORALY_SECTION_SIZE	0x31800 (198KB)	OTA Server が受信したファームウェアを一時的に保持する ROM サイズを指定します。BLE_OTA_APPLICATION_SECTION_SIZE と同一の値を設定してください。

マクロ名	デフォルト値	説明
BLE_OTA_DOWNLOADER_SECTION_SIZE	0x31800 (198KB)	ダウンローダセクションのサイズを指定します。
BLE_OTA_RELOCATER_SECTION_SIZE	0x4000 (16KB)	リロケータセクションのサイズを指定します。
BLE_OTA_STDLIB_SECTION_SIZE	0x14000 (80KB)	標準ライブラリのセクションサイズを指定します。
BLE_OTA_DL_FW_BUFFER_SIZE	0x4000	ダウンローダが受信するファームウェアブロックサイズの最大値を設定します。
BLE_OTA_DL_CVECT_BUFFER_SIZE	0x0400	可変ベクタテーブルを受信する際のバッファサイズを指定します。変更は不要です。
BLE_OTA_DATAFLASH_BLOCK_LENGTH	0x0400	データフラッシュのブロックサイズを指定します。変更は不要です。
BLE_OTA_USE_DATAFLASH_START_ADDRESS	0x00100400	プログラムの切り替えフラグ保持のために使用するデータフラッシュのブロックの開始アドレスを指定します。
BLE_OTA_CODEFLASH_BLOCK_LENGTH	0x0800	使用するマイコンのコードフラッシュの最低書き換えサイズを指定します。
BLE_OTA_CODEFLASH_START_ADDRESS	0xFFF80000	使用するマイコンの開始アドレスを指定します。
BLE_OTA_CODEFLASH_SIZE	0x00080000 (512KB)	使用するマイコンのコードフラッシュのサイズを指定します。
BLE_OTA_RAM_START_ADDRESS	0x00000000	使用するマイコンの RAM のスタートアドレスを指定します。
BLE_OTA_RAM_SIZE	0x00010000	使用するマイコンの RAM サイズを指定します。
BLE_OTA_TRAPERROR_LED_ENABLE	空マクロ	復帰不可能なエラーが発生した場合に、LED 点滅を有効にするかどうかを指定します。未定義の場合は、エラー発生の LED 制御が無効になります。

4.10 ユーザアプリケーションの設定

本章では、ブートローダにユーザアプリケーションのメイン関数を設定します。また、ユーザアプリケーションの起動前に初期化する RAM のセクションを設定します。

4.10.1 メイン関数の設定

OTA Server プログラムは、プログラムの切り替えのためにブートローダを実装しています。ユーザアプリケーションのメイン関数実行前に、bootloader_main 関数を呼び出します。

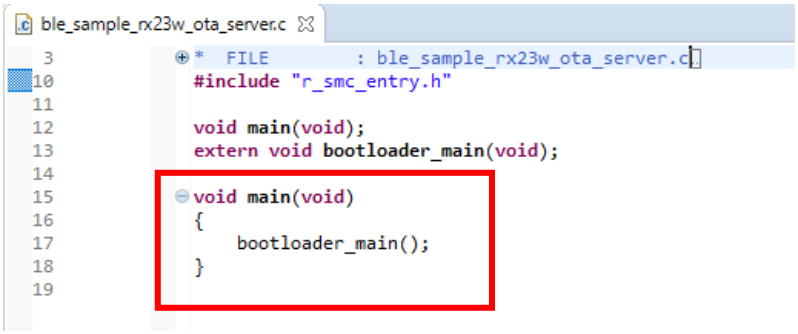

図 4.22 main 関数の変更

4.10.2 ユーザアプリケーションの main 関数の登録と初期化セクションの設定

ユーザアプリケーションの main 関数は、application_loader 関数を経由して実行されます。r_ble_config.h の MAIN_FUNCTION マクロにユーザアプリケーションの main 関数を設定してください。

```
r_ble_ota_config.h
 2   * DISCLAIMER
19
20   #ifndef R_BLE_OTA_CONFIG_H
21   #define R_BLE_OTA_CONFIG_H
22
23   #define BLE_OTA_DEBUG_PUTS
24
25   /* OTA Configuration */
26
27   /* Application Main Function */
28   #define MAIN_FUNCTION                          app_main
29   /* Project information */
30
31   /* BLE_OTA_PROJECT_NAME length < 18*/
32   #define BLE_OTA_PROJECT_NAME                "ota_sample"
33
34   /* BLE_OTA_PROJECT_NAME length = 18*/
35   #define BLE_OTA_RESET_CODE          "inge9ubled9hy4tljn"
36
```

図 4.23　メイン関数の設定

ユーザアプリケーションは、r_ble_ota_app_loader.c ファイルの application_loader 関数から呼び出されます。

application_loader 関数は、RAM の初期化を実行します。ユーザアプリケーションがデフォルト以外のセクションを利用する場合には、initialized_ram 関数を変更してください。

```
r_ble_ota_app_loader.c
 2   * DISCLAIMER
19
20   #include <string.h>
21   #include "r_ble_ota_header.h"
22   extern void MAIN_FUNCTION(void);
23
24   static void initialized_ram(void)
25   {
26       memcpy(__sectop("R_1"),__sectop("D_1"),__secsize("R_1"));
27       memcpy(__sectop("R_2"),__sectop("D_2"),__secsize("R_2"));
28       memcpy(__sectop("R"),__sectop("D"),__secsize("R"));
29
30       memset(__sectop("B"), 0x00, __secsize("B"));
31       memset(__sectop("B_1"), 0x00, __secsize("B_1"));
32       memset(__sectop("B_2"), 0x00, __secsize("B_2"));
33
34   #ifdef BSP_MCU_DOUBLE_PRECISION_FLOATING_POINT
35       memcpy(__sectop("R_8"), __sectop("R_8") ,__secsize("R_8"));
36       memset(__sectop("B_8"), __sectop("B_8") ,__secsize("B_8"));
37   #endif
38   }
39
40   void application_loader(void)
41   {
42       initialized_ram();
43       MAIN_FUNCTION();
44   }
45
```

図 4.24 r_ble_ota_app_loader.c

4.11 ユーザアプリケーションの機能追加

FWU_Client による OTA ファームウェアアップデートを行うために、ユーザアプリケーションに以下の処理を実装します。

1. Renesas OTA Reset Service とダウンローダへの切り替え処理
2. ボンディングの実施
3. GATT Service の Service Changed Characteristic の Indication 処理 (iOS を利用する場合のみ)

4.11.1 Renesas OTA Reset Service とダウンローダへの切り替え処理

Renesas OTA Reset Service は、FWU_Client が、ファームウェアのプロジェクト名やバージョン情報を読み出し、プログラムをダウンローダに、切り替えるために使用されます。

ユーザアプリケーションに、以下の機能を追加します。

- Renesas OTA Reset Service の各キャラクタリスティックへの値の設定
- Reset Code 書き込み時の一致確認とダウンローダへの切り替え処理

これらの機能は全て、パッケージに同梱されている r_ble_ota_resets.c ファイルに実装されています。プロジェクト内の本ファイルを同梱パッケージのものに置き換える事で機能を追加できます。

置き換え対象ファイル

- ${ProjectName}¥smc_gen¥Config_BLE_PROFILE¥r_ble_ota_resets.c
- ${ProjectName}¥smc_gen¥Config_BLE_PROFILE¥r_ble_ota_resets.h

パッケージ同梱の当該ファイル

- r01an5910xx0111-rx23w-otafwup¥ProjectSetting¥service¥service api¥r_ble_ota_resets.c
- r01an5910xx0111-rx23w-otafwup¥ProjectSetting¥service¥service api¥r_ble_ota_resets.h

本機能を有効にするために、app_main.c の R_BLE_Execute の実行前に、空の ota_resets_cb 関数を実装し、R_BLE_OTA_RESETS_Init 関数を実行してください。詳しい実装方法については、本サンプル(ble_sample_tbrx23w_ota_server)の app_main.c ファイルをご参照ください。

```
#include "r_ble_ota_resets.h"

void ota_resets_cb(uint16_t type, ble_status_t result,  st_ble_servs_evt_data_t *p_data)
{
}


/* Initialize Renesas OTA Reset Service server API */
R_BLE_OTA_RESETS_Init(ota_resets_cb);
```

図 4.25 app_main.c への Renesas OTA Reset Service の追加例

r_ble_ota_resets.c は、Project Information Characteristic と Version Information Characteristic に Read Request が来た時に、それぞれ、g_ble_ota_project_info 変数、各 HEADER セクションのバージョン情報を GATT データベースに設定しています。この処理は、それぞれ以下の関数に実装されています。

```
static void ble_ota_resets_Project_info_read_req_cb(const void *p_attr, uint16_t conn_hdl)
static void ble_ota_resets_version_info_read_req_cb(const void *p_attr, uint16_t conn_hdl)
```

これらの関数は、各キャラクタリスティック定義変数に登録されています。

```
/* Project Information characteristic definition */
static const st_ble_servs_char_info_t gs_project_info_char = {
    .start_hdl    = BLE_OTA_RESETS_PROJECT_INFO_DECL_HDL,
    .end_hdl      = BLE_OTA_RESETS_PROJECT_INFO_VAL_HDL,
    .char_idx     = BLE_OTA_RESETS_PROJECT_INFO_IDX,
    .app_size     = sizeof(st_ble_seq_data_t),
    .db_size      = BLE_OTA_RESETS_PROJECT_INFO_LEN,
    .decode       = (ble_servs_attr_decode_t)decode_st_ble_seq_data_t,
    .encode       = (ble_servs_attr_encode_t)encode_st_ble_seq_data_t,
    .read_req_cb  = ble_ota_resets_Project_info_read_req_cb,
};

/* Version Information characteristic definition */
static const st_ble_servs_char_info_t gs_version_info_char = {
    .start_hdl    = BLE_OTA_RESETS_VERSION_INFO_DECL_HDL,
    .end_hdl      = BLE_OTA_RESETS_VERSION_INFO_VAL_HDL,
    .char_idx     = BLE_OTA_RESETS_VERSION_INFO_IDX,
    .app_size     = sizeof(st_ble_ota_resets_version_info_t),
    .db_size      = BLE_OTA_RESETS_VERSION_INFO_LEN,
    .decode       = (ble_servs_attr_decode_t)decode_st_ble_ota_resets_version_info_t,
    .encode       = (ble_servs_attr_encode_t)encode_st_ble_ota_resets_version_info_t,
    .read_req_cb  = ble_ota_resets_version_info_read_req_cb,
};
```

