Bluetooth® Low Energy プロトコルスタック RL78/G14ホストサンプル

R01AN2807JJ0121
Rev.1.21
2022.01.31

要旨

このマニュアルは、ホストサンプルのハードウェア構成およびソフトウェア構成、動作確認の手順、ソフトウェア動作、動作シーケンスについて記載しています。

ホストサンプルは、BLE プロトコルスタック（Modem 構成）を有する RL78/G1D 評価ボードとシリアル接続した Renesas Starter Kit for RL78/G14 上で動作する、BLE プロトコルスタックを制御するためのサンプルプログラムです。

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
サンプルプログラムアプリケーションノート	R01AN1375J	R01AN1375E
rBLE コマンド仕様書	R01AN1376J	R01AN1376E
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/G14		
ユーザーズマニュアル ハードウェア編	R01UH0186J	R01UH0186E
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 4
1.1 環境 4

2. 構成 5
2.1 デバイス構成 5
2.2 ソフトウェア構成 6
2.3 周辺機能構成 8
2.4 ファイル構成 10

3. 手順 12
3.1 準備手順 12
3.1.1 Host MCU 12
3.1.2 BLE MCU 14
3.1.3 Host MCU－BLE MCU接続 17
3.1.4 スマートフォン 17
3.1.5 UART接続方式の変更 18
3.2 確認手順 19
3.2.1 Androidデバイス 19
3.2.2 iOSデバイス 21
3.3 変更手順 23
3.3.1 コンパイル・オプションの設定 23
3.3.2 コード生成ツールを使用した低レベル周辺ドライバの変更 23

4. 動作 24
4.1 コマンド動作・イベント動作 24
4.2 メインループ動作 24
4.3 UART 2線分岐接続方式 25
4.3.1 送信動作 25
4.3.2 受信動作 26
4.3.3 応用回路例 27

5. シーケンス 28
5.1 メインシーケンス 28
5.2 Step1. rBLE Initializeシーケンス 29
5.3 Step2. GAP Initializeシーケンス 29
5.4 Step3. Broadcastシーケンス 30
5.5 Step4. Connectionシーケンス 30
5.6 Step5. Profile Enableシーケンス 31
5.7 Step6. Remote Device Checkシーケンス 31
5.8 Step7. Pairingシーケンス 32
5.9 Step8. Start Encryptionシーケンス 34
5.10 Step9. Profile Communicationシーケンス 35
5.11 Step10. Disconnectionシーケンス 36

6. 付録 37

6.1 ROMサイズ・RAMサイズ 37
6.2 参考文献 37
6.3 用語説明 38

1. 概要

このマニュアルは、ホストサンプルのハードウェア構成およびソフトウェア構成、動作確認の手順、ソフトウェア動作、動作シーケンスについて記載しています。

ホストサンプルは、BLE プロトコルスタック（Modem 構成）を有する RL78/G1D 評価ボードとシリアル接続した Renesas Starter Kit for RL78/G14（以降、RSK）上で動作する、BLE プロトコルスタックを制御するためのサンプルプログラムです。RSK と RL78/G1D 評価ボードの間のシリアル通信は、UART 2 線接続方式[注 1]と UART 2 線分岐接続方式[注 2]をサポートします。

BLE プロトコルスタックの API の詳細につきましては、Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル（R01UW0088）を参照してください。

【注】 1. UART のデータ信号線である TxD、RxD を使用する通信方式です。
2. TxD、RxD に加えて、Host MCU がデータ送信時に BLE MCU を起床させるための信号として、Host MCU の TxD を分岐して BLE MCU の WAKEUP と接続する通信方式です。Host MCU と BLE MCU 間の接続については、「4.3.3 応用回路例」参照してください。

1.1 環境

ホストサンプルのビルドと動作確認で使用する環境を示します。

- ハードウェア環境
  - ホストマシン
    - 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）
  - デバイス
    - Renesas Starter Kit for RL78/G14
    - Renesas BLE Evaluation Board for RL78/G1D
    - Smart Phone（Android デバイスまたは iOS デバイス）
- 使用ツール
    - Renesas オンチップデバッギングエミュレータ E1
- ソフトウェア環境
    - Windows7 Service Pack1
    - e^2 studio V4.3.1.001 / RL78 Family C Compiler Package V1 V1.03.00
      または Renesas CS+ for CC V4.00.00 / RL78 Family C Compiler Package V1 V1.03.00
      または Renesas CS+ for CA, CX V3.02.00 / Renesas CA78K0R V1.72
    - Renesas Flash Programmer v3.02.00

2. 構成

2.1 デバイス構成

図 2-1にホストサンプルを使用する際のデバイス構成図を示します。

Host MCU である RL78/G14 と BLE MCU である RL78/G1D を UART で接続し、Local Device とします。Android デバイスまたは iOS デバイスのスマートフォンを準備し、Remote Device とします。

Local Device は Slave として、Remote Device は Master として動作します。RL78/G14 は双方向の UART 通信によって RL78/G1D の BLE プロトコルスタックを制御することで、スマートフォンとの BLE 通信を行います。

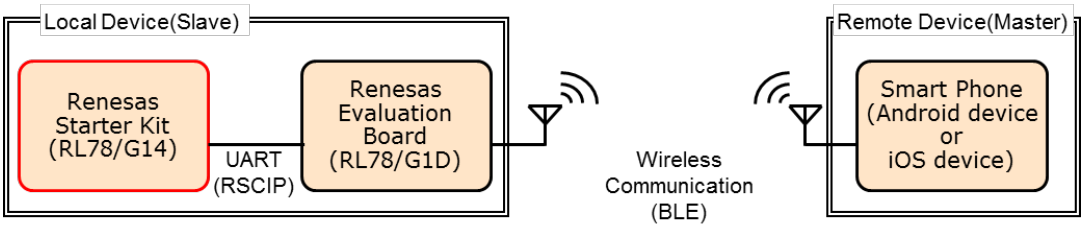

図 2-1 デバイス構成

ホストサンプルの概要は以下のとおりです。

- ✓ rBLE API を使用し以下の動作を実行。
  - ✧ 電源投入後、Broadcast 開始から接続までを自動実行。
  - ✧ Remote Device から要求があれば、ペアリング/暗号化を実行。
  - ✧ 接続完了後、SCP(Sample Custom Profile)を有効化。
  - ✧ Remote Device からの Notification 許可後、ADC 値を 1 秒おきにデータ送信。
- ✓ 動作シーケンス実行のための簡易的なスケジューラのみ実装。
- ✓ 実行すべき処理がない期間は、RL78/G14 を STOP モードに遷移。
- ✓ 対向デバイスは、スマートフォン（Android デバイスまたは iOS デバイス）を想定。

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。

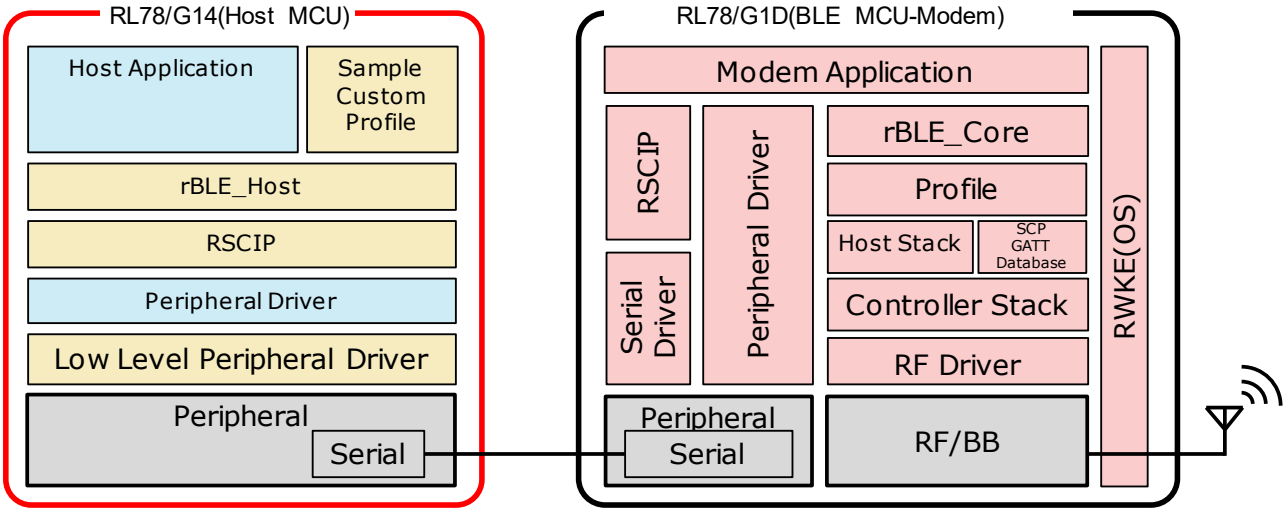

図 2-2　ソフトウェア構成

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、システムを制御するためのホストアプリケーション、GATT API を使用した Sample Custom Profile で構成されます。

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 プロトコルスタックに含まれており、コードファイルが提供されます。ソフトウェア開発時は、BLE プロトコルスタックが提供する最新のコードファイルをご使用ください。

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
Host Application	rBLE の初期化 rBLE コマンドの実行スケジューリング rBLE イベントコールバックの登録	コーディングが必要
Sample Custom Profile (SCP)	GATT API を使用した独自プロファイル	コーディング不要 (ソースコード提供) 注 1
rBLE_Host	rBLE API 提供 イベントコールバックの実行	コーディングが不要 (ソースコード提供) 注 1
RSCIP	シリアル通信プロトコルの制御	コーディングが不要 (ソースコード提供) 注 1
Peripheral Driver	Host MCU 周辺機能の制御	コーディングが必要
Low Level Peripheral Driver	Host MCU 周辺機能のプリミティブな制御	コーディングが不要 (ツール自動生成) 注 2

【注】 1. ソフトウェア開発用コードファイルは BLE プロトコルスタックが提供。
2. ソフトウェア開発用コードファイルはコード生成ツールが自動生成。

BLE MCU は、RF/BB を制御するための RF ドライバ、Sample Custom Profile の GATT Database、Host/Controller スタック、Profile、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）、Modem アプリケーションで構成されます。これらは BLE プロトコルスタック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
Modem Application	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
Peripheral Driver/Serial Driver	BLE MCU 周辺機能の制御
rBLE_Core	rBLE_API 提供
Profile	プロファイル機能の提供
Host Stack	GAP、GATT、SM、L2CAP 機能の提供
SCP GATT Database	Sample Custom Profile の GATT Database
Controller Stack	LL 機能の提供

2.3 周辺機能構成

図 2-3に RL78/G14 および RSK for RL78/G14 で使用する周辺機能構成図を示します。

RL78/G14 の周辺機能ではシリアル・アレイ・ユニット、12 ビット・インターバル・タイマ、A/D コンバータを使用します。RSK の周辺機能では可変抵抗、LCD、LED、SW を使用します。また、Host MCU が rBLE を使用するために最低限必要とする周辺機能はシリアル・アレイ・ユニット、12 ビット・インターバル・タイマです。

