RL78/G1D モジュール

モジュール制御ソフトウェア(モジュールファームウェア含む)

R01AN3362JJ0111
Rev.1.11
2022.01.31

要旨

本アプリケーションノートは、RL78/G1D モジュール(RY7011)を制御するソフトウェアについて記載します。

モジュールのソフトウェアは、Bluetooth® low energy プロトコルスタック V1.11(以降、BLE ソフトウェアと記載します)の Modem 構成をベースとして作成されています。モジュールには動作確認用のモジュールファームウェアが搭載され、それを制御する Host MCU 側のプログラムでシステムは構成されます。(ファームウェアとはマイコン装置に固定的に組み込まれたプログラムを指します)

サンプルプログラムとして、PC または RL78/G1D を Host MCU としたプログラムと、モジュールに搭載されたモジュールファームウェアのソースコードを同梱しています。Host MCU のプログラムは、BLE ソフトウェア V1.11 付属のサンプルプログラムと同様にコマンドラインで制御するプログラムです。他 MCU へ移植する際には、Host MCU 向けのサンプルプログラムアプリケーションノートも参照してください。

- RL78/G14 ホストサンプル(R01AN2807)
- RX113 ホストサンプル(R01AN3155)

また、BLE ソフトウェア V1.21 に対応したモジュールファームウェアと Host MCU プログラムのソースコードも参考に同梱しており、サンプルプログラムと BLE ソフトウェアのバージョンは以下のように対応しています。

- Renesas_BLE_Module_V101 : BLE ソフトウェア V1.11 を使用。モジュールに書き込まれているファームウェア。
- Renesas_BLE_Module_V111 : BLE ソフトウェア V1.21 を使用。

本書に記載する内容は以下になります。

- Modem 構成における Host MCU プログラム、モジュールファームウェアやプロファイルの概要 (第 1 章)
- 開発環境 (第 2 章)
- Host MCU プログラムへ追加された API やモジュールとのシリアル通信仕様 (第 3 章)
- モジュールファームウェアのソフトウェア仕様やハードウェア設定 (第 4 章)

動作確認デバイス

RL78/G1D モジュール(RY7011)

※注意事項

- コード・フラッシュ・メモリのブロック 254 には、出荷時検査フラグが書き込まれていますので、書き換えないでください。
- コード・フラッシュ・メモリのブロック 255 には、Bluetooth Device Address が書き込まれていますので、書き換えないでください。
- 統合開発環境で未使用のコード・フラッシュ・メモリを消去しない設定にしてください。
  (「5.1 コード・フラッシュ・メモリ書き換え設定」を参照してください)

関連資料

資料名	資料番号	
	和文	英文
RL78/G1D モジュール		
ファームウェア ユーザーズマニュアル	R01UW0160J	R01UW0160E
ユーザーズマニュアル ハードウェア編	R02UH0004J	R02UH0004E
Bluetooth Low Energy プロトコルスタック		
ユーザーズマニュアル	R01UW0095J	R01UW0095E
クイックスタートガイド	R01AN2767J	R01AN2767E
API リファレンスマニュアル　基本編	R01UW0088J	R01UW0088E
API リファレンスマニュアル　FMP 編 (廃止)	R01UW0089J	R01UW0089E
API リファレンスマニュアル　PXP 編 (廃止)	R01UW0090J	R01UW0090E
API リファレンスマニュアル　HTP 編 (廃止)	R01UW0091J	R01UW0091E
API リファレンスマニュアル　BLP 編 (廃止)	R01UW0092J	R01UW0092E
API リファレンスマニュアル　HRP 編 (廃止)	R01UW0097J	R01UW0097E
API リファレンスマニュアル　GLP 編 (廃止)	R01UW0103J	R01UW0103E
API リファレンスマニュアル　TIP 編 (廃止)	R01UW0106J	R01UW0106E
API リファレンスマニュアル　RSCP 編 (廃止)	R01UW0107J	R01UW0107E
API リファレンスマニュアル　ANP 編 (廃止)	R01UW0108J	R01UW0108E
API リファレンスマニュアル　PASP 編 (廃止)	R01UW0109J	R01UW0109E
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
GUI ツール	R01AN2469J	R01AN2469E
RL78/G14 ホストサンプル	R01AN2807J	R01AN2807E
RX113 ホストサンプル	R01AN3155J	R01AN3155E

目次

1. 概要 5
1.1 システム構成 5
1.1.1 Host MCU プログラム 5
1.1.2 モジュールファームウェア 7
1.1.3 独自プロファイル 7
1.2 プロファイル 8
1.2.1 プロファイル一覧 8
1.2.2 サービス 9
1.2.3 GATT データベース 10
1.3 参考文書 14

2. 開発環境 15
2.1 ハードウェア環境 15
2.2 ソフトウェア環境 15
2.3 ビルド準備 16
2.3.1 BLE プロトコルスタック CC-RL 向けライブラリ 16
2.3.2 EEPROM エミュレーションライブラリのインストール 17
2.3.3 フラッシュセルフプログラミングライブラリのインストール 17
2.3.4 フォルダ構成 17

3. Host MCU プログラム 18
3.1 処理フロー 18
3.2 モジュールとの通信 19
3.2.1 リンク確立 20
3.2.2 送信動作 21
3.2.3 受信動作 22
3.3 rBLE API コールと rBLE API イベント通知 23
3.4 サービスを非公開にする 24
3.5 rBLE コマンド API 27
3.5.1 RBLE_VS_Set_Params 28
3.5.2 RBLE_VS_Flash_Access 29
3.6 周辺機能 31
3.7 ビルド方法 31
3.7.1 rBLE_sample のビルド方法 31
3.7.2 RL78/G1D サンプルのビルド方法 32

4. モジュールファームウェア 33
4.1 ファームウェア仕様 34
4.2 周辺機能 35
4.3 未使用端子処理 36
4.4 メモリマップ 37
4.5 ソースファイル変更箇所 39
4.5.1 プロファイルの選択 39
4.5.2 UART 2 線分岐接続方式 39
4.5.3 汎用双方向通信サービス 40

4.5.4 rBLE コマンド処理 42
4.5.5 ファームウェア関数 45
4.5.6 RF スロー・クロック 47
4.6 ビルド方法 48
4.6.1 ファームウェアのビルド 48
4.6.2 ファームウェアアップデートファイル 49

5. Appendix 50
5.1 コード・フラッシュ・メモリ書き換え設定 50
5.2 フォルダ構成 51
5.2.1 モジュールファームウェア V1.01 格納フォルダ 51
5.2.2 モジュールファームウェア V1.11 格納フォルダ 57
5.3 BLE ソフトウェア V1.21 対応 60
5.3.1 モジュールファームウェア 60
5.3.2 HostMCU サンプルプログラム 60
5.3.3 ビルド環境 61
5.3.4 ビルド方法 61
5.4 参考文献 62
5.5 用語説明 63
5.6 略語および略称の説明 64

1. 概要

1.1 システム構成

モジュールのシステムは、RL78/G1D モジュール(RY7011)とモジュールを制御する Host MCU で構成される Modem 構成です。ここでは、Modem 構成におけるソフトウェアの構成を説明します。

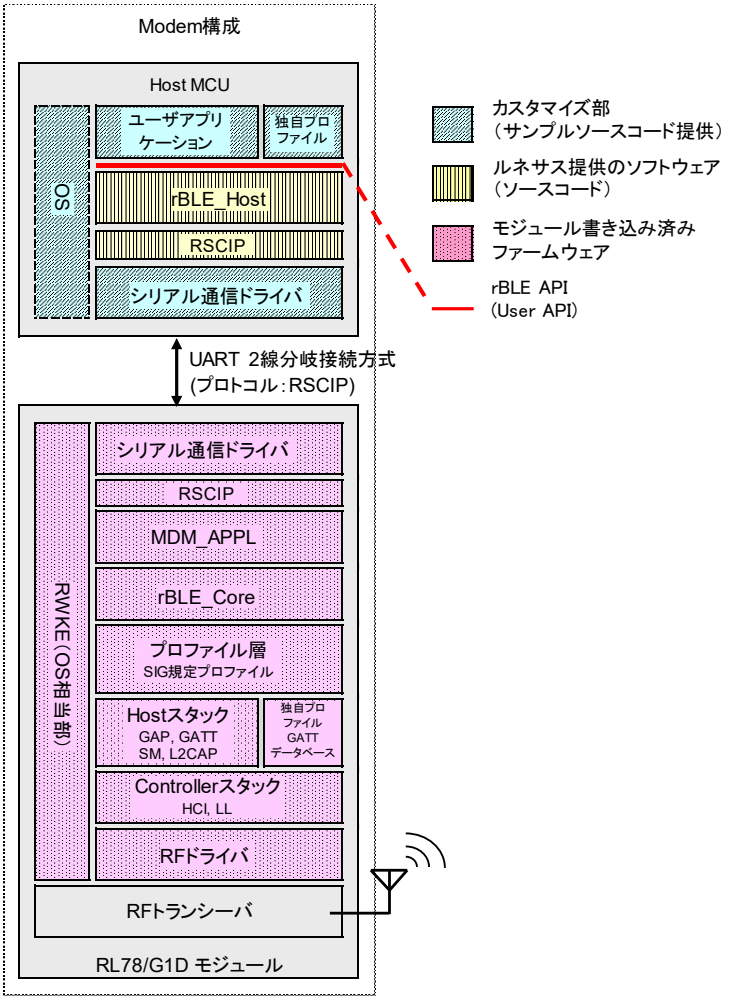

図 1-1　ソフトウェア構成図

1.1.1 Host MCU プログラム

Host MCU のプログラムは、ユーザアプリケーションと BLE 通信やモジュールとのシリアル通信を制御するためのプログラムで構成されます。ユーザアプリケーションはユーザ API である rBLE API を使用しモジュールを制御します。rBLE API は、GAP(Generic Access Profile)、GATT(Generic Attribute Profile)や各種プロファイルの API を提供し、モジュールからのイベント通知をコールバック関数で受け取ることができます。

ユーザアプリケーションとシリアル通信部の間にあり、モジュールに BLE 通信をさせるための rBLE コマンド処理を行うプログラムが rBLE_Host です。rBLE_Host は、ユーザアプリケーションに rBLE API や通知イベントを提供し、rBLE API に対応した rBLE コマンドパケットへの変換や、モジュールからの rBLE イベントパケットをユーザアプリケーションへのイベント通知に変換する処理を行います。

モジュールとの通信を制御するプログラムが RSCIP(Renesas Serial Communication Interface Protocol)です。RSCIP は、Host MCU とモジュールのリンク確立、rBLE コマンドパケットから RSCIP パケットへの変換やモジュールからの RSCIP パケットを rBLE イベントパケットに変換する処理を行います。また、シリアル通信で発生したエラーを再送によりリカバリする機能も持ちます。

モジュールと Host MCU の通信は、UART 2 線分岐接続方式でシリアル通信ドライバが制御します。UART 2 線分岐接続方式は、2 線の UART に加えて、Host MCU からの TxD ラインを分岐させてモジュールの WAKEUP と接続し、モジュールを省電力モードから起床するために利用した通信方式です。

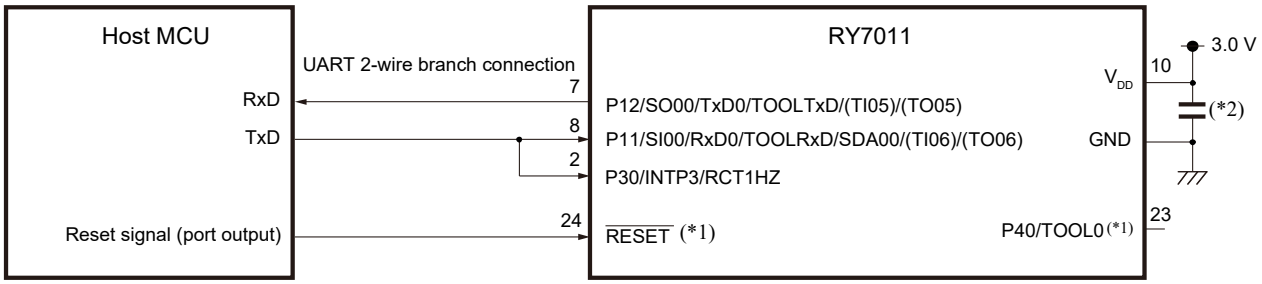

図 1-2　UART 2 線分岐接続方式 接続図

【注】*1　/RESET, P40/TOOL0 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)。

*2　供給電源や配線パターンの特性に応じて、VDD-GND 間に数 μF のバイパス・コンデンサを挿入してください。

表 1-1　Host MCU ソフトウェア構成

機能	用途
ユーザアプリケーション (*1)	rBLE API を使用し BLE 通信を行うユーザアプリケーションです。
独自プロファイル (*1)	独自プロファイルのユーザアプリケーション部です。 (Firmware Update, 汎用双方向通信)
OS (*1)	ユーザアプリケーションに OS を組み込むことが可能です。 (Host MCU サンプルプログラムでは使用していません)
rBLE_Host(*2)	ユーザアプリケーション、独自プロファイルからの rBLE API コールを受け、MDM_APPL への rBLE コマンド構築を行います。 MDM_APPL からの rBLE イベントパケット解析を行い、ユーザアプリケーション、独自プロファイルへ rBLE コールバック関数へのイベント通知を行います。 (Host MCU の違いによらない共通部分です)
RSCIP(*2)	Host MCU とモジュール間で使用する通信プロトコルです。Host MCU とモジュールのリンク確立(RSCIP パケットで通信できる状態にする)や、シリアル通信で発生したエラーについて再送によるリカバリ機能を提供します。 (Host MCU の違いによらない共通部分です)
シリアル通信ドライバ (*1)	モジュールとの通信を行う UART ドライバです。

【注】*1　ユーザカスタマイズ部です。Host MCU ソフトウェアではサンプルコードが提供されます。

*2　「5.2.1(2) Host MCU プログラム格納フォルダ」中で、rBLE_Host、RSCIP の記載があるファイルが当社から提供されるソースコードです。

1.1.2　モジュールファームウェア

Host MCU と同様にシリアル通信を制御するシリアル通信ドライバと RSCIP をモジュールも持ちます。モジュールのシリアル通信ドライバは、シリアル通信に加えて Host MCU からの通信により省電力モードから起床する WAKEUP 処理を行います。

MDM_APPL は、rBLE_Host からの rBLE コマンドパケットの解析、および rBLE_Host への rBLE イベントパケットの構築を行い、rBLE_Core を介して BLE スタックの機能を利用できるようにします。

rBLE_Core は、上位ブロックに対して、BLE スタック本体(プロファイル層～RF ドライバ)の機能を利用するためのインタフェースを提供します。

プロファイル層、Host スタック、Controller スタックが BLE 通信を制御する BLE スタックの本体です。Bluetooth SIG で規定されたプロファイル、GAP、GATT、SM(Security Manager)、L2CAP(Logical Link Control and Adaptation Protocol)や LL(Link Layer)が含まれます。

RF ドライバは、RL78/G1D の RF トランシーバ部を制御します。

RWKE は、他のブロックから共通して利用される基本的な機能を提供します。システム全体のスケジューリングとメモリ資源を管理します。

(モジュールファームウェアは、以降、ファームウェアと呼びます)

表 1-2　ファームウェア構成

機能	用途
シリアル通信ドライバ	UART 2 線分岐接続方式のドライバです。Host MCU と通信を行う UART ドライバと、モジュールを省電力モードから起床させる信号を受け付ける WAKEUP ドライバで構成されます。
RSCIP	Host MCU とモジュール間で使用する通信プロトコルです。Host MCU とモジュールのリンク確立(RSCIP パケットで通信できる状態にする)や、シリアル通信で発生したエラーについて再送によるリカバリ機能を提供します。
MDM_APPL	rBLE_Host からの rBLE コマンドパケットの解析、および rBLE_Host への rBLE イベントパケットの構築を行い、rBLE_Core を介して BLE スタックのサービスを利用可能にします。
rBLE_Core	上位ブロックに対して、BLE スタック本体(プロファイル層～RF ドライバ)のサービスを利用するためのインタフェースを提供します。
独自プロファイル GATT データベース	独自プロファイルの GATT データベース部 (Firmware Update, 汎用双方向通信)
プロファイル層	BLE スタック本体 プロファイル：FMP, PXP, HTP, BLP, HRP, GLP, TIP, RSCP, ANP, PASP Host スタック：SM, L2CAP, GAP, GATT Controller スタック：LL, HCI
Host スタック	
Controller スタック	
RWKE (Renesas Wireless Kernel Extension)	他のブロックから共通して利用される基本的な機能を提供し、システム全体のスケジューリングとメモリ資源を管理します。
RF ドライバ	RF トランシーバを制御します。

1.1.3　独自プロファイル

Modem 構成における独自プロファイルは、モジュール側と Host MCU 側に分割された構成になります。モジュール側には独自プロファイルの GATT データベース部、Host MCU 側にはユーザアプリケーションとして独自プロファイルが実装されます。

1.2　プロファイル

ここではファームウェアが対応しているプロファイルの仕様を示します。

Bluetooth SIG によるプロファイル・バージョンの非推奨、廃止計画により、Bluetooth Low Energy プロトコルスタックがサポートする Bluetooth SIG 規定プロファイルを使用した製品登録ができなくなったため各プロファイルを廃止しました。

製品登録については、「Bluetooth LE マイコン／モジュール Bluetooth 認証取得アプリケーションノート」(R01AN3177)を参照してください。

1.2.1　プロファイル一覧

ファームウェアが対応しているプロファイルの一覧を以下に示します。

表 1-3　Bluetooth SIG 規定プロファイル

Profile	Abbreviation	Role
Proximity Profile (廃止)	PXP	Monitor
		Reporter
Find Me Profile (廃止)	FMP	Locator
		Target
Heat Rate Profile (廃止)	HRP	Collector
		Sensor
Time Profile (廃止)	TIP	Client
		Server
Alert Notification Profile (廃止)	ANP	Client
		Server
Running Speed and Cadence Profile (廃止)	RSCP	Collector
		Sensor
Health Thermometer Profile (廃止)	HTP	Collector
		Thermometer
Blood Pressure Profile (廃止)	BLP	Collector
		Sensor
Glucose Profile (廃止)	GLP	Collector
		Sensor
Phone Alert Status Profile (廃止)	PASP	Client
		Server

