User's Manual

Bluetooth® Low Energy プロトコルスタック

ユーザーズマニュアル

ルネサスマイクロコンピュータ

対象デバイス

RL78/G1D

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.20　2022.01

ご注意書き

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

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

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

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

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

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

(Rev.4.0-1　2017.11)

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。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、ルネサス Bluetooth Low Energy マイコン（RL78/G1D）を使用した応用製品の開発に利用するソフトウェア「Bluetooth Low Energy プロトコルスタック」のセットアップ方法・構成・機能について説明するものです。本ソフトウェアを用いた応用システムを設計するユーザを対象にしています。このマニュアルを使用するには、マイクロコンピュータ、Bluetooth Low Energy に関する基本的な知識が必要です。

関連資料

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

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	このマニュアル	R01UW0095E
	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　HOGP 編 (廃止)	R01UW0093J	R01UW0093E
	API リファレンスマニュアル　ScPP 編 (廃止)	R01UW0094J	R01UW0094E
	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　CSCP 編 (廃止)	R01UW0098J	R01UW0099E
	API リファレンスマニュアル　CPP 編 (廃止)	R01UW0099J	R01UW0099E
	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
	API リファレンスマニュアル　LNP 編 (廃止)	R01UW0113J	R01UW0113E
	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

2. 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	
BAS	Battery Service	
BB	Base Band	
BD_ADDR	Bluetooth Device Address	
BLE	Bluetooth low energy	
BLP	Blood Pressure Profile	
BLS	Blood Pressure Service	
CPP	Cycling Power Profile	
CPS	Cycling Power Service	
CSCP	Cycling Speed and Cadence Profile	
CSCS	Cycling Speed and Cadence Service	
CSRK	Connection Signature Resolving Key	
CTS	Current Time Service	
DIS	Device Information Service	
EDIV	Encrypted Diversifier	
FMP	Find Me Profile	
GAP	Generic Access Profile	
GATT	Generic Attribute Profile	
GLP	Glucose Profile	
GLS	Glucose Service	
HCI	Host Controller Interface	
HID	Human Interface Device	
HIDS	HID Service	
HOGP	HID over GATT Profile	
HRP	Heart Rate Profile	
HRS	Heart Rate Service	
HTP	Health Thermometer Profile	
HTS	Health Thermometer Service	
IAS	Immediate Alert Service	
IRK	Identity Resolving Key	
L2CAP	Logical Link Control and Adaptation Protocol	
LE	Low Energy	
LL	Link Layer	
LLS	Link Loss Service	

LNP	Location and Navigation Profile	
LNS	Location and Navigation Service	
LTK	Long Term Key	
MCU	Micro Controller Unit	
MITM	Man-in-the-middle	
MTU	Maximum Transmission Unit	
OOB	Out of Band	
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	
ScPP	Scan Parameters Profile	
ScPS	Scan Parameters Service	
SM	Security Manager	
SMP	Security Manager Protocol	
STK	Short Term Key	
TK	Temporary Key	
TPS	Tx Power Service	
UART	Universal Asynchronous Receiver Transmitter	
UUID	Universal Unique Identifier	

略語／略称	フルスペル	備考
APP	Application	
CSI	Clocked Serial Interface	
IIC	Inter-Integrated Circuit	
RSCIP	Renesas Serial Communication Interface Protocol	
VS	Vendor Specific	

本製品は外国為替及び外国貿易法の規定により規制貨物等に該当しますので、日本国外に輸出する場合には、同法に基づき日本国政府の輸出許可が必要です。

目次

1. はじめに

このマニュアルは、ルネサス Bluetooth Low Energy マイコン（RL78/G1D）を使用した Bluetooth 応用製品の開発に利用するソフトウェア「Bluetooth Low Energy プロトコルスタック」(以降、BLE ソフトウェア)のセットアップ方法・構成・機能について説明するものです。BLE ソフトウェアの API 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を参照してください。

2. 適用

このマニュアルは Bluetooth Low Energy プロトコルスタック Version 1.20 以降に適用します。

3. 制限事項

この章では、BLE ソフトウェアの制限事項について記載します。

4. BLE ソフトウェアのインストール

4.1 内容物

BLE ソフトウェアの zip 圧縮パッケージには以下に示すものが含まれます。

- ドキュメント
  - Bluetooth Low Energy プロトコルスタック ユーザーズマニュアル（本書）
  - Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル
  - Bluetooth Low Energy プロトコルスタック サンプルプログラム アプリケーションノート
  - rBLE コマンド仕様書
- 実行ファイル作成用プロジェクト一式
  - 実行ファイル
  - BLE ソフトウェアライブラリ
  - サンプルソースコード
  - 各種パラメータ設定用ソースコード
  - e^2 studio 用プロジェクトファイル
  - CS+ for CC 用プロジェクトファイル
  - CS+ for CA,CX 用プロジェクトファイル
- PC 用サンプルアプリケーション一式
  - 実行ファイル
  - ソースコード
  - Microsoft Visual Studio Express 2013 用プロジェクトファイル
- HCI パケットモニタ PC 用アプリケーション一式
  - 実行ファイル
  - INI ファイル

4.2 BLE ソフトウェアのビルド環境

BLE ソフトウェアのビルド環境を下記に示します。

- 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）
- 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
  - Windows 7
  - Microsoft Visual Studio Express 2013 for Windows Desktop Update4
  - Microsoft .NET Framework 4＋言語パック

    Renesas CS+ for CC V4.00.00/RL78 コンパイラ CC-RL V1.03.00
    または e² studio 4.3.1.001/RL78 コンパイラ CC-RL V1.03.00
    または Renesas CS+ for CA,CX V3.02.00/Renesas CA78K0R V1.72
  - Renesas Flash Programmer v3.01.00
    (https://www.renesas.com/ja-jp/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html より入手可能)

PC 用サンプルアプリケーションの実行環境につきましては、Bluetooth Low Energy プロトコルスタック サンプルプログラム アプリケーションノートを参照ください。

4.3　インストール手順

パッケージを解凍して任意のフォルダにコピーしてください。
また、ご使用になられる開発環境に対応した Renesas 製 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、それぞれ 4.4.2 (7)、4.4.2 (8)を参照し、Renesas の web サイトより入手した後、以下のフォルダにコピーしてください。

【注】e² studio をご使用になる場合はフォルダパスにマルチバイト文字(全角文字)およびブランクを含まない場所にコピーしてください。

- EEPROM エミュレーションライブラリ
  - CS+ for CC/e² studio (CC-RL)
    ¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥dataflash¥cc_rl
  - CS+ for CA,CX
    ¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥dataflash¥cs

- コードフラッシュライブラリ
  - CS+ for CC/e² studio (CC-RL)
    ¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥codeflash¥cc_rl
  - CS+ for CA,CX
    ¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥codeflash¥cs

4.4　ファイル・フォルダ構成

インストール後のファイル・フォルダについての詳細を以降に記載します。

4.4.1　¥Renesas¥BLE_Software_Ver_X_XX¥Manual¥

このフォルダには、各種マニュアルが格納されています。ご一読ください。

4.4.2　¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D

(1)　¥ROM_File¥

このフォルダには、BLE ソフトウェアのうち RL78/G1D 上で動作するプログラムの実行ファイル(Hex ファイル)が格納されています。RL78/G1D の内蔵 Flash への書き込みを行ってください。

内蔵 Flash への書き込み方法は、Renesas Flash Programmer フラッシュ書き込みソフトウェア ユーザーズマニュアルを参照してください。

また、データ・フラッシュ領域に BD アドレスが書き込み済みの場合、フラッシュ書き込みソフトウェアの設定情報一覧の動作モードを「ブロック（コード・フラッシュ）」に変更してください。変更しない場合、書き込み済みの BD アドレスが消去されてしまう場合があります。

このフォルダに格納されている実行ファイルの内容について、表 4-1 に示します。各実行ファイルの作成方法については 6 章「実行ファイル作成方法」、Sample Custom プロファイルについては Bluetooth Low Energy プロトコルスタック サンプルプログラム アプリケーションノートを参照ください。

表 4-1 実行ファイルの概要

格納フォルダ	実行ファイル名	概要
¥ca78k0r		CS+ for CA,CX 版格納フォルダ
¥Embedded		Embedded 構成版格納フォルダ
	RL78_G1D_CE(SCP).hex	Sample Custom プロファイルに対応した実行ファイル
	RL78_G1D_CE(SIMPLE_SAMPLE).hex	簡易サンプルプログラムに対応した実行ファイル
¥Modem		Modem 構成版格納フォルダ
	RL78_G1D_CM(SCP).hex	Sample Custom プロファイルに対応した実行ファイル
	RL78_G1D_CM(DTM_2WIRE).hex	2-Wire UART Direct Test Mode に対応した実行ファイル
¥ccrl		CC-RL 版格納フォルダ
¥Embedded		Embedded 構成版格納フォルダ
	RL78_G1D_CCE(SCP).hex	Sample Custom プロファイルに対応した実行ファイル
	RL78_G1D_CCE(SIMPLE_SAMPLE).hex	簡易サンプルプログラムに対応した実行ファイル
¥Modem		Modem 構成版格納フォルダ
	RL78_G1D_CCM(SCP).hex	Sample Custom プロファイルに対応した実行ファイル

(2) ¥Project_Source¥

このフォルダには RL78/G1D 上で動作するプログラムの実行ファイル(Hex ファイル)をビルドするために必要となる BLE ソフトウェアライブラリおよびサンプルソースコードが格納されています。

(3) ¥Project_Source¥renesas¥tools¥project¥

このフォルダには RL78/G1D 上で動作するプログラムの実行ファイル(Hex ファイル)をビルドするための各開発環境向けのプロジェクト／ワークスペースファイルが格納されています。また、それぞれの開発環境ごとに、Embedded 構成と Modem 構成のプロジェクト／ワークスペースが格納されています。ご使用になられる開発環境にてビルドし、実行ファイルを生成してください。

ビルド方法につきましては 6.2 を参照ください。

(4) ¥Project_Source¥renesas¥tools¥project_devices¥

このフォルダには、R5F11AGG(128KB)および R5F11AGH(192KB)用の実行ファイル(Hex ファイル)をビルドするための各開発環境向けのプロジェクト／ワークスペースファイルが格納されています。また、それぞれの開発環境ごとに、Embedded 構成と Modem 構成のプロジェクト／ワークスペースが格納されています。ご使用になられる開発環境にてビルドし、実行ファイルを生成してください。

(5) ¥Project_Source¥renesas¥tools¥project_simple¥

このフォルダには、BLE ソフトウェアの使用方法を示す簡易サンプルプログラムの実行ファイル(Hex ファイル)をビルドするための各開発環境向けのプロジェクト／ワークスペースファイルが格納されています。ご使用になられる開発環境にてビルドし、実行ファイルを生成してください。

簡易サンプルプログラムにつきましてはサンプルプログラムアプリケーションノート(R01AN1375)を参照ください。

(6) ¥Project_Source¥renesas¥src¥

このフォルダには、ユーザより変更可能な各種パラメータ設定ファイル(ソースコード)が格納されています。必要に応じてパラメータを変更し、ビルドを行ってください。

変更可能な各種パラメータおよびその変更方法につきましては 6.1 を参照ください。

(7)　¥Project_Source¥renesas¥src¥driver¥dataflash¥cc_rl または ¥cs

このフォルダには、ご使用になられる開発環境にあった Renesas 製 EEPROM エミュレーションライブラリをコピーしてください。本ライブラリは、Renesas の web サイトより入手してください。参考に、BLE ソフトウェアで動作確認を行ったバージョンの入手方法を記載します。なお、web サイトのリニューアルなどにより、予告なく操作手順が変更になる場合があります。

Renesas の web サイト https://www.renesas.com/ より、「ホーム」→「製品情報」→「開発ツール」→「セルフプログラミングライブラリ」→「データフラッシュライブラリ」→「ダウンロード」の順に進め、各開発環境向けのライブラリをダウンロードしてください。

本フォルダにコピーするファイルは以下の通りです。

【注】BLE プロトコルスタック V1.20 より、EEPROM エミュレーションライブラリ Pack02 に統一しました。

CS+ for CC/e^2 studio (CC-RL)版：

「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」

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

CS+ for CA,CX 版：

「RL78 ファミリ CA78K0R コンパイラ用 EEPROM エミュレーションライブラリ Pack02 Ver.1.01」

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

(8)　¥Project_Source¥renesas¥src¥driver¥codeflash¥cc_rl または ¥cs

このフォルダには、ご使用になられる開発環境にあった Renesas 製コードフラッシュライブラリをコピーしてください。本ライブラリは、Renesas の web サイトより入手してください。参考に、BLE ソフトウェアで動作確認を行ったバージョンの入手方法を記載します。なお、web サイトのリニューアルなどにより、予告なく操作手順が変更になる場合があります。

Renesas の web サイト https://www.renesas.com/ より、「ホーム」→「製品情報」「開発ツール」→「セルフプログラミングライブラリ」→「コードフラッシュライブラリ」→「ダウンロード」の順に進め、各開発環境向けのライブラリをダウンロードしてください。

本フォルダにコピーするファイルは以下の通りです。

CS+ for CC/e^2 studio (CC-RL)版：

「RL78 ファミリ CC-RL コンパイラ用フラッシュセルフプログラミングライブラリ Type01 Ver.2.21」

- fsl.h
- fsl.lib
- fsl_types.h

CS+ for CA,CX 版：

「RL78 ファミリ CA78K0R コンパイラ用フラッシュセルフプログラミングライブラリ Type01 Ver.2.20」

- fsl.h
- fsl.lib
- fsl_types.h

4.4.3　¥Renesas¥BLE_Software_Ver_X_XX¥BLE_Sample¥

このフォルダには、Modem 構成で使用する際の PC 上で動作する BLE ソフトウェアのサンプルプログラムが格納されています。サンプルプログラムの詳細につきましては Bluetooth Low Energy プロトコルスタック サンプルプログラム アプリケーションノートを参照ください。

5. BLE ソフトウェアの構成

BLE ソフトウェアは Bluetooth Low Energy (Bluetooth v4.2)に対応した BLE スタックを含むソフトウェア一式です。次節より BLE ソフトウェアの具体的な構成を説明します。

5.1 構成

図 5-1 に BLE ソフトウェアの構成図を示します。

BLE ソフトウェアは、アプリケーションが RL78/G1D に搭載される構成(以降、Embedded 構成)と、別の MCU に搭載される構成(以降、Modem 構成)で動作し、両構成で、同じアプリケーションを使用することが可能な API を提供します。

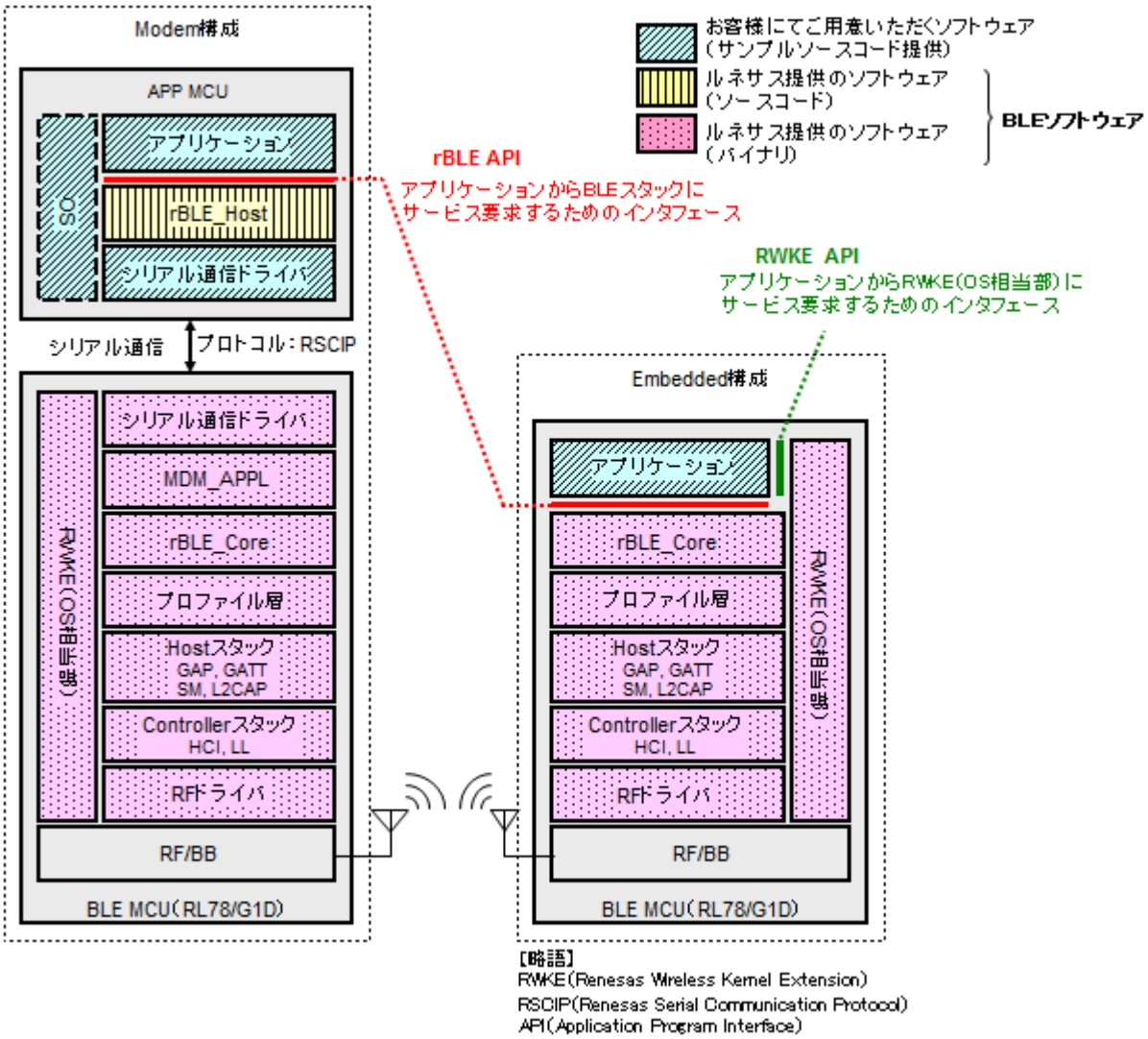


図 5-1 BLE ソフトウェアの構成

Modem 構成の BLE ソフトウェアは、APP MCU と BLE MCU(RL78/G1D)の 2 つのチップで動作し、APP MCU で動作する「rBLE_Host」部（図の▥）と BLE MCU で動作するソフトウェア（図の▩）で構成されます。

また、お客さまにてご用意いただくソフトウェア（図の▨）は、APP MCU の「アプリケーション」部と「シリアル通信ドライバ」部、および「OS」部になります。ただし、「rBLE_Host」部は OS のリソースを使

用していないため、APP MCU に OS が搭載されていない場合には、「OS」部のソフトウェアを用意する必要はありません。

APP MCU で動作するアプリケーションは、rBLE_Host を介して BLE MCU と BLE サービスのやり取りが行われます。APP MCU と BLE MCU は物理的に UART、CSI、IIC いずれかのシリアル通信で接続され、rBLE_Host の制御により RSCIP (Renesas Serial Communication Interface Protocol)を使用した通信が行われます。

一方、Embedded 構成時の BLE ソフトウェアは、BLE MCU(RL78/G1D)のみの 1 チップで動作します。お客様にてご用意いただくソフトウェアは、「アプリケーション」部のみとなり、BLE MCU 上に実装されます。

各ソフトウェアブロックの概要を表 5-1 に示します。

表 5-1　ソフトウェアブロックの概要

ブロック名称	概要
アプリケーション	お客様にご用意いただくアプリケーション
OS	お客様にご用意いただく OS
rBLE_Host	MDM_APPL へのコマンドパケットの構築および MDM_APPL からのイベントパケットの解析を行い、アプリケーションから rBLE API 発行を可能にする。
シリアル通信ドライバ(APP MCU)	UART、CSI、IIC いずれかにて BLE MCU と通信を行う。通信プロトコルには RSCIP を使用する。 【注】本ドライバはお客様にご用意いただく必要があります。
シリアル通信ドライバ(BLE MCU)	UART、CSI、IIC いずれかにて APP MCU と通信を行う。通信プロトコルには RSCIP を使用する。
MDM APPL	rBLE_Host からのコマンドパケットの解析および rBLE_Host への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する
rBLE_Core	上位モジュールに対して、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する。
プロファイル層	BLE スタック本体 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)	他のモジュールから共通して利用される基本的な機能を提供し、BLE MCU 全体を管理する。

5.2　rBLE API

BLE ソフトウェアは BLE MCU 上の BLE スタックのサービスをアプリケーションから利用可能とするため、アプリケーションに対してサービス利用のための API(rBLE API と呼称)を提供します。

rBLE API の提供する API にてアクセス可能な Bluetooth レイヤを下図に示します。

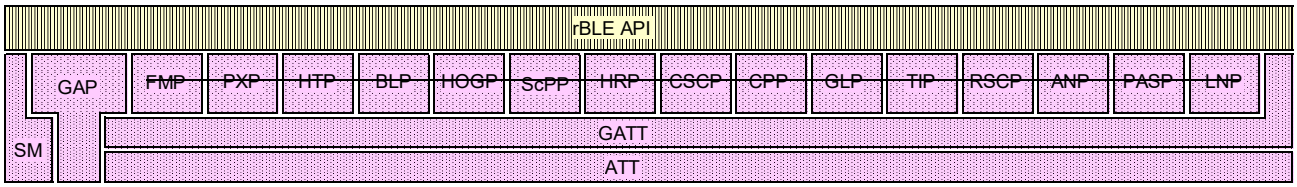


図 5-2　rBLE API と BLE スタック

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、BLE ソフトウェアがサポートするプロファイル(注) 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。

注：FMP(Find Me), PXP(Proximity), HTP(Health Thermometer), BLP(Blood Pressure), HOGP(HID Over GATT), ScPP(Scan Parameters), HRP(Heart Rate), CSCP(Cycling Speed and Cadence), CPP(Cycling Power), GLP(Glucose), TIP (Time Profile), RSCP(Running Speed and Cadence), ANP(Alert Notification), PASP(Phone Alert Status), LNP(Location and Navigation)

各レイヤのサポートする機能概要を表 5-2 に示します。各機能の詳細につきましては 7 章「機能説明」を参照ください。

表 5-2　rBLE API Bluetooth サポート機能

レイヤ	レイヤ概要	サポート機能概要
GAP (Generic Access Profile)	周辺デバイスの検索や、ピアデバイスとの接続・切断等のリンク管理、セキュリティ要件に応じた各種手続きを行う。	• 4 つの GAP ロール (Central, Peripheral, Broadcaster, Observer) • Broadcast および Scan • Discovery, Connection, Bonding モードおよびプロシージャ • セキュリティモード • リンクの確立、切断 • 接続パラメータの変更 • ランダムおよびスタティックアドレス • プライバシーフィーチャー
SM (Security Manager)	2 デバイス間のセキュリティを確保するためにペアリングを行い、通信内容の暗号化またはデータ署名を行う。またそれらに必要となるデバイス同士の情報交換を行う。	• ペアリング手続き • 各種ペアリングアルゴリズム (Passkey Entry, Just Works, OOB) • ペアリングおよびキー生成 • キー配布 • 認証、暗号化、データ署名によるセキュリティ

また、rBLE API には BLE MCU の RF 評価を行うための Direct Test Mode 用のインタフェースがあり、送信テストおよび受信テストを実施することが可能です。

5.3　BLE ソフトウェアで使用する機能

BLE ソフトウェアが使用する RL78/G1D の機能は以下の通りです。使用している機能は、ユーザアプリ等では使用しないでください。

表 5-3　BLE ソフトウェアで使用する機能

機能		Modem 構成	Embedded 構成	用途
データ・フラッシュ		使用	使用	BD アドレスを格納
12bit インターバルタイマ		使用	使用	消費電流ピーク通知機能または RF 用スロー・クロック部内蔵発振回路の監視にて使用(*1)
タイマ・アレイ・ユニット		チャネル 1	チャネル 1	外部 32.768kHz 発振子を使用する場合に plf_init 関数で使用
		チャネル 7	—	CSI または IIC ドライバを使用する設定の場合に使用
クロック出力／ブザー出力		PCLBUZ0	PCLBUZ0	RF 用スロー・クロック部内蔵発振回路を使用しない設定の場合に使用 (*2)
ポート機能		P23(出力) P30(入力)	—	P23：CSI または IIC 通信用出力端子 P30：UART または CSI 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 UART1	—	UART0 または UART1：シリアル通信用
		CSI00 CSI20 CSI21	 CSI21	CSI00 または CSI20：シリアル通信用 CSI21：MCU-RF 接続用
シリアル・インタフェース IICA		IICA0	—	IICA0：シリアル通信用
乗除積和算器		使用	使用	CS+ for CA,CX(CA78K0R)、e^2 studio/CS+ for CC(CC-RL)全てにおいて使用する設定でビルドしてください。
DAM コントローラ		DMA0、DMA1 DMA2、DMA3	 DMA2、DMA3	DMA0、DMA1：UART、CSI シリアル通信用(*3) DMA2、DMA3：MCU-RF 接続用
割り込み	外部端子	INTRF INTP3	INTRF	INTRF：RF 部から割り込み INTP3：外部起床用トリガ
	DMA	INTDMA0 INTDMA1 INTDMA2 INTDMA3	INTDMA2 INTDMA3	INTDMA0、INTDMA1：シリアル通信用 INTDAM2、INTDMA3：MCU-RF 接続用
	シリアル・アレイ・ユニット	INTCSImn INTSTm INTSRm INTSREm	—	INTCSImn：CSI 接続用(mn=00, 20) INTSTn、INTSRn、INTSREn：UART 接続用(m=0,1)
	シリアル・インタフェース IICA	INTIICA0	—	INTIICA0：IIC 接続用

	12bit インターバルタイマ	INTIT	INTIT	消費電流ピーク通知機能または RF 用スロー・クロック部内蔵発振回路の監視にて使用(*1)

【注】*1　12bit インターバルタイマ: 消費電流ピーク通知機能を使用しない場合、ユーザでの使用可能です。RF 用スロー・クロック部内蔵発振回路を使用する場合、ユーザでの使用はできません。

*2　クロック出力／ブザー出力: RF 用スロー・クロック部内蔵発振回路を使用する場合、ユーザでの使用が可能です。

*3　Modem 構成時に UART0、UART1、CSI00、CSI20 を選択した場合、ユーザでの使用はできません。

5.4　Modem 構成時のシリアル通信

APP MCU と BLE MCU は UART、CSI、IIC いずれかを介してシリアル通信を行い、通信プロトコルとして Renesas Serial Communication Interface Protocol (RSCIP)を使用します。RSCIP は、RFC 1055 にて規定される SLIP(Serial Line Internet Protocol)をベースに拡張しています。 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、通信データの信頼性を確保します。

APP MCU における RSCIP 機能およびシリアル通信ドライバの制御機能は、rBLE_Host の 1 機能である RSCIP ドライバが提供します。RSCIP ドライバは、RSCIP パケットをデータの転送単位として通信を行います。RSCIP の詳細につきましては rBLE コマンド仕様書をご参照ください。

シリアル通信は UART、CSI、IIC による通信機能を提供し、下記の接続方式があります。
なお、CSI20 では選択できない接続方式があります。

表 5-4　シリアル通信の接続方式

シリアル	接続方式	対応チャネル
UART	2 線接続方式	UART0, UART1※
	3 線接続方式	UART0, UART1
	2 線分岐接続方式	UART0, UART1
CSI	4 線接続方式	CSI00
	5 線接続方式	CSI00, CSI20
IIC	3 線接続方式	IICA0

※ UART1 は SNOOZE モードに対応していないため、MCU の STOP モード中に RxD 入力による復帰は行えないことにご注意ください。

UART による通信は以下の設定値で動作します。

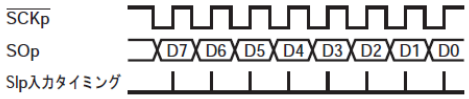
表 5-5　UART 設定値

設定	設定値
ボー・レート	（2 線式）4800bps～250kbps※ （3 線式、2 線分岐式）4800bps～250kbps
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

※ 2 線接続方式を選択し 4800bps より大きいボー・レートを設定した場合、BLE MCU の低消費電流を実現するための機能である Sleep 機能が無効となります。その他の接続方式では常に Sleep 機能は有効となります。

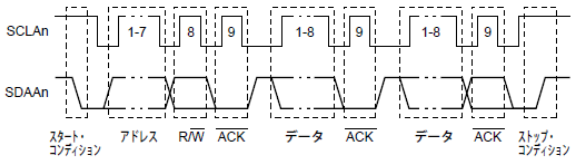
CSI による通信は以下の設定値で動作します。

表 5-6　CSI 設定値

設定	設定値
通信方式	ピアツーピアのマスタスレーブ方式 半二重のクロック同期式、マスタクロック供給
通信ロール	APP MCU：マスタ BLE MCU：スレーブ
ボー・レート	4800bps～250kbps（4 線式、5 線式）
データ長	8bit
データクロック位相	RL78/G1D シリアル・アレイ・ユニット位相モードタイプ 1 

IIC による通信は以下の設定値で動作します。

表 5-7　IIC 設定値

設定	設定値
通信方式	IIC バス・モード 半二重のクロック同期式、マスタクロック供給
通信ロール	APP MCU：マスタ BLE MCU：スレーブ
転送クロック	100kbps～400kbps
データ長	8bit
通信フォーマット	IIC シリアル・データ転送フォーマット 

【注】APP MCU 上に搭載するシリアル通信ドライバはお客様にご用意いただく必要があります。シリアル通信ドライバの要件につきましては Bluetooth Low Energy プロトコルスタック サンプルプログラム アプリケーションノートを参照ください。

5.4.1　UART 2 線接続方式

本接続方式では、APP MCU と BLE MCU は下記に示すように UART のデータ信号線である TxD、RxD を使用して通信します。また 4800bps より大きいボー・レートを設定した場合、BLE MCU の低消費電流を実現するための機能である Sleep 機能が無効となります。

APP MCU から BLE MCU への送信動作、および BLE MCU から APP MCU への送信動作とハンドシェイク動作はありません。全二重通信により任意のタイミングで RSCIP パケットを送受信します。

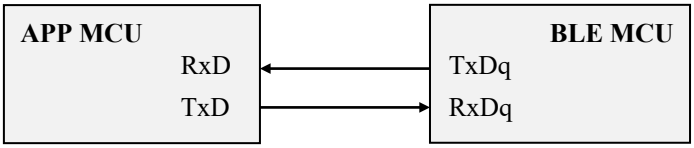


図 5-3　UART　2 線接続方式

BLE MCU 端子名称	方向	機能
TxDq (q=0,1)	BLE→APP	シリアル出力データ信号
RxDq (q=0,1)	APP→BLE	シリアル入力データ信号

※以降に記載するタイミングチャートでは、BLE MCU側の端子名称を記載します。

(1)　APP MCU の送信動作

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた送信シーケンスを示します。

[送信開始時]：rBLE_Host が送信関数を呼び出すことにより、シリアル通信ドライバは RSCIP パケットの送信動作を開始します。

[送信終了時]：RSCIP パケット送信の完了時、シリアル通信ドライバは送信完了通知関数を呼び出すことで、送信完了を rBLE_Host に通知します。

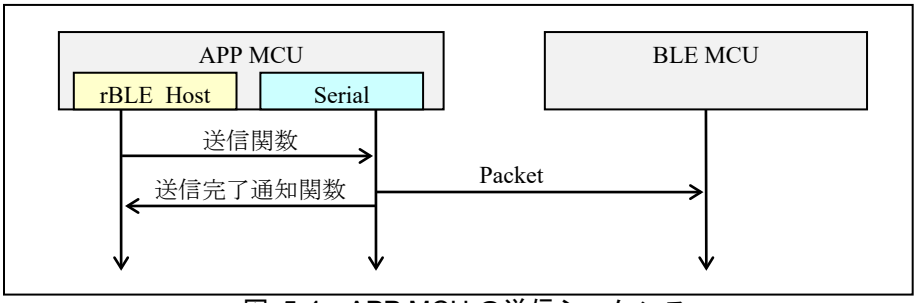


図 5-4　APP MCU の送信シーケンス

(2)　APP MCU の受信動作

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた受信シーケンスを示します。RSCIP パケットは可変長であるため rBLE_Host は 1 つの RSCIP パケットを受信するにあたり、受信関数を複数回呼び出します。

[受信開始時]：rBLE_Host は受信関数を呼び出します。これによりシリアル通信ドライバは RSCIP パケットの受信動作を開始し、データ受信を待ちます。

[パケット途中受信終了時]：シリアル通信ドライバは受信終了後、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知します。rBLE_Host は再度受信関数を呼び出すことで、シリアル通信ドライバは受信を再開します。

[パケット全体受信終了時]：シリアル通信ドライバは受信終了後、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知します。rBLE_Host は再度受信関数を呼び出し、次の RSCIP パケット受信を待ちます。

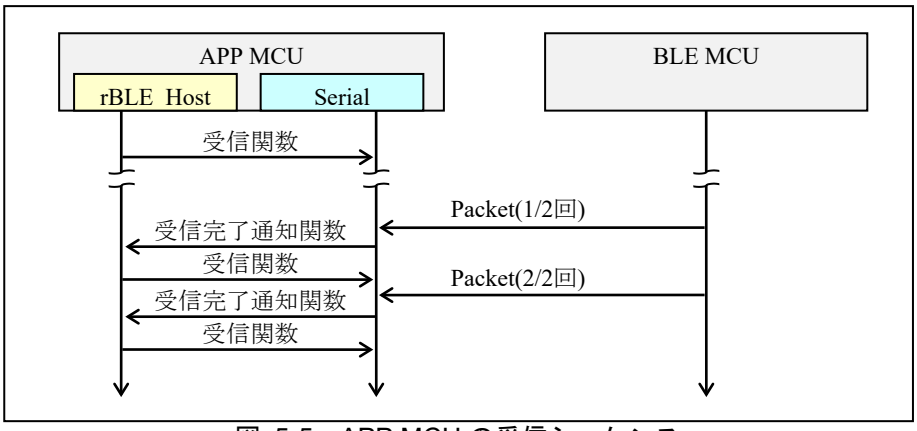


図 5-5　APP MCU の受信シーケンス

5.4.2　UART 3 線接続方式

本接続方式では、APP MCU と BLE MCU は下記に示すように UART のデータ信号線である TxD、RxD に加え、APP MCU がデータ送信時に BLE MCU を起床させるための制御信号線 WAKEUP を使用して通信します。

