Bluetooth® Low Energy プロトコルスタック
ホスト MCU シンプル API for RL78/G14

R01AN3784JJ0100
Rev.1.00
2022.01.31

要旨

本アプリケーションノートで説明するシンプル API は、少ない手順ですぐに Bluetooth low energy 技術を用いた無線データ通信のプログラムができる API です。モデム構成の Host MCU(Renesas Starter Kit for RL78/G14)で動作し、RL78/G1D モジュール(RY7011)、またはモデム構成の BLE ソフトウェアが動作する RL78/G1D を制御します。ルネサスで提供する Bluetooth Low Energy プロトコルスタック(以降、BLE ソフトウェアと表記)と独自プロファイルを使用し、任意なフォーマットでデータ通信を行うことができます。

動作確認デバイス

Renesas Starter Kit for RL78/G14

関連資料

資料名	資料番号	
	和文	英文
Bluetooth Low Energy プロトコルスタック	-	-
ユーザーズマニュアル	R01UW0095J	R01UW0095E
API リファレンスマニュアル　基本編	R01UW0088J	R01UW0088E
rBLE コマンド仕様書	R01AN1376J	R01AN1376E
クイックスタートガイド	R01AN2767J	R01AN2767E
BLE 仮想 UART アプリケーション	R01AN3130J	R01AN3130E
RL78/G1D	-	-
ユーザーズマニュアル ハードウェア編	R01UH0515J	R01UH0515E
RL78/G1D 評価ボード	-	-
ユーザーズマニュアル	R30UZ0048J	R30UZ0048E
RL78/G1D モジュール	-	-
RL78/G1D モジュール(RY7011) ユーザーズマニュアル ハードウェア編	R02UH0004J	R02UH0004E
RL78/G14	-	-
ユーザーズマニュアル ハードウェア編	R01UH0186J	R01UH0186E
CS+ コード生成ツール 統合開発環境 ユーザーズマニュアル RL78 API リファレンス編[CS+ for CA,CX][CS+ for CC]	R20UT3102J	R20UT3102E
e² studio コード生成ツール 統合開発環境 ユーザーズマニュアル RL78 API リファレンス編	R20UT3127J	R20UT3127E
AP4, Applilet3 ユーザーズマニュアル RL78 API リファレンス編	R20UT3125J	R20UT3125E
Renesas Starter Kit for RL78/G14	-	-
ユーザーズマニュアル	R20UT0785J	R20UT0785E
チュートリアルマニュアル	R20UT0786J	R20UT0786E
クイックスタートガイド	R20UT0787J	R20UT0787E
CPU Board Schematics	-	R20UT0784E

目次

1. 概要 5

2. シンプル API 構成 6
2.1 システム構成 6
2.2 ソフトウェア構成 7

3. 開発環境 9
3.1 ビルド環境 9
3.2 ツール 9
3.3 デバイス 9
3.4 周辺機能 10
3.4.1 周辺機能一覧 10
3.4.2 コード生成ツール設定 11
3.5 フォルダ構成 13
3.5.1 シンプル API プログラムのフォルダ構成 13
3.5.2 汎用双方向通信データベース差分ファイルのフォルダ構成 17
3.5.3 実行ファイルのフォルダ構成 17

4. プログラムの実行 18
4.1 ローカルデバイスの準備 20
4.1.1 RSK へのシンプル API プログラムの書き込み 20
4.1.2 モジュール評価ボードの準備 20
4.2 リモートデバイスの準備 20
4.2.1 RL78/G1D 評価ボードへの仮想 UART アプリケーションの書き込み 20
4.2.2 スマートフォン・アプリケーションのインストール 20
4.3 実行手順 (RL78/G1D 評価ボード) 21
4.4 実行手順 (Android) 22
4.5 実行手順 (iOS) 24

5. シンプル API の使用方法 26
5.1 ランダムシード 27
5.2 初期化 27
5.3 デバイスアドレス・フィルタ設定 27
5.4 周辺デバイス検索 28
5.5 ビーコン送信 29
5.6 接続 29
5.7 データ通信 30

6. シンプル API 仕様 32
6.1 API 32
6.1.1 R_BLES_initialize 32
6.1.2 R_BLES_whitelist 33
6.1.3 R_BLES_scan 33
6.1.4 R_BLES_advertise 34
6.1.5 R_BLES_connect 36

6.1.6 R_BLES_get_event 37
6.1.7 R_BLES_send_data 38
6.1.8 R_BLES_receive_data 38
6.1.9 R_BLES_disconnect 38
6.2 構造体 39
6.2.1 RBLE_BROADCAST_ENABLE_PARAM 39
6.2.2 RBLE_ADV_INFO 39
6.2.3 RBLE_SET_ADV_PARAM 39
6.2.4 RBLE_BD_ADDR 39
6.2.5 RBLE_SET_ADV_DATA 39
6.2.6 RBLE_ADV_DATA 39
6.2.7 RBLE_SET_SCAN_RSP_DATA 39
6.2.8 RBLE_SCAN_RSP_DATA 39
6.2.9 RBLE_CREATE_CONNECT_PARAM 39
6.2.10 RBLE_CONNECT_INFO 40
6.2.11 RBLE_SCANNING_INFO 40
6.2.12 RBLE_SET_SCAN_PARAMETER 40
6.2.13 RBLE_WHITELIST 40
6.2.14 RBLE_WLIST_DEV_ADDR 40
6.2.15 RBLE_DEV_ADDR_INFO 40
6.3 マクロ 41
6.3.1 ADV_REPORT_LIST_NUM 41
6.3.2 WL_DEVADDR_LIST_NUM 41
6.3.3 RBLES_RDBUF_SIZE 41
6.4 ペアリング情報 41

7. 注意事項 42
7.1 コード生成 (r_cg_macrodriver.h) 42
7.2 シンプル API 呼び出し方法 42

8. Appendix 43
8.1 BLE ソフトウェアの変更 43
8.1.1 Host MCU プログラムの準備 43
8.1.2 Host MCU プログラムのビルド 45
8.1.3 BLE MCU プログラムの準備 47
8.1.4 BLE MCU プログラムのビルド 48
8.1.5 Host MCU－BLE MCU 接続 51

1. 概要

このアプリケーションノートは、シンプル API の構成、開発環境、プログラム実行方法、API の使用方法、API 仕様について記載します。プログラム実行方法では、シンプル API を使用しデータ通信を行うサンプルプログラム(以降、シンプル API プログラムと表記)で動作確認を行います。

シンプル API は、Bluetooth low energy (以降、BLE と表記)技術を用いた無線データ通信を少ない手順ですぐにプログラミングできるようにした 9 個の API を提供します。モデム構成の Host MCU(Renesas Starter Kit for RL78/G14)で動作し、RL78/G1D モジュール(RY7011)、またはモデム構成の BLE ソフトウェアが動作する RL78/G1D を制御します。ルネサス製独自プロファイルの汎用双方向通信(仮想 UART)[注]プロファイルを使用し、任意なフォーマットでデータ通信することができます。

シンプル API でプログラミングできる内容を以下に示します。

- 周辺の BLE デバイスを検索するスキャニング
- 周辺の BLE デバイスにビーコンを送信するアドバタイジング
- スレーブデバイスとしてリモートデバイスと接続
- マスターデバイスとしてリモートデバイスと接続 (接続可能リモートデバイス数: 1 台)
- 汎用双方向通信プロファイルを使用したデータ通信
  (Just works による暗号通信を自動で実行)
- デバイスアドレスを使ったホワイトリストによるフィルタリング
  (パブリックアドレス 6 台/ランダムアドレス 6 台 ： 合計 12 台)

API の概要を以下に示します。

1.	R_BLES_initialize()	シンプル API の初期化と BLE MCU との通信の初期化
2.	R_BLES_whitelist()	ホワイトリストにデバイスアドレスを設定
3.	R_BLES_scan()	スキャニングの実行
4.	R_BLES_advertise()	アドバタイジングの実行とマスターデバイスとの接続
5.	R_BLES_connect()	スレーブデバイスとの接続
6.	R_BLES_get_event()	イベントの取得
7.	R_BLES_send_data()	データ送信
8.	R_BLES_receive_data()	データ受信
9.	R_BLES_disconnect()	リモートデバイスとの接続を切断

【注】 汎用双方向通信と仮想 UART は同一のプロファイルです。モデム構成の場合は「汎用双方向通信プロファイル」、組み込み構成の場合は「仮想 UART プロファイル」と表記します。プロファイルの詳細は「BLE 仮想 UART アプリケーション(R01AN3130)」を参照してください。

2. シンプル API 構成

2.1 システム構成

シンプル API プログラムが動作するシステム構成図を以下に示します。「図 2-1 システム構成(1) (RY7011)」は、ローカルデバイスとしてシンプル API が動作する Host MCU に Renesas Starter Kit for RL78/G14(以降、RSK と表記)を使用し、BLE MCU としてモジュール評価ボード(RM-110-RFB-2)を使用したモデム構成です。UART 2 線分岐接続方式のデフォルトボーレートは 115200bps です。リモートデバイスは RL78/G1D 評価ボード(RTK0EN0001D01001BZ)で仮想 UART アプリケーション(R01AN3130)を使用します。

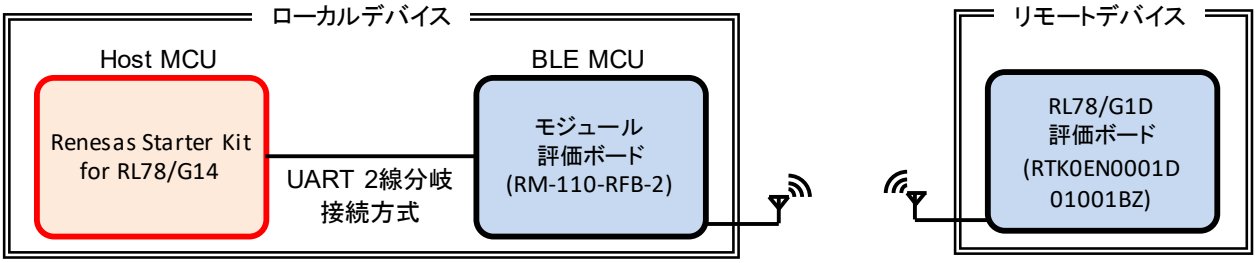

図 2-1 システム構成(1) (RY7011)

「図 2-2 システム構成(2) (BLE Evaluation Board)」は、ローカルデバイスの BLE MCU に RL78/G1D 評価ボード(RTK0EN0001D01001BZ)で Modem 構成の BLE ソフトウェアを使用した構成です。UART 2 線方式のデフォルトボーレートは 4800bps です。BLE ソフトウェアを使用する場合、BLE 通信を行うために汎用双方向通信プロファイルを組み込む必要があります。組み込み方法は「8.1 BLE ソフトウェアの変更」を参照して下さい。

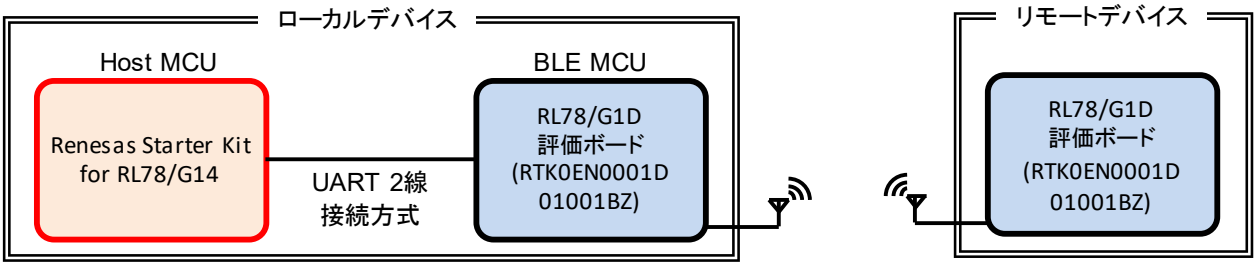

図 2-2 システム構成(2) (BLE Evaluation Board)

また、リモートデバイスにスマートフォン(Android または、iOS)を使用することもできます。本アプリケーションノートでは、スマートフォンとの接続方法についても説明します。

2.2　ソフトウェア構成

Host MCU である RL78/G14 と BLE MCU である RL78/G1D のソフトウェア構成図を示します。

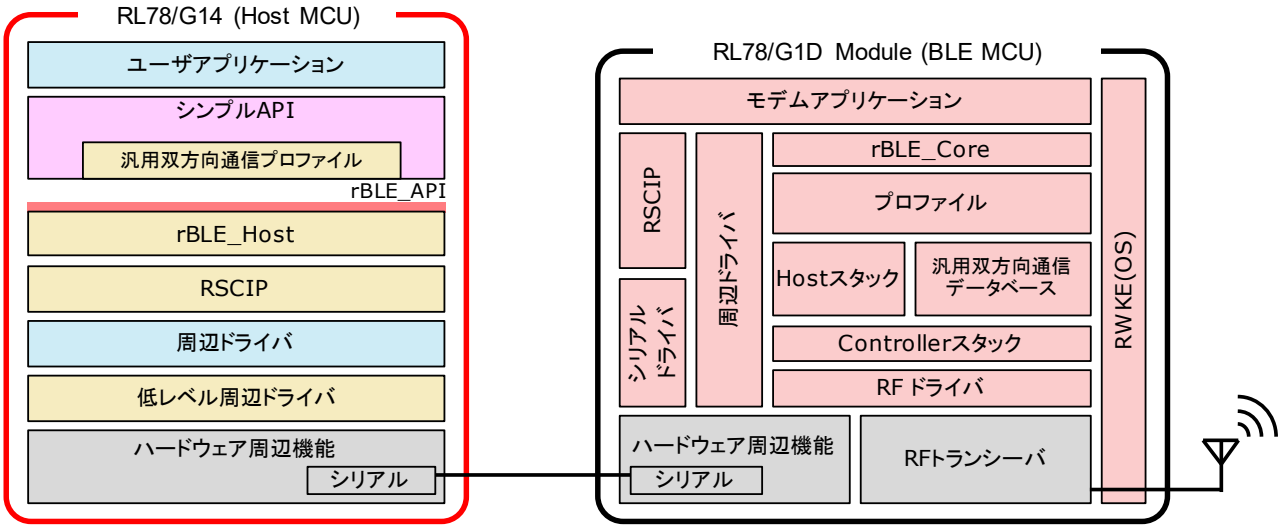

図 2-3 ソフトウェア構成

Host MCU は、MCU 周辺機能の制御と BLE MCU との通信を実行するための低レベル周辺ドライバ、周辺ドライバ、RSCIP（Renesas Serial Communication Interface Protocol）と、rBLE API をアプリケーションに提供するための rBLE_Host と GATT API を使用した汎用双方向通信プロファイル、システムを制御するためのユーザアプリケーションから構成されます。シンプル API は、rBLE API の呼び出しやスケジューリングを API 内部で実行し、ユーザアプリケーションにシンプルな API を提供します。