表 1-4　独自プロファイル

Profile	Role
汎用双方向通信	Client
	Server
Firmware Update	Sender
	Receiver

1.2.2　サービス

モジュールが対応しているサービスと UUID の一覧を以下に示します。

表 1-5　Bluetooth SIG 規定サービス

Service	UUID (HEX)
Generic Access Service	1800
Immediate Alert Service	1802
Link Loss Service	1803
Tx Power Service	1804
Current Time Service	1805
Reference Time Update Service	1806
Next DST Change Service	1807
Glucose Service	1808
Health Thermometer Service	1809
Device Information Service	180A
Heart Rate Service	180D
Phone Alert Status Service	180E
Blood Pressure Service	1810
Alert Notification Service	1811
Running Speed and Cadence Service	1814

表 1-6　独自サービス

Service	UUID (HEX)
汎用双方向通信	D68C0001-A21B-11E5-8CB8-0002A5D5C51B
Firmware Update	01010000-0000-0000-0000-000000000080

1.2.3　GATT データベース

モジュールが持つ GATT データベースを以下に示します。

表 1-7　GATT データベース一覧

Attribute Handle	Attribute Type	Attribute Value
0x0001	Primary Service Declaration	0x1800(Generic Access Service)
0x0002	Characteristic Declaration	Properties = 0x0A(RD, WR)
0x0003	0x2A00 (Device Name)	
0x0004	Characteristic Declaration	Properties = 0x0A(RD, WR)
0x0005	0x2A01 (Appearance)	
0x0006	Characteristic Declaration	Properties = 0x02(RD)
0x0007	0x2A04 (Peripheral Preferred Connection Parameters)	
0x000C	Firmware Update 「表 1-8　Firmware Update データベース」を参照。	
0x000D		
0x000E		
0x000F		
0x0010		
0x0011	Primary Service Declaration Firmware Update	0x1803(Link Loss Service)
0x0012	Characteristic Declaration	Properties = 0x0A(RD, WR)
0x0013	0x2A06 (Alert Level)	
0x0014	Primary Service Declaration	0x1804(Tx Power Service)
0x0015	Characteristic Declaration	Properties = 0x02(RD)
0x0016	0x2A07 (Tx Power Level)	
0x0017	Primary Service Declaration	0x1802(Immediate Alert Service)
0x0018	Characteristic Declaration	Properties = 0x04(WR_NO_RESP)
0x0019	0x2A06 (Alert Level)	
0x001A	Primary Service Declaration	0x1809(Health Thermometer Service)
0x001B	Characteristic Declaration	Properties = 0x20(IND)
0x001C	0x2A1C (Temperature Measurement)	
0x001D	0x2902 (Client Characteristic Configuration)	
0x001E	Characteristic Declaration	Properties = 0x02(RD)
0x001F	0x2A1D (Temperature Type)	
0x0020	Characteristic Declaration	Properties = 0x10(NTF)
0x0021	0x2A1E (Intermediate Temperature)	
0x0022	0x2902 (Client Characteristic Configuration)	
0x0023	Characteristic Declaration	Properties = 0x2A(RD, WR, IND)
0x0024	0x2A21 (Measurement Interval)	
0x0025	0x2902 (Client Characteristic Configuration)	
0x0026	0x2906 (Valid Range)	
0x0027	Primary Service Declaration	0x1810(Blood Pressure Service)
0x0028	Characteristic Declaration	Properties = 0x20(IND)
0x0029	0x2A35 (Blood Pressure Measurement)	
0x002A	0x2902 (Client Characteristic Configuration)	
0x002B	Characteristic Declaration	Properties = 0x10(NTF)
0x002C	0x2A36 (Intermediate Cuff Pressure)	
0x002D	0x2902 (Client Characteristic Configuration)	
0x002E	Characteristic Declaration	Properties = 0x02(RD)
0x002F	0x2A49 (Blood Pressure Feature)	
0x0030	Primary Service Declaration	0x180A(Device Information Service)
0x0031	Characteristic Declaration	Properties = 0x02(RD)
0x0032	0x2A23 (System ID)	
0x0033	Characteristic Declaration	Properties = 0x02(RD)
0x0034	0x2A24 (Model Number String)	
0x0035	Characteristic Declaration	Properties = 0x02(RD)

0x0036	0x2A25 (Serial Number String)	
0x0037	Characteristic Declaration	Properties = 0x02(RD)
0x0038	0x2A26 (Firmware Revision String)	
0x0039	Characteristic Declaration	Properties = 0x02(RD)
0x003A	0x2A27 (Hardware Revision String)	
0x003B	Characteristic Declaration	Properties = 0x02(RD)
0x003C	0x2A28 (Software Revision String)	
0x003D	Characteristic Declaration	Properties = 0x02(RD)
0x003E	0x2A29 (Manufacturer Name String)	
0x003F	Characteristic Declaration	Properties = 0x02(RD)
0x0040	0x2A2A (IEEE 11073-20601 Regulatory Certification Data List)	
0x0041	Primary Service Declaration	0x180D(Heart Rate Service)
0x0042	Characteristic Declaration	Properties = 0x10(NTF)
0x0043	0x2A37 (Heart Rate Measurement)	
0x0044	0x2902 (Client Characteristic Configuration)	
0x0045	Characteristic Declaration	Properties = 0x02(RD)
0x0046	0x2A38 (Body Sensor Location)	
0x0047	Characteristic Declaration	Properties = 0x08(WR)
0x0048	0x2A39 (Heart Rate Control Point)	
0x0049	Primary Service Declaration	0x1808(Glucose Service)
0x004A	Characteristic Declaration	Properties = 0x10(NTF)
0x004B	0x2A18 (Glucose Measurement)	
0x004C	0x2902 (Client Characteristic Configuration)	
0x004D	Characteristic Declaration	Properties = 0x10(NTF)
0x004E	0x2A34 (Glucose Measurement Context)	
0x004F	0x2902 (Client Characteristic Configuration)	
0x0050	Characteristic Declaration	Properties = 0x02(RD)
0x0051	0x2A51 (Glucose Feature)	
0x0052	Characteristic Declaration	Properties = 0x28(WR, IND)
0x0053	0x2A52 (Record Access Control Point)	
0x0054	0x2902 (Client Characteristic Configuration)	
0x0055	Primary Service Declaration	0x1805(Current Time　Service)
0x0056	Characteristic Declaration	Properties = 0x12(RD, NTF)
0x0057	0x2A2B (Current Time)	
0x0058	0x2902 (Client Characteristic Configuration)	
0x0059	Characteristic Declaration	Properties = 0x02(RD)
0x005A	0x2A0F (Local Time Information)	
0x005B	Characteristic Declaration	Properties = 0x02(RD)
0x005C	0x2A14 (Reference Time Information)	
0x005D	Primary Service Declaration	0x1807(Next DST Change Service)
0x005E	Characteristic Declaration	Properties = 0x02(RD)
0x005F	0x2A11 (Time with DST)	
0x0060	Primary Service Declaration	0x1806(Reference Time Update Service)
0x0061	Characteristic Declaration	Properties = 0x04(WR_NO_RESP)
0x0062	0x2A16 (Time Update Control Point)	
0x0063	Characteristic Declaration	Properties = 0x02(RD)
0x0064	0x2A17 (Time Update State)	
0x0065	Primary Service Declaration	0x1811(Alert Notification Service)
0x0066	Characteristic Declaration	Properties = 0x02(RD)
0x0067	0x2A47 (Supported New Alert Category)	
0x0068	Characteristic Declaration	Properties = 0x10(NTF)
0x0069	0x2A46 (New Alert)	
0x006A	0x2902 (Client Characteristic Configuration)	
0x006B	Characteristic Declaration	Properties = 0x02(RD)
0x006C	0x2A48 (Supported Unread Alert Category)	
0x006D	Characteristic Declaration	Properties = 0x10(NTF)

0x006E	0x2A45 (Unread Alert Status)	
0x006F	0x2902 (Client Characteristic Configuration)	
0x0070	Characteristic Declaration	Properties = 0x08(WR)
0x0071	0x2A44 (Alert Notification Control Point)	
0x0072	Primary Service Declaration	0x180E(Phone Alert Status Service)
0x0073	Characteristic Declaration	Properties = 0x12(RD, NTF)
0x0074	0x2A3F (Alert Status)	
0x0075	0x2902 (Client Characteristic Configuration)	
0x0076	Characteristic Declaration	Properties = 0x12(RD, NTF)
0x0077	0x2A41 (Ringer Setting)	
0x0078	0x2902 (Client Characteristic Configuration)	
0x0079	Characteristic Declaration	Properties = 0x04(WR_NO_RESP)
0x007A	0x2A40 (Ringer Control Point)	
0x007B	Primary Service Declaration	0x1814(Running Speed and Cadence Service)
0x007C	Characteristic Declaration	Properties = 0x10(NTF)
0x007D	0x2A53 (RSC Measurement)	
0x007E	0x2902 (Client Characteristic Configuration)	
0x007F	Characteristic Declaration	Properties = 0x02(RD)
0x0080	0x2A54 (RSC Feature)	
0x0081	Characteristic Declaration	Properties = 0x02(RD)
0x0082	0x2A5D (Sensor Location)	
0x0083	Characteristic Declaration	Properties = 0x28(WR, IND)
0x0084	0x2A55 (SC Control Point)	
0x0085	0x2902 (Client Characteristic Configuration)	
0x0086	汎用双方向通信 「表 1-9　汎用双方向通信データベース」を参照。	
0x0087		
0x0088		
0x0089		
0x008A		
0x008B		

(1)　Bluetooth SIG 規定プロファイル データベース仕様

Bluetooth SIG 規定プロファイルのデータベース仕様については「Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル」(R01UW0095)の下記章や、Bluetooth SIG のプロファイル仕様書(「5.4 参考文献」)を参照してください。

- 7.2 Generic Access Profile
- 7.5 Find Me Profile
- 7.6 Proximity Profile
- 7.7 Health Thermometer Profile
- 7.8 Blood Pressure Profile
- 7.11 Heat Rate Profile
- 7.14 Glucose Profile
- 7.15 Time Profile
- 7.16 Running Speed and Cadence Profile
- 7.17 Alert Notification Profile
- 7.18 Phone Alert Status Profile

(2) 独自プロファイル データベース仕様

モジュールに組み込まれている独自プロファイルの GATT データベース仕様を以下に示します。

表 1-8 Firmware Update データベース

<table>
<tr><th>Attribute Handle</th><th>Attribute Type</th><th>Attribute Value</th></tr>
<tr><td>0x000C</td><td>Primary Service Declaration (0x2800)</td><td>UUID: 01010000-0000-0000-0000-000000000080</td></tr>
<tr><td>0x000D</td><td>Characteristic Declaration (0x2803)</td><td>Property: Write(0x08)
Type: Characteristic Declaration
UUID: 02010000-0000-0000-0000-000000000080</td></tr>
<tr><td>0x000E</td><td>Value</td><td>Control Data
<table>
<tr><th>Cmd</th><th>Operation</th></tr>
<tr><td>0</td><td>Data transmission start
Params: Current Block Num / Size</td></tr>
<tr><td>1</td><td>Data transmission completion
Params: none</td></tr>
<tr><td>2</td><td>Data write confirmation
Params: none</td></tr>
<tr><td>3</td><td>Data transmission completion(all data)
Params: none</td></tr>
</table></td></tr>
<tr><td>0x000F</td><td>Characteristic Declaration (0x2803)</td><td>Property: Write Without Response(0x04)
Type: Characteristic Declaration
UUID: 03010000-0000-0000-0000-000000000080</td></tr>
<tr><td>0x0010</td><td>Value</td><td>Update Data
<table>
<tr><td>Update Data</td><td>1-19 byte</td></tr>
<tr><td>Check Sum</td><td>1 byte</td></tr>
</table></td></tr>
</table>

表 1-9 汎用双方向通信データベース

Attribute Handle	Attribute Type	Attribute Value
0x0086	Primary Service Declaration (0x2800)	UUID: 0xD68C0001-A21B-11E5-8CB8-0002A5D5C51B
0x0087	Characteristic Declaration (0x2803)	Property: Indicate(0x20) Type: Characteristic Declaration UUID: 0xD68C0002-A21B-11E5-8CB8-0002A5D5C51B
0x0088	Indication Value	本 Characteristic にデータを設定後、Indication を送信することで、サーバからクライアントへの文字送信を行います。一度に送信可能な文字数は、最大 20 文字です。
0x0089	Client Characteristic Configuration Descriptor (0x2902)	上記 Indication の有効・無効を Client から制御します。 0x0000: Indications disabled 0x0002: Indications enabled
0x008A	Characteristic Declaration (0x2803)	Property: Write(0x08) Type: Characteristic Declaration UUID: 0xD68C0003-A21B-11E5-8CB8-0002A5D5C51B
0x008B	Write Value	本 Characteristic に Write Request で文字を書き込むことにより、クライアントからサーバへの文字送信を行います。一度に送信可能な文字数は、最大 20 文字です。

1.3 参考文書

関連する文書を以下に示します。併せて参照して下さい。

- rBLE API
  - API リファレンスマニュアル 基本編(R01UW0088)
  - プロファイル各種の API リファレンスマニュアル

- rBLE コマンド
  - rBLE コマンド仕様書(R01AN1376)

- Firmware Update
  - Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル(R01UW0095)
    「11. FW アップデート機能の実装」

- UART 2 線分岐接続方式
  - ホストサンプルアプリケーションノート(R01AN2807)
  - RX113 向けホストサンプルアプリケーション(R01AN3155)

2. 開発環境

モジュール制御ソフトウェアの開発環境を以下に示します。

2.1 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）

－ 使用ツール

- Renesas オンチップデバッギングエミュレータ E1

2.2 ソフトウェア環境

- Windows 7 以降
- Microsoft Visual Studio Express 2015 for Windows Desktop.
- Microsoft .NET Framework 4＋言語パック
- e² studio V5.4.0.015 / RL78 コンパイラ CC-RL V1.02.00 and V1.03.00
  CC-RL V1.02.00: モジュールファームウェア V1.01 のビルドで使用。
  (モジュールに書き込まれているファームウェア)
  CC-RL V1.03.00: モジュールファームウェア V1.11 のビルドで使用。
- Renesas Flash Programmer V3

2.3 ビルド準備

本アプリケーションノートに同梱されているソフトウェアを、空白やマルチバイト文字を含まないパスにコピーしてください。

ファームウェアをビルドするためには以下のライブラリが必要です。各ライブラリを入手後、「5.2.1(3) モジュールファームウェア格納フォルダ」の指定フォルダにコピーして下さい。

- Renesas Bluetooth Low Energy プロトコルスタック V1.11 CC-RL 向けライブラリファイル
- RL78 ファミリ EEPROM エミュレーションライブラリ Pack02 パッケージ Ver.2.00(CA78K0R/CC-RL コンパイラ用)
- RL78 ファミリフラッシュセルフプログラミングライブラリ Type01 パッケージ Ver.3.00 (CA78K0R/CC-RL コンパイラ用)

2.3.1 BLE プロトコルスタック CC-RL 向けライブラリ

CC-RL 向けライブリ格納フォルダと、コピーするライブラリファイル:

¥BLE_Software_Ver_1_11¥RL78_G1D¥Project_Source¥renesas¥lib

- BLE_CONTROLLER_LIB_CCRL.lib
- BLE_HOST_lib_CCRL.lib
- BLE_rBLE_lib_CCRL.lib
- BLE_PROFILES_COMMON_LIB_CCRL.lib
- BLE_PROFILE_ANP_LIB_CCRL.lib
- BLE_PROFILE_BLP_LIB_CCRL.lib
- BLE_PROFILE_CPP_LIB_CCRL.lib
- BLE_PROFILE_CSP_LIB_CCRL.lib
- BLE_PROFILE_FMP_LIB_CCRL.lib
- BLE_PROFILE_GLP_LIB_CCRL.lib
- BLE_PROFILE_HGP_LIB_CCRL.lib
- BLE_PROFILE_HRP_LIB_CCRL.lib
- BLE_PROFILE_HTP_LIB_CCRL.lib
- BLE_PROFILE_LNP_LIB_CCRL.lib
- BLE_PROFILE_PAP_LIB_CCRL.lib
- BLE_PROFILE_PXP_LIB_CCRL.lib
- BLE_PROFILE_RSP_LIB_CCRL.lib
- BLE_PROFILE_SCP_LIB_CCRL.lib
- BLE_PROFILE_TIP_LIB_CCRL.lib

コピー先:

¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥lib

2.3.2　EEPROM エミュレーションライブラリのインストール

本ファームウェアでは CC-RL 用のライブラリを使用してください。

コピーするファイル:

- eel.h
- eel.lib
- eel_types.h
- fdl.h
- fdl.lib
- fdl_types.h

コピー先:

¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥dataflash¥cc_rl

2.3.3　フラッシュセルフプログラミングライブラリのインストール

本ファームウェアでは CC-RL 用のライブラリを使用してください。

コピーするファイル:

fsl.h

fsl.lib

fsl_types.h

コピー先:

¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥codeflash¥cc_rl

2.3.4　フォルダ構成

「5.2 フォルダ構成」を参照してください。Host MCU のルネサス提供のソフトウェア rBLE_Host、RSCIP についても「5.2.1(2) Host MCU プログラム格納フォルダ」中に記載しています。

3. Host MCU プログラム

3.1 処理フロー

Host MCU のプログラムは、アプリケーション、独自プロファイル、rBLE_Host、RSCIP、シリアル通信ドライバ、OS のブロックで構成されます。各ブロック間の処理フロー図を以下に示します。