全二重通信が可能ですが、APP MCU からの送信時にはハンドシェイクを行う必要があります。これは BLE MCU が受信の準備を完了していることを確認するために必要な動作です。また、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行してください。

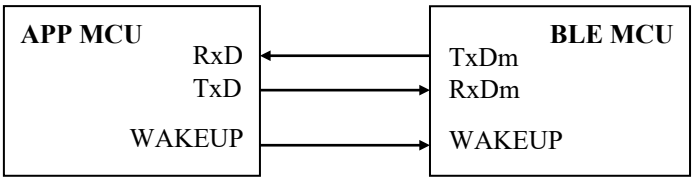


図 5-6　UART　3 線接続方式

BLE MCU 端子名称	方向	機能
TxDm (m=0,1)	BLE MCU→ APP MCU	シリアル出力データ信号
RxDm (m=0,1)	APP MCU→ BLE MCU	シリアル入力データ信号
WAKEUP(P30/INTP3) - Low Active	APP MCU→ BLE MCU	起床用外部トリガ入力信号 APP MCU は送信要求時、アクティブレベルに設定します BLE MCU からの ACK バイト(0x88)受信またはデータ受信を待ち、インアクティブレベルに戻します

※以降に記載するタイミングチャートでは、BLE MCU側の端子名称を記載します。

(1)　APP MCU の送信動作

APP MCU が BLE MCU へ RSCIP パケットを送信する場合のハンドシェイク手順は以下の[T1]～[T3]です。

[T1]：APP MCU は送信要求のため、WAKEUP 信号をアクティブレベルにします。
[T2]：APP MCU は BLE MCU からの ACK バイト(0x88)または RSCIP パケットを 1byte 検出します。
[T3]：APP MCU は WAKEUP 信号をインアクティブレベルにします。
[T4]：APP MCU は RSCIP パケットを送信します。

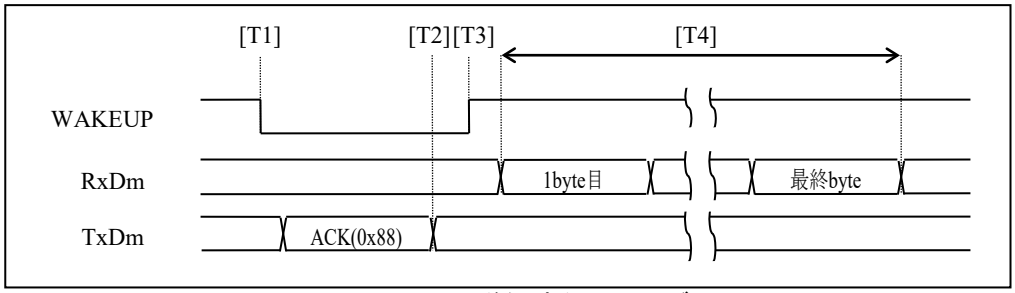


図 5-7　APP MCU の送信時タイミングチャート

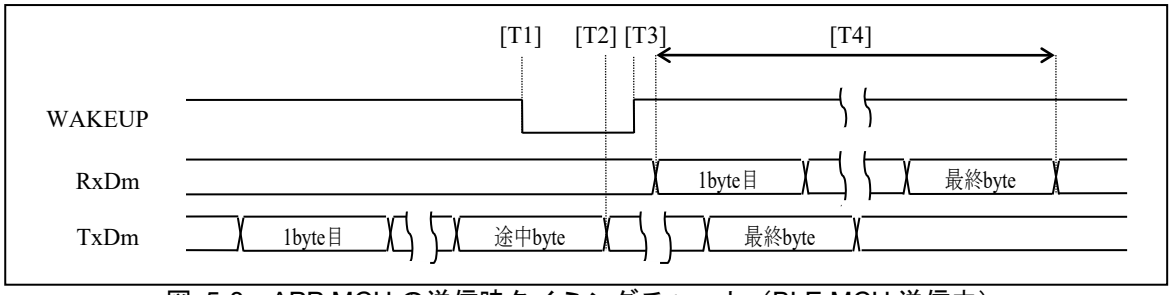


図 5-8　APP MCU の送信時タイミングチャート（BLE MCU 送信中）

シリアル通信ドライバは、送信要求後にタイムアウト監視を開始します。タイムアウトが発生した場合、シリアル通信ドライバは送信再要求のため、WAKEUP 信号をいったんインアクティブレベルに戻し、再度アクティブレベルにします[T1]。タイムアウト時間の推奨値は 5msec とします。

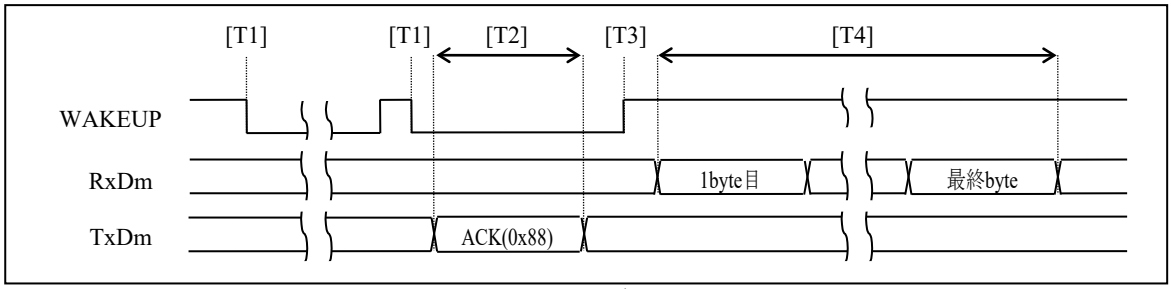


図 5-9　APP _MCU の送信時タイミングチャート（タイムアウト発生時）

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた送信シーケンスを示します。

[送信開始時] : rBLE_Host が送信関数を呼び出すことにより、シリアル通信ドライバは RSCIP パケットの送信動作を開始し、送信要求のため WAKEUP 信号をアクティブレベルにします[T1]。

[送信終了時] : RSCIP パケット送信[T1]～[T4]の完了時、シリアル通信ドライバは送信完了通知関数を呼び出すことで、送信完了を rBLE_Host に通知します。

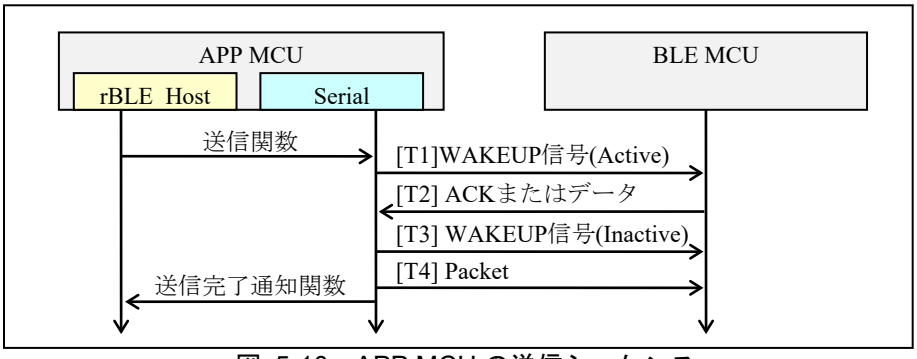


図 5-10　APP MCU の送信シーケンス

(2)　APP MCU の受信動作

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた受信シーケンスを示します。RSCIP パケットは可変長であるため rBLE_Host は 1 つの RSCIP パケットを受信するにあたり、受信関数を複数回呼び出します。

[受信開始時]：rBLE_Host は受信関数を呼び出します。これによりシリアル通信ドライバは RSCIP パケットの受信動作を開始し、データ受信を待ちます。

[パケット途中受信終了時]：シリアル通信ドライバは受信終了後、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知します。rBLE_Host は再度受信関数を呼び出すことで、シリアル通信ドライバは受信を再開します。

[パケット全体受信終了時]：シリアル通信ドライバは受信終了後、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知します。rBLE_Host は再度受信関数を呼び出し、次の RSCIP パケット受信を待ちます。

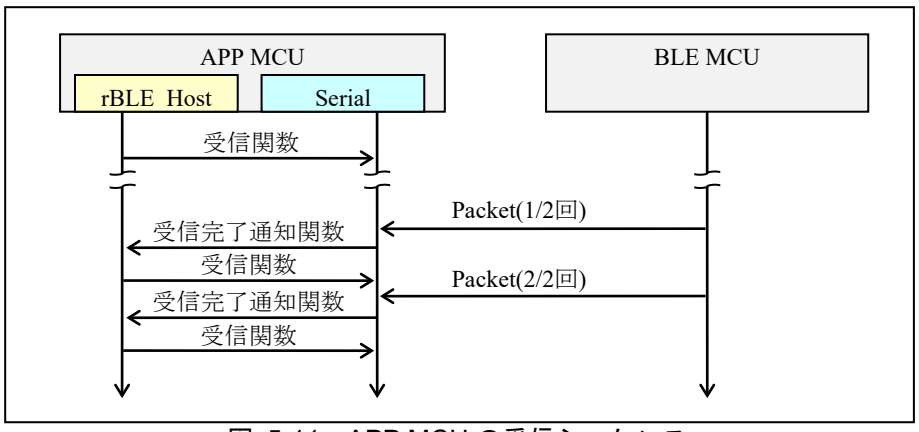


図 5-11　APP MCU の受信シーケンス

5.4.3　UART 2 線分岐接続方式

本接続方式では、APP MCU と BLE MCU は下記に示すように UART のデータ信号線である TxD、RxD に加え、APP MCU がデータ送信時に BLE MCU を起床させるため、APP MCU の TxD を分岐して BLE MCU の WAKEUP と接続し、通信します。

全二重通信が可能ですが、APP MCU からの送信時にはハンドシェイクを行う必要があります。これは BLE MCU が受信の準備を完了していることを確認するために必要な動作です。また、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行してください。

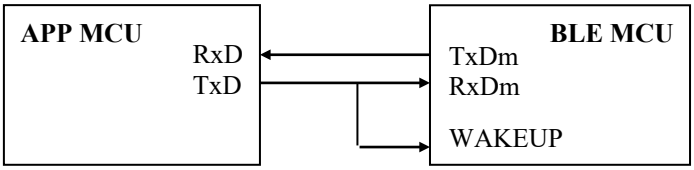


図 5-12　UART　2 線分岐接続方式

BLE MCU 端子名称	方向	機能
TxDm (m=0,1)	BLE MCU→ APP MCU	シリアル出力データ信号
RxDm (m=0,1)	APP MCU →BLE MCU	シリアル入力データ信号
WAKEUP(P30/INTP3) - Low Active	APP MCU →BLE MCU	起床用外部トリガ入力信号 APP MCU は送信要求時、アクティブレベルに設定します BLE MCU からの ACK バイト(0x88)受信またはデータ受信を待ち、インアクティブレベルに戻します

※以降に記載するタイミングチャートでは、BLE MCU 側の端子名称を記載します。

(1)　APP MCU の送信動作

APP MCU が BLE MCU へ RSCIP パケットを送信する場合のハンドシェイク手順は以下の[T1]～[T3]です。

[T1]：APP MCU は送信要求のため、REQ バイト(0xC0)を送信します。
[T2]：APP MCU は BLE MCU からの ACK バイト(0x88)または RSCIP パケットを 1byte 検出します。
[T3]：APP MCU は RSCIP パケットを送信します。

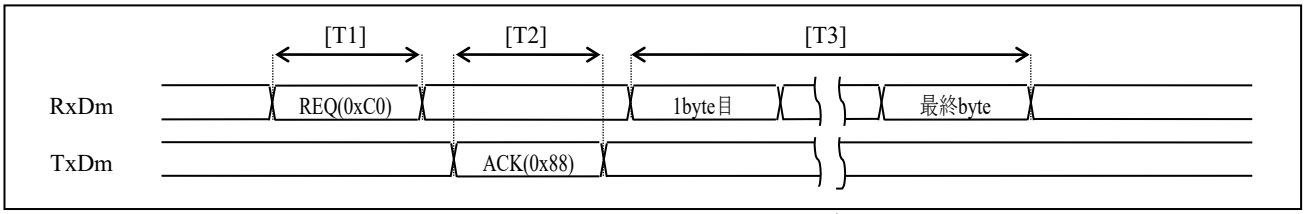


図 5-13　APP MCU の送信時タイミングチャート

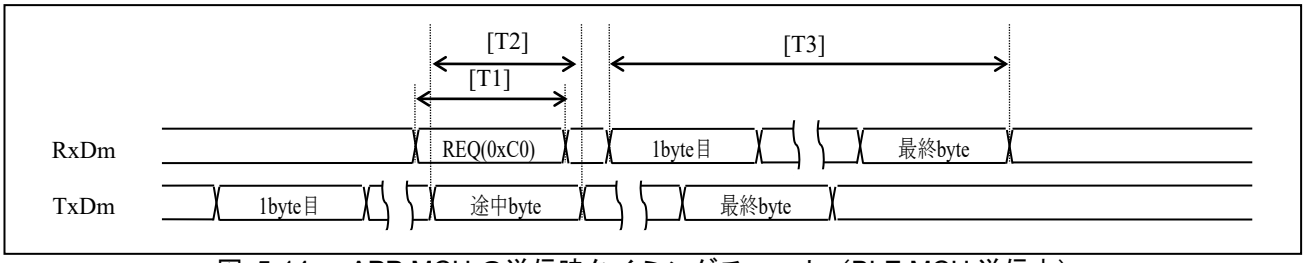


図 5-14　APP MCU の送信時タイミングチャート（BLE MCU 送信中）

シリアル通信ドライバは、送信要求後にタイムアウト監視を開始します。タイムアウトが発生した場合、シリアル通信ドライバは送信再要求のため、REQ バイトを送信します[T1]。タイムアウト時間の推奨値は 5msec とします。

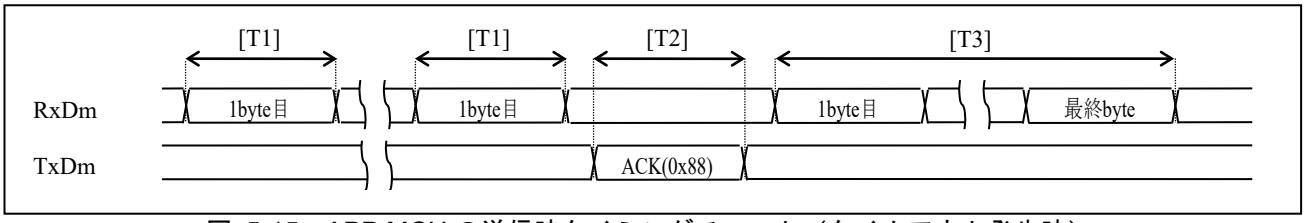


図 5-15　APP MCU の送信時タイミングチャート（タイムアウト発生時）

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた送信シーケンスを示します。

[送信開始時] : rBLE_Host が送信関数を呼び出すことにより、シリアル通信ドライバは RSCIP パケットの送信動作を開始し、送信要求のための REQ バイトを送信します[T1]。

[送信終了時] : RSCIP パケット送信[T1]～[T3]の完了時、シリアル通信ドライバは送信完了通知関数を呼び出すことで、送信完了を rBLE_Host に通知します。

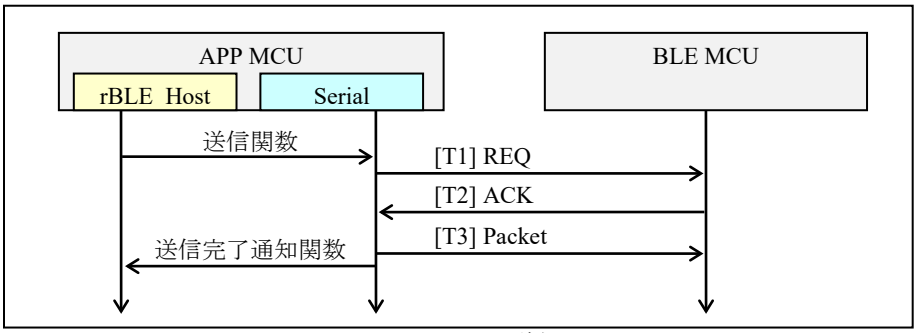


図 5-16　APP MCU の送信シーケンス

(2)　APP MCU の受信動作

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた受信シーケンスを示します。RSCIP パケットは可変長であるため rBLE_Host は 1 つの RSCIP パケットを受信するにあたり、受信関数を複数回呼び出します。

[受信開始時]：rBLE_Host は受信関数を呼び出します。これによりシリアル通信ドライバは RSCIP パケットの受信動作を開始し、データ受信を待ちます。

[パケット途中受信終了時]：シリアル通信ドライバは受信終了後、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知します。rBLE_Host は再度受信関数を呼び出すことで、シリアル通信ドライバは受信を再開します。

[パケット全体受信終了時]：シリアル通信ドライバは受信終了後、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知します。rBLE_Host は再度受信関数を呼び出し、次の RSCIP パケット受信を待ちます。

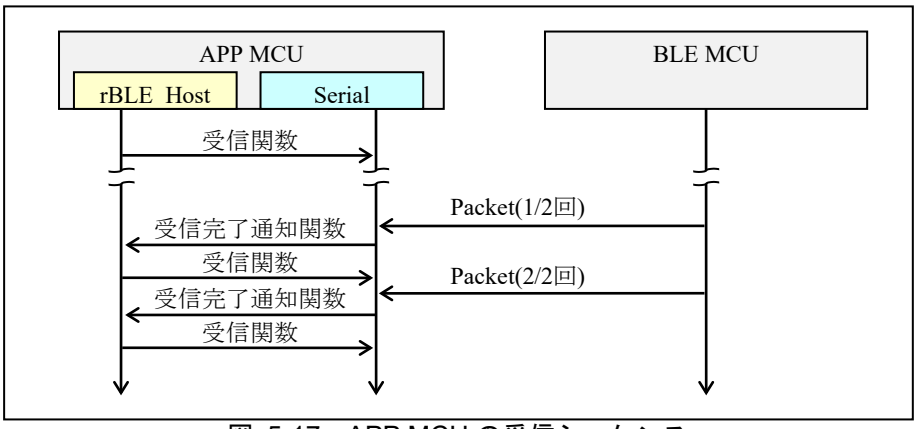


図 5-17　APP MCU の受信シーケンス

5.4.4　CSI 4 線接続方式

本接続方式では、APP MCU と BLE MCU は下記に示すように CSI のデータ信号線である SO、SI、SCK に加え、APP MCU と BLE MCU の通信方向、通信タイミングを制御するための制御信号線 SDIR を使用して通信します。

通信は半二重であり、送信時または受信時にはハンドシェイクを行う必要があります。これは BLE MCU が受信または送信の準備を完了していることを確認するため、BLE MCU からの送信要求を APP MCU に通知するため、半二重通信においてその通信方向を確定するために必要な動作です。また、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行してください。

BLE MCU 側には、APP MCU からの応答待ちのタイムアウトを実装しています。タイムアウトの実装の為、タイマ・アレイ・ユニットを使用しています。

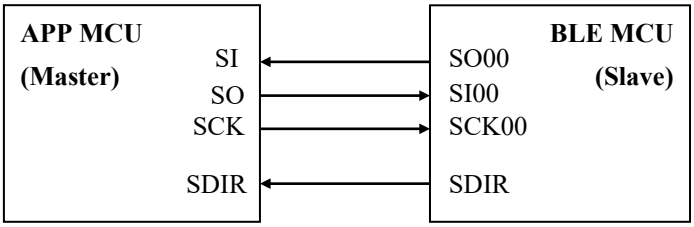


図 5-18　CSI　4 線接続方式

BLE MCU 端子名称	方向	概要
SO00	BLE MCU→ APP MCU	シリアル出力データ信号(MISO:MasterInput-SlaveOutput)
SI00	APP MCU→ BLE MCU	シリアル入力データ信号 (MOSI:MasterOutput-SlaveInput)
SCK00	APP MCU→ BLE MCU	データ通信タイミングクロック信号
SDIR(P23)	BLE MCU→ APP MCU	データ転送方向制御 / 通信応答制御信号 Low：データの転送方向は、BLE MCU⇒APP MCU High：データの転送方向は、APP MCU⇒BLE MCU パルス(High⇒Low⇒High)：パルス幅は BLE MCU 動作クロック時間×4 (APP MCU から Packet 先頭 1byte 送信完了時)BLE MCU からの通信許可応答 (APP MCU から Packet 先頭 1byte 送信完了以外の時)BLE MCU からの送信要求

※以降に記載するタイミングチャートでは、BLE MCU 側の端子名称を記載します。

(1)　APP MCU の送信動作

APP MCU が BLE MCU へ RSCIP パケットを送信する場合のハンドシェイク手順は以下の[T1]～[T3]です。

[T1]：APP MCU は送信要求のため、RSCIP パケット先頭 1 バイトを送信し、SDIR 信号のパルスを待ちます。
※SDIR 信号のパルス待ちでタイムアウトが発生した場合、先頭 1byte を再送信する必要があります。
[T2]：APP MCU は通信許可応答の SDIR 信号のパルスを検出します。
[T3]：APP MCU は RSCIP パケットの 2 バイト目から最終バイトまでを連続送信します。

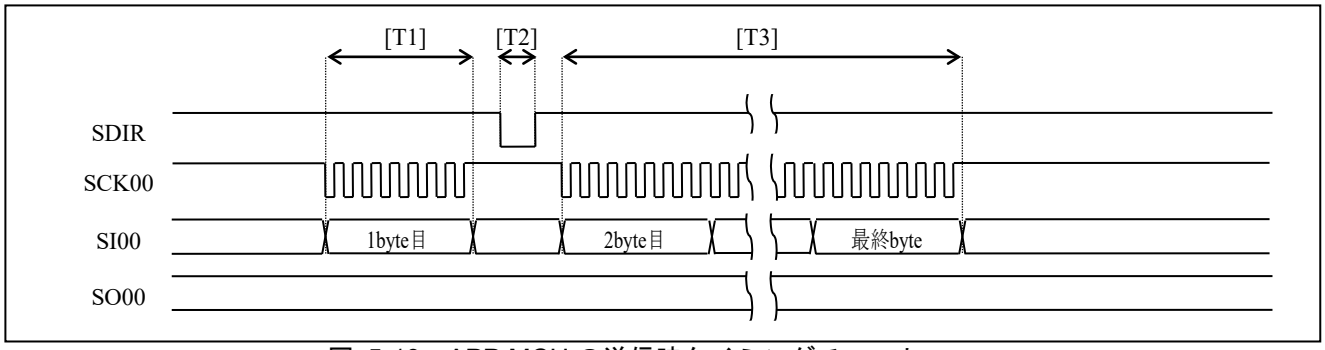


図 5-19　APP MCU の送信時タイミングチャート

シリアル通信ドライバは、送信要求後にタイムアウト監視を開始します。タイムアウトが発生した場合、シリアル通信ドライバは送信再要求のため、RSCIP パケットの先頭 1 バイトを再度送信します[T1]。タイムアウト時間の推奨値は 5msec とします。

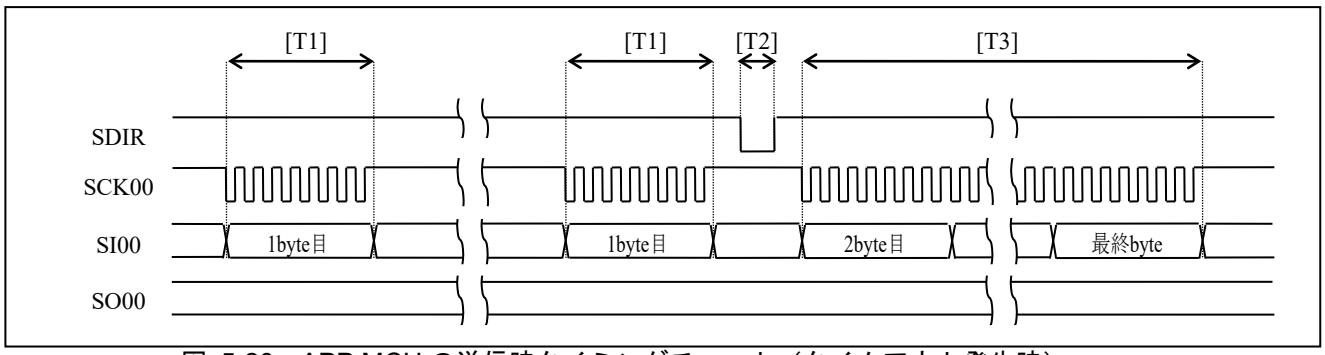


図 5-20　APP MCU の送信時タイミングチャート（タイムアウト発生時）

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた送信シーケンスを示します。

[送信開始時] : rBLE_Host が送信関数を呼び出すことにより、シリアル通信ドライバは RSCIP パケットの送信動作を開始します[T1]。

[送信終了時] : RSCIP パケット送信[T1]～[T3]の完了時、シリアル通信ドライバは送信完了通知関数を呼び出すことで、送信完了を rBLE_Host に通知します。

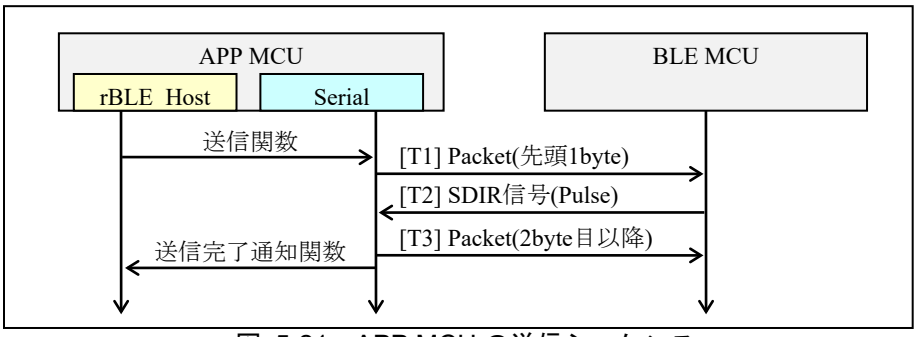


図 5-21　APP MCU の送信シーケンス

(2)　APP MCU の受信動作

APP MCU が BLE MCU から RSCIP パケットを受信する場合のハンドシェイク手順は以下の[R1]～[R5]です。半二重通信のため、[R1]～[R5]の実行時は APP MCU からの送信は禁止とします。

[R1]：APP MCU は送信要求の SDIR 信号のパルスを待ちます。
[R2]：APP MCU は SDIR 信号のパルスを検出すると、ACK バイト(0x88)を送信し、SDIR 信号の Low を待ちます。
[R3]：APP MCU は SDIR 信号の Low を検出します。
[R4]：APP MCU はクロックを供給し、RSCIP パケットを受信します。
[R5]：APP MCU は SDIR 信号の High を検出します。

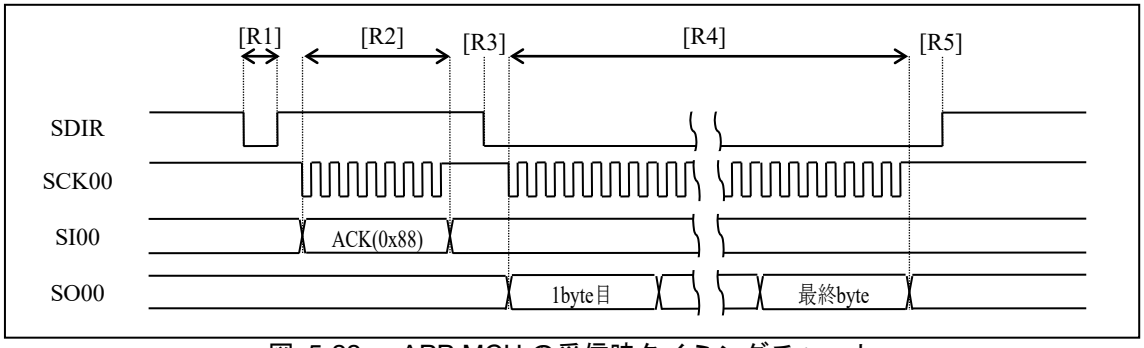


図 5-22　APP MCU の受信時タイミングチャート

APP MCU の送信要求と BLE MCU の送信要求が衝突した場合、BLE MCU は送信を延期し、APP MCU の RSCIP パケットを受信します。BLE MCU は RSCIP パケット受信が完了後に再度 SDIR 信号でパルスを出力します。

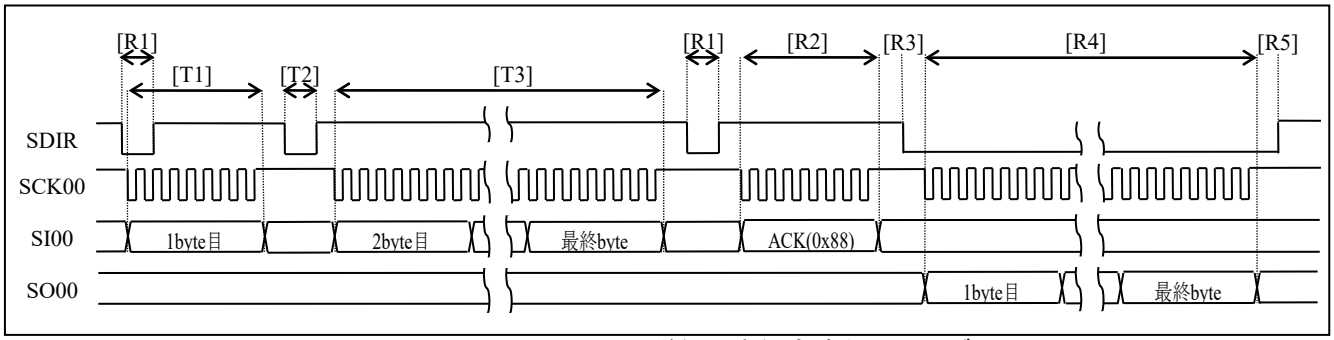


図 5-23 APP MCU と BLE MCU の送信要求衝突時タイミングチャート

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた受信シーケンスを示します。RSCIP パケットは可変長であるため rBLE_Host は 1 つの RSCIP パケットを受信するにあたり、受信関数を複数回呼び出します。

[受信開始時]：rBLE_Host は受信関数を呼び出します。これによりシリアル通信ドライバは RSCIP パケットの受信動作を開始し、SDIR 信号のパルスを待ちます[R1]。

[パケット途中受信終了時]：[R1]～[R4]の終了後、シリアル通信ドライバは、受信完了通知関数を呼び出すことで受信完了を rBLE_Host に通知し、rBLE_Host は再度受信関数を呼び出します。シリアル通信ドライバは、受信状態取得関数を確認し、パケット受信途中であれば、再度クロック供給し、受信を再開します[R4]。

[パケット全体受信終了時]：[R4]の終了後、シリアル通信ドライバは、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知し、rBLE_Host は再度受信関数を呼び出します。シリアル通信ドライバは、受信状態取得関数を確認し、パケット受信完了であれば、SDIR 信号が High となることを待ちます[R5]。

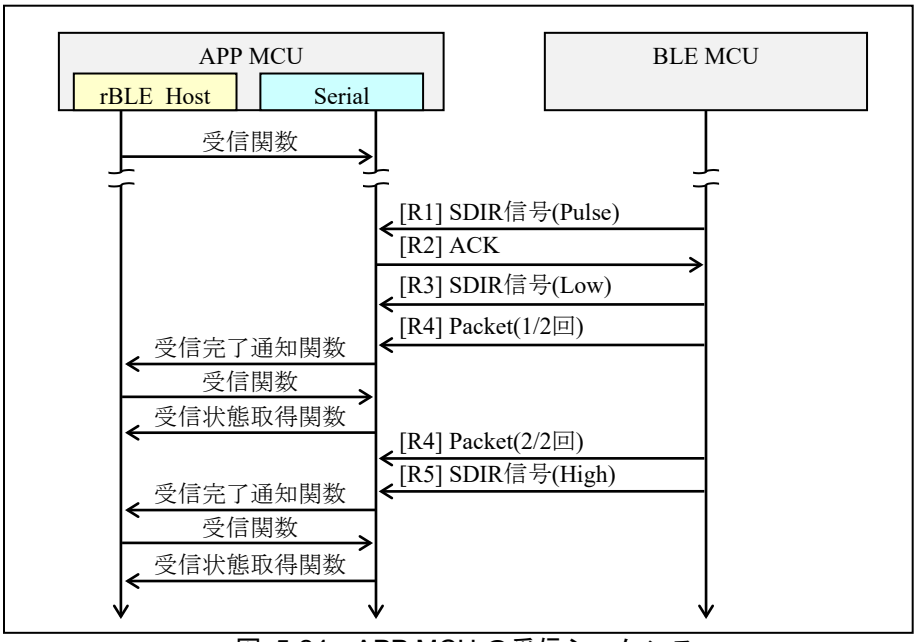


図 5-24　APP MCU の受信シーケンス

5.4.5　CSI 5 線接続方式

本接続方式では、APP MCU と BLE MCU は下記に示すように CSI のデータ信号線である SO、SI、SCK に加え、APP MCU と BLE MCU の通信方向、通信タイミングを制御するための制御信号線 SDIR および APP MCU がデータ送信時に BLE MCU を起床させるための制御信号線 WAKEUP を使用して通信します。

通信は半二重であり、送信時または受信時にはハンドシェイクを行う必要があります。これは BLE MCU が受信または送信の準備を完了していることを確認するため、BLE MCU からの送信要求を APP MCU に通知するため、半二重通信においてその通信方向を確定するために必要な動作です。また、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行してください。