低レベル周辺ドライバは、コード生成ツールが自動生成します。RSCIP、rBLE_Host は BLE ソフトウェアに含まれており、ソースコードが提供されます。ソフトウェア開発時は、最新の BLE ソフトウェアのソースコードをご使用ください。

表 2-1　Host MCU ソフトウェア構成

ソフトウェア	機能	ソフトウェア開発の要否
ユーザアプリケーション (シンプル API プログラム)	シンプル API の初期化 周辺ドライバの初期化	必要
シンプル API	シンプルな API の提供 rBLE API の呼び出しやスケジューリング	不要 (ソースコード提供) 注 3
rBLE_Host	rBLE API 提供 イベントコールバックの実行	不要 (ソースコード提供) 注 1
汎用双方向通信プロファイル	GATT API を使用した独自プロファイル	不要 (ソースコード提供) 注 3
RSCIP	シリアル通信プロトコルの制御	不要 (ソースコード提供) 注 1
周辺ドライバ	Host MCU 周辺機能の制御	必要
低レベル周辺ドライバ	Host MCU 周辺機能のプリミティブな制御	不要 (ツール自動生成) 注 2

【注】 1. ソースコードは BLE ソフトウェアが提供。
2. ソースコードはコード生成ツールが自動生成。
3. ソースコードは本アプリケーションノートのソースコードで提供。

BLE MCU のソフトウェアは、RF トランシーバを制御するための RF ドライバ、汎用双方向通信プロファイルの GATT Database、Host/Controller スタック、プロファイル、rBLE_Core と、Host MCU と通信するためのシリアル通信ドライバ、RSCIP と、システムを制御するための RWKE (Renesas Wireless Kernel Extension)、モデムアプリケーションで構成されます。これらは BLE ソフトウェアとしてビルド環境が提供されます。

表 2-2　BLE MCU ソフトウェア構成

ソフトウェア	機能
モデムアプリケーション	RSCIP と rBLE の制御
RWKE	システム全体のスケジューリングとメモリ資源の管理
RSCIP	シリアル通信プロトコルの制御
周辺ドライバ/シリアルドライバ	BLE MCU 周辺機能の制御
rBLE_Core	rBLE_API 提供
プロファイル	プロファイル機能の提供
Host スタック	GAP、GATT、SM、L2CAP 機能の提供
汎用双方向通信データベース	汎用双方向通信プロファイルの GATT Database
Controller スタック	Link Layer 機能の提供

3. 開発環境

シンプル API のビルドと動作確認で使用する環境を示します。

3.1 ビルド環境

- ホストマシン
  - Windows® 7 以降
  - PC/AT™ 互換機
  - プロセッサ：1.6GHz 以上
  - メイン・メモリ：1G バイト以上
  - ディスプレイ：1024×768 以上の解像度，65536 色以上
  - インタフェース：USB2.0（E1 および USB-シリアル変換ケーブル）
- 統合開発環境

  下記の統合開発環境に対応しています。いずれか一つをご使用ください。
  - Renesas CS+ for CA,CX V4.00.00 / Renesas CA78K0R V1.72
  - Renesas CS+ for CC V5.00.00 / RL78 コンパイラ CC-RL V1.04.00
  - e² studio V5.3.1.002 / RL78 コンパイラ CC-RL V1.04.00

3.2 ツール

- オンチップデバッギングエミュレータ&フラッシュプログラマ
  - Renesas オンチップデバッギングエミュレータ E1
- フラッシュ書き込みソフトウェア
  - Renesas Flash Programmer V3.02.01
- コード生成ツール
  - Applilet3 (3.05.00.01)

3.3 デバイス

- 評価ボード
  - Renesas Starter Kit for RL78/G14
  - 内藤電誠町田製作所製モジュール評価ボード (RM-110-RFB-2)
  - ルネサスエレクトロニクス製 RL78/G1D 評価ボード (RTK0EN0001D01001BZ)
  - Smart Phone (Android または、iOS)

3.4 周辺機能

3.4.1 周辺機能一覧

シンプル API プログラムで使用する RL78/G14 の周辺機能や RSK に実装されている周辺装置を以下に示します。

表 3-1 周辺機能

周辺機能/周辺装置	用途	必要性
シリアル・インタフェース (シリアル・アレイ・ユニット)注 1	UART2 線接続方式 (デフォルトボーレート 4800 bps) UART2 線分岐接続方式 (デフォルトボーレート 1152000 bps)	必須 注 2
インターバル・タイマ (12 ビット・インターバル・タイマ) 注 1	UART タイムアウト監視	必須 注 2
LED (ポート機能) 注 1	シンプル API プログラムの状態表示 LED0：P43 LED1：P44	任意
スイッチ (割り込み機能) 注 1	シンプル API プログラムの操作 SW1：INTP8 SW2：INTP9 SW3：INTP10	任意

【注 1】 周辺機能列の下段は RL78/G14 の周辺機能名です。

【注 2】 Host MCU が rBLE を使用するために最低限必要とする周辺機能を「必須」、その他を「任意」とする。

3.4.2 コード生成ツール設定

「図 2-1 システム構成(1) (RY7011)」で使用するコード生成ツールのデフォルト設定を以下に示します。「図 2-2 システム構成(2) (BLE Evaluation Board)」で使用する場合は、コード生成ツールで設定を変更しプログラムをビルドする必要があります。実施方法は「8.1.1 Host MCU プログラムの準備」、「8.1.2 Host MCU プログラムのビルド」を参照してください。

表 3-2　コード生成ツール設定

周辺機能		用途
クロック発生回路 クロック設定	動作モード設定	高速メイン・モード 2.7(V)≦VDD≦5.5(V)
	EVDD 設定	2.7(V)≦EVDD≦5.5(V)
	メイン・システム・クロック(fMAIN)設定	高速オンチップオシレータクロック(fIH)
	高速オンチップオシレータクロック設定	24(MHz)
	高速システムクロック設定	動作 X1 発振(f_X) 周波数 20(MHz) 発振安定時間 6553.6(us)
	サブシステム・クロック(fSUB)設定	動作 XT1 発振(fXT) 周波数 32.768(kHz) XT1 発振回路の発振モード選択 低消費発振 STOP,HALT モード時のクロック供給設定 供給許可
	低速内蔵発振クロック(fIL)設定	周波数 15(kHz)
	RTC,インターバル・タイマ動作クロック	32.768(f_{SUB})(kHz)
	CPU と周辺クロック設定	CPU と周辺クロック(f_{CLK})24000(fIH)(kHz)
ポート ポート 4	P43	出力 ：1
	P44	出力 ：1
割り込み 外部割り込み	INTP8	立下りエッジ 低優先
	INTP9	立下りエッジ 低優先
	INTP10	立下りエッジ 低優先
シリアル SAU1 UART2 受信	データ・ビット長設定	8 ビット
	データ転送方向設定	LSB
	パリティ設定	パリティなし
	ストップ・ビット長設定	1 ビット固定
	受信データ・レベル設定	標準
	転送レート設定	ボーレート 115200 (bps)
	割り込み設定	受信割り込み設定(INTSR2) 高
	コールバック機能設定	受信完了

		エラー
シリアル SAU1 UART2 送信	転送モード設定	単発モード
	データ・ビット長設定	8 ビット
	データ転送方向設定	LSB
	パリティ設定	パリティなし
	ストップ・ビット長設定	1 ビット
	送信データ・レベル設定	標準
	転送レート設定	ボーレート 115200 (bps)
	割り込み設定	送信割り込み設定(INTST2) 低
	コールバック機能設定	送信完了
インターバル・タイマ	インターバル・タイマ動作設定	使用する
	インターバル時間設定	10ms
	割り込み設定	インターバル信号検出(INTIT) 使用 低優先

3.5 フォルダ構成

3.5.1 シンプル API プログラムのフォルダ構成

(R)は BLE ソフトウェアに含まれているファイルであることを示します。ソフトウェア開発時は、BLE ソフトウェアが提供する最新のコードファイルをご使用ください。

```
ble_simple_api_rl78g14
 |
 ├──project
 |   ├──CS_CA                                          CS+ for CA, CX プロジェクト・フォルダ
 |   |   |   ble_simple_api_rl78g14.mtpj
 |   |   |
 |   |   └──cg                                         コード生成ファイル・フォルダ
 |   |           r_cg_cgc.c                            クロック生成ドライバ・コードファイル
 |   |           r_cg_cgc.h                            クロック生成ドライバ・ヘッダファイル
 |   |           r_cg_cgc_user.c                       クロック生成ドライバ・ユーザコードファイル
 |   |           r_cg_intc.c                           外部割り込みドライバ・コードファイル
 |   |           r_cg_intc.h                           外部割り込みドライバ・ヘッダファイル
 |   |           r_cg_intc_user.c                      外部割り込みドライバ・ユーザコードファイル
 |   |           r_cg_it.c                             インターバルタイマドライバ・コードファイル
 |   |           r_cg_it.h                             インターバルタイマドライバ・ヘッダファイル
 |   |           r_cg_it_user.c                        インターバルタイマドライバ・ユーザコードファイル
 |   |           r_cg_macrodriver.h                    汎用マクロ・ヘッダファイル
 |   |           r_cg_port.c                           ポートドライバ・コードファイル
 |   |           r_cg_port.h                           ポートドライバ・ヘッダファイル
 |   |           r_cg_port_user.c                      ポートドライバ・ユーザコードファイル
 |   |           r_cg_serial.c                         シリアルドライバ・コードファイル
 |   |           r_cg_serial.h                         シリアルドライバ・ヘッダファイル
 |   |           r_cg_serial_user.c                    シリアルドライバ・ユーザコードファイル
 |   |           r_cg_userdefine.h                     ユーザ定義マクロ・ヘッダファイル
 |   |           r_main.c                              メインループ・コードファイル
 |   |           r_systeminit.c                        周辺機能初期化・コードファイル
 |   |
 |   ├──CS_CCRL                                        CS+ for CC プロジェクト・フォルダ
 |   |   |   ble_simple_api_rl78g14.mtpj
 |   |   |   ble_simple_api_rl78g14.rcpe
 |   |   |   cstart.asm
 |   |   |   hdwinit.asm
 |   |   |   iodefine.h
 |   |   |   stkinit.asm
 |   |   |
 |   |   └──cg                                         コード生成ファイル・フォルダ
 |   |           r_cg_cgc.c
 |   |           r_cg_cgc.h
 |   |           r_cg_cgc_user.c
 |   |           r_cg_intc.c
 |   |           r_cg_intc.h
 |   |           r_cg_intc_user.c
 |   |           r_cg_it.c
 |   |           r_cg_it.h
 |   |           r_cg_it_user.c
 |   |           r_cg_macrodriver.h                    汎用マクロ・ヘッダファイル
 |   |                                                 (「7.1 コード生成 (r_cg_macrodriver.h)」にしたがって変更を加えています。)
 |   |           r_cg_macrodriver_g14_ccrl.h           汎用マクロ・ヘッダファイル(修正用ファイル)
 |   |           r_cg_port.c
 |   |           r_cg_port.h
```

```
│   │           r_cg_port_user.c
│   │           r_cg_serial.c
│   │           r_cg_serial.h
│   │           r_cg_serial_user.c
│   │           r_cg_userdefine.h
│   │           r_main.c
│   │           r_systeminit.c
│   │
│   └──e2studio                                   e2 studio V5 プロジェクト・フォルダ
│       │   .cproject
│       │   .HardwareDebuglinker
│       │   .info
│       │   .project
│       │   ble_simple_api_rl78g14 HardwareDebug.launch
│       │
│       ├──.settings
│       │   │   conflictresources.xml
│       │   │   Dependency_Scan_Preferences.prefs
│       │   │   org.eclipse.cdt.managedbuilder.core.prefs
│       │   │   Project_Generation_Prefrences.prefs
│       │   │
│       │   └──CodeGenerator
│       │           cgproject.cgp
│       │           cgprojectDatas.datas
│       │
│       └──src                                    コード生成ファイル・フォルダ
│           cstart.asm
│           iodefine.h
│           r_cg_cgc.c
│           r_cg_cgc.h
│           r_cg_cgc_user.c
│           r_cg_intc.c
│           r_cg_intc.h
│           r_cg_intc_user.c
│           r_cg_it.c
│           r_cg_it.h
│           r_cg_it_user.c
│           r_cg_macrodriver.h                   汎用マクロ・ヘッダファイル
│                                                (「7.1 コード生成 (r_cg_macrodriver.h)」にしたがっ
│                                                て変更を加えています。)
│           r_cg_macrodriver_g14_ccrl.h          r_cg_macrodriver.h 変更用ファイル
│           r_cg_port.c
│           r_cg_port.h
│           r_cg_port_user.c
│           r_cg_serial.c
│           r_cg_serial.h
│           r_cg_serial_user.c
│           r_cg_userdefine.h
│           r_main.c
│           r_systeminit.c
│           stkinit.asm
│
│
└──src
    │   ble_simple_api_rl78g14.c                 シンプル API プログラム
    │
    ├──platform
    │   └──rl78g14
    │       ├──driver
```

```
│   │   ├──plf
│   │   │     plf.c                          プラットフォームドライバ・コードファイル
│   │   │     plf.h                          プラットフォームドライバ・ヘッダファイル
│   │   │
│   │   ├──serial
│   │   │     uart.c                         UART ドライバ・コードファイル
│   │   │     uart.h                         UART ドライバ・ヘッダファイル
│   │   │
│   │   └──timer
│   │         timer.c                        タイマドライバ・コードファイル
│   │         timer.h                        タイマドライバ・ヘッダファイル
│   │
│   └──include
│         arch.h                     (R)    アーキテクチャ・ヘッダファイル
│         compiler.h                 (R)    コンパイラ・ヘッダファイル
│         iodefine.h                        レジスタアクセス用マクロ定義・ヘッダファイル
│         ll.h                       (R)    低レベルマクロ・ヘッダファイル
│         rscip_api.h                (R)    RSCIP コールバック・ヘッダファイル
│         rskrl78g14def.h                   RSK ヘッダファイル
│         types.h                    (R)    タイプ定義・ヘッダファイル
│
└──rBLE
    └──src
        ├──host
        │   rble_host.c                (R)    rBLE_Host・コードファイル
        │   rble_if_api_cb.c           (R)    rBLE API コールバック・コードファイル
        │
        ├──gap
        │     rble_api_gap.c            (R)    GAP API・コードファイル
        │
        ├──gatt
        │     rble_api_gatt.c           (R)    GATT API・コードファイル
        │
        ├──sm
        │     rble_api_sm.c             (R)    SM API・コードファイル
        │
        └──vs
              rble_api_vs.c             (R)    VS API・コードファイル
        │
        ├──include
        │   prf_sel.h                   (R)    プロファイル選択・ヘッダファイル
        │   rble.h                      (R)    rBLE マクロ定義・ヘッダファイル
        │   rble_api.h                  (R)    rBLE API・ヘッダファイル
        │   rble_api_custom.h           (R)    rBLE SCP API・ヘッダファイル
        │   rble_trans.h                (R)    rBLE 通信・ヘッダファイル
        │
        └──host
              rble_host.h               (R)    rBLE_Host・ヘッダファイル
        │
        ├──rbles_api
        │   rbles_api.c                         シンプル API コードファイル
        │   rbles_api.h                         シンプル API ヘッダファイル
        │
        ├──rscip
        │   rscip.c                      (R)    RSCIP・コードファイル
        │   rscip.h                      (R)    RSCIP・ヘッダファイル
        │   rscip_cntl.c                 (R)    RSCIP コントロール・コードファイル
        │   rscip_cntl.h                 (R)    RSCIP コントロール・ヘッダファイル
        │   rscip_ext.h                  (R)    RSCIP 外部コールバック・ヘッダファイル
```

```
│       rscip_uart.c                    (R)   RSCIP シリアル制御・コードファイル
│       rscip_uart.h                    (R)   RSCIP シリアル制御・ヘッダファイル
│
└──sample_profile
    │   db_handle.h                     (R)   データベースハンドル・ヘッダファイル
    │
    └──vuart
            vuart.h                            汎用双方向通信ヘッダファイル
            vuartc.c                           汎用双方向通信クライアント・コードファイル
            vuartc.h                           汎用双方向通信クライアント・ヘッダファイル
            vuarts.c                           汎用双方向通信サーバ・コードファイル
            vuarts.h                           汎用双方向通信サーバ・ヘッダファイル
```