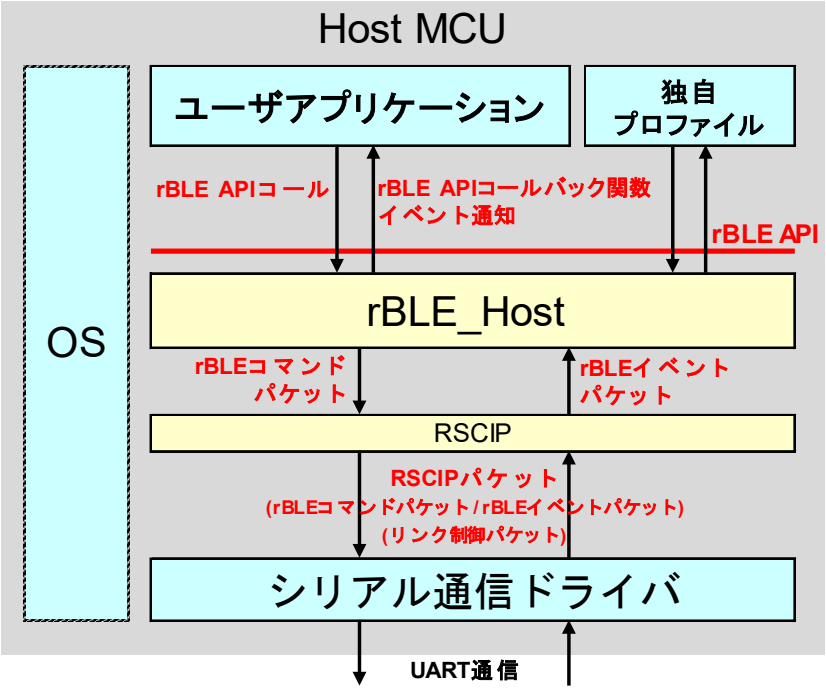

図 3-1　Host MCU 処理フロー

Host MCU のプログラムは、ユーザアプリケーションが呼び出す rBLE API でモジュールを制御します。モジュールは実行結果をイベントとして Host MCU に返し、ユーザアプリケーションは rBLE API コールバック関数でイベント通知として受け取ります。(ユーザアプリケーションとして実装される独自プロファイルも同じ動作を行います)

これらの rBLE API コールと rBLE API コールバック関数イベント通知の処理を行うブロックが rBLE_Host です。アプリケーションによって呼び出された rBLE API を rBLE コマンドパケットに変換、モジュールからの rBLE イベントパケットで rBLE API コールバック関数のイベント通知を行います。

RSCIP は、モジュールとの通信を制御して、Host MCU とモジュールの接続確立(リンク)、rBLE コマンドパケットから RSCIP パケットへの変換やモジュールからの RSCIP パケットを rBLE イベントパケットに変換する処理を行います。また、シリアル通信で発生したエラーを再送によりリカバリする機能も持ちます。

3.2　モジュールとの通信

Host MCU とモジュール間は UART 2 線分岐接続方式で通信が行われます。ここでは、モジュールとの通信手順を説明します。

Host MCU とモジュールの通信は、リンク(接続)を確立した後、リンク制御パケット、rBLE コマンドパケット、rBLE イベントパケットを含んだ RSCIP パケットを用いて行われます。

リンク制御パケット　: Host MCU とモジュール間のリンク確立に使用します。

rBLE コマンドパケット　: Host MCU からモジュールへのオペレーション指示に使用します。

rBLE イベントパケット　: モジュールから Host MCU への情報通知に使用します。

RSCIP パケットについての詳細は「rBLE コマンド仕様書」(R01AN1376)を参照してください。

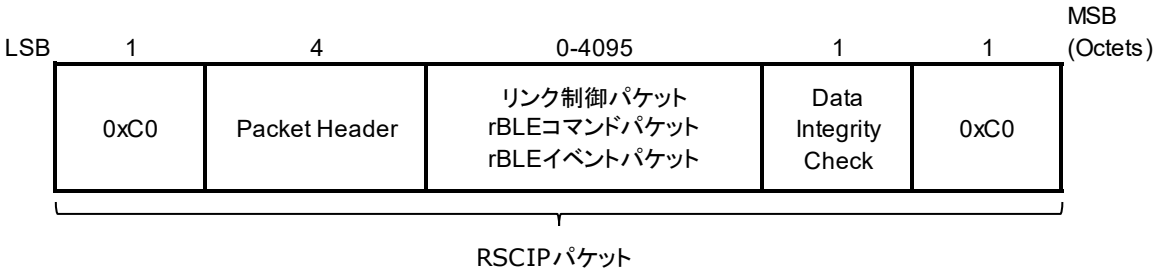

図 3-2　RSCIP パケット

3.2.1　リンク確立

Host MCU とモジュールが、rBLE コマンドパケットや rBLE イベントパケットを用いて通信を行うためには、まずリンクを確立します。リンクの確立で使用するパケットがリンク制御パケットです。

Host MCU とモジュールのリンクは、お互いにリンク制御パケットである「SYNC - SYNC RESPONSE」、「CONFIG - CONFIG RESPONSE」の交換を行い確立します。また、UART 2 線分岐接続方式では、Host MCU からモジュールに RSCIP パケットを送信する時、ハンドシェイクを行います。ハンドシェイクは、モジュールが受信の準備を完了していることを確認するための動作で、Host MCU から REQ バイト(0xC0)を送信し、モジュールから ACK バイト(0x88)または RSCIP パケットを受信します。

ハンドシェイクを含めた UART ドライバの状態遷移については、次の「3.2.2 送信動作」を参照してください。以下にリンク確立までの RSCIP パケット交換シーケンスを示します。

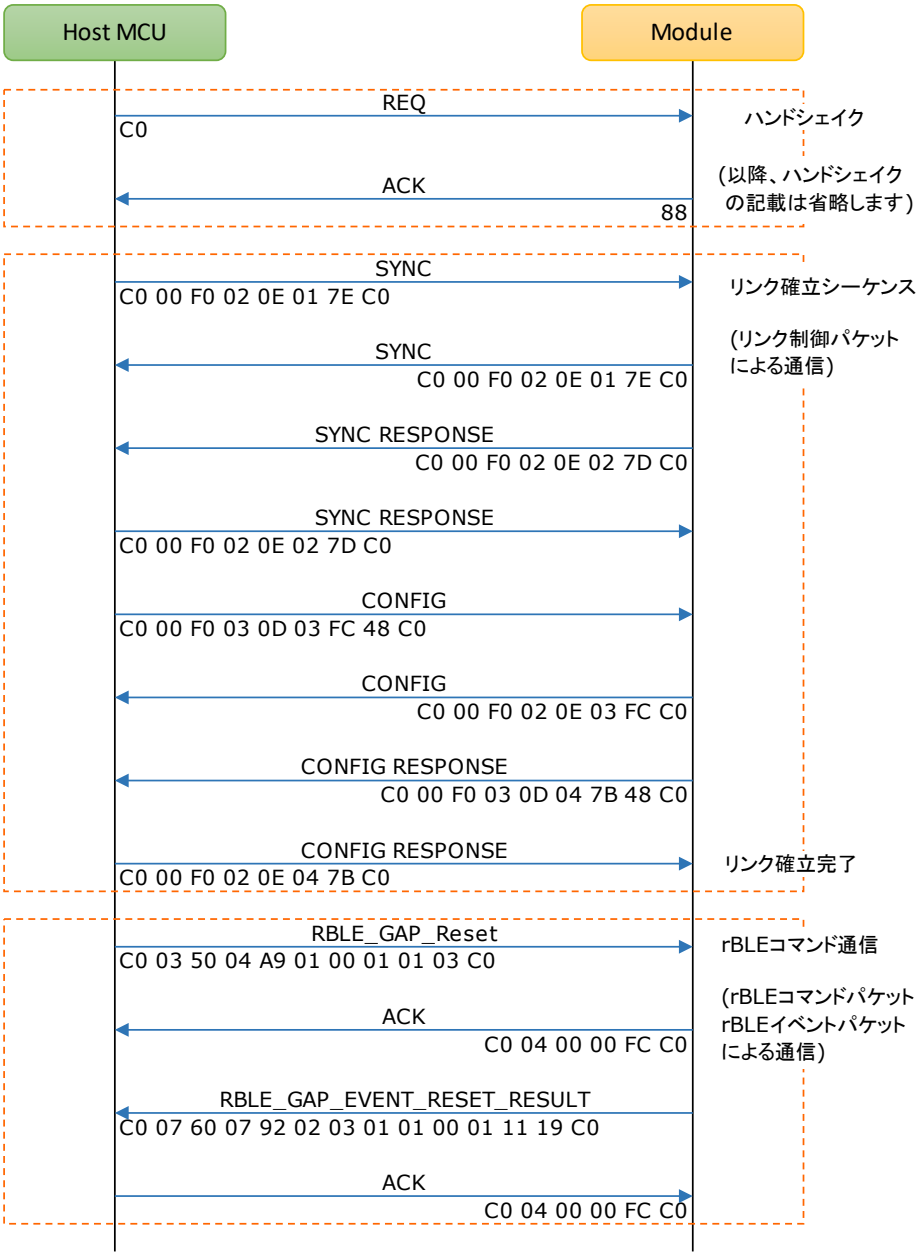

図 3-3　リンク確立 RSCIP パケット交換シーケンス

3.2.2　送信動作

Host MCU からモジュール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、モジュールから送信される 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 3-1　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 バイトの代わりにモジュールから RSCIP パケットを受信
T_REQUESTED	REQ バイト送信完了(モジュールからの ACK バイト待ち)
T_ACTIVE	RSCIP パケット送信中

Host MCU からモジュールへの送信は必ず 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 がモジュールからの RSCIP パケットを受信していない(図 3-4)

(b) Host MCU がモジュールからの RSCIP パケットを受信中(図 3-5)

(c) REQ バイト送信タイムアウト(図 3-6)

(a)　Host MCU がモジュールからの RSCIP パケットを受信していない

この状態は、モジュールから 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。モジュールは 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、モジュールに RSCIP パケットを送信します。

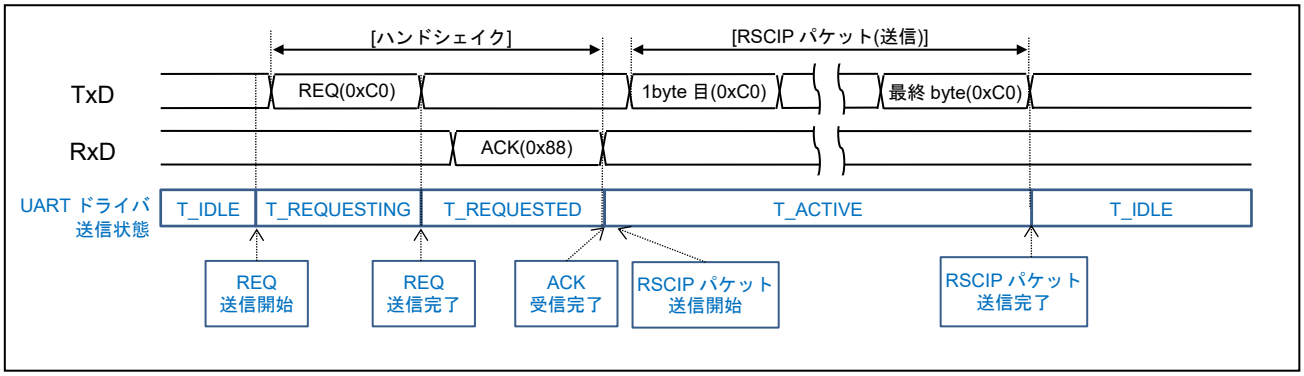

図 3-4　Host MCU がモジュールからの rBLE パケットを受信していない場合の動作

(b)　Host MCU がモジュールからの RSCIP パケットを受信中

この状態はモジュールが 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。モジュールは 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。ホストはモジュールからの RSCIP パケットを ACK バイトの代わりとし、モジュールに RSCIP パケットを送信します。

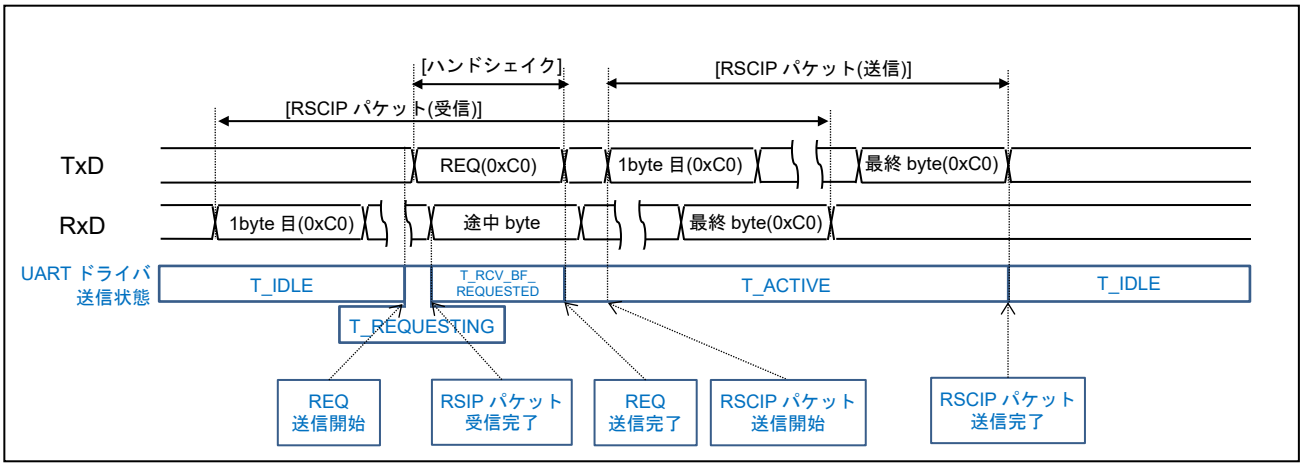

図 3-5　Host MCU がモジュールからのデータを受信している場合の動作

(c)　REQ バイト送信タイムアウト

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。

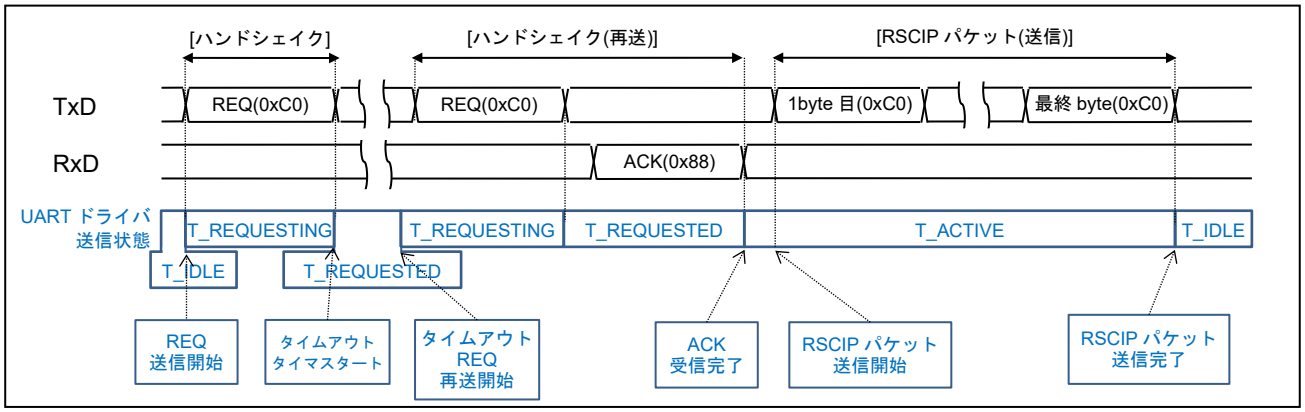

図 3-6　REQ バイトがタイムアウトした場合の動作

3.2.3　受信動作

受信時に UART ドライバの状態遷移はありません。モジュール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モジュールからの RSCIP パケットを待ち受けます。

3.3 rBLE API コールと rBLE API イベント通知

BLE 通信は、rBLE API の呼び出しと、rBLE API の実行結果や通信によるモジュールからの通知(これをイベント通知と呼びます)によりコントロールされます。イベント通知は、コールバック関数で受け取ります。コールバック関数は、rBLE の初期化や、GAP、SM、GATT、VS(Vendor Specific)、GATT ベースプロファイルのロール毎に用意する必要があり、コールバック登録関数で rBLE_Host に登録することによってイベント通知を受け取ることができるようになります。ここでは、rBLE API コールと rBLE API イベント通知の例を示します。

GAP コールバック関数の登録は、rBLE API の RBLE_GAP_Reset 関数で行います。引数に GAP のイベントを通知するコールバック関数と、SM のイベントを通知するコールバック関数を指定することで rBLE_Host に登録され、イベント通知を受け取ることが可能になります。RBLE_GAP_Reset 関数は、実行結果として BLE_GAP_EVENT_RESET_RESULT イベントが GAP コールバック関数に通知されます。

Broadcast を開始する場合は、RBLE_GAP_Broadcast_Enable 関数をコールします。RBLE_GAP_Broadcast_Enable 関数の実行結果として RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP イベントが GAP コールバック関数に通知されます。

rBLE API とイベント通知については「API リファレンスマニュアル 基本編」(R01UW0088)や、各 API リファレンスマニュアルを参照してください。

Source File : ¥Renesas_BLE_Module_V101¥BLE_Sample¥src¥rBLE¥src¥sample_app¥rble_sample_app.c

```
      /* GAP コールバック関数 */
 1762:static void RBLE_APP_GAP_CallBack( RBLE_GAP_EVENT *event )
 1763:{
 1764:  :
 1769:    switch( event->type ) {
 1770:        case RBLE_GAP_EVENT_RESET_RESULT:
        :
 1780:        case RBLE_GAP_EVENT_SET_NAME_COMP:
        :
 1783:        case RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP:
        :
 1786:        case RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP:
        :
 1789:        case RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP:
        :
```

RBLE_GAP_Reset イベント通知

RBLE_GAP_Broadcast_Enable イベント通知

```
      /* GAP リセット処理(コールバック関数登録) */
 2136:static BOOL RBLE_GAP_Reset_Test( void )
 2137:{
 2138:
 2139:    RBLE_STATUS Ret_Status;
 2140:
 2141:    /* API Call */
 2142:    Ret_Status = RBLE_GAP_Reset( &RBLE_APP_GAP_CallBack, &RBLE_APP_SM_CallBack );
        :
 2148:}
```