BLE MCU 側には、APP MCU からの応答待ちのタイムアウトを実装しています。タイムアウトの実装の為、タイマ・アレイ・ユニットを使用しています。

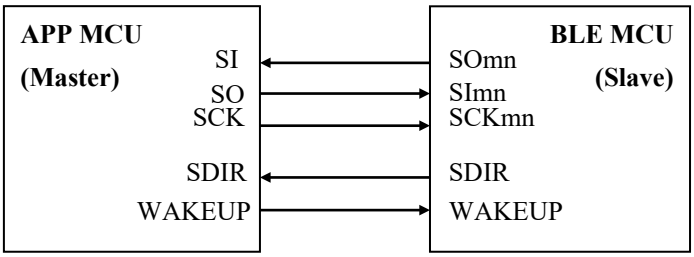


図 5-25　CSI 5 線接続方式

BLE MCU 端子名称	方向	概要
SOmn (mn=00,20)	BLE MCU→ APP MCU	シリアル出力データ信号(MISO:MasterInput-SlaveOutput)
SImn (mn=00,20)	APP MCU→ BLE MCU	シリアル入力データ信号 (MOSI:MasterOutput-SlaveInput)
SCKmn (mn=00,20)	APP MCU→ BLE MCU	データ通信タイミングクロック信号
SDIR(P23)	BLE MCU→ APP MCU	データ転送方向制御 / 通信応答制御信号 Low　：データの転送方向は、BLE MCU⇒APP MCU High　：データの転送方向は、APP MCU⇒BLE MCU パルス(High⇒Low⇒High)：パルス幅は BLE MCU 動作クロック時間×4 (WAKEUP 信号アクティブ時)BLE MCU からの通信許可応答 (WAKEUP 信号インアクティブ時)BLE MCU からの送信要求
WAKEUP(P30/INTP3) – Low Active	APP MCU→ BLE MCU	起床用外部トリガ入力信号 APP MCU からの送信要求時、アクティブレベルに設定 ※BLE MCU からの SDIR の応答を待って、インアクティブレベルに戻す

※以降に記載するタイミングチャートでは、BLE MCU 側の端子名称を記載します。

(1)　APP MCU の送信動作

APP MCU が BLE MCU へ RSCIP パケットを送信する場合のハンドシェイク手順は以下の[T1]～[T4]です。

[T1]：APP MCU は送信要求のため、WAKEUP 信号をアクティブレベルにし、SDIR 信号のパルスを待ちます。
※SDIR 信号のパルス待ちでタイムアウトが発生した場合、WAKEUP 信号をいったんインアクティブレベルに戻し、再度アクティブレベルにする必要があります。
[T2]：APP MCU は通信許可応答の SDIR 信号のパルスを検出します。
[T3]：WAKEUP 信号をインアクティブレベルに戻します。
[T4]：APP MCU は RSCIP パケットの先頭バイトから最終バイトまでを連続送信します。

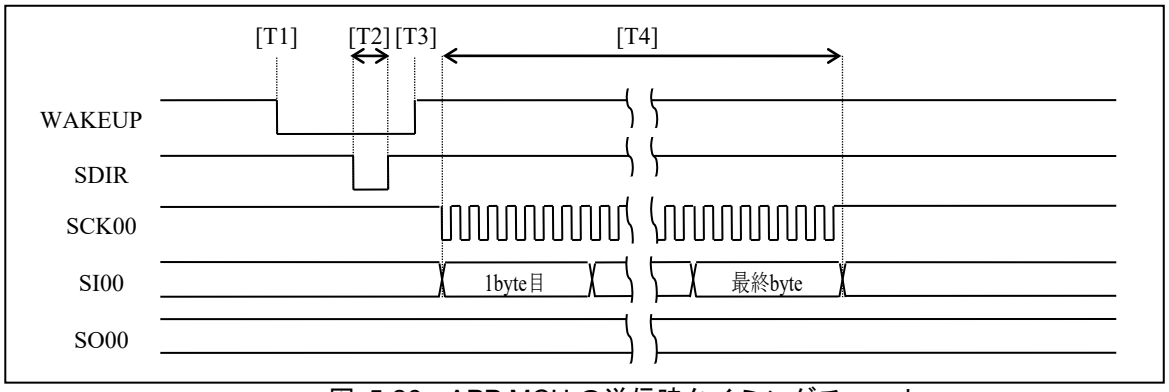


図 5-26　APP MCU の送信時タイミングチャート

シリアル通信ドライバは、送信要求後にタイムアウト監視を開始します。タイムアウトが発生した場合、シリアル通信ドライバは送信再要求のため、WAKEUP信号をいったんインアクティブレベルに戻し、アクティブレベルを再度出力します[T1]。タイムアウト時間の推奨値は5msecとします。

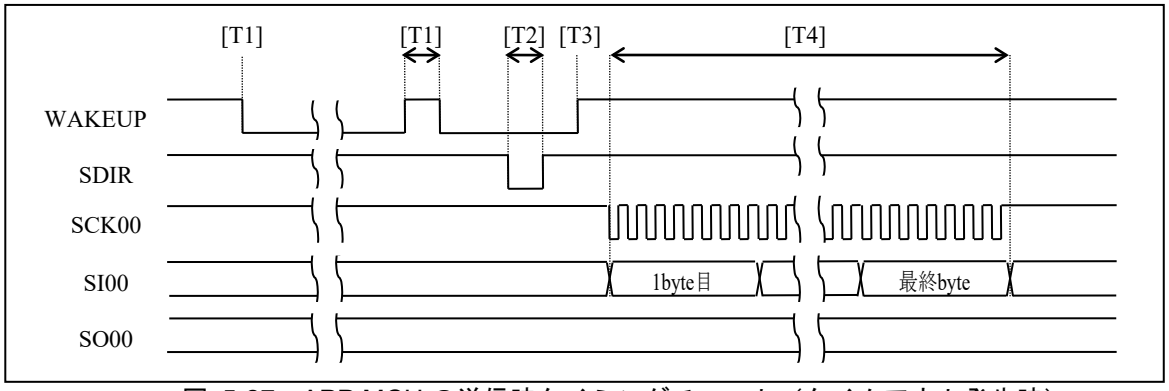


図 5-27　APP MCU の送信時タイミングチャート（タイムアウト発生時）

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた送信シーケンスを示します。

[送信開始時]：シリアル通信ドライバは、rBLE_Host が送信関数を呼び出すことにより RSCIP パケットの送信を開始し、WAKEUP 信号をアクティブレベルにします[T1]。

[送信終了時]：RSCIP パケット送信[T1]～[T4]の完了時、シリアル通信ドライバは rBLE_Host の送信完了通知関数を呼び出すことで、送信完了を通知します。

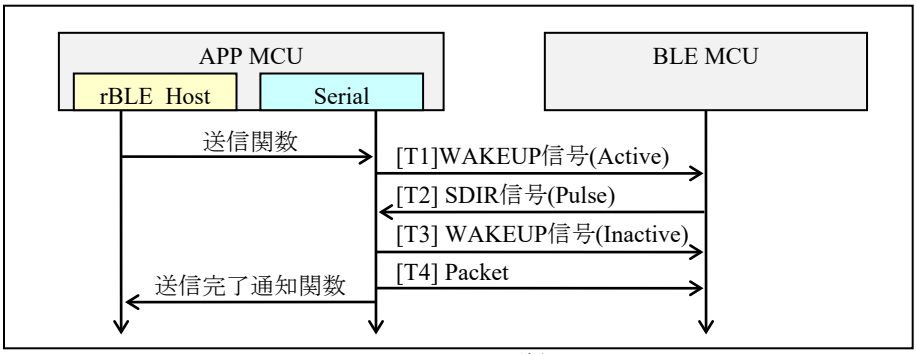


図 5-28　APP MCU の送信シーケンス

(2)　APP MCU の受信動作

APP MCU が BLE MCU から RSCIP パケットを受信する場合のハンドシェイク手順は以下の[R1]～[R5]です。
半二重通信のため、[R1]～[R5]の実行時は APP MCU からの送信は禁止とします。

[R1]：APP MCU は送信要求の SDIR 信号のパルスを待ちます。
[R2]：APP MCU は SDIR 信号のパルスを検出すると、ACK バイト(0x88)を送信し、SDIR 信号の Low を待ちます。
[R3]：APP MCU は SDIR 信号の Low を検出します。
[R4]：APP MCU はクロックを供給し、RSCIP パケットを受信します。
[R5]：APP MCU は SDIR 信号の High を検出します。

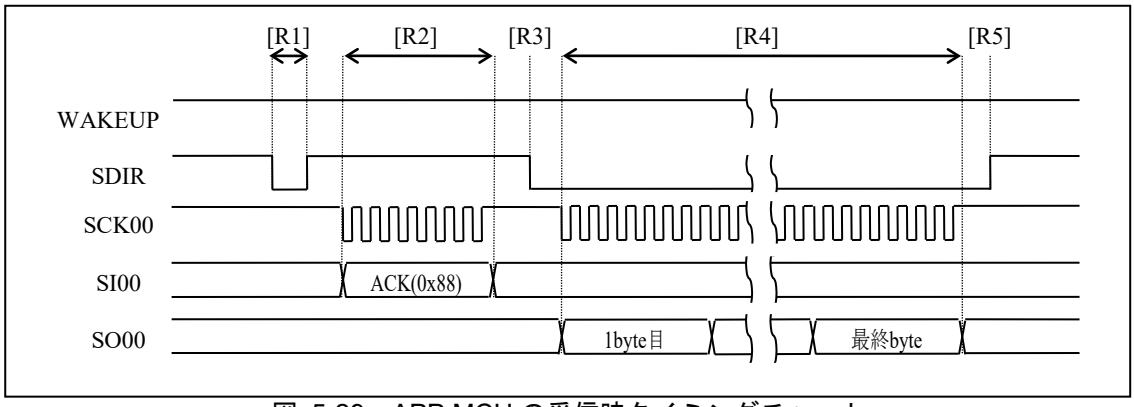


図 5-29　APP MCU の受信時タイミングチャート

APP MCU の送信要求と BLE MCU の送信要求が衝突した場合、BLE MCU は送信を延期し、APP MCU の RSCIP パケットを受信します。BLE MCU は RSCIP パケット受信が完了後に SDIR 信号でパルスを再度出力します。

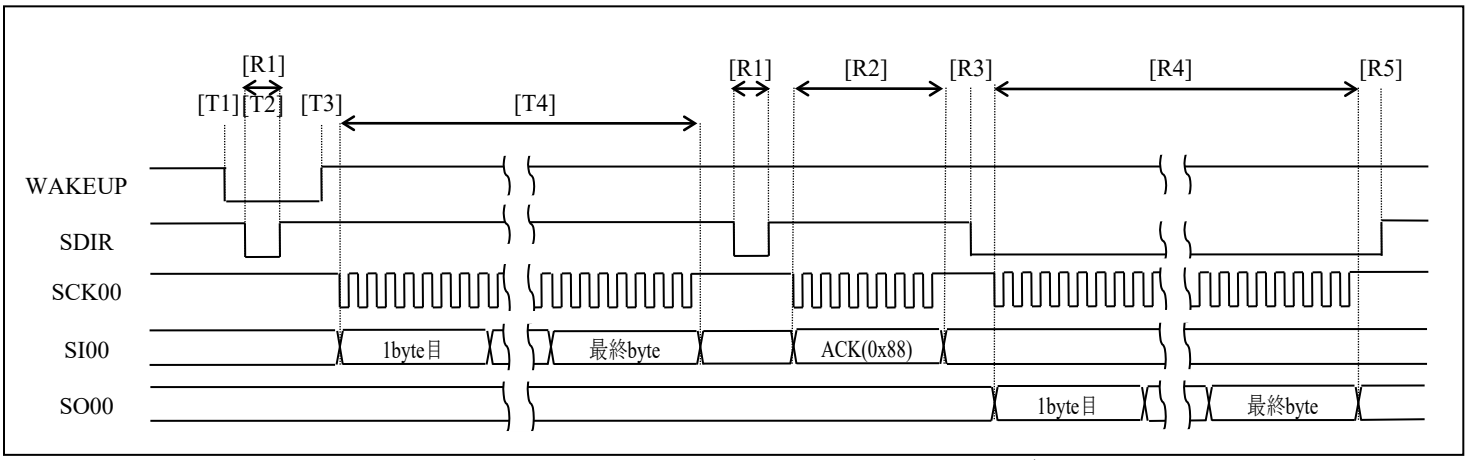


図 5-30　APP MCU と BLE MCU の送信要求衝突時タイミングチャート

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた受信シーケンスを示します。RSCIP パケットは可変長であるため rBLE_Host は 1 つの RSCIP パケットを受信するにあたり、受信関数を複数回呼び出します。

[受信開始時]：rBLE_Host は受信関数を呼び出します。これによりシリアル通信ドライバは RSCIP パケットの受信動作を開始し、SDIR 信号のパルスを待ちます[R1]。

[パケット途中受信終了時]：[R1]～[R4]の終了後、シリアル通信ドライバは、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知し、rBLE_Host は再度受信関数を呼び出します。シリアル通信ドライバは、受信状態取得関数を確認し、パケット受信途中であれば、再度クロック供給し、受信を再開します[R4]。

[パケット全体受信終了時]：[R4]の終了後、シリアル通信ドライバは、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知し、rBLE_Host は再度受信関数を呼び出します。シリアル通信ドライバは、受信状態取得関数を確認し、パケット受信完了であれば、SDIR 信号が High となることを待ちます[R5]。

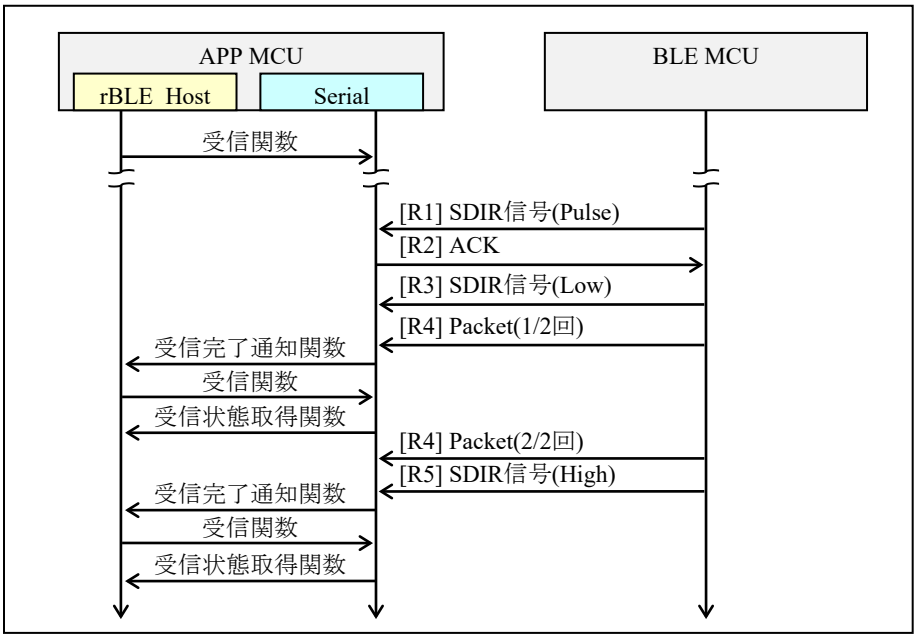


図 5-31　APP MCU の受信シーケンス

5.4.6　IIC 3 線接続方式

本接続方式では、APP MCU と BLE MCU は下記に示すように IIC のデータ信号線である SDA、SCL に加え、IIC スレーブ（BLE MCU）の送信要求を制御するための制御信号線 REQ を使用して通信します。

なお、SCL 端子と SDA 端子はオープン・ドレイン出力で使用するため、各ラインにはプルアップ抵抗が必要です。

通信は半二重であり、送信または受信は BLE MCU の IIC スレーブアドレスを指定して行います。確実な通信を行うため、送信時にはタイムアウトによる監視を行い、タイムアウト発生時には送信を再実行してください。BLE MCU 側には、APP MCU からの応答待ちのタイムアウトを実装しています。タイムアウトの実装の為、タイマ・アレイ・ユニットを使用しています。

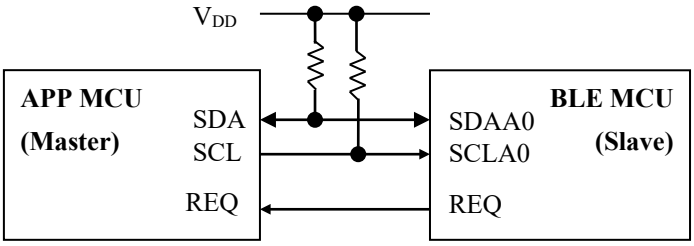


図 5-32　IIC　3 線接続方式

BLE MCU 端子名称	方向	概要
SDAA0	BLE MCU⇔ APP MCU	IIC シリアル・データ・バス・ライン
SCLA0	APP MCU→ BLE MCU	IIC シリアル・クロック・ライン
REQ(P23)	BLE MCU→ APP MCU	IIC スレーブデータ送信要求制御 Low：　BLE MCU のデータ送信要求あり High：　BLE MCU のデータ送信要求なし

※以降に記載するタイミングチャートでは、BLE MCU 側の端子名称を記載します。

(1)　APP MCU の送信動作

APP MCU が BLE MCU へ RSCIP パケットを送信する場合の手順は以下の[T1]～[T4]です。

[T1]：APP MCU は送信要求のため、スタートコンディションを生成します。
[T2]:APP MCU は BLE MCU のスレーブアドレス(7 ビット)と転送方向 Write(1 ビット)を送信し、BLE MCU の ACK を検出します。
[T3]：APP MCU は RSCIP パケットの先頭バイトから最終バイトまでを送信します。
[T4]：APP MCU はストップコンディションを生成し、通信を終了させます。

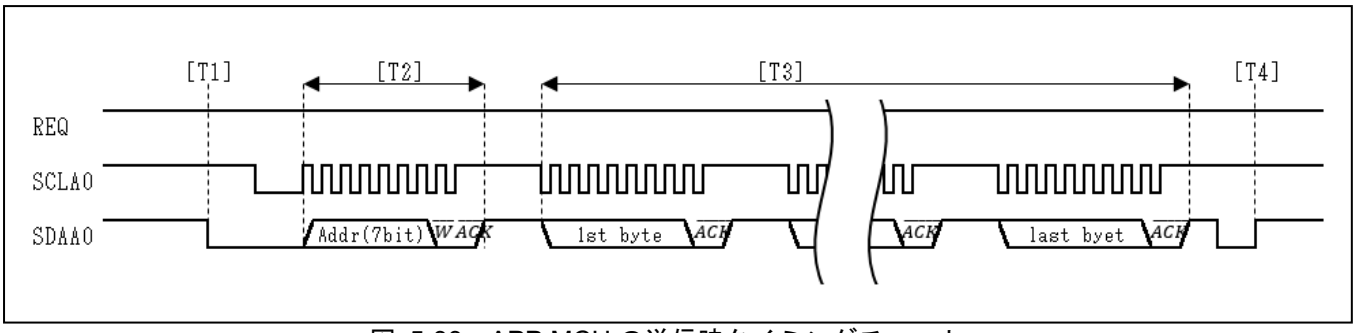


図 5-33　APP MCU の送信時タイミングチャート

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた送信シーケンスを示します。

[送信開始時]：rBLE_Host が送信関数を呼び出すことにより、シリアル通信ドライバは RSCIP パケットの送信動作を開始します[T1]。

[送信終了時]：RSCIP パケット送信[T1]～[T4]の完了時、シリアル通信ドライバは送信完了通知関数を呼び出すことで、送信完了を rBLE_Host に通知します。

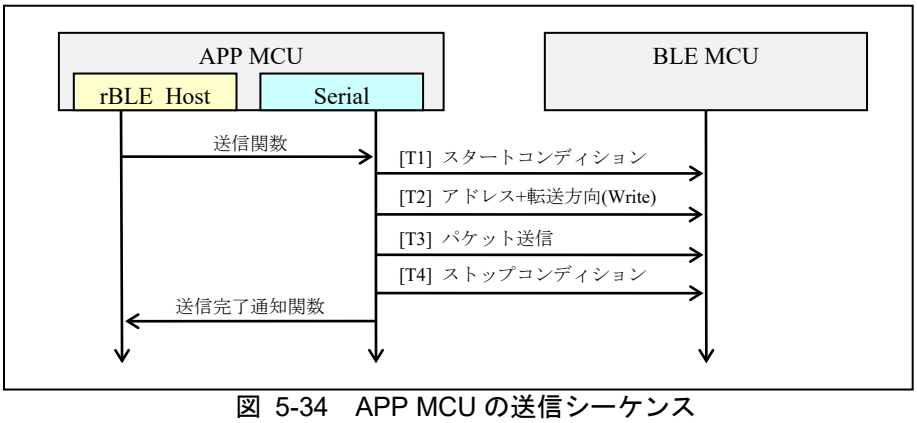


図 5-34　APP MCU の送信シーケンス

(2)　APP MCU の受信動作

APP MCU が BLE MCU から RSCIP パケットを受信する場合の手順は以下の[R1]～[R7]です。
半二重通信のため、[R1]～[R7]の実行時は APP MCU からの送信は禁止とします。

[R1]：APP MCU は BLE MCU からの REQ 信号 Low を待ちます。
[R2]：APP MCU は REQ 信号 Low エッジを検出後、スタートコンディションを生成します。
[R3]：APP MCU は BLE MCU のスレーブアドレス(7 ビット)と転送方向 Read(1)を送信し、BLE MCU の ACK を検出します。
[R4]：APP MCU はクロックを供給し、RSCIP パケットを受信し ACK を応答します。
[R5]：BLE MCU は最終バイト送信前に REQ 信号を High にします。
[R6]：APP MCU は REQ 信号の High を検出した場合、受信最終バイトとして NACK を応答します。
[R7]：APP MCU はストップコンディションを生成し、通信を終了させます。

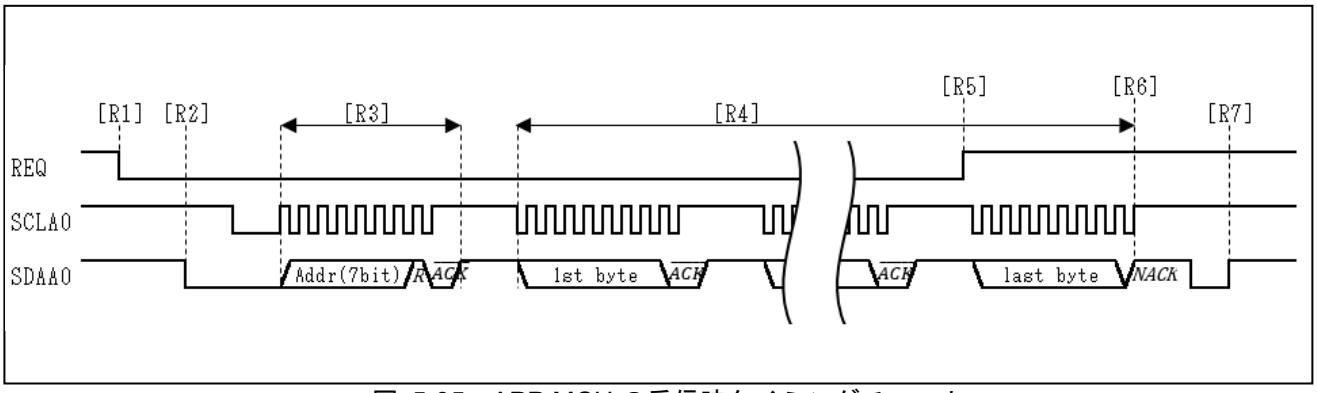


図 5-35　APP MCU の受信時タイミングチャート

APP MCU の送信要求と BLE MCU の送信要求が衝突した場合、BLE MCU は REQ 信号を High にして送信を延期し[R8]、APP MCU の RSCIP パケットを受信します。BLE MCU は RSCIP パケット受信完了後に REQ 信号 Low を再度出力します。

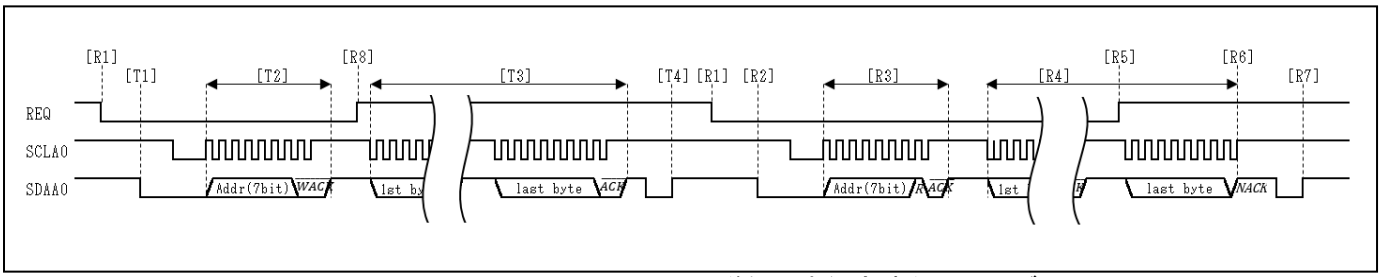


図 5-36　APP MCU と BLE MCU の送信要求衝突時タイミングチャート

rBLE_Host およびシリアル通信ドライバの関数呼び出しを含めた受信シーケンスを示します。RSCIP パケットは可変長であるため rBLE_Host は 1 つの RSCIP パケットを受信するにあたり、受信関数を複数回呼び出します。

[受信開始時]：rBLE_Host は受信関数を呼び出します。これによりシリアル通信ドライバは RSCIP パケットの受信動作を開始し、REQ 信号 Low を待ちます[R1]。

[パケット途中受信終了時]: [R4]の途中で指定された受信サイズに達した場合、シリアル通信ドライバは、受信完了通知関数を呼び出すことで、受信完了を rBLE_Host に通知し、rBLE_Host は再度受信関数を呼び出し、受信を再開します。

[パケット全体受信終了時]：[R7]の終了後、シリアル通信ドライバは、受信完了通知関数を呼び出すことで受信完了を rBLE_Host に通知し、rBLE_Host は再度受信関数を呼び出し、次の REQ 信号 Low を待ちます。

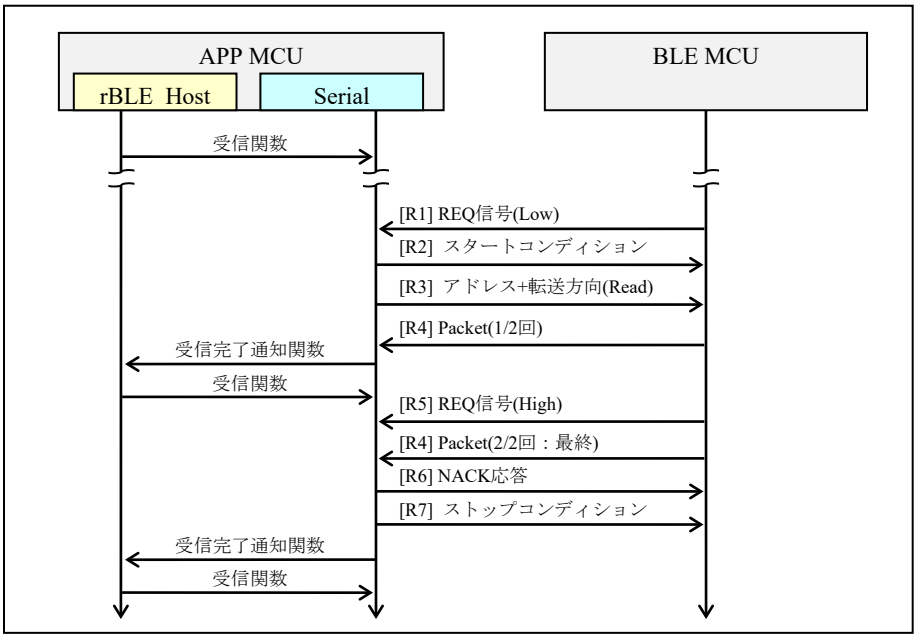


図 5-37　APP MCU の受信シーケンス

5.5　顧客固有情報

BLE ソフトウェアは、CodeFlash 最終ブロックの先頭 512 バイトを顧客固有情報領域として扱います。BLE ソフトウェアで参照する顧客固有情報を表 5-8 に示します。

Bluetooth Device アドレス(以降、BD アドレス)を本領域に書き込むことで、BLE MCU 毎に異なる BD アドレスを設定することが可能です。なお、本領域の BD アドレスは DataFlash 領域の BD アドレスより優先順位が低いため、顧客固有情報書き込み後も BD アドレスの変更は可能です。

本領域に書き込まれたデバイス名称は、GAP のデバイス名特性値として対向機に公開します。本領域に有効なデバイス名称が書き込まれていない場合、GATT データベースのデフォルト値が GAP のデバイス名特性値として対向機に公開されます。

表 5-8　BLE ソフトウェアで参照する顧客固有情報

顧客固有情報	アドレス	サイズ	記述
BDアドレス	0x3FC00 (※1)	6 byte	Bluetooth Device Address デバイスを識別するためのアドレス
デバイス名称	0x3FC06 (※1)	66 byte	Bluetooth Device Name デバイスを識別するためのユーザフレンドリーな名前 0x3FC06：デバイス名称長(1～65) 0x3FC07～0x3FC48：デバイス名称(UTF-8文字列)

※1 : 本アドレスは CodeFlash 256KB の値となります。最終ブロックのアドレスは CodeFlash のサイズによって異なります。

顧客固有情報は BLE ソフトウェアの実行ファイルとは別に CodeFlash へ書き込む必要があります。内蔵 Flash への書き込み方法は、Renesas Flash Programmer フラッシュ書き込みソフトウェア ユーザーズマニュアル (R20UT2907JJ0202) を参照してください。

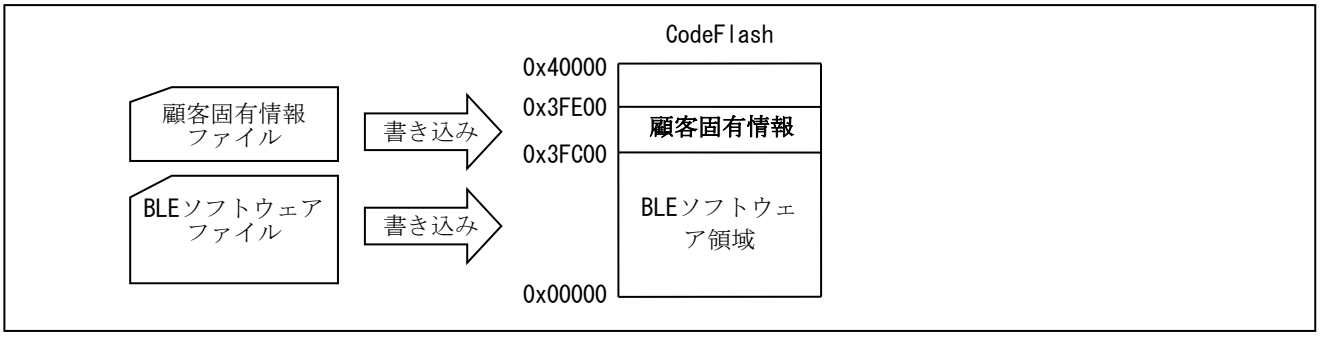


図 5-38　BLE ソフトウェアと顧客固有情報の書き込み領域

5.6　自 Bluetooth Device アドレスの決定

BLE ソフトウェアでは、3 種類の Bluetooth Device アドレス(以降、BD アドレス)格納領域を下記の順に検索し、自 BD アドレスを決定します。BD アドレス 6 バイト内に 0xFF 以外の値が設定されている場合に、有効な BD アドレスとみなします。

1. DataFlash 領域（7.20.4.1 Bluetooth Device アドレス書き込み機能 参照）
2. CodeFlash 顧客固有情報領域（5.5

顧客固有情報 参照）

3. config.h の CFG_TEST_BDADDR 定義値

BD アドレスの決定は BLE ソフトウェア起動時に行うため、DataFlash 領域に BD アドレスを書き込んだ場合、一度 BLE MCU をリセットする必要があります。

6.　実行ファイル作成方法

本章では、BLE ソフトウェアのうち RL78/G1D 上で動作するプログラムの実行ファイル(Hex ファイル)作成方法について説明します。

6.1　構成定義パラメータの変更

実行ファイル作成の際に、ユーザによって設定変更可能な項目を表 6-1 に示します。次項よりその設定方法について説明します。

表 6-1　構成定義パラメータ

ユーザ設定可能な項目		設定可能な値
最大同時接続台数		1～8
Heap 領域のサイズ		ユーザアプリが使用するサイズを追加可能
動作周波数設定		メイン・システム・クロック 高速オンチップ・オシレータ (HOCO) 4MHz、8MHz、16MHz、32MHz 発振子接続 (X1,X2) 4MHz、8MHz、16MHz 外部クロック入力 (RF または APP MCU) 4MHz、8MHz、16MHz サブシステム・クロック 発振子接続(XT1,XT2) 外部クロック入力(EXCLKS)
MCU 部の初期化設定		クロック(16.384kHz/32.768kHz)の出力可否
RF 部の初期化設定		RF 部ユーザ・オプション ・外部パワーアンプの使用可否 ・内蔵 DC-DC コンバータの使用可否 ・RF 用スロー・クロック部の選択 ・高速クロックの出力可否 ・高速クロックの選択 ・Sleep Clock Accuracy の選択 20ppm～500ppm
シリアル通信のボー・レート/転送クロック	UART	4800bps～250kbps
	CSI	4800bps～250kbps
	IIC	100kbps～400kbps
消費電流ピーク通知機能*1	機能のオン・オフ	オン、オフ
	通知開始時刻	1msec 前、2msec 前、4msec 前
	ピーク通知	ピーク通知後の処理内容
	ピーク終了通知	ピーク終了通知後の処理内容
HCI パケットモニタ機能*2		有効、無効
各プロファイルの有効・無効設定*2		有効、無効
GAP パラメータ	デバイスサーチパラメータ	サーチ時間 スキャンインターバル スキャンウインドウ
	Limited Discoverable モードパラメータ	発見可能時間

ユーザ設定可能な項目		設定可能な値
	Auto / Selective コネクション パラメータ	スキャンインターバル スキャンウインドウ コネクションインターバル スレーブレイテンシー スーパービジョンタイムアウト
	プライバシーパラメータ	プライベートアドレス変更間隔
GAP 特性	デバイス名称	UTF-8 文字列
	Appearance	Category 値
	Peripheral Preferred Connection Parameters	コネクションインターバル スレーブレイテンシー スーパービジョンタイムアウト
GATT 特性	Service Changed 特性	サービス変更開始ハンドル サービス変更終了ハンドル