図 4.26 r_ble_ota_resets.c の Read Request のコールバック関数の登録

r_ble_ota_resets.c では、Virtual Reset Button Characteristic への、Write Request イベントで Reset Code の一致を確認します。Reset Code が一致したときには、Write Comp イベントで対抗デバイスから切断します。その後、ダウンローダへのリセット処理を行います。

ダウンローダへの切り替え処理は r_ble_ota/utility/r_ble_ota_reset.c に実装された R_BLE_OTA_SwitchModeReset 関数を使用します。以下に R_BLE_OTA_SwitchModeReset 関数の使用方法を示します。DataFlash から現在の設定を読み出し、activate_mode にダウンローダを指定します。R_BLE_OTA_SwitchModeReset 関数は起動プログラムフラグ情報を、不揮発領域に書き出し MCU リセットを行います。

```
st_ble_ota_dataflash_info_t data_flash_info;

R_BLE_OTA_FLASH_DRIVER_Open();
R_BLE_OTA_FLASH_DRIVER_ReadDataFlash(&data_flash_info);
R_BLE_OTA_FLASH_DRIVER_Close();

data_flash_info.activate_mode = BLE_OTA_ACTIVATE_MODE_DOWNLOADER;
data_flash_info.status_info.code = BLE_OTA_STATUS_SUCCESS;
data_flash_info.status_info.source = BLE_OTA_ACTIVATE_MODE_APPLICATION;

R_BLE_OTA_SwitchModeReset(&data_flash_info);
```

図 4.27 R_BLE_OTA_SwitchModeReset の使用例

R_BLE_OTA_SwitchModeReset 関数の実行中は、コードフラッシュへのアクセスが禁止されます。割り込みが発生するとベクタテーブルを経由してコードフラッシュへのアクセスが発生します。何らかの割り込みの発生が想定される場合には、r_ble_resets.c 内の r_ble_ota_resets_gatts_cb で割り込み禁止を設定してください。

```
st_ble_ota_dataflash_info_t data_flash_info;

R_BLE_OTA_FLASH_DRIVER_Open();
R_BLE_OTA_FLASH_DRIVER_ReadDataFlash(&data_flash_info);
R_BLE_OTA_FLASH_DRIVER_Close();

data_flash_info.activate_mode = BLE_OTA_ACTIVATE_MODE_DOWNLOADER;
data_flash_info.status_info.code = BLE_OTA_STATUS_SUCCESS;
data_flash_info.status_info.source = BLE_OTA_ACTIVATE_MODE_APPLICATION;

R_BSP_InterruptsDisable();

R_BLE_OTA_SwitchModeReset(&data_flash_info);
```

図 4.28 割り込み禁止の設定例

r_ble_ota_resets.c は、R_BLE_API のホストスタックに GATT Server のコールバック関数を一つ登録します。GATTS コールバック関数の登録数を、ユーザアプリケーションの使用数+1 以上に設定してください。QE for BLE で生成された app_main.c では、最大値の BLE_GATTS_MAX_CB (15)に設定されています。

```
/* GATT server initialization parameters */
static st_ble_abs_gatts_init_param_t gs_abs_gatts_init_param =
{
    .p_cb_param = gs_abs_gatts_cb_param,
    .cb_num     = BLE_GATTS_MAX_CB,
};
```

図 4.29 コールバック関数の登録数の設定

4.11.2 ボンディングの実施

ダウンローダ実行中は、OTA Server BLE Protocol Stack の app_lib のセキュリティライブラリが指定する Data Flash からペアリング情報を読み出して使用しますが、ペアリング情報の不揮発領域の書き換えを行ないません。そのため、ユーザアプリケーションで、ボンディングを行う事を推奨します。

BLE Protocol Stack の app_lib のセキュリティライブラリを利用してユーザアプリケーション実行中にペアリング情報を Data Flash に保持するようにしてください。

セキュリティライブラリの使用方法は、”Bluetooth Low Energy プロトコルスタック 基本パッケージ ユーザーズマニュアル (R01UW0205)”の 5.2 章 セキュリティデータ管理”をご覧ください。また、”Bluetooth Low Energy アプリケーション開発者ガイド (R01AN5504)の 9 章 セキュリティ”もご覧ください。

4.11.3 GATT Service の Service Changed Characteristic の Indication 処理

iOS デバイスは、Bluetooth デバイスに関するいくつかの情報を BD アドレスに紐づけてキャッシュしています。キャッシュされる情報には GATT データベースの構造も含まれます。

OTA Server は更新中のダウンローダ実行状態とアプリケーション実行状態で別の GATT データベースを持つため、更新動作を行うためにキャッシュ情報の削除が必要です。

GATT データベースのキャッシュ削除には、GATT Service の Service Changed Characteristic の Indication を送信します。iOS 版 FWU-Client を使用する場合には、接続後に Service Changed Characteristic を Indication ができるように実装をお願いします。

```
uint16_t cccd = 0;
R_BLE_GATS_GetServChangedCliCnfg(p_data->conn_hdl, &cccd);
if((cccd & BLE_GATTS_CLI_CNFG_INDICATION) == BLE_GATTS_CLI_CNFG_INDICATION)
{
    st_ble_gats_serv_changed_t serv_changed;
    serv_changed.start_hdl = 0x0000;
    serv_changed.end_hdl = 0xffff;
    R_BLE_GATS_IndicateServChanged(p_data->conn_hdl, &serv_changed);
}
```

図 4.30 Service Changed Characteristic の Indication

本サンプルでは、Service Changed Characteristic の Client Characteristic Configuration Descriptor (CCCD)で Indication が許可された直後に Indication を送信しています。

5. 更新用ファームウェアの確認

ソースコードを変更し、ファームウェアが変更される場合には、該当するセクションのバージョン情報を設定してください。アプリケーションのバージョンは、ファームウェアが変更されるたびにインクリメントする事をお勧めします。

本 OTA サンプルプログラムは、r_ble_ota_config.h の XXX_VERSION マクロによってファームウェアの更新可否を判断しています。バージョン情報が矛盾していると更新後に正しく動作しません。

更新用ファームウェアの更新可否を判断するスクリプト(CompareFirmware.py)を提供しています。このスクリプトを利用して、ファームウェアの差分を確認出来ます。実行時引数に、ファームウェア情報が含まれるフォルダのディレクトリのパスを指定します。

実行例：

$> python CompareFirmware.py “ota_sample_Version1.10” “ota_sample_Version1.11”

実行後、各セクションの差分の数と比較結果が表示されます。

```
C:¥r01an5910xx0110-rx23w-otafwup¥Tool>python CompareFirmware.py "C:¥r01an5910xx0110-rx23w-otafwup¥SampleFirmware¥FWU Cli
ent¥ota_sample_Version1.10" "C:¥r01an5910xx0110-rx23w-otafwup¥SampleFirmware¥FWU Client¥ota_sample_Version1.11"
VariableVectorTableSection num of diff 14
ApplicationSection num of diff 62453
DownloderAndProtocolStackSection num of diff 0
VariableVectorTableBackupSection num of diff 0
StandardLibrarySection num of diff 0
SwapTemporarySection num of diff 0
SwapSection num of diff 0
APPLICATION_UPDATE_MODE
```

図 5.1 CompareFirmware.py 実行画面

結果の意味は以下の通りです。

表 5-1 CompareFile.py の比較結果

result	意味
APPLICATION_UPDATE_MODE	アプリケーションセクションのみにファームウェアの差分が存在します。アプリケーション更新モードで更新できます。
DOWNLOADER_UPDATE_MODE	アプリケーションセクションと、ダウンローダセクションにファームウェアの差分が存在します。BLE Protocol Stack 更新モードでファームウェアを更新できます。
FIRMWARE_ERROR	標準ライブラリセクション、若しくはリロケータセクションに差分が存在します。OTA による更新はできません。
FIRMWARE_INFORMATION_ERROR	json ファイルのセクション情報が一致しません。
NO_REQUIRED_UPDATE	json ファイルで示された各セクションの差分が存在しません。更新する必要はありません。

6. Android 版 FWU_Client

本節は、スマートフォンアプリ FWU_Client を Android Studio でビルド、デバッグする方法を簡単に説明します。Android Studio の詳細は Android Developer サイト(https://developer.android.com/)を参照してください。

6.1 ビルド手順

FWU_Client は Android Studio Version 4.1 で開発されました。Android Studio のメニューから Open を選択しパッケージ内の FWU_Client フォルダを選択しプロジェクトを開きます。メニューバーの Build から Make Project を選択しビルドを行います。

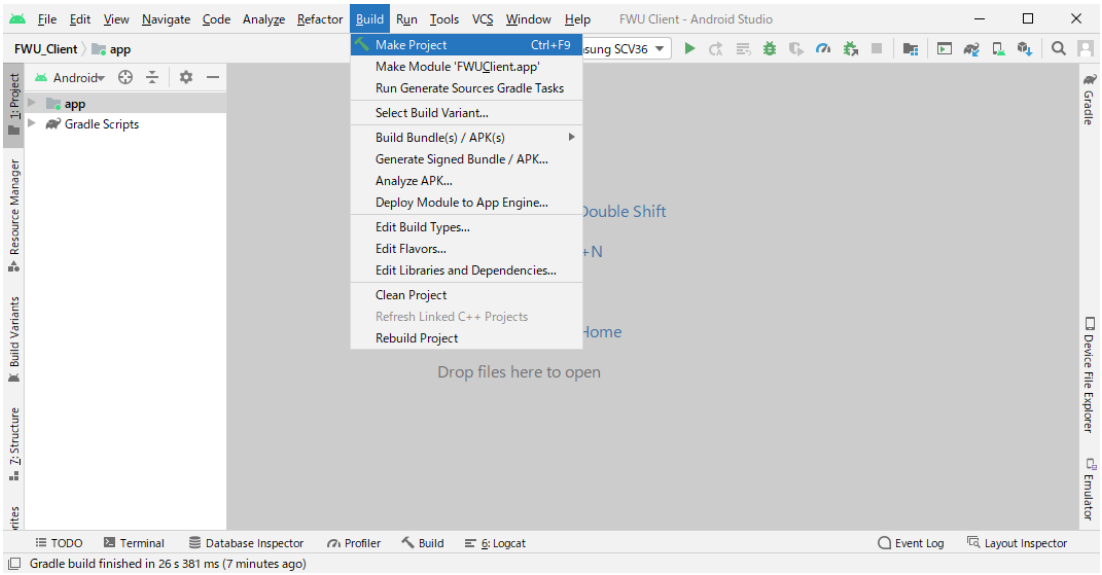

図 6.1　ビルドメニュー

6.2 デバッグ手順

Android デバイスの開発者オプションで USB デバッギングを有効にします。USB デバッギングを有効化した Android デバイスを PC に接続すると、デバッグデバイスリストに表示されます。メニューバーの”Run”から”Debug ‘app’”を選択しデバッグを開始します。