RBLE_GAP_Reset 関数コール

```
      /* Broadcast 開始処理 */
 2264:static BOOL RBLE_GAP_Broadcast_Enable_Test( void )
 2265:{
        :
 2351:    /* API Call */
 2352:    Ret_Status = RBLE_GAP_Broadcast_Enable( Parameter_p->disc_mode,
                                                  Parameter_p->conn_mode,
                                                  &Parameter_p->adv_info );
        :
 2356:}
```

RBLE_GAP_Broadcast_Enable 関数コール

3.4 サービスを非公開にする

ファームウェアの対応するサービスを使用しない場合は、対向デバイスに不要な処理を発生させないように、使用しないサービスを非公開に設定する必要があります。モジュール起動時に Host MCU プログラムからサービスの非公開設定を行ってください。

サービスを非公開にするには、rBLE API の RBLE_GATT_Set_Permission 関数を使用し、ローカル GATT データベースのパーミッションに RBLE_GATT_PERM_HIDE(0x2000)を設定します。

設定方法は、引数で指定した開始ハンドル(start_hdl)～終了ハンドル(end_hdl)の範囲に、パーミッション値(perm)を設定します。元のパーミッション値は、新しい値に設定されるので注意してください。設定が完了すると、rBLE API イベントの RBLE_GATT_EVENT_SET_PERM_CMP が通知されます。

※Attribute Handle の 0x0001～0x000B のパーミッションは変更しないで下さい。(「1.2.3 GATT データベース」を参照)

以下に、パーミッション設定で使用する API とイベントを示します。

- rBLE API

RBLE_STATUS RBLE_GATT_Set_Permission(RBLE_GATT_SET_PERM *set_perm)			
このファンクションは、指定ハンドル範囲のローカル GATT データベースのパーミッションを設定します。結果はパーミッション設定完了イベント RBLE_GATT_EVENT_SET_PERM_CMP で通知されます。			
Parameters:			
	**set_perm*	*start_hdl*	パーミッション設定開始ハンドル
		end_hdl	パーミッション設定終了ハンドル
		perm	パーミッション (API リファレンスマニュアル 基本編 7.1 Definitions - GATT アトリビュートパーミッション宣言を参照ください)
Return:			
	RBLE_OK (0x00)	正常終了	
	RBLE_STATUS_ERROR (0xF2)	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可	

- rBLE API イベント

RBLE_GATT_EVENT_SET_PERM_CMP			
このイベントは、ローカル GATT データベースのパーミッション設定結果を通知します。			
Parameters:			
	set_perm_cmp	*status*	パーミッション設定結果 (API リファレンスマニュアル 基本編 3.2 Generic Definitions - rBLE ステータス列挙型宣言を参照ください)

サービスを非公開にした後、再度公開する場合は、RBLE_GATT_Set_Permission 関数を使用しハンドルごとにパーミッション値を設定してください。表 3-2 にパーミッション初期値、表 3-3 にパーミッション値の定義を示します。

表 3-2　パーミッション初期値

Attribute Handle	Permission Value	Attribute Handle	Permission Value	Attribute Handle	Permission Value	Attribute Handle	Permission Value
0x000B	0x2011	0x002B	0x0001	0x004B	0x0100	0x006B	0x0001
0x000C	0x0001	0x002C	0x0100	0x004C	0x0011	0x006C	0x0001
0x000D	0x0001	0x002D	0x0011	0x004D	0x0001	0x006D	0x0001
0x000E	0x0010	0x002E	0x0001	0x004E	0x0100	0x006E	0x0100
0x000F	0x0001	0x002F	0x0001	0x004F	0x0011	0x006F	0x0011
0x0010	0x0010	0x0030	0x0001	0x0050	0x0001	0x0070	0x0001
0x0011	0x0001	0x0031	0x0001	0x0051	0x0001	0x0071	0x0010
0x0012	0x0001	0x0032	0x0001	0x0052	0x0001	0x0072	0x0001
0x0013	0x0011	0x0033	0x0001	0x0053	0x0120	0x0073	0x0001
0x0014	0x0001	0x0034	0x0001	0x0054	0x0011	0x0074	0x0101
0x0015	0x0001	0x0035	0x0001	0x0055	0x0001	0x0075	0x0011
0x0016	0x0001	0x0036	0x0001	0x0056	0x0001	0x0076	0x0001
0x0017	0x0001	0x0037	0x0001	0x0057	0x0101	0x0077	0x0101
0x0018	0x0001	0x0038	0x0001	0x0058	0x0011	0x0078	0x0011
0x0019	0x0010	0x0039	0x0001	0x0059	0x0001	0x0079	0x0001
0x001A	0x0001	0x003A	0x0001	0x005A	0x0001	0x007A	0x0010
0x001B	0x0001	0x003B	0x0001	0x005B	0x0001	0x007B	0x0001
0x001C	0x0100	0x003C	0x0001	0x005C	0x0001	0x007C	0x0001
0x001D	0x0011	0x003D	0x0001	0x005D	0x0001	0x007D	0x0100
0x001E	0x0001	0x003E	0x0001	0x005E	0x0001	0x007E	0x0011
0x001F	0x0001	0x003F	0x0001	0x005F	0x0001	0x007F	0x0001
0x0020	0x0001	0x0040	0x0001	0x0060	0x0001	0x0080	0x0001
0x0021	0x0100	0x0041	0x0001	0x0061	0x0001	0x0081	0x0001
0x0022	0x0011	0x0042	0x0001	0x0062	0x0010	0x0082	0x0001
0x0023	0x0001	0x0043	0x0100	0x0063	0x0001	0x0083	0x0001
0x0024	0x0121	0x0044	0x0011	0x0064	0x0001	0x0084	0x0110
0x0025	0x0011	0x0045	0x0001	0x0065	0x0001	0x0085	0x0011
0x0026	0x0001	0x0046	0x0001	0x0066	0x0001	0x0086	0x0001
0x0027	0x0001	0x0047	0x0001	0x0067	0x0001	0x0087	0x0001
0x0028	0x0001	0x0048	0x0010	0x0068	0x0001	0x0088	0x0100
0x0029	0x0100	0x0049	0x0001	0x0069	0x0100	0x0089	0x0011
0x002A	0x0011	0x004A	0x0001	0x006A	0x0011	0x008A	0x0001
						0x008B	0x0010

表 3-3　パーミッション定義値

Define Name	Value	Brief
RBLE_GATT_PERM_NONE	0x0000	No permission
RBLE_GATT_PERM_RD	0x0001	Read permission
RBLE_GATT_PERM_RD_UNAUTH	0x0002	Read permission (Unauthentication Required)
RBLE_GATT_PERM_RD_AUTH	0x0004	Read permission (Authentication Required)
RBLE_GATT_PERM_RD_AUTZ	0x0008	Read permission (Authorization Required)
RBLE_GATT_PERM_WR	0x0010	Write permission
RBLE_GATT_PERM_WR_UNAUTH	0x0020	Write permission (Unauthentication Required)
RBLE_GATT_PERM_WR_AUTH	0x0040	Write permission (Authentication Required)
RBLE_GATT_PERM_WR_AUTZ	0x0080	Write permission (Authorization Required)
RBLE_GATT_PERM_NI	0x0100	Notifications/Indications permission
RBLE_GATT_PERM_NI_UNAUTH	0x0200	Notifications/Indications permission (Unauthentication Required)
RBLE_GATT_PERM_NI_AUTH	0x0400	Notifications/Indications permission (Authentication Required)
RBLE_GATT_PERM_NI_AUTZ	0x0800	Notifications/Indications permission (Authorization Required)
RBLE_GATT_PERM_EKS	0x1000	Encryption key size Required
RBLE_GATT_PERM_HIDE	0x2000	Not expose
RBLE_GATT_PERM_ENC	0x4000	Encryption Required
RBLE_GATT_PERM_NOTIFY_COMP_EN	0x8000	Notification Complete Enable

3.5 rBLE コマンド API

ファームウェアの独自機能として、rBLE API の Vendor Specific 機能である RBLE_VS_Set_Params 関数と、RBLE_VS_Flash_Access 関数に独自 rBLE コマンドを追加しています。

RBLE_VS_Set_Params 関数は、ユーザが処理を割り当てることができる第 1 引数 param_id の 0x80 以降に以下の 3 つのコマンドを追加しています。

- ボーレート設定
- Firmware Update
- Software Reset

RBLE_VS_Flash_Access 関数は、パラメータ構造体メンバの id に定義を追加することで以下のコマンドを追加しています。

- ファームウェアバージョン読み出し

以下に rBLE API 仕様を示します。

3.5.1 RBLE_VS_Set_Params

RBLE_STATUS RBLE_VS_Set_Params (uint8_t param_id, uint8_t param_len, uint8_t *param_data)

このファンクションは、BLE MCU 内のパラメータ値を設定します。
結果はパラメータ設定完了イベント RBLE_VS_EVENT_SET_PARAMS_COMP で通知されます。

- ボーレート設定は、ボーレート番号をデータ・フラッシュに保存します。モジュール起動時にボーレート番号を読み出しシリアル通信ドライバを初期化します。ボーレート設定を実行した場合、リセットを行ってください。ボーレートはリセット後、有効になります。
- Firmware Update モードへの移行を実行した場合、1 秒後に Firmware Update モードへ移行します。
- Software Reset は、不正命令の実行による内部リセットを行います。Software Reset を実行した場合、1 秒後に内部リセットが発生します。

Parameters:

param_id

設定パラメータ ID

設定パラメータ ID	番号	説明
RBLE_VS_PARAM_UART_BAUD_ID	0x80	ボーレート設定
RBLE_VS_PARAM_FW_UPDATE	0xD9	Firmware Update モードへの移行
RBLE_VS_PARAM_SOFT_RESET	0xFF	Software Reset

param_len

パラメータ長

設定パラメータ ID	パラメータ長
RBLE_VS_PARAM_UART_BAUD_ID	1
RBLE_VS_PARAM_FW_UPDATE	使用しません
RBLE_VS_PARAM_SOFT_RESET	使用しません

**param_data*

パラメータ格納先へのポインタ(データは下位バイトより前詰め)

設定パラメータ ID	ボーレート番号
RBLE_VS_PARAM_UART_BAUD_ID	0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps 6: 250000 bps
RBLE_VS_PARAM_FW_UPDATE	使用しません
RBLE_VS_PARAM_SOFT_RESET	使用しません

Return:

RBLE_OK (0x00)	正常終了
RBLE_STATUS_ERROR (0xF2)	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_UNSUPPORTED (0x11)	未サポートの param_id
RBLE_PARAM_ERR (0xF3)	パラメータエラー

RBLE_VS_Set_Params を実行すると、コマンド実行結果が RBLE_VS_EVENT_SET_PARAMS_COMP イベントで通知されます。パラメータの"status"に実行結果が格納されます。

RBLE_VS_EVENT_SET_PARAMS_COMP			
このイベントは、パラメータ設定完了を通知します。			
Parameters:			
	status	*RBLE_OK (0x00)*	正常終了
		RBLE_STATUS_ERROR (0xF2)	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
		RBLE_UNSUPPORTED (0x11)	未サポートの param_id
		RBLE_PARAM_ERR (0xF3)	パラメータエラー

3.5.2 RBLE_VS_Flash_Access

RBLE_STATUS RBLE_VS_Flash_Access (RBLE_VS_FLASH_ACCESS_PARAM *param)		
このファンクションは、Data Flash へデータの書き込みまたは、データの読み出しを行います。 結果は Data Flash データアクセスコマンド完了イベント RBLE_VS_EVENT_FLASH_ACCESS_COMP で通知されます。 ※このファンクションを実行する前に、RBLE_VS_Flash_Management にて Data Flash へのアクセスを開始してください。また、データの書き込みまたは読み出しが完了するまで、パラメータで指定したバッファは保持しておく必要があります。		
Parameters:		
	cmd	Data Flash アクセスコマンド RBLE_VS_FLASH_CMD_WRITE：データ書き込み RBLE_VS_FLASH_CMD_READ：データ読み出し
	id	データ ID(0x01 – 0xFF) 設定パラメータ ID / 番号 / 説明 RBLE_VS_PARAM_EEL_ID_MODFWVER / 4 / ファームウェアバージョン
	size	データサイズ(1 ～ 255 バイト) 設定パラメータ ID / データサイズ RBLE_VS_PARAM_EEL_ID_MODFWVER / 2
	**addr*	書き込み・読み出しバッファへのポインタ
Return:		
	RBLE_OK (0x00)	正常終了
	RBLE_STATUS_ERROR (0xF2)	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可

RBLE_VS_Flash_Access を実行すると、Data Flash データアクセスコマンド実行結果が RBLE_VS_EVENT_FLASH_ACCESS_COMP イベントで通知されます。パラメータの"*addr"にファームウェアバージョン格納先のポインタが返されます。

RBLE_VS_EVENT_FLASH_ACCESS_COMP	
このイベントは、Data Flash データアクセスコマンド実行結果を通知します。	
Parameters:	
status	Data Flash アクセスコマンド実行結果
cmd	実行コマンド
id	データ ID
size	データサイズ
**addr*	データ格納先へのポインタ。ファームウェアバージョンが格納されます。 フォーマットは以下になります。 [0] : マイナーバージョン [1] : メジャーバージョン 例) V1.00 : [1]=0x01, [0]=0x00

3.6　周辺機能

Host MCU のプログラムで必要な周辺機能を以下に示します。

表 3-4　Host MCU プログラム周辺機能

機能	用途
UART	シリアル通信に使用します。 －　ボーレート　：4800、9600、19200、38400、57600、115200、250000 (デフォルト：115200 bps) －　データ長　：8bit －　パリティ　：なし －　ストップビット　：1bit －　フロー制御　：なし －　データ転送順序　：LSB ファースト
タイマ	RSCIP タイムアウト監視に使用します。

3.7　ビルド方法

Host MCU のサンプルプログラムとして BLE_Sample フォルダに PC と RL78/G1D で動作する 2 種類のプログラムを用意しています。格納フォルダは以下または「5.2 フォルダ構成」を参照してください。

- PC　：¥Renesas_BLE_Module_V101¥BLE_Sample¥project¥windows
- RL78/G1D　：¥Renesas_BLE_Module_V101¥BLE_Sample¥project¥e2studio¥BLE_Sample¥rBLE_sample

3.7.1　rBLE_sample のビルド方法

1. Microsoft Visual Studio Express 2015 for Windows Desktop を起動します。
2. [ファイル] - [プロジェクトを開く]を選択し、プロジェクトを開くダイアログを表示します。
3. プロジェクトファイルが格納されているパスを開き、rBLE_Sample.vcxproj を選択し[開く]をクリックします。
   - フォルダ例)
     ¥Renesas_BLE_Module_V101¥ BLE_Sample¥project¥windows
4. [ソリューション エクスプローラー] - [rBLE_Sample]の上で右クリックすると表示されるメニューから[ビルド]を選択し、ビルドを行います。
5. プロジェクトファイルが格納されるフォルダ直下に Debug フォルダが作成され、exe ファイル「rBLE_Sample.exe」が生成されます。
   - フォルダ例)
     ¥Renesas_BLE_Module_V101¥ BLE_Sample¥project¥windows¥Debug

3.7.2 RL78/G1D サンプルのビルド方法

1. e² studio を起動します。
2. [ファイル] - [インポート]を選択し、インポートダイアログを表示します。
3. [一般] - [既存プロジェクトをワークスペースへ]を選択し、[次へ]をクリックします。
4. [ルートディレクトリの選択]で e² studio のプロジェクトファイルが格納されているパスを選択し、[プロジェクト]に rBLE_sample が表示されることを確認します。
   - フォルダ例)
     ¥Renesas_BLE_Module_V101¥BLE_Sample¥project¥e2studio¥BLE_Sample¥rBLE_sample
5. [終了]をクリックします。
6. [プロジェクト・エクスプローラー] - [rBLE_sample]の上で右クリックすると表示されるメニューから[プロジェクトのビルド]を選択し、ビルドを行います。
7. プロジェクトファイルが格納されるフォルダ直下に DefaultBuild フォルダが作成され、ファームウェアの HEX ファイル「rBLE_sample_CCRL.hex」が生成されます。
   - フォルダ例)
     ¥Renesas_BLE_Module_V101¥BLE_Sample¥project¥e2studio¥BLE_Sample¥rBLE_sample¥DefaultBuild

4. モジュールファームウェア

モジュールに書き込まれているファームウェアは、BLE ソフトウェア V1.11 の Modem 構成をベースに作成されています。ここではファームウェアについて説明します。

- ファームウェア仕様
- 周辺機能
- 未使用端子処理
- ファームウェアのメモリマップ
- BLE ソフトウェアからの変更箇所

※注意事項

(1) コード・フラッシュ・メモリのブロック 254 には、出荷時検査フラグが書き込まれていますので、書き換えないでください。

(2) コード・フラッシュ・メモリのブロック 255 には、Bluetooth Device Address が書き込まれていますので、書き換えないでください。

(3) 統合開発環境で未使用のコード・フラッシュ・メモリを消去しない設定にしてください。
(「5.1 コード・フラッシュ・メモリ書き換え設定」を参照してください)

4.1 ファームウェア仕様

BLE ソフトウェア V1.11 からの変更内容、そしてファームウェアの設定を以下に示します。

- BLE ソフトウェア V1.11 からの変更内容
  - Host MCU との通信方式に UART 2 線分岐接続方式を選択
  - 独自プロファイル汎用双方向通信を追加
  - rBLE コマンドに以下の Vendor Specific コマンドを追加
    - ファームウェアバージョン読み出し
    - UART 2 線分岐接続方式のボーレート設定
    - Software Reset
    - Firmware Update 実行

- ファームウェア設定
  - BLE ソフトウェア構成　：Modem 構成
  - 同時接続台数　：6 台
  - CPU 動作周波数　：8MHz
  - RF スロー・クロック用オンチップ・オシレータ　：使用
  - DC-DC コンバータ　：使用
  - Host MCU -モジュール間通信方式　：UART 2 線分岐接続方式
  - ボーレート　：4800、9600、19200、38400、57600、115200、250000 (デフォルト：115200 bps)