【注】 *1　消費電流ピーク通知機能の設定変更については 7.20.1 を参照ください
*2　HCI パケットモニタ機能および設定変更については 12 を参照ください。
*3　ユーザでのプロファイルの作成方法は 7.4 を参照ください。

6.1.1　最大同時接続台数

Master 動作時に、同時に接続可能なリモートデバイスの台数を、1～8 の範囲で任意に設定することが可能です。接続に必要となるメモリを Heap メモリから確保するため、用途に応じて同時接続台数を制限することで、RAM 使用量を削減することができます。Slave としてのみ動作する場合は、最大同時接続台数を 1 としてください。

最大同時接続台数は、下記の定義マクロの値で変更可能です。プロジェクトのコンパイルオプションで定義しています。

定義マクロ名：CFG_CON

【注】 全ての接続パラメータにおいて、最大同時接続台数の接続を保証するものではありません。また、確保可能な Heap サイズにより、最大同時接続台数が制限される場合があります。

6.1.2　Heap メモリの確保

配列 ke_mem_heap にて、BLE ソフトウェアが使用する Heap 領域を確保しています。現在の設定値（BLE_HEAP_SIZE）は、BLE ソフトウェアが動作するのに最低限のメモリ量です。Embedded 構成などで、ユーザアプリを BLE MCU 上で動作させ、ke_malloc を使用する場合には、ユーザアプリが必要な分を追加してください。

Heap メモリサイズは、下記のソースファイルで変更可能です。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78
ファイル名：config.h

6.1.3　動作周波数変更

BLE ソフトウェアは、BLE MCU の動作周波数に依存する処理が実装されています。よって、BLE MCU の動作周波数を変更する場合、BLE ソフトウェアに対して動作周波数を定義マクロで設定する必要があります。以下の定義マクロ一覧から使用する動作周波数に対応するものをプロジェクトのコンパイルオプションで定義し、設定してください。なお、BLE ソフトウェアが対応可能な、BLE MCU の動作周波数は、4MHz、8MHz、16MHz、32MHz の 4 種類となります。

BLE MCU の動作周波数として、BLE MCU 内蔵のオンチップ・オシレータの他、外部発振子の接続、外部からのクロック供給による指定も可能です。高速オンチップ・オシレータ以外の供給を指定した場合の動作周波数は 4MHz、8MHz、16MHz の 3 種類となります。

定義マクロを複数指定することは出来ません。

表 6-2　動作周波数設定用マクロ一覧

定義マクロ名	対応周波数
CLK_HOCO_4MHZ	高速オンチップ・オシレータ 4MHz 動作時
CLK_HOCO_8MHZ	高速オンチップ・オシレータ 8MHz 動作時(デフォルト)
CLK_HOCO_16MHZ	高速オンチップ・オシレータ 16MHz 動作時
CLK_HOCO_32MHZ	高速オンチップ・オシレータ 32MHz 動作時
CLK_X1_4MHZ	発振子接続 4MHz 動作時
CLK_X1_8MHZ	発振子接続 8MHz 動作時
CLK_X1_16MHZ	発振子接続 16MHz 動作時
CLK_EX_RF_4MHZ	外部クロック入力(RF 部から供給) 4MHz 動作時
CLK_EX_RF_8MHZ	外部クロック入力(RF 部から供給) 8MHz 動作時
CLK_EX_RF_16MHZ	外部クロック入力(RF 部から供給) 16MHz 動作時
CLK_EX_4MHZ	外部クロック入力(APP MCU または他から供給) 4MHz 動作時
CLK_EX_8MHZ	外部クロック入力(APP MCU または他から供給) 8MHz 動作時
CLK_EX_16MHZ	外部クロック入力(APP MCU または他から供給) 16MHz 動作時

また、BLE ソフトウェアに設定する動作周波数は、ユーザ・オプション・バイトで設定した値を同じ値としてください。ユーザ・オプション・バイトの指定の方法は、以下の方法で設定してください。詳しい内容は、各開発環境のマニュアルを参照してください。初期状態は、「高速オンチップ・オシレータ 8MHz 動作」となっています。

- CS+版(CS+ for CA,CX 及び CS+ for CC 共通)

  「プロジェクト・ツリー」の「CA78K0R（ビルド・ツール）」(CS+ for CC の場合は「CC-RL (ビルド・ツール)」)を右クリックして「プロパティ」を選択し、「リンク・オプション」タブ内の「デバイス」にて「ユーザ・オプション・バイト値」を変更してください。動作周波数に対応したユーザ・オプション・バイトの設定は表 6-3 を参考にしてください。

表 6-3　ユーザ・オプション・バイト設定

ユーザ・オプション・バイト設定			動作周波数	動作モード
000C0	000C1	000C2		
(任意)	(任意)	2B	4MHz	低電圧メインモード
		AA	8MHz	低速メインモード
		E9	16MHz	高速メインモード
		E8	32MHz	

- e² studio 版
  「プロジェクト・エクスプローラー」にて右クリックして「Renesas Tool Settings」を選択し、「C/C++ビルド」の「設定」を選択し、「ツール設定」タブの「Linker」の「デバイス」にて「ユーザ・オプション・バイト値」を変更してください。

BLE MCU と RF 部のサブシステム・クロックの供給モードについて、以下の定義マクロ一覧から使用する環境に対応するものをプロジェクトのコンパイルオプションで定義し、設定してください。
定義マクロを複数同時に指定することは出来ません。

表 6-4　サブシステム・クロック供給モード設定用マクロ一覧

定義マクロ名	対応周波数
CLK_SUB_XT1	MCU　32.768kHz　発振子接続(XT1,XT2) RF 部　16.384kHz　PCLBUZ0 から供給　(デフォルト)
CLK_SUB_EX_OT	MCU　32.768kHz　外部クロック入力(EXCLKS) RF 部　32.768kHz　RF 用スロー・クロック部内蔵発振回路使用
上記定義なし	MCU　15kHz　MCU の低速オンチップ・オシレータ使用 RF 部　32.768kHz　RF 用スロー・クロック部内蔵発振回路使用

6.1.4　MCU 部の初期化設定

RL78/G1D の MCU 部は、arch_main.c 内の plf_init 関数を実行することで初期化します。plf_init 関数では以下の設定について、ユーザによる選択が可能です。RF 用スロー・クロック部内蔵発振回路を使用しない設定ならば、MCU 部からクロック(16.384kHz)を RF 部に供給する必要があります。クロックの出力設定は plf_init 関数の引数 plf_flg にて行います。また、クロックの出力設定には表 6-5 に示すマクロを使用することが可能です。

- クロックの出力設定（PCLBUZ0 から 16.384kHz または 32.768kHz クロックの出力/未出力）

クロック出力回路(PCLBUZ0)の詳細については、RL78/G1D ユーザーズマニュアル ハードウェア編を参照してください。

表 6-5　MCU 部初期化設定マクロ

マクロ名	値	設定	内容
PLF_PCLBUZ_NONE	0x00	クロック出力設定 (いずれかを選択)	クロック(16.384kHz/32.768kHz)の出力なし
PLF_PCLBUZ_16KHZ	0x01		クロック(16.384kHz)の出力あり(デフォルト)
PLF_PCLBUZ_32KHZ	0x02		クロック(32.768kHz)の出力あり

表 6-6　MCU 部初期化関数

関数名	void plf_init(const uint8_t plf_flg);
概要	MCU 部の初期化
説明	MCU 部の高速オシレータ、汎用ポート、RF 部割り込み関連レジスタの初期化設定を行います。さらに、引数 plf_flg が PLF_PCLBUZ_16KHZ または PLF_PCLBUZ_32KHZ の設定ならば、PCLBUZ0 から 16.384kHz または 32.768kHz のクロックを出力させる設定を行います。

引数	const uint8_t plf_flg	MCU 部初期化設定 設定値は「表 6-5　MCU 部初期化設定マクロ」を参照して設定してください。
戻り値	なし	

BLE ソフトウェアでは、plf_init 関数の引数について、表 6-7 に示す設定をプロジェクトのコンパイルオプションによって選択することが可能です。これらの設定以外で使用する場合は、表 6-5 の設定を引数に指定してください。

表 6-7　スロー・クロック出力設定

コンパイルオプション		選択される引数
スロー・クロック出力設定		
	CLK_SUB_XT1（デフォルト）	PLF_PCLBUZ_16KHZ
	CLK_SUB_EX_OT	PLF_PCLBUZ_NONE
	指定なし	PLF_PCLBUZ_NONE

6.1.5　RF 部の初期化設定

RL78/G1D の RF 部は、arch_main.c 内の rf_init 関数、rwble_init 関数を実行することで初期化します。rf_init 関数では以下の設定について、ユーザによる選択が可能です。各設定は rf_init 関数の引数 rf_flg にて行います。また、各設定には表 6-8 に示すマクロを使用することが可能です。

- 外部パワーアンプの使用可否設定（TXSELH_RF，TXSELL_RF からの制御信号出力の有無）
- 内蔵 DC-DC コンバータの使用可否設定（内蔵 DC-DC コンバータの使用/未使用）
- RF 用スロー・クロック部の設定（RF 部内蔵 32kHz 使用/未使用、未使用時は MCU から EXSLK_RF へのクロック供給が必要）
- 高速クロックの出力設定（高速クロック出力なし/16MHz/8MHz/4MHz のいずれかを選択）

rwble_init 関数では以下の設定について、ユーザによる選択が可能です。

- Sleep Clock Accuracy 設定(20ppm～500ppm)

各設定の詳細については、RL78/G1D ユーザーズマニュアル ハードウェア編を参照してください。

表 6-8　RF 部初期化設定マクロ

マクロ名	値	設定	内容
RF_EXPA_OFF	0x0000	外部 PA 設定 (いずれかを選択)	外部パワーアンプ未使用(デフォルト)
RF_EXPA_ON	0x0001		外部パワーアンプ使用
RF_DCDC_ON	0x0000	DC-DC 設定 (いずれかを選択)	RF 部内蔵 DC-DC コンバータ使用(デフォルト)
RF_DCDC_OFF	0x0002		RF 部内蔵 DC-DC コンバータ未使用
RF_INT_32KHZ	0x0000	RF 用スロー・クロック設定 (いずれかを選択)	RF 部内蔵発振回路(32kHz)使用
RF_EXT_32KHZ	0x0020		EXSLK_RF 32kHz 使用
RF_EXT_16KHZ	0x0040		EXSLK_RF 16kHz 使用 (デフォルト)
RF_CLK_NONE	0x0000	高速クロック出力設定 (いずれかを選択)	RF 部高速クロックの外部出力なし(デフォルト)
RF_CLK_16MHZ	0x0300		16MHz
RF_CLK_8MHZ	0x0400		8MHz
RF_CLK_4MHZ	0x0500		4MHz

表 6-9　RF 部初期化関数

関数名	bool rf_init(const uint16_t rf_flg);	
概要	RF 部の初期化	
説明	引数 rf_flg の設定に従い、RF 部の初期化設定を行います。本関数の実行により、RF 部の動作モードは POWEROFF から STANDBY_RF まで遷移します。	
引数	const uint16_t rf_flg	RF 部初期化設定 設定値は「表 6-8　RF 部初期化設定マクロ」を参照して各設定の論理和を設定してください。
戻り値	true	初期化成功
	false	初期化失敗

表 6-10　Sleep Clock Accuracy 設定マクロ

マクロ名	値	内容
SCA_500PPM	0	500ppm
SCA_250PPM	1	250ppm
SCA_150PPM	2	150ppm
SCA_100PPM	3	100ppm
SCA_75PPM	4	75ppm
SCA_50PPM	5	50ppm(デフォルト)
SCA_30PPM	6	30ppm
SCA_20PPM	7	20ppm

表 6-11　Sleep Clock Accuracy 設定関数

関数名	void rwble_init(const struct bd_addr *bd_addr, const uint8_t sca);	
概要	Bluetooth Device アドレス、SCA 設定	
説明	BLE ソフトウェアに Bluetooth Device アドレスと､Sleep Clock Accuracy を設定します。	
引数	const struct bd_addr *bd_addr	Bluetooth Device アドレス設定 設定方法は「5.6 自 Bluetooth Device アドレスの決定」、「7.20.4.1 Bluetooth Device アドレス書き込み機能」を参照してください。
	const uint8_t sca	Sleep Clock Accuracy 設定 設定値は「表 6-10 Sleep Clock Accuracy 設定マクロ」を参照してください。
戻り値	なし	

BLE ソフトウェアでは、rf_init 関数、rwble_init 関数の引数について、表 6-12 に示す組み合わせでプロジェクトのコンパイルオプションによって選択することが可能です。これらの組み合わせ以外で使用する場合は、表 6-8、表 6-10 の設定をそれぞれの関数の引数に指定してください。

表 6-12　組み合わせ設定

コンパイルオプション		選択される引数
スロー・クロック設定		
	CLK_SUB_XT1（デフォルト）	RF_EXT_16KHZ / SCA_50PPM
	CLK_SUB_EX_OT	RF_INT_32KHZ / SCA_250PPM
	指定なし	RF_INT_32KHZ / SCA_250PPM
高速クロック出力		
	CLK_EX_RF_4MHZ	RF_CLK_4MHZ
	CLK_EX_RF_8MHZ	RF_CLK_8MHZ
	CLK_EX_RF_16MHZ	RF_CLK_16MHZ
	指定なし（デフォルト）	RF_CLK_NONE

6.1.5.1　擬似乱数のシード値設定

RF 部の初期化時に発生するアナログ情報及び顧客固有情報の BD アドレスからシード値を生成し、標準 C ライブラリ srand 関数に設定します。下記の関数を rf_init 関数実行後に実行してください。以降、標準 C ライブラリ rand 関数から擬似乱数値を取得できます。

【注】本機能は MCU 起動ごとに異なるシード値の生成を目的としますが、乱数性を保証するものではありません。

表 6-13　擬似乱数シード値設定関数

関数名	void input_rand_value(uint16_t val, uint32_t uinfo_top);	
概要	擬似乱数シード値設定	
説明	擬似乱数のためのシード値を生成し、標準 C ライブラリ srand 関数に設定します。 本関数実行後に srand 関数を実行すると、rand 関数のシードを変更することが可能です。	
引数	uint16_t val	ユーザ指定の乱数値 任意の値を設定します。シード値生成の追加要素となります。
	uint32_t uinfo_top	顧客固有情報領域の先頭アドレス
戻り値	なし	

6.1.6　シリアル通信の選択

Modem 構成時の BLE MCU とのシリアル通信において UART、CSI、IIC から接続方式を選択することができます。UART、CSI、IIC の選択は下記①の手順、接続方式の選択は下記②の手順で行います。

① UART、CSI、IIC の選択とチャネル選択

・UART を選択する場合

下記のフォルダにある uart.c、uart.h を、BLE ソフトウェアのご使用になられる開発環境プロジェクトに追加し、csi.c、csi.h、または iic_slave.c、iic_slave.h を追加している場合はプロジェクトから除外します。

※デフォルトでは本ファイルを追加しており UART を選択している状態となっています。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥uart

ファイル名：uart.c、uart.h

チャネルの選択は uart.c のチャネル設定マクロで選択することができます。

表 6-14　UART チャネル設定マクロ

マクロ名	値	内容

UART_CHANNEL	0	UART0(デフォルト)
	1	UART1

・CSI を選択する場合

下記のフォルダにある csi.c、csi.h を、BLE ソフトウェアのご使用になられる開発環境プロジェクトに追加し、uart.c、uart.h、または iic_slave.c、iic_slave.h を追加している場合はプロジェクトから除外します。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥csi
ファイル名：csi.c、csi.h

チャネルの選択は csi.c のチャネル設定マクロで選択することができます。

表 6-15　CSI チャネル設定マクロ

マクロ名	値	内容
CSI_CHANNEL	0	CSI00(デフォルト)
	1	CSI20

・IIC を選択する場合

下記のフォルダにある iic_slave.c、iic_slave.h を、BLE ソフトウェアのご使用になられる開発環境プロジェクトに追加し、uart.c、uart.h、または csi.c、csi.h を追加している場合はプロジェクトから除外します。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥iic
ファイル名：iic_slave.c、iic_slave.h

IIC スレーブアドレスは iic_slave.h のアドレス設定マクロで設定することができます。

表 6-16　IIC スレーブアドレス設定マクロ

マクロ名	範囲	デフォルト値
IIC_SLAVE_ADDRESS	0x00-0x7F	0x50

② 接続方式の選択

下記のフォルダにある serial.h で定義されるマクロの定数値を変更することで、接続方式を選択できます。上記①の手順にて UART を選択した場合は、2 線接続方式または 3 線接続方式または 2 線分岐方式を選択可能です。上記①の手順にて CSI を選択した場合は、4 線接続方式または 5 線接続方式を選択可能です。上記①の手順にて IIC を選択した場合は、3 線接続方式を選択可能です。選択する接続方式に対応する図 6-1 のマクロの定数値のみを(1)に変更し、その他の接続方式に対応するマクロの定数値を(0)に変更してください。

※デフォルトでは UART の 2 線接続方式を選択している状態となっています。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥serial
ファイル名：serial.h

表 6-17　シリアル通信の接続方式

シリアル	接続方式	有効にするマクロ
UART	2 線接続方式	SERIAL_U_2WIRE(デフォルト)
	3 線接続方式	SERIAL_U_3WIRE
	2 線分岐接続方式	SERIAL_U_DIV_2WIRE
CSI	4 線接続方式	SERIAL_C_4WIRE
	5 線接続方式	SERIAL_C_5WIRE
IIC	3 線接続方式	SERIAL_I_3WIRE

```
39:    /*
40:     * DEFINES
41:     ************************************************
42:     */
43:    #define SERIAL_U_2WIRE     (1)
44:    #define SERIAL_U_3WIRE     (0)
45:    #define SERIAL_U_DIV_2WIRE (0)
46:    #define SERIAL_C_4WIRE     (0)
47:    #define SERIAL_C_5WIRE     (0)
48:    #define SERIAL_I_3WIRE     (0)
```

図 6-1　接続方式変更サンプルコード

③　WAKEUP 信号の使用有無の設定

下記表に従って、上記②で選択した接続方式に対応する値を wakeup.c の USE_WAKEUP_SIGNAL_PORT マクロ(Modem Setting 側)に設定する必要があります。設定すべき値は下記の表をご確認ください。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥wakeup

ファイル名：wakeup.c

表 6-18　シリアル通信の接続方式

シリアル	接続方式	USE_WAKEUP_SIGNAL_PORT (Modem Setting 側)に設定する値
UART	2 線接続方式	0
	3 線接続方式	1
	2 線分岐接続方式	1
CSI	4 線接続方式	0
	5 線接続方式	1
IIC	3 線接続方式	0

```
22:    #ifndef CONFIG_EMBEDDED
23:     #define USE_WAKEUP_SIGNAL_PORT  (0) /* Modem Setting */
24:    #else
25:     #define USE_WAKEUP_SIGNAL_PORT  (0) /* Embedded Setting */
26:    #endif
```

図 6-2　WAKEUP 信号の使用有無設定サンプルコード

6.1.7　UART ボー・レートの設定

Modem 構成時の BLE MCU とのシリアル通信で UART を選択する場合、UART0 のボー・レートは、下記のソースファイルで変更可能です。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥uart
ファイル名：uart.c
関数名：void serial_init(void)

ボー・レートを変更するには、UART0 初期化関数(serial_init)で設定されている下記レジスタの設定値を変更します。シリアル・クロック選択レジスタ 0(SPS0)で分周された MCU の動作クロックが UART0 の動作クロックとなり、UART0 動作クロックと希望するボー・レートからシリアル・データ・レジスタ(SDR00、SDR01)の設定値を算出し設定することでボー・レートを変更することができます。
各レジスタの設定値は、6.1.7.1 シリアル・クロックの選択、6.1.7.2 ボー・レートの計算を参照して下さい。

表 6-19　ボー・レート設定レジスタ

レジスタ略号	レジスタ名称	備考
SPS0	シリアル・クロック選択レジスタ 0	UART0 動作クロック設定
SDR00	シリアル・データ・レジスタ 00	UART0 送信ボー・レート設定
SDR01	シリアル・データ・レジスタ 01	UART0 受信ボー・レート設定

【注】ボー・レート設定の詳細については、RL78/G1D ユーザーズマニュアル ハードウェア編を参照して下さい。

6.1.7.1　シリアル・クロックの選択

UART0 の動作クロック(f_{mck})は、シリアル・クロック選択レジスタ 0(SPS0)の PRS00[3:0]で設定します。

表 6-20　シリアル・クロック選択レジスタ設定値

PRS 003	PRS 002	PRS 001	PRS 000	動作クロック(CK00)の選択				
					f_{clk} = 4MHz	f_{clk} = 8MHz	f_{clk} = 16MHz	f_{clk} = 32MHz
0	0	0	0	f_{clk}	4MHz	8MHz	16MHz	32MHz
0	0	0	1	$f_{clk}/2$	2MHz	4MHz	8MHz	16MHz
0	0	1	0	$f_{clk}/2^2$	1MHz	2MHz	4MHz	8MHz
0	0	1	1	$f_{clk}/2^3$	500kHz	1MHz	2MHz	4MHz
0	1	0	0	$f_{clk}/2^4$	250kHz	500kHz	1MHz	2MHz
0	1	0	1	$f_{clk}/2^5$	125kHz	250kHz	500kHz	1MHz
0	1	1	0	$f_{clk}/2^6$	62.5kHz	125kHz	250kHz	500kHz

PRS 003	PRS 002	PRS 001	PRS 000		動作クロック(CK00)の選択			
					f_{clk} = 4MHz	f_{clk} = 8MHz	f_{clk} = 16MHz	f_{clk} = 32MHz
0	1	1	1	$f_{clk}/2^7$	31.3kHz	62.5kHz	125kHz	250kHz
1	0	0	0	$f_{clk}/2^8$	15.6kHz	31.3kHz	62.5kHz	125kHz
1	0	0	1	$f_{clk}/2^9$	7.81kHz	15.6kHz	31.3kHz	62.5kHz
1	0	1	0	$f_{clk}/2^{10}$	3.91kHz	7.81kHz	15.6kHz	31.3kHz
1	0	1	1	$f_{clk}/2^{11}$	1.95kHz	3.91kHz	7.81kHz	15.6kHz
1	1	0	0	$f_{clk}/2^{12}$	977Hz	1.95kHz	3.91kHz	7.81kHz
1	1	0	1	$f_{clk}/2^{13}$	488Hz	977Hz	1.95kHz	3.91kHz
1	1	1	0	$f_{clk}/2^{14}$	244Hz	488Hz	977Hz	1.95kHz
1	1	1	1	$f_{clk}/2^{15}$	122Hz	244Hz	488Hz	977Hz
上記以外				設定禁止				

6.1.7.2　ボー・レートの計算

ボー・レートを設定するシリアル・データ・レジスタ(SDR00、SDR01)ビット[15:9]への設定値は下記の計算式で求めることができます。

SDRmn[15:9] ＝ (UART0 の動作クロック(f_{mck}) ÷ 2 ÷ ボー・レート) － 1

4800bps を設定する場合、f_{clk}=8MHz であり SPS0=0x0003 の時、SDR00、SDR01 は 0xCE00 となります。
250kbps を設定する場合、f_{clk}=8MHz であり SPS0=0x0002 の時、SDR00、SDR01 は 0x0600 となります。

6.1.7.3　ボー・レート変更サンプルコード

ソースコード内にはボー・レートを 250kbps に変更するサンプルコードが含まれています。Modem 構成の初期状態では 4800bps に設定されていますが、ソースコード内 387 行目"#if"文の定数値を(0)にすることで 250kbps にボー・レートを変更することが可能です。

```
387:      #if (1)
388:        #ifndef CONFIG_EMBEDDED
389:        /* MCK = fclk/n = 1MHz */
390:        write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) | UART_VAL_SPS_1MHZ)));
391:
392:        /* baudrate 4800bps(when MCK = 1MHz) */
393:        write_sfrp(UART_TXD_SDR, (uint16_t)0xCE00U);
394:        write_sfrp(UART_RXD_SDR, (uint16_t)0xCE00U);
395:        #else /*CONFIG_EMBEDDED*/
396:        /* MCK = fclk/n = 2MHz */
397:        write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) | UART_VAL_SPS_2MHZ)));
398:
399:        /* baudrate 250000bps(when MCK = 2MHz) */
400:        write_sfrp(UART_TXD_SDR, (uint16_t)0x0600U);
401:        write_sfrp(UART_RXD_SDR, (uint16_t)0x0600U);
402:        #endif /*CONFIG_EMBEDDED*/
403:      #else
404:        /* MCK = fclk/n = 2MHz */
405:        write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) | UART_VAL_SPS_2MHZ)));
406:        /* baudrate 250000bps(when MCK = 2MHz) */
407:        write_sfrp(UART_TXD_SDR, (uint16_t)0x0600U);
408:        write_sfrp(UART_RXD_SDR, (uint16_t)0x0600U);
409:      #endif
```

図 6-3　ボー・レート変更サンプルコード

また２線接続方式を選択した場合に限り、BLE MCU の低消費電流を実現するための機能である Sleep 機能を有効または無効とするための STOP 許可フラグを設定する必要があります。

ボー・レートを 4800bps に設定する場合には、416 行目"#if"文の定数値を(1)にすることで STOP 許可フラグを true に設定します。ボー・レートを 4800bps より大きい値に設定する場合には、416 行目"#if"文の定数値を(0)にすることで STOP 許可フラグを false に設定します。

```
414:    /* set stop permission */
415:    #if SERIAL_U_2WIRE
416:    #if (1)
417:      #ifndef CONFIG_EMBEDDED
418:      /* if baudrate is 4800bps, set enable */
419:      stop_flg = true;
420:      #else /*CONFIG_EMBEDDED*/
421:      /* if baudrate is over than 4800bps, set disable */
422:      stop_flg = false;
423:      #endif /*CONFIG_EMBEDDED*/
424:    #else
425:    /* if baudrate is over than 4800bps, set disable */
426:    stop_flg = false;
427:    #endif
428:    #else
```

図 6-4　ボー・レート変更サンプルコード

6.1.8　CSI ボー・レートの設定

Modem 構成時の BLE MCU とのシリアル通信で CSI を選択する場合、CSI のボー・レートは APP MCU が供給するクロックにより決定するため、設定の変更は不要です。

6.1.9　IIC 転送クロックの設定

Modem 構成時の BLE MCU とのシリアル通信で IIC を選択する場合、IIC の転送クロックは APP MCU が供給するクロックにより決定するため、設定の変更は不要です。

6.1.10　サブ・クロック安定までの待ち時間

サブ・クロックの安定までの時間をシステムにあった時間に変更してください。下記ファイル内の設定マクロ値を修正してください。なお、初期状態は 1 秒間の待ち時間となっています。

フォルダ：Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥driver¥plf

ファイル名： plf.c

表 6-21　サブ・クロック安定待ち設定マクロ

マクロ名	デフォルト値
WAITTIME_XT1	977 ※タイマ・クロック 977Hz に対するカウント値を設定してください。

6.1.11　プロファイル・サービスの設定

プロファイルやサービスのユーザ設定可能項目は、以下のソースファイルの変数または、定義マクロにて変更可能です。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78
ファイル名：prf_config.c, prf_config.h

フォルダ：¥ Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78
ファイル名：prf_sel.h

ユーザでのプロファイル作成方法については 7.4 を参照ください。

【注】　上記ファイルに定義される既存プロファイルの GATT データベース構造は変更できません。順序の変更や、要素の追加・削除は行わないでください。オプションの特性を非公開にしたい場合は RBLE_GATT_PERM_HIDE パーミッションを使用してください。

6.1.11.1　プロファイルの有効・無効設定

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、本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本節を廃止しました。

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。

6.1.11.2　GAP パラメータの設定

GAP の各種モード･プロシージャに関連するパラメータは､表 6-22 の定義マクロの値により設定可能です。ソースファイル中に宣言しています。

表 6-22　GAP パラメータ設定マクロ

定義マクロ名	概要	備考
GAP_DEV_SEARCH_TIME	デバイスサーチ時間	デバイスサーチ(Limited / General Discovery プロシージャ)パラメータ
GAP_DEV_SEARCH_SCAN_INTV	スキャンインターバル	
GAP_DEV_SEARCH_SCAN_WINDOW	スキャンウインドウ	
GAP_LIM_ADV_TIMEOUT	発見可能(Advertising)時間	Limited Discoverable モードパラメータ
GAP_SCAN_FAST_INTV	スキャンインターバル	Auto / Selective コネクションパラメータ
GAP_SCAN_FAST_WINDOW	スキャンウインドウ	
GAP_INIT_CONN_MIN_INTV	最小コネクションインターバル	
GAP_INIT_CONN_MAX_INTV	最大コネクションインターバル	
GAP_CONN_SLAVE_LATENCY	スレーブレイテンシー	
GAP_DEV_SUPERVISION_TIMEOUT	スーパービジョンタイムアウト	
GAP_RESOLVABLE_PRIVATE_ADDR_INTV	プライベートアドレス変更間隔	製品の用途に合わせて任意の値を設定して下さい。

6.1.11.3　GAP 特性の設定

以下に示す GAP の特性値を設定することができます。

- Device Name
- Appearance
- Peripheral Preferred Connection Parameters

デバイスを識別するためのユーザフレンドリーな名前を示す Device Name 特性値の初期値は、表 6-23 の定義マクロの値により設定可能です。

表 6-23　Device Name 設定マクロ

定義マクロ名	概要	備考
GAP_DEV_NAME	デバイス名称初期値	UTF-8 文字列にてデバイス名称を設定してください。

【注】 本定義値よりも CodeFlash 最終ブロックに書き込まれたデバイス名称(5.5 参照)が優先されます。また、BLE ソフトウェア起動後は API にてデバイス名称を変更可能です。

デバイスの外観を示す Appearance 特性値の初期値は、表 6-24 の定義マクロの値により設定可能です。

表 6-24　Appearance 設定マクロ

定義マクロ名	概要	備考
GAP_APPEARANCE_CHAR_VAL	デバイスの外観設定初期値	下記を参照し、製品に合わせて設定して下さい。 https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.characteristic.gap.appearance.xml

GAP Appearance 特性の詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part C Section 12.2 を参照ください。

ペリフェラルデバイスの望む接続パラメータである Peripheral Preferred Connection Parameters 特性値は、表 6-25 の定義マクロの値により設定可能です。ソースファイル中に宣言しています。

表 6-25　Peripheral Preferred Connection Parameters 設定マクロ

定義マクロ名	概要	備考
GAP_PPCP_CONN_INTV_MAX	最大コネクションインターバル	製品の用途に合わせて任意の値を設定して下さい。
GAP_PPCP_CONN_INTV_MIN	最小コネクションインターバル	
GAP_PPCP_SLAVE_LATENCY	スレーブレイテンシー	
GAP_PPCP_STO_MULT	スーパービジョンタイムアウト	

GAP Peripheral Preferred Connection Parameters 特性の詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part C Section 12.3 を参照ください。

6.1.11.4　GATT 特性の設定

GATT データベース構造の変更をクライアントに通知するための Service Changed 特性値のハンドル情報は、表 6-26 の定義マクロの値により設定可能です。

表 6-26　Service Changed 設定マクロ

定義マクロ名	概要	備考
GATT_SERVICE_CHANGED_START_HDL	GATT データベース変更開始ハンドル	デフォルトでは、Service Changed 特性は非公開に設定しています。
GATT_SERVICE_CHANGED_END_HDL	GATT データベース変更終了ハンドル	

【注】 GATT データベース構造を変更する可能性がある場合、Service Changed 特性を公開してください。また、GATT データベースを変更し、クライアントから通知を求められている場合は rBLE API を使用して Indication を行ってください。

GATT Service Changed 特性の詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part G Section 7.1 を参照ください。

6.1.11.5　Health Thermometer Service 特性の設定（廃止）

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、本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本節を廃止しました。

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。

6.1.11.6　Blood Pressure Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.7　HID Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.8　Battery Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.9　Device Information Service の製品情報設定（廃止）

※「6.1.11.5」を参照。

6.1.11.10　Heart Rate Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.11　Cycling Speed and Cadence Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.12　Cycling Power Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.13　Glucose Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.14　Current Time Service 特性の設定（廃止）

※「6.1.11.5」を参照。

6.1.11.15 Running Speed and Cadence Service 特性の設定 (廃止)

※「6.1.11.5」を参照。

6.1.11.16 Alert Notification Service 特性の設定 (廃止)

※「6.1.11.5」を参照。

6.1.11.17 Location and Navigation Service 特性の設定 (廃止)

※「6.1.11.5」を参照。

6.2　プロジェクトのビルド

実行ファイル作成のためのプロジェクトビルド手順は下記のとおりです。

1. 6.1を参考にご使用の環境に合わせて設定可能パラメータを変更します。
2. 開発環境と使用するBLEソフトウェアの構成に合うように、以下のフォルダから表 6-27に示すプロジェクトファイルまたはワークスペースファイルをダブルクリックし、それぞれの環境でファイルを開きます。
¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥tools¥project¥
3. CS+の場合は、[ビルド]メニューより[ビルドプロジェクト]を実行します。また、e² studioの場合は、[プロジェクト]メニューから[プロジェクトのビルド]を実行します。
4. ビルドが終了すると、表 6-27の実行ファイル生成フォルダに実行ファイルが出力されます。

表 6-27 開発環境とビルド環境の対応

開発環境	構成	プロジェクトファイル／ ワークスペースファイル	実行ファイル 生成フォルダ
CS+ for CA,CX (CA78K0R)	Modem	CubeSuite¥BLE_Modem¥BLE_Modem.mtpj	rBLE_emb¥DefaultBuild
	Embedded	CubeSuite¥BLE_Embedded¥BLE_Embedded.mtpj	BLE_Emb¥DefaultBuild
CS+ for CC (CC-RL)	Modem	CS_CCRL¥BLE_Modem¥BLE_Modem.mtpj	rBLE_Mdm¥DefaultBuild
	Embedded	CS_CCRL¥BLE_Embedded¥BLE_Embedded.mtpj	rBLE_Emb¥DefaultBuild
e² studio (CC-RL)	Modem	e2studio¥BLE_Modem¥rBLE_Mdm¥ (*1)	rBLE_Mdm¥DefaultBuild
	Embedded	e2studio¥BLE_Embedded¥rBLE_Emb¥ (*1)	rBLE_Emb¥DefaultBuild