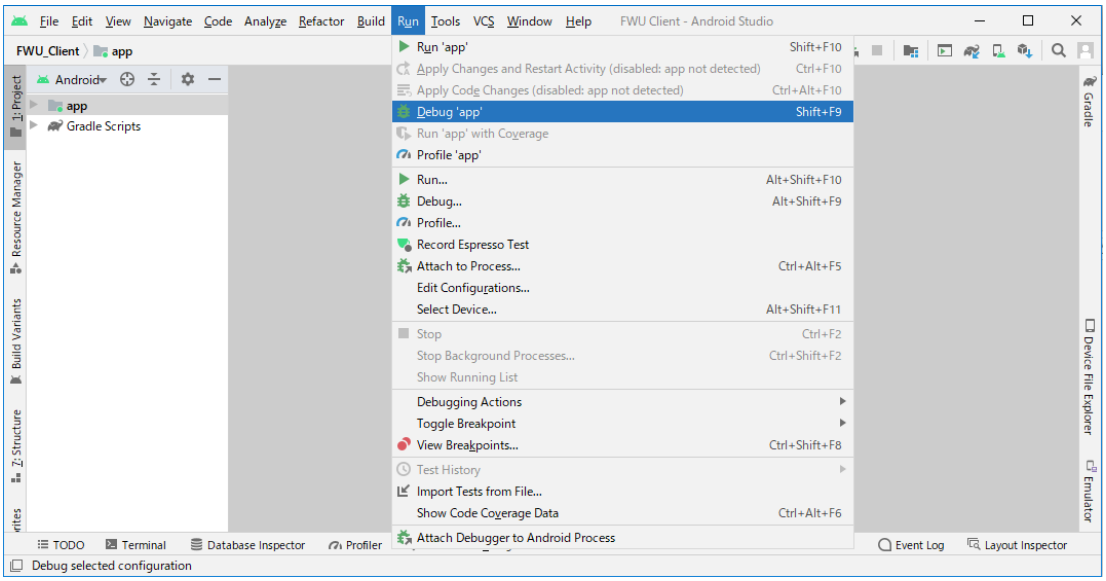

図 6.2　デバッグメニュー

6.3　Android 10.0 の API 仕様変更への対応

Android 版 FWU_Client では、ファイルの読み込みに Android10.0 で仕様変更された API が使用されています。以下の API で取得したディレクトリのパスに対してファイルアクセスが禁止されるようになりました。ACTION_OPEN_DOCUMENT_TREE インテント等を使用してファームウェアフォルダへのアクセス権をユーザから取得するようにしてください。

(utils/FIleUtils.java 39 行目) Environment.getExternalStorageDirectory()

また、Bluetooth LE のスキャン実行に必要な権限が強化されています。パーミッションチェックについて、以下の修正をお願い致します。

(bluetooth_device_scan/BluetoothDeviceScanActivity.java 480,482 行目)

ACCESS_COARSE_LOCATION → ACCESS_FINE_LOCATION

7. iOS 版 FWU_Client

本節では、iOS 向けスマートフォンアプリ FWU_Client の開発環境を説明します。

7.1 動作確認環境

動作確認環境を示します。

表 7-1 iOS 版 FWU_Client の動作確認環境

分類	環境
開発環境	Xcode 12.5.1
動作デバイス	iPhone 8 iOS 14.7.1
開発デバイス	MacBook Air (M1,2020) macOS Big Sur version 11.2.3

7.2 デバッグ手順

2.4.1 節をご覧ください。

7.3 iOS の GATT データベースのキャッシュ機能

iOS デバイスは、Bluetooth デバイスに関するいくつかの情報を BD アドレスに紐づけてキャッシュしています。キャッシュされる情報には GATT データベースの構造も含まれます。iOS 版 FWU_Client は、OTA Server の GATT データベースを確実に検出するために、接続後 Service Changed Characteristic の Indication を待機します。

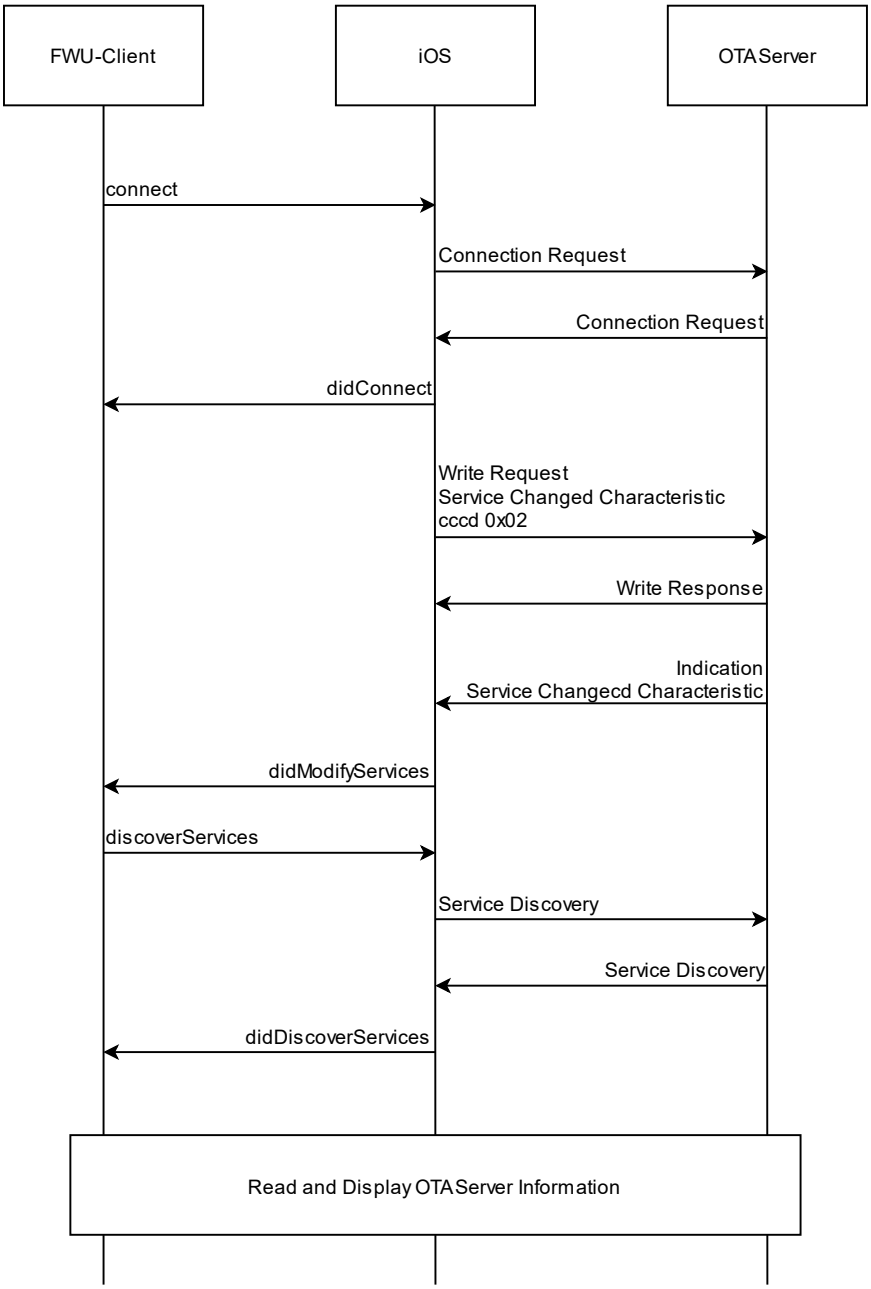

図 7.1 iOS 版 FWU_Clinet の接続時のシーケンスチャート

8. Python 版 FWU_Client

8.1 システム構成

Python 版 FWU_Client は、RX23W 用プログラム OTA Client と、OTA Client を制御する Python Application からなります。OTA Client は、Bluetooth LE を利用して OTA Server のファームウェア更新をします。Python Application は、OTA Client が定義するアプリケーションパケットを用いて更新用ファームウェアを転送します。ファームウェア転送経路には BLE FIT Module が提供する app_lib のコマンドラインインタフェース(CLI)を利用して通信します。

Python Application は、制御コマンドを受け取ってファームウェアの更新を実行します。

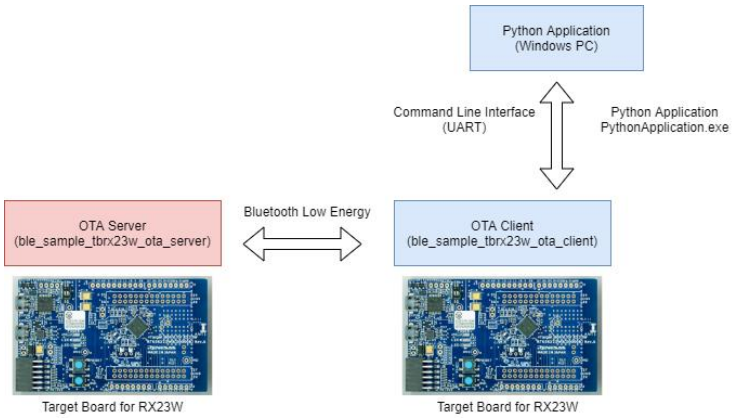

図 8.1 システム構成図

8.2 制御コマンド

Python 版 FWU_Client の Python Application は表 8-1 のコマンドを受け取って OTA Server のファームウェアアップデートを実行します。

