Bluetooth® Low Energy Protocol Stack

Host MCU Simple API for RL78/G14

R01AN3784EJ0100
Rev.1.00
Jan 31, 2022

Introduction

The simple API described in this application note is an API that allows you to program data communications using Bluetooth low energy instantly with few steps. It runs on a Host MCU (Renesas Starter Kit for RL78/G14) of modem configuration and controls the RL78/G1D module (RY7011) or the BLE MCU of RL78/G1D on which the modem configuration Bluetooth Low Energy Protocol Stack (hereinafter, referred to as "BLE software") operates. Using BLE software provided by Renesas and custom profiles, data communication can be performed in a free format.

Target Device

Renesas Starter Kit for RL78/G14

Related Documents

Document Name	Document No.
Bluetooth Low Energy Protocol Stack	-
User's Manual	R01UW0095E
API Reference Manual: Basics	R01UW0088E
Application Note: rBLE Command Specification	R01AN1376E
Quick Start Guide	R01AN2767E
BLE Virtual UART Application	R01AN3130E
RL78/G1D	-
User's Manual: Hardware	R01UH0515E
RL78/G1D Evaluation Board	-
User's Manual	R30UZ0048E
RL78/G1D Module	-
RL78/G1D Module (RY7011) User's Manual: Hardware	R02UH0004E
RL78/G14	-
User's Manual: Hardware	R01UH0186E
CS+ Code Generator Tool Integrated Development Environment User's Manual: RL78 API Reference[CS+ for CA,CX][CS+ for CC]	R20UT3102E
e2 studio Code Generator Integrated Development Environment User's Manual: RL78 API Reference	R20UT3127E
AP4, Applilet3 User's Manual: RL78 API Reference	R20UT3125E
Renesas Starter Kit for RL78/G14	-
User's Manual	R20UT0785E
Tutorial Manual	R20UT0786E
Quick Start Guide	R20UT0787E
CPU Board Schematics	R20UT0784E

Contents

1. Overview

This application note describes the configuration of simple API, develop environment, program execution method, how to use API, and API specification. In the program execution method, operation is confirmed with a sample program (hereinafter referred to as "simple API program") that uses simple API and performs data communication.

Simple API provides nine APIs that you can program data communications using Bluetooth Low Energy (hereinafter referred to as "BLE") with fewer steps. It runs on a Host MCU (Renesas Starter Kit for RL78/G14) of modem configuration and controls the RL78/G1D module (RY7011) or the BLE MCU of RL78/G1D on which the modem configuration BLE software operates. Data communication can be performed in a free format using the general purpose communication (Virtual UART) [Note] of Renesas custom profile.

The contents that can be programmed with the simple API are shown below.

- Scanning for searching peripheral BLE devices.
- Advertising to send beacons to neighboring BLE devices.
- Connect to remote device as slave device.
- Connect to remote device as master device. (number of connectable remote devices: 1)
- Data communication using general purpose communication profile.
  (Automatically execute encrypted communication by Just works)
- Filtering by white list using BD address.
  (6 public addresses / 6 random addresses: total 12 BD addresses)

An outline of the simple API is shown below.

1.	R_BLES_initialize()	Initialize of simple API
2.	R_BLES_whitelist()	Set device address to white list
3.	R_BLES_scan()	Execute of scanning
4.	R_BLES_advertise()	Execute of advertising and connect to master device
5.	R_BLES_connect()	Connect to slave device
6.	R_BLES_get_event()	Get events
7.	R_BLES_send_data()	Send data
8.	R_BLES_receive_data()	Receive data
9.	R_BLES_disconnect()	Disconnection between remote device

Note: The general purpose communication and the virtual UART are the same profile. In the case of a modem configuration, it is "general purpose communication profile", in the case of embedded configuration it is "virtual UART profile". For details of profiles, see "BLE Virtual UART Application (R01AN3130)".

2. Simple API Structure

2.1 System Configuration

The system configuration diagram used with the simple API program is shown below. Simple API is used in modem configuration. "Figure 2-1 System configuration (1) (RY7011)" uses the Renesas Starter Kit for RL78/G14 (hereinafter referred to as "RSK") for the Host MCU on which the simple API operates as local device. The BLE MCU uses the Module Evaluation Board (RM-110-RFB-2). The default baud rate of UART 2-wire branching connection method is 115,200 bps. The remote device uses the virtual UART (R01AN3130) application on the RL78/G1D evaluation board (RTK0EN0001D01001BZ).

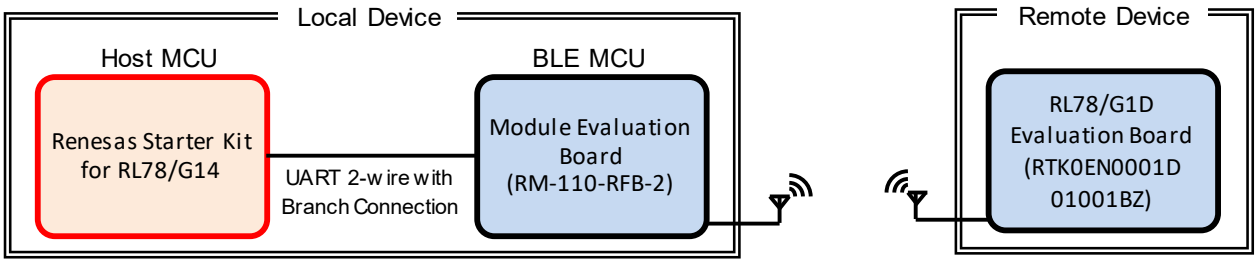

Figure 2-1 System configuration (1) (RY7011)

"Figure 2-2 System configuration (2) (BLE Evaluation Board)" is a configuration using the BLE software of modem configuration with the RL78 / G1D evaluation board (RTK0EN0001D01001BZ) as the BLE MCU of local device. The default baud rate of the UART 2 wire system is 4800 bps. When using BLE software, it is necessary to incorporate a general purpose communication profile in order to perform BLE communication. Refer to "8.1 Change of BLE software" for the installation method.

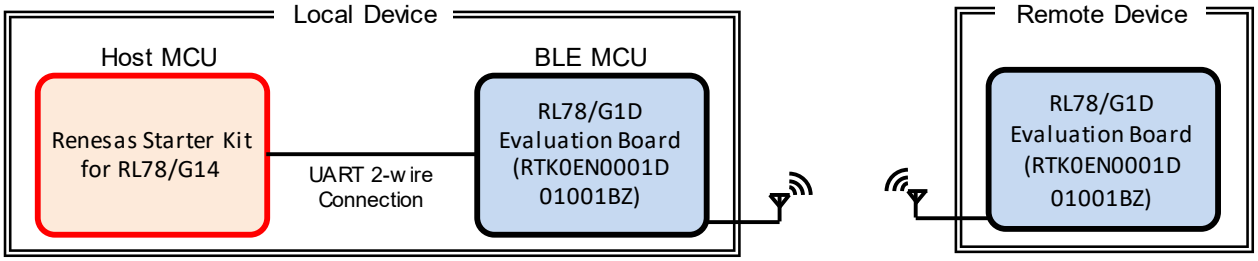

Figure 2-2 System configuration (2) (BLE Evaluation Board)

It can also use smartphone (Android or iOS) as Remote Device. In this case, encrypted communication by Just works is not done. This document also explains how to connect with a smartphone.

2.2 Software Configuration

This figure shows the software configuration diagram of the Host MCU RL78/G14 and the BLE MCU RL78/G1D.

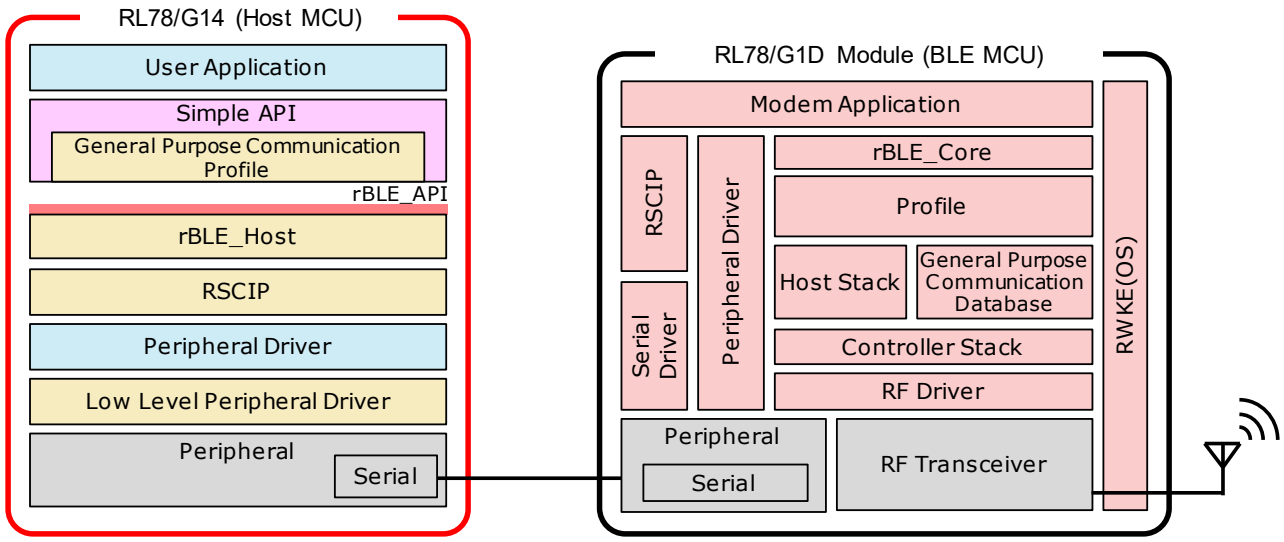

Figure 2-3 Software configuration

The software of Host MCU consists of Low Level Peripheral drivers and Peripheral drivers which controls MCU peripheral hardware, RSCIP (Renesas Serial Communication Interface Protocol), rBLE_Host which provides rBLE APIs, User Application which controls the system, and General Purpose Communication Profile using the GATT API. The simple API executes calling and scheduling of the rBLE API inside the API and provides a simple API to the user application.

Low Level Peripheral driver code is generated by the Code Generator. RSCIP and rBLE_Host are included in BLE software package and provided code. When developing software, it is necessary to use the latest code which is provided by BLE software package.

Table 2-1 Software configuration

Software	Functions	Necessity of software development
User Application (Simple API Program)	Initializing simple API Initializing peripheral driver	Need
Simple API	Providing a simple API RBLE API call and scheduling	No need (provided by package) Note3
rBLE_Host	Providing rBLE APIs Executing rBLE event callbacks	No need (provided by package) Note1
General Purpose Communication Profile	Custom profile using GATT API	No need (provided by package) Note3
RSCIP	Controlling serial communication	No need (provided by package) Note1
Peripheral Driver	Controlling Host MCU peripheral hardware	Need
Low Level Peripheral Driver	Controlling Host MCU peripheral hardware primitively	No need (generated by tool) Note2

Notes: 1. Source code is provided by BLE software.
2. Code generation tool automatically generates source code.
3. Source code is provided by source code of this application note.