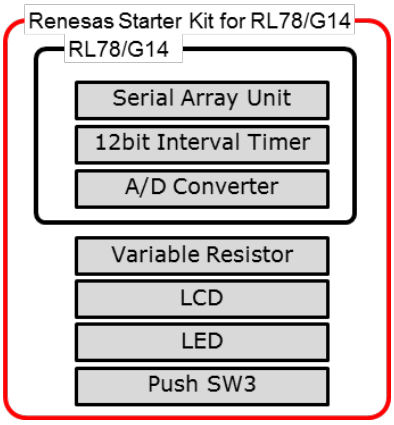

図 2-3 周辺機能構成

表 2-3 周辺機能

周辺機能	用途	必要性[注]
Serial Array Unit	MCU 間通信(SNOOZE 可能な UART2 または UART0)	必須
12bit Interval Timer	UART タイムアウト監視および Profile Notification トリガ	必須
A/D Converter	Notification データ生成	任意
Variable Resistor(RV1)	Notification データ生成(ADC と接続)	任意
LCD(LCD)	BLE コマンド/BLE イベント表示およびペアリング中の PassKey 表示	任意
LED(LED0)	MCU 状態表示(点灯：RUN 状態、消灯：STOP 状態)	任意
SW(SW3)	BLE 接続切断トリガ	任意

【注】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。

CS+のコード生成ツールで設定する RL78/G14 周辺機能の主な設定を示します。

表 2-4 周辺機能設定

周辺機能		用途
クロック発生回路 └クロック	動作モード	高速メイン・モード 2.7(V) ≦VDD≦5.5(V)
	メインシステムクロック(fMAIN)	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	24(MHz)
	サブシステムクロック(fSUB)	動作 XT1 発振(fXT) 32.768(kHz) 低消費発振 STOP,HALT モード時のクロック供給許可
	RTC,インターバル・タイマ動作クロック	32.768(kHz)
	CPU と周辺クロック(fCLK)	24000(kHz)
割り込み └外部割り込み	INT10	INTP10 使用 立下りエッジ検出 低優先
シリアル └SAU1 └チャネル	チャネル 0	UART2 送信/受信機能
シリアル └SAU1 └UART2 └受信	データ・ビット長	8 ビット
	データ転送方向	LSB
	パリティ	パリティなし
	ストップ・ビット長	1 ビット
	受信データ・レベル	標準
	転送レート	4800(bps)
	割り込み	受信完了割り込み設定(INTSR2) 高優先
	コールバック機能	受信完了 エラー
シリアル └SAU1 └UART2 └送信	転送モード	単発モード
	データ・ビット長	8 ビット
	データ転送方向	LSB
	パリティ	パリティなし
	ストップ・ビット長	1 ビット
	送信データ・レベル	標準
	転送レート	4800(bps)
	割り込み	送信完了割り込み設定(INTST2) 低優先
	コールバック機能	送信完了
A/D コンバータ	A/D コンバータ動作	使用する
	コンパレータ動作	停止
	分解能	8 ビット
	トリガ・モード	ソフトウェア・トリガ・モード
	動作モード	ワンショット・セレクト・モード
	割り込み	A/D の割り込み許可(INTAD) 低優先
インターバル・タイマ	インターバル・タイマ	使用する
	インターバル時間	10(ms)
	割り込み	インターバル信号検出(INTIT) 低優先

2.4 ファイル構成

ホストサンプルのファイル構成を示します。

（R）は BLE プロトコルスタック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 プロトコルスタックが提供する最新のコードファイルをご使用ください。

HostSampleRL78G14		
├──Platform		
│ ├──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
│ │ └──lcd		
│ │ lcd.c		LCD ドライバ・コードファイル
│ │ lcd.h		LCD 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
│ ll.h	(R)	低レベルマクロ・ヘッダファイル
│ types.h	(R)	タイプ定義・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
│ iodefine.h		レジスタアクセス用マクロ定義・ヘッダファイル
├──rBLE		
│ ├──sample_app		
│ │ app.c		ホストアプリケーション・コードファイル
│ ├──sample_profile		
│ │ └──scp		
│ │ scps.c	(R)	SCP Server API・コードファイル
│ ├──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
│ ├──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
│ └──include		
│ │ db_handle.h	(R)	データベースハンドル・ヘッダファイル
│ │ 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
│ │ rble_app.h	(R)	ホストアプリケーション・ヘッダファイル
│ └─host		
│ rble_host.h	(R)	rBLE_Host・ヘッダファイル
└─project		
CS_CCRL		CS+ for CC 向けプロジェクト
CubeSuite		CS+ for CA,CX 向けプロジェクト
e2studio		e²studio 向けプロジェクト

各プロジェクトのディレクトリは、それぞれの開発環境が提供するコード生成ツールを使用して生成した以下の低レベル周辺ドライバを含みます。各プロジェクトのドライバは、割り込みハンドラの宣言などの実装方法が異なりますが、動作は同じです。

3. 手順

3.1 準備手順

3.1.1 Host MCU

Host MCU である RL78/G14 の準備手順を示します。UART 接続方式を変更する場合は、3.1.5章を参照してください。

まず、HostMCU 上で動作するファームウェアのビルドを行います。使用する開発環境によって、手順が異なります。

(1) e²studio

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 3-1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 3-1に示すパスにファームウェアファイルが生成されます。

(2) CS+

1. 表 3-1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 3-1に示すパスにファームウェアファイルが生成されます。

表 3-1 Host MCU のプロジェクトファイルとファームウェアファイルの生成ディレクトリ

e²studio		
	Project Directory	HostSampleRL78G14¥project¥e2studio¥HostSample
	Firmware File	HostSampleRL78G14¥project¥e2studio¥HostSample¥DefaultBuild¥HostSample.hex
CS+ for CC		
	Project File	HostSampleRL78G14¥project¥CS_CCRL¥RSKRL78G14.mtpj
	Firmware File	HostSampleRL78G14¥project¥CS_CCRL¥HostSample¥DefaultBuild¥HostSample.hex
CS+ for CA,CX		
	Project File	HostSampleRL78G14¥project¥CubeSuite¥RSKRL78G14.mtpj
	Firmware File	HostSampleRL78G14¥project¥CubeSuite¥HostSample¥DefaultBuild¥HostSample.hex

次に起動方法を示します。

1. 表 3-2を参照して 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]でファームウェア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 3-2 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

3.1.2 BLE MCU

BLE MCU である RL78/G1D の準備手順を示します。UART 接続方式を変更する場合は、3.1.5章を参照してください。

※電源は、AC 電源アダプタまたは USB 経由による電源供給を選択できます。

※ファームウェアファイルは、BLE プロトコルスタックに含まれる RL78_G1D_CM(SCP).hex を使用することも可能です。本ファイルを使用する場合は、手順 8 から開始してください。ただし、手順 2 の設定変更がないため、iOS デバイスとの Pairing は実行されません。

まず、ソースコードを準備します。

1. 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。使用するコンパイラによってコピー先のディレクトリが異なります。
    - 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. Remote Device とのペアリングを行うには次のように変更します。ただし、本変更は、必須ではありません。prf_config.c を開き、[Sample Custom Notify Cfg Value]を検索、検索ヒット直下の Attribute Permission 設定(RBLE_GATT_PERM_RD|RBLE_GATT_PERM_WR)を (RBLE_GATT_PERM_RD|RBLE_GATT_PERM_WR_UNAUTH)に変更し、Sample Custom Profile の Notify Configuration を Write permission (Unauthentication Required)に変更します。これにより Remote Device からの Notification 設定コマンドに対してセキュリティが設定されます。

次にビルドを行います。ビルド手順は、使用する開発環境によって異なります。

(1) e²studio

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 3-3**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」上で右クリックし、「Renesas Tool Settings」を選択し、「ツール設定」→「Compiler」→「ソース」→「定義マクロ」に「USE_SAMPLE_PROFILE」を追加します（noUSE_SAMPLE_PROFILE が定義されている場合は、”no”を削除します。）。
6. 「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。
7. 表 3-3に示すパスにファームウェアファイルが生成されます。

(2) CS+

1. 表 3-3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」で rBLE_emb サブプロジェクトのビルドツールを右クリックし、「プロパティ」を選択、「コンパイル・オプション」タブの「プリプロセス」→「定義マクロ」で「USE_SAMPLE_PROFILE」を定義します。（noUSE_SAMPLE_PROFILE が定義されている場合は、”no”を削除します。）
3. 「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。
4. 表 3-3に示すパスにファームウェアファイルが生成されます。

表 3-3 BLE MCU のプロジェクトファイルとファームウェアファイルの生成ディレクトリ

e²studio		
	Project Directory	RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem
	Firmware File	RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm¥DefaultBuild¥rBLE_Mdm_CCRL.hex
CS+ for CC		
	Project File	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CS_CCRL¥BLE_Modem¥BLE_Modem.mtpj
	Firmware File	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CS_CCRL¥BLE_Modem¥rBLE_Mdm¥DefaultBuild¥rBLE_Mdm_CCRL.hex
CS+ for CA,CX		
	Project File	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CubeSuite¥BLE_Modem¥BLE_Modem.mtpj
	Firmware File	RL78_G1D¥Project_Source¥renesas¥tools¥project¥CubeSuite¥BLE_Modem¥rBLE_emb¥DefaultBuild¥rBLE_emb.hex

最後に生成したファームウェアの書き込みを行います。

1. 表 3-4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。[操作]タブの[プログラムファイル]で上記で生成したファームウェアファイルを選択します。
7. [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。
8. 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 3-4　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

3.1.3 Host MCU－BLE MCU 接続

Host MCU と BLE MCU の接続手順を示します。

【注】UART2 線分岐接続方式の接続については、「4.3.3 応用回路例」参照してください。

1. 表 3-5を参照して RSK の端子と RL78/G1D 評価ボードの端子を接続します。
2. RL78/G1D 評価ボードに電源供給を開始します。
3. RSK に電源供給を開始します。

表 3-5 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

3.1.4 スマートフォン

スマートフォンの準備手順を示します。

Remote Device とする Android デバイスまたは iOS デバイスに以下のアプリをインストールします。

- (Android デバイス) “BLE Scanner:Read,Write,Notify” - Pixel’s Perception
  https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=ja
- (iOS デバイス) “GATTBrowser” - Renesas Electronics
  https://itunes.apple.com/jp/app/gattbrowser/id1163057977?mt=8

3.1.5 UART 接続方式の変更

UART 接続方式を変更する場合のソースファイル変更箇所を示します。

(1) Host MCU

ホストサンプルの UART 接続方式は uart.h の下記マクロで選択します。

表 3-6 uart.h 変更方法

マクロ	説明
SERIAL_U_DIV_2WIRE	0 : UART 2 線接続方式 <デフォルト設定> 1 : UART 2 線分岐接続方式

(2) BLE MCU

BLE MCU ファームウェアの UART 接続方式は BLE プロトコルスタックに含まれる serial.h と wakeup.c の下記マクロで選択します。

表 3-7 serial.h 変更方法

マクロ	説明
SERIAL_U_2WIRE (1) SERIAL_U_DIV_2WIRE (0)	UART 2 線接続方式の選択: <デフォルト設定> SERIAL_U_2WIRE を(1)にしてください。その他のマクロは(0)にしてください。 UART 2 線分岐接続方式の選択: SERIAL_U_DIV_2WIRE を(1)にしてください。その他のマクロは(0)にしてください。

表 3-8 wakeup.c 変更方法

マクロ	説明
USE_WAKEUP_SIGNAL_PORT (0) /* Modem Setting */	0 : UART 2 線接続方式 <デフォルト設定> 1 : UART 2 線分岐接続方式