表 8-1 Python Application のコマンド一覧

コマンド	引数	動作
scan	timeout (s) scan_mode filter_type filter_data	OTA Client が timeout 時間の間スキャンを実行します。 scan_mode 0: すべてのデバイス 1: ユーザ指定 filter_type と filter_data 2: Renesas OTA Reset Service UUID 3: Renesas OTA Service UUID 例: scan 5 0 scan 5 1 9 FWU-DEV
conn	timeout (s) bd_address addr_type [public:0,random:1]	OTA Client が指定したデバイスに接続要求を実行し OTA Server のファームウェア情報を読み出します。 例: conn 5 74:90:50:FF:FF:FF 0 conn 5 DF:3C:31:5D:08:D9 1
update	Update_mode [app,ble] Firmware_directory_path	conn コマンドで接続したデバイスに対して OTA によるアップデートを開始します。 例: update app C:¥FWU Client¥ota_sample_Version1.11 update ble C:¥FWU Client¥ota_sample_Version1.12
disconn	なし	conn コマンドで接続したデバイスとの接続を切断します。
exit	なし	Python Application を終了します。

8.3　各コマンドの動作シーケンス

Python 版 FWU_Client の各コマンド実行時の動作シーケンスを示します。OTA Client と Python Application 間のパケットの詳細については 8.4.2 節をご覧ください。

8.3.1　scan コマンド

scan コマンドは、周辺の Bluetooth LE デバイスを Scan しアドバタイズパケットのデータを表示します。図中の赤文字は OTA Client と Python Application 間のパケットを示します。

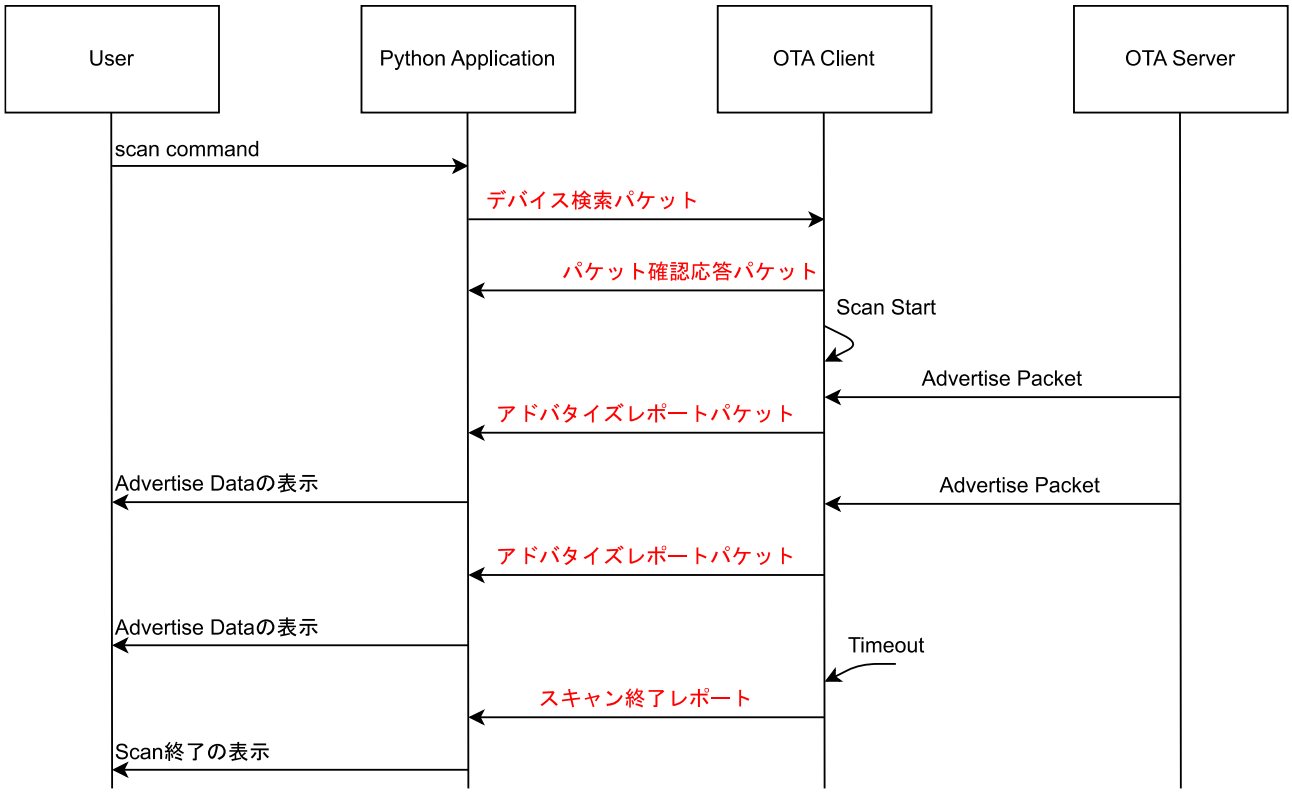

図 8.2 scan コマンドの実行シーケンス

8.3.2　conn コマンド

conn コマンドは、OTA Server への接続とファームウェア情報の読み出し表示を行います。非 OTA Server に接続された場合には切断されます。OTA Client の持つ鍵情報が OTA Server と異なる場合は、OTA Client の鍵情報を削除し切断します。図中の赤文字は OTA Client と Python Application 間のパケットを示します。

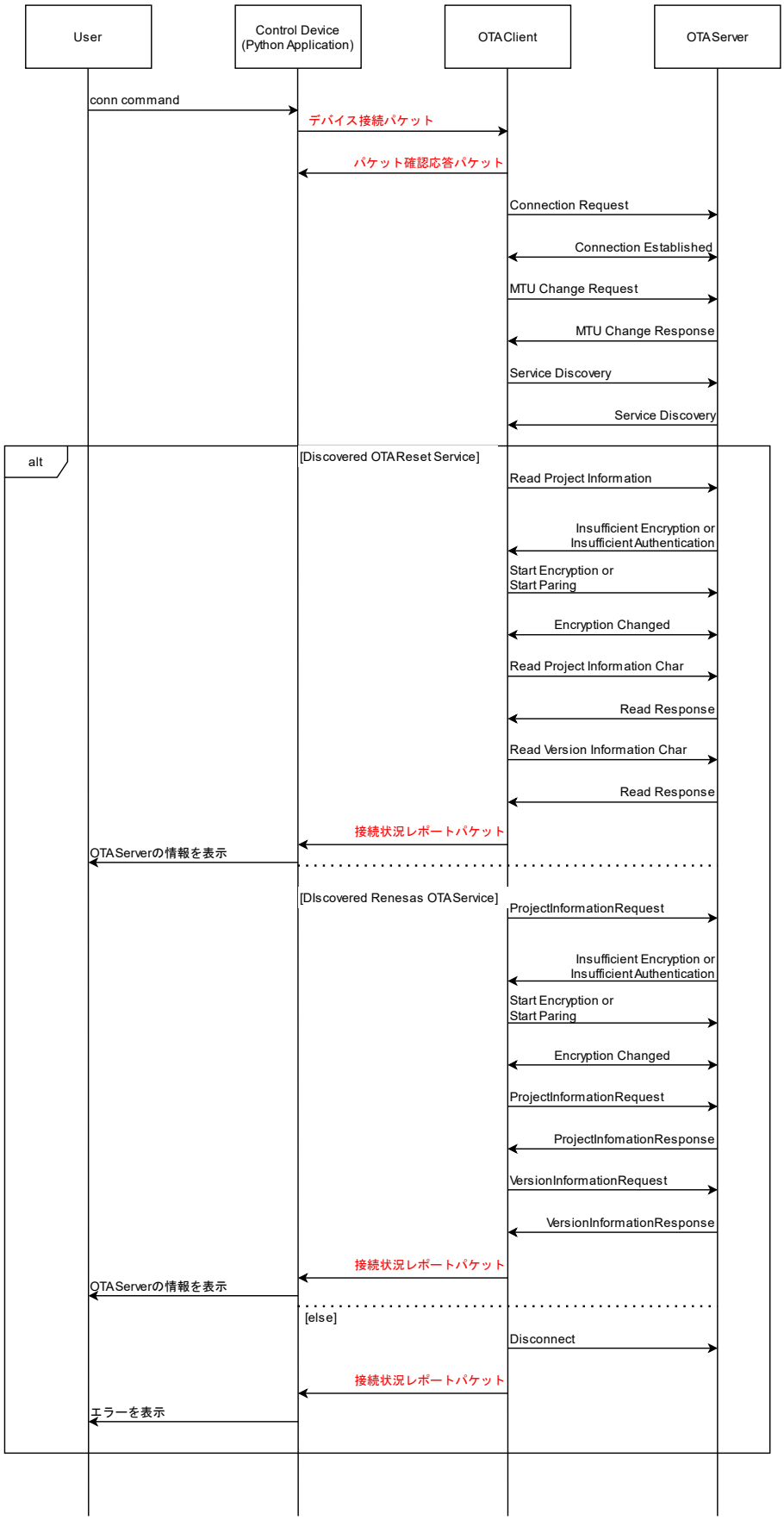

図 8.3 conn コマンドの実行シーケンス

8.3.3 disconn コマンド

disconn コマンドは、OTA Server との接続を切断します。図中の赤文字は OTA Client と Python Application 間のパケットを示します。