The software of BLE MCU consists of RF driver which controls RF transceiver, Host/Controller stacks, Profiles, rBLE_Core, Serial Driver and RSCIP for communicating with Host MCU, RWKE (Renesas Wireless Kernel Extension) which manages the system and Modem application.

The build environment and tools and source code and libraries are provided by module FW and BLE software.

Table 2-2 BLE MCU software configuration

Software	Functions
Modem Application	Controlling RSCIP and rBLE
RWKE	Managing the whole system schedule and memory resource.
RSCIP	Controlling serial communication
Peripheral Driver/Serial Driver	Controlling BLE MCU peripheral hardware
rBLE_Core	Providing rBLE APIs
Profile	Providing Profiles functions
Host Stack	Providing GAP, GATT, SM, L2CAP functions
General Purpose Communication Profile GATT Database	GATT Database of General Purpose Communication Profile
Controller Stack	Providing Link Layer functions

3. Development Environment

It shows the environment used for simple API build and operation check.

3.1 Build Environment

- Host Environment
  - Windows 7 or later
  - PC/AT™ compatible computer
  - Processor ：At least 1.6GHz
  - Main Memory ：At least 1GB
  - Display ：1024 x 768 or higher resolution and 65,536 colors
  - Interface ：USB2.0 (E1 and USB to serial conversion cable)

- Integrated Development Environment (IDE)

  It support the following integrated development environment. Please use one of them.
  - Renesas CS+ for CA,CX V4.00.00 / Renesas CA78K0R V1.72
  - Renesas CS+ for CC V5.00.00 / RL78 Compiler CC-RL V1.04.00
  - e^2 studio V5.3.1.002 / RL78 Compiler CC-RL V1.04.00

3.2 Tools

- On-chip debugging emulator & Flash programmer
  - Renesas on-chip debugging emulator E1
- Flash programming software
  - Renesas Flash Programmer V3.02.01
- Code generator tool
  - Applilet3 (3.05.00.01)

3.3 Devices

- Evaluation board
  - Renesas Starter Kit for RL78/G14
  - Naito Densei Machida Mfg. Module Evaluation Board (RM-110-RFB-2)
  - Renesas BLE Evaluation Board for RL78/G1D (RTK0EN0001D01001BZ)
  - Smart Phone (Android or iOS)

3.4 Peripheral Functions

3.4.1 Peripheral Functions List

Peripheral functions of the RL78/G14 used in the simple API program and peripheral devices mounted on the RSK are shown below.

Table 3-1 Peripheral Functions

Peripheral Hardware	Purpose	Necessity
Serial Interface (Serial Allay Unit)Note1	UART 2-wire with branch connection (default baudrate 115200 bps) UART 2-wire connection (default baudrate 4800 bps)	Mandatory Note2
Interval Timer (12bit Interval Timer) Note1	Monitoring UART timeout	Mandatory Note2
LED (Port) Note1	Simple API program operation display LED0：P43 LED1：P44	Optional
SW (Interrupt) Note1	Operation of simple API program SW1：INTP8 SW2：INTP9 SW3：INTP10	Optional

Notes: 1. The lower row of the peripheral hardware column is the peripheral function name of RL78/G14.
2. The peripheral functions that the Host MCU needs at minimum to use the rBLE are "Mandatory" and the others are "Optional".

3.4.2 Code Generator Setting

The default setting of the code generation tool used in "Figure 2-1 System configuration (1) (RY7011)" is shown below. When using in "Figure 2-2 System configuration (2) (BLE Evaluation Board)", it need to change the settings with the code generation tool and build the program. Refer to "8.1.1 Preparation of Host MCU Program", "8.1.2 Build of Host MCU Program" for the implementation method.

Table 3-2 Code generator tool setting

Peripheral function		Purpose
Clock Generator Clock setting	Operation mode setting	High speed main mode 2.7(V)≦VDD≦5.5(V)
	EVDD setting	2.7(V)≦EVDD≦5.5(V)
	Main system clock (fMAIN) setting	High-speed OCO (fIH)
	High-speed OCO clock setting	24(MHz)
	High-speed system clock setting	Operation X1oscillation (f_X) Frequency 20 (MHz) Stable time 6553.6(us)
	Subsystem clock (fSUB) setting	Operation XT1oscillation (fXT) Frequency 32.768 (kHz) XT1oscillator oscillation mode setting: Low power consumption Subsystem clock in STOP,HALT mode setting: Enables supply
	Internal low-speed oscillation clock (fIL) setting	Frequency 15 (kHz)
	RTC, and interval timer operation clock setting	32.768 (fSUB) (kHz)
	CPU and peripheral clock setting	CPU and peripheral clock (fCLK) 24000 (fIH) (kHz)
Port Port4	P43	Out: 1
	P44	Out: 1
Interrupt External Interrupt	INTP8	Valid edge: Falling Priority: Low
	INTP9	Valid edge: Falling Priority: Low
	INTP10	Valid edge: Falling Priority: Low
Serial SAU1 UART2 Receive	Data length setting	8 bits
	Transfer direction setting	LSB
	Parity setting	None
	Stop bit length setting	1 bit fixed
	Receive data level setting	Normal
	Transfer rate setting	Baudrate 115200 (bps)
	Interrupt setting	Reception end interrupt priority (INTSR2): High
	Callback function setting	Reception end Reception error

Serial SAU1 UART2 Transmit	Transfer mode setting	Single transfer mode
	Data length setting	8 bits
	Transfer direction setting	LSB
	Parity setting	None
	Stop bit length setting	1 bit
	Transmit data level setting	Normal
	Transfer rate setting	Baudrate 115200 (bps)
	Interrupt setting	Transmit end interrupt priority (INTST2): Low
	Callback function setting	Transmission end
Interval Timer	Interval timer operation setting	Used
	Interval timer value setting	10 ms
	Interrupt setting	Detection of interval signal (INTIT) Priority: Low

3.5 Folder Structure

3.5.1 Folder Structure of Simple API Program

(R) indicates that it is a file included in BLE software. Please use the latest code file provided by BLE software during software development.

ble_simple_api_rl78g14	
├──project	
│ ├──CS_CA	CS+ for CA, CX project folder
│ │ │ ble_simple_api_rl78g14.mtpj	
│ │ │	
│ │ └──cg	
│ │ r_cg_cgc.c	clock generator driver code file
│ │ r_cg_cgc.h	clock generator driver header file
│ │ r_cg_cgc_user.c	clock generator driver user code file
│ │ r_cg_intc.c	external interrupt driver code file
│ │ r_cg_intc.h	external interrupt driver header file
│ │ r_cg_intc_user.c	external interrupt driver user code file
│ │ r_cg_it.c	interval timer driver code file
│ │ r_cg_it.h	interval timer driver header file
│ │ r_cg_it_user.c	interval timer driver user code file
│ │ r_cg_macrodriver.h	macro header file
│ │ r_cg_port.c	port driver code file
│ │ r_cg_port.h	port driver header file
│ │ r_cg_port_user.c	port driver user code file
│ │ r_cg_serial.c	serial driver code file
│ │ r_cg_serial.h	serial driver header file
│ │ r_cg_serial_user.c	serial driver user code file
│ │ r_cg_userdefine.h	user defined macro header file
│ │ r_main.c	main loop code file
│ │ r_systeminit.c	peripheral initialization code file
│	
│ ├──CS_CCRL	CS+ for CC project folder
│ │ │ ble_simple_api_rl78g14.mtpj	
│ │ │ ble_simple_api_rl78g14.rcpe	
│ │ │ cstart.asm	
│ │ │ hdwinit.asm	
│ │ │ iodefine.h	
│ │ │ stkinit.asm	
│ │ │	
│ │ └──cg	code generate files folder
│ │ r_cg_cgc.c	
│ │ r_cg_cgc.h	
│ │ r_cg_cgc_user.c	
│ │ r_cg_intc.c	
│ │ r_cg_intc.h	
│ │ r_cg_intc_user.c	
│ │ r_cg_it.c	
│ │ r_cg_it.h	
│ │ r_cg_it_user.c	
│ │ r_cg_macrodriver.h	macro header file (This file is made changes according to "7.1 Code Generation (r_cg_macrodriver.h)".)
│ │ r_cg_macrodriver_g14_ccrl.h	macro header file (recovery file)
│ │ r_cg_port.c	
│ │ r_cg_port.h	
│ │ r_cg_port_user.c	
│ │ r_cg_serial.c	

```
│   │       r_cg_serial.h
│   │       r_cg_serial_user.c
│   │       r_cg_userdefine.h
│   │       r_main.c
│   │       r_systeminit.c
│   │
│   └──e2studio                                          e2 studio V5 project folder
│       │   .cproject
│       │   .HardwareDebuglinker
│       │   .info
│       │   .project
│       │   ble_simple_api_rl78g14 HardwareDebug.launch
│       │
│       ├──.settings
│       │   │   conflictresources.xml
│       │   │   Dependency_Scan_Preferences.prefs
│       │   │   org.eclipse.cdt.managedbuilder.core.prefs
│       │   │   Project_Generation_Prefrences.prefs
│       │   │
│       │   └──CodeGenerator
│       │       cgproject.cgp
│       │       cgprojectDatas.datas
│       │
│       └──src                                           code generate files folder
│           cstart.asm
│           iodefine.h
│           r_cg_cgc.c
│           r_cg_cgc.h
│           r_cg_cgc_user.c
│           r_cg_intc.c
│           r_cg_intc.h
│           r_cg_intc_user.c
│           r_cg_it.c
│           r_cg_it.h
│           r_cg_it_user.c
│           r_cg_macrodriver.h                         macro header file
│                                                      (This file is made changes according to "7.1 Code
│                                                      Generation (r_cg_macrodriver.h)".)
│           r_cg_macrodriver_g14_ccrl.h                macro header file (recovery file)
│           r_cg_port.c
│           r_cg_port.h
│           r_cg_port_user.c
│           r_cg_serial.c
│           r_cg_serial.h
│           r_cg_serial_user.c
│           r_cg_userdefine.h
│           r_main.c
│           r_systeminit.c
│           stkinit.asm
│
└──src
    │   ble_simple_api_rl78g14.c                        Simple API program
    │
    ├──platform
    │   └──rl78g14
    │       ├──driver
    │       │   ├──plf
    │       │   │       plf.c                           platform driver code file
```

```
│   │   │   plf.h                                  platform driver header file
│   │   │
│   │   ├──serial
│   │   │   uart.c                                 uart driver code file
│   │   │   uart.h                                 uart driver header file
│   │   │
│   │   └──timer
│   │       timer.c                                timer driver code file
│   │       timer.h                                timer driver header file
│   │
│   └──include
│       arch.h                          (R)        architecture header file
│       compiler.h                      (R)        compiler header file
│       iodefine.h                                 Macro definition for register access header file
│       ll.h                            (R)        low level macro header file
│       rscip_api.h                     (R)        RSCIP callback header file
│       rskrl78g14def.h                            RSK header file
│       types.h                         (R)        type definition header file
│
└──rBLE
   └──src
      ├──host
      │   rble_host.c                   (R)        rBLE_Host code file
      │   rble_if_api_cb.c              (R)        rBLE API callback code file
      │
      ├──gap
      │   rble_api_gap.c                (R)        GAP API code file
      │
      ├──gatt
      │   rble_api_gatt.c               (R)        GATT API code file
      │
      ├──sm
      │   rble_api_sm.c                 (R)        SM API code file
      │
      └──vs
          rble_api_vs.c                 (R)        VS API code file
      │
      ├──include
      │   prf_sel.h                     (R)        profile select header file
      │   rble.h                        (R)        rBLE macro header file
      │   rble_api.h                    (R)        rBLE API header file
      │   rble_api_custom.h             (R)        rBLE SCP API header file
      │   rble_trans.h                  (R)        rBLE communication header file
      │
      └──host
          rble_host.h                   (R)        rBLE_Host header file
      │
      ├──rbles_api
      │   rbles_api.c                               Simple API code file
      │   rbles_api.h                               Simple API header file
      │
      ├──rscip
      │   rscip.c                       (R)        RSCIP code file
      │   rscip.h                       (R)        RSCIP header file
      │   rscip_cntl.c                  (R)        RSCIP control code file
      │   rscip_cntl.h                  (R)        RSCIP control header file
      │   rscip_ext.h                   (R)        RSCIP external callback header file
      │   rscip_uart.c                  (R)        RSCIP serial communication code file
```