- UART 設定
  - データ長　：8bit
  - パリティ　：なし
  - ストップビット　：1bit
  - フロー制御　：なし
  - データ転送順序　：LSB ファースト

4.2　周辺機能

ファームウェアが使用する RL78/G1D モジュールの周辺機能を以下に示します。

表 4-1 ファームウェアで使用する RL78/G1D モジュール周辺機能

機能		機能名	用途
データ・フラッシュ		使用	BD アドレス、ボーレート番号を格納
12bit インターバルタイマ		使用	RF スロー・クロック用オンチップ・オシレータの監視に使用
タイマ・アレイ・ユニット		タイマ 00	rBLE コマンドのウェイトに使用
ポート機能		P30/INTP3 (入力)	UART 2 線分岐接続方式 外部起床用トリガ入力端子 (INTP3)
シリアル・アレイ・ユニット		UART0	Host MCU または PC との UART 2 線分岐接続方式通信用
乗除積和算器		使用	e² studio/CS+ for CC(CC-RL)において新たにプロジェクトを作成する場合には、必ず「演算器を使用する」を選択してください。
DMA コントローラ		DMA0、DMA1 DMA2、DMA3	DMA0、DMA1：UART 2 線分岐接続方式通信用 DMA2、DMA3：MCU-RF 接続用
割り込み	外部端子	INTRF INTP3	INTRF：RF 部から割り込み INTP3：UART 2 線分岐接続方式 外部起床用トリガ
	DMA	INTDMA0 INTDMA1 INTDMA2 INTDMA3	INTDMA0、INTDMA1：UART 2 線分岐接続方式通信用 INTDAM2、INTDMA3：MCU-RF トランシーバ接続用
	シリアル・アレイ・ユニット	INTST0 INTSR0 INTSRE0	UART 2 線分岐接続方式通信用
	12bit インターバルタイマ	INTIT	RF スロー・クロック用オンチップ・オシレータの監視に使用
	タイマ・アレイ・ユニット	INTTM00	rBLE コマンドのウェイトに使用

4.3　未使用端子処理

ファームウェアでのモジュール未使用端子処理を表 **4-2** の通りに設定しています。

表 4-2 未使用端子処理

端子番号	端子名	ソフトウェア設定		
		機能	入出力	内蔵プルアップ
1	GND	-	-	-
2	P30/INTP3/RTC1HZ	P30	入力	接続しない
3	P16/TI01/TO01/INTP5	P16	入力	接続する
4	P15/SCK20/SCL20(TI02)/(TO02)	P15	入力	接続する
5	P14/SI20/SDA20/(SCLA0)/(TI03)/(TO03)	P14	入力	接続する
6	P13/SO20/(SDAA0)/(TI04)/(TO04)	P13	入力	接続する
7	P12/SO00/TxD0/TOOLTxD/(TI05)/(TO05)	TxD0	出力	-
8	P11/SI00/RxD0/TOOLRxD/SDA00/(TI06)/(TO06)	RxD0	入力	接続しない
9	P10/SCK00/SCL00/(TI07)/(TO07)	P10	入力	接続する
10	VDD	-	-	-
11	GND		-	-
12	P147/ANI18	P147	入力	接続する
13	P23/ANI3	P23	出力	-
14	P22/ANI2	P22	出力	-
15	P21/ANI1/AVREFM	P21	出力	-
16	P20/ANI0/AVREFP	P20	出力	-
17	P03/ANI16/RxD1	P03	出力	-
18	P02/ANI17/TxD1	P02	入力	接続する
19	P01/TO00	P01	入力	接続する
20	GND	-	-	-
21	P00/TI00	P00	入力	接続する
22	P120/ANI19	P120	出力	-
23	P40/TOOL0	P40	入力	接続しない
24	/RESET	/RESET	入力	接続しない
25	P137/INTP0	P137	入力	接続しない
26	P124/XT2/EXCLKS	P124	入力	接続しない
27	P123/XT1	P123	入力	接続しない
28	P60/(SCLA0)	P60	入力	接続しない
29	P61/(SDAA0)	P61	入力	接続しない
30	GND	-	-	-
31	IC	-	-	-
32～42	GND	-	-	-

4.4　メモリマップ

ファームウェアのメモリマップを以下に示します。 色のセクションはユーザコードセクションです。

表 4-3 メモリマップ(Code Flash)

Memory	Block Number	Section Name	Start Address	End Address	Size (byte)	Brief
Code Flash	000-003	.vect	0x00000	0x0007F	128	ベクタテーブル
		.callt0	0x00080	0x0009D	30	CALLT テーブル
		.option_byt	0x000C0	0x000C3	4	オプション・バイト
		.security_i	0x000C4	0x000CD	10	セキュリティ ID
		.monitor1	0x000CE	0x000D7	10	OCD モニタ
		ROM_BOOT0	0x000D8	0x00125	78	ブートコード
		.RLIB	0x00126	0x00910	2027	ランタイムライブラリ
		MAIN_CN0_f	0x00FFE	0x00FFF	2	FW アップデート回数管理
	004-007	MAIN_CN1_f	0x01FFE	0x01FFF	2	FW アップデート回数管理
	008	.SLIB	0x02000	0x02011	18	標準ライブラリ
	012-016	HDB_CNST_n	0x03000	0x0312B	300	Host データベース
		HST_CNST_n	0x0312C	0x03B3D	2578	Host スタック CONST データ
		CNT_CNST_n	0x03B3E	0x0409B	1374	Controller スタック CONST データ
		RBL_CNST_n	0x0409C	0x042ED	594	RBLE CONST データ
		MAINCNST_n	0x042EE	0x04343	86	main CONST データ
		DFL_CNST_f	0x04344	0x04361	30	DFL CONST データ
		MOD_CNST_n	0x04362	0x0437D	28	モジュール CONST データ
	017-021	.const	0x04400	0x05599	4506	ユーザ CONST データ
	024-038	RBL_CODE_f	0x06000	0x098F1	14578	RBLE コード
		PLF_CODE_f	0x098F2	0x09994	163	Platform コード
		UARTCODE_f	0x09995	0x09B1B	391	UART コード
	039-040	.monitor2	0x09C00	0x09DFF	512	OCD モニタ
		.constf	0x09E00	0x09E00	0	ユーザ CONST データ
		.sdata	0x09E00	0x09E00	0	ユーザ初期値ありデータ
		.data	0x09E00	0x09EC5	198	ユーザ初期値ありデータ
		.text	0x09EC6	0x09EFE	57	ユーザコード
		UARTCODE_n	0x09EFF	0x0A065	359	UART コード
	042	ACS_TBL_n	0x0A800	0x0A827	40	アクセステーブル 0
		CLK_TBL_n	0x0A850	0x0A85B	12	クロックテーブル 0
		TSK_DESC_n	0x0A880	0x0AA7B	508	タスクディスクリプタ 0
	043	ACS_TBL_n	0x0AC00	0x0A827	0	アクセステーブル 1
		CLK_TBL_n	0x0AC50	0x0A85B	0	クロックテーブル 1
		TSK_DESC_n	0x0AC80	0x0AA7B	0	タスクディスクリプタ 1
	044-140	CNT_CODE_n	0x0B000	0x0B3AB	940	Controller スタックコード
		HST_CODE_n	0x0B3AC	0x0B42C	129	Host スタックコード
		RBL_CODE_n	0x0B42D	0x0B45C	48	RBLE コード
		MOD_CODE_n	0x0B45D	0x0B48C	48	モジュールコード
		FDL_CNST_f	0x0B48E	0x0B497	10	FDL CONST データ
		EEL_CNST_f	0x0B498	0x0B49E	7	EEL CONST データ
		FDL_CODE	0x0B4A0	0x0B6C3	548	FDL コード
		EEL_CODE	0x0B6C4	0x0C138	2677	EEL コード
		FSL_FCD	0x0C13A	0x0C1F0	183	FSL 初期化コード
		FSL_BCD	0x0C1F2	0x0C2F1	256	FSL コード
		FSL_BECD	0x0C2F2	0x0C371	128	FSL コード
		FSL_RCD	0x0C372	0x0C3D1	96	FSL コード
		DFL_CODE_f	0x0C3D2	0x0C747	886	Data Flash コード
		CFL_CODE_f	0x0C748	0x0C923	476	Code Flash コード
		MAINCODE_f	0x0C924	0x0CFB1	1678	main コード
		HST_CODE_f	0x0CFB2	0x17C04	44115	Host スタックコード
		CNT_CODE_f	0x17C05	0x22F80	45948	Controller スタックコード
		MOD_CODE_f	0x22F81	0x230AF	303	モジュールコード
	142-184	.textf	0x23800	0x2E0BB	43196	ユーザコード/Profile コード

表 4-4 メモリマップ(RAM)

MEMORY	SECTION NAME	START ADDRESS	END ADDRESS	SIZE (byte)	Brief
RAM	CNT_DATA_n	0xFB310	0xFB93D	1582	Controller スタックデータ
	MAINDATA_n	0xFB93E	0xFB9D9	156	main データ
	HDB_DATA_n	0xFB9DA	0xFBA59	128	Host データベースデータ
	HST_DATA_n	0xFBA5A	0xFBCAF	598	Host スタックデータ
	RBL_DATA_n	0xFBCB0	0xFC1F7	1352	RBLE データ
	CFL_DATA_n	0xFC1F8	0xFC1FF	8	Code Flash データ
	DFL_DATA_n	0xFC200	0xFC207	8	Data Flash データ
	UARTDATA_n	0xFC208	0xFC226	31	UART データ
	MOD_DATA_n	0xFC228	0xFC22A	3	モジュールデータ
	.bss	0xFC230	0xFF9F3	14276	ユーザ初期値無しデータ
	.dataR	0xFF9F4	0xFFAB9	198	ユーザ初期値ありデータ
	FDL_SDAT	0xFFE30	0xFFE31	2	FDL データ
	EEL_SDAT	0xFFE32	0xFFE32	1	EEL データ
	.sbss	0xFFE34	0xFFE34	0	ユーザ初期値無しデータ
	.sdataR	0xFFE34	0xFFE34	0	ユーザ初期値ありデータ

4.5 ソースファイル変更箇所

BLE ソフトウェアからファームウェアへのソースファイル変更箇所を示します。

4.5.1 プロファイルの選択

Bluetooth SIG で規定されているプロファイルの選択を変更しています。ファームウェアのプロファイルを変更する場合は、定義値を変更してください。"0"で無効。"1"で有効です。

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78¥prf_sel.h

```
38:  #define PRF_SEL_PXPM   1   /* Proximity Profile Monitor role */
39:  #define PRF_SEL_PXPR   1   /* Proximity Profile Reporter role */
40:  #define PRF_SEL_FMPL   1   /* Find Me Profile Locator role */
41:  #define PRF_SEL_FMPT   1   /* Find Me Profile Target role */
42:  #define PRF_SEL_HTPC   1   /* Health Thermometer Profile Collector role */
43:  #define PRF_SEL_HTPT   1   /* Health Thermometer Profile Thermometer role */
44:  #define PRF_SEL_BLPC   1   /* Blood Pressure Profile Collector role */
45:  #define PRF_SEL_BLPS   1   /* Blood Pressure Profile Sensor role */
46:  #define PRF_SEL_HGHD   0   /* HID over GATT Profile HID Device role */
47:  #define PRF_SEL_HGBH   0   /* HID over GATT Profile Boot Host role */
48:  #define PRF_SEL_HGRH   0   /* HID over GATT Profile Report Host role */
49:  #define PRF_SEL_SPPC   0   /* Scan Parameters Profile Scan Client role */
50:  #define PRF_SEL_SPPS   0   /* Scan Parameters Profile Scan Server role */
51:  #define PRF_SEL_HRPC   1   /* Heart Rate Profile Collector role */
52:  #define PRF_SEL_HRPS   1   /* Heart Rate Profile Sensor role */
53:  #define PRF_SEL_CSCC   0   /* Cycling Speed and Cadence Profile Collector role */
54:  #define PRF_SEL_CSCS   0   /* Cycling Speed and Cadence Profile Sensor role */
55:  #define PRF_SEL_GLPC   1   /* Glucose Profile Collector role */
56:  #define PRF_SEL_GLPS   1   /* Glucose Profile Sensor role */
57:  #define PRF_SEL_CPPC   0   /* Cycling Power Profile Collector role */
58:  #define PRF_SEL_CPPS   0   /* Cycling Power Profile Sensor role */
59:  #define PRF_SEL_TIPC   1   /* Time Profile Client role */
60:  #define PRF_SEL_TIPS   1   /* Time Profile Server role */
61:  #define PRF_SEL_ANPC   1   /* Alert Notification Profile Client role */
62:  #define PRF_SEL_ANPS   1   /* Alert Notification Profile Server role */
63:  #define PRF_SEL_LNPS   0   /* Location and Navigation Profile Sensor role */
64:  #define PRF_SEL_LNPC   0   /* Location and Navigation Profile Collector role */
65:  #define PRF_SEL_PASC   1   /* Phone Alert Status Profile Server role */
66:  #define PRF_SEL_PASS   1   /* Phone Alert Status Profile Client role */
67:  #define PRF_SEL_RSCC   1   /* Running Speed and Cadence Profile Collector role */
68:  #define PRF_SEL_RSCS   1   /* Running Speed and Cadence Profile Sensor role */
```

4.5.2 UART 2 線分岐接続方式

ファームウェアは Host MCU との通信に UART 2 線分岐接続方式を選択しています。UART 2 線分岐接続方式の場合、WAKEUP も同時に有効にします。"0"で無効。"1"で有効です。

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥serial¥serial.h

```
51:  #define SERIAL_U_2WIRE     (0)
52:  #define SERIAL_U_3WIRE     (0)
53:  #define SERIAL_U_DIV_2WIRE (1)
54:  #define SERIAL_C_4WIRE     (0)
55:  #define SERIAL_C_5WIRE     (0)
56:  #define SERIAL_I_3WIRE     (0)
```

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥wakeup¥wakeup.c

```
26:    #define USE_WAKEUP_SIGNAL_PORT  (1) /* Modem Setting & Uart divide 2wire */
```

4.5.3 汎用双方向通信サービス

汎用双方向通信の GATT データベースを追加しています。

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78¥prf_config.c

```
137: /** Virtual UART Service */
138: static const uint8_t vuart_svc[RBLE_GATT_128BIT_UUID_OCTET] = RBLE_SVC_VUART;
139:
140: /** Virtual UART Service Indication Characteristic */
141: static const struct atts_char128_desc vuart_indication_char = {RBLE_GATT_CHAR_PROP_IND,
142:  {(uint8_t)(VUART_HDL_INDICATION_VAL & 0xFF),
       (uint8_t)(VUART_HDL_INDICATION_VAL >> 8) & 0xFF},
143:   RBLE_CHAR_VUART_INDICATION};
144:
145: uint8_t vuart_indication_char_val[20] = {0};
146: struct atts_elmt_128 vuart_indication_char_val_elmt = {RBLE_CHAR_VUART_INDICATION,
147:                                                       RBLE_GATT_128BIT_UUID_OCTET,
148:                                                  &vuart_indication_char_val[0]};
149:
150: /** Virtual UART Service Write Characteristic */
151: static const struct atts_char128_desc vuart_write_char = {RBLE_GATT_CHAR_PROP_WR,
152:  {(uint8_t)(VUART_HDL_WRITE_VAL & 0xFF), (uint8_t)(VUART_HDL_WRITE_VAL >> 8) & 0xFF},
153:   RBLE_CHAR_VUART_WRITE};
154: uint8_t vuart_write_char_val[20] = {0};
155: struct atts_elmt_128 vuart_write_char_val_elmt = {RBLE_CHAR_VUART_WRITE,
156:                                                  RBLE_GATT_128BIT_UUID_OCTET,
157:                                                  &vuart_write_char_val[0]};
158: uint16_t vuart_indication_enable = 0x0000u;

:

2100: { RBLE_DECL_PRIMARY_SERVICE,
2101:   sizeof(vuart_svc), sizeof(vuart_svc),
       TASK_ATTID(TASK_RBLE, VUART_IDX_SVC), RBLE_GATT_PERM_RD, (void *)&vuart_svc },
2102:    { RBLE_DECL_CHARACTERISTIC,
2103:      sizeof(vuart_indication_char), sizeof(vuart_indication_char),
         TASK_ATTID(TASK_RBLE, VUART_IDX_INDICATION_CHAR),
         RBLE_GATT_PERM_RD, (void *)&vuart_indication_char },
2104: { DB_TYPE_128BIT_UUID,
2105:   sizeof(vuart_indication_char_val), sizeof(vuart_indication_char_val),
       TASK_ATTID(TASK_RBLE, VUART_IDX_INDICATION_VAL), RBLE_GATT_PERM_NI,
       (void *)&vuart_indication_char_val_elmt },
2106: { RBLE_DESC_CLIENT_CHAR_CONF,
2107:   sizeof(vuart_indication_enable), sizeof(vuart_indication_enable),
       TASK_ATTID(TASK_RBLE, VUART_IDX_INDICATION_CFG),
       (RBLE_GATT_PERM_RD|RBLE_GATT_PERM_WR), (void *)&vuart_indication_enable },
2108:    { RBLE_DECL_CHARACTERISTIC,
2109:      sizeof(vuart_write_char), sizeof(vuart_write_char),
         TASK_ATTID(TASK_RBLE, VUART_IDX_WRITE_CHAR), RBLE_GATT_PERM_RD,
         (void *)&vuart_write_char },
2110: { DB_TYPE_128BIT_UUID,
2111:   sizeof(vuart_write_char_val), sizeof(vuart_write_char_val),
       TASK_ATTID(TASK_RBLE, VUART_IDX_WRITE_VAL), RBLE_GATT_PERM_WR,
       (void *)&vuart_write_char_val_elmt },
```

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78¥prf_config.h

```
500: VUART_IDX_SVC,
501: VUART_IDX_INDICATION_CHAR,
502: VUART_IDX_INDICATION_VAL,
503: VUART_IDX_INDICATION_CFG,
504: VUART_IDX_WRITE_CHAR,
505: VUART_IDX_WRITE_VAL,
```