3.5.2 汎用双方向通信データベース差分ファイルのフォルダ構成

BLE ソフトウェアに追加する汎用双方向通信データベース差分ファイルのフォルダ構成を以下に示します。

```
BLE_Software_Ver_1_20_diff_vuart
 │
 └──RL78_G1D
      └──Project_Source
           ├──rBLE
           │    └──src
           │         └──include
           │                 rble_api_vuart.h          汎用双方向通信ヘッダファイル
           │
           └──renesas
                └──src
                     └──arch
                          └──rl78
                                db_handle.h            汎用双方向通信 Attribute database handles 定義
                                prf_config.c           汎用双方向通信 Attribute database index 定義
                                prf_config.h           汎用双方向通信サービ定義
```

3.5.3 実行ファイルのフォルダ構成

実行ファイルのフォルダ構成を以下に示します。

```
ROM_File
 │
 ├──BLE_MCU
 │      rBLE_Mdm_CCRL.hex                BLE ソフトウェアに汎用双方向通信データベースを追加し
 │                                       た、RL78/G1D 評価ボードで動作する実行ファイル。
 │                                       (UART 2 線接続方式(4800 bps))
 │                                       本実行ファイルの作成方法は「8.1」を参照して下さい。
 │
 └──Host_MCU
        ble_simple_api_rl78g14_2wire_4200.hex   RSK で動作するシンプル API プログラム
                                                (UART 2 線接続方式(4800 bps))
                                                本実行ファイルの作成方法は「8.1」を参照して下さい。
        ble_simple_api_rl78g14_div2wire_115200.hex  RSK で動作するシンプル API プログラム
                                                (UART 2 線分岐接続方式(115200 bps))
```

4. プログラムの実行

本章では、「図 2 1 システム構成(1) (RY7011)」の BLE MCU にモジュール評価ボードを使用する構成を使用して、シンプル API プログラムの実行方法について説明します。

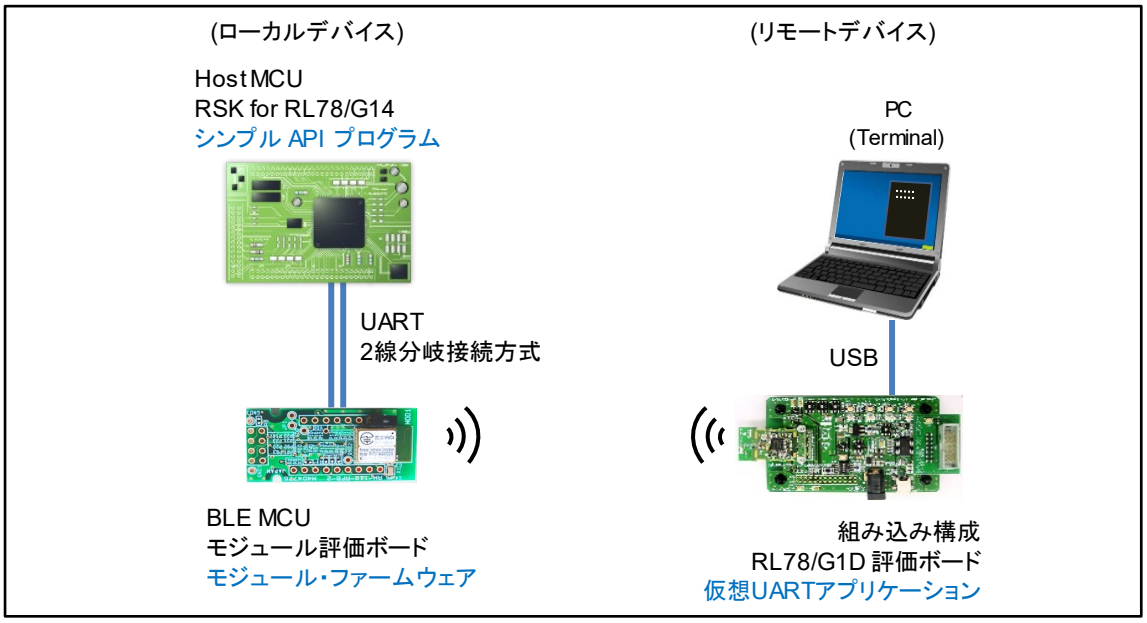

図 4-1　動作確認環境 (1)

また、リモートデバイスにスマートフォンを使用した場合の動作確認環境を以下に示します。

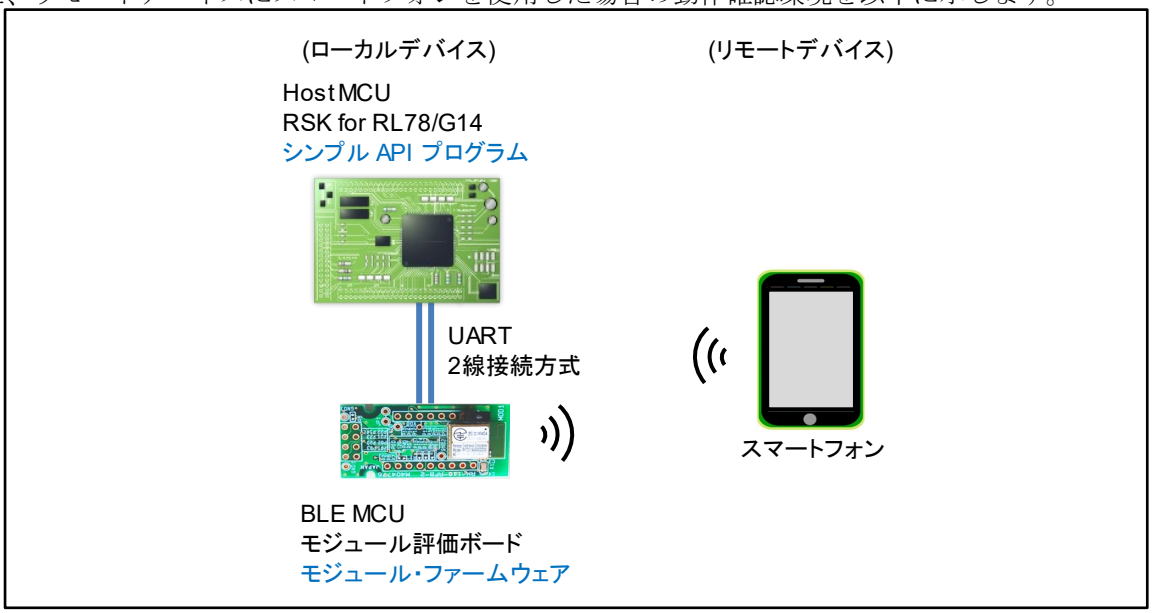

図 4-2　動作確認環境 (2)

「図 4-1　動作確認環境 (1)」、「図 4-2　動作確認環境 (2)」のローカルデバイスで使用できる実行ファイルを以下に示します。リモートデバイスについては「4.2 リモートデバイスの準備」を参照してください。

表 4-1　実行ファイル (RSK+モジュール評価ボード)

ローカルデバイス	実行ファイル	説明
RSK	ble_simple_api_rl78g14_div2wire_115200.hex	Host MCU で動作するシンプル API プログラム ・UART 2 線分岐接続方式 ・ボーレート 115200 bps
モジュール評価ボード	-	BLE MCU で動作するプログラム モジュール評価ボードに書き込まれているファームウェアを使用。

「図 2-2 システム構成(2) (BLE Evaluation Board)」で示す、BLE MCU に RL78/G1D 評価ボードを使用する構成の場合、Host MCU と BLE MCU のプログラムを準備する必要があります。「8.1 BLE ソフトウェアの変更」を参照してプログラムを準備し、使用するリモートデバイスに合わせて、RL78/G1D 評価ボードの場合は「4.3」、スマートフォン(Android)の場合は「4.4」、スマートフォン(iOS)の場合は「4.5」を実行してください。以下に示す、本構成のローカルデバイスで使用できる実行ファイルを本アプリケーションノートに同梱しています。「8.1.2(4)実行ファイルの書き込みとボードの設定」、「8.1.4(4)実行ファイルの書き込みとボードの設定」を参照して実行ファイルをローカルデバイスに書き込んでください。

表 4-2　実行ファイル (RSK+RL78/G1D 評価ボード)

ローカルデバイス	実行ファイル	説明
RSK	ble_simple_api_rl78g14_2wire_4200.hex	Host MCU で動作するシンプル API プログラム 注 1 ・UART 2 線接続方式 ・ボーレート 4800 bps
RL78/G1D 評価ボード	rBLE_Mdm_CCRL.hex	BLE MCU で動作するプログラム BLE ソフトウェアに汎用双方向通信プロファイルのデータベースを組み込んだ実行ファイル注 2

【注】 1. 「表 8-1　Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」で示される各統合開発環境の実行ファイルも使用できます。
2. 「表 8-4 BLE MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」で示される各統合開発環境の実行ファイルも使用できます。

4.1 ローカルデバイスの準備

4.1.1 RSK へのシンプル API プログラムの書き込み

"ble_simple_api_rl78g14_div2wire_115200.hex"ファイルを Host MCU の RSK へ書き込みます。「8.1.2(4)実行ファイルの書き込みとボードの設定」を参照して書き込んでください。

4.1.2 モジュール評価ボードの準備

モジュール評価ボードにはモジュールファームウェアが書き込まれています。「8.1.5(1)RSK とモジュール評価ボードの接続」を参照し Host MCU と接続してください。

4.2 リモートデバイスの準備

使用するリモートデバイスに応じて、「4.2.1」または「4.2.2」のいずれかを実施してください。

4.2.1 RL78/G1D 評価ボードへの仮想 UART アプリケーションの書き込み

仮想 UART アプリケーションを組み込み構成の RL78/G1D 評価ボードに書き込みます。アプリケーションノート「仮想 UART アプリケーション (R01AN3130)」を下記 URL からダウンロードしてください。

https://www.renesas.com/document/scd/bluetooth-low-energy-protocol-stack-ble-virtual-uart-application

「7.2 ビルド手順」を参照してプログラムを RL78/G1D 評価ボードに書き込んでください。

4.2.2 スマートフォン・アプリケーションのインストール

リモートデバイスとする Android デバイスまたは iOS デバイスに GATTBrowser をインストールしてください。

- (Android デバイス) “GATTBrowser” - Renesas Electronics

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

- (iOS デバイス) “GATTBrowser” - Renesas Electronics

https://itunes.apple.com/jp/app/gattbrowser/id1163057977?mt=8

4.3 実行手順 (RL78/G1D 評価ボード)