*1：e² studio のプロジェクトファイルは、e² studio のワークスペースにインポートする必要があります。

6.3　補足事項

BLE ソフトウェアは、複数のライブラリを提供するため、ライブラリの組み合わせを間違えると、ビルドが成功しても正常に動作しません。このことを避けるため、BLE ソフトウェア内でライブラリの組み合わせをチェックしてユーザに通知する機能を持っており、GAP リセット完了時の表 6-28 のステータスで判別可能です。

また、BLE ソフトウェアにプロトタイプのプロファイルが追加された場合などに、評価用バージョンのライブラリを提供するケースがあります。この場合にも、GAP リセット完了時の表 6-28 のステータスで判別可能です。

【注】評価用バージョンのライブラリは、お客様の製品へ適用せず、正式リリースをお待ちください。

表 6-28 ライブラリ管理ステータス

ステータス	値	概要
RBLE_VERSION_FAIL	0xF7	ライブラリ組み合わせエラー
RBLE_TEST_VERSION	0xF8	BLE ソフトはテストバージョン

また、BLE ソフトウェアは、表 6-29 に示すサイズのスタックを必要とします。

表 6-29 BLE ソフトウェアスタックサイズ

必要スタックサイズ	824 バイト

上記サイズには API 呼び出し元や、BLE ソフトウェアが呼び出すアプリケーションのコールバック関数のスタックサイズは含まれません。このため、アプリケーションによっては上記以上のスタックが必要となる場合があります。アプリケーション開発時には、十分なスタックサイズを確保してください。

7.　機能説明

本章では、BLE ソフトウェアの機能について説明します。

7.1　Controller スタック

Controller スタックは Host Controller Interface(HCI)、Link Layer(LL)を含み、Host スタックの要求に応じて RF/BB を制御し、Advertising、Scanning 等のパケット処理を行います。

7.1.1　Advertising

Advertising は接続を確立するため、または Scanning デバイスに対してユーザデータを周期的に提供するために使用されます。Advertising では Advertise チャネル上にパケットを送信し、Scanning デバイスからのレスポンスを受信、これに応答します。この状態のデバイスを Advertiser と呼びます。

Advertiser が、他のデバイスから接続要求を受け取り接続した場合は、スレーブデバイスとして動作します。

Advertising には 4 つのタイプがあり、タイプにより許可されるレスポンスの関係について示します。

表 7-1　Advertising イベントタイプ

Advertising イベントタイプ		許可されるレスポンス	
		SCAN_REQ	CONNECT_REQ
Connectable Undirected Event	ADV_IND	YES	YES
Connectable Directed Event	ADV_DIRECT_IND	NO	YES
Non-connectable Undirected Event	ADV_NONCONN_IND	NO	NO
Scannable Undirected Event	ADV_SCAN_IND	YES	NO

SCAN_REQ：追加の情報を要求します。

CONNECT_REQ：接続の確立を開始します。

Advertising は決められた 3 つの Advertise チャネル(37ch、38ch、39ch)のうち任意のチャネルを使用します。Advertising データの送信を行う期間を Advertising イベントと呼び、各 Advertising イベント間の時間は *T_advEvent* として次の通り計算されます。

T_advEvent = *advInterval* + *advDelay*

- *advInterval*：20ms～10.24s の範囲で 0.625ms の整数倍
  (Scannable Undirected イベントタイプまたは Non-connectable Undirected イベントタイプの場合、100ms 以下にはならない。Connectable undirected イベントタイプの場合 20ms 以上)
- *advDelay*：0ms～10ms の範囲を持つ疑似ランダム値

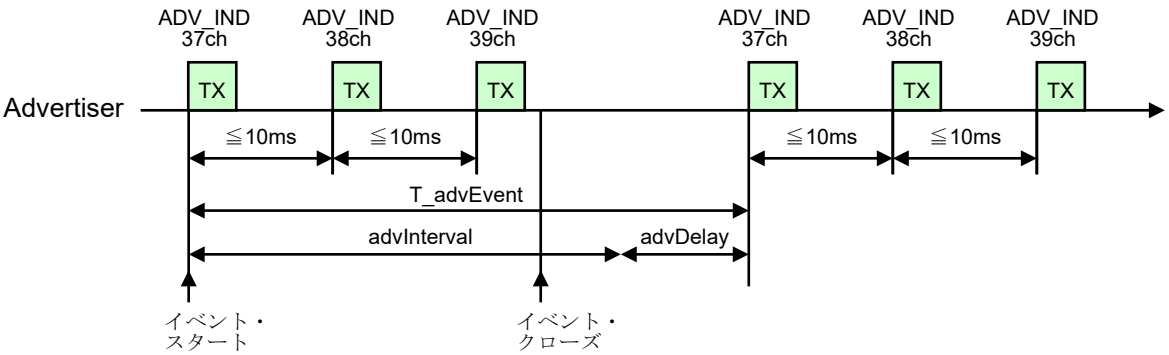


図 7-1　Advertising イベント(ADV_IND)

Advertising の詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.4.2 を参照ください。

7.1.2 Scanning

Scanning は Advertiser からブロードキャストされるデータを受信するために使用されます。Advertising チャネル上で Advertiser からのパケットを待つデバイスを Scanner と呼び、Scanning には Passive Scanning と Active Scanning の 2 タイプがあります。

7.1.2.1 Passive Scanning

Passive Scanning では、Scanner はパケットを受信するのみで、どのようなパケットも送りません。

7.1.2.2 Active Scanning

Active Scanning では、Scanner は Advertiser からの Advertising パケットを待ち、Advertising イベントタイプに応じて応答することができます。ADV_IND パケットまたは ADV_SCAN_IND パケットを受信した場合、Scanner は SCAN_REQ パケットを Advertiser に送信することで追加の情報を取得することが可能です。

Active Scanning 中に 38ch にて ADV_IND パケットを受信した Scanner の動作例を示します。

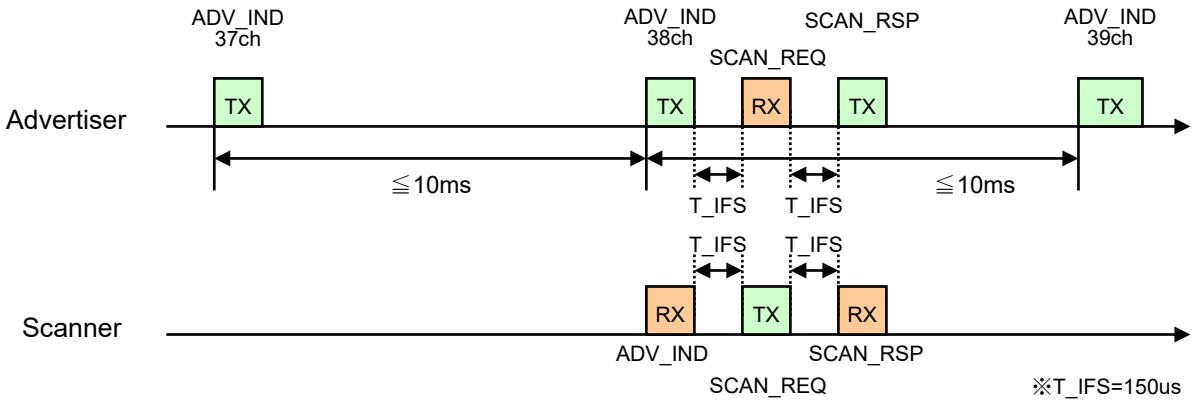


図 7-2 Active Scanning(ADV_IND)

Scanning の詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.4.3 を参照ください。

7.1.3 Initiating

Initiating は他のデバイスとの接続を確立します。他のデバイスと接続するために、Advertising チャネル上で Advertising パケットを待つデバイスを Initiator と呼びます。ADV_IND パケット、または ADV_DIRECT_IND パケットを受信した Initiator は CONNECT_REQ パケットを送った後に Initiating を終了しマスタとして動作します。

CONNECT_REQ パケットを送信後の最初のパケットは、1.25ms + *transmitWindowOffset* 後に開始され *transmitWindowSize* 以内に送信されます。

- *transmitWindowOffet*：0ms～*connInterval* の範囲で 1.25ms の倍数
- *transmitWindowSize*：1.25ms～10ms の範囲で 1.25ms の倍数

接続中はマスタとスレーブが *connInterval* によって交互にパケットを送受信します。

- *connInterval*：7.5ms から 4.0s の範囲で 1.25ms の倍数

Initiator が Advertiser からの ADV_IND パケットを受信した後、マスタデバイスとなるまでの動作例を示します。

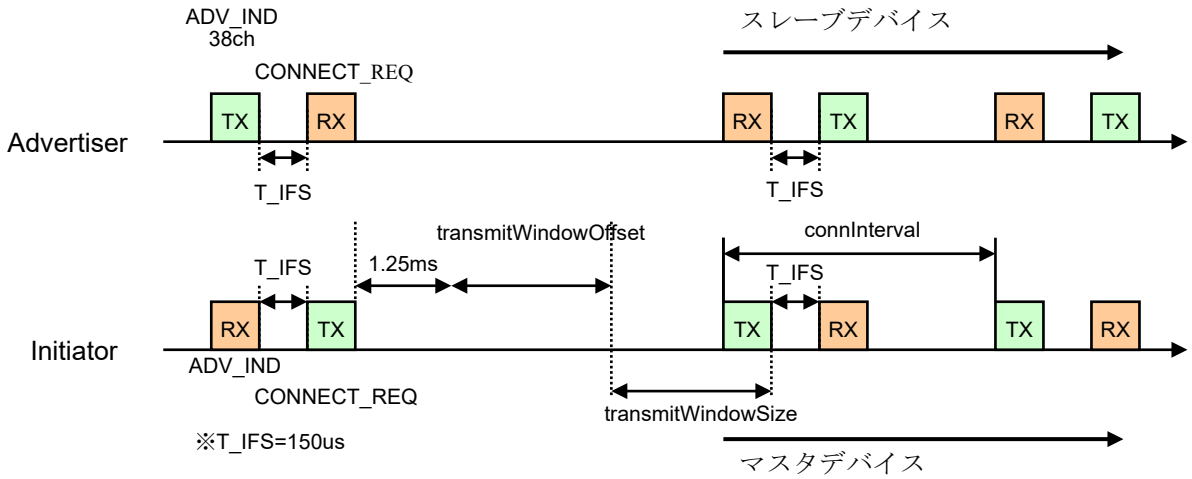


図 7-3　コネクションセットアップ

Initiating の詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.4.4 を参照ください。

7.1.4　White List

White List は Advertise パケット、Scan パケット、および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ために使用され、デバイスアドレスとアドレスタイプ(Public アドレス、または Random アドレス)を含みます。

White List は Controller スタックの Link Layer ブロックで管理され、リセット状態において White List は空になっており、アプリケーションによって設定されます。

White List の詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.3.1 を参照ください。

7.1.4.1　Advertising フィルタポリシー

Advertising フィルタポリシーはどのように Scan と接続要求を処理するのかを決定します。Connectable directed advertising を使用している時、Advertising フィルタポリシーは無視されます。それ以外はアプリケーションによって設定される下記の Advertising フィルタポリシーの 1 つを使用します。

- White List を使用せず、全てのデバイスからの Scan と接続要求を処理(初期状態)
- White List に登録されたデバイスからのみ Scan と接続要求を処理
- 全てのデバイスからの Scan 要求を処理し、White List にあるデバイスからのみ接続要求を処理
- 全てのデバイスからの接続要求を処理し、White List にあるデバイスからのみ Scan 要求を処理

Advertising フィルタポリシーの詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.3.2 を参照ください。

7.1.4.2　Scanner フィルタポリシー

Scanner フィルタポリシーは、どのように Advertising パケットを処理するのかを決定します。Host スタック

によって設定される下記の Scanner フィルタポリシーの 1 つを使用します。

- White List を使用せず、全てのデバイスからの Advertising パケットを処理(初期状態)
- White List に登録されたデバイスからのみ Advertising パケットを処理

Scanner フィルタポリシーの詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.3.3 を参照ください。

7.1.4.3 Initiator フィルタポリシー

Initiator フィルタポリシーは、どのように Advertising パケットを処理するのかを決定します。アプリケーションによって設定される下記の Initiator フィルタポリシーの 1 つを使用します。

- White List に登録された全てのデバイスからの Connectable Advertising パケットを処理
- White List を無視し、アプリケーションによって指定されたシングルデバイスからの Connectable Advertising パケットを処理

Initiator フィルタポリシーの詳細は Bluetooth Core Specification v4.2[Vol. 6], Part B Section 4.3.4 を参照ください。

7.2 Generic Access Profile

Generic Access Profile(以降、GAP)は、周辺デバイスの検索や、ピアデバイスとの接続・切断等のリンク管理、セキュリティ要件に応じた各種手続きを行います。

7.2.1 GAP ロール

BLE ソフトウェアは、表 7-2 に示す GAP で規定される 4 つのロール(役割)を全てサポートしています。

表 7-2 GAP ロール

GAP ロール	概要
Broadcaster	Advertising イベントを送信する Link Layer では Advertiser と呼ばれる
Observer	Advertising イベントを受信する Link Layer では Scanner と呼ばれる
Central	物理リンクの確立を行う Link Layer では Master と呼ばれる
Peripheral	物理リンクの確立を受け入れる Link Layer では Slave と呼ばれる

GAP ロールの詳細は Bluetooth Core Specification v4.2[Vol. 3], Part C Section 9 を参照ください。

7.2.2 GAP モードおよびプロシージャ

本項では、BLE ソフトウェアのサポートする GAP モードおよび GAP プロシージャについて説明します。各モードおよびプロシージャでは表 7-3 に示す Advertising イベントタイプを使い分けます。

表 7-3 Advertising タイプ

Advertising イベントタイプ	概要
Connectable Undirected	CONNECT_REQ または SCAN_REQ に応答可能(接続可能)
Connectable Directed	指定デバイスとのみ接続可能
Scannable Undirected	SCAN_REQ に応答可能(接続不可)
Non-connectable Undirected	Advertiser からの情報送信のみ(接続不可)

7.2.2.1 Broadcast モードおよび Observation プロシージャ

Broadcast モードおよび Observation プロシージャは、2 デバイス間で接続の確立なく通信を行うことを可能にします。

Broadcast モードにおけるデバイスは Broadcaster と呼ばれ、Advertising イベントにてデータをブロードキャストします。送信したデータに対するリモートデバイスからの応答はないため、通信の信頼性は保証されません。送信可能な Advertising イベントタイプは、Non-connectable Undirected イベントまたは Scannable Undirected イベントです。Advertising データの AD タイプフラグは、LE General Discoverable Mode・LE Limited Discoverable Mode 共に 0 に設定しなければなりません。

Observation プロシージャを実行するデバイスは Observer と呼ばれ、Advertising イベントを受信します。

rBLE API は Broadcast モードおよび Observation プロシージャを実行するための API を用意しています。Broadcast モード時の Advertising データはユーザより任意に設定可能です。

Broadcast モードおよび Observation プロシージャの詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part C Section 9.1 を参照ください。

7.2.2.2 Discovery モードおよびプロシージャ

Discovery モードおよびプロシージャにて、周辺デバイスの検索が可能になります。Discovery モードは、Advertising データを送信することでリモートデバイスから発見されることが可能なモードで、Peripheral デバイスが実行します。Discovery プロシージャは、Scan により Advertising データを受信し、周辺の Peripheral デバイスを発見するプロシージャで、Central デバイスが実行します。

表 7-4 は各 Discovery モードと送信可能な Advertising イベントタイプおよび Advertising データの AD タイプフラグ設定値の関係を示しています。

表 7-4 Discovery モード

Discovery モード	送信可能な Advertising イベントタイプ	Flags AD Type		概要
		LE General Discoverable Mode	LE Limited Discoverable Mode	
Non-Discoverable	• Non-connectable Undirected • Scannable Undirected • 送信しない	0	0	General または Limited Discovery プロシージャを実行するデバイスから発見されない
Limited Discoverable	• Non-connectable Undirected • Scannable Undirected • Connectable Undirected	0	1	General または Limited Discovery プロシージャを実行するデバイスから限られた期間発見されることが可能
General Discoverable	• Non-connectable Undirected • Scannable Undirected • Connectable Undirected	1	0	General Discovery をプロシージャ実行するデバイスから発見されることが可能

表 7-5 に各 Discovery プロシージャの概要を示します。

表 7-5 Discovery プロシージャ

Discovery プロシージャ	概要
Limited Discovery	Limited Discoverable モードのデバイスのみ発見することが可能
General Discovery	General または Limited Discovery モードのデバイスを発見することが可能
Name Discovery	GATT を使用して、接続可能なリモートデバイスのデバイス名称を取得

rBLE API は各 Discovery モード、周辺デバイス検索およびデバイス名称取得を実行するための API を用意しています。周辺デバイス検索では、既知デバイスのみを検索することも可能です。また、Discovery モード時の Advertising データは、ユーザより任意に設定可能です。実行するモードに適した Advertising データの AD タイプフラグを設定する必要があります。

各モードおよびプロシージャの詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part C Section 9.2 を参照くだ

さい。

7.2.2.3　Connection モードおよびプロシージャ

Connection モードおよびプロシージャにて、他デバイスとの接続を確立することが可能となります。Connection モードは、Advertising データを送信することでリモートデバイスから接続されることが可能なモードで、Peripheral デバイスが実行します。Connection プロシージャは、Peripheral デバイスとの接続を確立するするプロシージャで、Central デバイスが実行します (リンクを切断する Terminate connection プロシージャについては Central、Peripheral どちらからでも実行可能です)。

表 7-6 は各 Connection モードと送信可能な Advertising イベントタイプの関係を示しています。

表 7-6　Connection モード

Connection モード	送信可能な Advertising イベントタイプ	概要
Non-connectable	• Non-connectable Undirected • Scannable Undirected	接続の確立を許可しない
Directed connectable	• Connectable Directed	auto connection establishment プロシージャまたは general connection establishment プロシージャを実行する既知のデバイスからのみ接続可能
Undirected connectable	• Connectable Undirected	auto connection establishment プロシージャまたは general connection establishment プロシージャを実行するデバイスから接続可能

表 7-7 に各 Connection プロシージャの概要を示します。

表 7-7　Connection プロシージャ

Connection プロシージャ	概要
Auto connection establishment	イニシエータの White List を使用し、Directed connectable モードまたは Undirected connectable モードのデバイスと自動的に接続する
General connection establishment	Directed connectable モードまたは Undirected connectable モードの既知デバイスと接続する
Selective connection establishment	White List のデバイスと Host が選択した接続コンフィギュレーションパラメータで接続する
Direct connection establishment	Host が選択した接続コンフィギュレーションパラメータで 1 台の既知デバイスと接続する
Connection parameter update	確立しているコネクションのコネクションパラメータを変更する
Terminate connection	ピアデバイスとのコネクションを終了する

rBLE API は各 Connection モードおよび各 Connection プロシージャを実行するための API を用意しています。Connection モード時の Advertising データは、ユーザより任意に設定可能です。また、Connection モードは Discovery モードと組み合わせての実行が可能です。

各モードおよびプロシージャの詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part C Section 9.3 を参照くだ

さい。

7.2.2.4 Bonding モードおよびプロシージャ

Bonding は 2 台の接続したデバイスが信頼できる関係を築くために、セキュリティおよび固有の情報を交換し、格納します。セキュリティおよび固有の情報はボンディング情報と呼ばれます。デバイスがボンディング情報を格納する時、「ボンディングした」と表現します。

Bonding には表 7-8 に示すモードがあります。

表 7-8 Bonding モード

Bonding モード	概要
Non-Bondable	ピアデバイスとのボンディングを許可しない
Bondable	Bondable モードのピアデバイスとのボンディング可能

Bonding プロシージャは、未ボンドのデバイスがボンディングを必要とするサービスへアクセスする際に実行されます。ボンディングにはペアリングが使用されます。

rBLE API はボンディングの実行およびボンディングの要求に応答するための API を用意しています。

ボンディングの詳細は Bluetooth Core Specification v4.2[Vol. 3], Part C Section 9.4 を参照ください。

7.2.3 セキュリティについて

本項では、GAP にて規定される BLE のセキュリティについて説明します。

7.2.3.1 セキュリティモード

デバイス、サービスまたはサービス要求のセキュリティ要件は、「セキュリティモード」および「セキュリティレベル」という言葉で表現されます。デバイスやサービスごとにセキュリティ要件が存在するかもしれませんが、2 つのデバイス間の物理リンクは、一つのセキュリティモードにて動作します。各セキュリティ要件を満たすためには、ペアリングが必要になります。ペアリングには MITM(中間者攻撃)から保護される Authenticated ペアリングと、MITM から保護されない Unauthenticated ペアリングがあります。それらのペアリングと、暗号化やデータ署名の使用・不使用によりセキュリティモード・レベルは分類されます。表 7-9 に BLE 規格で規定されるセキュリティモードおよびセキュリティレベルを示します。

表 7-9 セキュリティモードおよびレベル

セキュリティモード	セキュリティレベル	概要
LE Security Mode 1	1	セキュリティ無し(認証なし、暗号化なし)
	2	Unauthenticated ペアリングによる暗号化
	3	Authenticated ペアリングによる暗号化
	4	Authenticated LE Secure Connections ペアリングによる暗号化
LE Security Mode 2	1	Unauthenticated ペアリングによるデータ署名
	2	Authenticated ペアリングによるデータ署名

【注】 BLE ソフトウェアは LE セキュリティモード 1 レベル 4(LE Secure Connections)には未対応です。

LE セキュリティモード 1 レベル 2 は、LE セキュリティモード 1 レベル 1 のセキュリティ要件を満たします。

LE セキュリティモード 1 レベル 3 は、LE セキュリティモード 1 レベル 2 のセキュリティ要件を満たします。また、LE セキュリティモード 1 レベル 3 は、LE セキュリティモード 2 のセキュリティ要件を満たします。

ある物理リンクにて、LE セキュリティモード 1 と LE セキュリティモード 2 レベル 2 が必要とされる時、LE セキュリティモード 1 レベル 3 が適用されます。
ある物理リンクにて、LE セキュリティモード 1 レベル 3 と LE セキュリティモード 2 が必要とされる時、LE セキュリティモード 1 レベル 3 が適用されます。
ある物理リンクにて、LE セキュリティモード 1 レベル 2 と LE セキュリティモード 2 レベル 1 が必要とされる時、LE セキュリティモード 1 レベル 2 が適用されます。

rBLE API はセキュリティモード設定の API を用意しています。また、各プロファイルにてセキュリティモードを設定することが可能です。
セキュリティモードの詳細は Bluetooth Core Specification v4.2[Vol. 3], Part C Section 10.2 を参照ください。

7.2.3.2 Authentication プロシージャ

Authentication(認証)プロシージャは、デバイスがリモートデバイスにサービス要求を開始するとき、またはデバイスがリモートデバイスからのサービス要求を受けた時に、セキュリティ要件を確立するためのものです。認証は LE セキュリティモード 1 および 2 をカバーします。Authentication プロシージャは接続確立後に開始されます。
セキュリティが不要または、セキュリティ要件を既に満たしている場合、サービス要求は継続され、そうでない場合はペアリングが必要になります。
rBLE API ではボンディングを実行することでペアリングを実行することが可能です。
Authentication プロシージャの詳細は Bluetooth Core Specification v4.2[Vol. 3], Part C Section 10.3 を参照ください。

7.2.3.3 データ署名

データ署名は、暗号化されない接続において、2 つのデバイス間の認証されたデータを転送するために使用されます。データ署名は、早いコネクションセットアップおよび早いデータ転送を必要とするサービスによって使用されます。
サービスが LE セキュリティモード 2 を必要とする場合、データ署名が使用されます。
BLE ソフトウェアでは、ATT の Signed Write コマンド(Bluetooth Core Specification v4.2[Vol. 3], Part F Section 3.4.5.4)を送信する際にデータ署名が使用されます。rBLE API にて事前にデータ署名用のキーCSRK を設定しておく必要があります。また、受信した署名付きデータの正当性の確認は BLE ソフトウェア内部で行います。この時必要となるリモートデバイスの CSRK を rBLE API にてアプリケーションに要求します。なお、リモートデバイスより配布された CSRK の管理および自デバイスの CSRK の生成はアプリケーションで行う必要があります。
データ署名の詳細は Bluetooth Core Specification v4.2[Vol. 3], Part C Section 10.4 を参照ください。

7.2.3.4 プライバシーフィーチャー

プライバシー機能は、ランダムアドレスを使用することで、悪意のあるデバイスからの追跡や特定されることを防ぐことができます。
rBLE API はプライバシーフィーチャーを有効にする API を用意しており、事前に IRK を設定しておくことで、ロールに応じたランダムアドレスの生成を行います。なお、IRK の生成・管理はアプリケーションで行う必要があります。
プライバシーフィーチャーの詳細は Bluetooth Core Specification v4.1[Vol. 3], Part C Section 10.7 を参照ください。

7.2.4 Bluetooth デバイスアドレス

Bluetooth デバイスを識別するために、48bit の Bluetooth デバイスアドレス(BD アドレス)と呼ばれる識別子がそれぞれのデバイスに必要となります。BLE 規格では、Bluetooth デバイスアドレスは大きく分けてパブリックアドレスとランダムアドレスの２つが規定されています。

7.2.4.1 パブリックアドレス

パブリックアドレスは、IEEE802-2001 仕様に準拠したアドレス方式によって定義され、24bit の OUI (Organizationally Unique Identifier)を含むアドレスです。それぞれのデバイスに一意的に与えられ、デバイスのライフタイムの間、変化することはありません。

パブリックアドレスを使用するためには IEEE Registration Authority への登録が必要となります。

7.2.4.2 ランダムアドレス

ランダムアドレスのサブタイプを表 7-10 に示します。

表 7-10 ランダムアドレス

ランダムアドレスの種類		概要
スタティックアドレス		IEEE Registration Authority への登録を必要としない場合、パブリックアドレスの代用として使用する事ができます。電源投入時に毎回生成することも、ライフタイムの間、同一のアドレスを使用する事も可能です。ただし、電源断まではそのスタティックアドレスを変更することはできません。
プライベートアドレス		プライベートアドレスはプライバシー保護のために使用され、定期的にアドレスを変更することで、悪意のあるデバイスからの追跡・特定を困難にします。接続中や Advertising 中に変更することも可能です。プライベートアドレスには、以下のサブタイプがあります。
	Non-resolvable プライベートアドレス	Bluetooth v4.1 以降では、接続にこのアドレスを用いることはできません。したがって、BLE ソフトウェアではサポートしません。
	Resolvable プライベートアドレス	Identity Resolving Key(IRK)と乱数から生成されるアドレスで、IRK を共有するデバイスのみが、Resolvable Private Address Resolution プロシージャによってデバイスを特定することができます。

【注】 BLE 規格では、ランダムアドレスの衝突回避や検出をサポートしていないため、わずかながら他のデバイスとの重複の可能性があります。また、プライベートアドレスを使用した場合、アドレス解決処理が必要となるため再接続に時間を要します。

- Resolvable プライベートアドレスの解決
  Resolvable Private Address Resolution プロシージャを使用して、Host が IRK を持つ全てのピアデバイスの Resolvable プライベートアドレスを解決することが可能です。Resolvable プライベートアドレスが解決されるならば、Host はピアデバイスにこのアドレスを結びつけることができます。Host が複数の IRK を格納しているのであれば、アドレス解決が成功するまで、格納されている IRK についてこのプロシージャを繰り返します。

BLE ソフトウェアでは、Non-resolvable 以外のアドレスタイプを利用可能です。プライベートアドレスは、アプリケーションから渡された IRK を使用して、プライバシーフィーチャーの有効化時に自動生成します。Resolvable プライベートアドレスの解決は、BLE ソフトウェア内部で行います。この時必要となるリモートデ

バイスの IRK を rBLE API にてアプリケーションに要求します。なお、リモートデバイスの IRK についてはアプリケーションで管理する必要があります。

ランダムアドレスの詳細は Bluetooth Core Specification v4.2[Vol. 3], Part C Section 10.8 を参照ください。

7.2.5　Advertising および Scan レスポンスデータフォーマット

Advertising および Scan Response データは図 7-4 に示すフォーマットで作成します。

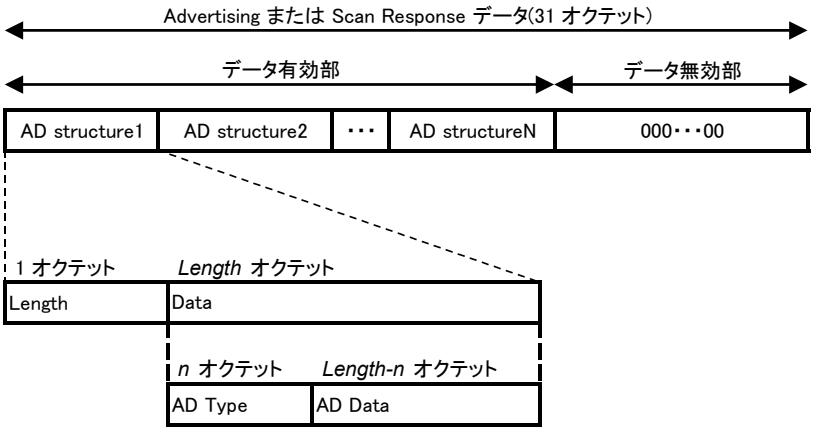


図 7-4　Advertising および Scan Response データフォーマット

Advertising および Scan Response データには以下の特徴があります。

- データは総計 31 オクテット
- 複数の AD structure から構成される
- AD structure は 1 オクテットの Length 情報と、Length オクテットのデータから構成
- Length オクテットのデータは n オクテットの AD Type と Length - n オクテットの AD Data から構成
- 全ての AD structure の合計サイズが 31 オクテットに満たない場合は 0 パディング
- 全て 0 のデータは、Advertising または Scan Response の早期終了を可能にするためのみに使用される
- Advertising または Scan Response データの有効部分のみが Air 上に送信される
- Advertising または Scan Response データは、advertising events にて送信される
- Advertising データは、ADV_IND・ADV_NONCONN_IND および ADV_SCAN_IND パケットの AdvData フィールドに配置される。
- Scan Response データは、SCAN_RSP パケットの ScanRspData フィールドにて送られる

AD structure に使用可能な AD Type の定義と AD Data フォーマットを表 7-11 に示します。

表 7-11　AD Type の定義と AD Data フォーマット

<table>
<tr><th>AD Type</th><th>AD Type Value</th><th>AD Data 概要</th></tr>
<tr><td>Flags</td><td>0x01</td><td>Flags は下記ビットから構成。
<table>
<tr><th>Bit</th><th>概要</th></tr>
<tr><td>0</td><td>LE Limited Discoverable Mode</td></tr>
<tr><td>1</td><td>LE General Discoverable Mode</td></tr>
<tr><td>2</td><td>BR/EDR 未サポート</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>3</td><td>LE と BR/EDR 同時動作可能(Controller)</td></tr><tr><td>4</td><td>LE と BR/EDR 同時動作可能(Host)</td></tr></table>• Flags AD Type は Scan Response データに含めてはならない
• Advertising データには 1 つの Flags AD Type しか含めない</td></tr>
<tr><td rowspan="4">Service</td><td>0x02</td><td>利用可能な 16bit サービス UUID</td></tr>
<tr><td>0x03</td><td>利用可能な 16bit サービス UUID(完全なリスト)</td></tr>
<tr><td>0x06</td><td>利用可能な 128bit サービス UUID</td></tr>
<tr><td>0x07</td><td>利用可能な 128bit サービス UUID(完全なリスト)</td></tr>
<tr><td rowspan="2">Local Name</td><td>0x08</td><td>短いローカルデバイス名称</td></tr>
<tr><td>0x09</td><td>完全なローカルデバイス名称</td></tr>
<tr><td>TX Power Level</td><td>0x0A</td><td>Advertising パケットの送信パワーレベル(1Byte)
0xXX : -127 ～ +127dBm</td></tr>
<tr><td rowspan="3">OOB Data</td><td>0x0D</td><td>Class of device (3Byte)</td></tr>
<tr><td>0x0E</td><td>Simple Pairing Hash C (16Byte)</td></tr>
<tr><td>0x0F</td><td>Simple Pairing Randomizer R (16Byte)</td></tr>
<tr><td>TK Value</td><td>0x10</td><td>ペアリングで使用される TK(Temporary Key) (128bit)</td></tr>
<tr><td>OOB Flags</td><td>0x11</td><td>OOB Flags は下記ビットから構成。<table><tr><th>Bit</th><th>概要</th></tr><tr><td>0</td><td>OOB Flags フィールド
(0 = OOB データ無し, 1 = OOB データ有り)</td></tr><tr><td>1</td><td>Host が LE をサポートする</td></tr><tr><td>2</td><td>LE と BR/EDR 同時動作可能(Host)</td></tr><tr><td>3</td><td>アドレスタイプ
(0 = パブリックアドレス, 1 = ランダムアドレス)</td></tr></table></td></tr>
<tr><td>Slave Connection Interval Range</td><td>0x12</td><td>全ての論理リンクの為に、Peripheral の要求する接続インターバルを含む。
Central は Peripheral の本 AD Type のデータを使用すべきである。
最初の 2Byte は最小コネクションインターバル
N = 0x0006～0x0C80 (Time=N*1.25msec)
次の 2Byte は最大コネクションインターバル
N = 0x0006～0x0C80 (Time=N*1.25msec)
0xFFFF は don’t care</td></tr>
<tr><td rowspan="2">Service Solicitation</td><td>0x14</td><td>16bit サービス UUID リスト</td></tr>
<tr><td>0x15</td><td>128bit サービス UUID リスト</td></tr>
<tr><td>Service Data</td><td>0x16</td><td>16bit サービス UUID に続く、追加のサービスデータ</td></tr>
<tr><td>Manufacturer Specific Data</td><td>0xFF</td><td>メーカ独自のデータ
最初の 2Byte にはカンパニーID が含まれる</td></tr>
</table>

表 7-12 に Advertising データの設定例を示します。例では AD Type に Flags、完全なデバイス名称および 16bit サービス UUID を設定しています。