3.2 確認手順

3.2.1 Android デバイス

Remote Device に Android デバイスを使用する場合の確認手順を示します。

1. インストール済みの GATTBrowser を起動します。
2. デバイスの検索結果から、Renesas-BLE と表示されたデバイスと接続を開始します。（図 A1 の矢印 1）
3. UUID が 02000000-0000-0000-0000-000000000080 の行を選択します。
4. [Notification Off]を選択し、Notification を有効化します。（図 A3 の矢印 1）
5. パスキー入力ダイアログが表示されたら、RSK の LCD に表示されたパスキー（6 桁の数字）を入力します[注 1][注 2]（図 A4 の矢印 1）。USE_PAIRING_JUSTWORKS マクロを定義している場合、パスキーの入力は求められません。
6. Notification の受信が開始され、16 進で表記されたデータの bit16:31 が 1 秒間隔で更新（インクリメント）されることを確認します。（図 A5 の矢印 1）
7. RSK のポテンショメータをプラスドライバで回転させると、16 進で表記されたデータの bit0:7 の値が変化することを確認します。（図 A5 の矢印 1）
8. [Notification On]を選択し、Notification を無効化します。（図 A5 の矢印 2）
9. 画面上部の[←]を選択し、デバイスとの接続を切断します。（図 A5 の矢印 3）（図 A6 の矢印 1）
10. 手順 2～4 を再度実行し、パスキー入力ダイアログが表示されないこと、Notification の受信が再開されることを確認します。[注 3]
11. 手順 8～9 を再度実行し、デバイスとの接続を切断します。

【注】 1. パスキー入力ダイアログが前面に表示されず、通知パネル上に表示される場合があります。その場合は、通知パネルから「ペア設定リクエスト」を選択して、パスキーの入力を行ってください。
2. 一度ペアリングが成功すると Android デバイスがペアリング情報を保持するため、以降はパスキーの入力が不要になります。ペアリングを再実行する場合には、[設定]→[Bluetooth]→[ペアリングされたデバイス]に表示された Renesas-BLE で[切断]を選択し、Android デバイスに保存されたペアリング情報を削除します。（ペアリング情報の削除は、Android デバイスによって名称や手順が異なる場合があります。）
3. Android 6.0/.6.0.1 には、一度ペアリングしたデバイスとの再接続に失敗する問題があります。再接続を行う場合は、Android 5 系を使用するか、【注 2】に従い、ペアリング情報を削除してから再接続して下さい。
https://code.google.com/p/android/issues/detail?id=202850&can=1&q=pairing%20bonded&colspec=ID%20Status%20Priority%20Owner%20Summary%20Stars%20Reporter%20Opened

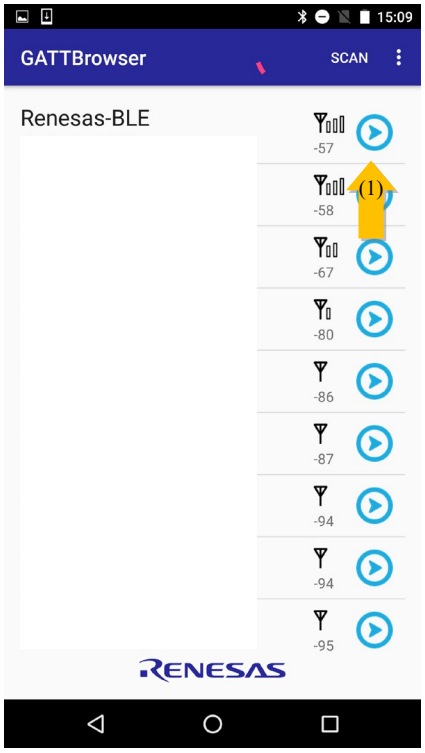

図 A1

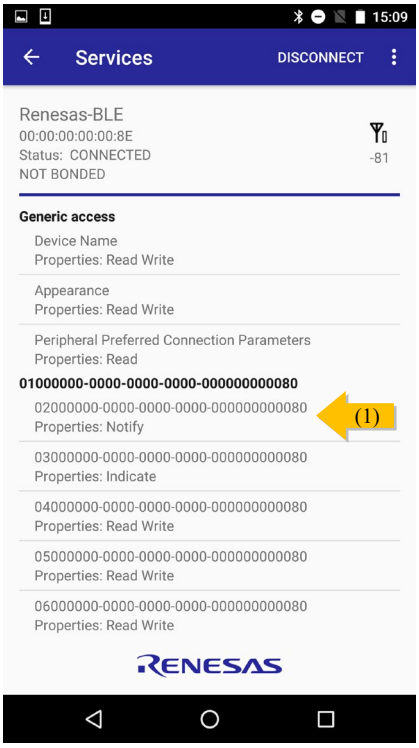

図 A2

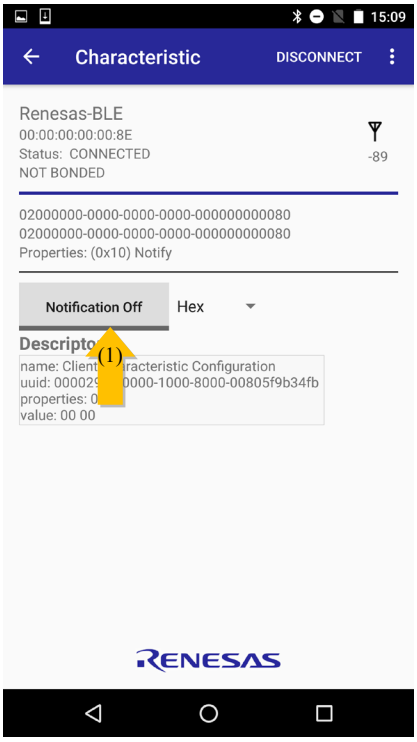

図 A3

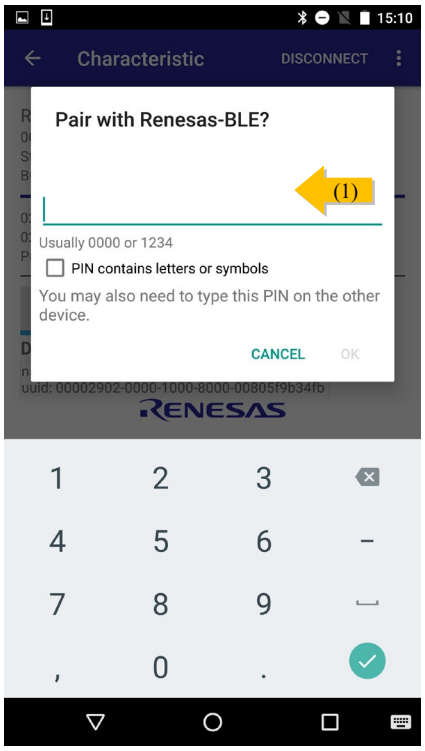

図 A4

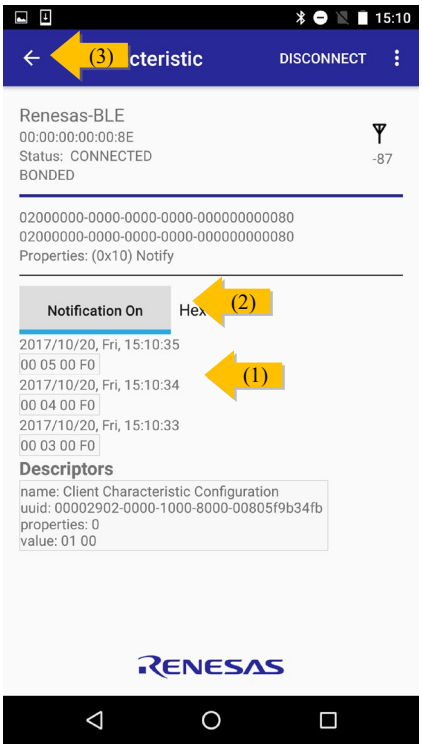

図 A5

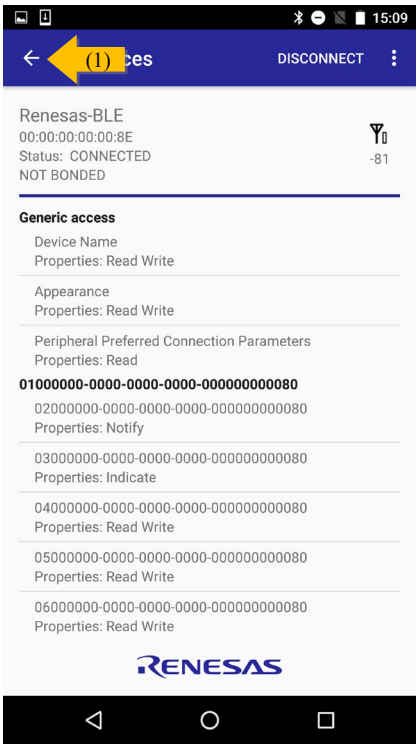

図 A6

3.2.2 iOS デバイス

Remote Device に iOS デバイスを使用する場合の確認手順を示します。

1. インストール済みの GATTBrowser を起動します。
2. デバイスの検索結果から、Renesas-BLE と表示されたデバイスと接続を開始します。（図 i1 の矢印 1）
3. [UUID: 0x02000000-0000-0000-0000-000000000080]を選択します。（図 i2 の矢印 1）
4. [Enable Notification]を押下します。（図 i3 の矢印 1）
5. パスキー入力ダイアログが表示されたら、RSK の LCD に表示されたパスキー(6 桁の数字)を入力します。（図 i4 の矢印 1）注 1 USE_JUSTWORKS_PAIRING マクロを設定している場合、ペアリングの要求を受信したことを示すダイアログが表示されるため、「ペアリング」を選択して要求を受諾します。
6. Notification の受信が開始され、16 進で表記されたデータの bit16:31 が 1 秒間隔で更新（インクリメント）されることを確認します。（図 i5 の矢印 1）注 2
7. RSK のポテンショメータをプラスドライバで回転させると、16 進で表記されたデータの bit0:7 の値が変化することを確認します。（図 i5 の矢印 1）
8. [Disable Notification]を押下します。（図 i5 の矢印 2）
9. [< Services]→[< Back]を押下し、接続を切断します。（図 i5 の矢印 3）（図 i6 の矢印 1）
10. 手順 2～4 を再度実行し、パスキー入力ダイアログが表示されないこと、Notification の受信が再開されることを確認します。
11. 手順 8~9 を再度実行し、デバイスとの接続を切断します。

【注】 1. ペアリングを再実行する場合には、[設定]→[一般]→[Bluetooth]→[デバイス]に表示された Renesas-BLE で[このデバイスの登録を解除]を選択し、iOS デバイスに保存されたペアリング情報を削除します。（ペアリング情報の削除手順は iOS のバージョンで名称や手順が多少異なる可能性があります。）
2. パスキー入力後、Notification の受信が自動的に開始されない場合があります。その場合は、再度[Enable Notification]を押下してください。