「図 4-1 動作確認環境 (1)」での実行手順を示します。

1. ローカルデバイスとリモートデバイスに電源を供給し、リモートデバイスに接続した PC でターミナルを起動します。
2. RSK 上の LED0(緑)が点灯します。
3. ターミナル上で簡易 AT コマンドモードであることを確認します。(エスケープキーの押下で簡易 AT コマンドモードと仮想 UART モードがトグルします。)
4. ターミナル上で「AT-C=(ローカルデバイスのデバイスアドレス)」を実行します。
5. ローカルデバイスとリモートデバイスが接続し、ターミナル上に「CONNECT」が表示されます。
6. RSK 上の LED1 (橙)が点灯します。
7. エスケープキーを押下し、仮想 UART モードにします。
8. RSK 上の SW2 を押します。ターミナル上に"Hello"と表示されます。
9. エスケープキーを押下し、簡易 AT モードにします。
10. RSK 上の SW3 を押します。ローカルデバイスとリモートデバイスが切断し、ターミナル上に「DISCONNECT」が表示されます。
11. RSK 上の LED1 (橙)が消灯します。

下図、赤色の数字は実行手順の番号を示します。

```
[Virtual UART Mode]

[AT Command Mode]          3.
AT-C=777777770000          4.

OK

CONNECT                    5.

[Virtual UART Mode]        7.
Hello                      8.
[AT Command Mode]          9.

DISCONNECT                 10.
```

図 4-3 ターミナル表示

4.4 実行手順 (Android)

「図 4-2 動作確認環境 (2)」で Android デバイスを使用する場合の実行手順を示します。

1. ローカルデバイスに電源を供給し、スマートフォンで GATTBrowser を起動します。
2. RSK 上の LED0(緑)が点灯します。
3. デバイスの検索結果から"RSK-RL78/G14"と表示されたデバイス名の右端にある丸矢印をタップし接続します。
   (図 A1 - 矢印(1))
4. Renesas Virtual UART Service の"Indication Characteristic"をタップします。
   (図 A2 - 矢印(2))
5. "Indication Off"をタップし"Indication On"にします。
   (図 A3 - 矢印(3))
6. ローカルデバイスとリモートデバイスが接続し、RSK 上の LED1 (橙)が点灯します。
7. "HEX"から"String"に変更します。
   (図 A4 - 矢印(4))
8. RSK 上の SW2 を押すと、"Hello"と表示されます。
   (図 A4 - 矢印(5))
9. RSK 上の SW3 を押します。
10. ローカルデバイスとリモートデバイスが切断し、「Disconnected」が表示されます。
    (図 A5 - 矢印(6))
11. RSK 上の LED1 (橙)が消灯します。

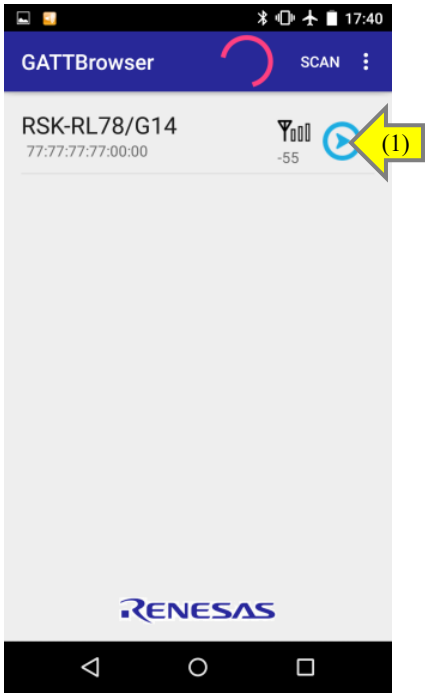

図 A1

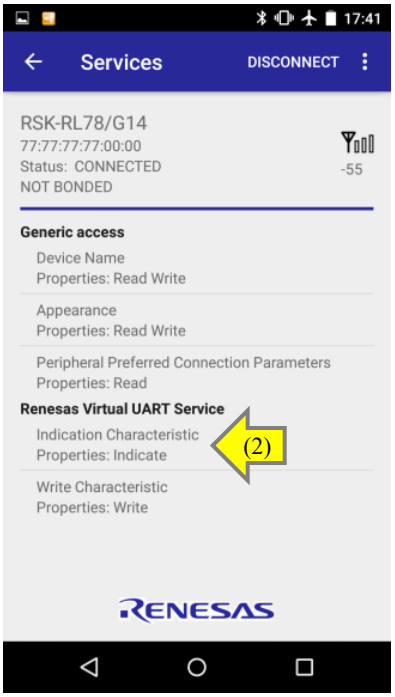

図 A2

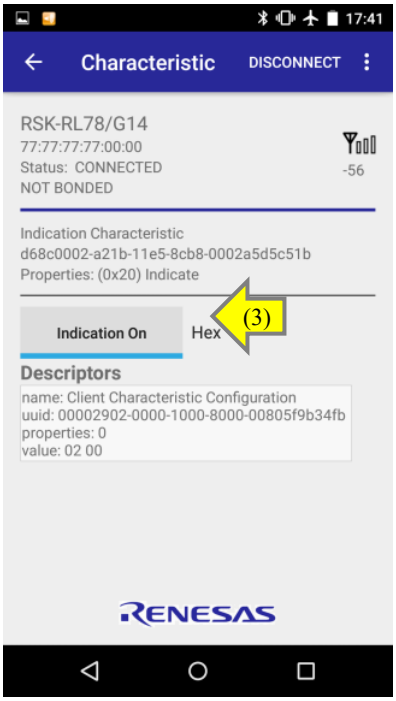

図 A3

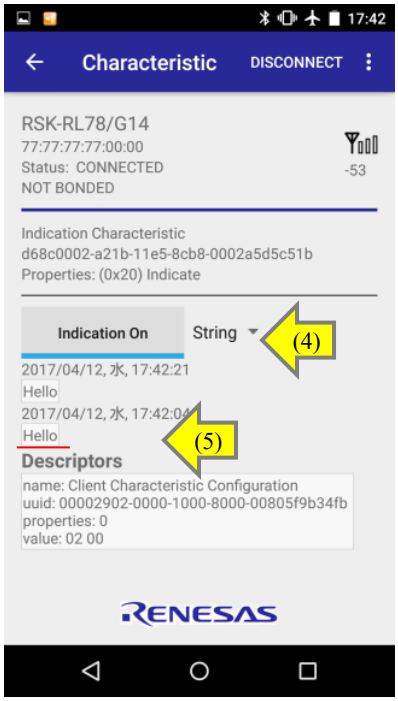

図 A4

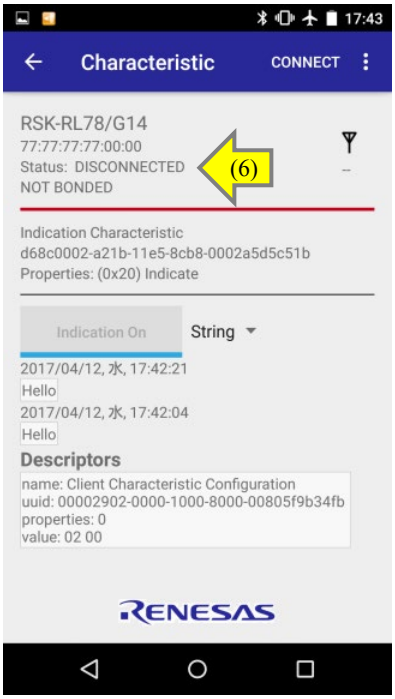

図 A5

4.5 実行手順 (iOS)

「図 4-2　動作確認環境 (2)」で iOS デバイスを使用する場合の実行手順を示します。

1. ローカルデバイスに電源を供給し、スマートフォンで GATTBrowser を起動します。
2. RSK 上の LED0(緑)が点灯します。
3. デバイスの検索結果から"Renesas-BLE"注 と表示されたデバイス名をタップします。
   (図 i1 - 矢印(1))
   【注】iOS の場合、OS に情報がキャッシュされるため別名が表示される場合があります。
4. "LocalName:RSK-RL78/G14"と表示されることを確認します。
   (図 i2 - 矢印(2))
5. 右端の丸矢印をタップし接続します。
   (図 i2 - 矢印(3))
6. Service 一覧の"Indication Characteristic"を選択します
   (図 i3 - 矢印(4))
7. "Enable Indication"をタップします。
   (図 i4 - 矢印(5))
8. ローカルデバイスとリモートデバイスが接続し、RSK 上の LED1 (橙)が点灯します
9. RSK 上の SW2 を押します。
10. "Hello"と表示されます。
    (図 i5 - 矢印(6))
11. RSK 上の SW3 を押します。
12. ローカルデバイスとリモートデバイスが切断し、「Disconnected」が表示されます
    (図 i6 - 矢印(7))
13. RSK 上の LED1 (橙)が消灯します

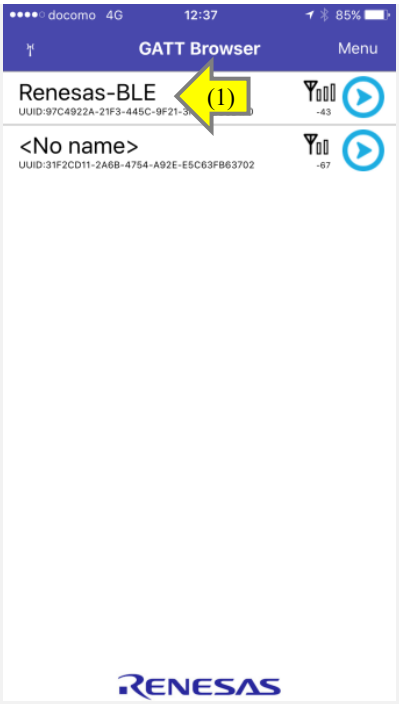

図 i1

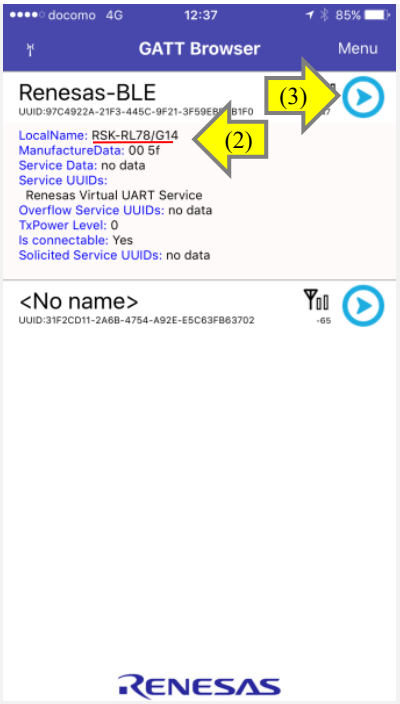

図 i2

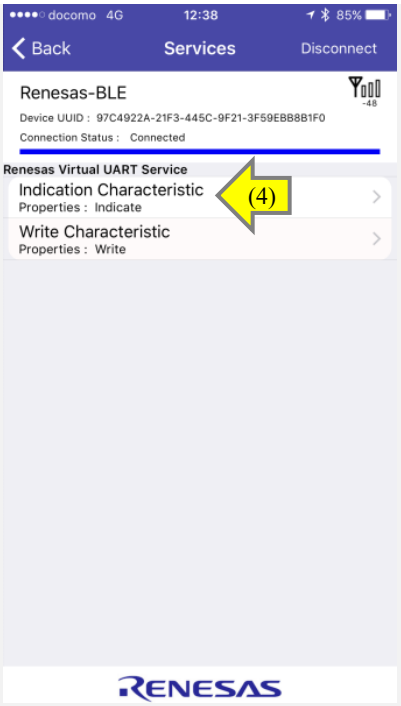

図 i3

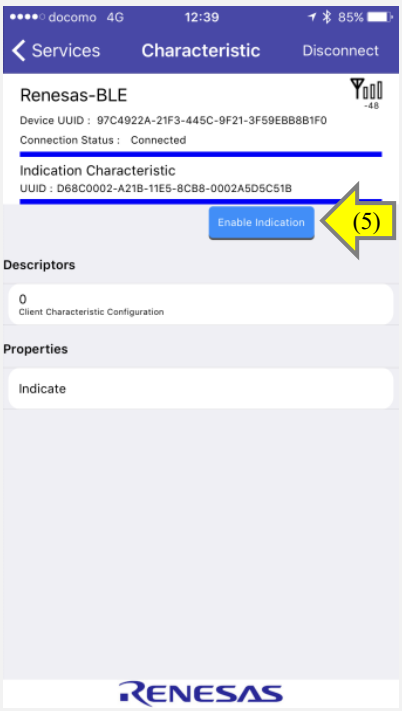

図 i4

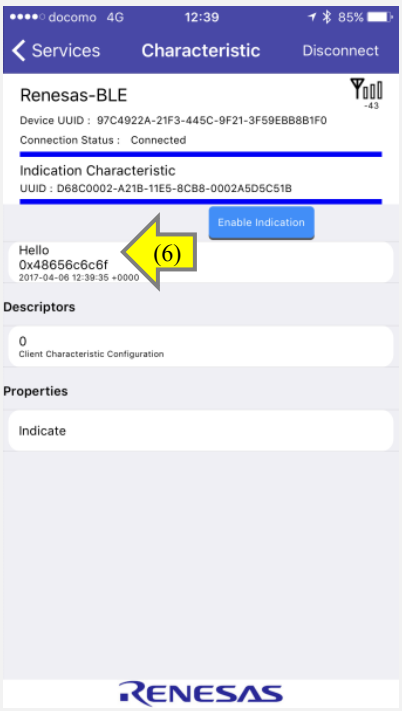

図 i5

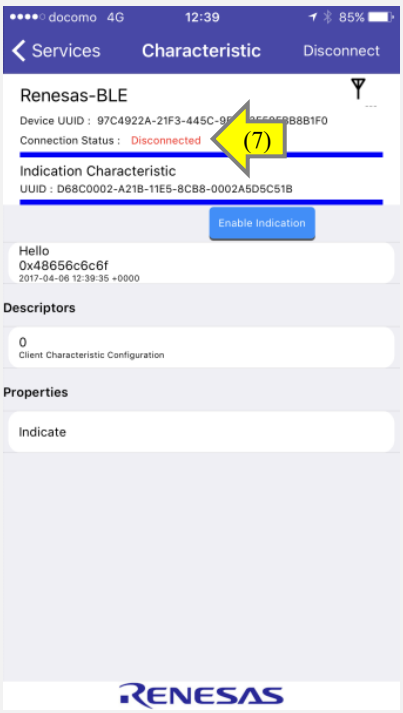

図 i6

5. シンプル API の使用方法

ここでは、シンプル API を使用して BLE 通信を行う方法を、シンプル API プログラムを例に説明します。シンプル API プログラムの main 関数は下記ソースファイルに記述されています。