表 7-12 Advertising データの例

値	意味
0x02	この AD structure のデータ長(2 オクテット)
0x01	AD type = Flags
0x06	LE Limited Discoverable Bit および BR/EDR 未サポート Bit をセット
0x08	この AD structure のデータ長(8 オクテット)
0x09	AD type = 完全なローカルデバイス名称

値	意味
0x52	‘R’
0x65	‘e’
0x6E	‘n’
0x65	‘e’
0x73	‘s’
0x61	‘a’
0x73	‘s’
0x07	この AD structure のデータ長(7 オクテット)
0x03	AD type = 利用可能な 16bit サービス UUID(完全なリスト)
0x02	Immediate Alert サービス(UUID：0x1802)
0x18	
0x03	Link Loss サービス(UUID：0x1803)
0x18	
0x04	Tx Power サービス(UUID：0x1804)
0x18	

Advertising および Scan Response データはユーザにより任意に設定可能です。上記フォーマットに従い利用シーンに応じて適切に設定していただく必要があります。

Advertising および Scan Response データフォーマットの詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part C Section 11 を参照ください。AD Type の詳細は、Supplement to the Bluetooth Core Specification v5, Part A を参照ください。

7.3 Security Manager

Security Manager(以下、SM)は、ペアリング、暗号化、プライベートアドレス解決およびデータ署名など、BLE のセキュアな通信に関する部分を担当します。

リンクの暗号化やデータ署名等で使用されるキーを生成するためにペアリングが行われます。ペアリングを開始するデバイスをイニシエータ(Initiator)、それに応答するデバイスをレスポンダ(Responder)と呼びます。ペアリングは下記および図 7-5 に示すフェーズにて実行されます。

- フェーズ 1：ペアリングフィーチャーの交換
- フェーズ 2：STK の生成。STK 生成方法はフェーズ 1 にて交換した情報に基づく
- フェーズ 3：生成されたキーの配布。フェーズ 2 で生成したキーを使用し暗号化されたリンクで行う

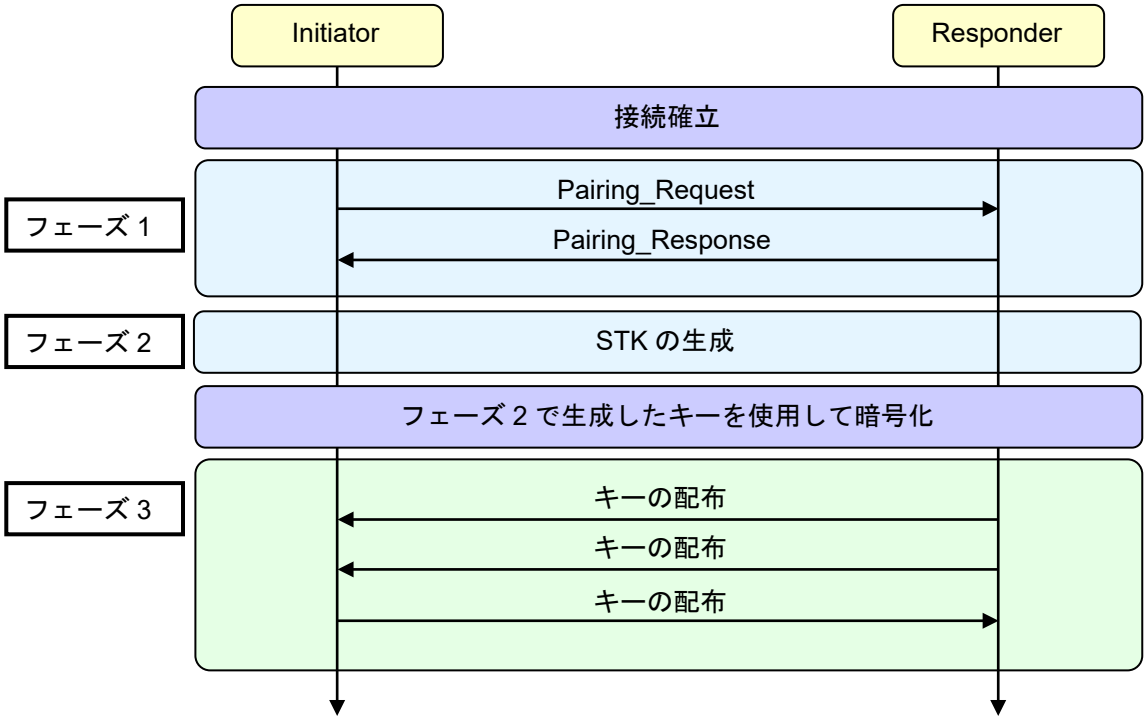


図 7-5 LE ペアリングフェーズ

次に、ペアリング、暗号化、プライベートアドレス解決およびデータ署名などで扱われるキーについて下表にまとめます。

表 7-13 キーの定義

キータイプ	概要	生成
IRK (Identity Resolving Key)	128bit ランダムアドレスの生成・解決に使用される	アプリケーションで生成する
CSRK (Connection Signature Resolving Key)	128bit 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	アプリケーションで生成する
LTK (Long Term Key)	128bit (同意されたキーサイズに応じて部分的に使用) 暗号化のためのセッションキーを生成する為に使用される	アプリケーションで生成する

キータイプ	概要	生成
EDIV (Encrypted Diversifier)	16bit LTK を識別するために使用される。LTK が配布されるたびに EDIV は生成される	アプリケーションで生成する
Rand (Random Number)	64bit LTK を識別するために使用される。LTK が配布されるたびに Rand は生成される	アプリケーションで生成する
STK (Short Term Key)	128bit TK を使用しペアリングフェーズ 2 にて生成される。 フェーズ 2 後のリンクの暗号化に使用される	アプリケーションより渡された TK を使用し、BLE ソフトウェアで生成する
TK (Temporary Key)	128bit STK 生成のためにペアリングフェーズ 2 で使用される	アプリケーションで生成する

各キーの生成や詳細は Bluetooth Core Specification v4.2[Vol. 3], Part H Section 2.4.1 および 2.4.2 を参照ください。

7.3.1 ペアリングフィーチャーの交換

ペアリングフェーズ 1 にて、イニシエータとレスポンダはペアリングフィーチャーの交換を行います。交換されるフィーチャーのフィールドを以下に記します。ここで交換される情報を基に、フェーズ 2 で使用されるペアリングメソッドが決定されます。

- IO Capability
  本フィールドは、デバイスの持つ入力装置・出力装置の能力について示します。Input Capability(表 7-14)と Output Capability(表 7-15)を組み合わせたものが IO Capability(表 7-16)となります。

表 7-14 Input Capability

Input Capability	概要
No input	デバイスには'Yes'または'No'を示すことが可能な入力装置がない
Yes / No	少なくとも'Yes'または'No'を示すことが可能なボタンが 2 つ、または'Yes'または'No'を示すことが可能なメカニズムがある(例 : 'Yes'はタイムリミット内にボタン押下、'No'はタイムアウトによって示される)
Keyboard	0～9 までの数値入力と、確認が可能なキーボードを持つ。また、少なくとも’Yes’または’No’を示すことが可能なボタンが 2 つ、または’Yes’または’No’を示すことが可能なメカニズムがある

表 7-15 Output Capability

Output Capability	概要
No output	6 桁の数値を表示または通知可能な出力装置がない
Numeric output	6 桁の数値を表示または通知可能な出力装置を持つ

Input Capability(表 7-14)と Output Capability(表 7-15)は表 7-16 に示す一つの IO Capability にマップされ、ペアリングフィーチャーの交換に使用されます。

表 7-16 IO Capability マッピング

Input \ Output	No output	Numeric output
No input	NoInputNoOutput	DisplayOnly
Yes / No	NoInputNoOutput[1]	DisplayYesNo
Keyboard	KeyboardOnly	KeyboardDisplay

【注】1. Yes/No と No Output の組み合わせは NoInputNoOutput とみなします。

- OOB 認証データの有効性
  Bluetooth の通信以外の方法を用いてやり取りされたリモートデバイスとの認証に必要となるデータを OOB(Out Of Band)データと呼びます。本フィールドは、リモートデバイスとの認証に必要な OOB 認証データの有無を示します。

- 認証要件
  本フィールドは、認証における以下のセキュリティプロパティを示します。
  - MITM(中間者)攻撃からの保護が必要／不要
  - Bonding をする／しない

- 暗号化キーサイズ
  本フィールドは、リンクの暗号化に使用するキーサイズを示します。暗号化の際には両デバイスが示したキーサイズの小さいサイズを使用します。サイズは 7Byte(56bit)から 16Byte(128bit)まで 1Byte 単位で指定可能です。

- キー配布
  本フィールドは、ペアリングのイニシエータおよびレスポンダがフェーズ 3 のキー配布にて配布を要求するキーを示します。配布を要求するキーは下記より複数選択可能です。
  - EncKey(LTK, EDIV, および Rand)
  - IdKey(IRK)
  - Sign(CSRK)

rBLE API はボンディングの実行または応答 API にて上述のペアリングフィーチャーを任意に設定可能です。自デバイスの機能や用途に応じて適切なペアリングフィーチャーを設定ください。

各フィーチャーの詳細は Bluetooth Core Specification v4.2Vol. 3], Part H Section 2.3.1～2.3.4 を参照ください。

7.3.2 STK の生成

フェーズ 2 では、フェーズ 1 にて互いのデバイスが交換した情報を使用し、ペアリングの方法(STK の生成方法)を決定します。ペアリングにより、STK 生成に必要となる TK が決定されます。BLE のペアリング方法の種類と特徴を以下に纏めます。

- Just Works
  ペアリング中の盗聴や中間者攻撃から保護されません。ペアリング中に盗聴や中間者攻撃がなければ、それ以降の接続は暗号化によりセキュリティが確保されます。
  ペアリングの際にユーザアクションを必要としません。

両デバイスで使用される TK は 0x00000000000000000000000000000000 です。

- Passkey Entry
  双方のデバイスで同じ 6 桁の数値を入力または、一方のデバイスで 6 桁の数値(乱数)を表示し、もう一方のデバイスでは他方に表示された 6 桁の数値を入力します。
  その 6 桁の数値(10 進数：000,000～999,999)を TK として使用します。
  中間者攻撃から保護されます(中間者攻撃の成功確率は 0.000001)。TK の範囲が限られているため、ペアリング中の盗聴からの保護は限定的です。ペアリング中に盗聴がなければ、それ以降の接続は暗号化によりセキュリティが確保されます。

- Out of Band
  ペアリング前にやり取りされた 128bit の乱数を TK として使用します。TK のやり取りに使用された方法が中間者攻撃に耐力があるのであれば OOB によるペアリングは中間者攻撃から保護されます。
  TK の値は 128bit 以内であれば制限が無いため、Just Works や Passkey Entry よりセキュアだと言えます。
  しかし、互いのデバイスが TK のやり取り可能なインタフェースを持つ必要があります。

双方のデバイスが OOB データを所有する場合は、OOB によるペアリングが行われます。双方のデバイスが中間者攻撃からの保護を必要としない場合は、Just Works によるペアリングが行われます。それ以外のペアリング方法は、フェーズ 1 で交換された情報の互いの IO Capability から決定されます。ペアリングの結果生成されるキーが認証要件を満たさない場合、ペアリングは失敗します。

表 7-17 にイニシエータとレスポンダの IO Capability から決定されるペアリング方法、Passkey Entry の役割およびペアリングで生成されるキーについて示します。

表 7-17 IO Capability によるペアリング方法のマッピング

Initiator / Responder	DisplayOnly	DisplayYesNo	KeyboardOnly	NoInput NoOutput	Keyboard Display
DisplayOnly	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry イニシエータが入力、レスポンダが表示 Authenticated	Just Works Unauthenticated	Passkey Entry イニシエータが入力、レスポンダが表示 Authenticated
DisplayYesNo	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry イニシエータが入力、レスポンダが表示 Authenticated	Just Works Unauthenticated	Passkey Entry イニシエータが入力、レスポンダが表示 Authenticated
KeyboardOnly	Passkey Entry イニシエータが表示、レスポンダが入力 Authenticated	Passkey Entry イニシエータが表示、レスポンダが入力 Authenticated	Passkey Entry イニシエータ、レスポンダ共に入力 Authenticated	Just Works Unauthenticated	Passkey Entry イニシエータが表示、レスポンダが入力 Authenticated
NoInputNoOutput	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
KeyboardDisplay	Passkey Entry イニシエータが表示、レスポンダが入力 Authenticated	Passkey Entry イニシエータが表示、レスポンダが入力 Authenticated	Passkey Entry イニシエータが入力、レスポンダが表示 Authenticated	Just Works Unauthenticated	Passkey Entry イニシエータが表示、レスポンダが入力 Authenticated

Authenticated：中間者攻撃から保護されるキーが生成されます

Unauthenticated：中間者攻撃から保護されないキーが生成されます

BLE ソフトウェアではアプリケーション、およびリモートデバイスから渡されたペアリングフィーチャーより、ペアリングメソッドを決定し STK の生成を行います。Passkey Entry および OOB によるペアリングの場合、STK 生成に必要となる TK をアプリケーションに要求します。

各ペアリングメソッドの詳細は Bluetooth Core Specification v4.2[Vol. 3], Part H Section 2.3.5 を参照ください。

7.3.3 キーの配布

フェーズ 3 では必要に応じてキーの配布が行われます。キーの配布方法はフェーズ 2 のペアリング方法に関係なく同じです。

スレーブからマスタへ配布可能なキーは下記のとおりです。

- LTK, EDIV, および Rand
- IRK
- CSRK

マスタからスレーブへ配布可能なキーは下記のとおりです。

- LTK, EDIV, および Rand

- IRK
- CSRK

全てのキーを配布する前にフェーズ 2 で生成された STK を使用してリンクを暗号化しなければなりません(STK を使用したリンクの暗号化は、BLE ソフトウェアが自動的に行います)。

BLE ソフトウェアでは、アプリケーションおよびリモートデバイスから渡されたペアリングフィーチャーを基にキーの配布を行います。IRK または CSRK の配布を行う場合、事前にアプリケーションより設定されたキーを用いて自動的に配布します。LTK の配布を行う場合は、アプリケーションに対し LTK の要求を行い、アプリケーションより渡された LTK を配布します。また、リモートデバイスより配布されたキーはすべてアプリケーションへ通知します。

ペアリング後の LTK を使用したリンクの暗号化はアプリケーションより実行ください。

【注】キーの管理(キーの記憶、デバイスとの結びつけ)は BLE ソフトウェアでは行いません。アプリケーションにて実装ください。

7.4　Generic Attribute Profile

Generic Attribute Profile(以降、GATT)は、Attribute Protocol(ATT)を使用したサービスフレームワークを規定します。このフレームワークはサービスやサービスの特性のフォーマットや手続きを定義します。

GATT にはサーバとクライアントのロールがあります。サーバは図 7-6 に示すようにサービスでグループ化された特性(Characteristic)と呼ばれるデータをクライアントに公開します。公開されるサービスや特性は UUID と呼ばれる識別子で識別されます。サーバ上では、データは ATT の基礎となるアトリビュートハンドルにて管理されます。

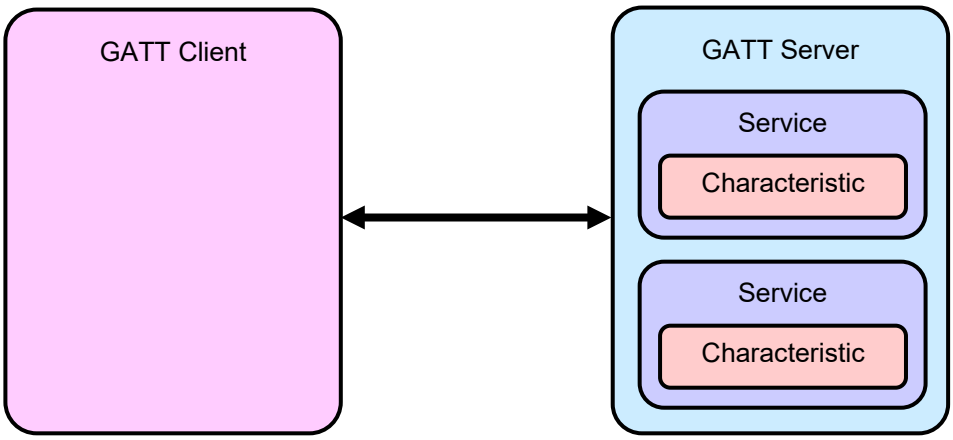


図 7-6　Generic Attribute Profile

GATT サーバで公開される特性には下表に示すプロパティが設定されます。設定されるプロパティは、各サービスにて規定されます。

表 7-18　特性プロパティ

プロパティ	値	概要
Broadcast	0x01 (RBLE_GATT_CHAR_PROP_BCAST)	特性コンフィギュレーション記述子を使用して、サーバが特性値をブロードキャストする
Read	0x02 (RBLE_GATT_CHAR_PROP_RD)	クライアントから特性値の読み出しが可能
Write Without Response	0x04 (RBLE_GATT_CHAR_PROP_WR_NO_RESP)	クライアントから特性値の書き込みが可能 (書き込みに対するサーバのレスポンスなし)
Write	0x08 (RBLE_GATT_CHAR_PROP_WR)	クライアントから特性値の書き込みが可能 (書き込みに対するサーバのレスポンスあり)
Notify	0x10 (RBLE_GATT_CHAR_PROP_NTF)	サーバからクライアントへ特性値を通知する (クライアントからサーバへの受信確認なし)
Indicate	0x20 (RBLE_GATT_CHAR_PROP_IND)	サーバからクライアントへ特性値を表示する (クライアントからサーバへの受信確認あり)

プロパティ	値	概要
Authenticated Signed Writes	0x40 (RBLE_GATT_CHAR_PROP_AUTH)	クライアントから特性値の署名付き書き込みが可能
Extended Properties	0x80 (RBLE_GATT_CHAR_PROP_EXT_PROP)	拡張プロパティ

GATT ではサーバ上で公開されるデータのやり取りを行うために、以下に示す手続きが定義されています。

- クライアントからサーバの公開するサービスの検索
- クライアントからサーバの公開する特性の検索
- クライアントからサーバの公開する特性記述子の検索
- クライアントからサーバの公開する特性値の読み出し(Read)
- クライアントからサーバの公開する特性値の書き込み(Write)
- クライアントからサーバの公開する特性コンフィギュレーション記述子の読み出し(Read)
- クライアントからサーバの公開する特性コンフィギュレーション記述子の書き込み(Write)
- サーバからクライアントへ特性値の表示(Indication)(クライアントからサーバへの受信確認有り)
- サーバからクライアントへ特性値の通知(Notification)

rBLE API では、上記の手続きを実現するための API を用意しています。この API を使用してユーザで新規にプロファイルを作成することが可能です。

また、各プロファイルでは、サービスや特性の検索は自動で行い、特性値の読み出し・書き込みを行うための API を、それぞれのプロファイルごとに用意しています。各プロファイルは GATT をベースにしており、上述の仕組みを用いて機能を実現しています。

GATT の詳細は Bluetooth Core Specification v4.2[Vol. 3], Part G を参照ください。

7.4.1 GATT データベース

GATT サーバ上で公開されるデータを、GATT データベースと呼びます。各プロファイルは、GATT データベースのデータを交換することで、ユースケースを実現しています。GATT データベースに登録するデータは、各サービスにて規定されます。

7.4.1.1 データベース構造

データベースは、プロファイルで使用される「サービス」や「特性」などの「エレメント」から構成されます。すべての「エレメント」は「アトリビュート」に含まれます。「アトリビュート」は、プロファイル・サービスのデータを運ぶためのコンテナとして、ATT にて使用されます。「アトリビュート」は「アトリビュートハンドル」で管理されます。

GATT の上位に位置するプロファイルは、ユースケースを実現するために、一つ以上の「サービス」を必要とします。「サービス」は「特性」や他のサービスへの参照により構成されます。各特性は、「特性値」または「特性記述子」などを含んでおり、それらにはサービスのデータが格納されています。

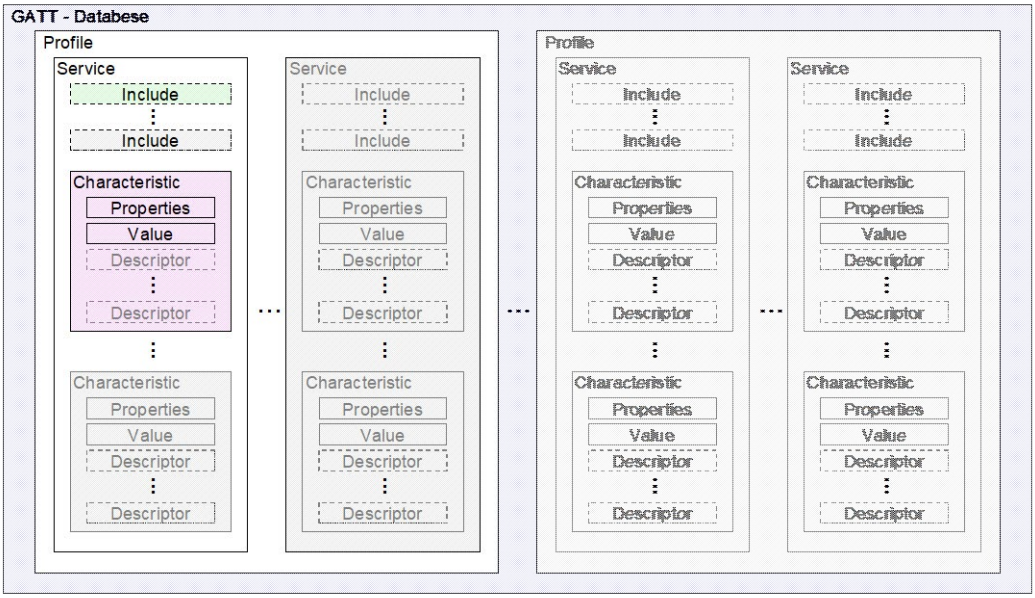


図 7-7　GATT データベース構造

BLE ソフトウェアでは、下記ソースファイルの変数にて GATT データベースを構築しています。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78

ファイル名：prf_config_host.c, prf_config.c

変数名：struct atts_desc atts_desc_list_host [ ], struct atts_desc atts_desc_list_prf [ ]

GATT データベースは RL78/G1D の RAM 上(struct atts_desc *atts_desc_list)に展開され、ビルド時のデータベース構造は動作中に変更できません。

変数 atts_desc_list の各配列要素が「アトリビュート」に相当します。BLE ソフトウェアでは、配列の要素番号を「アトリビュートハンドル」として管理しています。

各アトリビュートは以下の項目から構成されます。

表 7-19　アトリビュート構成要素

メンバ変数名	概要
type	アトリビュートタイプ UUID
maxlength	アトリビュート値の最大長
length	アトリビュート値の現在長
taskid	上位　6bit：アトリビュートの属するプロファイルタスク ID 下位 10bit：アトリビュートを識別するためのインデックス
perm	アトリビュートのパーミッション
*value	アトリビュート値格納先へのポインタ

7.4.1.2　　アトリビュートタイプ

アトリビュートは、図 7-8 に示すとおりサービス定義・インクルード定義・特性定義から構成されます。

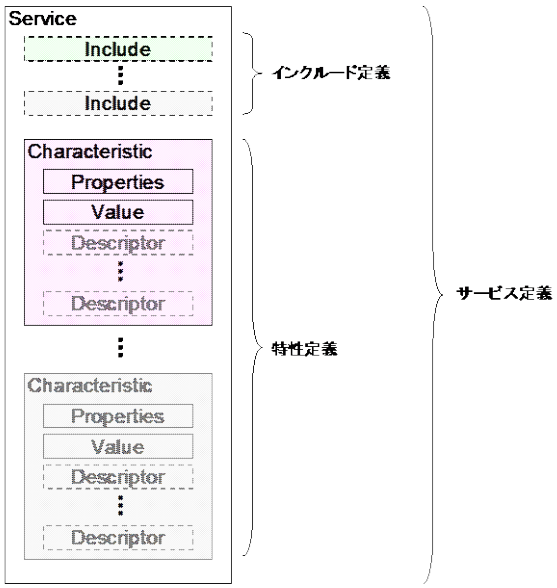


図 7-8　アトリビュート構成

それぞれのアトリビュートには表 7-20 に示すパーミッションが設定されており、読み書きやセキュリティ要件などで、GATT クライアントからのアクセスを制限することができます。

表 7-20　アトリビュートパーミッション

パーミッション値	概要
RBLE_GATT_PERM_NONE	許可なし
RBLE_GATT_PERM_RD	読み出し可
RBLE_GATT_PERM_RD_UNAUTH	読み出しには Unauthenticated ペアリングが必要
RBLE_GATT_PERM_RD_AUTH	読み出しには Authenticated ペアリングが必要
RBLE_GATT_PERM_RD_AUTZ	読み出しには Authorization が必要
RBLE_GATT_PERM_WR	書き込み可
RBLE_GATT_PERM_WR_UNAUTH	書き込みには Unauthenticated ペアリングが必要
RBLE_GATT_PERM_WR_AUTH	書き込みには Authenticated ペアリングが必要
RBLE_GATT_PERM_WR_AUTZ	書き込みには Authorization が必要
RBLE_GATT_PERM_NI	サーバからの通知・表示可
RBLE_GATT_PERM_NI_UNAUTH	通知・表示には Unauthenticated ペアリングが必要
RBLE_GATT_PERM_NI_AUTH	通知・表示には Authenticated ペアリングが必要
RBLE_GATT_PERM_NI_AUTZ	通知・表示には Authorization が必要
RBLE_GATT_PERM_EKS	十分なキーサイズでの暗号化が必要
RBLE_GATT_PERM_HIDE	公開しない
RBLE_GATT_PERM_ENC	暗号化が必要
RBLE_GATT_PERM_NOTIFY_COMP_EN	Notification の送信完了を通知

(1)　サービス定義

サービス定義は、サービス宣言から始まり、インクルード定義や特性定義を含みます。サービス定義の終わりは、次のサービス宣言の前か、アトリビュートハンドルが最大に達した場合です。サービス定義内に配置されるインクルード定義や特性定義は、そのサービスの一部であるとみなします。

下表に、サービス宣言のアトリビュート構成を示します。

表 7-21 サービス宣言

アトリビュートタイプ UUID	アトリビュート値	パーミッション
0x2800：Primary Service (RBLE_DECL_PRIMARY_SERVICE) または 0x2801：Secondary Service (RBLE_DECL_SECONDARY_SERVICE)	16bit または 128bit サービス UUID	Read Only (RBLE_GATT_PERM_RD)

サービス定義、宣言の詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part G Section 3.1 を参照ください。

(2) インクルード定義

インクルード定義は、他のサービスを参照する場合に定義し、一つのインクルード宣言のみを含みます。表 7-22 に、インクルード宣言のアトリビュート構成を示します。

表 7-22 インクルード宣言

アトリビュートタイプ UUID	アトリビュート値	パーミッション
0x2802：Include (RBLE_DECL_INCLUDE)	• Included Service アトリビュートハンドル • グループ終了ハンドル • サービス UUID	Read Only (RBLE_GATT_PERM_RD)

インクルード定義、宣言の詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part G Section 3.2 を参照ください。

(3) 特性定義

特性定義は、特性宣言から始まり、特性値宣言や特性記述子宣言を含みます。特性定義の終わりは、次の特性宣言または次のサービス宣言の前か、アトリビュートハンドルが最大に達した場合です。それぞれの宣言は個別のアトリビュートに含まれます。

特性定義は、特性宣言と特性値宣言の 2 つを必要とします。特性値宣言は、特性宣言につづいて配置することが規定されています。オプションの特性記述子宣言は、特性値宣言の後に配置されます。表 7-23 に、特性宣言のアトリビュート構成を示します。

表 7-23 特性宣言

アトリビュートタイプ UUID	アトリビュート値	パーミッション
0x2803：Characteristic (RBLE_DECL_CHARACTERISTIC)	• 特性プロパティ(上位プロファイルにより規定。表 7-18 参照。) • 特性値ハンドル • 特性 UUID	Read Only (RBLE_GATT_PERM_RD)

特性宣言につづく、特性値宣言のアトリビュート構成を表 7-24 に示します。すべての特性定義には、特性値宣言が含まれます。

表 7-24 特性値宣言

アトリビュートタイプ UUID	アトリビュート値	パーミッション
特性 UUID	特性値	上位プロファイルにより規定または実装依存

特性記述子宣言は、特性値に関連する情報を含めるために定義されます。GATT では、表 7-25 に示す上位層で使用され得る標準的な特性記述子を定義しています。

表 7-25　特性記述子

特性記述子名	概要
特性拡張プロパティ	特性の拡張プロパティを表す
特性ユーザ説明	ユーザによる特性値の説明文
クライアント特性コンフィギュレーション	特性がクライアントによってどのように構成されるかを定義
サーバ特性コンフィギュレーション	特性がサーバによってどのように構成されるかを定義
特性プレゼンテーションフォーマット	特性値のフォーマットを定義
特性集合体フォーマット	プレゼンテーションフォーマットのリスト

特性記述子宣言のアトリビュート構成を表 7-26 に示します。

表 7-26　特性記述子宣言

アトリビュートタイプ UUID	アトリビュート値	パーミッション
0x2900：Characteristic Extended Properties (RBLE_DESC_CHAR_EXT_PROPERTIES)	拡張プロパティビット	Read Only (RBLE_GATT_PERM_RD)
0x2901：Characteristic User Description (RBLE_DESC_CHAR_USER_DESCRIPTION)	特性の説明 (UTF-8)	上位プロファイルにより規定または実装依存
0x2902：Client Characteristic Configuration (RBLE_DESC_CLIENT_CHAR_CONF)	特性コンフィギュレーションビット	Read (RBLE_GATT_PERM_RD) Write(RBLE_GATT_PERM_WR) 認証・承認の要否は上位依存
0x2903：Server Characteristic Configuration (RBLE_DESC_SERVER_CHAR_CONF)	特性コンフィギュレーションビット	
0x2904：Characteristic Format (RBLE_DESC_CHAR_PRESENTATION_FMT)	• フォーマット • 指数 • 単位 • 名前空間 • 説明	Read Only (RBLE_GATT_PERM_RD)
0x2905：Characteristic Aggregate Format (RBLE_DESC_CHAR_AGGREGATE_FMT)	プレゼンテーションフォーマット宣言のハンドルリスト	Read Only (RBLE_GATT_PERM_RD)

特性定義の詳細は、Bluetooth Core Specification v4.2[Vol. 3], Part G Section 3.3 を参照ください。

7.4.1.3　データベース構築例

GATT データベース配列 atts_desc_list_host[ ] (prf_config_host.c)の、第 1～第 3 配列要素は、表 7-27 に示す GAP サービスのデバイス名称特性を構築しています。データベース配列では、仕様で規定される内容に加え、アトリビュート値の現在長・最大長や、タスク ID などの情報を付加しています。

表 7-27　GAP サービスのデバイス名称特性

ハンドル	アトリビュートタイプ	アトリビュート UUID	パーミッション	アトリビュート値
0x0001	サービス宣言	0x2800	Read Only	0x1800 (Generic Access)

<table>
<tr><th>ハンドル</th><th>アトリビュートタイプ</th><th>アトリビュートUUID</th><th>パーミッション</th><th>アトリビュート値</th></tr>
<tr><td></td><td></td><td>(Primary Service)</td><td></td><td></td></tr>
<tr><td>0x0002</td><td>特性宣言</td><td>0x2803
(Characteristic)</td><td>Read Only</td><td><table><tr><td>プロパティ</td><td>Read, Write</td></tr><tr><td>特性値ハンドル</td><td>0x0003</td></tr><tr><td>特性 UUID</td><td>0x2A00</td></tr></table></td></tr>
<tr><td>0x0003</td><td>特性値宣言</td><td>0x2A00
(Device Name)</td><td>Read
Write(Unauthenticated ペアリングが必要)</td><td>デバイス名称
(Renesas-BLE)</td></tr>
</table>

7.4.2　ユーザプロファイル作成方法

BLE ソフトウェアでは、ユーザによるプロファイル作成が可能です。本項では、ユーザによるプロファイル作成方法について説明します。

7.4.2.1　GATT クライアントロール

GATT クライアントとなるプロファイルのロールは、rBLE API を使用してサービスや特性の検索および、特性値・特性記述子の読み出し・書き込みを行うことができます。また、サーバからの特性値の表示や通知は、登録されたコールバック関数にイベント通知されます。サーバからの特性値表示に対する応答は、BLE ソフトウェアが自動的に行います。

サービス検索や特性検索などの手続きや要件は、それぞれのプロファイル仕様書を参照ください。

7.4.2.2　GATT サーバロール

GATT サーバとなるプロファイルのロールは、GATT データベースを構築することにより実現可能です。

リモート GATT クライアントからのサービス・特性検索や特性値の読み出しは、BLE ソフトウェアが GATT データベースを参照し、パーミッション・プロパティに応じた応答を自動的に返します。

リモートデバイスからの特性値の書き込みは、BLE ソフトウェアが該当する特性値のパーミッションおよびプロパティの確認を行い、書き込み可能であれば rBLE API にてアプリケーションへ通知します。BLE ソフトウェアでは、データベースのアトリビュート値の更新は行いません。書き込み要求値の正当性をアプリケーションで確認し、データベースの更新を行ってください。また、書き込み要求に応答が必要な場合は、アプリケーションから行ってください。

特性値の表示や通知は、rBLE API にて実行可能です。

以下に、GATT データベース構築方法について記します。7.4.1 を参照し、データベースの構築を行ってください。なお、データベースに登録するサービスや特性などのアトリビュートは、それぞれのプロファイル・サービス仕様書を参照してください。

(1)　アトリビュート識別番号の定義

prf_config.h に定義しているアトリビュートインデックス列挙体の最後尾に、ユーザ作成プロファイルで必要となるアトリビュート分のインデックス「XXXX_IDX_XXXX」を追加してください。このインデックス値は

BLE ソフトウェア内部でデータベース要素の識別に使用します。

【注】　prf_config.h に定義済みの列挙値は変更しないでください。BLE ソフトウェアとデータベースに不整合が生じ、正常に動作しません。

(2)　アトリビュートハンドルの定義

db_handle.h に定義しているアトリビュートハンドル列挙体の最後尾にある DB_HDL_MAX の直前に、ユーザ作成プロファイルで必要となるアトリビュート分のハンドル「XXXX_HDL_XXXX」を追加してください。この列挙値は、データベースの配列要素番号と同値になるよう設定しています。アプリケーションではこの列挙値を用いて、アトリビュートを識別することができます。また、特性宣言などアトリビュート値にアトリビュートハンドルが必要な場合は、この列挙体宣言を使用することができます。