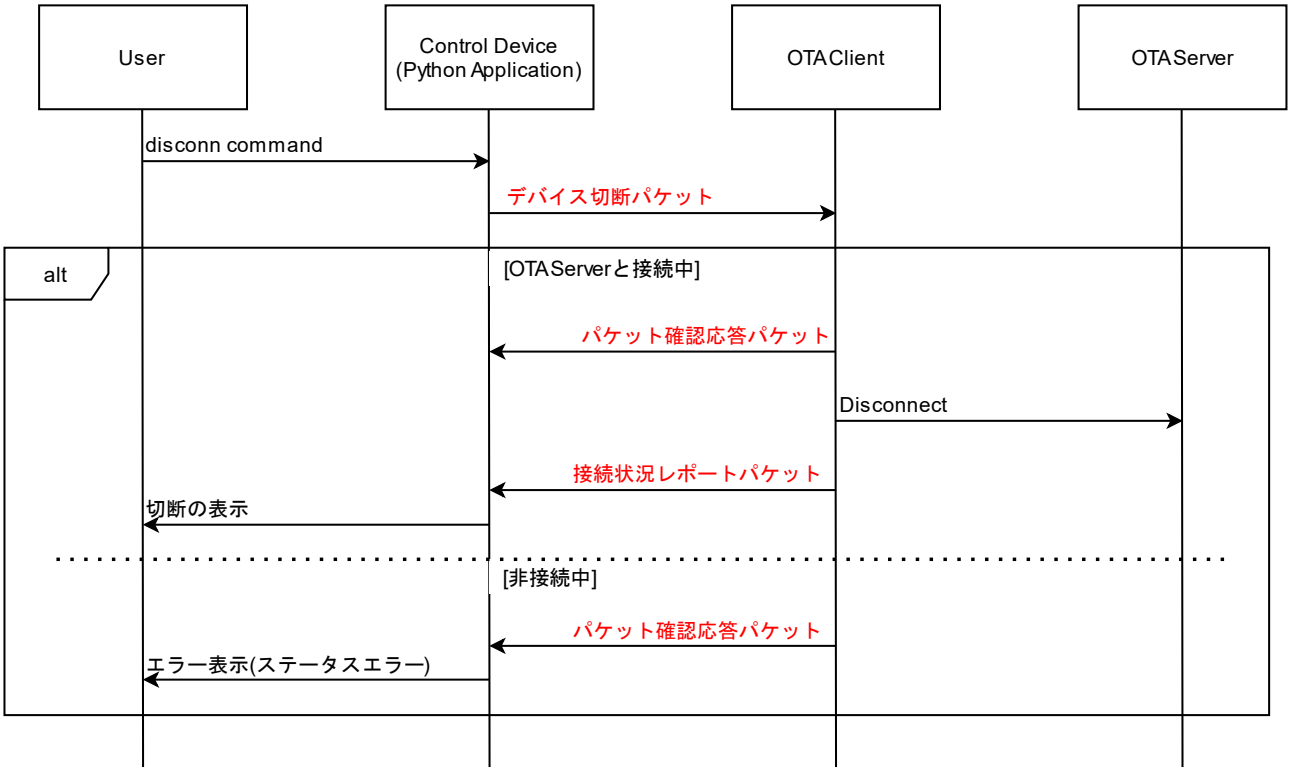

図 8.4 disconn コマンドの実行シーケンス

8.3.4 update コマンド

update コマンドは、3.5 節に示した OTA Server の OTA 更新シーケンスを実行します。ファームウェアブロックを送信するたびに Python Application からファームウェアを取得します。図中の赤文字は OTA Client と Python Application 間のパケットを示します。

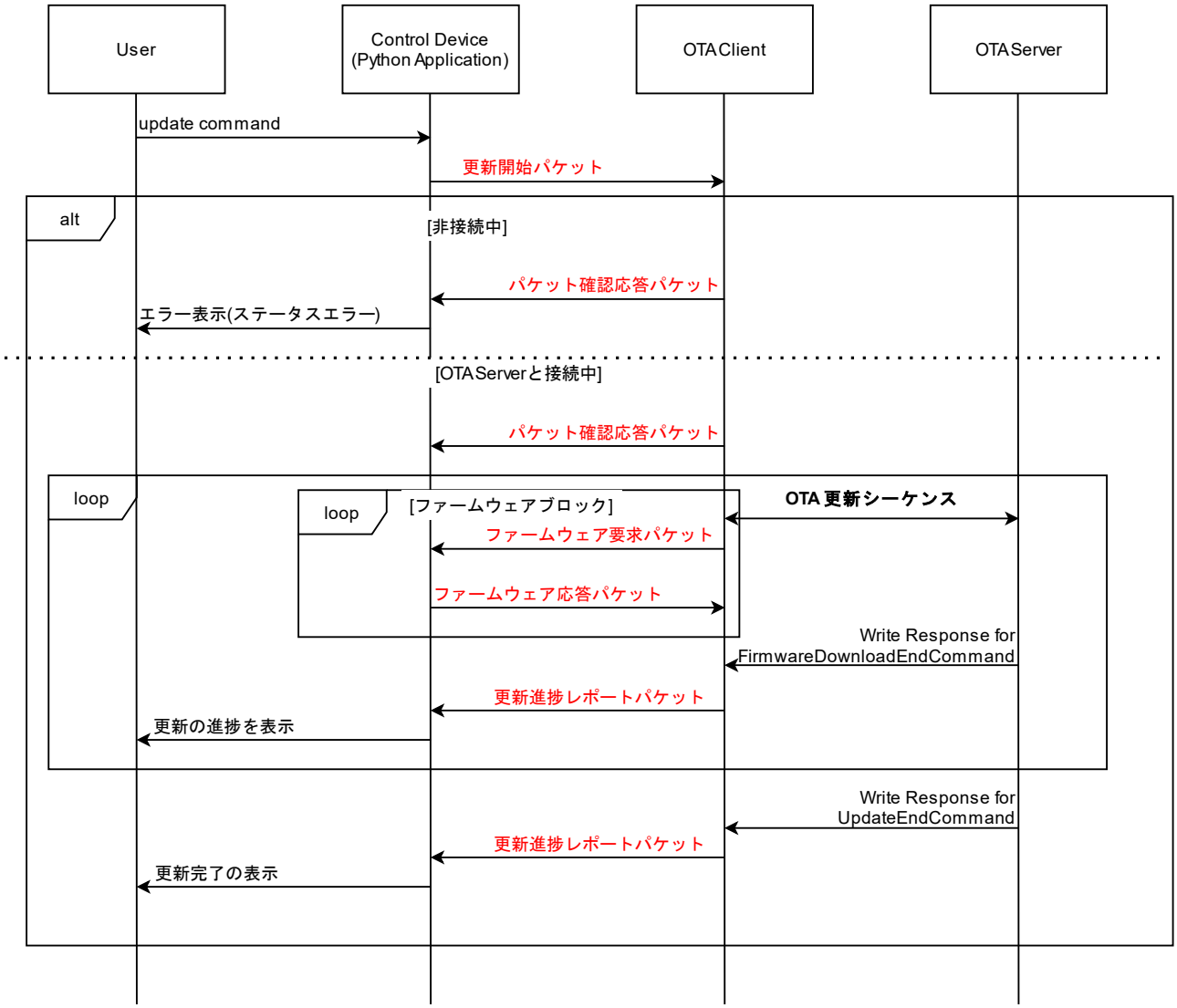

図 8.5 update コマンドの動作シーケンス

8.3.5　exit コマンド

exit コマンドは Python Application を終了するコマンドです。

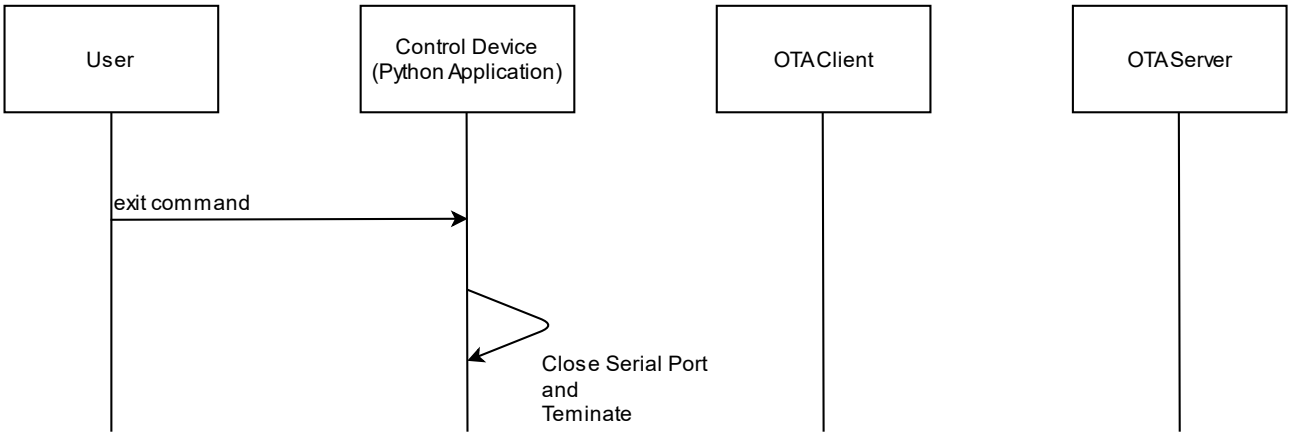

図 8.6 exit コマンドの動作シーケンス

8.4 OTA Client

OTA Client は、OTA Server のファームウェア更新機能を持ちます。OTA Client は、8.4.2 節に定義されるアプリケーションパケットによって更新機能を提供します。アプリケーションパケットによって OTA Client を操作するデバイスをコントロールデバイスと呼びます。