rscip_uart.h	(R)	RSCIP serial communication header file
sample_profile		
db_handle.h	(R)	database handle header file
vuart		
vuart.h		General purpose communication header file
vuartc.c		General purpose communication client code file
vuartc.h		General purpose communication client header file
vuarts.c		General purpose communication server code file
vuarts.h		General purpose communication server header file

3.5.2 Folder Structure of General Purpose Communication Database Difference File

Folder structure of the general purpose communication database difference file.

BLE_Software_Ver_1_20_diff_vuart	
└──RL78_G1D	
└──Project_Source	
├──rBLE	
│ └──src	
│ └──include	
│ rble_api_vuart.h	General purpose communication header file
└──renesas	
└──src	
└──arch	
└──rl78	
db_handle.h	General purpose communication Attribute database handles definition
prf_config.c	General purpose communication Attribute database index definition
prf_config.h	General purpose communication service definition

3.5.3 Folder Structure of Execution File

Folder structure of the executable file is shown below.

ROM_File	
├──BLE_MCU	
│ rBLE_Mdm_CCRL.hex	An execution file that runs on the RL78/G1D evaluation board which added a general purpose communication database to BLE software. (UART 2-wire with branch communication (4800 bps)) For details on how to create this execution file please refer to "8.1".
└──Host_MCU	
ble_simple_api_rl78g14_2wire_4200.hex	Simple API program that runs on RSK (UART 2-wire communication (4800 bps)) For details on how to create this executable file please refer to "8.1".
ble_simple_api_rl78g14_div2wire_115200.hex	Simple API program that runs on RSK (UART 2-wire with branch communication (115200 bps))

4. Execute Program

This chapter explains how to execute a simple API program by using a configuration that uses a module evaluation board for the BLE MCU in "Figure 2-1 System configuration (1) (RY7011)".

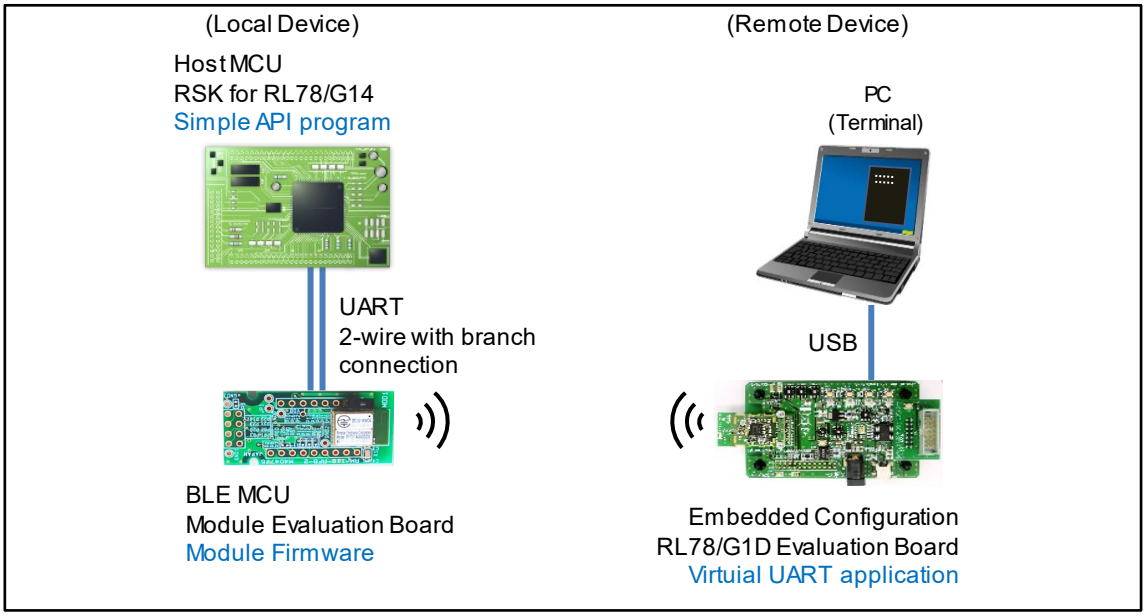

Figure 4-1 Operational Check Environment (1)

It also describes how to execute when using a smartphone as a remote device.

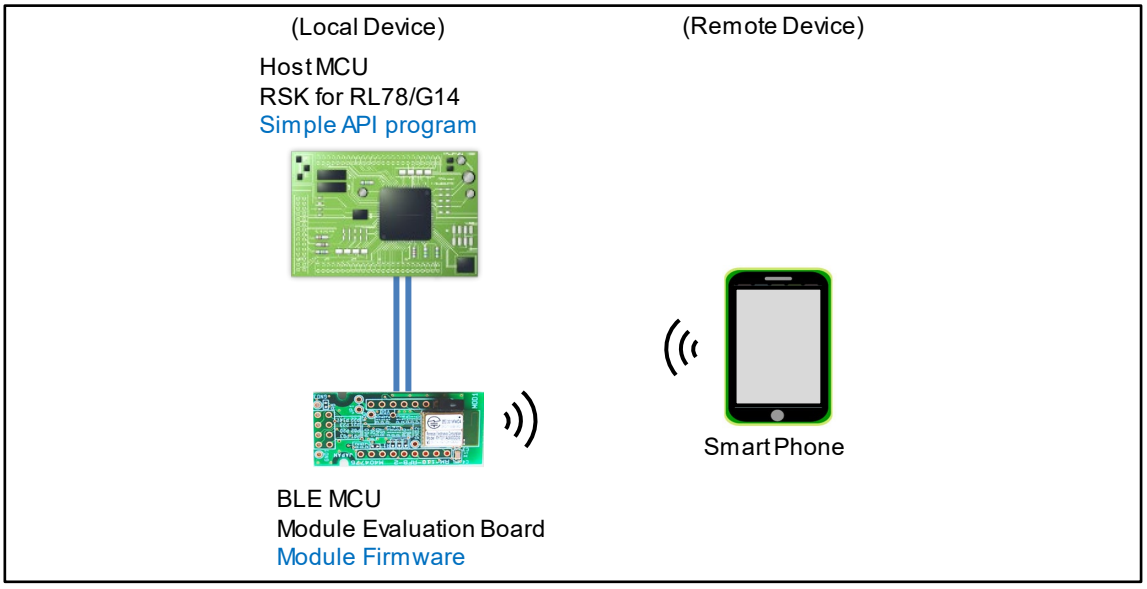

Figure 4-2 Operational Check Environment (2)

The executable files that can be used on the local device in "Figure 4-1 Operational Check Environment (1)" and "Figure 4-2 Operational Check Environment (2)" are shown below. For details on remote devices, refer to "4.2 Preparing Remote Devices".

Table 4-1 Execution file (RSK + Module Evaluation Board)

Local device	Execution file	Description
RSK	ble_simple_api_rl78g14_div2wire_115200.hex	Simple API program running on Host MCU. - UART 2-wire with branch connection - Baudrate 15200 bps
Module Evaluation Board	-	Program operating on BLE MCU. Use the firmware written on the module evaluation board.

In the configuration shown in "Figure 2-2 System configuration (2) (BLE Evaluation Board)" where the RL78/G1D evaluation board is used for the BLE MCU, it is necessary to prepare the programs of the Host MCU and the BLE MCU. Please prepare the program referring to "8.1 Change of BLE software" and execute "4.3" for RL78/G1D Evaluation Board, "4.4" for Smartphone (Android), "4.5" for Smartphone (iOS) depending on the remote device to be used.The following executable file that can be used with the local device of this configuration is included in this application note. Please refer to "8.1.2(4)Write executable file and board setting" and "8.1.4(4)Write executable file and board setting" and write the executable file to the local device.

Table 4-2 Execution file (RSK + RL78/G1D Evaluation Board)

Local device	Execution file	Description
RSK	ble_simple_api_rl78g14_div2wire_42000.hex	Simple API program running on Host MCU. Note1 - UART 2-wire connection - Baudrate 4800 bps
RL78/G1D Evaluation Board	rBLE_Mdm_CCRL.hex	Program operating on BLE MCU. Executable file incorporating a database of general purpose communication profile in BLE software. Note2

Notes: 1. It can also use the executable file of each integrated development environment indicated in "Table 8-1 Host MCU project file and execution file generation directory".
2. It can also use the executable file of each integrated development environment indicated in "Table 8-4 BLE MCU project file and execution file generation directory".

4.1 Preparing Local Device

4.1.1 Write a simple API program to RSK

Writes "ble_simple_api_rl78g14_div2wire_115200.hex" file to RSK of Host MCU. Refer to "8.1.2(4) Write executable file and board setting".

4.1.2 Preparing Module Evaluation Board

Writes "rBLE_Mdm_CCRL.hex" file to RL78/G1D evaluation board of BLE MCU. Refer to "8.1.4(4) Write executable file and board setting".

4.2 Preparing Remote Devices

Depending on the remote device to be used, execute either "4.2.1" or "4.2.2".

4.2.1 Write a Virtual UART application to RL78/G1D Evaluation Board

Writes Virtual UART application to RL78/G1D evaluation board of embedded configuration. Please download the application note "Virtual UART Application (R10AN3130)" from the following URL.

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

Please refer to "7.2 Build Procedure" and write the program to the RL78/G1D evaluation board.

4.2.2 Application installation on smartphone

Install the following application on Android device or iOS device to be set to Remote Device.

- (Android device) “GATTBrowser” - Renesas Electronics

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

- (for iOS device) “GATTBrowser” – Renesas Electronics

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

4.3 Execution Procedure (RL78/G1D Evaluation Board)

It shows the execution procedure in "Figure 4-1 Operational Check Environment (1)".