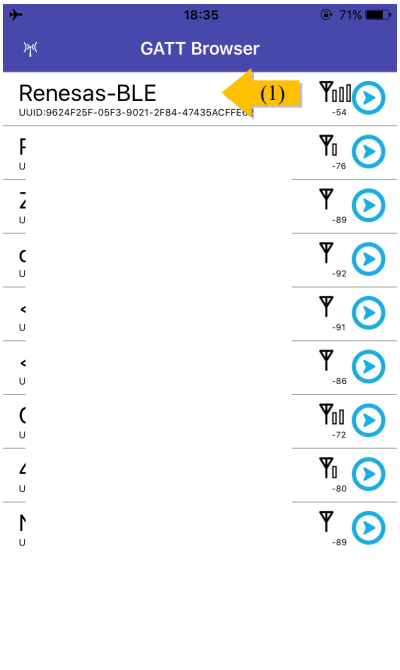

図 i1

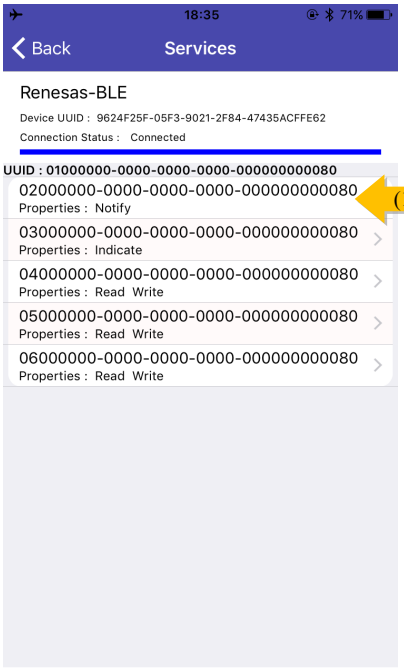

図 i2

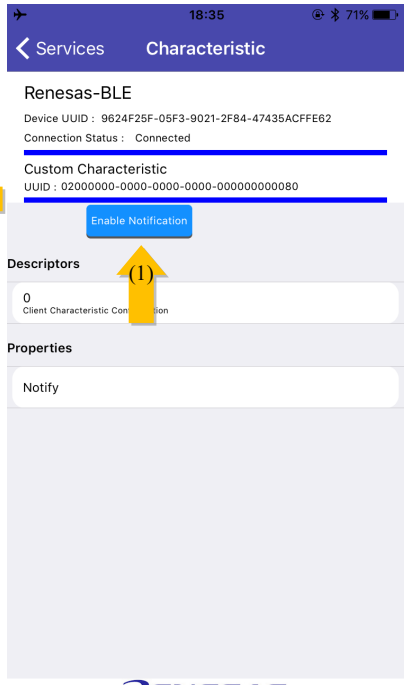

図 i3

図 i4

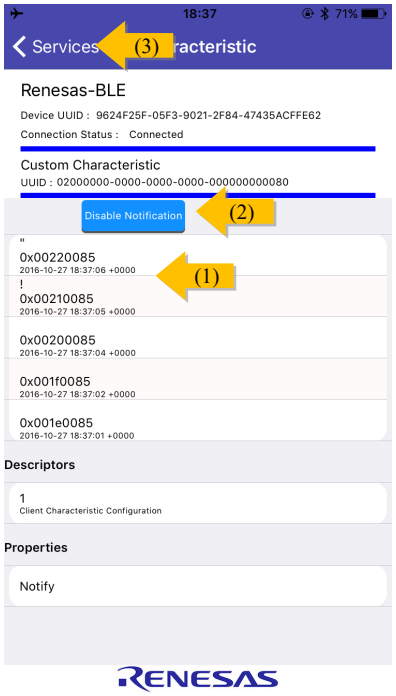

図 i5

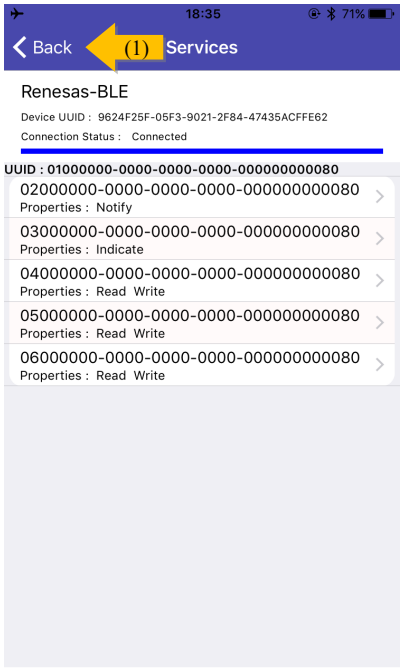

図 i6

3.3 変更手順

3.3.1 コンパイル・オプションの設定

ホストサンプルではコンパイル・オプションを変更することにより、動作を変更することができます。

コンパイル・オプションは、CS+の[プロジェクトツリー]で HostSample サブプロジェクトのビルドツールを右クリックし、[プロパティ]を選択、[コンパイル・オプション]タブの[プリプロセス]→[定義マクロ]で変更してください。

表 3-9 動作変更マクロ

マクロ	マクロ定義時	マクロ未定義時
USE_PAIRING_JUSTWORKS	JustWorks でペアリングを実行	＜デフォルト設定＞ PassKeyEntry でペアリングを実行
USE_RSK_LCD	<デフォルト設定> RSK の LCD を使用	RSK の LCD は使用しない ※LCD を使用しない場合、生成された PassKey も表示されないため、ペアリングは JustWorks で実行してください。
USE_RSK_LED	<デフォルト設定> RSK の LCD を使用	RSK の LED は使用しない
USE_RSK_SW	<デフォルト設定> RSK の SW を使用	RSK の SW は使用しない ※ホストサンプルからの接続の切断はできません。
USE_RSK_ADC	<デフォルト設定> RSK の A/D コンバータを使用	RSK の ADC は使用しない ※コード生成ツールで[A/D コンバータ動作設定]を[使用しない]に設定してください。

3.3.2 コード生成ツールを使用した低レベル周辺ドライバの変更

各開発環境が提供するコード生成ツールを使用して低レベル周辺ドライバの変更を行うことができます。ただし、以下のファイルは、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ので、コード生成ツールにより新たに生成したコードで上書きしないでください。

- r_cg_macrodriver.h

4. 動作

Host Application（以降、APP）と rBLE を中心に、ソフトウェア動作を示します。

4.1 コマンド動作・イベント動作

図 4-1に APP と rBLE によるコマンド動作・イベント動作を示します。

1. APP が rBLE API をコールして、コマンドを発行します。
2. rBLE がコマンドで指定された処理を実行します。
3. rBLE は処理完後、APP コールバック関数をコールして、イベントを通知します。
4. APP は必要ならば、APP コールバック関数で次のコマンド発行を要求します。

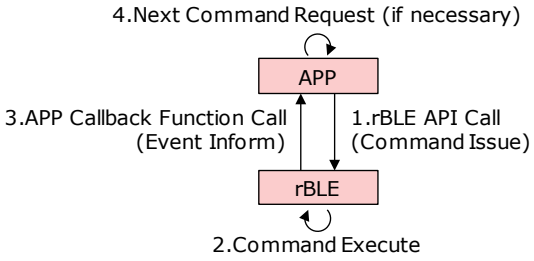

図 4-1 コマンド動作・イベント動作

4.2 メインループ動作

図 4-2にホストサンプルのメインループ動作を示します。

メインループは APP のコマンド動作と rBLE のイベント動作を実現するために、rBLE API コールを行う APP スケジューラと、APP コールバック関数コールを行う rBLE スケジューラ、MCU を STOP モードに遷移させる MCU モード管理処理を実行します。

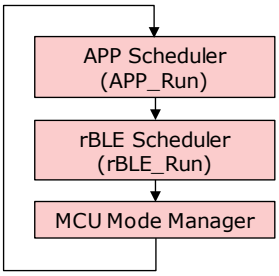

図 4-2 メインループ動作

APP スケジューラは、コマンド要求キューを持ち、コマンド要求キューにコマンド要求がセットされているならば、rBLE API をコールします。

rBLE スケジューラは、イベントキューを持ち、イベントキューにイベントがセットされているならば、APP コールバック関数をコールします。

MCU モード管理処理は、コマンド要求キューとイベントキューに何もセットされていなければ、MCU を STOP に遷移させます。MCU は割り込みによって STOP から復帰します。

4.3 UART 2 線分岐接続方式

UART 2 線分岐接続方式での UART ドライバ通信方法について示します。

4.3.1 送信動作

Host MCU から BLE MCU への送信を行うには、ハンドシェイクを行う必要があります。ハンドシェイクは Host MCU から送信する REQ バイト(0xC0)と、BLE MCU から送信される ACK バイト(0x88)または RSCIP パケットによって行われます。また、ハンドシェイクを行う時にはタイマによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行します。Host MCU の UART ドライバは、このハンドシェイクを行うため、送信状況によって 5 つの状態を持ちます。

表 4-1 UART ドライバ送信状態

状態	説明
T_IDLE	UART ドライバ初期化、RSCIP パケット送信完了
T_REQUESTING	REQ バイト送信中
T_RCV_BF_REQUESTED	ACK バイトの代わりに BLE MCU から RSCIP パケットを受信
T_REQUESTED	REQ バイト送信完了(BLE MCU からの ACK バイト待ち)
T_ACTIVE	RSCIP パケット送信中

Host MCU から BLE MCU への送信は必ず REQ バイトから開始されます。REQ バイトを送信した後、Host MCU は受信状態により次の動作のいずれかに分岐します。

(a) Host MCU が BLE MCU からの RSCIP パケットを受信していない(図 4-3)

(b) Host MCU が BLE MCU からの RSCIP パケットを受信中(図 4-4)

(c) ACK バイト受信タイムアウト(図 4-5)

(a) Host MCU が BLE MCU からの RSCIP パケットを受信していない

この状態は、BLE MCU から RSCIP パケットが送信されておらず、Host MCU から REQ バイトを送信した後、Host MCU が ACK バイトの受信を待っている状態です。BLE MCU は REQ バイトを受信し ACK バイトを送信します。ACK バイトを受信した Host MCU は、BLE MCU に RSCIP パケットを送信します。

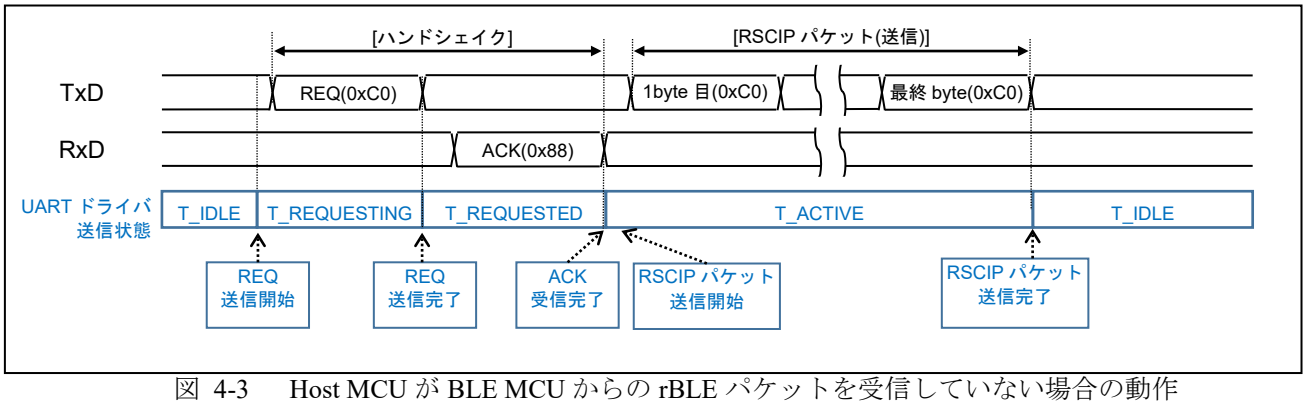

図 4-3 Host MCU が BLE MCU からの rBLE パケットを受信していない場合の動作

(b) Host MCU が BLE MCU からの RSCIP パケットを受信中

この状態は BLE MCU が RSCIP パケットを送信しており、Host MCU は RSCIP パケットを受信している状態です。BLE MCU は REQ を受信しても ACK バイトを返さず、送信している RSCIP パケットを ACK バイトの代わりとします。ホストは BLE MCU からの RSCIP パケットを ACK バイトの代わりとし、BLE MCU に RSCIP パケットを送信します。