【注】　アトリビュートハンドル列挙値は、データベースの配列要素番号と同値になるよう設定しています。

(3)　データベース配列の構築

prf_config.c のデータベース配列 atts_desc_list[ ]の最後尾に、ユーザ作成プロファイルで必要となるサービス定義(7.4.1.2(1))、インクルード定義(7.4.1.2(2))、特性定義(7.4.1.2(3))を追加してください。各配列の要素番号は(2)で定義した列挙値と同値になるようにしてください。以下にデータベース構造体の各メンバ変数について説明します。

- type
  アトリビュートタイプの UUID を設定してください。7.4.1.2 を参照してください。
  アトリビュートタイプが 128bit UUID の場合は、DB_TYPE_128BIT_UUID マクロ(prf_config.h で定義)を設定してください。(後述の*value にて、128bit UUID を設定します。)

- length および maxlength
  アトリビュート値の現在サイズ、最大サイズを設定してください。

- taskid
  「TASK_ATTID」マクロ(prf_config.c で定義)を使用して、タスク ID「TASK_RBLE」と、(1)で定義した該当するアトリビュートの列挙値を結合し、設定してください。

- perm
  アトリビュートのパーミッションをサービスの仕様に従い適切に設定してください。BLE ソフトウェアでは、このパーミッションにより、リモート GATT クライアントからの該当アトリビュートへのアクセス制限を行います。

- *value
  アトリビュートタイプが 16bit UUID の場合：
  アトリビュート値格納先へのポインタを設定してください。アトリビュート値については、サービスの仕様に従い適切に設定してください。インクルード宣言のアトリビュート値設定には「ATTS_INCL」マクロ(prf_config.c で定義)が使用できます。特性宣言のアトリビュート値設定には「ATTS_CHAR」マクロ(prf_config.c で定義)が使用できます。
  特性宣言において、宣言するアトリビュートが 16bit UUID の場合、アトリビュート値には 16bit UUID

用の特性宣言構造体 struct atts_char_desc 型の変数を使用します。宣言するアトリビュートが 128bit UUID の場合、アトリビュート値には 128bit UUID 用の特性宣言構造体 struct atts_char128_desc 型の変数を使用します。

アトリビュートタイプが 128bit UUID の場合：
構造体 struct atts_elmt_128 型の変数へのポインタを設定してください。この構造体には、UUID、UUID 長、アトリビュート値へのポインタを設定します。

【注】1 使用しない既存プロファイル・ロールの取り外しは、マクロ定義の操作(6.1.11.1)により行ってください。
2 GAP および GATT のアトリビュート構成は変更しないでください。
3 ユーザ作成プロファイルの属するタスク ID は TASK_RBLE としてください。
4 アトリビュートのパーミッションおよびアトリビュート値は、サービスの仕様に従い適切に設定してください。
5 アトリビュート識別番号、ハンドルの列挙体およびデータベース配列は、今後のバージョンアップにより要素が増減する場合があります。
6 128bit UUID は下位バイトより前詰めで設定してください。

7.5 Find Me Profile (廃止)

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、本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本節を廃止しました。

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。

7.6 Proximity Profile (廃止)

※「7.5」を参照。

7.7 Health Thermometer Profile (廃止)

※「7.5」を参照。

7.8 Blood Pressure Profile (廃止)

※「7.5」を参照。

7.9 HID over GATT Profile (廃止)

※「7.5」を参照。

7.10 Scan Parameters Profile (廃止)

※「7.5」を参照。

7.11 Heart Rate Profile (廃止)

※「7.5」を参照。

7.12 Cycling Speed and Cadence Profile (廃止)

※「7.5」を参照。

7.13 Cycling Power Profile (廃止)

※「7.5」を参照。

7.14 Glucose Profile (廃止)

※「7.5」を参照。

7.15 Time Profile (廃止)

※「7.5」を参照。

7.16 Running Speed and Cadence Profile (廃止)

※「7.5」を参照。

7.17 Alert Notification Profile (廃止)

※「7.5」を参照。

7.18 Phone Alert Status Profile (廃止)

※「7.5」を参照。

7.19 Location and Navigation Profile (廃止)

※「7.5」を参照。

7.20 Vendor Specific

Vendor Specific(以降、VS)はルネサス独自の拡張機能を提供します。

7.20.1 消費電流ピーク通知機能

7.20.1.1 機能概要

BLE MCU は、データの送受信時に消費電流が多くなります。消費電流ピーク通知機能は、BLE ソフトウェアが組み込まれたシステムにおいて消費電流が多くなることを事前に通知する機能を提供します。この機能を使用することで、BLE MCU での消費電流が多くなる場面で一時的に他の処理を止めるなどを行い、システムでの多重負荷を避けることが可能です。

【注】RF 用スロー・クロック部内蔵発振回路(RF 部内蔵の 32kHz オシレータ)を使用する場合、本機能は使用できません。

7.20.1.2 機能仕様

消費電流ピーク通知機能は、コールバック関数である消費電流ピーク通知関数と消費電流ピーク通知終了関数を BLE ソフトウェア内部から呼び出しことで実現します。また、本機能は ON/OFF の設定が可能です。機能を使用する場合は、下記の定義マクロを有効にしてください。プロジェクトのコンパイルオプションで定義しています。

定義マクロ名：CFG_USE_PEAK

機能の設定については、BLE ソフトウェアのメイン関数である arch_main.c の main 関数内の BLE 初期化が完了する前までに設定を行ってください。また、設定できる時間は、消費電流がピークとなる 1msec 前、2msec 前、4msec 前の 3 種類です。本機能は、設定時間を管理するのに 12bit インターバルタイマを使用します。本機能を使用する場合、他の機能で 12bit インターバルタイマを使用しないでください。複数接続時にデータの送受信動作が重なる場合、コールバック関数が複数回呼ばれる場合があります。例として、peak_start、peak_start、peak_end、peak_end の順でコールバック関数を呼ぶことがあります。

なお、本機能を有効にした場合、システム全体としての消費電流が増えることがあります。

7.20.1.3 関数仕様

消費電流ピーク通知機能では、以下の３つの関数を使用します。

(1) 消費電流ピーク通知設定関数

消費電流ピーク通知機能設定関数は、消費電流ピーク通知機能の設定を行います。通知機能の ON/OFF 及び、通知時間を設定できます。本関数が呼ばれない場合は、通知は行われません。

BLE 初期化後に、本関数を呼ぶと設定不可の状態なため、PEAK_ERROR_STATE を返します。

表 7-28 消費電流ピーク通知機能設定関数

関数名	uint8_t peak_init (uint16_t peak_time)	
概要	消費電流ピーク通知機能設定関数	
説明	消費電流ピーク通知機能の設定を行います。	
引数	uint16_t peak_time	PEAK_TIME_OFF：通知機能を行わない。
		PEAK_TIME_1　：ピークの 1msec 前に通知開始
		PEAK_TIME_2　：ピークの 2msec 前に通知開始
		PEAK_TIME_4　：ピークの 4msec 前に通知開始
戻り値	PEAK_OK	正常終了
	PEAK_ERROR_PARM	パラメータエラー
	PEAK_ERROR_STATE	設定不可状態

(2)　消費電流ピーク通知関数

消費電流ピーク通知関数は、消費電流ピーク通知機能設定関数で設定された時間に BLE ソフトウェア内部から呼ばれます。本関数の処理は、ユーザが記述する必要があります。本関数内部で外部ポートの出力を行うことで他の MCU への通知が可能となります。ただし、BLE ソフトウェアの制限として、本関数内部は、必要最低限の処理のみとしてください。消費電流ピーク通知機能を使用しない場合は、空関数としてください。

表 7-29 消費電流ピーク通知関数

関数名	void peak_start (void)
概要	消費電流ピーク通知関数
説明	消費電流ピーク通知機能設定関数にて設定された時間に BLE ソフトウェアから呼ばれます。（コールバック関数）
引数	なし
戻り値	なし

(3)　消費電流ピーク終了通知関数

消費電流ピーク終了通知関数は、消費電流のピークである送受信が完了した後、BLE ソフトウェア内部から呼ばれます。本関数の処理は、ユーザが記述する必要があります。本関数内部にて消費電流ピーク通知関数で開始した出力を停止するなどを行ってください。ただし、消費電流ピーク通知関数同様、BLE ソフトウェアの制限として、必要最低限の処理のみとしてください。消費電流ピーク通知機能を使用しない場合は、空関数としてください。

表 7-30 消費電流ピーク終了通知関数

関数名	void peak_end (void)
概要	消費電流ピーク終了通知関数
説明	消費電流のピークが過ぎたときに BLE ソフトウェアから呼ばれます。（コールバック関数）
引数	なし
戻り値	なし

7.20.2 Sleep 機能

7.20.2.1 機能概要

Sleep 機能とは、低消費電流を目的とし、データの送受信の合間など BLE MCU が空いている時間に、BLE MCU を低消費電流状態に移行させる機能です。この状態は、BLE MCU 内の MCU 部を STOP モード、RF 部を SLEEP_RF モードにする Sleep 状態と、MCU 部を STOP モード、RF 部を DEEP_SLEEP モードにする DeepSleep 状態の 2 種類があります。

7.20.2.2 動作概要

データの送受信などが行われていない状態では、BLE MCU 内の MCU 部、RF 部ともにアイドル状態となります。このままでは、無駄な電流を消費してしまうため、MCU 部と RF 部を低消費電流状態に移行させます。低消費電流状態への移行は、BLE ソフトウェアが判断し、移行処理が行われます。また、Sleep 状態、DeepSleep 状態のどちらが選択されるかは、BLE MCU の空き時間情報から判断しています。よって、ユーザアプリでは、移行などの処理を行う必要はありません。

ただし、ユーザアプリを動作させるためなどで BLE MCU を Sleep 状態または、DeepSleep 状態に移行させたくない場合、Sleep 移行確認関数にて、false を戻すように変更してください。Sleep 移行確認関数は、BLE ソフトウェアが Sleep 状態へ移行する直前で呼び出します。この関数の戻り値が false の場合、Sleep 状態への移行を中断します。なお、Sleep 移行確認関数の仕様は以下の通りです。

表 7-31 Sleep 移行確認関数

関数名	bool sleep_check_enable (void)	
概要	Sleep 移行確認関数	
説明	BLE ソフトウェアの Sleep 状態へ移行の許可・禁止の確認	
引数	なし	
戻り値	true	Sleep 状態への移行を許可
	false	Sleep 状態への移行を禁止

7.20.2.3 アプリ実装時の注意点

BLE ソフトウェアは、データ送受信の頻度、コネクションの確立のための通信の頻度を基に、空き時間を算出します。この為、頻繁なデータ送受信や、短い間隔でコネクションの確認を行うと Sleep 状態、または DeepSleep 状態への移行が行われません。結果として、システムの消費電流が増えることになります。

また、arch_main.c 内の sleep_cont 関数は、手を加えずにそのまま使用してください。Sleep 機能が動作するのに重要な関数であるため、改造されますと Sleep 機能が正常に動作しなくなることがあります。

7.20.3 リセット処理

BLE ソフトウェアはメモリが枯渇してしまった場合、またはハードウェア・エラーが発生した場合に、ハードウェアのリセットを行います。

リセット要因発生時には、下記に示す関数が呼び出されます。現状は下記関数内にて不正命令を実行することで強制的にリセットを行っています。

フォルダ：¥Renesas¥BLE_Software_Ver_X_XX¥RL78_G1D¥Project_Source¥renesas¥src¥arch¥rl78
ファイル名：arch_main.c
関数名：void platform_reset(uint32_t error)

7.20.4 rBLE API による独自機能

本項では rBLE API にて提供されるルネサス独自機能について説明します。

7.20.4.1 Bluetooth Device アドレス書き込み機能

Bluetooth Device アドレス(以降、BD アドレス)書き込み機能は、不揮発性である DataFlash 領域にアクセスし BD アドレスの書き込みを可能にします。ここで BD アドレスとは、Bluetooth デバイスを一意に識別するためのアドレスです。BLE にて使用する BD アドレスは Public アドレスまたは Random アドレスがありますが、本機能の対象は Public アドレスを想定しています。

また DataFlash へのアクセスには、DataFlash ドライバが必要となります。DataFlash ドライバはサンプルコードとして提供します。

BD アドレス書込み時の動作は以下の通りです。

1. rBLE APIでDataFlashへのアクセス開始を設定
2. rBLE APIで書き込むBDアドレスを設定
3. BLEスタックがBDアドレス書込み機能を呼出し
4. BDアドレス書込み機能がDataFlashにBDアドレスを書き込み
5. rBLE APIでDataFlashへのアクセス停止を設定

7.20.4.2 DirectTestMode

DirectTestMode は、BLE 規格で規定された機能であり、BLE MCU の送信/受信能力をテストするための機能を提供します。主な特徴は以下の通りです。

- 受信/送信 DirectTestMode 開始コマンド、DirectTestMode 停止コマンドを rBLE API で提供
- 受信 DirectTestMode は、開始すると全ての 625usec スロットでパケット受信動作を実行
- 送信 DirectTestMode は、開始すると全ての 625usec スロットでパケット送信動作を実行
- 受信/送信 DirectTestMode は、DirectTestMode の停止コマンドを受けるまで動作を継続
- DirectTestMode の停止コマンドを受けると、DirectTestMode を終了し、イベントで完了を通知
- 受信 DirectTestMode 動作していた場合は、受信パケット数をイベントのパラメータで返却

DirectTestMode の詳細は Bluetooth Core Specification v4.2[Vol. 6], Part F を参照ください。

7.20.4.3 拡張 DirectTestMode

拡張 DirectTestMode は、DirectTestMode の柔軟な動作を実行するための機能を提供します。主な特徴は以下の通りです。

- DirectTestMode 動作を定義するパラメータ設定コマンドを rBLE API で提供
- パラメータ設定コマンドのパラメータで DirectTestMode 時の受信/送信パケット数を指定可能※1
- DirectTestMode 開始後、指定パケット数に達すると自動で DirectTestMode を終了、イベントで完了を通知
- 指定パケット数が 0 の場合、BLE 規格のとおり DirectTestMode の停止コマンドを受けるまで受信/送信動作を継続
- 指定パケット数が 0 以外の場合でも、DirectTestMode 終了コマンドにより途中終了可能
- パラメータ設定コマンドで指定した受信/送信パケット数は、DirectTestMode 実行終了後も保持

※1：受信 DirectTestMode 動作時のパケット数指定に関しては、指定されたパケット数よりも数パケット分多く通知されることがあります。これは、BLE ソフトウェア内部で指定されたパケット数を超えた場合にそのパケットを処理している間に受信したパケットも処理するため、その分のパケット数が加算されるためです。

7.20.4.4 バースト転送機能

バースト転送は、消費電流の測定を想定した、デバイスの常時受信/常時送信動作機能を提供します。主な特徴は以下の通りです。

- バースト転送を指定するコマンドは、拡張 DirectTestMode のパラメータ設定コマンドと共有
- 受信バースト転送の実行時、DirectTestMode の待受時間が無限となりデバイスは常時受信動作
- 送信バースト転送の実行時、DirectTestMode のパケットペイロード長が無限となりデバイスは常時送信動作
- 受信/送信バースト転送の開始は、受信/送信 DirectTestMode の開始コマンドで実行
- DirectTestMode の停止コマンドを受けるまで、バースト転送を継続
- DirectTestMode のパケット数を指定が 0 以外の場合も、DirectTestMode の停止コマンドを受けるまでバースト転送を継続

7.20.4.5　連続搬送波(CW)出力機能

連続搬送波(CW)出力は、電波法に基づく技術適合試験の試験項目に対応した連続搬送波(CW)を出力する機能を提供します。主な特徴は以下の通りです。

- 連続搬送波(CW)出力を指定するコマンドは、拡張 DirectTestMode のパラメータ設定コマンドと共有
- 連続搬送波(CW)出力の実行時、パケットペイロード長が無限となりデバイスは常時送信動作
- 連続搬送波(CW)出力の開始は、送信 DirectTestMode の開始コマンドで実行
- DirectTestMode の停止コマンドを受けるまで、連続搬送波(CW)出力を継続
- DirectTestMode のパケット数を指定が 0 以外の場合も、DirectTestMode の停止コマンドを受けるまで連続搬送波(CW)出力を継続

7.20.4.6　受信 DirectTestMode での RSSI 読出し機能

受信 DirectTestMode での RSSI 読出し機能は、受信 DirectTestMode 実行時の RSSI 値取得を可能にします。主な特徴は以下の通りです。

- 受信 DirectTestMode 時の RSSI 取得コマンドを rBLE API で提供
- 受信 DirectTestMode 時に取得した RSSI 値をイベントで通知
- RSSI 値の取得は、受信 DirectTestMode の実行開始から、DirectTestMode 完了後の通常パケット受信直前まで可能

7.20.4.7　送信パワー選択機能

送信パワー設定は、送信パワーの設定変更を可能にします。送信パワーは、Advertising/Scanning/Initiating ハンドルまたはコネクションハンドル別に設定が可能です。主な特徴は以下の通りです。

- 送信パワー設定コマンドを rBLE API で提供。
- 接続中のコネクションハンドル毎または Advertising/Scanning/Initiating 時の送信パワーを個別に設定可能

なお、送信パワーの初期値は、RBLE_VS_TXPW_HIGH(0dBm)です。Advertising/Scanning/Initiating の送信パワーは実行前に、Master/Slave 接続時の送信パワーは接続後に変更が可能となります。

設定方法は、Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル 基本編を参照してください。

7.20.4.8　GPIO 端子機能

RF トランシーバが持つ GPIO 端子を汎用ポートとして入出力アクセスすることを可能にします。なお、DeepSleep に遷移した場合、GPIO[2:0]端子は GPIO 機能の入力設定、GPIO[3]端子は兼用機能の出力設定にリセットされるため、出力値は維持されません。DeepSleep からの起床時に、設定した出力値に復帰します。

設定方法は、Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル 基本編を参照してください。

【注】GPIO[3]端子は DeepSleep 遷移時にリセットされ出力設定となります。GPIO[3]端子を入力ポートとして使用する場合は注意してください。

7.20.4.9 アダプタブル機能

パケット受信時に信号強度を測定し、受信後に最適なモードへと遷移するスレーブ動作専用の機能です。アダプタブル機能のモードには、信号強度が強の時に RF ロー・パワーモード、信号強度が中の時に RF ノーマルモード、信号強度が低の時に RF ハイ・パフォーマンスモードに移行する 3 つのモードが存在します。いずれも自動センスして自動遷移します。

- RF ロー・パワーモード : パケット受信時に測定した信号強度が強の時、省電力での動作を行います。マスタとスレーブ間で十分な信号強度が保てるような近距離、中距離のアプリケーションに最適なモードとなります。
- RF ノーマルモード : パケット受信時に測定した信号強度が中の時、省電力での動作が可能な RF ロー・パワーモードと、RF 特性をあげる RF ハイ・パフォーマンスモードの両方をバランスよく保つモードとなります。マスタとスレーブ間で信号強度が一定レベル保てる近距離、中距離のアプリケーションに最適なモードとなります。また、アダプタブル機能がオフ設定の場合、本モードに固定されます。
- RF ハイ・パフォーマンスモード : パケット受信時に測定した信号強度が低の時、RF 特性をあげる動作を行います。マスタとスレーブ間で信号強度が低くなりがちな中距離，長距離のアプリケーションに最適なモードとなります。

BLE ソフトウェアにより、アダプタブル機能のオン／オフを制御することが可能です。また、アダプタブル機能のオン設定時にモード遷移の通知／非通知を選択することが可能です。

アダプタブル機能の初期設定はオフ設定になっています。使用する場合はオン設定にしてください。また、マスタ動作時には使用できないため、マスタ動作時はアダプタブル機能をオフ設定で使用してください。

設定方法は、Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル 基本編を参照してください。

7.20.4.10 RF トランシーバ電源制御機能

RF トランシーバに供給される電源の制御を可能にします。電源制御コマンドは以下の通りです。

- RF 電源 OFF : RF トランシーバへの電源供給を停止します。PCLBUZ0 から 16.384kHz または 32.768kHz クロックを出力している場合、RF 電源 OFF 中はクロック出力を停止します。
- RF 電源 ON（DC-DC コンバータ有効） : 内蔵 DC-DC コンバータを有効に設定し、RF トランシーバへの電源供給を再開します。
- RF 電源 OFF（DC-DC コンバータ無効） : 内蔵 DC-DC コンバータを無効に設定し、RF トランシーバへの電源供給を再開します。

【注】本機能の使用に際して、以下の点にご注意ください。

- ON/OFF の設定に関わらず電源制御コマンドが受け付けられた場合は、RWKE のカーネルイベント、メッセージ、タイマキューが初期化されます。
- RF 電源 OFF 中は下記の機能が使用できません。
  - ✧ RF 部からの高速クロック出力
  - ✧ RF 部の GPIO[3:0]端子 機能
  - ✧ RWKE のタイマ管理機能
  - ✧ RSCIP の再送機能
- RF 電源を ON した後は、必ず GAP リセットを実行してください。また、RF 電源 OFF 中は、GAP リセットのみが受け付け可能です。GAP リセット実行により RF 電源が ON され、内蔵 DC-DC コンバータの使用については、直前の ON 時の設定が引き継がれます。

8. EEPROM エミュレーションライブラリ

8.1 EEPROM エミュレーションライブラリについて

BLE ソフトウェアでは、BD アドレスを BLE-MCU のデータ・フラッシュ領域に格納するために、EEPROM エミュレーションライブラリを使用しています。BLE ソフトウェアは、EEPROM エミュレーションライブラリの全てのバージョンおよび機能の動作を保証するものではありません。動作確認を行ったバージョンは、以下の通りです。他のバージョンの使用や機能を追加する際には、ドライバの変更が必要となる場合があります。

【注】 各開発環境およびコンパイラによってダウンロードするファイルが異なります。

【注】 BLE プロトコルスタック V1.20 より、EEPROM エミュレーションライブラリ Pack02 に統一しました。

動作確認バージョン

CS+ for CC/e^2 studio(CC-RL) 版：

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

CS+ for CA,CX 版：

RL78 ファミリ CA78K0R コンパイラ用 EEPROM エミュレーションライブラリ Pack02 Ver.1.01

なお、EEPROM エミュレーションライブラリは、BLE ソフトウェアの一部ではなく、別製品となります。EEPROM エミュレーションライブラリに付属の使用条件をご確認の上、ご使用ください。

8.2 EEPROM エミュレーションライブラリの設定について

EEPROM エミュレーションライブラリを使用する場合、本ライブラリによる使用禁止 RAM 領域にスタックやデータバッファ等が配置されないようにする必要があります。リンクディレクティブファイル等により、メモリ配置を行ってください。なお、設定方法については、RL78 ファミリ EEPROM エミュレーションライブラリ Pack02 Ver.1.01 リリースノート（R20UT3485JJ0101）を参照してください。

8.3 EEPROM エミュレーションライブラリ使用時の注意事項

ユーザアプリから EEPROM エミュレーションライブラリを介して、データ・フラッシュ領域を使用することが可能です。使用方法については、EEPROM エミュレーションライブラリのアプリケーションノートを参照してください。

また、EEPROM エミュレーションライブラリによるデータ・フラッシュへの書き込み、読み出しは BLE-MCU の処理時間を占有することとなり、他の処理へ影響を及ぼす可能性があります。よって、EEPROM エミュレーションライブラリを使用する処理は、通信動作などが行われない期間（電源投入直後など）で行うように設計してください。

9.　コードフラッシュライブラリ

9.1　コードフラッシュライブラリについて

BLE ソフトウェアでは、FW アップデート機能において、BLE-MCU のコード・フラッシュ領域にソースコードを書き込む際に、コードフラッシュライブラリを使用しています。BLE ソフトウェアは、コードフラッシュライブラリの全てのバージョンおよび機能の動作を保証するものではありません。動作確認を行ったバージョンは、以下の通りです。他のバージョンの使用や機能を追加する際には、ドライバの変更が必要となる場合があります。

【注】　各開発環境およびコンパイラによってダウンロードするファイルが異なります。

動作確認バージョン

CS+ for CC/e^2 studio(CC-RL) 版：

RL78 ファミリ CC-RL コンパイラ用フラッシュセルフプログラミングライブラリ Type01 Ver.2.21

CS+ for CA,CX 版：

RL78 ファミリ CA78K0R コンパイラ用フラッシュセルフプログラミングライブラリ Type01 Ver.2.20

なお、コードフラッシュライブラリは、BLE ソフトウェアの一部ではなく、別製品となります。コードフラッシュライブラリに付属の使用条件をご確認の上、ご使用ください。

※FW アップデート機能については Bluetooth Low Energy プロトコルスタック サンプルプログラムアプリケーションノートを参照してください。

9.2　コードフラッシュライブラリの設定について

コードフラッシュライブラリを使用する場合、本ライブラリによる使用禁止 RAM 領域にスタックやデータバッファ等が配置されないようにする必要があります。リンクディレクティブファイル等により、メモリ配置を行ってください。なお、設定方法については、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 Ver.2.20 リリースノート（R20UT0777JJ0102）または RL78 ファミリ CC-RL コンパイラ用フラッシュ・セルフ・プログラミング・ライブラリ Type01 Ver.2.21 リリースノート（R20UT3470JJ0100）を参照してください。

9.3　コードフラッシュライブラリ使用時の注意事項

使用方法については、コードフラッシュライブラリのアプリケーションノートを参照してください。

10.　ユーザアプリ作成時の注意点

10.1　RWKE のタイマ管理機能について

RWKE は、BLE プロトコルスタックを動作させるために設計された基本ソフトウェアです。Embedded 構成時などで BLE-MCU 上にユーザがアプリケーションを作成する場合、RWKE の機能を使用することができます。ただし、RWKE のタイマ管理機能を使用する場合は、RWKE のタスク上で使用してください。割り込み処理などから使用した場合、動作を保証しません。詳しくは、Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル 基本編の RWKE の章を参照してください。

10.2　タスク、および割り込みハンドラでの割り込み禁止時間について

BLE の通信処理に影響を及ぼす可能性があるため、タスクの処理時間、および割り込みハンドラでの割り込み禁止区間は短く設定してください。1msec 以内を推奨値としています。

10.3　サイズの大きいデータの送信について

BLE ソフトは、送受信のインターバル内に複数のデータの送信を要求されると、1 回のイベントで複数回の送受信を行います。複数回の送受信が行われると、他の接続の送受信動作に影響を及ぼし、他の接続が維持できなくなる可能性があります。よって、一つの接続に対して、インターバル内での複数回の送受信要求の発行を行わないように設計してください。

1 回のイベントで複数回の送受信を行う必要がある場合、該当の接続のインターバルを他の接続と違う値としてください。例えば、通常の接続はインターバルの設定を 500ms とした場合、複数回送信する可能性のある接続のインターバルは 520ms などに設定し、500ms との公倍数が大きな値となるように設定してください。このことにより、送受信のイベントが近くなる確率が減り、通信の失敗の回数を減らすことができます。

10.4　BLE MCU の処理性能について

Master デバイス、Slave デバイス間でデータを受信すると、データ保持用にメモリを動的に確保します。このメモリの確保は、RWKE のメモリ管理機能を使用しています。システムとしてのメモリの枯渇を防ぐために、次のインターバルまでに処理が完了できるように設計してください。

10.4.1　Modem 構成の場合

Modem 構成の場合には、データの処理のボトルネックはシリアル通信となります。BLE MCU と APP MCU の通信速度を 4800bps とした場合、1byte の転送時間は、約 2ms となります。シリアルの通信時間と BLE ソフトの動作時間から、BLE-MCU の動作クロックを 8MHz とした場合 1 秒間に 100byte ほどの処理性能となります。よって、Master と各 Slave 間の送受信できる総データ量は 1 秒間に 100byte に収まるようにシステムを設計してください。すべて同じインターバルとすべて違うインターバルの例を下記に示します。

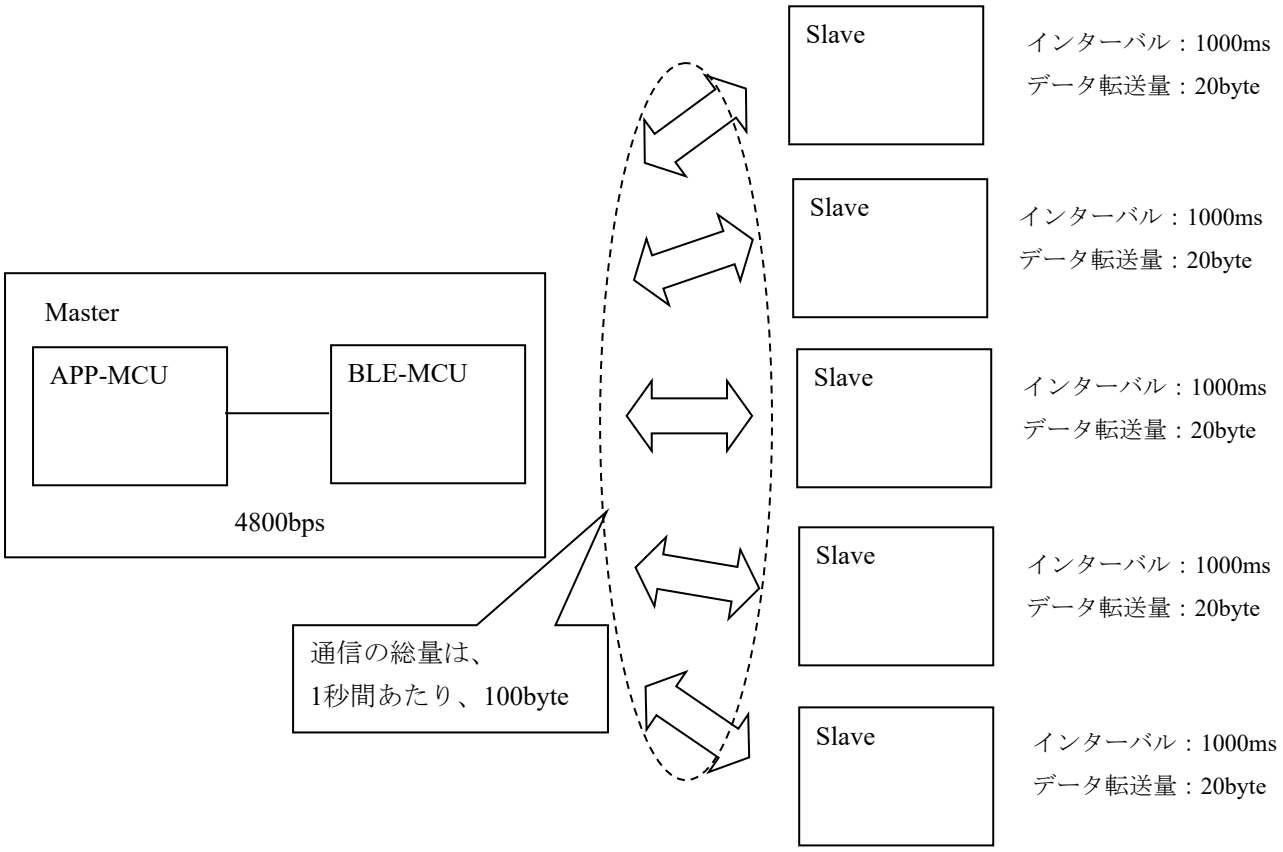


図 10-1 接続台数が 5 台ですべて同じインターバルの場合

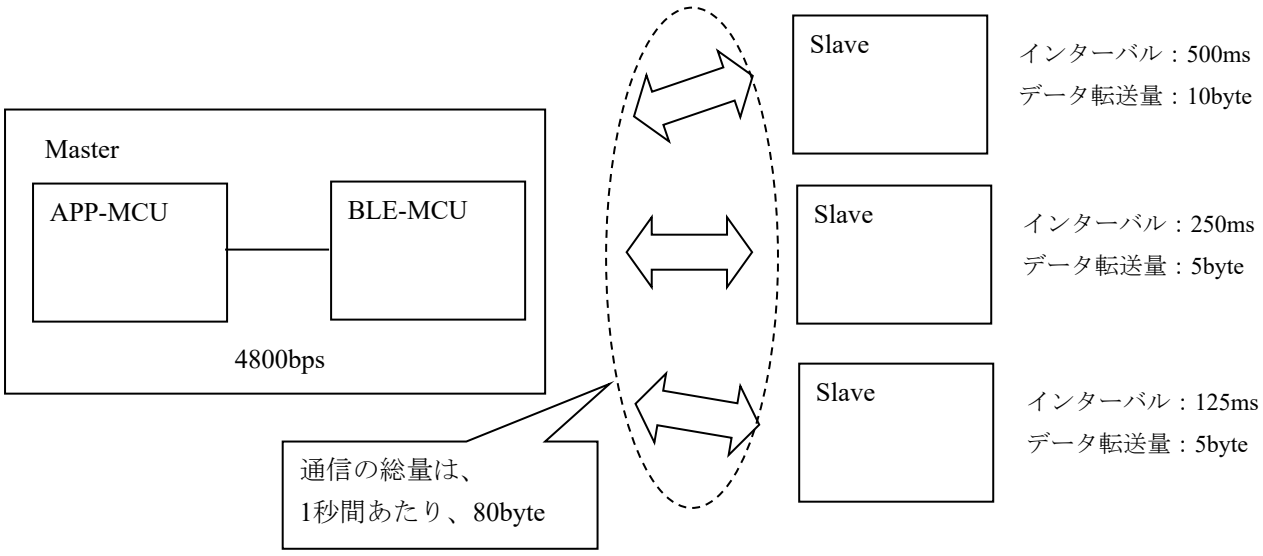


図 10-2 接続台数が 3 台ですべて違うインターバルの場合

11.　FW アップデート機能の実装

11.1　FW アップデート機能とは

BLE を制御するための FW をアップデートする機能で、以下のレイヤのアップデートが可能です。
・GATT BASED プロファイル(BLP/HRP など)
・ユーザアプリケーション