1. Supplies power to the local device and the remote device, and starts the terminal on the PC connected to the remote device.
2. LED0 (green) will turn on a light on RSK.
3. Make sure it is in simple AT command mode on the terminal. (Simple AT command mode and virtual UART mode will toggle by pressing the escape key.)
4. Execute "AT-C = (device address of local device)" on the terminal.
5. Local device and Remote device connect and "CONNECT" is displayed on the terminal.
6. LED1 (orange) will turn on a light on RSK.
7. Press the escape key to enter virtual UART mode.
8. Press SW2 on RSK. "Hello" is displayed on the terminal.
9. Press the escape key to enter simple AT command mode.
10. Press SW3 on RSK. Local device and Remote device disconnect and "DISCONNECT" is displayed on the terminal.
11. LED1 (orange) will turn off a light on RSK.

In the figure below, the red number indicates the execution procedure number.

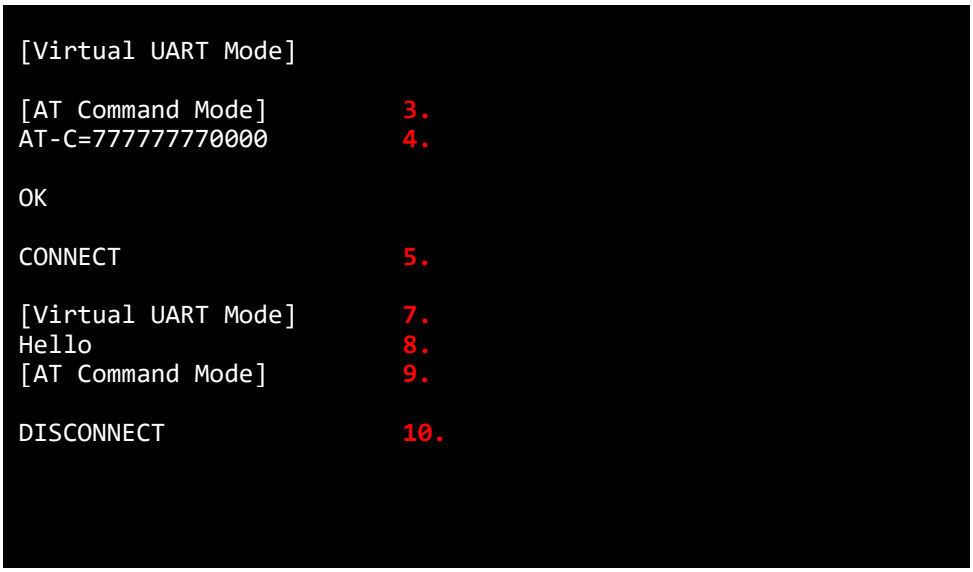

Figure 4-3 Display on terminal

4.4 Execution Procedure (Android)

The following shows the execution procedure when using an Android device in "Figure 4-2 Operational Check Environment (2)".

1. Supplies power to the local device and launches BLE Scanner on the smartphone.
2. LED 0 (green) on RSK turn on a light.
3. From the search result of the device, tap the round arrow at the right side of the device name labeled "RSK-RL78/G14" to connect.
   (Figure A1 - Arrow(1))
4. Tap "Indication Characteristic" of Renesas Virtual UART Service.
   (Figure A2 - Arrow(2))
5. Tap "Indication Off" to "Indication On".
   (Figure A3 - Arrow(3))
6. Local device and remote device are connected and LED 1 (orange) on RSK turn on alights.
7. Change from "HEX" to "String".
   (Figure A4 - Arrow(4))
8. Press SW2 on RSK.
   (Figure A4 - Arrow(5))
9. Press SW3 on RSK.
10. The local device and the remote device are disconnected and "Disconnected" is displayed.
    (Figure A5 - Arrow(6))
11. LED 1 (orange) on RSK turn off a light.

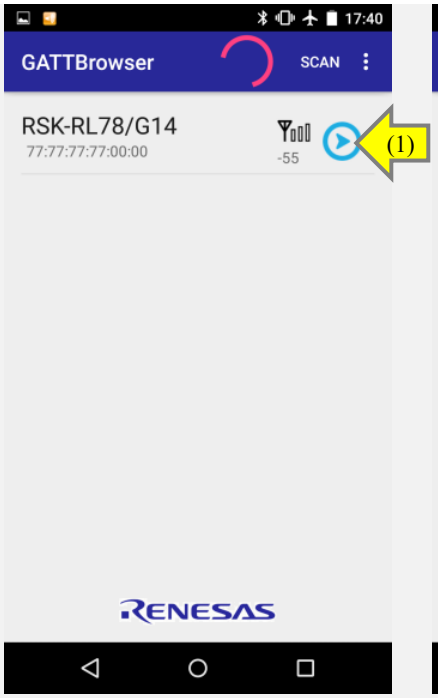

図 A1

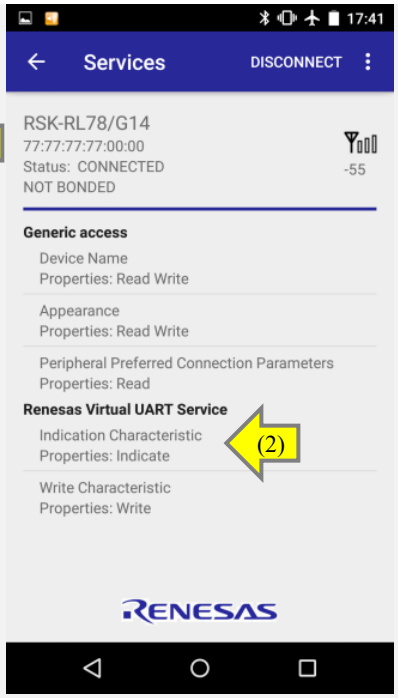

図 A2

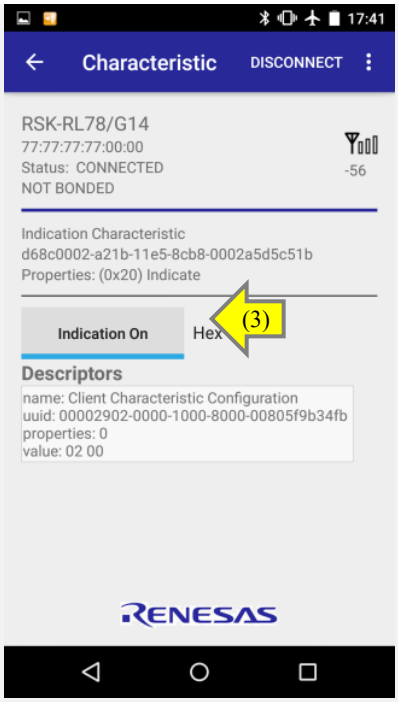

図 A3

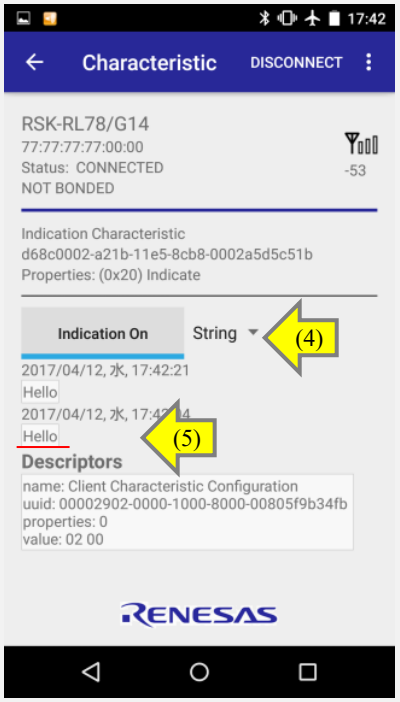

図 A4

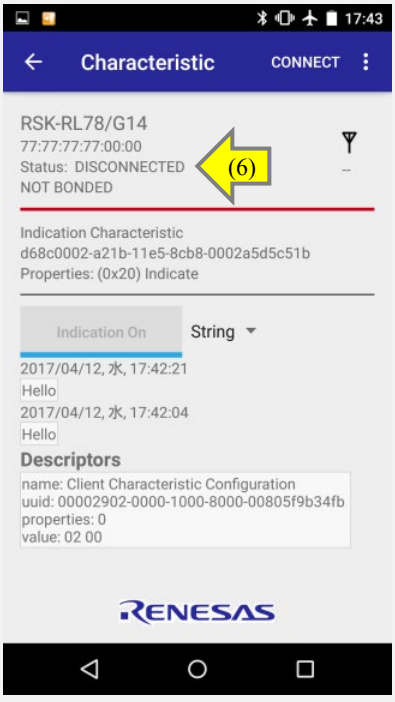

図 A5

4.5 Execution Procedure (iOS)

The following shows the execution procedure when using an iOS device in "Figure 4-2 Operational Check Environment (2)".

1. Supplies power to the local device and launches GATTBrowser on the smartphone.
2. LED 0 (green) on RSK turn on a light.
3. From the device search result, connect to the device indicated as "Renesas-BLE" Note.
   (Figure i1 - Arrow(1)).
   Note: In case of iOS, aliases may be displayed because information is cached in the OS.
4. Confirm that "LocalName: RSK-RL78/G14" is displayed.
   (Figure i2 - Arrow(2)).
5. Tap the arrow of right side to connect.
   (Figure i2 - Arrow(3)).
6. Select "Indication Characteristic" in the Service list.
   (Figure i3 - Arrow(4)).
7. Tap "Enable Indication".
   (Figure i4 - Arrow(5)).
8. Local device and remote device are connected and LED 1 (orange) on RSK turn on alights.
9. Press SW2 on RSK.
10. "Hello" will be displayed.
    (Figure i5 - Arrow(6)).
11. Press SW3 on RSK.
12. The local device and the remote device are disconnected and "Disconnected" is displayed.
    (Figure i6 - Arrow(7)).
13. LED 1 (orange) on RSK turn off a light.