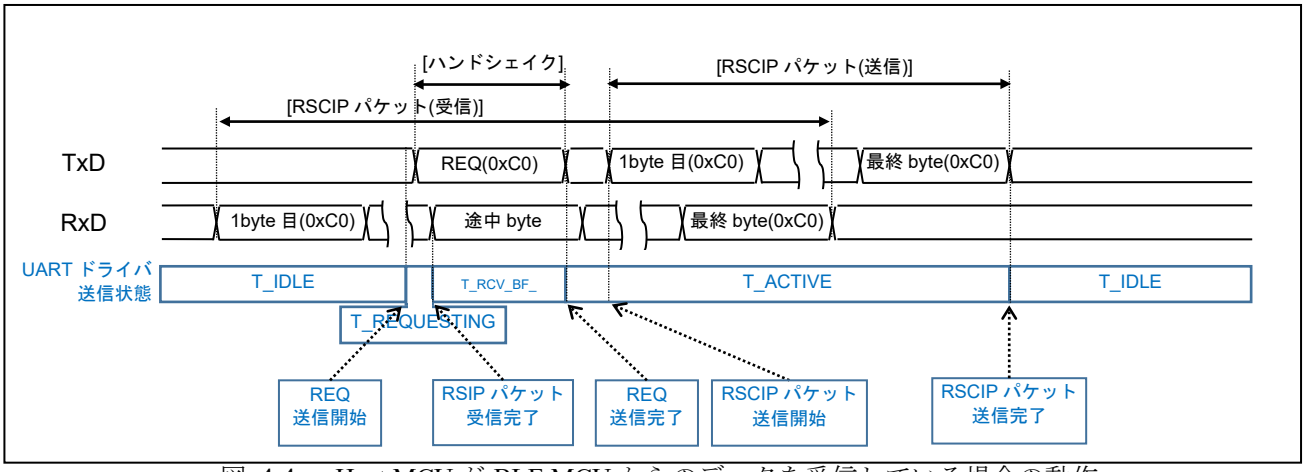

図 4-4 Host MCU が BLE MCU からのデータを受信している場合の動作

(c) ACK バイト受信タイムアウト

REQ バイトを送信した後 Host MCU は、タイムアウトタイマを動作させます。一定時間 ACK バイトを受信できなかった場合、REQ バイトを再送します。

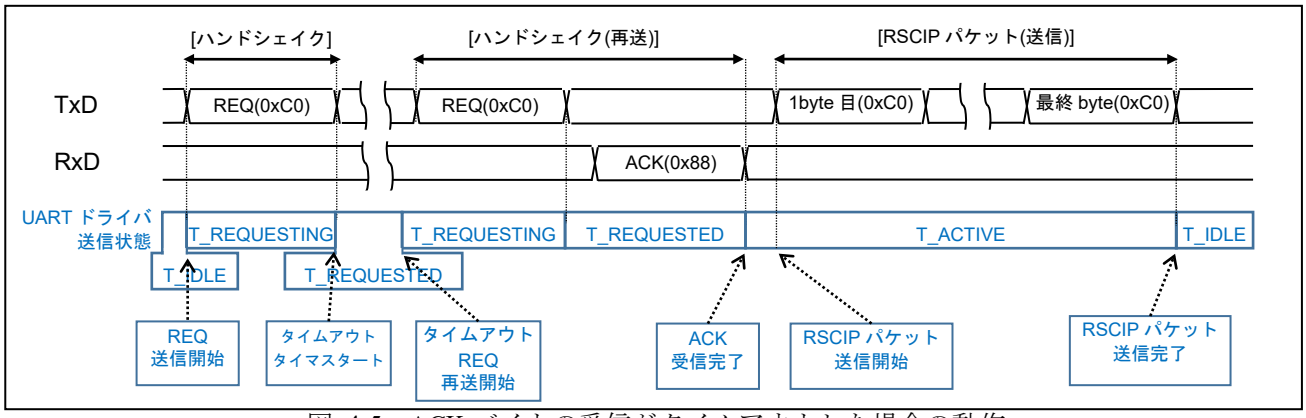

図 4-5 ACK バイトの受信がタイムアウトした場合の動作

4.3.2 受信動作

受信時に UART ドライバの状態遷移はありません。BLE MCU からのデータを受信するために、rBLE_Host から指定されたバイト数で BLE MCU からの RSCIP パケットを待ち受けます。

4.3.3 応用回路例

Host MCU と BLE MCU の UART 2 線分岐接続例を示します。

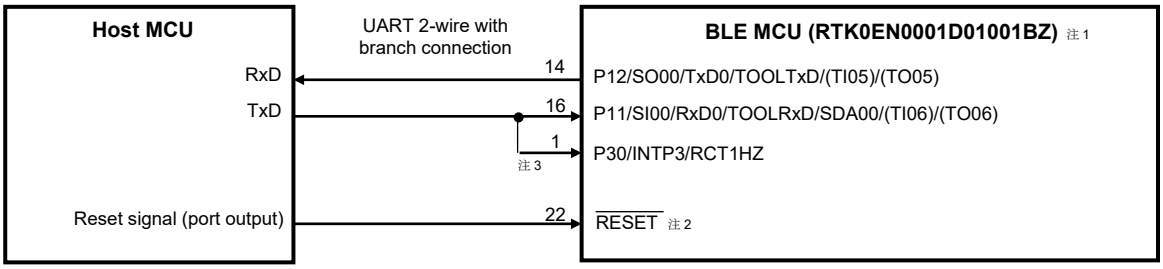

図 4-6 UART 2 線分岐接続方式

【注】 1. Pin 番号は、RL78/G1D 評価ボードの CN4 外部拡張コネクタ番号です。
2. /RESET 端子は、必要に応じてプルアップ／プルダウン抵抗を追加してください（RL78/G1D ユーザーズマニュアル ハードウェア編（R01UH0515）参照）。
3. RL78/G1D 評価ボードの P30/INTP3/RCT1HZ (WAKEUP)に USB の VBUS 検知が割当られています。USB から電源を供給する場合、RSK の TXD を分岐して RL78/G1D 評価ボードの INTP3 に接続しないで下さい。

5. シーケンス

Local Device の Host MCU と BLE MCU、Remote Device であるスマートフォン、また Local Device のソフトウェアである Host MCU の APP と rBLE_Host、BLE MCU の rBLE_Core のシーケンスを示します。

5.1 メインシーケンス

メインシーケンス図では、処理ブロックとして Step1～10 までを定義し、処理ブロックの順序と関連するデバイスまたはソフトウェアの範囲を示します。処理ブロック Step1～10 の詳細は次節以降に記載します。

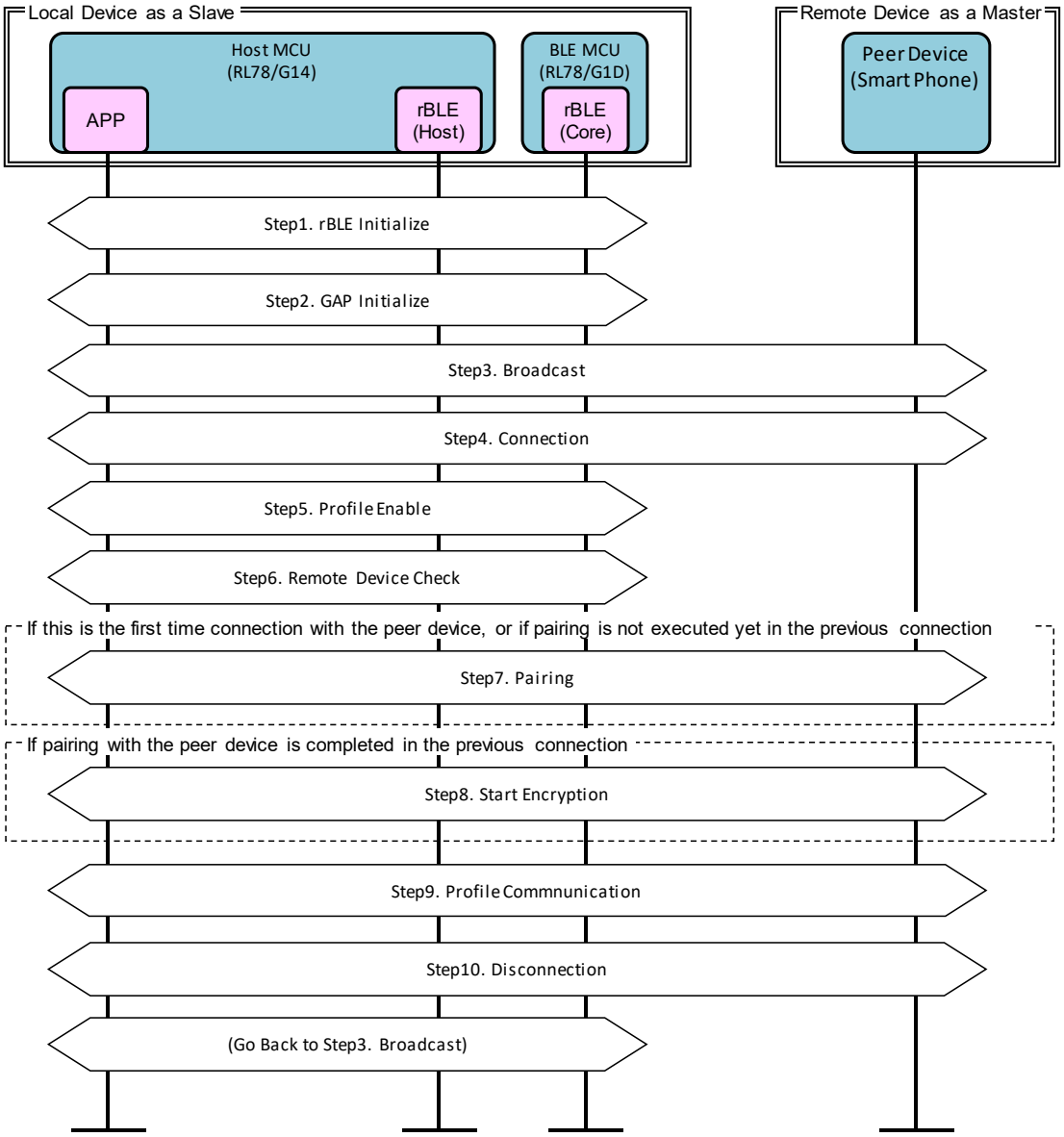

図 5-1 メインシーケンス

5.2 Step1. rBLE Initialize シーケンス

APP は RBLE_Init 関数をコールし、rBLE（rBLE_Host/rBLE_Core）を初期化します。rBLE の初期化が完了し BLE MCU とのシリアル通信が確立されると、rBLE から RBLE_MODE_ACTIVE イベントが通知されます。