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78¥db_handle.h

```
VUART_HDL_SVC,
VUART_HDL_INDICATION_CHAR,
VUART_HDL_INDICATION_VAL,
VUART_HDL_INDICATION_CFG,
VUART_HDL_WRITE_CHAR,
VUART_HDL_WRITE_VAL,
```

4.5.4　rBLE コマンド処理

rBLE API の Vendor Specific 機能に追加したコマンド処理を行う関数について説明します。

RBLE_VS_Set_Params 関数は、第 1 引数 param_id に 0x80 以降が指定されると arch_main.c の RBLE_User_Set_Params 関数を呼び出します。RBLE_User_Set_Params 関数では渡された param_id に対応した以下の 3 つの処理を行います。

- ボーレート設定
- Firmware Update
- Software Reset

RBLE_VS_Flash_Access 関数は、dataflash.c の dataflash_rw 関数を呼び出します。パラメータで指定した値によりファームウェアから情報を読み出す処理を行います。

- ファームウェアバージョン読み出し

RBLE_User_Set_Params 関数、dataflash_rw 関数の説明を以下に示します。

(1) RBLE_User_Set_Params

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78¥arch_main.c

RBLE_STATUS RBLE_User _Set_Params (uint8_t param_id, uint8_t param_len, uint8_t *param_data)

このファンクションは、rBLE API である RBLE_VS_Set_Params 関数パラメータ param_id の 0x80 以降を使用することで呼び出され、param_id ごとの処理を追加することができます。

- ボーレート設定は、ボーレート番号をデータ・フラッシュに保存します。モジュール起動時にボーレート番号を読み出しシリアル通信ドライバを初期化します。ボーレート設定を実行した場合、リセットを行ってください。ボーレートはリセット後、有効になります。
- Firmware Update モードへの移行を実行した場合、1 秒後に Firmware Update モードへ移行します。
- Software Reset は、不正命令の実行による内部リセットを行います。Software Reset を実行した場合、1 秒後に内部リセットが発生します。

Parameters:

param_id

設定パラメータ ID

設定パラメータ ID	番号	説明
MODFW_SET_PARAM_ID_BAUDID	0x80	ボーレート設定
MODFW_SET_PARAM_ID_FWUP	0xD9	Firmware Update モードへの移行
MODFW_SET_PARAM_ID_SOFTRST	0xFF	Software Reset

param_len

パラメータ長

設定パラメータ ID	パラメータ長
MODFW_SET_PARAM_ID_BAUDID	UARTBAUD_ID_LEN(1)
MODFW_SET_PARAM_ID_BAUDID	使用しません
MODFW_SET_PARAM_ID_SOFTRST	使用しません

**param_data*

パラメータ格納先へのポインタ(データは下位バイトより前詰め)

設定パラメータ ID	ボーレート番号
MODFW_SET_PARAM_ID_BAUDID	0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps 6: 250000 bps
MODFW_SET_PARAM_ID_BAUDID	使用しません
MODFW_SET_PARAM_ID_SOFTRST	使用しません

Return:

RBLE_OK (0x00)	正常終了
RBLE_ERR (0xF0)	データ・フラッシュ保存エラー
RBLE_STATUS_ERROR (0xF2)	rBLE モードが RBLE_MODE_ACTIVE 以外のため実行不可
RBLE_UNSUPPORTED (0x11)	未サポートの param_id
RBLE_PARAM_ERR (0xF3)	パラメータエラー

(2) dataflash_rw

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥dataflash¥dataflash.c

uint8_t dataflash_rw(const uint8_t mode, const df_rw_t rw, const uint8_t id, uint8_t* addr)		
このファンクションは、Data Flash へデータの書き込みまたは、データの読み出しを行います。ファームウェアではこの関数を利用し、BD アドレス、ボーレート ID の書き込みと読み出し。モジュール FW バージョンを読み出します。		
Parameters:		
	mode	Data Flash モード DF_MODE_POLLING(0x00)　：Polling Mode DF_MODE_ENFORCED(0xFF)　：Enforced Mode
	rw	Read or Write DF_READ(0)　：読み出し DF_WRITE(1)：書き込み
	id	Data ID(1 ～ 255 バイト) EEL_ID_BDA(0)　：BD アドレス EEL_ID_REMOTE_BDA(1)　：リモート BD アドレス EEL_ID_UARTBAUD(2)　：ボーレート ID EEL_ID_MODFW_VER(3)　：モジュール FW バージョン読み出し
	**addr*	書き込み、読み出しバッファへのポインタ
Return:		
	DF_OK (0x00)	正常終了
	DF_ERR_BUSY (0x01)	Data Flash アクセス状態
	DF_ERR_DISALLOWED (0x03)	実行禁止

4.5.5 ファームウェア関数

モジュールの初期化や追加した rBLE コマンドの処理を行う関数を追加しています。以下に作成した関数を示します。

Source File : ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥module¥module.c
¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥module¥module.h

(1) r_modfw_init

void r_modfw_init(void)	
モジュールのポート設定を行います。	
Parameters:	
なし	
Return:	
なし	

(2) r_modfw_flash_write_uart_baudrate_id

RBLE_STATUS r_modfw_flash_write_uart_baudrate_id(uint8_t param_len, uint8_t *param_data)		
RBLE_User_Set_Params 関数から呼び出され、ボーレート ID を Data Flash に書き込みます。		
Parameters:		
	param_len	ボーレート ID 長
	**param_data*	ボーレート ID
Return:		
	RBLE_OK (0x00)	正常終了
	RBLE_ERR (0xF0)	Data Flash 関数実行失敗
	RBLE_PARAM_ERR (0xF3)	パラメータエラー

(3) r_modfw_flash_read_uart_baudrate_id

void r_modfw_flash_read_uart_baudrate_id(void)
モジュール起動時に呼び出され、ボーレート ID を Data Flash から読み出しグローバル変数にセットします。
Parameters:
なし
Return:
なし

(4) r_modfw_force_reset

void r_modfw_force_reset(void)
RBLE_User_Set_Params 関数から呼び出され、タイマで 1 秒待ちます。Software Reset は割り込みハンドラ(r_modfw_inttm00_isr)から実行します。
Parameters:
なし
Return:
なし

(5) r_modfw_update

void r_modfw_update(void)	
RBLE_User_Set_Params 関数から呼び出され、タイマで 1 秒待ちます。Firmware Update は割り込みハンドラ(r_modfw_inttm00_isr)から実行します。	
Parameters:	
なし	
Return:	
なし	

(6) r_modfw_get_ver

void r_modfw_get_ver(uint8_t* addr)	
dataflash_rw 関数から呼び出され、ファームウェアバージョンを返します。	
Parameters:	
**addr*	ファームウェアバージョンを格納するバッファのポインタ
Return:	
なし	

(7) r_modfw_set_timer_1s

void r_modfw_set_timer_1s(void)	
1 秒ウェイトするタイマ用に、タイマ 00 の初期化と開始を行います。	
Parameters:	
なし	
Return:	
なし	

(8) r_modfw_stop_timer

void r_modfw_stop_timer(void)	
タイマ 00 停止します。	
Parameters:	
なし	
Return:	
なし	

(9) r_modfw_inttm00_isr

void r_modfw_inttm00_isr(void)	
タイマ 00 の割り込みハンドラです。Software Reset と、Firmware Update を実行します。	
Parameters:	
なし	
Return:	
なし	

4.5.6　RF スロー・クロック

ファームウェアの初期設定では、RF 内部回路で使用する 32.768kHz クロック(RF スロー・クロック)に RF スロー・クロック用オンチップ・オシレータを使用します。

XT1、XT2 端子に 32.768kHz の水晶振動子を接続して、RF スロー・クロックとして使用する場合には、以下の定義マクロの先頭から"no"を削除して"CLK_SUB_XT1"を指定してください。

コンパイラ定義マクロ

noCLK_SUB_XT1

- 定義マクロ

1. e² studio を起動します。
2. [ファイル] - [インポート]を選択し、インポートダイアログを表示します。
3. [一般] - [既存プロジェクトをワークスペースへ]を選択し、[次へ]をクリックします。
4. [ルートディレクトリの選択]で e² studio のプロジェクトファイルが格納されているパスを選択し、[プロジェクト]に rBLE_Mdm が表示されることを確認します。
   - フォルダ例) ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm
5. [終了]をクリックします。
6. [プロジェクト･エクスプローラー] - [rBLE_Mdm]の上で右クリックすると表示されるメニューから[Renesas Tool Settings]を選択します。
7. [Compiler] - [ソース]を選択し、[定義マクロ]テキストボックスを表示します。

4.6 ビルド方法

ファームウェアのビルド方法を示します。「2.3 ビルド準備」を参照し、各ライブラリの入手と、所定フォルダへのコピーを行ってください。

- Renesas Bluetooth Low Energy プロトコルスタック V1.11 CC-RL 向けライブラリファイル
- RL78 ファミリ CC-RL コンパイラ用 EEPROM エミュレーションライブラリ Pack02 Ver.1.01
- RL78 ファミリフラッシュセルフプログラミングライブラリ Type01 パッケージ Ver.2.21B (CA78K0R/CC-RL コンパイラ用)

4.6.1 ファームウェアのビルド

1. e² studio を起動します。
2. [ファイル] - [インポート]を選択し、インポートダイアログを表示します。
3. [一般] - [既存プロジェクトをワークスペースへ]を選択し、[次へ]をクリックします。
4. [ルートディレクトリの選択]で e² studio のプロジェクトファイルが格納されているパスを選択し、[プロジェクト]に rBLE_Mdm が表示されることを確認します。
   - フォルダ例) ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm
5. [終了]をクリックします。
6. [プロジェクト・エクスプローラー] - [rBLE_Mdm]の上で右クリックすると表示されるメニューから[プロジェクトのビルド]を選択し、ビルドを行います。
7. プロジェクトファイルが格納されるフォルダ直下に DefaultBuild フォルダが作成され、ファームウェアの HEX ファイル「rBLE_Mdm_CCRL.hex」が生成されます。
   - フォルダ例) ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm¥DefaultBuild

4.6.2　ファームウェアアップデートファイル

ファームウェアアップデートで用いるバイナリファイルは、ファームウェア「rBLE_Mdm_CCRL.hex」と同じフォルダに作成されます。作成されるファイル名は「rBLE_Mdm_CCRL.bin」です。

ファームウェアアップデートの動作確認で使用する場合は、バイナリファイルを「RL78_G1D_MODULE.bin」にリネームしてください。使用方法は「RL78/G1D モジュール ファームウェア ユーザーズマニュアル」(R01UW0160)を参照してください。

※注意事項

- ファームウェアアップデートではプロファイルの変更のみ行ってください。
- セクションの配置を変更しないでください。
- ファームウェアアップデートで使用するバイナリファイルは、ユーザコードセクション以外のメモリ配置をファームウェアと同一にしなければいけません。必ずファームウェアのバイナリファイルとの差分を確認してください。ユーザコードセクションの配置については「4.4 メモリマップ」の色部分を参照してください。

バイナリファイルを作成するためのオプション設定を以下に示します。

1. [プロジェクト･エクスプローラー] - [rBLE_Mdm]の上で右クリックすると表示されるメニューから[Renesas Tool Settings]を選択します。
2. 設定メニューから[ビルド・ステップ]タブの[ビルド後のステップ] - [コマンド]の欄に、下記コマンドを設定します。
   - rlink -form=binary "rBLE_Mdm_CCRL.abs" -output="rBLE_Mdm_CCRL.bin"=0-3ffff -space=ff
3. [プロジェクト・エクスプローラー] - [rBLE_Mdm]の上で右クリックすると表示されるメニューから[プロジェクトのビルド]を選択し、ビルドを行います。
4. プロジェクトファイルが格納されるフォルダ直下に DefaultBuild フォルダが作成され、ファームウェアアップデートバイナリファイル「rBLE_Mdm_CCRL.bin」が生成されます。
   - ¥Renesas_BLE_Module_V101¥RL78_G1D¥Project_Source¥renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm¥DefaultBuild

5. Appendix

5.1 コード・フラッシュ・メモリ書き換え設定

コード・フラッシュ・メモリのブロック 254、255 には出荷時検査フラグと Bluetooth Device Address が書き込まれています。統合開発環境と E1 エミュレータを使用してファームウェアをモジュールにダウンロードする場合、コード・フラッシュ・メモリを消去しない設定にしてください。

- e² studio

1. [プロジェクト・エクスプローラー] - [(プロジェクト名)]の上で右クリックすると表示されるメニューから[プロパティ]を選択し、「プロパティ:(プロジェクト名)」ダイアログを表示します。
2. [実行/デバッグ設定]に表示された拡張子".x"のファイルを選択し、[編集]をクリックします。
3. [構成の編集]ダイアログの[Debugger]タブ-[Connection Settings]タブを選択します。
4. [フラッシュ] - [起動時にフラッシュ ROM を消去]で「いいえ」を選択し、[OK]をクリックします。

- CS+

1. [プロジェクト・ツリー] . -[(プロジェクト名)(サブプロジェクト)]- [RL78 E1(Serial)(デバッグ・ツール)]を選択し[プロパティ]タブを表示します。
2. [ダウンロード・ファイル設定]タブを選択し、[ダウンロード] - [ダウンロード・モードの選択]で「データ優先」を選択します。

5.2 フォルダ構成

5.2.1 モジュールファームウェア V1.01 格納フォルダ

(1) ルートフォルダ

```
Renesas_BLE_Module_V101
├ BLE_Sample                                  Host MCU プログラム格納フォルダ
├ RL78_G1D                                   モジュールファームウェア格納フォルダ
└ ROM_File                                   モジュールファームウェア HEX ファイル格納フォルダ
  └ rBLE_Mdm_CCRL_module_fw_v101.hex         モジュールファームウェア v1.01 HEX ファイル
```