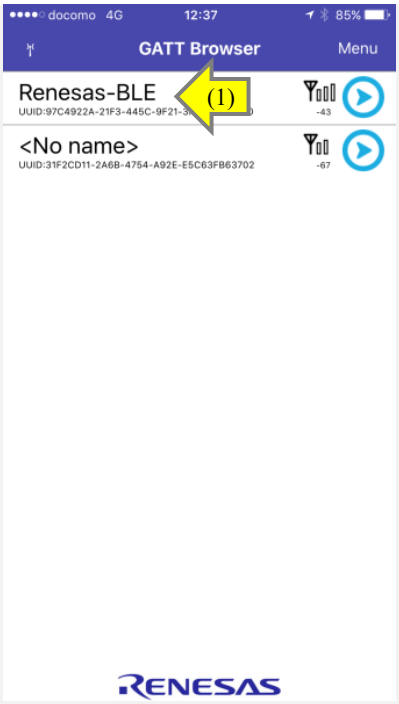

図 i1

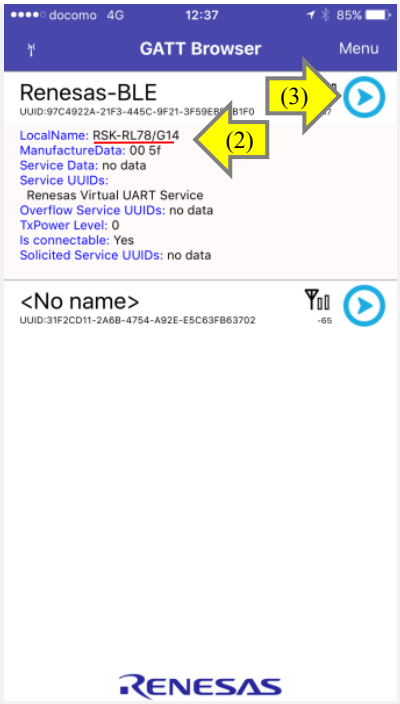

図 i2

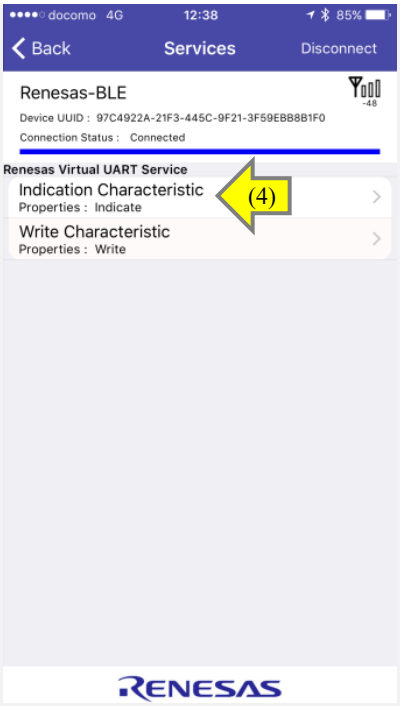

図 i3

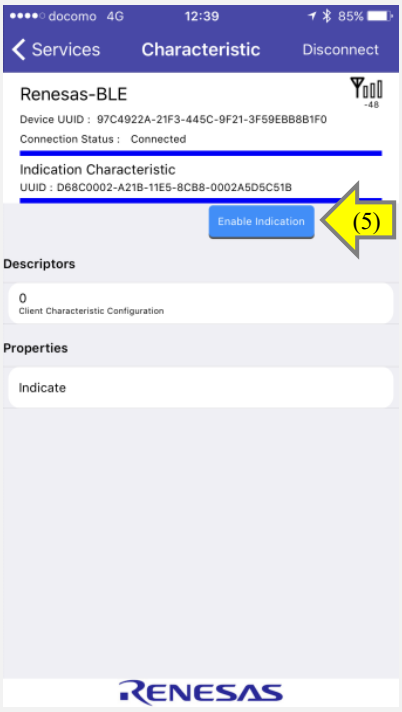

図 i4

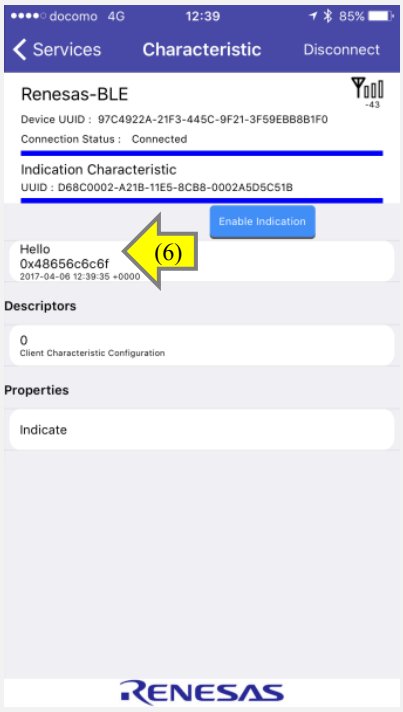

図 i5

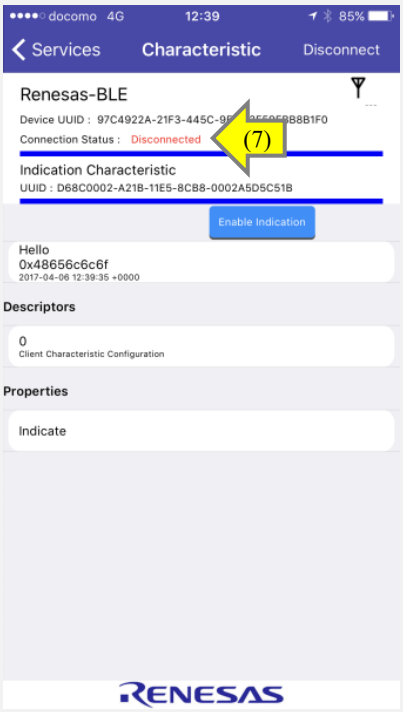

図 i6

5. How to use Simple API

This section explains a simple API program as an example of how to perform BLE communication using simple API. The main function of the simple API program is described in the following source file.

Simple API can not be called from interrupt handling. To use an interrupt as a trigger, set a flag in interrupt handling and call the API outside of interrupt handling.

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

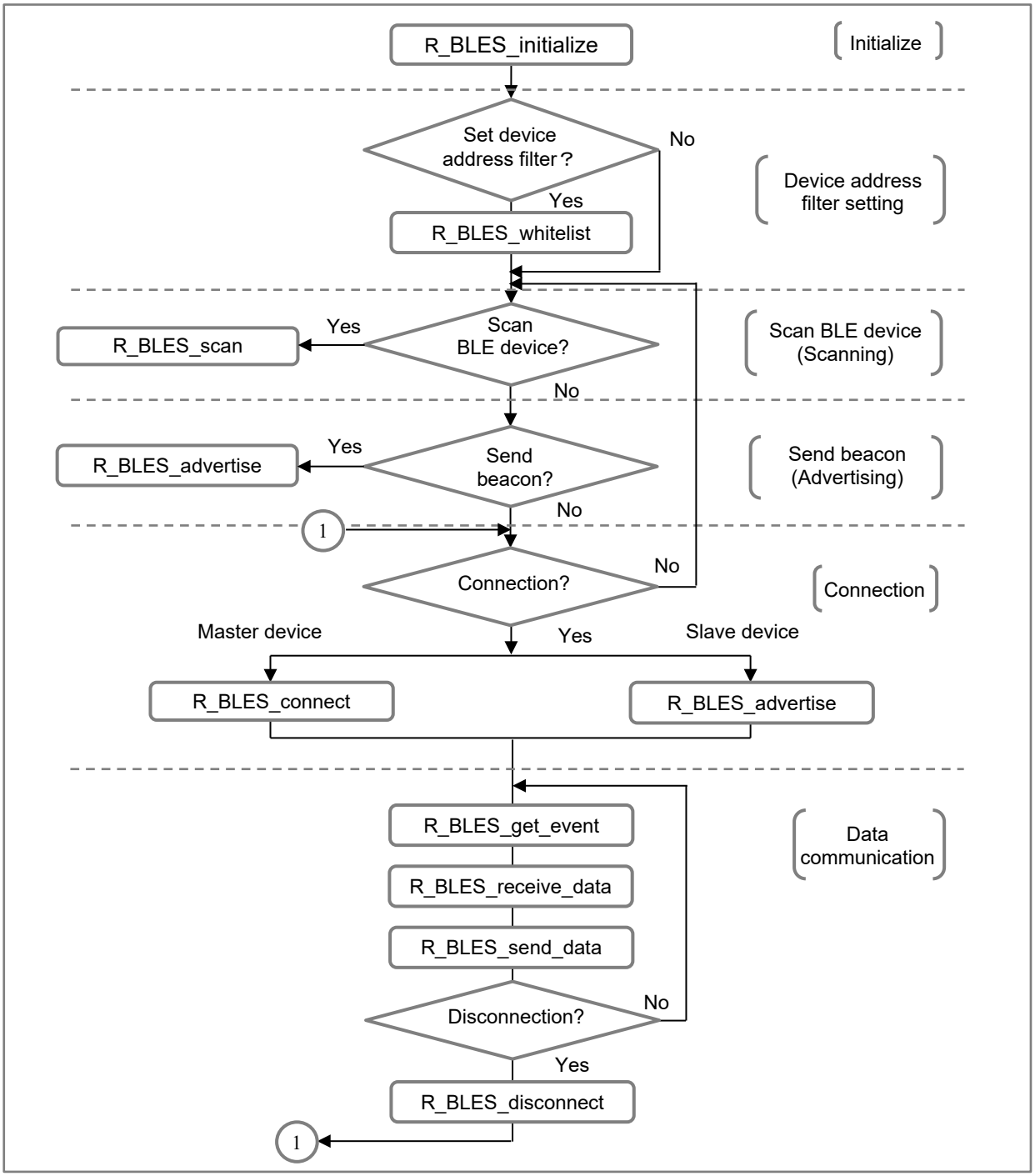

Figure 5-1 Simple API flow chart

5.1 Random Seed

When connecting as a slave device using the simple API, use the pseudo-random value obtained from the rand function to generate the key used in pairing. Before initializing the simple API with the R_BLES_initialize function, set the seed value with the srand function. In the program, "0x12ef" is set as a sample.

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

5.2 Initialize

First, initialize communication between the Host MCU and the BLE MCU and initialize the simple API using the R_BLES_initialize function. Also, make the RL78/G1D module or other services in the BLE software private so that only the generic purpose communication profile can be referenced from the connected remote device.

For function specifications, see "6.1.1 R_BLES_initialize".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

```
R_BLES_initialize();
```

5.3 Device Address Filter Setting

In the following cases, use the R_BLES_whitelist function to set the device address in the whitelist.

- When scanning a specific BLE device issuing a beacon (advertising).
- When a response (scan request) to a beacon (advertising) is accepted only from a specific BLE device.

Specify the device address structure with the first argument. The total number of device addresses that can be set is 12 total, with 6 public addresses and 6 random addresses.

For function specifications, see "6.1.2 R_BLES_whitelist".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

The device address structure registered in the whitelist is shown below. Up to 12 device addresses can be registered. To change the number of device addresses to be registered, change the definition value of "WL_DEVADDR_LIST_NUM". Then, make the device address defined in the structure the same as the definition value.

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

```
/************************************************************************
Whitelist
************************************************************************/
RBLE_WHITELIST  WhiteList = {
    {
        /* Address type      Device address                        Flag */
        {{RBLE_ADDR_PUBLIC, {0x51, 0x55, 0x77, 0x77, 0x77, 0x77}}, TRUE},
        {{RBLE_ADDR_PUBLIC, {0x02, 0x90, 0xa0, 0x23, 0x07, 0x10}}, TRUE},
        :
        {{RBLE_ADDR_RAND,   {0x01, 0x00, 0xde, 0xfa, 0xfe, 0xca}}, TRUE},
        {{RBLE_ADDR_RAND,   {0x02, 0x00, 0xde, 0xfa, 0xfe, 0xca}}, TRUE},
        :
    }                                                   Define the same number of
};                                                      device addresses as
                                                        WL_DEVADDR_LIST_NUM.
```

The definition of device address number is shown below.