以降の章にFWアップデート機能を実現するために必要な処理や実装にあたり注意すべき点について記載します。
また、以降の章では FW アップデート対象となるデバイスを Receiver デバイス、FW アップデート用データを送信するデバイスを Sender デバイスと記載します。

11.2　FW アップデート機能実現のために必要な機能

FW アップデート機能を実現するために以下の機能を実装する必要があります。

〇Receiver デバイス向け
・コードフラッシュメモリへの書き込み機能
・FW アップデート用データ送受信プロファイル
・FW アップデート制御アプリケーション(受信側)

〇Sender デバイス向け
・FW アップデート用データ送受信プロファイル
・FW アップデート制御アプリケーション(送信側)

各機能について詳細や機能の実装例を記載します。

11.2.1　コードフラッシュメモリへの書き込み機能

Receiver デイバスで実装が必要な機能で、Sender デバイスから送信された FW アップデートデータを自身のコードフラッシュメモリに対して消去→書き込み、ベリファイなどを行います。
コードフラッシュメモリの消去、書き込み、ベリファイにはコードフラッシュライブラリを使用します。
※コードフラッシュライブラリの入手方法などは 9.コードフラッシュライブラリを参照してください。

コードフラッシュライブラリでは 1Block(1024byte)単位でのメモリ消去が可能なため、1Block 消去→1Block 書き込みを繰り返し、FW をアップデートさせていきます。

11.2.2　FW アップデートデータ送受信プロファイル

Sender デバイスから Receiver デバイスに対して FW アップデートデータを送信するために独自のプロファイルを用意する必要があります。
FW アップデートデータ送受信プロファイルの仕様については FW アップデートデータの送信シーケンスに合わせて必要な機能を実装してください。
FW アップデートデータ送受信プロファイルについては他の GATT BASED プロファイルと異なり FW アップデート機能によるアップデートは行えません。

※FW アップデート中に動作する機能については FW アップデート対象外となります。

以下に FW アップデートデータ送受信シーケンスの例を記載します。

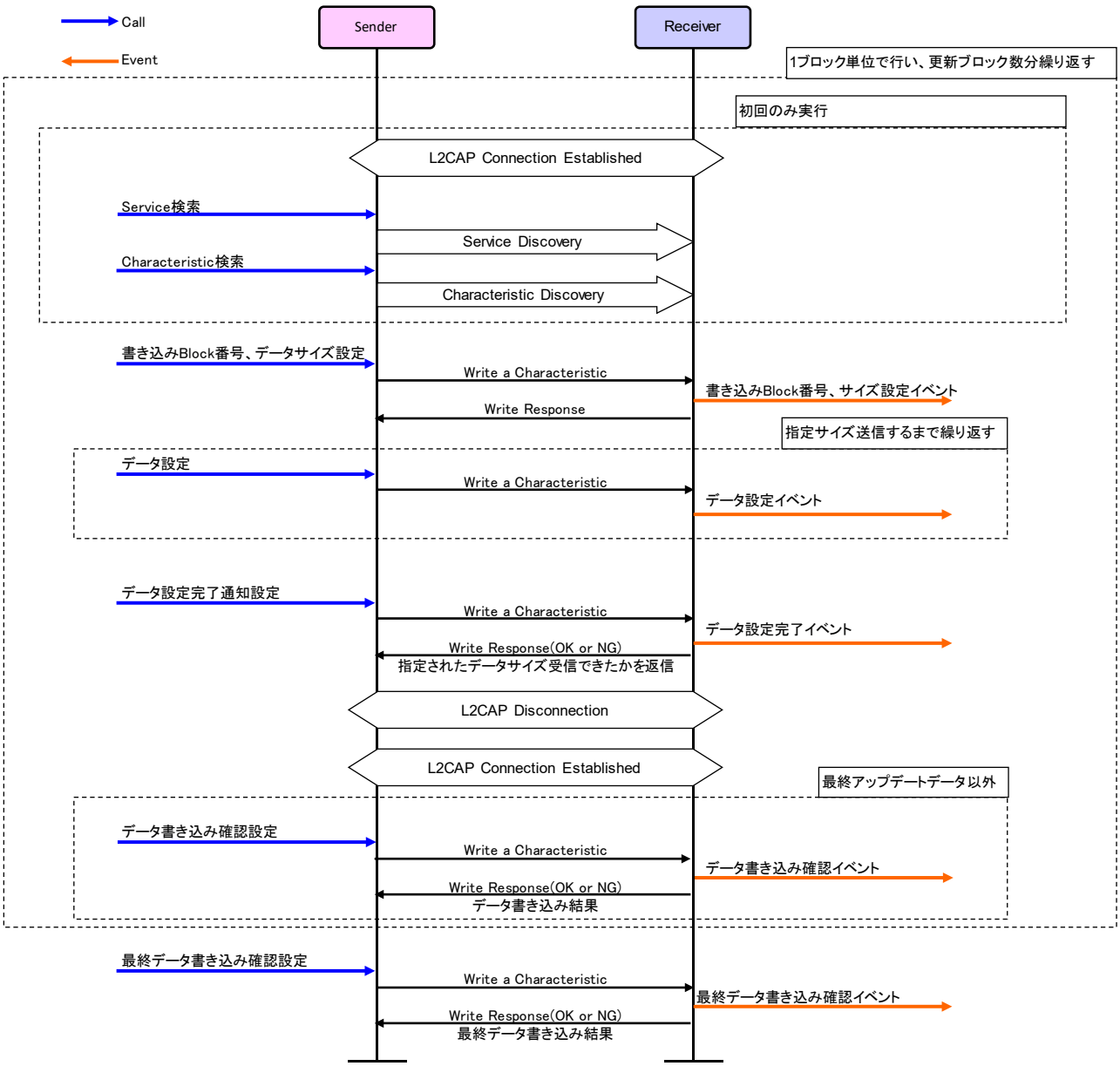


図 11-1 FW アップデートデータ送受信シーケンス例

11.2.3 FW アップデート制御アプリケーション(Receiver デバイス)

Receiver デバイスで実装が必要な機能で、FW アップデートデータを受信し、コードフラッシュメモリに書き込む制御を行います。
FW アップデートデータ制御アプリケーションは FW アップデートデータ送受信プロファイル同様に FW アプデート機能によるアップデートの対象外となります。

FW アップデートではマイコンのブート・スワップ機能を使用します。
ブート・スワップ機能の詳細は HW のユーザーズマニュアルを参照してください。

FW アップデート制御アプリケーションを開発するにあたり、実装する必要のある処理があります。
以下に実装が必要な処理を記載します。

・FW アップデート開始時には必ず Block4～7(0x01000～0x01FFF)のコードフラッシュメモリの消去を行います。
・Block4～7 の消去後にはコードフラッシュライブラリに実装されているマイコンのリセット関数(FSL_ForceReset)を実行し、1 度マイコンをリセットします。
・Block7 に書き込みを行う場合、0x01FFE に書き込むデータには 0x00FFE に書き込まれている値+1 を書き込みます。※FW アップデートでは 0x00FFE と 0x01FFE を使用して FW アップデート回数を管理します。
・Block42(0x0A800～0x0ABFF)と Block43(0x0AC00～0x0AFFF)は FW アップデートを行う毎に領域を切り替えて使用します。そのため、現在使用していない Block に対してデータの書き込みを行います。
※詳細は 11.3.1 に記載。
・FW アップデートの最後(全ての FW アップデートデータを書き込んだ後)には必ずブートフラグ切り替え関数(FSL_InvertBootFlag)、リセット関数を実行し、ブートクラスタの切り換えを行います。

以下に FW アップデートの Receiver デバイスにおける概略処理フローとデータ書き込みフローの例を記載します。

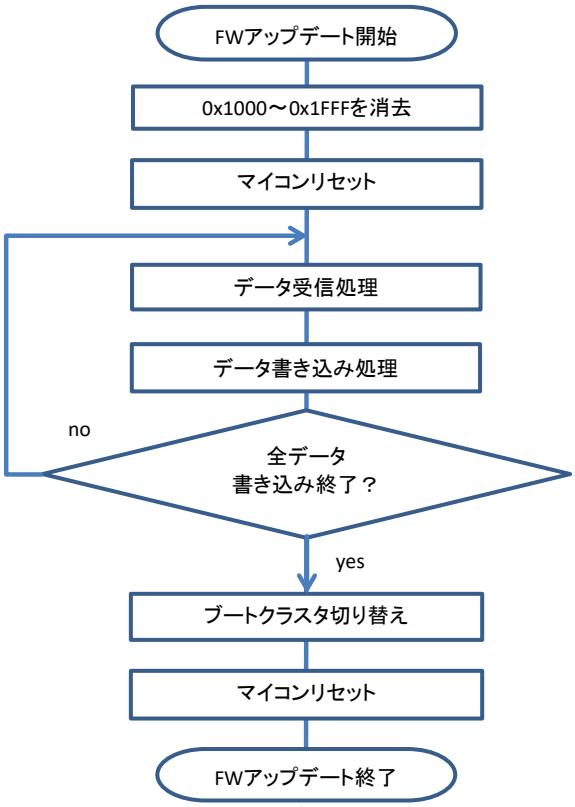


図 11-2 FW アップデート概略フロー例

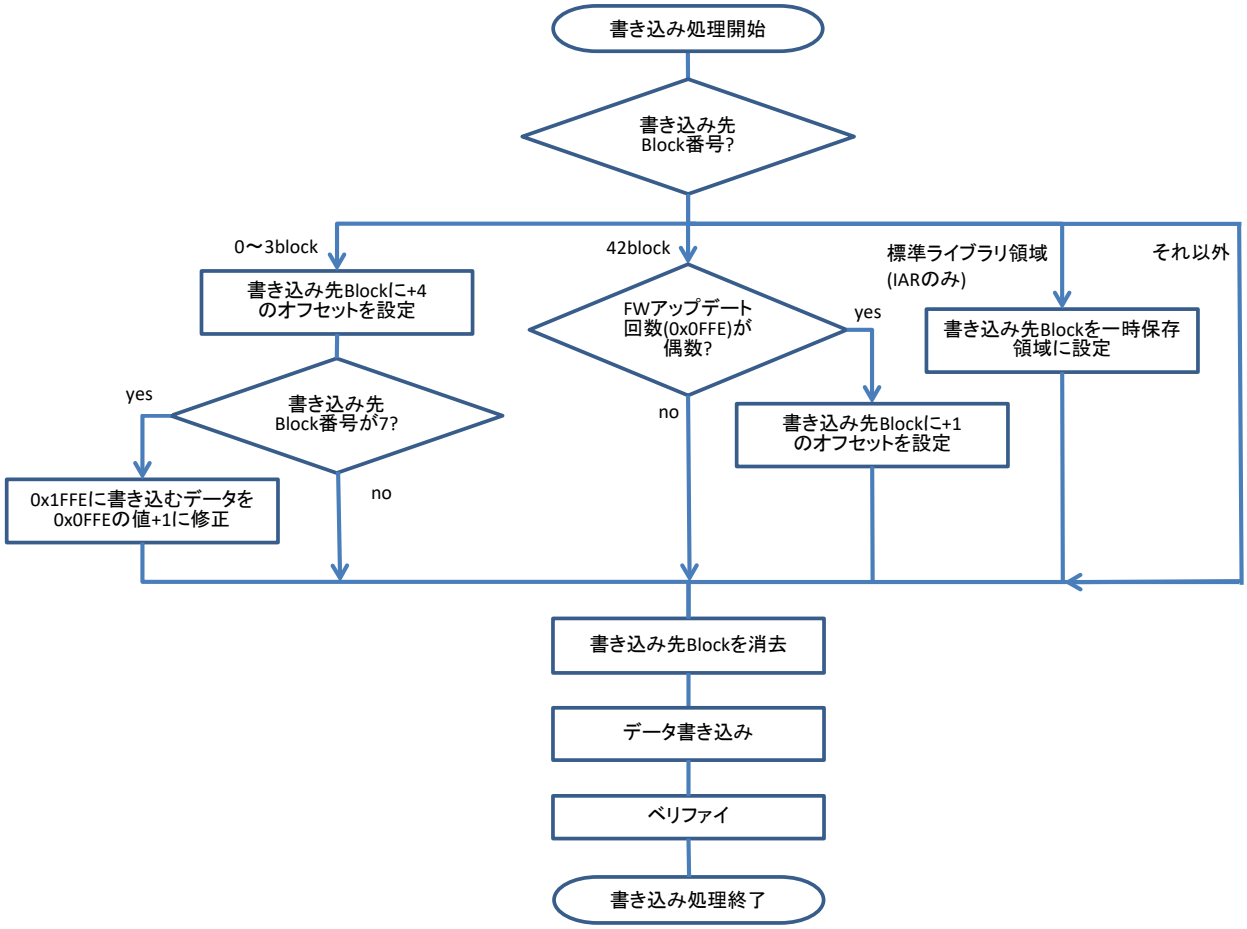


図 11-3 FW アップデートデータ書き込みフロー例

11.2.4　FW アップデート制御アプリケーション(Sender デバイス)

Sender デバイスで実装が必要な機能で、FW アップデートデータを送信するための制御アプリケーションです。

本機能では FW アップデートデータの取得、FW アップデートデータ送受信プロファイルの制御、FW アップデートデータの管理を制御する必要があります。

11.3　FW アップデート機能実装における制限と特殊処理

FW アップデート機能を実装するにあたり発生する制限や、特殊な処理についての詳細を記載します。

11.3.1　領域切り替え制御

FW アップデート機能を実現するために、1 部の領域を切り替えながら使用する必要があります。

FW アップデートでは FW アップデートの対象となる領域と対象外の領域があり、基本的には FW アップデート中に動作するコード部分は FW アップデートの対象外となっています。

しかし、FW アップデート中にアクセスする必要があるが、FW アップデートによってコードの更新を行う必要のある領域が存在します。そのため、FW アップデート中にアクセスする領域と FW アップデートによって書き換えを行う領域を別に確保する必要があります。

アクセスする領域と書き換えを行う領域の管理は FW アップデート回数を使用して行います。

以下に領域切り替え制御のイメージを記載します。

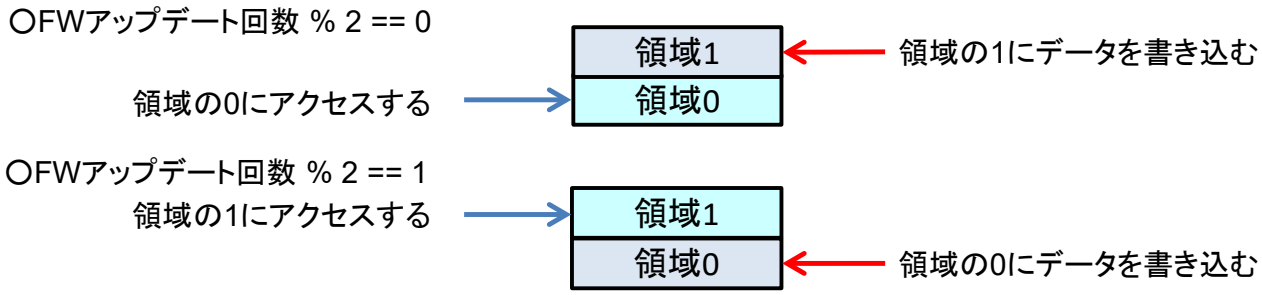


図 11-4 領域切り替え制御イメージ図

FWアップデートによるFWアップデートデータの書き込みは現在参照していない領域に対して消去→書き込みを行い、新しく書き込んだ領域へはFWアップデート完了(ブートクラスタの切り換え)後からアクセスを行います。

11.3.2 FW アップデート機能実装における制限

FWアップデート機能を実装するにあたり、ReceiverデバイスのFWには以下の制限があります。

・FWアップデート中はFWアップデート制御アプリケーション、FWアップデートデータ送受信プロファイル、Hostスタック以下、コードフラッシュライブラリ以外動作できません。
※通常時に動作するアプリケーションや、GATT-Basedプロファイルの関数などは動作できません。

・FWアップデートの前後でリンクするランタイムライブラリ、標準ライブラリは変更できません。
ランタイムライブラリや標準ライブラリのリンクを変更させないための対策についてはサンプルプログラムアプリケーションのFWアップデート環境を作成する際の注意事項を参照してください。

・FWアップデート制御アプリケーション、FWアップデートデータ送受信プロファイルにおいて初期値付き変数は使用できません。

11.3.3 アップデート対象領域とユーザ RAM 領域

以下に FW アップデート機能実装時のアップデート対象領域とユーザ RAM 領域の一覧を記載します。

環境	構成	領域	用途
CS+ for CA,CX (CA78K0R)	Embedded	0x00000～0x01FFF	ブートクラスタ 0/1
		0x04400～0x0A7FF	ROM 配置変数用領域
		0x0B000～0x0DFFF	Near に配置する関数、割り込み関数、初期値付き変数の初期値用の領域
		0x31400～0x3FBFF	コード用領域
		0xFBD10～0xFFE1F	RAM 配置変数用領域 ※スタックメモリ含む
	Modem	0x00000～0x01FFF	ブートクラスタ 0/1
		0x04400～0x0A7FF	ROM 配置変数用領域
		0x0B000～0x0DFFF	Near に配置する関数、割り込み関数、初期値付き変数の初期値用の領域
		0x33C00～0x3FBFF	コード用領域
		0xFC210～0xFFE1F	RAM 配置変数用領域 ※スタックメモリ含む
e^2 studio / CS+ for CC (CC-RL)	Embedded	0x00000～0x01FFF	ブートクラスタ 0/1
		0x04400～0x0A7FF	ROM 配置変数用領域
		0x0B000～0x0DFFF	Near に配置する関数、割り込み関数、初期値付き変数の初期値用の領域
		0x30000～0x3FBFF	コード用領域
		0xFBD20～0xFFE1F	RAM 配置変数用領域 ※スタックメモリ含む
	Modem	0x00000～0x01FFF	ブートクラスタ 0/1
		0x04400～0x0A7FF	ROM 配置変数用領域
		0x0B000～0x0DFFF	Near に配置する関数、割り込み関数、初期値付き変数の初期値用の領域
		0x30000～0x3FBFF	コード用領域
		0xFC220～0xFFE1F	RAM 配置変数用領域 ※スタックメモリ含む

※上記の領域はメモリ配置を変更しない場合の初期設定です。メモリ配置を変更した場合はこの通りにはなりません。

12.　HCI パケットモニタ機能

HCI パケットモニタ機能は、BLE プロトコルスタックの内部情報を仮想的に HCI パケットフォーマットのデータとして表示する機能です。本機能を使用することで、Advertising や接続時のパラメータ、対向デバイスとの送受信データなどをリアルタイムで容易に確認することができます。

12.1　HCI パケットモニタの機能構成

図 12-1 に HCI パケットモニタの機能構成を示します。本機能では BLE プロトコルスタックの内部情報を UART1 から出力します。PC では、受信した BLE プロトコルタックの内部情報を HCI パケットフォーマットに変換し Wireshark 上に表示します。

【注】　Wireshark は UNIX/Linux や Mac OS X、Windows で動作するネットワークプロトコルアナライザです。標準で Bluetooth のパケット解析に対応しており、本機能はそれを使用しています。

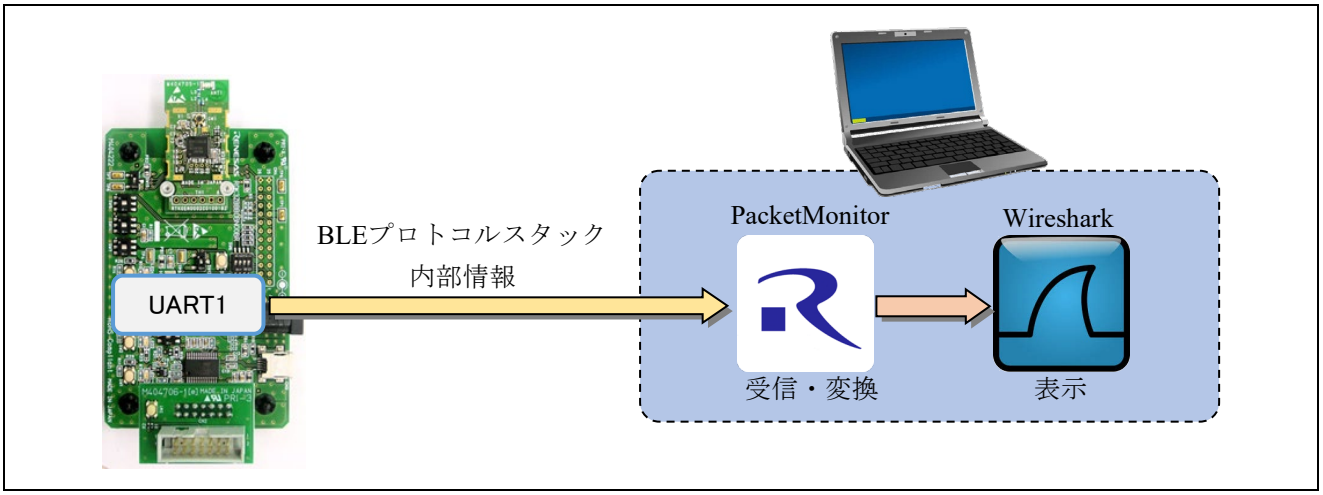


図 12-1　HCI パケットモニタ機能構成

本機能を使用するためには、以下のハードウェア・ソフトウェアが別途必要となります。

- ハードウェア
  - UART ←→ RS-232C (USB) 変換ケーブル
    - ボー・レート　：1Mbps 以上に対応
    - TTL レベル　　：3.3V
- ソフトウェア
  - Wireshark (https://www.wireshark.org/ より入手可能)

なお、以下の環境にて本機能の動作確認を行っています。

- ハードウェア
  - UART ←→ USB 変換ケーブル

    TTL-232RG-VREG3V3-WE (http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm)
- ソフトウェア
  - Windows 7 Enterprise Service Pack 1 (32-bit)
  - Wireshark Version 1.12.7 (32-bit)

12.2　HCI パケットモニタ機能の有効化

HCI パケットモニタ機能は、下記マクロをプロジェクトのコンパイルオプションで定義することで有効になります。

定義マクロ名：CFG_PKTMON　(デフォルトでは、noCFG_PKTMON としています。)

【注1】本機能は、デバッグ目的でのみ使用してください。
【注2】本機能では BLE プロトコルスタックの内部情報を出力するために UART1 を使用します。そのため Modem 構成において APP MCU との通信インタフェースとして UART1 を使用する場合や、ユーザアプリケーションで UART1 を使用する場合は、本機能を有効にすることはできません。

12.3　HCI パケットモニタ機能の使用方法

本節では、HCI パケットモニタ機能を使用するための準備・使用方法について示します。

12.3.1　　準備

HCI パケットモニタ機能を使用するためには、事前にそれぞれ以下に示す準備が必要となります。

- **RL78/G1D**

  HCI パケットモニタ機能を有効にして作成した HEX ファイルを書き込んでください。HCI パケットモニタ機能の有効方法については、12.2 節を参照して下さい。

- **評価ボード**

  UART←→RS-232C(USB) 変換ケーブルを評価ボードに接続してください。評価ボードと変換ケーブルの接続信号名を表 12-1 をに示します。

表 12-1　評価ボードと変換ケーブルの接続信号名

評価ボード			変換ケーブル信号名
コネクタ	pin	信号名	
CN4	4	GND	GND
CN4	12	TXD1	RXD

- **PC**

  Wireshark を下記より入手し、インストールしてください。

  https://www.wireshark.org/

  初めて Wireshark で Bluetooth のパケット解析を使用する場合は、Wireshark を起動しステータスバーの“Profile”をクリックして“Bluetooth”を選択してください。

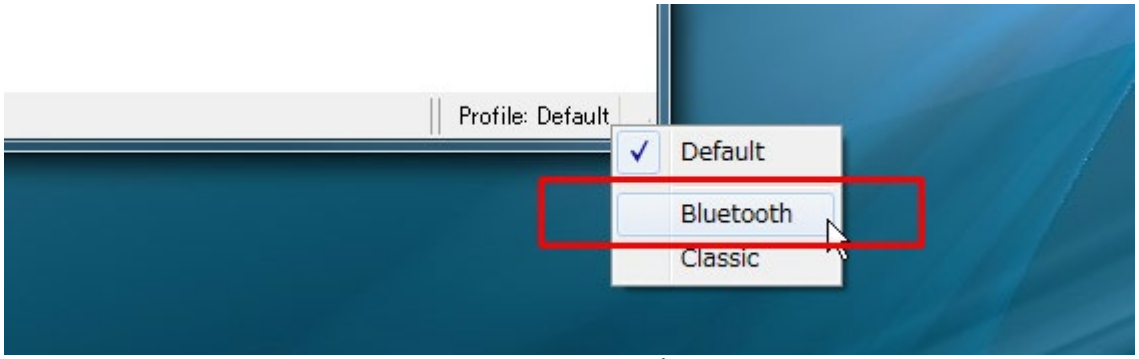


図 12-2　Profile 選択

12.3.2　使用方法

以下に示す手順を実行することで、HCI パケットをモニタリングすることができます。

(1) PacketMonitorの起動

ご使用の開発環境に合った PacketMonitor の実行ファイルを起動してください。

CS+ for CA, CX (CA78K0R)の場合　：　`PacketMonitor_for_CA.exe`

上記以外　：　`PacketMonitor.exe`

PacketMonitor の実行ファイルは、以下のフォルダに格納されています。

¥Renesas¥BLE_Software_Ver_X_XX¥PacketMonitor

(2) Wireshark実行ファイルの選択

PacketMonitor を起動すると、図 12-3 に示すダイアログが表示されます。事前にインストールした Wireshark の実行ファイル(Wireshark.exe)を選択し、[Run]ボタンを押下してください。

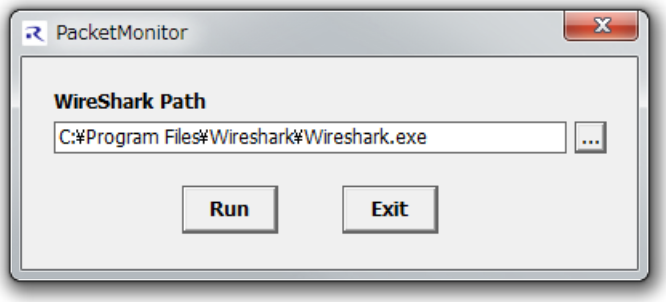


図 12-3　Wireshrak.exe の選択

(3) シリアルポートの設定

[Run]ボタンを押下すると、図 12-4 に示すシリアルポート設定ダイアログが表示されます。

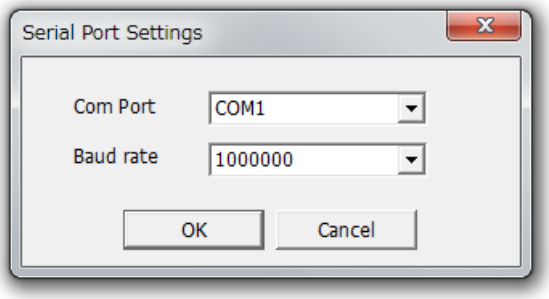


図 12-4　シリアルポートの設定

UART←→RS232C(USB)変換ケーブルを接続したシリアルポートを選択してください。ボー・レートは RL78/G1D の動作周波数に応じて以下のいずれかを選択し、[OK]ボタンを押下してください。

4MHz 動作時　：　`500,000bps`

8, 16, 32MHz 動作時　：　`1,000,000bps`

【注】　RL78/G1D のボー・レートは内部で固定になっています。変更することはできません。

(4) Wiresharkの起動

シリアルポートの設定が完了し[OK]ボタンを押下すると、自動的に Wireshark が起動します。Wireshark が完全に起動するまでお待ちください。

(5) RL78/G1Dの電源投入

Wireshark の起動後、RL78/G1D の電源を投入すると HCI パケットのキャプチャが開始します。電源投入後は、通常どおりアプリケーションの動作が可能です。

12.4　HCI パケットモニタ画面

図 12-5 に HCI パケットモニタ中の Wireshark の画面(デフォルト設定)を示します。Wireshark では、HCI パケットが種別ごとに色分け表示されます。また ATT や SMP のプロトコル解析にも対応しています。

画面上段には Host-Controller 間でやり取りされるパケットが時系列で表示されます。中段・下段には、上段で選択したパケットの詳細が表示されます。

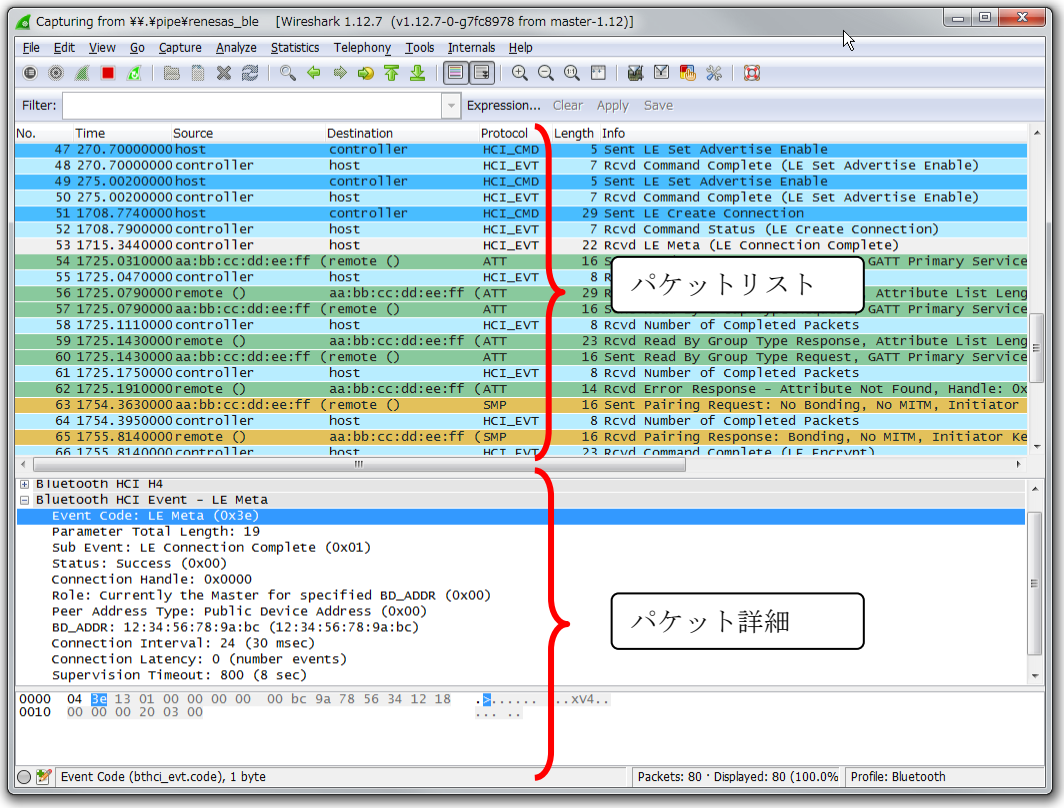


図 12-5　HCI パケットモニタ中の Wireshark

HCI コマンド・イベントパケットの詳細については、*Bluetooth Core Specification v4.2[Vol. 2], Part E Section 7 HCI Commands and Events* を参照してください。

ATT のパケットフォーマットについては *Bluetooth Core Specification v4.2[Vol. 3], Part F Section 3.3 Attribute PDU* を参照してください。

SM のコマンドフォーマットについては *Bluetooth Core Specification v4.2[Vol. 3], Part H Section 3 Security Manager Protocol* を参照してください。

付録 A　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. HID over GATT Profile Specification v1.0, Bluetooth SIG
13. HID Service Specification v1.0, Bluetooth SIG
14. Battery Service Specification v1.0, Bluetooth SIG
15. Scan Parameters Profile Specification v1.0, Bluetooth SIG
16. Scan Parameters Service Specification v1.0, Bluetooth SIG
17. Heart Rate Profile Specification v1.0, Bluetooth SIG
18. Heart Rate Service Specification v1.0, Bluetooth SIG
19. Cycling Speed and Cadence Profile Specification v1.0, Bluetooth SIG
20. Cycling Speed and Cadence Service Specification v1.0, Bluetooth SIG
21. Cycling Power Profile Specification v1.0, Bluetooth SIG
22. Cycling Power Service Specification v1.0, Bluetooth SIG
23. Glucose Profile Specification v1.0, Bluetooth SIG
24. Glucose Service Specification v1.0, Bluetooth SIG
25. Time Profile Specification v1.0, Bluetooth SIG
26. Current Time Service Specification v1.0, Bluetooth SIG
27. Next DST Change Service Specification v1.0, Bluetooth SIG
28. Reference Time Update State Service Specification v1.0, Bluetooth SIG
29. Alert Notification Service Specification v1.0, Bluetooth SIG
30. Alert Notification Profile Specification v1.0, Bluetooth SIG
31. Location and Navigation Service Specification v1.0, Bluetooth SIG
32. Location and Navigation Profile Specification v1.0, Bluetooth SIG
33. Phone Alert Status Service Specification v1.0, Bluetooth SIG
34. Phone Alert Status Profile Specification v1.0, Bluetooth SIG
35. Bluetooth SIG Assigned Numbers https://www.bluetooth.com/specifications/assigned-numbers
36. Services & Characteristics UUID https://www.bluetooth.com/specifications/gatt
37. Personal Health Devices Transcoding White Paper v1.2, Bluetooth SIG

付録 B　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。
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 のロールのひとつで、Advertising データを送信します。
Observer	Observer	GAP のロールのひとつで、Advertising データを受信します。
Central	Central	GAP のロールのひとつで、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 のロールのひとつで、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為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。
スキャンインターバル	Scan Interval	Advertising データの受信を行う間隔です。
スキャンウインドウ	Scan Window	スキャンインターバルごとに Advertising データの受信をおこなう期間です。

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。
スレーブレイテンシー	Slave Latency	スレーブレイテンシーの回数分のコネクションインターバルにおいて Slave デバイスは受信をする必要がありません。
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。

Bluetooth Low Energy プロトコルスタック
ユーザーズマニュアル

発行年月日　2022 年 1 月 31 日 Rev.1.20

発行　　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）

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

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

Colophon 3.1

Bluetooth Low Energy プロトコルスタック

R01UW0095JJ0120

ルネサス エレクトロニクス株式会社