ソースファイル ：ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

シンプル API は割り込み処理の中から呼び出すことはできません。割り込みをトリガにする場合は、割り込み処理でフラグをセットし、割り込み処理の外で API を呼び出して下さい。

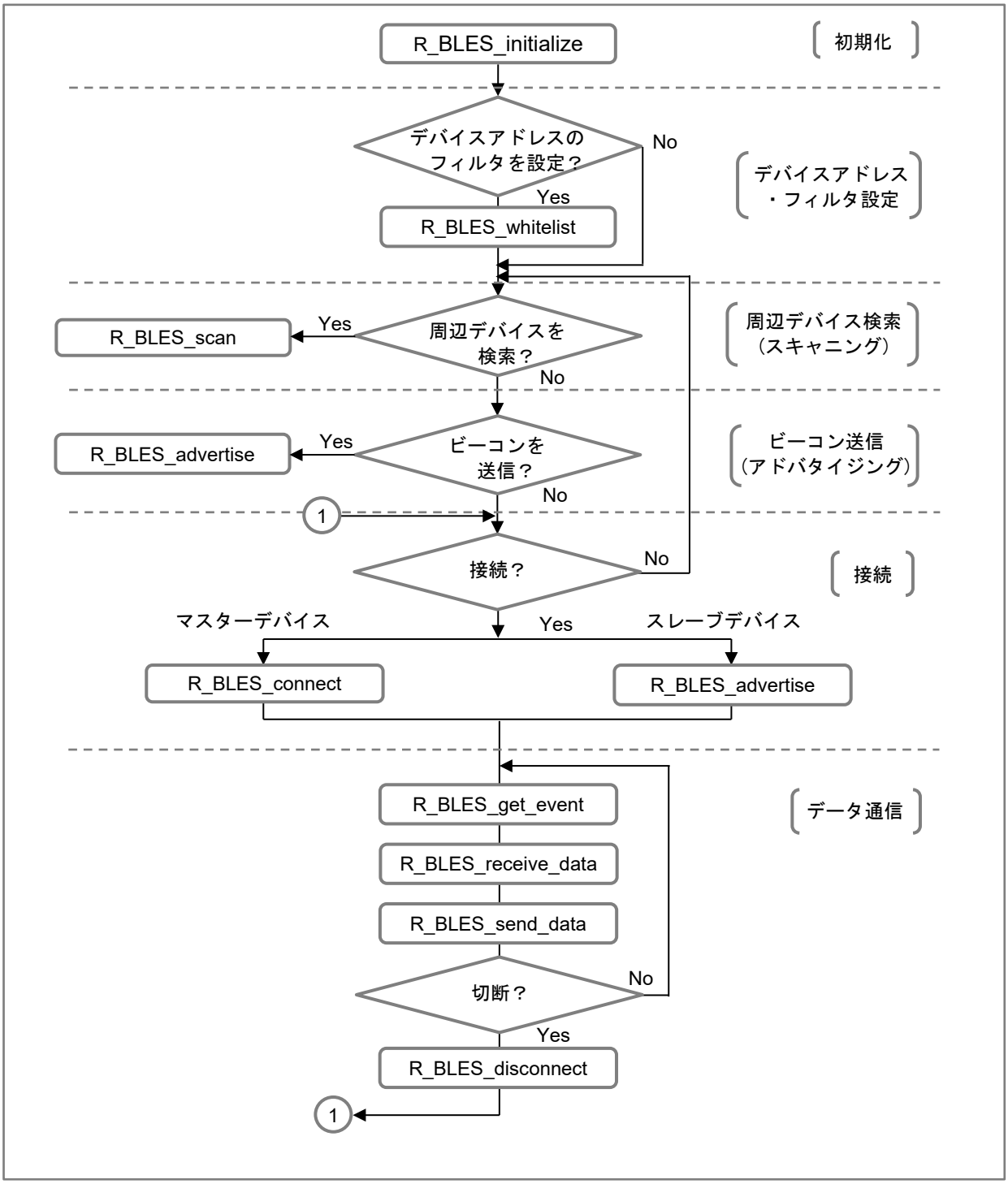

図 5-1　シンプル API 通信フローチャート

5.1 ランダムシード

シンプル API ではスレーブデバイスとして接続した時、ペアリングで使用するキーの生成に rand 関数から取得した疑似乱数値を使用します。R_BLES_initialize 関数でシンプル API を初期化する前に srand 関数でシード値を設定してください。プログラムではサンプルとして"0x12ef"を設定しています。

使用例)

ソースファイル: ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
/* The function is used to initialize the pseudo-random number generator */
/* for function rand() by passing the argument seed.                     */
srand(0x12ef);
```

5.2 初期化

最初に、R_BLES_initialize 関数を使用して、Host MCU と BLE MCU 間通信の初期化と、シンプル API の初期化を行います。また、接続したリモートデバイスから汎用双方向通信プロファイルのみ参照できるように、RL78/G1D モジュール、または BLE ソフトウェアに入っている他のサービスを非公開にします。

関数仕様は「6.1.1 R_BLES_initialize」を参照してください。

使用例)

ソースファイル: ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
R_BLES_initialize();
```

5.3 デバイスアドレス・フィルタ設定

ビーコン(アドバタイジング)を出している特定の BLE デバイスを検索(スキャニング)する場合や、ビーコン(アドバタイジング)に対する応答(スキャン・リクエスト)を特定の BLE デバイスからのみ受け付けるようにする場合に、R_BLES_whitelist 関数を使用してホワイトリストにデバイスアドレスを設定します。第 1 引数でデバイスアドレス構造体を指定します。設定できるデバイスアドレス数は、パブリックアドレスが 6 台、ランダムアドレスが 6 台の合計 12 台です。

関数仕様は「6.1.2 R_BLES_whitelist」を参照してください。

使用例)

ソースファイル: ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
R_BLES_whitelist(&WhiteList);
```

ホワイトリストに登録するデバイスアドレス構造体を以下に示します。登録できるデバイスアドレスは最大 12 台です。登録するデバイスアドレス数を変更する場合は、"WL_DEVADDR_LIST_NUM"定義値を変更してください。そして構造体の中で定義するデバイスアドレスを定義値と同じ数にしてください。

ソースファイル：ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
/*****************************************************************************
Whitelist
*****************************************************************************/
RBLE_WHITELIST  WhiteList = {
    {
        /* Address type     Device address                    Flag */
        {{RBLE_ADDR_PUBLIC, {0x51, 0x55, 0x77, 0x77, 0x77, 0x77}}, TRUE},
        {{RBLE_ADDR_PUBLIC, {0x02, 0x90, 0xa0, 0x23, 0x07, 0x10}}, TRUE},
        :
        {{RBLE_ADDR_RAND,   {0x01, 0x00, 0xde, 0xfa, 0xfe, 0xca}}, TRUE},
        {{RBLE_ADDR_RAND,   {0x02, 0x00, 0xde, 0xfa, 0xfe, 0xca}}, TRUE},
        :
    }
};
```

WL_DEVADDR_LIST_NUM と同じ数のデバイスアドレスを定義してください。

デバイスアドレス数の定義を以下に示します。

ソースファイル: ble_simple_api_rl78g14¥src¥rBLE¥src¥rbles_api¥rbles_api.h

```
#define WL_DEVADDR_LIST_NUM     (12)    /* Number of device address for white list */
```

5.4　周辺デバイス検索

ビーコン(アドバタイジング)を出している BLE デバイスを検索(スキャニング)する場合に、R_BLES_scan 関数を使用します。第 1 引数にスキャンニングの動作を設定するパラメータ構造体を指定します。第 2 引数に、検索で見つかった周辺デバイスのアドバタイジング情報が格納されます。第 3 引数はスキャニング実行時間です。

R_BLES_scan 関数の呼び出し後、第 2 引数で指定したアドバタイジング情報格納バッファがフルになるか、第 3 引数で指定したスキャニング実行時間が経過すると関数から戻ります。

関数仕様は「6.1.3 R_BLES_scan」を参照してください。

使用例)

ソースファイル：ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
R_BLES_scan(&ScanParam, &AdvReportList, DUR_5S);
```

アドバタイジング情報を格納する構造体のサイズを変更する場合は"ADV_REPORT_LIST_NUM"定義値を変更します。アドバタイジング情報数の定義を以下に示します。

ソースファイル：ble_simple_api_rl78g14¥src¥rBLE¥src¥rbles_api¥rbles_api.h

```
#define ADV_REPORT_LIST_NUM     (30)    /* Number of advertising report */
```

5.5 ビーコン送信

R_BLES_advertise 関数を使用してビーコン(アドバタイジング)を送信します。第 1 引数でアドバタイジングの動作を設定するパラメータ構造体を指定します。第 2 引数はアドバタイジング実行時間です。

ビーコンのようにリモートデバイスと接続を行わない設定の場合、アドバタイジング実行時間が経過すると関数から戻ります。

関数仕様は「6.1.4 R_BLES_advertise」を参照してください。

使用例)

ソースファイル: ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
R_BLES_advertise(&AdvParam, DUR_5S);
```

ビーコンとして使用する場合のパラメータ推奨設定を以下に示します。

表 5-1　ビーコン推奨設定

構造体メンバ	定義	説明
disc_mode	RBLE_GAP_BROADCASTER	アドバタイジングで、データをブロードキャスト
conn_mode	0	ブロードキャスターとして動作
adv_type	RBLE_GAP_ADV_NONCONN_UNDIR	アドバタイザーからの情報送信のみ

5.6 接続

リモートデバイスと接続を行います。ローカルデバイスがマスターデバイスとしてリモートデバイスと接続を行う場合、R_BLES_connect 関数を使用します。第 1 引数で接続時の動作を設定するパラメータ構造体を指定します。接続が完了すると、第 2 引数に接続情報が格納されます。第 3 引数は接続実行時間です。

R_BLES_connect 関数の呼び出し後、リモートデバイスと接続が完了するか、または接続実行時間が経過すると関数から戻ります。関数の戻り値が"RLBE_OK"の場合、接続が完了します。接続完了後「5.7 データ通信」処理を実行してください。

関数仕様は「6.1.5 R_BLES_connect」を参照してください。

使用例)

ソースファイル: ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
while(1)
{
    ret = R_BLES_connect(&CreConParam, &ConInfo, DUR_5S);
    if(ret == RBLE_OK)
    {
        break;
    }
}
```

接続にはリモートデバイス(接続相手)のデバイスアドレスが必要で、2 種類の指定方法があります。

(1) ホワイトリストを使用せず第 1 引数の接続パラメータでリモートデバイスを指定

(2) ホワイトリストを使用しホワイトリストでリモートデバイスを指定

(1)の場合は、第 1 引数のメンバ peer_addr_type と peer_addr にリモートデバイスのアドレスタイプとデバイスアドレスを設定して下さい。

(2)の場合は、ホワイトリストにリモートデバイスのデバイスアドレスを設定します。設定方法は「5.3 デバイスアドレス・フィルタ設定」を参照してください。

ローカルデバイスがスレーブデバイスとしてリモートデバイスと接続を行う場合、R_BLES_advertise 関数を使用します。第 1 引数でアドバタイジングの動作を設定するパラメータ構造体を指定します。第 2 引数はアドバタイジング(接続)実行時間です。

R_BLES_advertise 関数の呼び出し後、リモートデバイスと接続が完了するか、またはアドバタイジング(接続)実行時間が経過すると関数から戻ります。関数の戻り値が"RBLE_CONNECTED"の場合、接続が完了します。接続完了後「5.7 データ通信」処理を実行してください。

関数仕様は「6.1.4 R_BLES_advertise」を参照してください。

使用例)

ソースファイル：ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
while(1)
{
    ret = R_BLES_advertise(&AdvParam, DUR_5S);
    if(ret == RBLE_CONNECTED)
    {
        break;
    }
}
```

5.7 データ通信

リモートデバイスとの接続完了後、データ通信を行います。データ通信は、データ通信で発生するイベントを取得する R_BLES_get_event 関数、BLE で受信したデータを取り出す R_BLES_receive_data 関数、データを送信する R_BLES_send_data 関数の 3 つの関数で行います。

R_BLES_get_event 関数は、接続完了後のデータ通信中に必ず定期的に呼び出す必要があります。データ通信で発生したイベントは第 1 引数の変数に格納されます。発生するイベントは、"RBLES_EVENT_DISCONNECT"(切断)と、"RBLES_EVENT_RECEIVE_DATA"(データ受信)の 2 種類です。"RBLES_EVENT_DISCONNECT"イベントは、リモートデバイスとの接続が切断されたことを示すイベントです。このイベントが発生した場合は、データ通信のループ処理から抜けて再度接続を実行してください。"RBLES_EVENT_RECEIVE_DATA"イベントは、データを受信したことを示すイベントです。このイベントが発生した場合は、R_BLES_receive_data 関数で受信したデータを読み出してください。

R_BLES_receive_data 関数は、第 1 引数に受信データを格納するバッファを指定します。第 2 引数は読み出すデータ数です。関数の返り値に、読み出すことができたデータ数を返します。