Source file: ble_simple_api_rl78g14\src\rBLE\src\rbles_api\rbles_api.h

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

5.4 Scanning for Peripheral Devices

Use the R_BLES_scan function to search (scaning) BLE devices that issue beacons (advertising). In the first argument, specify the parameter structure that sets the scanning operation. The second argument stores the advertising information of the peripheral devices found in the search. The third argument is the scanning execution time. After calling the R_BLES_scan function, when the advertising information storage buffer specified by the second argument becomes full or the scanning execution time specified by the third argument elapses, it returns from the function.

For function specifications, see "6.1.3 R_BLES_scan".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

To change the size of the structure storing the advertising information, change the "ADV_REPORT_LIST_NUM" definition value. The definition of the number of advertising information is shown below.

Source file: ble_simple_api_rl78g14\src\rBLE\src\rbles_api\rbles_api.h

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

5.5 Beacon Transmission

Send a beacon (advertising) using the R_BLES_advertise function. The first argument specifies the parameter structure that sets up the advertising behavior. The second argument is the advertising execution time. If you do not connect to a remote device like a beacon, it returns from the function after the advertisement execution time has elapsed.

For function specifications, see "6.1.4 R_BLES_advertise".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

The recommended parameter setting when using as a beacon is shown below.

Table 5-1 Beacon recommended setting value

Structure member	Definition	Description
disc_mode	RBLE_GAP_BROADCASTER	Broadcast data by advertising
conn_mode	0	Operates as a Broadcaster
adv_type	RBLE_GAP_ADV_NONCONN_UNDIR	Only information sent from Advertiser

5.6 Connection

Connect to the remote device. When a local device connects to a remote device as a master device, use the R_BLES_connect function. The first argument specifies the parameter structure that sets the behavior when connecting. When the connection is completed, the connection information is stored in the second argument. The third argument is the connection execution time. After calling the R_BLES_connect function, the function returns when the connection with the remote device is completed or the connection execution time has elapsed. If the return value of the function is "RLBE_OK", the connection is completed. After completing the connection, execute "5.7 Data Communication" processing.

For function specifications, see "6.1.5 R_BLES_connect".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

The connection requires the device address of the remote device (connection partner), and there are two kinds of specification methods.

(1) Specify the remote device with the connection parameter of the first argument without using the white address.

(2) Use the whitelist and specify the remote device in the whitelist.

(1): Set the remote device's address type and device address in peer_addr_type and peer_addr of the first argument.

(2): Set the device address of the peer device in the whitelist. For the setting method, refer to "5.3 Device Address Filter Setting".

If local device is connecting to a remote device as a slave device, use the R_BLES_advertise function. The first argument specifies the parameter structure that sets up the advertising behavior. The second argument is the advertising (connection) execution time. After calling the R_BLES_advertise function, it returns from the function when the connection with the remote device is completed or when the advertising (connection) execution time elapses. If the return value of the function is "RBLE_CONNECTED", the connection is completed. After completing the connection, execute "5.7 Data Communication" processing.

For function specifications, see "6.1.4 R_BLES_advertise".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

5.7 Data Communication

After completing connection with the remote device, data communication is performed. Data communication is performed with the following three functions

- R_BLES_get_event function to acquire events occurring in data communication.
- R_BLES_receive_data function to retrieve data received by BLE.
- R_BLES_send_data function to send data.

The R_BLES_get_event function must be called regularly during data communication after the connection is completed. Events generated by data communication are stored in the variable of the first argument. There are two types of events, "RBLES_EVENT_DISCONNECT" (disconnection) and "RBLES_EVENT_RECEIVE_DATA" (data reception). The "RBLES_EVENT_DISCONNECT" event is an event indicating that the connection with the remote device has been disconnected. If this event occurs, exit the loop of data communication and reconnect again. The "RBLES_EVENT_RECEIVE_DATA" event is an event indicating that data has been received. When this event occurs, read the received data with the R_BLES_receive_data function.

The R_BLES_receive_data function specifies the buffer that stores received data in the first argument. The second argument is the number of data to read. Returns the number of data that could be read into the return value of the function.

For data transmission, use the R_BLES_send_data function. For data transmission, prepare the data transmission request variable "r_send_data_req", prepare the data to be transmitted by AD conversion interrupt, Switch interrupt, etc. and set the data transmission request flag. Check the data transmission request flag in the loop processing in which the R_BLES_get_event function is executed and call the R_BLES_send_data function. Specify the transmit data buffer as the first argument. The second argument is the number of transmitted data.

The R_BLES_disconnect function is used to terminate the connection with the remote device when terminating data communication. Prepare disconnection request variable "r_disconnect_req" and set disconnection request flag by SW interrupt etc. Check the disconnection request flag in the loop processing in which the R_BLES_get_event function is executed and call the R_BLES_disconnect function. Please reconnect from the connection after disconnection.

For function specifications, see "6.1.6 R_BLES_get_event", "6.1.7 R_BLES_send_data", "6.1.8 R_BLES_receive_data" and "6.1.9 R_BLES_disconnect".

Example)

Source file: ble_simple_api_rl78g14\src\ble_simple_api_rl78g14.c

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

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

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

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

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

If the reception data can not be retrieved in time, the internal buffer of the simple API that stores the received data may overflow. Overflowed data will be discarded. To change the size of the internal buffer, change the definition value of "RBLES_RDBUF_SIZE".

Source file: ble_simple_api_rl78g14\src\rBLE\src\rbles_api\rbles_api.h

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

6. Simple API Specification

It shows API, structure, and macro specifications defined by simple API. When using the simple API, it is not possible to call rBLE API individually from the user application. Use only simple API. Also, simple API can't be called from interrupt handling. Call it after leaving interrupt processing.

6.1 API

10.	R_BLES_initialize()	Initialize of simple API
11.	R_BLES_whitelist()	Set device address to white list
12.	R_BLES_scan()	Execute of scanning
13.	R_BLES_advertise()	Execute of advertising and connect to master device
14.	R_BLES_connect()	Connect to slave device
15.	R_BLES_get_event()	Get events
16.	R_BLES_send_data()	Send data
17.	R_BLES_receive_data()	Receive data
18.	R_BLES_disconnect()	Disconnection between remote device

6.1.1 R_BLES_initialize

RBLE_STATUS R_BLES_initialize(void)

Initialize between Host MCU and BLE MCU communication, initialize simple API. Hide the RL78/G1D module or other services in the BLE software so that you can refer only to the general purpose communication profile from the connected remote device. It returns from this API at the completion of initialization of simple API.

Note: This function can't be used in interrupt processing.

Parameters:

none

Return:

RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

6.1.2 R_BLES_whitelist

RBLE_STATUS R_BLES_whitelist(RBLE_WHITELIST *whitelist)

Add BD address to white list. It returns from this API at the completion of adding BD address to white list.

Note: This function can't be used in interrupt processing.

Parameters:

*whitelist			BD addresses are added to the white list
	dev_info		BD address information
		dev_addr_type	BD address type RBLE_ADDR_PUBLIC: Public BD address RBLE_ADDR_RAND: Random BD address
		dev_addr	BD address
	dev_en		BD address available flag TRUE: available BD address (It is added to white list) FALSE: unavailable BD address (It is not added to white list)

Return:

RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_PARAM_ERR	0xF3	Invalid parameter

6.1.3 R_BLES_scan

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

Executes scanning for the time specified by duration. Store advertising report acquired during scanning execution in *adv_report. When *adv_report is full, interrupt scanning. It returns from this API at scanning execution time expired or advertising report list is full.

Note: This function can't be used in interrupt processing.

Parameters:

*scan_param		Scanning parameter
	scan_type	Scanning type RBLE_SCAN_PASSIVE: Executes passive scanning. (No SCAN_REQ packets shall be sent.) RBLE_SCAN_ACTIVE: Executes active scanning. (SCAN_REQ packets may be sent.)
	scan_intv	Scan interval N = 2 - 10240 (Time = 2.5 msec - 10240 msec (2.5 msec - 10.24 sec)) * If N = 2 is specified, the scan interval is 2.5 msec
	scan_window	Scan window size N = 2 - 10240 (Time = 2.5 msec - 10240 msec (2.5 msec - 10.24 sec)) * If N = 2 is specified, the scan interval is 2.5 msec * Be sure to set "scan interval > scan window size"
	own_addr_type	Local device address RBLE_ADDR_PUBLIC: Public BD address RBLE_ADDR_RAND: Random BD address
	scan_filt_policy	Scanning filter policy RBLE_SCAN_ALLOW_ADV_ALL: Accept all advertising packets. RBLE_SCAN_ALLOW_ADV_WLST: Accept advertising packets in white list only. RLBE_SCAN_ALLOW_VUART_SRV: Accept advertising packets included UUID of the general

		purpose communication profile (virtual UART profile).
	filter_dup	Duplicate filter RBLE_SCAN_FILT_DUPLIC_DIS: Disables duplicated filtering of received data. RBLE_SCAN_FILT_DUPLIC_EN: Enables duplicated filtering of received data.
*adv_report		Advertising report
	adv_list[]	Advertising report list
	evt_type	Advertising event type 0x00: Connectable undirected advertising 0x01: Connectable directed advertising 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising 0x04: Scan Response
	adv_addr_type	BD address type of advertiser RBLE_ADDR_PUBLIC: Public BD address RBLE_ADDR_RAND: Random BD address
	adv_addr	BD address of advertiser
	data_len	Advertising data length
	data[]	Advertising or scan response data Note: For details about the advertising and scan response data formats, see Bluetooth Low Energy Protocol Stack User's Manual.
	rssi	RSSI when advertising data is received
	adv_list_num	Number of received advertising reports
duration		Scanning execution time N = 1 - 60000 (Time = N × 10 msec (10 msec - 600 sec))

Return:

RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_PARAM_ERR	0xF3	Invalid parameter
RBLE_ADVLIST_FULL	0xFA	Advertising report list is full

6.1.4 R_BLES_advertise

RBLE_STATUS R_BLES_advertise(RBLE_BROADCAST_ENABLE_PARAM *adv_param, uint16_t duration)

Execute advertising for the time specified by duration. When advertising to allow connection, connect with the master device. It returns from this API at advertising execution time expired or connection completion with master device.

Note: This function can't be used in interrupt processing.