本サンプルでは、Python Application を OTA Client のコントロールデバイスとして実装しています。

8.4.1 ブロック図

OTA Client は 3 つのモジュールとそれを利用するメインモジュールから成ります。

図 8.7 にブロック図を示します。

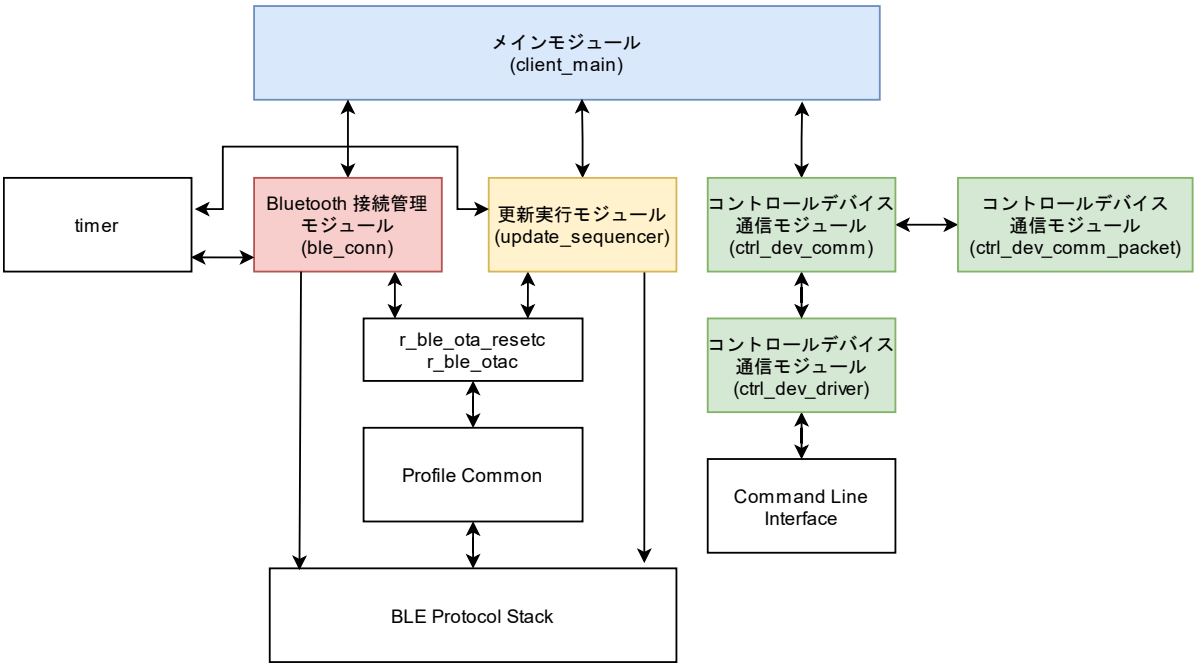

図 8.7 OTA Client のブロック図

各モジュールの機能を説明します。

1. Bluetooth 接続管理モジュール (ble_conn)

   ble_conn は、OTA Server デバイスの Scan と接続、プロジェクト情報の読み出しを行います。

2. 更新実行モジュール(update_sequencer)

   update_sequencer は、接続している OTA Server のファームウェア更新を行います。

3. コントロールデバイス通信モジュール(ctrl_dev_comm, ctrl_dev_comm_packet,ctrl_dev_driver)

   ctrl_dev_comm は、8.4.2 節で定義されるアプリケーションパケットを受け取りメインモジュールに通知します。

   ctrl_dev_comm_packet は、8.4.2 節で定義されるアプリケーションパケットのデータ構造とシリアライズ関数、デシリアライズ関数を実装しています。

   ctrl_dev_driver は、任意の通信経路から受信したアプリケーションパケットデータを ctrl_dev_comm に通知します。本サンプルでは、BLE FIT Module の app_lib のコマンドラインインタフェース機能を利用してアプリケーションパケットを受け取っています。

4. メインモジュール(client_main)

   client_main は、上記三つのモジュールを利用して OTA Server のファームウェアアップデートを実現します。各モジュールの初期化、通信パラメータの設定を行います。

8.4.2 アプリケーションパケット

OTA Client は、周辺デバイスの検索、更新デバイスへの接続、接続デバイスのファームウェアアップデートシーケンスの実行に関して、アプリケーションパケットを定義します。コントロールデバイスと OTA Client は以下に定義するパケットによりデータ通信を行います。

OTA Client 通信パケットは、以下のデータ構造を持ちます。N は通信経路に依存します。N は 70 ~255 バイトです。チェックサムフィールドは、パケット種別コードから、パケットデータを 1 バイトずつ加算した値です。

パケット種別コード	パケットデータ	チェックサム
1byte	N byte	1byte

各パケットデータと動作は以下の通りです。

パケット名	デバイス検索			パケット種別コード	0x01
送信者	コントロールデバイス				
Client の動作	OTA Client は指定されたデータでアドバタイズパケットをフィルタリングします。OTA Client は指定された時間スキャンを実行します。 デバイスの検索結果は、アドバタイズレポートパケットで通知されます。				
パケットデータ					
パケット種別コード	スキャン時間(秒)	フィルタデータタイプ	フィルターデータ長	フィルターデータ	チェックサム
1byte	1byte	1byte	1byte	N byte	1byte

パケット名	デバイス接続		パケット種別コード	0x02
送信者	コントロールデバイス			
Client の動作	OTA Client は、指定された Bluetooth LE デバイスに対して接続します。 接続の結果は、接続状況レポートパケットで通知されます。			
パケットデータ				
パケット種別コード	タイムアウト時間(秒)	BD アドレス 74:50:90:ff:00:ff → 0x74,0x50,0x90,0xff,0x00,0xff	アドレスタイプ ランダム 0x01 パブリック 0x00	チェックサム
1byte	1byte	6byte	1byte	1byte

パケット名	デバイス切断	パケット種別コード	0x03
送信者	コントロールデバイス		
Client の動作	OTA Client は、接続中のデバイスから切断します。 切断結果は接続状況レポートパケットで通知されます。		
パケットデータ			

パケット種別コード	チェックサム
1byte	1byte

パケット名	更新開始	パケット種別コード	0x04
送信者	コントロールデバイス		
Client の動作	OTA Client は、接続中のデバイスに対してファームウェア更新シーケンスを実行します。更新モードでアプリケーション更新か BLE Protocol Stack 更新かを選択します。 更新中には、更新進捗パケットとファームウェア要求パケットが OTA Client から送信されます。		
パケットデータ			

パケット種別コード	更新モード 0x00: アプリケーション更新 0x01: BLE Protocol Stack 更新	可変ベクタテーブルセクションの開始アドレス	可変ベクタテーブルのサイズ(byte)	アプリケーションセクションの開始アドレス	アプリケーションセクションのサイズ(byte)
1byte	1byte	4byte	4byte	4byte	4byte

Downloader セクションの開始アドレス	Downloader セクションのセクションサイズ(byte)	バージョン情報	プロジェクト名	リセットコード	ファームウェアブロックサイズ(KB)	チェックサム
4byte	4byte	6byte	18byte	18byte	1byte	1byte

パケット名	ファームウェアデータ応答		パケット種別コード	0x05
送信者	コントロールデバイス			
Client の動作	OTA Client から通知されたファームウェア要求パケットに応答するパケットです。ファームウェア要求パケットで指定されたアドレスとサイズのファームウェアを送信します。			
パケットデータ				
パケット種別コード	ファームウェア要求パケット番号	ファームウェアデータ長	ファームウェアデータ	チェックサム
1byte	1byte	1byte	N byte	1byte

パケット名	パケット確認応答		パケット種別コード	0x06
送信者	コントロールデバイス			
Client の動作	OTA Client から通知されたパケットのチェックサムが一致しなかった時に送るパケット			
パケットデータ				
パケット種別コード	応答パケット種別	エラーコード	チェックサム	
1byte	1byte	1byte	1byte	

パケット名	アドバタイズレポート		パケット種別コード	0x81	
送信者	OTA Client				
Client の動作	デバイス検索パケットに対する応答です。スキャンによって発見されたデバイスをコントロールデバイスに通知します。				
パケットデータ					
パケット種別コード	BD アドレス	BD アドレスタイプ 0x01：ランダム 0x00：パブリック	アドバタイズデータ長	アドバタイズデータ	チェックサム
1byte	6byte	1byte	1byte	N byte	1 byte

パケット名	スキャン終了レポート	パケット種別コード	0x82
送信者	OTA Client		
Client の動作	スキャンが終了したことをコントロールデバイスに通知します。		
パケットデータ			
パケット種別コード	チェックサム		
1byte	1byte		

パケット名	接続状況レポート		パケット種別コード	0x83	
送信者	OTA Client				
Client の動作	接続中の Bluetooth LE デバイスの情報を通知します。				
パケットデータ					
パケット種別コード	接続状況 0x00: 接続成功 0x01: 接続失敗 0x02: 読出失敗 0x03: timeout 0x04: UUID 不適 0x05:切断された	BD アドレス	プロジェクト名	バージョン情報	OTA Server の実行中のプログラム 0x00: ユーザアプリケーション 0x01: ダウンローダ
1byte	1byte	6byte	18byte	8byte	1byte
チェックサム					
1byte					

パケット名	更新進捗レポート	パケット種別コード	0x84
送信者	OTA Client		
Client の動作	OTA Server のファームウェアアップデートの進捗をコントロールデバイスに通知します。 更新シーケンス実行中にコントロールデバイスに定期的に送信されます。		
パケットデータ			

パケット種別コード	更新状態 0x00:更新成功 0x01:更新中 0x02:更新失敗	更新失敗理由	更新中のアドレス	送信ブロック数	総ブロック数	チェックサム
1byte	1byte	1byte	4byte	1byte	1byte	1byte

パケット名	ファームウェアデータ要求	パケット種別コード	0x85
送信者	OTA Client		
Client の動作	更新に必要なファームウェアをコントロールデバイスに要求します。		
パケットデータ			

パケット種別コード	ファームウェア要求パケット番号	開始アドレス	サイズ	チェックサム
1byte	1byte	4byte	1byte	1byte

パケット名	パケット確認応答	パケット種別コード	0x86
送信者	OTA Client		
Client の動作	デバイス検索、デバイス接続、デバイス切断、更新開始パケットに対して応答を返します。		
パケットデータ			

パケット種別コード	応答パケット種別コード	パケットエラーコード	チェックサム
1byte	1byte	1byte	1byte

表 8-2 パケットエラーコード一覧

エラーコード	エラーコード名	エラー理由
0x00	成功	成功
0x01	パラメータエラー	引数が範囲外
0x02	ステータスエラー	OTA Client が、パケットを受け付けない状態である。
0x03	チェックサムエラー	チェックサムが違う
0x04	その他のエラー	その他のエラー

8.4.3 コマンドラインインタフェースでの通信

本サンプルの OTA Client は、BLE FIT Module の CLI を用いてコントロールデバイスと通信します。コントロールデバイスからデータを受信は、ota crtl コマンドを使用します。

コントロールデバイスは 1 バイトデータを 2 バイトの文字列としてコマンドに入力します。例えば、接続要求コマンド実行時のコマンド構造は以下の通りです。

ota ctrl 02745090ff00ff0054

コントロールデバイスへのパケット送信は、CLI の R_BLE_CLI_Printf 関数を使用しています。1 バイトのデータを 2 文字の文字列として先頭に”app:”をつけて表示します。