(2) Host MCU プログラム格納フォルダ

```
BLE_Sample
├ project
│ ├ CS_CCRL                                  CS+ for CC 用プロジェクト格納フォルダ
│ │ └ BLE_Sample
│ │   ├ rBLE_sample
│ │   │ └ rBLE_sample.mtsp                   CS+ for CC サブプロジェクト
│ │   └ BLE_Sample.mtpj                      CS+ for CC プロジェクト
│ ├ e2studio                                 e² studio 用プロジェクト格納フォルダ
│ │ └ BLE_Sample
│ │   └ rBLE_sample
│ │     ├ .cproject                          ビルドオプション情報管理ファイル
│ │     ├ .DefaultBuildlinker                リンカオプション情報管理ファイル
│ │     ├ .info                              ツールチェイン情報管理ファイル
│ │     ├ .project                           プロジェクト情報管理ファイル
│ │     └ rBLE_sample_CCRL.x.launch          デバッグ情報管理ファイル
│ └ windows                                  PC 用プロジェクト格納フォルダ
│   ├ Exe
│   │ ├ rBLE_Sample.exe                      PC サンプルプログラム実行ファイル
│   │ ├ run_master.bat                       起動用(引数設定済み)バッチファイル
│   │ └ run_slave.bat                        起動用(引数設定済み)バッチファイル
│   ├ rBLE_Sample.sln
│   ├ rBLE_Sample.vcxproj                    Microsoft Visual Studio Express 2015 向けプロジェクト
│   └ rBLE_Sample.vcxproj.filters
└ src
  ├ Platform                                 Platform 依存ソースコード格納フォルダ
  │ ├ G1D_cs_iar                             組み込み依存ソースコード格納フォルダ
  │ │ ├ driver
  │ │ │ ├ console
  │ │ │ │ ├ Console.c                        組み込み向けコンソール入力サンプルドライバファイル
  │ │ │ │ └ console.h                        組み込み向けコンソール入力サンプルドライバヘッダファイル
  │ │ │ ├ led
  │ │ │ │ ├ led.c                            RL78 向け LED サンプルドライバファイル
  │ │ │ │ └ led.h                            RL78 向け LED サンプルドライバヘッダファイル
  │ │ │ ├ plf
  │ │ │ │ ├ plf.c                            RL78 プラットフォームサンプルドライバファイル
  │ │ │ │ └ plf.h                            RL78 プラットフォームサンプルドライバヘッダファイル
  │ │ │ ├ serial
  │ │ │ │ ├ csi.c                            RL78 向け CSI サンプルドライバファイル
  │ │ │ │ ├ csi.h                            RL78 向け CSI サンプルドライバヘッダファイル
  │ │ │ │ ├ iic_master.c                     RL78 向け IIC マスタサンプルドライバファイル
  │ │ │ │ ├ iic_master.h                     RL78 向け IIC マスタサンプルドライバヘッダファイル
  │ │ │ │ ├ serial.h                         RL78 向けシリアル通信サンプルドライバヘッダファイル
  │ │ │ │ ├ term_uart.c                      RL78 向けターミナル UART サンプルドライバファイル
  │ │ │ │ ├ term_uart.h                      RL78 向けターミナル UART サンプルドライバヘッダファイル
  │ │ │ │ ├ uart.c                           RL78 向け UART サンプルドライバファイル
  │ │ │ │ └ uart.h                           RL78 向け UART サンプルドライバヘッダファイル
  │ │ │ └ timer
```

```
│ │ │     ├ timer.c                    RL78 向けタイマサンプルドライバファイル
│ │ │     └ timer.h                    RL78 向けタイマサンプルドライバヘッダファイル
│ │ ├ include
│ │ │ ├ arch.h                          RL78 向け基本マクロ／関数定義サンプルヘッダファイル
│ │ │ ├ compiler.h                      RL78 向けコンパイラ固有命令サンプルヘッダファイル
│ │ │ ├ iodefine.h                      RL78 CC-RL 向けデバイスヘッダファイル
│ │ │ ├ ll.h                            RL78 向けローレベルマクロ定義サンプルヘッダファイル
│ │ │ ├ port.h                          RL78 向け汎用ポートサンプルドライバヘッダファイル
│ │ │ ├ rscip_api.h                     RSCIP API サンプルヘッダファイル
│ │ │ └ types.h                         共通型定義サンプルヘッダファイル
│ │ ├ cstart.asm                        CC-RL 向けスタートアップルーチンファイル
│ │ ├ hdwinit.asm                       CC-RL 向けハードウェア初期化ファイル
│ │ ├ rBLE_Sample.c                     RL78 向けサンプルファイル
│ │ └ stkinit.asm                       CC-RL 向けスタック初期化ファイル
│ └ win32_vc                            Windows 依存ソースコード格納フォルダ
│   ├ driver
│   │ ├ console
│   │ │ ├ Console.c                     Windows 向けコンソール入力サンプルドライバファイル
│   │ │ └ console.h                     Windows 向けコンソール入力サンプルドライバヘッダファイル
│   │ ├ serial
│   │ │ ├ serial_drv.h                  Windows 向けシリアルサンプルドライバインタフェースヘッダファイル
│   │ │ ├ Serial_drv_sample.cpp         Windows 向けシリアルサンプルドライバインタフェースファイル
│   │ │ ├ SerialIF.cpp                  Windows 向けシリアルサンプルドライバファイル
│   │ │ └ SerialIF.h                    Windows 向けシリアルサンプルドライバヘッダファイル
│   │ └ timer
│   │   ├ timer.cpp                    Windows 向けタイマサンプルドライバファイル
│   │   └ timer.h                      Windows 向けタイマサンプルドライバヘッダファイル
│   ├ include
│   │ └ types.h                        Windows 向けヘッダファイル
│   ├ rBLE_Sample.cpp                  Windows 向けサンプルファイル
│   ├ StdAfx.cpp
│   └ StdAfx.h
└ rBLE                                BLE サンプルプログラム格納フォルダ
  └ src
    ├ host
    │ ├ anp
    │ │ ├ rble_api_anpc.c              rBLE Host ANP Client ファイル                    rBLE_Host
    │ │ └ rble_api_anps.c              rBLE Host ANP Server ファイル                    rBLE_Host
    │ ├ blp
    │ │ ├ rble_api_blpc.c              rBLE Host BLP Collector ファイル                 rBLE_Host
    │ │ └ rble_api_blps.c              rBLE Host BLP Sensor ファイル                    rBLE_Host
    │ ├ cpp
    │ │ ├ rble_api_cppc.c              rBLE Host CPP Collector ファイル                 rBLE_Host
    │ │ └ rble_api_cpps.c              rBLE Host CPP Sensor ファイル                    rBLE_Host
    │ ├ cscp
    │ │ ├ rble_api_cscpc.c             rBLE Host CSCP Collector ファイル                rBLE_Host
    │ │ └ rble_api_cscps.c             rBLE Host CSCP Sensor ファイル                   rBLE_Host
    │ ├ fmp
    │ │ ├ rble_api_fmpl.c              rBLE Host FMP Locator ファイル                   rBLE_Host
    │ │ └ rble_api_fmpt.c              rBLE Host FMP Target ファイル                    rBLE_Host
    │ ├ gap
    │ │ └ rble_api_gap.c               rBLE Host GAP ファイル                           rBLE_Host
    │ ├ gatt
    │ │ └ rble_api_gatt.c              rBLE Host GATT ファイル                          rBLE_Host
    │ ├ glp
    │ │ ├ rble_api_glpc.c              rBLE Host GLP Collector ファイル                 rBLE_Host
    │ │ └ rble_api_glps.c              rBLE Host GLP Sensor ファイル                    rBLE_Host
    │ ├ hogp
    │ │ ├ rble_api_hgbh.c              rBLE Host HOGP Boot Host ファイル                rBLE_Host
    │ │ ├ rble_api_hghd.c              rBLE Host HOGP HID Device ファイル               rBLE_Host
    │ │ └ rble_api_hgrh.c              rBLE Host HOGP Report Host ファイル              rBLE_Host
    │ ├ hrp
    │ │ ├ rble_api_hrpc.c              rBLE Host HRP Collector ファイル                 rBLE_Host
```

```
│ │ └ rble_api_hrps.c          rBLE Host HRP Sensor ファイル                rBLE_Host
│ ├ htp
│ │ ├ rble_api_htpc.c          rBLE Host HTP Collector ファイル             rBLE_Host
│ │ └ rble_api_htpt.c          rBLE Host HTP Thermometer ファイル           rBLE_Host
│ ├ lnp
│ │ ├ rble_api_lnpc.c          rBLE Host LNP Collector ファイル             rBLE_Host
│ │ └ rble_api_lnps.c          rBLE Host LNP Sensor ファイル                rBLE_Host
│ ├ pasp
│ │ ├ rble_api_paspc.c         rBLE Host PASP Client ファイル               rBLE_Host
│ │ └ rble_api_pasps.c         rBLE Host PASP Server ファイル               rBLE_Host
│ ├ pxp
│ │ ├ rble_api_pxpm.c          rBLE Host PXP Monitor ファイル               rBLE_Host
│ │ └ rble_api_pxpr.c          rBLE Host PXP Reporter ファイル              rBLE_Host
│ ├ rscp
│ │ ├ rble_api_rscpc.c         rBLE Host RSCP Collector ファイル            rBLE_Host
│ │ └ rble_api_rscps.c         rBLE Host RSCP Sensor ファイル               rBLE_Host
│ ├ scpp
│ │ ├ rble_api_sppc.c          rBLE Host ScPP Client ファイル               rBLE_Host
│ │ └ rble_api_spps.c          rBLE Host ScPP Server ファイル               rBLE_Host
│ ├ sm
│ │ └ rble_api_sm.c            rBLE Host SMP ファイル                       rBLE_Host
│ ├ tip
│ │ ├ rble_api_tipc.c          rBLE Host TIP Client ファイル                rBLE_Host
│ │ └ rble_api_tips.c          rBLE Host TIP Server ファイル                rBLE_Host
│ ├ vs
│ │ └ rble_api_vs.c            rBLE Host VS ファイル                        rBLE_Host
│ ├ rble_host.c                rBLE Host ファイル                           rBLE_Host
│ └ rble_if_api_cb.c           rBLE Host Event ファイル                     rBLE_Host
├ include
│ ├ host
│ │ └ rble_host.h              rBLE Host ヘッダファイル                     rBLE_Host
│ ├ prf_sel.h                  プロファイル選択設定ヘッダファイル
│ ├ rble.h                     rBLE ヘッダファイル                          rBLE_Host
│ ├ rble_api.h                 rBLE API ヘッダファイル                      rBLE_Host
│ ├ rble_api_custom.h          Custom Profile 追加 API ヘッダファイル
│ ├ rble_api_fwup.h            FW アップデート追加 API ヘッダファイル
│ ├ rble_app.h                 サンプルプログラムヘッダファイル
│ └ rble_trans.h               rBLE パケットデータ用ヘッダファイル          rBLE_Host
├ rscip
│ ├ rscip.c                    RSCIP インタフェースファイル                 RSIP
│ ├ rscip.h                    RSCIP ヘッダファイル                         RSIP
│ ├ rscip_cntl.c               RSCIP 制御用ファイル                         RSIP
│ ├ rscip_cntl.h               RSCIP 制御用ヘッダファイル                   RSIP
│ ├ rscip_ext.h                RSCIP 外部要求ヘッダファイル                 RSIP
│ ├ rscip_uart.c               RSCIP シリアルファイル                       RSIP
│ └ rscip_uart.h               RSCIP シリアルヘッダファイル                 RSIP
├ sample_app                   サンプルアプリケーション格納フォルダ
│ ├ menu_sel.c                 メニュー選択サンプルファイル
│ ├ menu_sel.h                 メニュー選択サンプルヘッダファイル
│ ├ r_vuart_app.c              汎用双方向通信アプリケーションファイル
│ ├ r_vuart_app.h              汎用双方向通信アプリケーションヘッダファイル
│ ├ r_vuart_app_param.c        汎用双方向通信アプリケーションパラメータファイル
│ ├ r_vuart_console.c          汎用双方向通信コンソールドライバファイル
│ ├ r_vuart_console.h          汎用双方向通信コンソールドライバヘッダファイル
│ ├ rble_fw_up_sender_app.c    FW アップデート用サンプルプログラムファイル
│ ├ rble_sample_app.c          サンプルプログラムファイル
│ └ rble_sample_parameter.c    サンプルプログラム PTS 向けパラメータファイル
└ sample_profile               独自プロファイル格納フォルダ
  ├ fwup                       FW Update Profile 格納フォルダ
  │ └ fwups.c                  FW Update Profile Sender ファイル
  ├ scp                        Sample Custom Profile 格納フォルダ
  │ ├ scpc.c                   Sample Custom Profile Client ファイル
```

```
│  └ scps.c                    Sample Custom Profile Server ファイル
├ vuart                        汎用双方向通信格納フォルダ
│  ├ vuart.h                   汎用双方向通信共通ヘッダファイル
│  ├ vuartc.c                  汎用双方向通信 Client ファイル
│  ├ vuartc.h                  汎用双方向通信 Client ヘッダファイル
│  ├ vuarts.c                  汎用双方向通信 Server ファイル
│  └ vuarts.h                  汎用双方向通信 Server ヘッダファイル
└ db_handle.h                  Attribute database handles ヘッダファイル
```

(3)　モジュールファームウェア格納フォルダ

```
RL78_G1D
└ Project_Source
  ├ bleip                                   BLE スタック格納フォルダ
  │  └ src
  │     ├ common
  │     │  └ co_bt.h                        BLE スタック共通ヘッダファイル
  │     └ rwble
  │        ├ rwble.h                        BLE ソフトウェアエントリーポイントヘッダファイル
  │        ├ rwble_config.h                 BLE ソフトウェアコンフィギュレーションヘッダファイル
  │        └ rwble_mem_cont.h               最大接続数関連パラメータ外部参照ヘッダファイル
  ├ rBLE                                    rBLE 格納フォルダ
  │  └ src
  │     ├ include
  │     │  ├ rble.h                         rBLE ヘッダファイル
  │     │  ├ rble_api.h                     rBLE API ヘッダファイル
  │     │  ├ rble_api_custom.h              Custom Profile 追加 API ヘッダファイル
  │     │  ├ rble_api_fwup.h                FW Update Profile 追加 API ヘッダファイル
  │     │  ├ rble_api_vuart.h               汎用双方向通信追加 API ヘッダファイル
  │     │  ├ rble_app.h                     サンプルプログラムヘッダファイル
  │     │  ├ rble_rwke.h                    RWKE API ヘッダファイル
  │     │  ├ rble_trans.h                   rBLE パケットデータ用ヘッダファイル
  │     │  └ rscip_api.h                    RSCIP API ヘッダファイル
  │     ├ sample_app
  │     │  ├ Console.c                      コンソール入力サンプルドライバファイル
  │     │  ├ console.h                      コンソール入力サンプルドライバヘッダファイル
  │     │  ├ menu_sel.c                     メニュー選択サンプルファイル
  │     │  ├ menu_sel.h                     メニュー選択サンプルヘッダファイル
  │     │  ├ rble_fw_up_receiver_app.c      FW アップデート用サンプルプログラムファイル
  │     │  ├ rble_sample_app.c              サンプルプログラムファイル
  │     │  └ rble_sample_parameter.c        サンプルプログラム PTS 向けパラメータファイル
  │     └ sample_profile
  │        ├ fwup                           FW Update Profile 格納フォルダ
  │        │  └ fwupr.c                     FW Update Profile Receiver ファイル
  │        └ scp                            Sample Custom Profile 格納フォルダ
  │           ├ scpc.c                      Sample Custom Profile Client ファイル
  │           └ scps.c                      Sample Custom Profile Server ファイル
  └ renesas
     ├ config
     │  └ split
     │     └ emb
     │        ├ r_lk.dr                     Embedded 構成用リンクディレクティブファイル
     │        ├ r_lk_fw_emb.dr              Embedded 構成 FW アップデートリンクディレクティブファイル
     │        ├ r_lk_fw_modem.dr            Modem 構成 FW アップデート用リンクディレクティブファイル
     │        └ r_lk_modem.dr               Modem 構成用リンクディレクティブファイル
     ├ lib                                  BLE ソフトウェアライブラリ格納フォルダ
     ├ src
     │  ├ arch
     │  │  └ rl78
     │  │     ├ ll
```

```
│ │     │ └ ll.h                          ローレベルマクロ定義ヘッダファイル
│ │     ├ arch.h                           BLE ソフトウェア基本マクロ／関数定義ヘッダファイル
│ │     ├ arch_main.c                      BLE ソフトウェアメインファイル
│ │     ├ config.h                         BLE ソフトウェアコンフィギュレーションヘッダファイル
│ │     ├ db_handle.h                      Attribute database handles ヘッダファイル
│ │     ├ fw_update_count0.c               Firmware Update ブートクラスタ書き換えカウント定義ファイル
│ │     ├ fw_update_count1.c               Firmware Update ブートクラスタ書き換えカウント定義ファイル
│ │     ├ hw_config.h                     BLE ソフトウェア H/W コンフィギュレーションヘッダファイル
│ │     ├ ke_conf.c                       RWKE タスク管理ファイル
│ │     ├ main.c                           メイン関数ファイル
│ │     ├ prf_config.c                     プロファイル向けパラメータ設定ファイル
│ │     ├ prf_config.h                     プロファイル向けパラメータ設定ヘッダファイル
│ │     ├ prf_config_host.c                プロファイル向けパラメータ設定ファイル
│ │     ├ prf_sel.h                       プロファイル選択設定ヘッダファイル
│ │     ├ rble_core_config.c               rBLE_Core 向けパラメータ設定ファイル
│ │     ├ rble_core_config.h               rBLE_Core 向けパラメータ設定ヘッダファイル
│ │     ├ rble_modem_config.c             MDM_APPL 向けパラメータ設定ファイル
│ │     ├ rble_modem_config.h             MDM_APPL 向けパラメータ設定ヘッダファイル
│ │     ├ rwble_mem.c                     可変メモリ確保設定ファイル
│ │     ├ rwble_mem.h                     可変メモリ確保設定ヘッダファイル
│ │     └ rwke_api.h                      RWKE API ヘッダファイル
│ ├ compiler
│ │ ├ ccrl
│ │ │ ├ cstart.asm                        CC-RL 向けスタートアップルーチンファイル
│ │ │ ├ hdwinit.asm                       CC-RL 向けハードウェア初期化ファイル
│ │ │ └ stkinit.asm                       CC-RL 向けスタック初期化ファイル
│ │ ├ compiler.h                          CC-RL 向けデバイスヘッダファイル
│ │ └ iodefine.h                          コンパイラ固有命令ヘッダファイル
│ ├ driver
│ │ ├ codeflash
│ │ │ ├ cc_rl                             CC-RL 向けコードフラッシュライブラリ格納用フォルダ
│ │ │ ├ codeflash.c                       コードフラッシュドライバファイル
│ │ │ └ codeflash.h                       コードフラッシュドライバヘッダファイル
│ │ ├ csi
│ │ │ ├ csi.c                             CSI ドライバファイル
│ │ │ └ csi.h                             CSI ドライバヘッダファイル
│ │ ├ dataflash
│ │ │ ├ cc_rl                             CC-RL 向けデータ･フラッシュライブラリ格納用フォルダ
│ │ │ ├ dataflash.c                       データ･フラッシュドライバファイル
│ │ │ ├ dataflash.h                       データ･フラッシュドライバヘッダファイル
│ │ │ ├ eel_descriptor.c                  EEL Pack1 向け EEPROM 記述子ファイル
│ │ │ ├ eel_descriptor.h                  EEL Pack1 向け EEPROM 記述子ヘッダファイル
│ │ │ ├ eel_descriptor_t02.c              EEL Pack2 向け EEPROM 記述子ファイル
│ │ │ ├ eel_descriptor_t02.h              EEL Pack2 向け EEPROM 記述子ヘッダファイル
│ │ │ ├ fal_descriptor.c                  EEL Pack1 向けフラッシュ記述子ファイル
│ │ │ ├ fal_descriptor.h                  EEL Pack1 向けフラッシュ記述子ヘッダファイル
│ │ │ ├ fdl_descriptor_t02.c              EEL Pack2 向けフラッシュ記述子ファイル
│ │ │ └ fdl_descriptor_t02.h              EEL Pack2 向けフラッシュ記述子ヘッダファイル
│ │ ├ DTM2Wire
│ │ │ ├ DTM2Wire.c                        2-Wire UART Direct Test Modeドライバファイル
│ │ │ └ DTM2Wire.h                        2-Wire UART Direct Test Modeドライバヘッダファイル
│ │ ├ iic
│ │ │ ├ iic_slave.c                       IIC スレーブドライバファイル
│ │ │ └ iic_slave.h                       IIC スレーブドライバヘッダファイル
│ │ ├ led
│ │ │ ├ led.c                             LED ドライバファイル
│ │ │ └ led.h                             LED ドライバヘッダファイル
│ │ ├ module
│ │ │ ├ module.c                          ファームウェアファイル
│ │ │ └ module.h                          ファームウェアヘッダファイル
│ │ ├ peak
│ │ │ ├ peak.h                            ピーク通知割り込みファイル
│ │ │ └ peak_isr.c                        ピーク通知ドライバヘッダファイル
```

```
│ │ ├ pktmon
│ │ │ └ pktmon.h                    HCI パケットモニタドライバヘッダファイル
│ │ ├ plf
│ │ │ ├ plf.c                       RL78 プラットフォームドライバファイル
│ │ │ └ plf.h                       RL78 プラットフォームドライバヘッダファイル
│ │ ├ port
│ │ │ └ port.h                      汎用ポートドライバヘッダファイル
│ │ ├ push_sw
│ │ │ ├ push_sw.c                   プッシュスイッチドライバファイル
│ │ │ └ push_sw.h                   プッシュスイッチドライバヘッダファイル
│ │ ├ rf
│ │ │ └ rf.h                        RF ドライバ共通ヘッダファイル
│ │ ├ serial
│ │ │ └ serial.h                    シリアル通信ドライバヘッダファイル
│ │ ├ uart
│ │ │ ├ uart.c                      UART ドライバファイル
│ │ │ └ uart.h                      UART ドライバヘッダファイル
│ │ └ wakeup
│ │   ├ wakeup.c                    ウェイクアップドライバファイル
│ │   └ wakeup.h                    ウェイクアップドライバヘッダファイル
│ └ types.h                          共通型定義ヘッダファイル
└ tools
  └ project
    ├ CS_CCRL                        CS+ for CC 向けプロジェクト格納フォルダ
    │ └ BLE_Modem
    │   ├ rBLE_Mdm
    │   │ ├ rBLE_Mdm.mtsp            モジュール FWROM 作成用 CS+ for CC サブプロジェクト
    │   │ └ sect_mdm_modfw.hsi       モジュール FW セクション情報ファイル
    │   └ BLE_Modem.mtpj             モジュール FW 向け CS+ for CC プロジェクト
    └ e2studio                       e² studio 向けプロジェクトファイル格納フォルダ
      └ BLE_Modem
        └ rBLE_Mdm
          ├ .cproject                モジュール FW ビルドオプション情報管理ファイル
          ├ .DefaultBuildlinker      モジュール FW リンカオプション情報管理ファイル
          ├ .info                    モジュール FW ツールチェイン情報管理ファイル
          ├ .project                 モジュール FW プロジェクト情報管理ファイル
          ├ rBLE_Mdm_CCRL.x.launch   モジュール FW デバッグ情報管理ファイル
          └ sect_mdm_modfw.esi       モジュール FW セクション情報ファイル
```