Parameters:

*adv_param		Advertising parameter
	disc_mode	Discovery mode RBLE_GAP_NON_DISCOVERABLE: Not discoverable by any device performing either the general discovery procedure or the limited discovery procedure RBLE_GAP_GEN_DISCOVERABLE: Discoverable by devices performing the general discovery procedure RBLE_GAP_LIM_DISCOVERABLE: Discoverable for a limited period of time by other devices performing the limited or general device discovery procedure RBLE_GAP_BROADCASTER: Data is broadcast by an Advertising event
	conn_mode	Connection mode

0	Operates as a Broadcaster
RBLE_GAP_NON_CONNECTABLE	Connection not allowed
RBLE_GAP_UND_CONNECTABLE	Connectable
RBLE_GAP_DIR_CONNECTABLE	Only connectable with a known device

adv_info

adv_intv_min	Please set the same value as adv_intv_max.
adv_intv_max	Maximum advertising interval N = 20 - 10240 (Time = 20 msec - 10240 msec (20 msec - 10.24 sec))
adv_type	Advertising type

RBLE_GAP_ADV_CONN_UNDIR	Can respond to CONNECT_REQ or SCAN_REQ
RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY	Only connectable with specified device
RBLE_GAP_ADV_DISC_UNDIR	Can respond to SCAN_REQ
RBLE_GAP_ADV_NONCONN_UNDIR	Only information sent from Advertiser
RBLE_GAP_ADV_CONN_DIR_LOW_DUTY	Only connectable with specified device

own_addr_type	Own BD address type

RBLE_ADDR_PUBLIC	Public BD address
RBLE_ADDR_RAND	Random BD address

direct_addr_type	Direct connection address type

RBLE_ADDR_PUBLIC	Public BD address
RBLE_ADDR_RAND	Random BD address

direct_addr	Direct connection address * It is enabled when adv_type selects RBLE_GAP_ADV_CONN_DIR_HIGH_DUTY or RBLE_GAP_ADV_CONN_DIR_LOW_DUTY.
adv_chnl_map	Advertising channel

RBLE_ADV_CHANNEL_37	Use channel 37
RBLE_ADV_CHANNEL_38	Use channel 38
RBLE_ADV_CHANNEL_39	Use channel 39
RBLE_ADV_ALL_CHANNELS	Use all channels (37, 38, and 39)

adv_filt_policy	Advertising filter policy

RBLE_ADV_ALLOW_SCAN_ANY_CON_ANY	Allow SCAN_REQ from any. Allow CONNECT_REQ from any.
RBLE_ADV_ALLOW_SCAN_WLST_CON_ANY	Allow SCAN_REQ from white list only. Allow CONNECT_REQ from any.
RBLE_ADV_ALLOW_SCAN_ANY_CON_WLST	Allow SCAN_REQ from any. Allow CONNECT_REQ from white list only.
RBLE_ADV_ALLOW_SCAN_WLST_CON_WLST	Allow SCAN_REQ from white list only. Allow CONNECT_REQ from white list only.

adv_data_len	Advertising data length
adv_data	Advertising data
scan_rsp_data_len	Scan response data length
data	Scan response data

duration	Advertising execution time N = 1 - 60000 (Time = N × 10 msec (10 msec - 600 sec))

Return:

RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_PARAM_ERR	0xF3	Invalid parameter

RBLE_CONNECTED	0xF9	Connection completion with master device

6.1.5 R_BLES_connect

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

Connect with remote device. Connection result information is stored to *conn_info. It will time out when the time specified by duration expired without connecting to the remote device. It returns from this API at connection execution time expired or connection completion with remote device.

Note: This function can't be used in interrupt processing.

Parameters:

*conn_param		Connection parameter
	scan_intv	Scan interval N = 2 - 10240 (Time = 2.5 msec - 10240 msec (2.5 msec - 10.24 sec)) * If N = 2 is specified, the scan interval is 2.5 msec
	scan_window	Scan window N = 2 - 10240 (Time = 2.5 msec - 10240 msec (2.5 msec - 10.24 sec)) * If N = 2 is specified, the scan interval is 2.5 msec * Be sure to set "scan interval > scan window size"
	init_filt_policy	Initiator filter policy

RBLE_GAP_INIT_FILT_IGNORE_WLST	Connect to the device specified by peer_addr_type, peer_addr without using the White List.
RBLE_GAP_INIT_FILT_USE_WLST	Use the White List to connect to the device registered in the White List. (peer_addr_type, peer_addr is ignored.)

	peer_addr_type	Remote BD address type RBLE_ADDR_PUBLIC: Public BD address RBLE_ADDR_RAND: Random BD address * This parameter is only available when init_filt_policy is RBLE_GAP_INIT_FILT_IGNORE_WLST.
	peer_addr	Remote BD address This parameter is only available when init_filt_policy is RBLE_GAP_INIT_FILT_IGNORE_WLST.
	own_addr_type	Local BD address RBLE_ADDR_PUBLIC: Public BD address RBLE_ADDR_RAND: Random BD address
	con_intv_min	Please set the same value as con_intv_max.
	con_intv_max	Maximum connection interval N = 7 - 4000 (Time = 7.5 msec - 4000 msec (7.5 msec - 4 sec))
	con_latency	Connection slave latency N = 0000 ～ 499
	superv_to	Super vision timeout N = 100 -32000 (Time = 100 msec - 32000 msec (100 msec - 32 sec))
*conn_info		Connection result information
	status	reserved
	role	reserved
	conhdl	reserved
	peer_addr_type	Remote BD address type RBLE_ADDR_PUBLIC: Public BD address RBLE_ADDR_RAND: Random BD address
	peer_addr	Remote BD address
	idx	reserved
	con_interval	Connection interval N = 0x0006 - 0x0C80 (Time = N x 1.25 msec (7.5 msec - 4 sec))

	con_latency	Connection slave latency N = 0000 - 499
	sup_to	Super vision timeout N = 0x000A - 0x0C80 (Time = N x 10 msec (100 msec-32 sec))
	clk_accuracy	Clock accuracy 500 ppm: 0 250 ppm: 1 150 ppm: 2 100 ppm: 3 75 ppm: 4 50 ppm: 5 30 ppm: 6 20 ppm: 7
	duration	Connection execution time N = 1 - 60000 (Time = N × 10 msec (10 msec - 600 sec))

Return:

RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_PARAM_ERR	0xF3	Invalid parameter
RBLE_TIMEOUT	0xFB	Connection timeout

6.1.6 R_BLES_get_event

RBLE_STATUS R_BLES_get_event(uint8_t *evt)

It notifies the event of data reception and disconnection from the connected remote device. After connecting with the remote device, must call this function periodically.

Note: This function can't be used in interrupt processing.

Parameters:

*evt	Event RBLES_EVENT_NONE: 0x00: No event RBLES_EVENT_DISCONNECT: 0x01: Disconnected from remote device RBLES_EVENT_RECEIVE_DATA: 0x02: Received data

Return:

RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_PARAM_ERR	0xF3	Invalid parameter

6.1.7 R_BLES_send_data

RBLE_STATUS R_BLES_send_data(uint8_t *txbuf, uint8_t len)		
Send data to the connected remote device. When connected as a master device, it sends data with Write Request and returns from this API upon receiving Write Response. When connected as a slave device, data is transmitted at Indication and return from this API at reception of Confirmation. Note: This function can't be used in interrupt processing.		
Parameters:		
*txbuf	Transmission data buffer	
len	Number of transmission data (Maximum 20 bytes)	
Return:		
RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_PARAM_ERR	0xF3	Invalid parameter

6.1.8 R_BLES_receive_data

uint16_t R_BLES_receive_data(uint8_t *rxbuf, uint16_t len)	
Retrieve received data stored in the internal buffer. If the received data stored in the internal buffer is less than the number of bytes specified by len, the portion stored in the internal buffer is stored in *rxbuf. Note: This function can't be used in interrupt processing.	
Parameters:	
*rxbuf	Reception data buffer
len	Number of getting data (byte)
Return:	
0	No reception data
0 以外	Number of getting data

6.1.9 R_BLES_disconnect

RBLE_STATUS R_BLES_disconnect(void)		
Disconnect from the remote device. It returns from this API when disconnection with the remote device is completed. Note: This function can't be used in interrupt processing.		
Parameters:		
none		
Return:		
RBLE_OK	0x00	Success
RBLE_ERR	0xF0	Sequence error
RBLE_TRANS_ERR	0xF1	Communication error between Host MCU and BLE MCU
RBLE_STATUS_ERROR	0xF2	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

6.2 Structure

6.2.1 RBLE_BROADCAST_ENABLE_PARAM

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

6.2.2 RBLE_ADV_INFO

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

6.2.3 RBLE_SET_ADV_PARAM

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

6.2.4 RBLE_BD_ADDR

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

6.2.5 RBLE_SET_ADV_DATA

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

6.2.6 RBLE_ADV_DATA

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

6.2.7 RBLE_SET_SCAN_RSP_DATA

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

6.2.8 RBLE_SCAN_RSP_DATA

Create Connection Command parameters structure		
uint16_t	scan_intv	Scan interval
uint16_t	scan_window	Scan window size
uint8_t	init_filt_policy	Initiator filter policy
uint8_t	peer_addr_type	Peer address type
RBLE_BD_ADDR	peer_addr	Peer BD address
uint8_t	own_addr_type	Own address type
uint8_t	reserved	-
uint16_t	con_intv_min	Minimum of connection interval
uint16_t	con_intv_max	Maximum of connection interval
uint16_t	con_latency	Connection latency

uint16_t	superv_to	Link supervision timeout
uint16_t	ce_len_min	Minimum CE length
uint16_t	ce_len_max	Maximum CE length

6.2.9 RBLE_CONNECT_INFO

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

6.2.10 RBLE_SCANNING_INFO

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

6.2.11 RBLE_SET_SCAN_PARAMETER

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

6.2.12 RBLE_WHITELIST

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

6.2.13 RBLE_WLIST_DEV_ADDR

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

6.2.14 RBLE_DEV_ADDR_INFO

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

6.3 Macro

6.3.1 ADV_REPORT_LIST_NUM

ADV_REPORT_LIST_NUM	Number of advertising report list (default value : 30)

6.3.2 WL_DEVADDR_LIST_NUM

WL_DEVADDR_LIST_NUM	Number of BD address for white list (Maximum 12) (default value : 12)

6.3.3 RBLES_RDBUF_SIZE

RBLES_RDBUF_SIZE	Number of internal receive buffer (byte) (default value : 100)

6.4 Pairing Information

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

7. CAUTIONS

7.1 Code Generation (r_cg_macrodriver.h)

You can update the low level peripheral drivers by using Code Generation Tool. Following file is updated in this project to avoid the type declaration confliction.

- r_cg_macrodriver.h

Since the corrected file is placed in the code generation folder of each integrated development environment project, delete the generated r_cg_macrodriver.h and rename the corrected file before use.

Table 7-1 Modified r_cg_macrodriver.h

IDE	Code generation folder	Corrected r_cg_macrodriver.h name
CS+ for CC	ble_simple_api_rl78g14\project\CC_CCRL\cg	r_cg_macrodriver_g14_ccrl.h
e^2 studio	ble_simple_api_rl78g14\project\e2studio\src	r_cg_macrodriver_g14_ccrl.h

7.2 About calling simple API

Simple API can not be called from interrupt handling. Since it is communicating with the BLE MCU using the UART interrupt in the simple API, calling the simple API from the interrupt handling will not return from the API.