例えば、OTA Client が、ファームウェアの 0xfff80000 アドレスから 0xf0byte のファームウェアを要求する場合のファームウェアデータ要求パケットは以下のように出力されます。

app:84010000f8ff0f8B

アプリケーションパケットを送受信する通信経路は r_ble_ota_cl_ctrl_dev_driver.c に実装されています。この実装を変更する事で独自の通信経路でアプリケーションパケットをやり取りできます。
r_ble_ota_cl_ctrl_dev_driver.h に定義されている関数を実装します。また、通信経路から受信したパケットのバイナリデータを、R_BLE_OTA_CL_CTRL_DEV_DRIVER_Init 関数で渡されるコールバック関数に通知します。

8.4.4　app_lib の修正箇所

OTA Client は、OTA Server の GATT データベースの変更対応と CLI で独自の通信を行うために BLE FIT Module の app_lib の一部を修正しています。修正箇所と修正理由は以下の通りです。

表 8-3 app_lib の修正

修正箇所	修正理由	切替マクロ
src/r_ble_rx23w/app_lib/cli/r_ble_cli.c src/r_ble_rx23w/app_lib/cli/r_ble_cli.h	CLI のワンラインの拡張 CLI コマンド実行時のエコーバックの停止	BLE_OTA_CLI_CONFIG
src/r_ble_rx23w/app_lib/profile_cmn/r_ble_servc_if.h	GATT データベース初期化機能の追加	BLE_OTA_SERVC_CONFIG

8.4.5　高速通信サンプルプログラムの使用

OTA Client は、OTA Server へのファームウェアの転送に、高速通信サンプルプログラムのフロー制御 API を使用しています。高速通信サンプルプログラムについては、以下のドキュメントをご参照ください。

- RX23W グループ 高速通信用サンプルプログラム (R01AN5437)

8.4.6　注意事項

- r_ble_ota_cl_client_main.c のスキャンパラメータの Slow_period はゼロを指定してください。
- OTA Client は、複数台接続をサポートしていません。

8.5 Python Application

Python Application は、制御コマンドをもとに OTA Client を制御します。Python Application は OTA Client のコントロールデバイスとして動作します。Python Application の機能は、OTA Client のアプリケーションパケットの制御と制御用コマンドのインタフェースの提供です。

8.5.1 ブロック図

Python Application は 7 つのモジュールから成ります。図 8.8 に各モジュールのブロック図を示します。

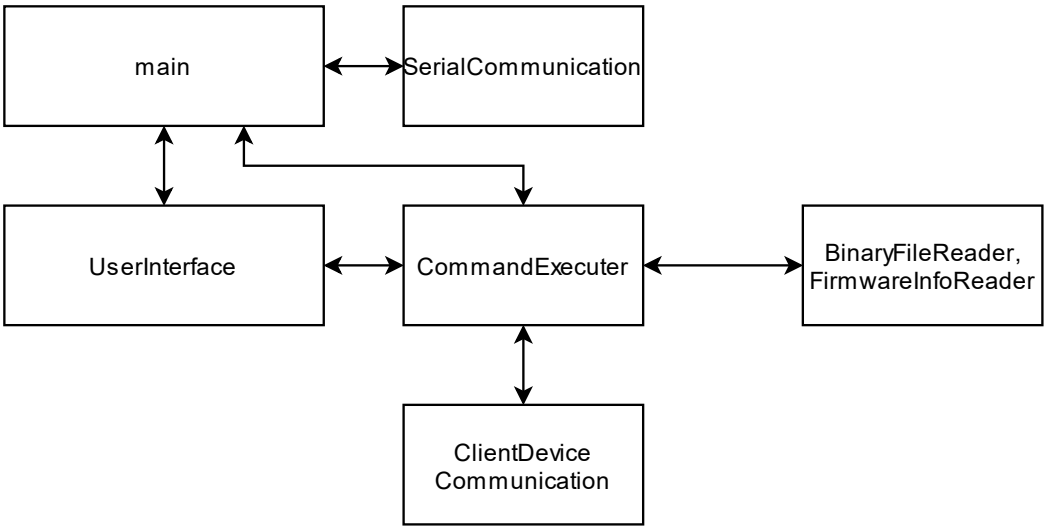

図 8.8 Python Application のブロック図

各モジュールの機能は以下の通りです。

- main.py

  Serial ポートの設定と入力されたコマンドを解析、実行します。

- SerialCommunication.py

  PySerial モジュールを利用して OTA Client とのパケットデータの送受信を行います。

- UserInterface.py

  標準入出力のラッパー層です。

- CommandExecuter.py

  入力された制御コマンドのパラメータからアプリケーションパケットを作成し、8.3 章に示したシーケンスを実行します。

- ClientDeviceCommunication.py

  8.4.2 章のアプリケーションパケットのデータ構造とシリアライズ関数を実装しています。

- BinaryFileReader.py, FirmwareInfoReader.py

  それぞれ、更新用ファームウェアの firmware.bin ファイルと firmwareinfo.json ファイルを読み出します。

8.5.2 ログ出力

Python Application は、ユーザインタフェースの表示内容とアプリケーションパケットの通信履歴をテキストファイルに出力します。

表 8-4 Python Application が出力するログ

ファイル名	ログの内容
AllSerialLog.txt	アプリケーションパケットの通信履歴
UserInterfaceLog.log	ユーザインタフェースのコマンドログ

9. Version1.00 からの変更点

9.1 ユーザアプリケーションに Service Changed Characteristic を実装しました。

本サンプルでは、Service Changed Characteristic の CCCD の書き込み後に Service Changed Characteristic の Indication を行っています。

9.2 FIT のバージョン更新

BLE FIT Module のバージョンを 2.11 から 2.20 に変更しました。

また、OTA Client、OTA Server のプロジェクトに含まれる General Purpose Input/Output Driver (GPIO) の FIT モジュールのバージョンを 3.70 から 3.90 に変更しました。

9.3 OTA Server のコード変更

Version1.10 では OTA Server の一部のコードを更新しています。

r_ble_ota/downloader/r_ble_ota_ble_app_lib/r_ble_ota_ble_interface.c

- R_BLE_OTA_GATTS_IndicateDataControl について関数内で R_BLE_GATTS_Indication を呼ぶように変更しました。
- Advertise Packet に r_ble_ota_config.h で定義される BLE_OTA_DOWNLOADER_DEVICE_NAME を <<complete device name>>としてスキャンレスポンスデータに追加しました。
- Downloader 用の GATT データベースの GAP Service の Device Name キャラクタリスティックに、BLE_OTA_DOWNLOADER_DEVICE_NAME を設定するように変更しました。
- GATT Service の Service Changed Characteristic の CCCD 書き込み時に、Service Changed Characteristic の Indication を行う処理を追加しました。

r_ble_ota/downloader/r_ble_ota_downloader.c

- r_ble_ota_dl_data_control_cmd_handler 関数内の BLE_OTA_DL_CMD_PROJECT_INFORMATION_REQUEST、BLE_OTA_DL_CMD_VERSION_INFORMATION_REQUEST イベントで実行していた R_BLE_OTA_GATTS_IndicateDataControl 関数を r_ble_ota_dl_event_handler 関数の BLE_OTA_DL_EVENT_DATA_CONTROL_WRITE_COMP イベントで実行するように変更しました。

r_ble_ota/r_ble_ota_config.h

- Device Name 設定用マクロを追加しました。

smc_gen/app_main.c

- GATT Service の Service Changed Characteristic の CCCD 書き込み時に、Service Changed Characteristic の Indication を行う処理を追加しました。

smc_gen/r_ble_gats.h

- GATT Service の Service Changed Characteristic の CCCD に関するイベント列挙体を追加しました。

smc_gen/r_ble_ota_resets.c

- R_BLE_OTA_FLASH_DRIVER_ReadDataFlash の呼び出し前後に、R_BLE_OTA_FLASH_DRIVER の Open/Close 処理を追加しました。

10. 制限事項/注意事項

OTA ファームウェア更新サンプルプログラムには以下の制限事項および注意事項があります。

- BSP および FLASH の FIT モジュールは更新対象外のセクションに配置するため、OTA ファームウェア更新は行えません。そのため、ファームウェア更新前後で BSP および FLASH の設定を変更しないでください。
- BLE プロトコルスタックおよびダウンローダを配置する Downloader セクションの最大サイズは、Application セクションのサイズ以下(デフォルト 198KB)となります。
- OTA ファームウェア更新サンプルプログラムは、ユーザアプリケーションが Peripheral であることを前提とします。
- BLE FIT モジュールのコンフィグを変更した場合には BLE プロトコルスタック/ダウンローダ/アプリケーションを更新する必要があります
- r_ble_rx23w_config.h の BLE_CFG_EN_SEC_DATA で指定したデータフラッシュのブロックにリモートデバイスの IRK 情報がなく、かつリモートデバイスが RPA を使用している場合、リモートデバイスの RPA が変わってしまうとファームウェア更新が継続できなくなります。ペアリングパラメータでリモートの IRK 要求を有効にし、ユーザアプリケーションでペアリングを実施するようにしてください。
- 実行中のプログラムをダウンローダへ切り替えると、アプリケーションの更新が完了するまでユーザアプリケーションの起動はできません。
- 本サンプルではダウンローダ実行中の MCU 省電力モードは未サポートです。
- 本サンプルでは BLE のライブラリの設定は Balance タイプのみサポート対象となります。
- Flash FIT モジュールの BGO(background operation/interrupts) Mode は使用できません。
- CCRX コンパイラのバージョンは変更できません。
- Android 版 FWU_Client は、MTU サイズを 247 に設定します。
- Android 版 FWU_Client では、ファイルの読み込みに Android10.0 で仕様変更された API が使用されています。Android10.0 以降を使用する場合は 6.3 章を参考にコードを変更してください。

11. 更新に失敗する場合の確認項目

11.1 Error Response が返る場合

1. BLE_OTA_ERROR_CODE_RESET_CODE_ERROR (0x84) の場合

本エラーは、ResetCode が、FWU_Client と OTA Server の間で異なる場合に発生します。本エラーが発生した場合には、更新用ファームウェア内の firmware_infomation.json ファイルを確認します。本ファイル内の”ResetCode”の値が、OTA Server のものと一致することを確認します。