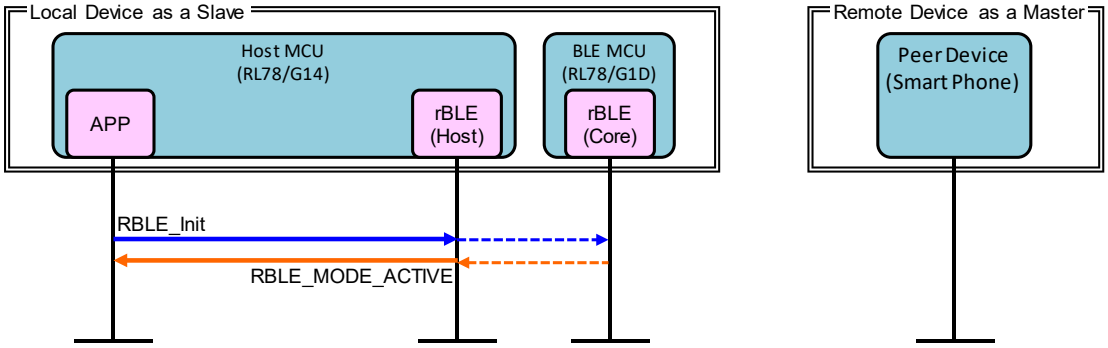

図 5-2 rBLE Initialize シーケンス

5.3 Step2. GAP Initialize シーケンス

APP は RBLE_GAP_Reset 関数をコールし、GAP を初期化します。初期化が完了すると、rBLE から RBLE_GAP_EVENT_RESET_RESULT イベントが通知されます。

APP は RBLE_GAP_Set_Bonding_Mode 関数をコールし、Bonding 許可を設定します。設定が完了すると、rBLE から RBLE_GAP_EVENT_SET_BONDING_MODE_COMP イベントが通知されます。

APP は RBLE_GAP_Set_Security_Request 関数をコールし、セキュリティレベルを設定します。設定が完了すると、rBLE から RBLE_GAP_EVENT_SET_SECURITY_REQUESET_COMP イベントが通知されます。

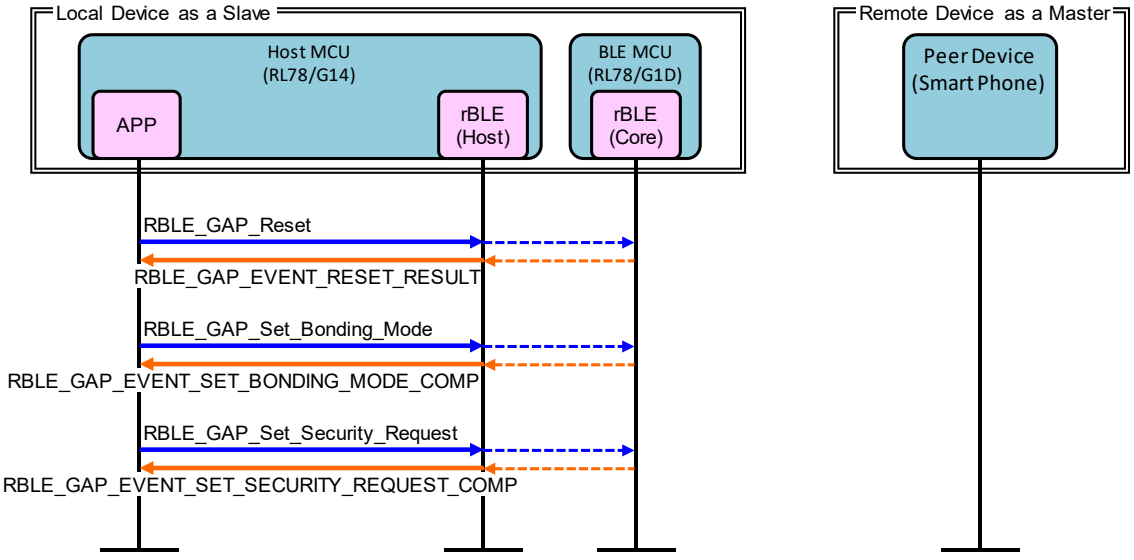

図 5-3 GAP Initialize シーケンス

5.4 Step3. Broadcast シーケンス

Local Device を Slave として接続するための Broadcast を開始します。

APP は RBLE_GAP_Broacast_Enable 関数をコールし、Broadcast を開始します。開始が完了すると、rBLE から RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP イベントが通知されます。

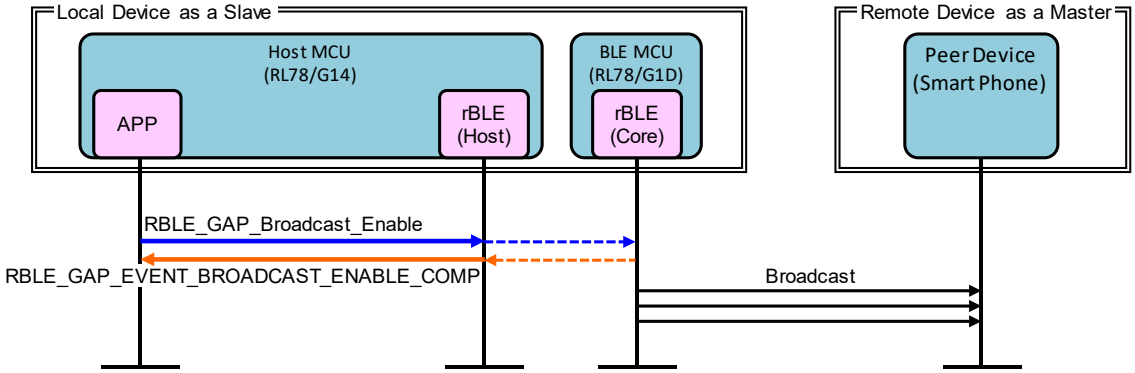

図 5-4 Broadcast シーケンス

5.5 Step4. Connection シーケンス

Local Device からの Broadcast を受信した Remote Device は、接続を要求します。

Remote Device から ConnectionRequest が送信され、Local Device と Remote Device の接続が確立されると、rBLE から RBLE_GAP_EVENT_CONNECTION_COMP イベントが通知されます。

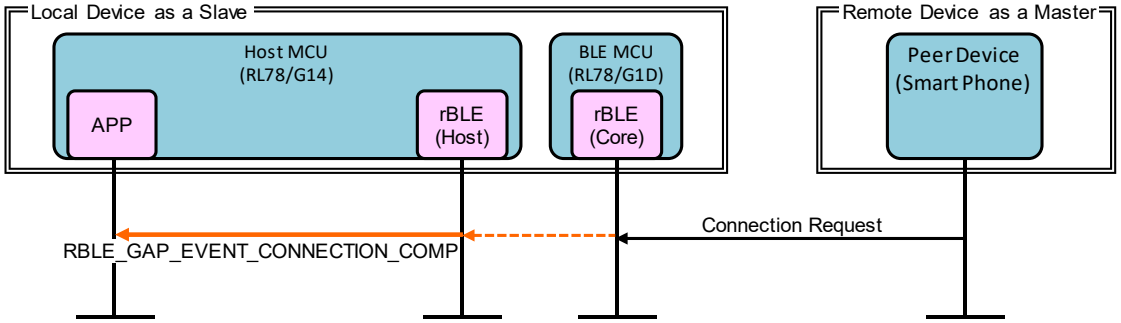

図 5-5 Connection シーケンス

5.6 Step5. Profile Enable シーケンス

データ送信に利用する SCP（Sample Custom Profile）を有効化します。

APP は RBLE_SCP_Server_Enable 関数をコールし、SCP を有効化します。有効化が完了すると、rBLE から RBLE_SCP_EVENT_SERVER_ENABLE_COMP イベントが通知されます。

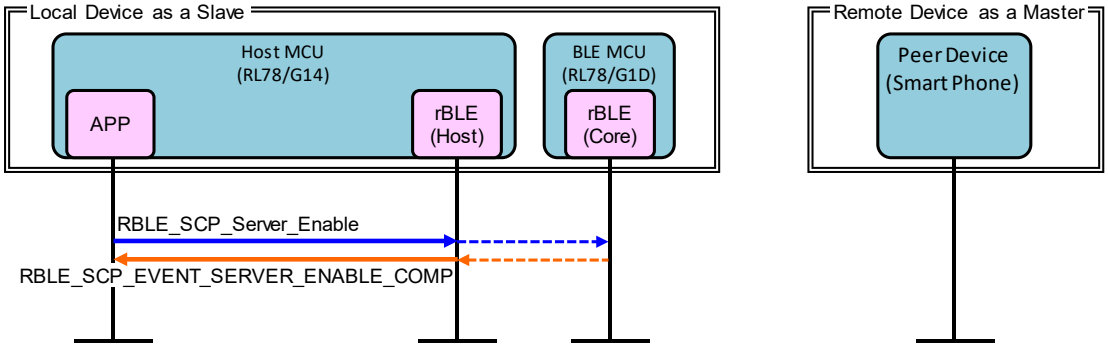

図 5-6 Profile Enable シーケンス

5.7 Step6. Remote Device Check シーケンス

接続完了後は、Remote Device とのセキュリティ情報を確認します。

Remote Device アドレスが Public アドレス、または Resolvable Private Address 以外の Random アドレスの場合、Remote Device に関するセキュリティ情報の要求として、rBLE から RBLE_SM_CHK_BD_ADDR_REQ イベントが通知されます。APP は RBLE_SM_Chk_Bd_Addr_Req_Resp 関数をコールし、保持しているセキュリティ情報の通知またはセキュリティ情報を保持していないことを通知します。

Remote Device アドレスが Resolvable Private Address の場合アドレス解決のための IRK（Identity Resolving Key）の要求として、rBLE から RBLE_SM_IRK_REQ_IND イベントが通知されます。APP は RBLE_SM_Irk_Req_Resp 関数をコールし、IRK を通知または IRK を保持していないことを通知します。アドレスが解決した場合、rBLE から RBLE_GAP_EVENT_RPA_RESOLVED イベントが通知されます。

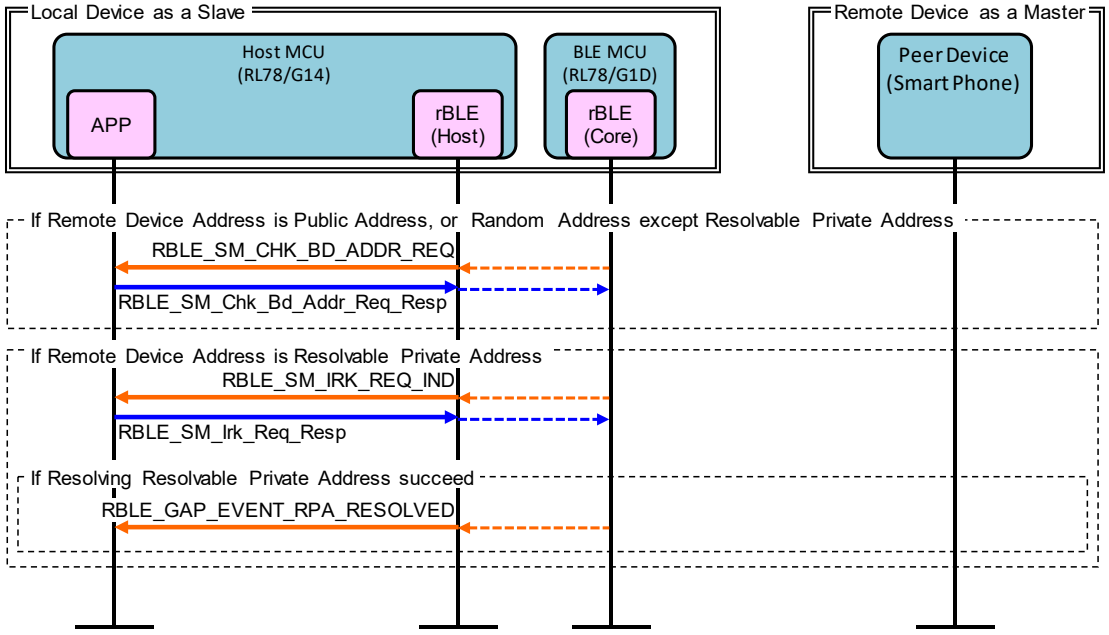

図 5-7 Remote Device Check シーケンス

5.8 Step7. Pairing シーケンス

Remote Device との初回接続または以前の接続にて Pairing を行っていない場合、Remote Device からの Pairing 要求により Pairing シーケンスを開始します。Pairing シーケンスは PHASE1、PHASE2、暗号化開始、PHASE3 で構成されます。