データの送信は、R_BLES_send_data 関数を使用します。データの送信は、データ送信要求変数"r_send_data_req"を用意し、AD 変換割り込みや、スイッチ割り込み等で送信するデータを準備しデータ送信要求フラグをセットします。R_BLES_get_event 関数が実行されるループ処理の中でデータ送信要求フラグ

をチェックし R_BLES_send_data 関数を呼び出してください。第 1 引数に送信データバッファを指定します。第 2 引数は送信データ数です。

R_BLES_disconnect 関数は、データ通信を終了する場合にリモートデバイスとの接続を切断するために使用します。切断要求変数"r_disconnect_req"を用意し、スイッチ割り込み等で切断要求フラグをセットします。R_BLES_get_event 関数が実行されるループ処理の中で切断要求フラグをチェックし R_BLES_disconnect 関数を呼び出してください。切断後は接続から再実行してください。

関数仕様は「6.1.6 R_BLES_get_event」、「6.1.7 R_BLES_send_data」、「6.1.8 R_BLES_receive_data」、「6.1.9 R_BLES_disconnect」を参照してください。

使用例)

ソースファイル：ble_simple_api_rl78g14¥src¥ble_simple_api_rl78g14.c

```
while(1)
{
    R_BLES_get_event(&evt);

    if(evt == RBLES_EVENT_DISCONNECT)
    {
        /* disconnection */
        break;
    }
    else if(evt == RBLES_EVENT_RECEIVE_DATA)
    {
        /* get recive data */
        rxnum = R_BLES_receive_data(rxbuf, 20);
    }
    else
    {
        /* do nothing */
    }

    if(r_send_req == TRUE)
    {
        /* send data */
        R_BLES_send_data((uint8_t *)"Hello", 5);
        r_send_req = FALSE;
    }

    if(r_disconnect_req == TRUE)
    {
        R_BLES_disconnect();

        /* disconnection */
        r_disconnect_req = FALSE;
        break;
    }
}
```

受信データの取り出しが間に合わない場合、受信データを格納するシンプル API の内部バッファが溢れる可能性があり、溢れたデータは破棄されます。

内部バッファのサイズを変更する場合は、"RBLES_RDBUF_SIZE"定義値を変更します。

ソースファイル：ble_simple_api_rl78g14¥src¥rBLE¥src¥rbles_api¥rbles_api.h

```
#define RBLES_RDBUF_SIZE       (100)   /* Ring buffer size of receive data. */
```

6. シンプル API 仕様

シンプル API で定義する、API、構造体、マクロの仕様を示します。シンプル API を使用する場合、ユーザアプリケーションから個々に rBLE_API を呼び出すことはできません。シンプル API のみを使用して下さい。また、シンプル API は割り込み処理の中から呼び出すことはできません。必ず割り込み処理を抜けてから呼び出してください。

6.1 API

1. R_BLES_initialize()　シンプル API の初期化
2. R_BLES_whitelist()　ホワイトリストにデバイスアドレスを設定
3. R_BLES_scan()　スキャニングの実行
4. R_BLES_advertise()　アドバタイジングの実行とマスターデバイスとの接続
5. R_BLES_connect()　スレーブデバイスとの接続
6. R_BLES_get_event()　イベントの取得
7. R_BLES_send_data()　データ送信
8. R_BLES_receive_data()　データ受信
9. R_BLES_disconnect()　マスターデバイス、またはスレーブデバイスとの接続を切断

6.1.1 R_BLES_initialize

RBLE_STATUS R_BLES_initialize(void)		
Host MCU と BLE MCU 間通信、シンプル API を初期化します。接続したリモートデバイスから汎用双方向通信プロファイルのみ参照できるように、RL78/G1D モジュール、または BLE ソフトウェアに入っている他のサービスを非公開にします。シンプル API の初期化完了で本 API からリターンします。 ※ 本関数は割り込み処理の中では使用できません。		
Parameters:		
none		
Return:		
RBLE_OK	0x00	正常終了
RBLE_ERR	0xF0	シーケンスエラー
RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.1.2 R_BLES_whitelist

RBLE_STATUS R_BLES_whitelist(RBLE_WHITELIST *whitelist)

ホワイトリストにデバイスアドレスを追加します。ホワイトリストへ全てのデバイスアドレスの追加が完了すると本 API からリターンします。

※ 本関数は割り込み処理の中では使用できません。

Parameters:

*whitelist			ホワイトリストに登録するデバイスアドレス
	dev_info		デバイスアドレス情報
		dev_addr_type	デバイスアドレス・タイプ RBLE_ADDR_PUBLIC：パブリックアドレス RBLE_ADDR_RAND：ランダムアドレス
		dev_addr	デバイスアドレス
	dev_en		デバイスアドレス有効化フラグ TRUE：有効なデバイスアドレス (ホワイトリストに登録されます) FALSE：無効なデバイスアドレス (ホワイトリストに登録されません)

Return:

RBLE_OK	0x00	正常終了
RBLE_ERR	0xF0	シーケンスエラー
RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_PARAM_ERR	0xF3	パラメータエラー

6.1.3 R_BLES_scan

RBLE_STATUS R_BLES_scan(RBLE_SCANNING_INFO *scan_param,
RBLE_ADV_REPORT_LIST *adv_report, uint16_t duration)

スキャニングを duration で指定した時間実行します。スキャニング実行中に取得したアドバタイジング・レポートを *adv_report に格納します。*adv_report が一杯になった場合、スキャンを中断します。スキャニング実行時間が経過、またはアドバタイジング・レポート・リストが一杯になると本 API からリターンします。

※ 本関数は割り込み処理の中では使用できません。

Parameters:

*scan_param		スキャニング・パラメータ
	scan_type	スキャンニング・タイプ RBLE_SCAN_PASSIVE：パッシブ・スキャニング(受信のみ)を実行する RBLE_SCAN_ACTIVE：アクティブ・スキャニング(SCAN_REQ を送信)を実行する
	scan_intv	スキャン・インターバル N = 2 ～ 10240 (Time = 2.5 msec ～ 10240 msec (2.5 msec ～ 10.24 sec)) ※ N=2 を指定した場合、スキャン・インターバルは 2.5 msec になります
	scan_window	スキャン・ウインドウ N = 2 ～ 10240 (Time = 2.5 msec ～ 10240 msec (2.5 msec ～ 10.24 sec)) ※ N=2 を指定した場合、スキャン・ウインドウは 2.5 msec になります ※ 必ず、スキャン・インターバル＞スキャン・ウインドウ の設定にしてください
	own_addr_type	ローカルデバイスのデバイスアドレス RBLE_ADDR_PUBLIC：パブリックアドレス RBLE_ADDR_RAND：ランダムアドレス
	scan_filt_policy	スキャニング・フィルタ・ポリシー RBLE_SCAN_ALLOW_ADV_ALL：全てのアドバタイジング・パケットを受信する

	RBLE_SCAN_ALLOW_ADV_WLST: ホワイトリストに登録されているデバイスからのアドバタイジング・パケットを受信する RLBE_SCAN_ALLOW_VUART_SRV: 汎用双方向通信プロファイルの UUID が登録されているアドバタイジング・パケットを受信する
filter_dup	重複フィルタ RBLE_SCAN_FILT_DUPLIC_DIS: 重複するアドバタイジング・パケットを受信しない RBLE_SCAN_FILT_DUPLIC_EN: 重複するアドバタイジング・パケットを受信する
*adv_report	アドバタイジング・レポート
adv_list[]	アドバタイジング・レポート・リスト
evt_type	アドバタイジング・イベントタイプ 0x00: Connectable undirected advertising 0x01: Connectable directed advertising 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising 0x04: Scan Response
adv_addr_type	アドバタイザーのアドレスタイプ RBLE_ADDR_PUBLIC: パブリックアドレス RBLE_ADDR_RAND: ランダムアドレス
adv_addr	アドバタイザーのデバイスアドレス
data_len	アドバタイジング・データ長
data[]	アドバタイジングまたはスキャン・レスポンスのデータ ※　アドバタイジング・データおよびスキャン・レスポンス・データのフォーマットは、Bluetooth Low Energy プロトコルスタック・ユーザーズ・マニュアルを参照してください。
rssi	アドバタイジング・データ受信時の RSSI
adv_list_num	受信したアドバタイジング・レポートの数
duration	スキャニング実行時間 N = 1 ～ 60000 (Time = N × 10 msec (10 msec ～ 600 sec))

Return:

RBLE_OK	0x00	正常終了
RBLE_ERR	0xF0	シーケンスエラー
RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_PARAM_ERR	0xF3	パラメータエラー
RBLE_ADVLIST_FULL	0xFA	アドバタイジング・レポート・リストがフル

6.1.4　R_BLES_advertise

```
RBLE_STATUS R_BLES_advertise(RBLE_BROADCAST_ENABLE_PARAM *adv_param,
                             uint16_t duration)
```

アドバタイジングを duration で指定した時間実行します。接続を許可するアドバタイジングの場合、マスターデバイスと接続します。アドバタイジング実行時間が経過、またはマスターデバイスとの接続完了で本 API からリターンします。

※ 本関数は割り込み処理の中では使用できません。

Parameters:

*adv_param	アドバタイジング・パラメータ
disc_mode	発見可能モード RBLE_GAP_NON_DISCOVERABLE: General または Limited Discovery プロシージャを実行するデバイスから発見されない RBLE_GAP_GEN_DISCOVERABLE: General Discovery をプロシージャ実行するデバイスから発見されることが可能

RBLE_GAP_LIM_DISCOVERABLE	General または Limited Discovery プロシージャを実行するデバイスから限られた期間発見されることが可能
RBLE_GAP_BROADCASTER	Advertising イベントにてデータをブロードキャスト

conn_mode　接続可能モード

0	Broadcaster として動作
RBLE_GAP_NON_CONNECTABLE	接続の確立を許可しない
RBLE_GAP_UND_CONNECTABLE	接続を許可する
RBLE_GAP_DIR_CONNECTABLE	既知のデバイスからのみ接続を許可する

adv_info

adv_intv_min	adv_intv_max と同じ値を設定してください
adv_intv_max	最大アドバタイジング・インターバル N = 20 ～ 10240 (Time = 20 msec ～ 10240 msec (20 msec ～ 10.24 sec))
adv_type	アドバタイジング・タイプ

RBLE_GAP_ADV_CONN_UNDIR	CONNECT_REQ または SCAN_REQ に応答可能
RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY	指定デバイスとのみ接続可能
RBLE_GAP_ADV_DISC_UNDIR	SCAN_REQ に応答可能
RBLE_GAP_ADV_NONCONN_UNDIR	Advertiser からの情報送信のみ
RBLE_GAP_ADV_CONN_DIR_LOW_DUTY	指定デバイスとのみ接続可能

own_addr_type　ローカルデバイスのデバイスアドレス・タイプ

RBLE_ADDR_PUBLIC	パブリックアドレス
RBLE_ADDR_RAND	ランダムアドレス

direct_addr_type　ダイレクト接続アドレスタイプ

RBLE_ADDR_PUBLIC	パブリックアドレス
RBLE_ADDR_RAND	ランダムアドレス

direct_addr　ダイレクト接続アドレス

※　adv_type で、RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY または RBLE_GAP_ADV_CONN_DIR_LOW_DUTY を選択した時に有効になります。

adv_chnl_map　アドバタイジング・チャネル

RBLE_ADV_CHANNEL_37	37ch を使用する
RBLE_ADV_CHANNEL_38	38ch を使用する
RBLE_ADV_CHANNEL_39	39ch を使用する
RBLE_ADV_ALL_CHANNELS	全チャネルを使用する

adv_filt_policy　アドバタイジング・フィルタ・ポリシー

RBLE_ADV_ALLOW_SCAN_ANY_CON_ANY	SCAN_REQ　：全て許可 CONNECT_REQ　：全て許可
RBLE_ADV_ALLOW_SCAN_WLST_CON_ANY	SCAN_REQ　：ホワイトリストのみ許可 CONNECT_REQ　：全て許可
RBLE_ADV_ALLOW_SCAN_ANY_CON_WLST	SCAN_REQ　：全て許可 CONNECT_REQ　：ホワイトリストのみ許可
RBLE_ADV_ALLOW_SCAN_WLST_CON_WLST	SCAN_REQ　：ホワイトリストのみ許可 CONNECT_REQ　：ホワイトリストのみ許可

adv_data_len	アドバタイジング・データ長
adv_data	アドバタイジング・データ
scan_rsp_data_len	スキャン・レスポンス・データ長
data	スキャン・レスポンス・データ
duration	アドバタイジング実行時間 N = 1 ～ 60000 (Time = N × 10 msec (10 msec ～ 600 sec))

Return:

RBLE_OK	0x00	正常終了
RBLE_ERR	0xF0	シーケンスエラー

RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_PARAM_ERR	0xF3	パラメータエラー
RBLE_CONNECTED	0xF9	マスターデバイスと接続完了

6.1.5 R_BLES_connect

RBLE_STATUS R_BLES_connect(RBLE_CREATE_CONNECT_PARAM *conn_param,
RBLE_CONNECT_INFO *conn_info, uint16_t duration)

リモートデバイスと接続します。リモートデバイスとの接続情報を*conn_info に格納します。リモートデバイスと接続できずに duration で指定した時間が経過するとタイムアウトします。接続実行時間が経過、またはリモートデバイスとの接続完了で本 API からリターンします。

※ 本関数は割り込み処理の中では使用できません。

Parameters:

*conn_param	接続パラメータ
scan_intv	スキャン・インターバル N = 2 ～ 10240 (Time = 2.5 msec ～ 10240 msec (2.5 msec ～ 10.24 sec)) ※ N=2 を指定した場合、スキャン・インターバルは 2.5 msec になります
scan_window	スキャン・ウインドウ N = 2 ～ 10240 (Time = 2.5 msec ～ 10240 msec (2.5 msec ～ 10.24 sec)) ※ N=2 を指定した場合、スキャン・ウインドウは 2.5 msec になります ※ 必ず、スキャン・インターバル＞スキャン・ウインドウ の設定にしてください
init_filt_policy	イニシエータ・フィルタ・ポリシー