2. BLE_OTA_ERROR_CODE_INVALID_ADDRESS (0x81)、
   BLE_OTA_ERROR_CODE_INVALID_CMD (0x90) の場合

本エラーは、FWU_Client と OTA Server の間で更新シーケンスの整合性が取れない場合に発生します。OTA Server は更新シーケンス中に切断されるとリセットを行います。通信環境の改善によりエラーが発生しなくなる可能性があります。

また、OTA Server のアプリケーションがウォッチドックタイマを使用している場合には、WDT によるリセットがダウンローダ実行中に発生する可能性があります。ユーザアプリケーションからダウンローダへは、ソフトウェアリセットを介して切り替えられますが、RX23W シリーズではソフトウェアリセットによって、WDT は停止されません。ダウンローダのメインループで、WDT のカウントをクリアしてください。

11.2 更新開始直後に停止してしまう場合

更新シーケンスの開始直後に停止する場合には、ユーザアプリケーションからダウンローダへの切り替え後の再接続に失敗している可能性があります。

firmware_information.json ファイル内の“ResetCode”の値が“0000000000000000000”の場合は、アプリケーション切り替え処理が実装されていない可能性が高いです。「4.11.1 節 Renesas OTA Reset Service とダウンローダへの切り替え処理 」に従って OTA Reset Service のサービス API ファイルを置き換えてください。

実行プログラムの切り替え時に割込みが発生すると、コードフラッシュ書き換え中のコードフラッシュへの不正な読み出しが発生し動作停止する可能性があります。割り込みが発生する場合には、「4.11.1 節 Renesas OTA Reset Service とダウンローダへの切り替え処理」を参照に割込み禁止を設定してください。

11.3 Smart Configurator のコード生成機能を使用する場合

Smart Configurator のコード生成機能を使用する場合、追加の設定と実装が必要になります。図 11.1 コード生成機能の例にコード生成機能の例を示します。本機能を使用してコード生成を行った場合ブートローダ実行中にアプリケーションセクションへの関数呼び出しが発生します。そのため、ファームウェアの更新後に正常に動作しない可能性があります。

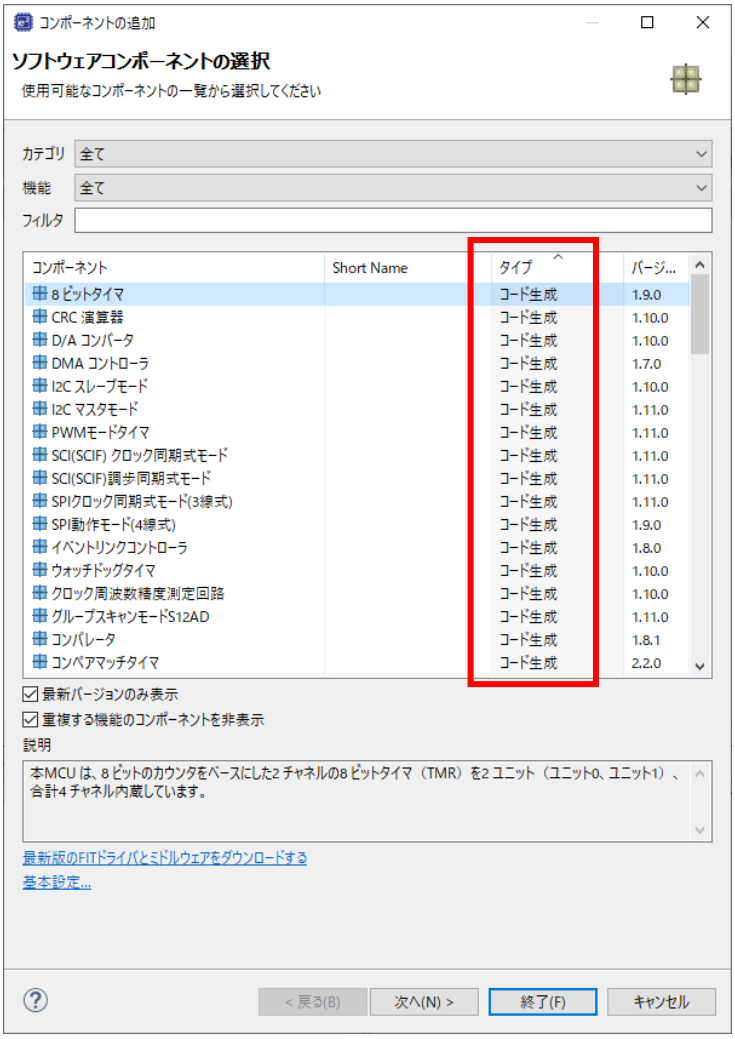

図 11.1 コード生成機能の例

本節では、この機能を使用した場合の対応方法を説明します。本ガイドでは、8 ビットタイマを例に説明します。

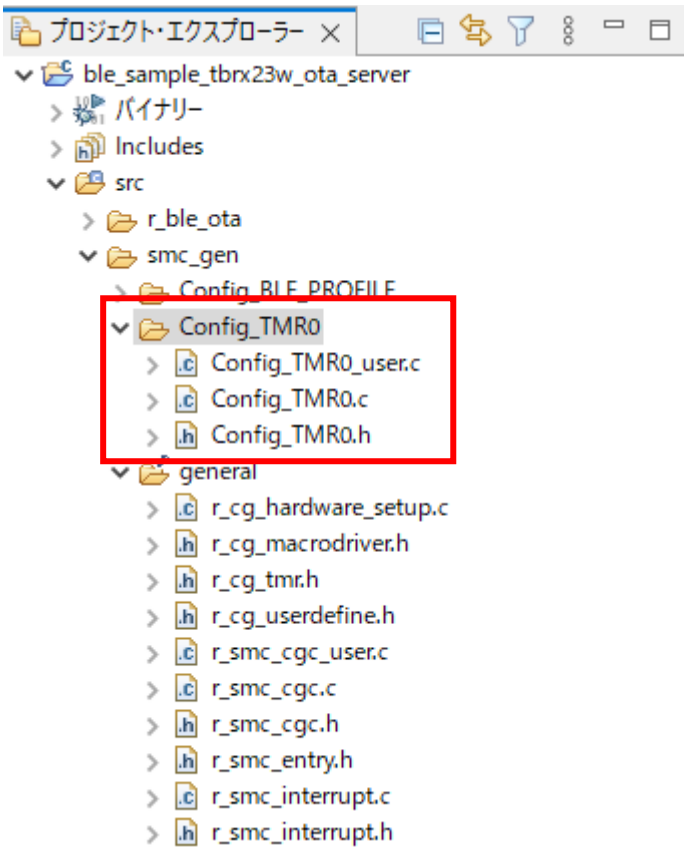

図 11.2 コード生成機能による生成フォルダ構成

まず最初に、ブートローダ実行中のアプリケーションセクション内の関数呼び出しを無効にします。general¥r_cg_hardware_setup.c ファイル内の R_Systeminit 関数内に生成されたコード生成機能部分のプログラムを無効にします。黄色ハイライト部分を追加します。

```
void R_Systeminit(void)
{
    /* Enable writing to registers related to operating modes, LPC, CGC, software
reset and LVD */
    SYSTEM.PRCR.WORD = 0xA50FU;

    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;

    /* Write 0 to the target bits in the POECR2 registers */
    POE.POECR2.BYTE = 0x00U;

    /* Initialize clocks settings */
    R_CGC_Create();

#if 0
    /* Set peripheral settings */
    R_Config_TMR0_Create();
#endif

    /* Register undefined interrupt */

R_BSP_InterruptWrite(BSP_INT_SRC_UNDEFINED_INTERRUPT,(bsp_int_cb_t)r_undefined_exc
eption);

    /* Disable writing to MPC pin function control registers */
    MPC.PWPR.BIT.PFSWE = 0U;
    MPC.PWPR.BIT.B0WI = 1U;

    /* Enable protection */
    SYSTEM.PRCR.WORD = 0xA500U;
}
```

図 11.3 コード生成機能の無効化

次に、ユーザアプリケーション用に R_Systeminit 関数と同様の機能を持つレジスタ設定用関数を app_main.c に実装します。ここでは、関数名を R_Systeminit_App とします。ここでは、app_main.c ファイルに実装します。追加する処理は以下の二つです。

- コンフィギュレーションヘッダファイルのインクルード
- R_Systeminit_App 関数の実装(R_CGC_Create 関数、R_BSP_InterruptWrite 関数の呼び出しは不要)

```
#include "Config_TMR0.h"

■■■
```

12. Appendix

12.1 OTA ファームウェア更新時間（参考値）

同梱したサンプルファームウェアの更新時間を示します。

バージョン	更新サイズ	対向の RX23W	iPhone 8 (iOS)	ZenFone 4 (Android)
Version1.10 →Version 1.11	64KB アプリケーションセクション：64KB	20 秒	29 秒	160 秒
Version1.10→Version 1.12	218KB アプリケーションセクション：64KB ダウンローダセクション　　：154KB	70 秒	97 秒	450 秒

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021.06.30	-	新規発行
1.10	2021.09.30	5	1.1 章に iOS による更新を追加しました。
		21	2.4 章 iOS 版 FWU_Client を追加しました。
		72	4.11.3 節に Service Changed Characteristic の Indication を追加しました。
		77	7 章 iOS 版 FWU_Client を追加しました
		79	8 章 Python 版 FWU_Client を追加しました。
		96	9 章 Version1.00 からの変更点を追加しました。
		97	10 章に制限事項を追加しました。
		103	12.1 章 OTA ファームウェアの更新時間に iOS 版の時間を追加しました。
1.11	2022.04.12	39	3.5 章に Error Response を返す場合の動作を追記しました。
		45	3.8 章にリロケータの電源遮断時の動作を追加しました。
		47	3.9 章にダウンローダの電源遮断時の動作を追加しました。
		53	ROM から RAM へマッピングするセクションの設定に、標準ライブラリセクション(SL_*)の記入漏れを修正しました。
		68	4.10 章をユーザアプリケーションの設定と 4.11 章 ユーザアプリケーションの機能追加に章分けしました。
		70	r_ble_ota_resets.c ファイルの実装に関する説明を追記しました。
		98	11 章 更新に失敗する場合の確認項目を追加しました。
		-	軽微な表現の修正

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、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。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

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

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

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因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、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、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.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

© 2021 Renesas Electronics Corporation. All rights reserved.