PHASE1 では、Local Device および Remote Device のペアリングフィーチャーを交換します。

Remote Device から Pairing Request が送信されると、rBLE から RBLE_GAP_EVENT_BONDING_REQ_IND イベントが通知されます。APP は RBLE_GAP_Bonding_Response 関数をコールし、Remote Device に Pairing Response を送信します。

PHASE2 では、STK（Short Term Key）を生成します。

TK（Temporary Key）の要求として、rBLE から RBLE_SM_TK_REQ_IND イベントが通知されます。APP は RBLE_SM_Tk_Req_Resp 関数をコールして TK（Temporary Key）を rBLE に通知します。BLE MCU による STK の生成が完了すると、STK を使用した暗号化を開始します。

PHASE3 では、Local Device および Remote Device の暗号化キーを配布します。

LTK（Long Term Key）の要求として、rBLE から RBLE_SM_LTK_REQ_IND イベントが通知されます。APP は RBLE_SM_Ltk_Req_Resp 関数をコールして LTK を通知し、Remote Device に Encryption Information（LTK）を送信します。

Remote Device から Encryption Information（LTK）が送信されると、rBLE から RBLE_SM_KEY_IND イベントが通知されます。

Remote Device から Identity Information（IRK）が送信されると、rBLE から RBLE_SM_KEY_IND イベントが通知されます。

Pairing が完了すると、rBLE から RBLE_GAP_EVENT_BONDING_COMP イベントが通知されます。

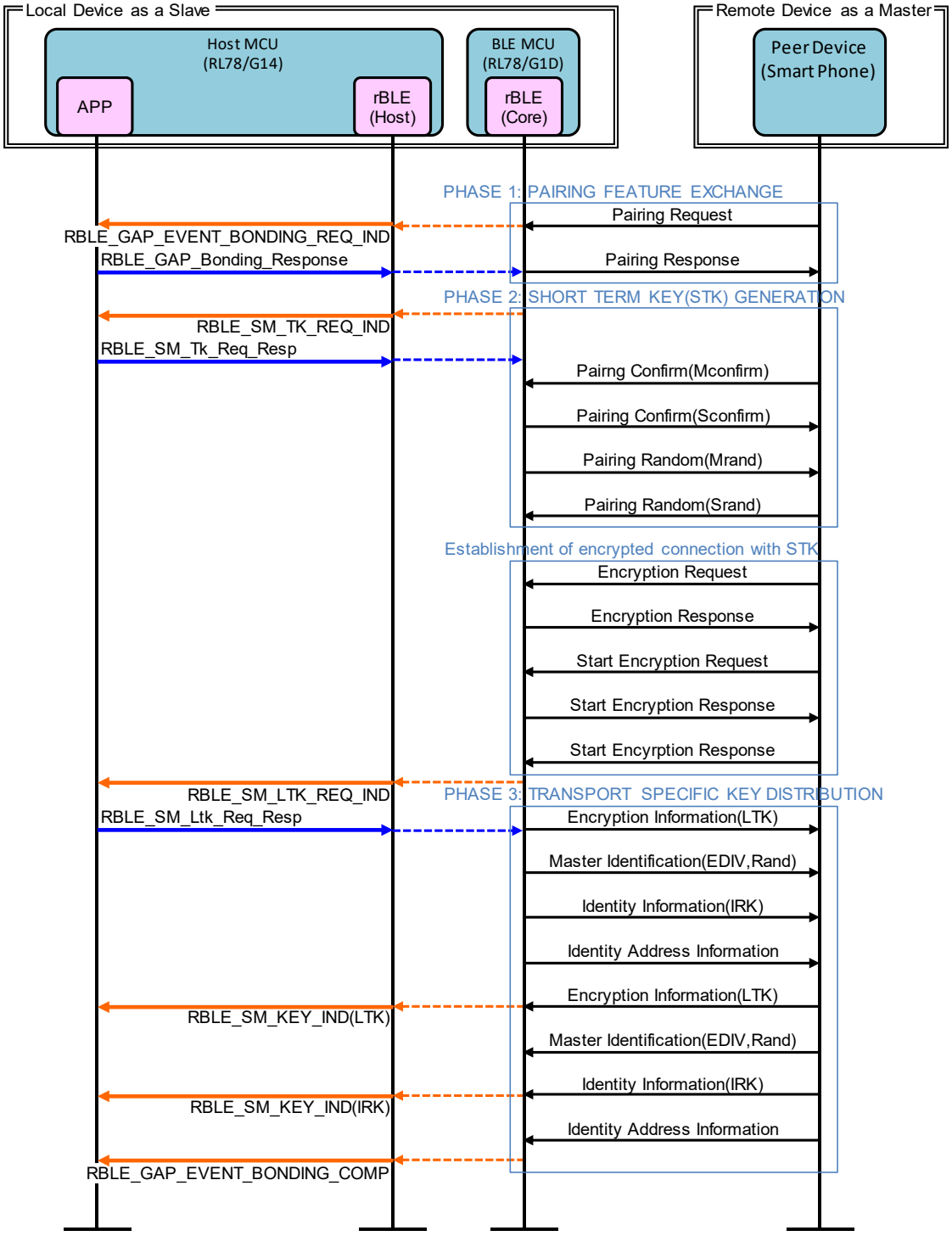

図 5-8 Pairing シーケンス

5.9 Step8. Start Encryption シーケンス

以前の接続で Pairing が完了した場合、LTK（Long Term Key）による暗号化を開始します。

Remote Device から Encryption Request が送信されると、rBLE から RBLE_SM_LTK_REQ_FOR_ENC_IND イベントが通知されます。APP は RBLE_SM_Ltk_Req_Resp 関数をコールして LTK を通知し、Remote Device に Encryption Response を送信します。

BLE MCU は、Remote Device からの Start Encryption Request に対する Start Encryption Response を送信します。

暗号化開始が完了すると、rBLE から RBLE_SM_ENC_START_IND イベントが通知されます。

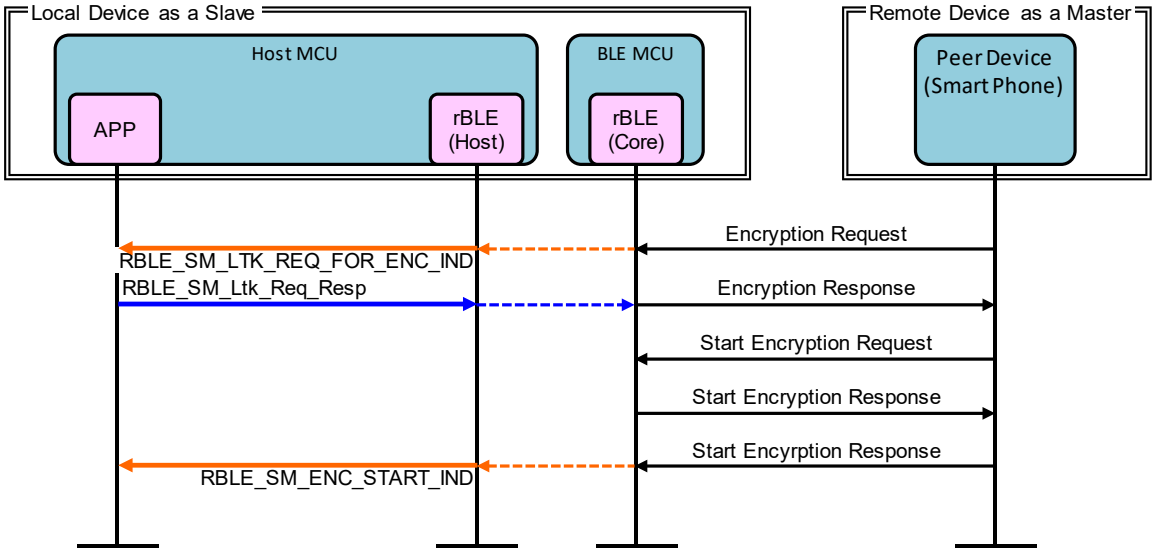

図 5-9 Start Encryption シーケンス

5.10 Step9. Profile Communication シーケンス

SCP（Sample Custom Profile）を利用し Notification によるデータ送信を開始します。

Remote Device から Notification を許可する Write Client Characteristic Configuration が送信されると、rBLE から RBLE_SCP_EVENT_SERVER_CHG_INDNTF_IND イベントが通知されます。

APP はインターバル・タイマ動作を開始し、一定周期で INTIT 割り込みが発生します。INTIT 割り込みが発生すると、A/D 変換を開始し、A/D 変換が完了すると INTAD 割り込みが発生します。INTAD 割り込みが発生すると、RBLE_SCP_Server_Send_Notify 関数をコールし、A/D 変換結果を Notification データとして Remote Device に送信します。

Remote Device から Notification を停止する Write Client Characteristic Configuration が送信されると、rBLE から RBLE_SCP_EVENT_SERVER_CHG_INDNTF_IND イベントが通知されます。APP はインターバル・タイマ動作を停止し、データ送信を停止します。