RBLE_GAP_INIT_FILT_IGNORE_WLST	ホワイトリストを使用せず、peer_addr_type, peer_addr で指定されたデバイスと接続する
RBLE_GAP_INIT_FILT_USE_WLST	ホワイトリストを使用し、に登録されているデバイスと接続する。(peer_addr_type, peer_addr は無視される)

peer_addr_type	リモートデバイスのアドレスタイプ

RBLE_ADDR_PUBLIC	パブリックアドレス
RBLE_ADDR_RAND	ランダムアドレス

※ このパラメータは init_filt_policy が RBLE_GAP_INIT_FILT_IGNORE_WLST の時のみ有効です。

peer_addr	リモートデバイスのデバイスアドレス ※ このパラメータは init_filt_policy が RBLE_GAP_INIT_FILT_IGNORE_WLST の時のみ有効です。
own_addr_type	ローカルデバイスのデバイスアドレス

RBLE_ADDR_PUBLIC	パブリックアドレス
RBLE_ADDR_RAND	ランダムアドレス

con_intv_min	con_intv_max と同じ値を設定してください
con_intv_max	最大コネクション・インターバル N = 7 ～ 4000 (Time = 7.5 msec ～ 4000 msec (7.5 msec ～ 4 sec))
con_latency	コネクション・スレーブ・レイテンシー N = 0000 ～ 499
superv_to	スーパービジョン・タイムアウト N = 100 ～32000 (Time = 100 msec ～ 32000 msec (100 msec ～ 32 sec))
*conn_info	接続結果情報
status	reserved
role	reserved
conhdl	reserved
peer_addr_type	リモートデバイスのアドレスタイプ

RBLE_ADDR_PUBLIC	パブリックアドレス

		RBLE_ADDR_RAND　ランダムアドレス
	peer_addr	リモートデバイスのデバイスアドレス
	idx	reserved
	con_interval	コネクション・インターバル N = 0x0006 ～ 0x0C80 (Time = N x 1.25msec (7.5 msec～4.0 sec))
	con_latency	コネクション・スレーブ・レイテンシー N = 0000 ～ 499
	sup_to	スーパービジョン・タイムアウト N = 0x000A ～ 0x0C80 (Time = N x 10 msec (100 msec～32 sec))
	clk_accuracy	クロック精度 500 ppm: 0 250 ppm: 1 150 ppm: 2 100 ppm: 3 75 ppm: 4 50 ppm: 5 30 ppm: 6 20 ppm: 7
duration		接続実行時間 N = 1 ～ 60000 (Time = N x 10 msec (10 msec ～ 600 sec))

Return:

RBLE_OK	0x00	正常終了
RBLE_ERR	0xF0	シーケンスエラー
RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_PARAM_ERR	0xF3	パラメータエラー
RBLE_TIMEOUT	0xFB	接続タイムアウト

6.1.6 R_BLES_get_event

RBLE_STATUS R_BLES_get_event(uint8_t *evt)

接続しているリモートデバイスからのデータ受信や接続解除のイベントを通知します。リモートデバイスとの接続後は本関数を定期的に呼び出してください。

※ 本関数は割り込み処理の中では使用できません。

Parameters:

*evt	イベント RBLES_EVENT_NONE: 0x00 イベント無し RBLES_EVENT_DISCONNECT: 0x01 リモートデバイスとの接続を切断した RBLES_EVENT_RECEIVE_DATA: 0x02 データを受信した

Return:

RBLE_OK	0x00	正常終了
RBLE_ERR	0xF0	シーケンスエラー
RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_PARAM_ERR	0xF3	パラメータエラー

6.1.7 R_BLES_send_data

RBLE_STATUS R_BLES_send_data(uint8_t *txbuf, uint8_t len)			
接続しているリモートデバイスへデータを送信します。マスターデバイスとして接続している場合、Write Request でデータを送信し、Write Response の受信で本 API からリターンします。スレーブデバイスとして接続している場合、Indication でデータを送信し、Confirmation の受信で本 API からリターンします。 ※ 本関数は割り込み処理の中では使用できません。			
Parameters:			
	*txbuf	送信データの格納バッファ	
	len	送信データ数 (最大 20 バイト)	
Return:			
	RBLE_OK	0x00	正常終了
	RBLE_ERR	0xF0	シーケンスエラー
	RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
	RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
	RBLE_PARAM_ERR	0xF3	パラメータエラー

6.1.8 R_BLES_receive_data

uint16_t R_BLES_receive_data(uint8_t *rxbuf, uint16_t len)		
内部バッファに格納された受信データを取り出します。内部バッファに格納された受信データが len で指定したバイト数に満たない場合、内部バッファに格納されている分を*rxbuf に格納します。 受信データの取り出しが間に合わず内部バッファが溢れてしまう場合、溢れたデータは破棄されます。 ※ 本関数は割り込み処理の中では使用できません。		
Parameters:		
	*rxbuf	受信データの格納バッファ
	len	受信データ数 (単位:バイト)
Return:		
	0	受信データなし
	0 以外	取り出したデータ数

6.1.9 R_BLES_disconnect

RBLE_STATUS R_BLES_disconnect(void)			
リモートデバイスとの接続を切断します。切断完了で本 API からリターンします。 ※　本関数は割り込み処理の中では使用できません。			
Parameters:			
	none		
Return:			
	RBLE_OK	0x00	正常終了
	RBLE_ERR	0xF0	シーケンスエラー
	RBLE_TRANS_ERR	0xF1	Host MCU と BLE MCU 間の通信エラー
	RBLE_STATUS_ERROR	0xF2	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

6.2 構造体

6.2.1 RBLE_BROADCAST_ENABLE_PARAM

Advertising parameter structure		
uint16_t	disc_mode	Discovery Mode
uint16_t	conn_mode	Connectable Mode
RBLE_ADV_INFO	adv_info	Advertising Infomation

6.2.2 RBLE_ADV_INFO

Advertising Infomation		
RBLE_SET_ADV_PARAM	adv_param	Advertising parameter structure
RBLE_SET_ADV_DATA	adv_data	Advertising data structure
RBLE_SET_SCAN_RSP_DATA	scan_rsp_data	Scan response data structure

6.2.3 RBLE_SET_ADV_PARAM

Advertising Infomation		
uint16_t	adv_intv_min	Minimum interval for advertising
uint16_t	adv_intv_max	Maximum interval for advertising
uint8_t	adv_type	Advertising type
uint8_t	own_addr_type	Own address type
uint8_t	direct_addr_type	Direct address type
RBLE_BD_ADDR	direct_addr	Direct Bluetooth device address
uint8_t	adv_chnl_map	Advertising channel map
uint8_t	adv_filt_policy	Advertising filter policy
uint8_t	reserved	-

6.2.4 RBLE_BD_ADDR

BD Address structure		
uint8_t	addr[6]	6-byte array address value

6.2.5 RBLE_SET_ADV_DATA

Advertising Data Command parameters structure		
uint8_t	adv_data_len	Advertising data length
RBLE_ADV_DATA	adv_data	Advertising data - maximum 31 bytes

6.2.6 RBLE_ADV_DATA

Advertising data structure		
uint8_t	data[31]	Maximum length data bytes array

6.2.7 RBLE_SET_SCAN_RSP_DATA

Set Scan Response Data Command parameters structure		
uint8_t	scan_rsp_data_len	Scan response data length
RBLE_SCAN_RSP_DATA	data	Advertising data - maximum 31 bytes

6.2.8 RBLE_SCAN_RSP_DATA

Scan response data structure		
uint8_t	data[31]	Maximum length data bytes array

6.2.9 RBLE_CREATE_CONNECT_PARAM

Create Connection Command parameters structure		
uint16_t	scan_intv	Scan interval
uint16_t	scan_window	Scan window size
uint8_t	init_filt_policy	Initiator filter policy
uint8_t	peer_addr_type	Peer address type

RBLE_BD_ADDR	peer_addr	Peer BD address
uint8_t	own_addr_type	Own address type
uint8_t	reserved	-
uint16_t	con_intv_min	Minimum of connection interval
uint16_t	con_intv_max	Maximum of connection interval
uint16_t	con_latency	Connection latency
uint16_t	superv_to	Link supervision timeout
uint16_t	ce_len_min	Minimum CE length
uint16_t	ce_len_max	Maximum CE length

6.2.10 RBLE_CONNECT_INFO

Connection Information structure		
uint8_t	status	Confirmation status
uint8_t	role	Role
uint16_t	conhdl	Connection handle
uint8_t	peer_addr_type	Peer address type
RBLE_BD_ADDR	peer_addr	Peer BT address
uint8_t	idx	Connection Index
uint16_t	con_interval	Connection interval
uint16_t	con_latency	Connection latency
uint16_t	sup_to	Link supervision timeout
uint8_t	clk_accuracy	Clock accuracy
uint8_t	reserved3	-

6.2.11 RBLE_SCANNING_INFO

Scanning information referenced		
RBLE_SET_SCAN_PARAMETER	set_scan	Scan parameter command structure
uint8_t	filter_dup	Filtering policy
uint8_t	reserved	-

6.2.12 RBLE_SET_SCAN_PARAMETER

Set Scan Parameters Command parameters structure		
uint8_t	scan_type	Scan type
uint8_t	reserved	-
uint16_t	scan_intv	Scan interval
uint16_t	scan_window	Scan window size
uint8_t	own_addr_type	Own address type
uint8_t	scan_filt_policy	Scan filter policy

6.2.13 RBLE_WHITELIST

Scanning information referenced		
RBLE_WLIST_DEV_ADDR	dev_list[WL_DEVADDR_LIST_NUM]	Device address list structure

6.2.14 RBLE_WLIST_DEV_ADDR

Scanning information referenced		
RBLE_DEV_ADDR_INFO	dev_info	Device address structure
bool	dev_en	Device address enable flag

6.2.15 RBLE_DEV_ADDR_INFO

Add Device(Remove Device) to White List Command parameters structure		
uint8_t	dev_addr_type	Type of address of the device to be added to(removed from) the White List
RBLE_BD_ADDR	dev_addr	Address of device to be added to(removed from) White List

6.3 マクロ

6.3.1 ADV_REPORT_LIST_NUM

ADV_REPORT_LIST_NUM	アドバタイジング・レポート・リスト・サイズ (デフォルト設定： 30)

6.3.2 WL_DEVADDR_LIST_NUM

WL_DEVADDR_LIST_NUM	ホワイトリスト設定デバイスアドレス・サイズ (最大 12) (デフォルト設定： 12)

6.3.3 RBLES_RDBUF_SIZE

RBLES_RDBUF_SIZE	内部受信バッファ・サイズ (byte) (デフォルト設定： 100)

6.4 ペアリング情報

・パラメータ

Bonding	Bondable Mode
Security Mode	Unauthenticated pairing with encryption
Pairing Method	Just Works
IO capability	No Input No Output
OOB flag	OOB Data not present
Authentication Requirements	No MITM Bonding
Encryption key size	128 [bit]
Initiator key distribution	None
Responder key distribution	Encryption key

7. 注意事項

7.1 コード生成 (r_cg_macrodriver.h)

CS+ for CC、e² studio のコード生成ツールから生成される下記のファイルは、型宣言の重複を避けるために本プロジェクトで修正を加えています。

- r_cg_macrodriver.h

新たにコードを生成した場合はファイルが上書きされてしまいます。修正後のファイルを各統合開発環境プロジェクトのコード生成フォルダに置いていますので、生成した r_cg_macrodriver.h を削除し、修正後のファイルをリネームして使用して下さい。

表 7-1　修正後の r_cg_macrodriver.h

統合開発環境	コード生成フォルダ	修正後の r_cg_macrodriver.h 名
CS+ for CC	ble_simple_api_rl78g14¥project¥CC_CCRL¥cg	r_cg_macrodriver_g14_ccrl.h
e² studio	ble_simple_api_rl78g14¥project¥e2studio¥src	r_cg_macrodriver_g14_ccrl.h

7.2 シンプル API 呼び出し方法

シンプル API は割り込み処理の中から呼び出すことはできません。シンプル API の中で UART 割り込みを使用し BLE MCU と通信しているため、割り込み処理の中からシンプル API を呼び出すと、API から戻らなくなります。

8. Appendix

8.1 BLE ソフトウェアの変更

本アプリケーションノートに同梱するソフトウェアは、「図 2-1 システム構成(1) (RY7011)」で利用するように初期設定されています。「図 2-2 システム構成(2) (BLE Evaluation Board)」で利用するために、本章の説明に従って Host MCU プログラム(シンプル API プログラム)と BLE MCU プログラム(BLE ソフトウェア)の設定を変更してください。

本章にて CS+ for CC を用いてビルドしたファイルが本アプリケーションノートに同梱されています。詳しくは「3.5.3 実行ファイルのフォルダ構成」を参照してください。

- Host MCU

  シンプル API プログラムの変更

  - UART 接続方式の変更　　: 2 線分岐接続方式 → 2 線接続方式
  - UART ボーレートの変更　　: 115200 bps → 4800 bps

- BLE MCU

  BLE ソフトウェアの変更

  - 汎用双方向通信データベースの追加

8.1.1 Host MCU プログラムの準備

シンプル API プログラムの UART のボーレートと Host MCU と BLE MCU 間の接続方式を変更します。UART のボーレートは、RL78/G1D モジュールの初期ボーレート(115200bps)から、モデム構成 BLE ソフトウェアの初期ボーレート(4800bps)に、統合開発環境のコード生成ツールを使用して変更します。そして、接続方式を UART 2 線分岐接続方式から UART 2 線接続方式に変更します。

コード生成については注意事項があります。CS+ for CC、e² studio の場合、型宣言の重複を避けるため "r_cg_macrodriver.h"に修正を加えています。修正方法は「7.1 コード生成」を参照してください。