5.2.2　モジュールファームウェア V1.11 格納フォルダ

BLE ソフトウェア V1.21 への対応で更新したファイルのみを掲載します。

(1)　ルートフォルダ

```
Renesas_BLE_Module_V111
├ BLE_Sample                                  Host MCU プログラム格納フォルダ
├ RL78_G1D                                    モジュールファームウェア格納フォルダ
└ ROM_File                                    モジュールファームウェア HEX ファイル格納フォルダ
  └ rBLE_Mdm_CCRL_module_fw_v111.hex          モジュールファームウェア V1.11 HEX ファイル
```

(2)　Host MCU プログラム格納フォルダ

```
BLE_Sample
├ project
│ ├ CS_CCRL                                   CS+ for CC 用プロジェクト格納フォルダ
│ │ └ BLE_Sample
│ │   ├ rBLE_sample
│ │   │ └ rBLE_sample.mtsp                    CS+ for CC サブプロジェクト
│ │   └ BLE_Sample.mtpj                       CS+ for CC プロジェクト
│ ├ e2studio                                  e² studio 用プロジェクト格納フォルダ
│ │ └ BLE_Sample
│ │   └ rBLE_sample
│ │     ├ .cproject                           ビルドオプション情報管理ファイル
│ │     ├ .DefaultBuildlinker                 リンカオプション情報管理ファイル
│ │     ├ .info                               ツールチェイン情報管理ファイル
│ │     ├ .project                            プロジェクト情報管理ファイル
│ │     └ rBLE_sample_CCRL.x.launch           デバッグ情報管理ファイル
│ └ windows                                   PC 用プロジェクト格納フォルダ
│   ├ Exe
│   │ └ rBLE_Sample.exe                       PC サンプルプログラム実行ファイル
│   ├ rBLE_Sample.sln
│   ├ rBLE_Sample.vcxproj                     Microsoft Visual Studio Express 2015 向けプロジェクト
│   └ rBLE_Sample.vcxproj.filters
└ src
  ├ Platform                                  Platform 依存ソースコード格納フォルダ
  │ ├ G1D_cs_iar                              組み込み依存ソースコード格納フォルダ
  │ │ └ driver
  │ │   ├ plf
  │ │   │ └ plf.c                             RL78 プラットフォームサンプルドライバファイル
  │ │   └ serial
  │ │     ├ csi.c                             RL78 向け CSI サンプルドライバファイル
  │ │     ├ csi.h                             RL78 向け CSI サンプルドライバヘッダファイル
  │ │     └ uart.c                            RL78 向け UART サンプルドライバファイル
  │ └ win32_vc                                Windows 依存ソースコード格納フォルダ
  │   └ driver
  │     └ console
  │       ├ Console.c                         Windows 向けコンソール入力サンプルドライバファイル
  │       └ console.h                         Windows 向けコンソール入力サンプルドライバヘッダファイル
  └ rBLE                                      BLE サンプルプログラム格納フォルダ
    └ src
      ├ host
      │ └ rble_if_api_cb.c                    rBLE Host Event ファイル                    rBLE_Host
      ├ include
      │ ├ prf_sel.h                           プロファイル選択設定ヘッダファイル
      │ ├ rble.h                              rBLE ヘッダファイル                         rBLE_Host
      │ ├ rble_api.h                          rBLE API ヘッダファイル                     rBLE_Host
      │ └ rble_app.h                          サンプルプログラムヘッダファイル
```

```
    ├ sample_app                          サンプルアプリケーション格納フォルダ
    │  ├ r_vuart_app.c                    汎用双方向通信アプリケーションファイル
    │  ├ r_vuart_app.h                    汎用双方向通信アプリケーションヘッダファイル
    │  ├ r_vuart_app_param.c              汎用双方向通信アプリケーションパラメータファイル
    │  ├ r_vuart_console.c                汎用双方向通信コンソールドライバファイル
    │  ├ r_vuart_console.h                汎用双方向通信コンソールドライバヘッダファイル
    │  ├ rble_fw_up_sender_app.c          FW アップデート用サンプルプログラムファイル
    │  ├ rble_sample_app.c                サンプルプログラムファイル
    │  ├ rble_sample_anp.c                 〃
    │  ├ rble_sample_blp.c                 〃
    │  ├ rble_sample_cpp.c                 〃
    │  ├ rble_sample_cscp.c                〃
    │  ├ rble_sample_custom.c              〃
    │  ├ rble_sample_fmp.c                 〃
    │  ├ rble_sample_fwup.c                〃
    │  ├ rble_sample_gap_sm_gatt.c         〃
    │  ├ rble_sample_glp.c                 〃
    │  ├ rble_sample_hogp.c                〃
    │  ├ rble_sample_hrp.c                 〃
    │  ├ rble_sample_htp.c                 〃
    │  ├ rble_sample_lnp.c                 〃
    │  ├ rble_sample_pasp.c                〃
    │  ├ rble_sample_pxp.c                 〃
    │  ├ rble_sample_rscp.c                〃
    │  ├ rble_sample_scpp.c                〃
    │  ├ rble_sample_tip.c                 〃
    │  ├ rble_sample_vendor.c              〃
    │  └ rble_sample_vuart.c               〃
    └ sample_profile                      独自プロファイル格納フォルダ
       ├ scp                              Sample Custom Profile 格納フォルダ
       │  └ scpc.c                        Sample Custom Profile Client ファイル
       ├ vuart                            汎用双方向通信格納フォルダ
       │  ├ vuartc.c                      汎用双方向通信 Client ファイル
       │  ├ vuarts.c                      汎用双方向通信 Server ファイル
       │  └ vuarts.h                      汎用双方向通信 Server ヘッダファイル
       └ db_handle.h                      Attribute database handles ヘッダファイル
```

(3) モジュールファームウェア格納フォルダ

```
RL78_G1D
└ Project_Source
   ├ rBLE                                 rBLE 格納フォルダ
   │  └ src
   │     ├ include
   │     │  ├ rble.h                     rBLE ヘッダファイル
   │     │  ├ rble_api.h                 rBLE API ヘッダファイル
   │     │  └ rble_app.h                 サンプルプログラムヘッダファイル
   │     └ sample_app
   │        └ rble_fw_up_receiver_app.c  FW アップデート用サンプルプログラムファイル
   └ renesas
      ├ lib                               BLE ソフトウェアライブラリ格納フォルダ
      ├ src
      │  ├ arch
      │  │  └ rl78
      │  │     ├ arch_main.c              BLE ソフトウェアメインファイル
      │  │     ├ config.h                 BLE ソフトウェアコンフィギュレーションヘッダファイル
      │  │     ├ db_handle.h              Attribute database handles ヘッダファイル
      │  │     ├ ke_conf.c                RWKE タスク管理ファイル
      │  │     ├ main.c                   メイン関数ファイル
      │  │     ├ prf_config.c             プロファイル向けパラメータ設定ファイル
```

```
│ │     ├ prf_config.h          プロファイル向けパラメータ設定ヘッダファイル
│ │     ├ prf_sel.h             プロファイル選択設定ヘッダファイル
│ │     └ rwble_mem.c           可変メモリ確保設定ファイル
│ ├ compiler
│ │ └ compiler.h                CC-RL 向けデバイスヘッダファイル
│ └ driver
│   ├ dataflash
│   │ ├ dataflash.c             データ･フラッシュドライバファイル
│   │ ├ dataflash.h             データ･フラッシュドライバヘッダファイル
│   │ └ fdl_descriptor_t02.h    EEL Pack2 向けフラッシュ記述子ヘッダファイル
│   ├ DTM2Wire
│   │ └ DTM2Wire.c              2-Wire UART Direct Test Modeドライバファイル
│   ├ module
│   │ └ module.c                ファームウェアファイル
│   ├ plf
│   │ └ plf.c                   RL78 プラットフォームドライバファイル
│   ├ push_sw
│   │ └ push_sw.c               プッシュスイッチドライバファイル
│   ├ uart
│   │ └ uart.c                  UARTドライバファイル
│   └ wakeup
│     └ wakeup.c                ウェイクアップドライバファイル
└ tools
  └ project
    ├ CS_CCRL                   CS+ for CC 向けプロジェクト格納フォルダ
    │ └ BLE_Modem
    │   ├ rBLE_Mdm
    │   │ └ rBLE_Mdm.mtsp       モジュール FWROM 作成用 CS+ for CC サブプロジェクト
    │   └ BLE_Modem.mtpj        モジュール FW 向け CS+ for CC プロジェクト
    └ e2studio                  e² studio 向けプロジェクトファイル格納フォルダ
      └ BLE_Modem
        └ rBLE_Mdm
          ├ .info               モジュール FW ツールチェイン情報管理ファイル
          └.project             モジュール FW プロジェクト情報管理ファイル
```

5.3 BLE ソフトウェア V1.21 対応

BLE ソフトウェア V1.21 に対応したモジュールファームウェアと HostMCU サンプルプログラムのアップデート内容を示します。併せて下記のドキュメントも参照してください。

- Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル (R01UW0095)
- API リファレンスマニュアル 基本編 (R01UW0088)
- BLE 仮想 UART アプリケーション (R01AN3130)

5.3.1 モジュールファームウェア

BLE ソフトウェア V1.21:

- 自デバイスの BD アドレスとして RPA(Resolvable Private Address)を使用してスマートフォンとペアリングした場合、ペアリングやペアリング後の再接続に失敗する問題を修正。
- CPU のクロック供給に外部クロックを使用した場合、データ・フラッシュへのアクセスができない問題を修正。
- 特定の動作設定条件において Slave デバイスとして動作した場合、対向デバイスとの通信が不安定となる問題を修正。
  - メイン・システム・クロック 4MHz を選択、かつ RF スロー・クロック用オンチップ・オシレータ(32.768kHz)を選択、かつ最大同時接続台数(CFG_CON)を 1 台に設定。
  - メイン・システム・クロック 16MHz を選択、かつ CC-RL コンパイラを使用。
- RL78/G1D のテクニカルアップデート（TN-RL*-A083A）に従い、内部未使用端子への設定を変更しました。詳細はテクニカルアップデートを参照してください。

BLE ソフトウェア V1.20:

- 暗号化開始時に通知されるイベントを変更し、新たなイベント(RBLE_SM_LTK_REQ_FOR_ENC_IND)を追加。
- Modem 構成の WAKEUP 信号を使用するシリアル通信方式の場合に、WAKEUP 要求後に受信データを取りこぼす可能性がある問題を修正。
- Master で動作時に、2 台目以降の Slave から Service Discovery 出来ない問題を修正。
- モジュールファームウェアのバージョンを V1.11 に変更。

5.3.2 HostMCU サンプルプログラム

BLE ソフトウェア V1.21:

- 汎用双方向通信(仮想 UART)の Server role で、RBLE_VUART_Server_Send_Indication の第 2 引数に 20 バイトの指定ができるように修正。

BLE ソフトウェア V1.20:

- 最大 Long 特性値サイズ(RBLE_GATT_MAX_LONG_VALUE)の定義値を 80 から 72 に修正。
- 汎用双方向通信(仮想 UART)の簡易 AT コマンドモードに、AT-CI=<con_intv>、AT-CI?、ATE0、ATE1 のコマンドを追加。

5.3.3　ビルド環境

e2 studio のバージョンを更新しました。

- e2 studio V5.4.0.015/RL78 コンパイラ CC-RL V1.03.00

Windows 用 rBLE_sample のビルド環境を更新しました。

- Microsoft Visual Studio Express 2015 for Windows Desktop.

5.3.4　ビルド方法

次のようにフォルダ名を読み替えてビルドを行ってください。

BLE_Software_Ver_1_11　→ BLE_Software_Ver_1_21

Renesas_BLE_Module_V101　→ Renesas_BLE_Module_V111

(1)　モジュールファームウェア

「2.3 ビルド準備」を参考にライブラリファイルを各コピー先フォルダにコピーし、「4.6.1 ファームウェアのビルド」を参考にビルドしてください。(Bluetooth Low Energy プロトコルスタックのライブラリファイル名(*_CCRL.lib)は V1.11 と同じ名前です。)

(2)　Host MCU プログラム

「4.6 ビルド方法」を参考にビルドしてください。

5.4 参考文献

1. Bluetooth Core Specification v4.2, Bluetooth SIG
2. Find Me Profile Specification v1.0, Bluetooth SIG
3. Immediate Alert Service Specification v1.0, Bluetooth SIG
4. Proximity Profile Specification v1.0, Bluetooth SIG
5. Link Loss Service Specification v1.0, Bluetooth SIG
6. Tx Power Service Specification v1.0, Bluetooth SIG
7. Health Thermometer Profile Specification v1.0, Bluetooth SIG
8. Health Thermometer Service Specification v1.0, Bluetooth SIG
9. Device Information Service Specification v1.1, Bluetooth SIG
10. Blood Pressure Profile Specification v1.0, Bluetooth SIG
11. Blood Pressure Service Specification v1.0, Bluetooth SIG
12. Heart Rate Profile Specification v1.0, Bluetooth SIG
13. Heart Rate Service Specification v1.0, Bluetooth SIG
14. Glucose Profile Specification v1.0, Bluetooth SIG
15. Glucose Service Specification v1.0, Bluetooth SIG
16. Time Profile Specification v1.0, Bluetooth SIG
17. Current Time Service Specification v1.0, Bluetooth SIG
18. Next DST Change Service Specification v1.0, Bluetooth SIG
19. Reference Time Update State Service Specification v1.0, Bluetooth SIG
20. Alert Notification Service Specification v1.0, Bluetooth SIG
21. Alert Notification Profile Specification v1.0, Bluetooth SIG
22. Phone Alert Status Service Specification v1.0, Bluetooth SIG
23. Phone Alert Status Profile Specification v1.0, Bluetooth SIG
24. Bluetooth SIG Assigned Numbers https://www.bluetooth.com/specifications/assigned-numbers
25. Personal Health Devices Transcoding White Paper v1.4, Bluetooth SIG

5.5 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。
クライアント特性コンフィギュレーション記述子	Client Characteristic Configuration Descriptor	クライアント特性コンフィギュレーション記述子を持つ特性値の GATT サーバからの送信(Notification / Indication)を制御するために使用します。
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。

5.6　略語および略称の説明

略語／略称	フルスペル	備考
ANP	Alert Notification Profile	
ANS	Alert Notification Service	
API	Application Programming Interface	
ATT	Attribute Protocol	
BB	Base Band	
BLE	Bluetooth low energy	
BLP	Blood Pressure Profile	
BLS	Blood Pressure Service	
CTS	Current Time Service	
DIS	Device Information Service	
FMP	Find Me Profile	
GAP	Generic Access Profile	
GATT	Generic Attribute Profile	
GLP	Glucose Profile	
GLS	Glucose Service	
HCI	Host Controller Interface	
HRP	Heart Rate Profile	
HRS	Heart Rate Service	
HTP	Health Thermometer Profile	
HTS	Health Thermometer Service	
IAS	Immediate Alert Service	
L2CAP	Logical Link Control and Adaptation Protocol	
LE	Low Energy	
LL	Link Layer	
LLS	Link Loss Service	
NDCS	Next DST Change Service	
MCU	Micro Controller Unit	
OS	Operating System	
PASP	Phone Alert Status Profile	
PASS	Phone Alert Status Service	
PXP	Proximity Profile	
RF	Radio Frequency	
RSCP	Running Speed and Cadence Profile	
RSCS	Running Speed and Cadence Service	
RSSI	Received Signal Strength Indication	
RTUD	Reference Time Update Service	
SM	Security Manager	
SMP	Security Manager Protocol	
TIP	Time Pforile	

TPS	Tx Power Service	
UART	Universal Asynchronous Receiver Transmitter	
UUID	Universal Unique Identifier	

略語／略称	フルスペル	備考
RSCIP	Renesas Serial Communication Interface Protocol	
RWKE	Renesas Wireless Kernel Extension	
VS	Vendor Specific	

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/

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
1.00	2016.07.08	－	初版発行
1.10	2016.10.21	－ － 57 60	BLE ソフトウェア V1.20 に対応したモジュールファームウェアと Host MCU プログラムのソースコードを、アプリケーションノートに同梱。 サンプルプログラムのフォルダを以下のように変更。 ・Renesas_BLE_Module_V101 ・Renesas_BLE_Module_V110 モジュールファームウェア V1.10 のフォルダ構成を追記。 BLE ソフトウェア V1.20 に対応した更新内容を追記。
1.11	2017.11.1	－ － － 57 60	BLE ソフトウェア V1.21 に対応したモジュールファームウェアと Host MCU プログラムのソースコードを、アプリケーションノートに同梱。 サンプルプログラムのフォルダを以下のように変更。 ・Renesas_BLE_Module_V111 BLE ソフトウェア V1.20 に対応したモジュールファームウェアと Host MCU プログラムのソースコードを削除。 モジュールファームウェア V1.11 のフォルダ構成を記載。 BLE ソフトウェア V1.21 に対応した更新内容を記載。
1.11	2022.01.31	－	BLE ソフトウェアで Bluetooth SIG で規定された Profile のサポートが終了したことに伴う修正。

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：http://japan.renesas.com/contact/

© 2016 Renesas Electronics Corporation. All rights reserved.
Colophon 4.0