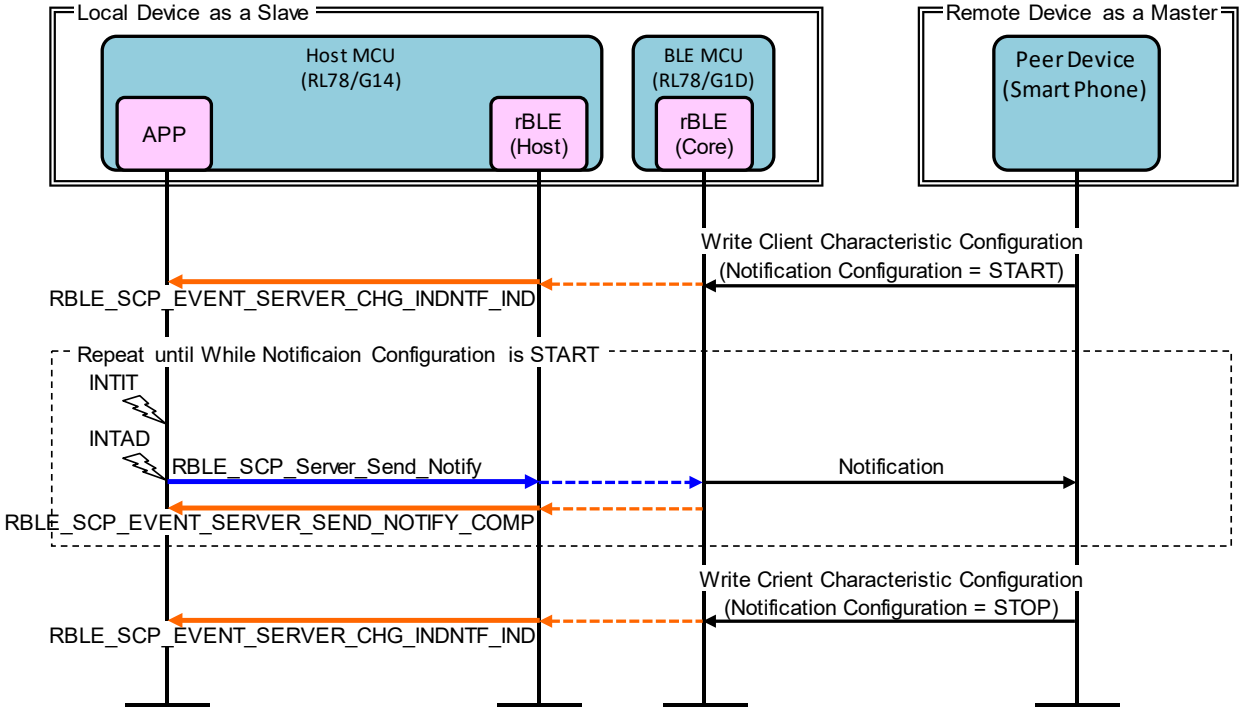

図 5-10 Profile Communication シーケンス

5.11 Step10. Disconnection シーケンス

Remote Device または Local Device から接続の切断を要求します。

Remote Device から接続の切断のための Disconnect が送信されると、切断が完了し RBLE_GAP_EVENT_DISCONNECT_COMP イベントが通知されます。

APP は INTP10 割り込みが発生すると、RBLE_GAP_Disconnect 関数をコールし、Remote Device に Disconnect を送信して Remote Device との接続を切断します。切断が完了すると、rBLE から RBLE_GAP_EVENT_DISCONNECT_COMP イベントが通知されます。

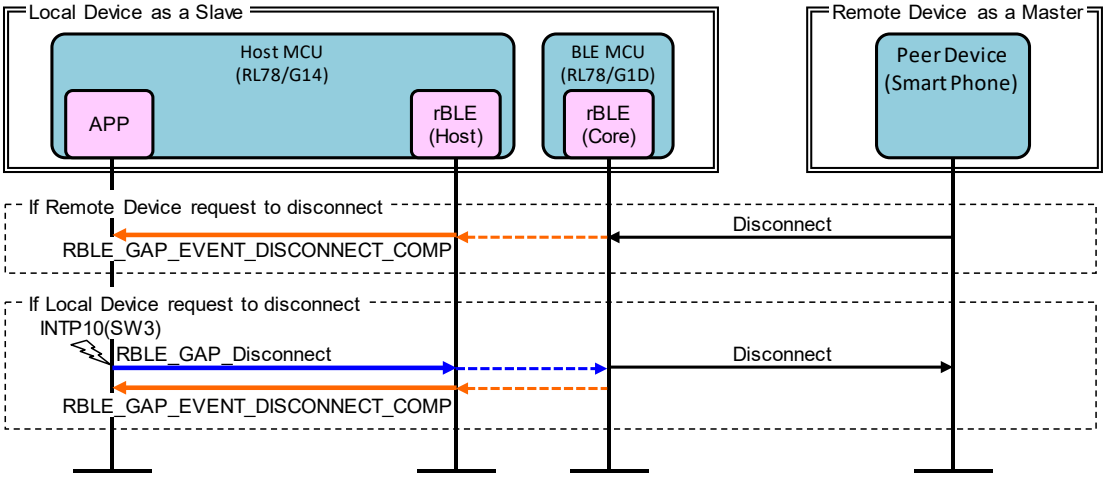

図 5-11 Disconnection シーケンス

6. 付録

6.1 ROM サイズ・RAM サイズ

表 6-1に示すホストサンプルが使用する ROM サイズ、RAM サイズを示します。

ビルド設定：ホストサンプル提供時のデフォルト設定

- 以下のマクロを定義
  - USE_RSK_LCD
  - USE_RSK_LED
  - USE_RSK_SW
  - USE_RSK_ADC
- ペアリング方法として Passkey を使用
- 接続方式として 2 線式を使用

表 6-1 ROM サイズ・RAM サイズ

コンパイラ	ROM (bytes)	RAM (bytes)
CC-RL	20,702	4,524
CA78K0R	29,625	4,736

6.2 参考文献

1. Bluetooth Core Specification v4.2, Bluetooth SIG
2. Bluetooth SIG Assigned Numbers https://www.bluetooth.com/specifications/assigned-numbers/
3. Services UUID https://www.bluetooth.com/specifications/assigned-numbers/
4. Characteristics UUID https://www.bluetooth.com/specifications/assigned-numbers/

6.3 用語説明

用語	英語	説明
サービス	Service	サービスは GATT サーバから GATT クライアントへ提供され、GATT サーバはインタフェースとしていくらかの特性を公開します。 サービスは公開された特性へのアクセス手順について規定します。
プロファイル	Profile	1 つ以上のサービスを使用してユースケースの実現を可能にします。使用するサービスは各プロファイルの仕様にて規定されます。
特性	Characteristic	特性はサービスを識別する値で、各サービスにて公開する特性やそのフォーマットが定義されます。
ロール	Role	役割。それぞれのデバイスが、プロファイルやサービスで規定される役割を果たすことで、ユースケースの実現が可能になります。
コネクションハンドル	Connection Handle	リモートデバイスとの接続を識別するための Controller スタックによって決定されるハンドルです。ハンドルの有効範囲は 0x0000～0x0EFF です。
UUID	Universally Unique Identifier	一意に識別するための識別子です。BLE 規格ではサービスや特性等を識別するために 16bit の UUID が定義されています。
BD アドレス	Bluetooth Device Address	Bluetooth デバイスを識別するための 48bit のアドレスです。BLE 規格ではパブリックアドレスとランダムアドレスが規定されており、少なくともどちらか一方をサポートする必要があります。
パブリックアドレス	Public Address	IEEE に登録し割り当てられた 24bit の OUI(Organizationally Unique Identifier)を含むアドレスです。
ランダムアドレス	Random Address	乱数を含むアドレスで、以下の 3 つに分類されます。 スタティックアドレス Non-resolvable private アドレス Resolvable private アドレス
スタティックアドレス	Static Address	上位 2bit は共に 1 で、残 46bit は全てが 1 または 0 ではない乱数からなるアドレスです。電源断まではそのスタティックアドレスを変更できません。
Non-resolvable private アドレス	Non-resolvable private Address	上位 2bit は共に 0 で、残 46bit は全てが 1 または 0 ではない乱数からなるアドレスです。スタティックおよびパブリックアドレスと等しくてはなりません。 短い期間でアドレスを変更することで攻撃者からの追跡を困難にする目的で使用されます。

Resolvable private アドレス	Resolvable private Address	IRK と 24bit の乱数から生成されるアドレスです。上位 2bit は 0 と 1、上位の残 22bit は全てが 1 または 0 ではない乱数で、下位 24bit は IRK と上位の乱数を元に計算されます。 短い期間でアドレスを変更することで攻撃者からの追跡を困難にする目的で使用されます。 IRK を対向機に配布することで、対向機はその IRK を使用してデバイスを特定することが可能です。
Broadcaster	Broadcaster	GAP のロールで、Advertising データを送信します。
Observer	Observer	GAP のロールで、Advertising データを受信します。
Central	Central	GAP のロールで、物理リンクの確立を行います。Link Layer では Master と呼ばれます。
Peripheral	Peripheral	GAP のロールで、物理リンクの確立を受け入れます。Link Layer では Slave と呼ばれます。
Advertising	Advertising	接続確立や、データ送信の目的のために特定チャネル上でデータを送信します。
Scan	Scan	Advertising データを受信します。Scan には、ただ受信するのみの Passive Scan と、SCAN_REQ を送信することで追加情報を要求する Active Scan があります。
White List	White List	接続済みやボンディング済みなどの既知デバイスを White List に登録しておくことで、Advertising データや接続要求を受け取ることを許可するデバイスをフィルタリングすることが可能です。
デバイス名	Device Name	Bluetooth デバイスに任意につけられたデバイスを識別するためのユーザフレンドリーな名前です。 BLE 規格では、GAP の特性として GATT サーバによって対向機に公開されます。
Reconnection Address	Reconnection Address	Non-resolvable private アドレスを使用して、短い期間でアドレスを変更する場合、攻撃者だけでなく対向機もデバイスの特定が困難になります。そのため対向機の公開する Reconnection Address 特性に新しい Reconnection Address を設定することで再接続時のアドレスを通知します。
コネクションインターバル	Connection Interval	接続確立後に定期的にデータの送受信を行う間隔です。
コネクションイベント	Connection Event	コネクションインターバルごとにデータの送受信を行う期間です。
スーパービジョンタイムアウト	Supervision Timeout	対向機からの応答がなく、リンクが切断されたとみなすタイムアウト時間です。
Passkey Entry	Passkey Entry	ペアリング方式の一つで、互いのデバイスで 6 桁の数値入力または、一方で 6 桁の数値表示、もう一方でその数値入力を行います。

Just Works	Just Works	ペアリング方式の一つで、ユーザアクションを必要としません。
OOB	OOB	ペアリング方式の一つで、Bluetooth 以外の通信方式で取得したデータを使用してペアリングを行います。
IRK	Identity Resolving Key	Resolvable private アドレスの生成や解決に用いる 128bit のキーです。
CSRK	Connection Signature Resolving Key	データ署名の作成および、受信データの署名の確認に使用される 128bit のキーです。
LTK	Long Term Key	暗号化に使用される 128bit のキーです。使用するキーサイズはペアリング時に同意されたサイズになります。
STK	Short Term Key	キー交換時に暗号化するために使用される 128bit のキーです。TK を用いて生成されます。
TK	Temporary Key	STK 生成に必要となる 128bit のキーです。Just Works の場合は 0、Passkey Entry は入力された 6 桁の数値、OOB は OOB データが TK の値となります。

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/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.05.29	ー	初版発行
1.01	2015.10.30	10 17	2.4 節 ファイル構成：ファイルツリーを修正 表 3-3 端子接続：RSK 端子番号を修正
1.10	2016.08.26	- 4 6 18 25 37	ドキュメントタイトルを変更 「1 概要」に 2 線分岐接続方式のサポートを追加 「2.2 ソフトウェア構成」に Sample Custom Profile を追加 「3.1.5 UART接続方式の変更」にソースプログラム変更方法を追加。 「4.3 UART 2線分岐接続方式」に動作説明を追加。 「6.1 ROMサイズ・RAMサイズ」に 2 線分岐接続方式を追加。
1.20	2016.10.27	10 12 14 21 23 34	「2.4 ファイル構成」を更新 「3.1 準備手順」に各開発環境における手順を追加 「3.2.2 iOSデバイス」で使用するアプリを変更 「3.3.2 コード生成」を追加 「5.9 Step8. Start Encryptionシーケンス」を BLE プロトコルスタック V1.20 で変更された暗号化シーケンスにあわせて修正
1.21	2017.10.20	19	「3.2.1 Androidデバイス」を GATTBrowser を使用した手順に変更
1.21	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。

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　〒100-0004　千代田区大手町2-6-2（日本ビル）

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