(1) シンプル API プログラムのボーレート変更

- CS+ for CA, CX / CS+ for CC

  1. 「表 8-1　Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」示すプロジェクトファイルをダブルクリックします。これにより CS+が起動します。
  2. 「プロジェクトツリー」内の「コード生成(設計ツール)」ツリーから「シリアル」をダブルクリックします。
  3. 「コード生成」タブ→「SAU1」タブ→「UART2」タブ→「受信」タブと「送信」タブの「ボーレート」の項目を"4800"に変更して下さい。
  4. 「コード生成」ボタンを押し、コードを生成して下さい。

- e² studio

1. e² studio を起動します。
2. 「プロジェクトエクスプローラー」上で右クリックし、表示されたメニューから「インポート」を選択します。
3. 「インポート」ウィンドが表示されるので、「既存プロジェクトをワークスペースへ」を選択し、「次」をクリックします。
4. 「ルート・ディレクトリの選択」フォームに、「表 8-1　Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すプロジェクトディレクトリを選択します。選択後、「プロジェクト」内に指定したプロジェクトが表示されていることを確認し、「終了」をクリックします。すると、「インポート」ウィンドが閉じられます。
5. 「プロジェクトエクスプローラー」上に表示されたツリーの「コード生成」→「周辺機能」→「シリアル」をダブルクリックします。
6. 「SAU1」タブ→「UART2」タブ→「受信」タブと「送信」タブの「ボーレート」の項目を"4800"に変更して下さい。
7. 「コード生成」ボタンを押し、コードを生成して下さい。

(2)　UART 接続方式変更

UART 接続方式を、2 線接続方式に変更します。"uart.h"で定義されている以下の定義値を変更してください。

ファイル：

ble_simple_api_rl78g14¥src¥platform¥rl78g14¥driver¥serial¥uart.h

定義値：

#define SERIAL_U_DIV_2WIRE　(1) → (0)

(3)　統合開発環境の定義マクロ変更

使用する統合開発環境のプロジェクトで定義マクロを以下のように変更してください。

USE_MODULE_RY7011　　→ noUSE_MODULE_RY7011

USE_FW_UPDATE_PROFILE　　→ noUSE_FW_UPDATE_PROFILE

8.1.2 Host MCU プログラムのビルド

統合開発環境ごとに Host MCU プログラムのビルド方法を示します。

(1) CS+ for CA, CX / CS+ for CC

1. 「表 8-1 Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すプロジェクトファイルをダブルクリックします。これにより CS+が起動します。
2. [プロジェクトツリー内]の[ble_simple_api_rl78g14 (プロジェクト)]を右クリックし、ドロップダウンメニューから[ble_simple_api_rl78g14 をビルド]を選択して、ビルドを開始します。
3. 「表 8-1 Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すパスに実行ファイルが生成されます。

(2) e² studio

1. e² studio を起動します。
2. [プロジェクトエクスプローラー]上で右クリックし、表示されたメニューから[インポート]を選択します。
3. [インポート]ウィンドが表示されるので、[既存プロジェクトをワークスペースへ]を選択し、[次]をクリックします。
4. [ルート・ディレクトリの選択]フォームに、「表 8-1 Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すプロジェクトディレクトリを選択します。選択後、[プロジェクト]内に指定したプロジェクトが表示されていることを確認し、[終了]をクリックします。すると、[インポート]ウィンドが閉じられます。
5. [プロジェクトエクスプローラー]上に表示されたプロジェクト上で右クリックし、[プロジェクトのビルド]を選択し、ビルドを開始します。
6. 「表 8-1 Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すパスに実行ファイルが生成されます。

表 8-1　Host MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ

CS+ for CC		
	プロジェクトファイル	ble_simple_api_rl78g14¥project¥CS_CA¥ble_simple_api_rl78g14.mtpj
	実行ファイル	ble_simple_api_rl78g14¥project¥CS_CA¥DefaultBuild¥ble_simple_api_rl78g14.hex
CS+ for CA,CX		
	プロジェクトファイル	ble_simple_api_rl78g14¥project¥CS_CCRL¥ble_simple_api_rl78g14.mtpj
	実行ファイル	ble_simple_api_rl78g14¥project¥CS_CCRL¥DefaultBuild¥ble_simple_api_rl78g14.hex
e² studio		
	プロジェクトディレクトリ	ble_simple_api_rl78g14¥project¥e2studio
	実行ファイル	ble_simple_api_rl78g14¥project¥e2studio¥HardwareDebug¥ble_simple_api_rl78g14.hex

(3)　実行ファイルの書き込みとボードの設定

1. 「表 8-2　ジャンパ設定」参照して RSK ボードのジャンパを設定します。
2. E1 エミュレータを RSK に接続後、E1 エミュレータを PC に接続します。
3. AC 電源アダプタを RSK に接続後、AC 電源アダプタから電源供給を開始します。
4. RFP (Renesas Flash Programmer)を起動し、[ファイル]→[新しいプロジェクトを作成]を選択します。マイクロコントローラの選択画面で[RL78]を選択し、[接続]を押下して、ワークスペースを作成します。
5. [操作]タブの[プログラムファイル]で実行ファイルを選択します。
6. [操作]タブの[スタート]を押下して書き込み開始後、正常終了と表示されることを確認します。
7. AC 電源アダプタ、E1 エミュレータを RSK から取り外します。

表 8-2　ジャンパ設定

ジャンパ J5 設定	ジャンパ J6 設定	電源供給源	入力電圧	レギュレータ IC 供給電圧
Pin2-3 短絡	開放	PWRコネクタ	5V	3.3V

8.1.3 BLE MCU プログラムの準備

シンプル API では汎用双方向通信プロファイルを使用するため、BLE MCU で使用する BLE ソフトウェアに汎用双方向通信プロファイルのデータベースを組み込む必要があります。

(1) BLE ソフトウェア・ソースコードの準備

EEPROM エミュレーションライブラリ、コードフラッシュライブラリを Renesas の web サイトより入手し、以下のフォルダにコピーします。

BLE ソフトウェアのインストール詳細は「クイックスタートガイド (R01AN2767)」の「4. インストールする」を参照して下さい。

- EEPROM エミュレーションライブラリ
  - CC-RL を使用する場合

    RL78_G1D¥Project_Source¥renesas¥src¥driver¥dataflash¥cc_rl
  - CA78K0R を使用する場合

    RL78_G1D¥Project_Source¥renesas¥src¥driver¥dataflash¥cs

- コードフラッシュライブラリ
  - CC-RL を使用する場合

    RL78_G1D¥Project_Source¥renesas¥src¥driver¥codeflash¥cc_rl
  - CA78K0R を使用する場合

    RL78_G1D¥Project_Source¥renesas¥src¥driver¥codeflash¥cs

(2) 汎用双方向通信データベースの追加

シンプル API プログラムパッケージの中に BLE ソフトウェアの変更に必要な差分ファイルが入っています。コピー元の"Project_Source"フォルダを、BLE ソフトウェアのコピー先フォルダに上書きしてください。

コピー元フォルダ(シンプル API パッケージ) :

BLE_Software_Ver_1_20_diff_vuart¥RL78_G1D¥Project_Source

コピー先フォルダ(BLE ソフトウェア) :

Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D

差分のファイルを下記に示します。

BLE ソフトウェアを既に変更されている場合は、下記、差分ファイル中の"#ifdef USE_VUART_PROFILE ~ #endif"で囲まれた部分を、BLE ソフトウェアに反映してください。また追加ファイルを BLE ソフトウェアの同じフォルダにコピーしてください

表 8-3　汎用双方向通信データベース差分ファイル

フォルダ	ファイル	説明
RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78	db_handle.h	(差分) 汎用双方向通信 Attribute database handles 定義
	prf_config.c	(差分) 汎用双方向通信 Attribute database index 定義
	prf_config.h	(差分) 汎用双方向通信サービ定義
RL78_G1D¥Project_Source¥rBLE¥src¥include	rble_api_vuart.h	(追加) 汎用双方向通信ヘッダファイル

(3)　BLE ソフトウェアのプロジェクト定義マクロの変更

使用する統合開発環境のモデム構成プロジェクトに、下記の定義マクロを追加してください。

USE_VUART_PROFILE

8.1.4　BLE MCU プログラムのビルド

BLE MCU である RL78/G1D 評価ボードで動作するプログラムのビルド方法を示します。

(1)　CS+ for CA, CX / CS+ for CC

1. 「表 8-4 BLE MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すプロジェクトファイルをダブルクリックします。これにより CS+が起動します。
2. [プロジェクトツリー]内の[BLE_Emb（サブプロジェクト）]を右クリックし、ドロップダウンメニューから[BLE_Emb をビルド]を選択して、ビルドを開始します。
3. 「表 8-4 BLE MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すパスに実行ファイルが生成されます。

(2)　e² studio

1. e² studio を起動します。
2. [プロジェクトエクスプローラー]上で右クリックし、表示されたメニューから[インポート]を選択します。

3. [インポート]ウィンドが表示されるので、[既存プロジェクトをワークスペースへ]を選択し、[次]をクリックします。
4. [ルート・ディレクトリの選択]フォームに、「表 8-4 BLE MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すプロジェクトディレクトリを選択します。選択後、[プロジェクト]内に指定したプロジェクトが表示されていることを確認し、[終了]をクリックします。すると、[インポート]ウィンドが閉じられます。
5. [プロジェクトエクスプローラー]上に表示されたプロジェクト上で右クリックし、[プロジェクトのビルド]を選択し、ビルドを開始します。
6. 「表 8-4 BLE MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ」に示すパスに実行ファイルが生成されます。

表 8-4 BLE MCU のプロジェクトファイルと実行ファイルの生成ディレクトリ

CS+ for CC		
	プロジェクトファイル	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CS_CCRL¥BLE_Modem¥BLE_Modem.mtpj
	実行ファイル	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CS_CCRL¥BLE_Modem¥rBLE_Mdm¥DefaultBuild¥rBLE_Mdm_CCRL.hex
CS+ for CA,CX		
	プロジェクトファイル	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CubeSuite¥BLE_Modem¥BLE_Modem.mtpj
	実行ファイル	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CubeSuite¥BLE_Modem¥rBLE_emb¥DefaultBuild¥rBLE_emb.hex
e² studio		
	プロジェクトディレクトリ	RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem
	実行ファイル	RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm¥DefaultBuild¥rBLE_Mdm_CCRL.hex

(3) 実行ファイルの書き込みとボードの設定

1. 「表 8-5　スイッチ設定」を参照して RL78/G1D 評価ボードのスライドスイッチを設定します。
2. E1 エミュレータを RL78/G1D 評価ボードに接続後、E1 エミュレータを PC に接続します。
3. 電源を RL78/G1D 評価ボードに接続後、電源供給を開始します。
4. RFP（Renesas Flash Programmer）を起動し、[ファイル] → [新しいプロジェクトを作成]を選択します。マイクロコントローラの選択画面で[RL78]を選択し、[接続]を押下して、ワークスペースを作成します。
5. [ブロック設定]タブでコードフラッシュの Block255 およびデータフラッシュの全 Block の Erase と P.V のチェックを外します。
6. [操作設定]タブで[コマンド]として、[消去]と[書き込み]が選択されていることを確認します。
7. [操作]タブの[プログラムファイル]で生成した実行ファイルを選択します。
8. [スタート]を押下して書き込み開始後、正常終了と表示されることを確認します。
9. 電源、E1 エミュレータを RL78/G1D 評価ボードから取り外します。

表 8-5　スイッチ設定

スイッチ	設定	機能
SW7	2-3 接続(右側) <デフォルト設定>	AC電源アダプタまたはUSBからレギュレータ経由で電源供給
SW8	1-2 接続(左側) <デフォルト設定>	AC 電源アダプタから電源供給 ※USB から電源供給する場合は 2-3 接続(右側)
SW9	1-2 接続(左側)	外部拡張インタフェースと接続
SW10	1-2 接続(左側) <デフォルト設定>	モジュールに電源供給
SW11	2-3 接続(右側) <デフォルト設定>	E1デバッガ3.3V 以外から電源供給
SW12	2-3 接続(右側) <デフォルト設定>	(デフォルト固定)
SW13	1-2 接続(左側) <デフォルト設定>	USB 接続

8.1.5 Host MCU－BLE MCU 接続

Host MCU と BLE MCU の接続手順を示します。

(1) RSK とモジュール評価ボードの接続

「表 8-6 端子接続」を参照して RSK の端子とモジュール評価ボードの端子を接続してください。

表 8-6 端子接続

RL78/G14 端子(RSK 端子)	モジュール評価ボード	用途
TXD2(J3-Pin16)	RxD0(TH19)	UART(Host MCU→BLE MCU)
RXD2(J3-Pin15)	TxD0(TH23)	UART(BLE MCU→Host MCU)
Vss(GND1)	GND1 or GND2 or GND3	電源グラウンド

【注】 モジュール評価ボードのジャンパ(TH18 - TH19)はショートしてください。

(2) RSK と RL78/G1D 評価ボードの接続

「表 8-7 端子接続」を参照して RSK の端子と RL78/G1D 評価ボードの端子を接続してください。

表 8-7 端子接続

RL78/G14 端子(RSK 端子)	RL78/G1D 端子(評価ボード端子)	用途
TXD2(J3-Pin16)	RxD0(CN4-Pin16)	UART(Host MCU→BLE MCU)
RXD2(J3-Pin15)	TxD0(CN4-Pin14)	UART(BLE MCU→Host MCU)
Vss(GND1)	Vss(CN4-Pin26)	電源グラウンド

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
https://www.renesas.com/

お問合せ先
https://www.renesas.com/contact/

Bluetooth は、Bluetooth SIG, Inc., U.S.A.の登録商標です。
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。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.0	2017.04.14	-	初版発行
1.0	2022.01.31	-	Bluetooth Low Energy プロトコルスタックでの IAR サポート終了に伴う修正。

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. 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、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

RENESAS

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/

© 2017 Renesas Electronics Corporation. All rights reserved.
Colophon 4.1