8. Appendix

8.1 Change of BLE software

The software included in this application note is initialized to be used in "Figure 2-1 System configuration (1) (RY7011)". Please change the setting of the Host MCU program (Simple API program) and BLE MCU program (BLE software) as described in this chapter for use in "Figure 2-2 System configuration (2) (BLE Evaluation Board)".

Files that were built using the CS+ for CC in this chapter are included in this application note. For details, refer to "3.5.3 Folder Structure of Execution File".

- Host Environment

  Change of simple API program

  - Change of UART connection : 2-wire with branch connection → 2-wire connection
  - Change of UART baud rate : 115200 bps → 4800 bps

- Host Environment

  Change of BLE software

  - Added general purpose communication database

8.1.1 Preparation of Host MCU Program

Change the baud rate of UART and UART connection method between Host MCU and BLE MCU of simple API program. The baud rate of the UART is changed from the initial baud rate (115,200 bps) of the RL78/G1D module to the initial baud rate (4800 bps) of the modem configuration BLE software using the code generation tool of the integrated development environment. Then, change the connection method from UART 2-wire branch connection method to UART 2-wire connection method.

There are notes on code generation. In CS + for CC, e2 studio modified "r_cg_macrodriver.h" to avoid duplication of type declarations. For the correction method, refer to "7.1 Code Generation".

(1) **Change baud rate of simple API program**

- CS+ for CA, CX / CS+ for CC
  1. Double click the project file shown in "Table 8-1".
  2. Open [ble_simple_api_rl78g14 (project)] tree, and open [Code Generator (Design Tool)] tree. Double click [Serial], then open [Code Generator] tab.
  3. Choose [Code Generator] tab → [SAU1] tab → [UART2] tab → [Receive] tab and [Transmit] tab. Change baud rate from 115200 bps to 4800 bps.
  4. Push [Generate Code] button and generate code.

- e² studio
  1. Launch e2 studio.
  2. Right click on [Project Explorer] and select [Import…] from the dropdown menu.
  3. [Import] window is popped up and select [Existing Projects into Workspace] and click [Next >].
  4. Fill [Select root directory:] form with the project directory shown in "Table 8-1". Make sure that the project you selected is displayed in [Projects:] and click [Finish]. Then the windows is closed.
  5. Open [ble_simple_api_rl78g14] tree, and open [Code Generator] tree, and open [Peripheral Functions] tree. Double click [Serial], then open [Peripheral Functions] tab.

6. Choose [SAU1] tab → [UART2] tab → [Receive] tab and [Transmit] tab. Change baud rate from 115200 bps to 4800 bps.
7. Push [Generate Code] button and generate code.

(2) **Change UART connection method**

Change the UART connection method to 2 wire connection method. Change the definition value defined in "uart.h" as follows.

File:

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

Definition value:

#define SERIAL_U_DIV_2WIRE (1) → (0)

(3) **Change definition macro of IDE**

Change the definition macro in the project of the integrated development environment to be used as follows.

USE_MODULE_RY7011 → noUSE_MODULE_RY7011

USE_FW_UPDATE_PROFILE → noUSE_FW_UPDATE_PROFILE

8.1.2 Build of Host MCU Program

Indicates how to build the Host MCU program for each IDE.

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

1. Double click the project file shown in "Table 8-1".
2. Right click on [ble_simple_api_rl78g14 (Project)] in [Project Tree] and select [Build ble_simple_api_rl78g14] from the dropdown menu.
3. Refer "Table 8-1" for the execution file generate path.

(2) **e² studio**

1. Launch e² studio.
2. Right click on [Project Explorer] and select [Import…] from the dropdown menu .
3. [Import] window is popped up and select [Existing Projects into Workspace] and click [Next >].
4. Fill [Select root directory:] form with the project folder shown in "Table 8-1".
5. Make sure that the project you selected is displayed in [Projects:] and click [Finish]. Then the windows is closed.
6. Right click on the project just imported on [Project Explorer] and Select [Build Project] from the dropdown menu.
7. Refer "Table 8-1" for the execution file generate path.

Table 8-1 Host MCU project file and execution file generation directory

CS+ for CC		
	Project file	ble_simple_api_rl78g14\project\CS_CA\ble_simple_api_rl78g14.mtpj
	Execution file	ble_simple_api_rl78g14\project\CS_CA\DefaultBuild\ble_simple_api_rl78g14.hex
CS+ for CA,CX		
	Project file	ble_simple_api_rl78g14\project\CS_CCRL\ble_simple_api_rl78g14.mtpj
	Execution file	ble_simple_api_rl78g14\project\CS_CCRL\DefaultBuild\ble_simple_api_rl78g14.hex
e² studio		
	Project folder	ble_simple_api_rl78g14\project\e2studio
	Execution file	ble_simple_api_rl78g14\project\e2studio\HardwareDebug\ble_simple_api_rl78g14.hex

(3) **Write executable file and board setting**

1. Refer to "Table 8-2" and set jumpers on the RSK board.
2. Connect E1 emulator to RSK and connect E1 emulator to PC.
3. Connect AC power adapter to RSK and supply power from AC power adapter to RSK.
4. Start RFP (Renesas Flash Programmer) and create workspace by selecting [File] → [Create New Project], select [RL78] as [Microcontroller] and push [Connect] button.
5. Select execution file on [Operation] tab → [Program File].
6. Push [Start] button on [Operation] tab to start writing and confirm that [SUCCESS] is displayed.
7. Remove AC power adapter and E1 emulator from RSK board.

Table 8-2 Jumper Setting

JumperJ5 setting	JumperJ6 setting	Power source	Input voltage	Regulator supply voltage
Pin2-3 shorted	open	PWR connector	5V	3.3V

8.1.3 Preparation of BLE MCU Program

Simple API uses the general purpose communication profile. It is necessary to incorporate a database of the general purpose communication profile in BLE software used in BLE MCU.

(1) **Preparation of BLE software source code**

Download the EEPROM Emulation Library and Code Flash Library from the Renesas web site, and copy the library to following directories.

For details on installing BLE software, refer to "4. Installing Software" in "Quick Start Guide (R01AN2767)".

- EEPROM Emulation Library
  - CC-RL

    RL78_G1D\Project_Source\renesas\src\driver\dataflash\cc_rl
  - CA78K0R

    RL78_G1D\Project_Source\renesas\src\driver\dataflash\cs

- Code Flash Library
  - CC-RL

    RL78_G1D\Project_Source\renesas\src\driver\codeflash\cc_rl
  - CA78K0R

    RL78_G1D\Project_Source\renesas\src\driver\codeflash\cs

(2) **Addition of general purpose communication database**

The simple API program package contains the difference file necessary for changing BLE software. Overwrite the copy source "Project_Source" folder to the copy destination folder of BLE software.

Source folder (Simple API package):

BLE_Software_Ver_1_20_diff_vuart\RL78_G1D\Project_Source

Copy destination folder (BLE software):

Renesas\BLE_Software_Ver_X_XX\RL78_G1D

The difference file is shown below. The difference is enclosed in "#ifdef USE_VUART_PROFILE - # endif".

Table 8-3 Difference files of general purpose communication database

Folder	File	Description
RL78_G1D\Project_Source\renesas\src\arch\rl78	db_handle.h	Definition of general purpose communication Attribute database handles
	prf_config.c	Definition of general purpose communication Attribute database index
	prf_config.h	Definition of general purpose communication service
RL78_G1D\Project_Source\rBLE\src\include	rble_api_vuart.h	(New file) General purpose communication header file

(3) **Change of project definition macro of BLE software**

Add the following definition macro to the modem configuration project of the integrated development environment to be used.

USE_VUART_PROFILE

8.1.4 Build of BLE MCU Program

This section shows how to build a program that runs on the BLE MCU RL78/G1D evaluation board.

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

1. Double click the project file shown in "Table 8-4".
2. Right click on “BLE_Emb” in “Project Tree” and select “Build BLE_Emb” from the dropdown menu.
3. Refer "Table 8-4" for the execution file generate path.

(2) **e² studio**

1. Launch e2 studio.
2. Right click on [Project Explorer] and select [Import…] from the dropdown menu.
3. [Import] window is popped up and select [Existing Projects into Workspace] and click [Next >].
4. Fill [Select root directory:] form with the project folder shown in "Table 8-4". Make sure that the project you selected is displayed in [Projects:] and click [Finish]. Then the windows is closed.
5. Right click on the project on [Project Explorer] and select [Build Project] from the dropdown menu.
6. Refer "Table 8-4" for the execuiton file generate path.

Table 8-4 BLE MCU project file and execution file generation directory

CS+ for CC		
	Project file	RL78_G1D\Project_Source\renesas\tools\project\CS_CCRL\BLE_Modem\BLE_Modem.mtpj
	Execution file	RL78_G1D\Project_Source\renesas\tools\project\CS_CCRL\BLE_Modem\rBLE_Mdm\DefaultBuild\rBLE_Mdm_CCRL.hex
CS+ for CA,CX		
	Project file	RL78_G1D\Project_Source\renesas\tools\project\CubeSuite\BLE_Modem\BLE_Modem.mtpj
	Execution file	RL78_G1D\Project_Source\renesas\tools\project\CubeSuite\BLE_Modem\rBLE_emb\DefaultBuild\rBLE_emb.hex
e² studio		
	Project folder	RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem
	Execution file	RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem\rBLE_Mdm\DefaultBuild\rBLE_Mdm_CCRL.hex

(3) **Write executable file and board setting**

1. Refer to the "Table 8-5 Switch Setting" and set the slide switch of the RL78/G1D evaluation board
2. Connect E1 emulator to BLE Evaluation Board and connect E1 emulator to PC.
3. Connect the power source to BLE Evaluation Board and supply power to BLE Evaluation Board.
4. Start RFP (Renesas Flash Programmer) and create workspace by selecting [File] → [Create New Project]. And then select [RL78] as [Microcontroller] and push [Connect] button.
5. Uncheck Erase and P.V checkbox on the [Block Setting] tab → [Code Flash 1] → [Block255] and all blocks of [Data Flash 1].
6. Select [Erase] and [Write] in [Command] on the [Operation Setting] tab.
7. Select execution file on [Operation] tab → [Program File].
8. Push [Start] button on [Operation] tab to start writing and confirm that [SUCCESS] is displayed.
9. Remove the power source and E1 emulator from BLE Evaluation Board.

Table 8-5 Switch Setting

Switch	Setting	Function
SW7	2-3 connected (right) <Default>	Power supplied from AC adapter or USB via regulator
SW8	1-2 connected (left) <Default>	Power supplied from AC adapter Note: If power supplied from USB, connect 2-3 (Right)
SW9	1-2 connected (left)	Connected to an external extension interface.
SW10	1-2 connected (left) <Default>	Power supplied to the module.
SW11	2-3 connected (right) <Default>	Power supplied from a source other than the E1 debugger.
SW12	2-3 connected (right) <Default>	(fixed default)
SW13	1-2 connected (left) <Default>	Connected to USB interface.

8.1.5 Connection of Host MCU and BLE MCU

The procedure for connecting the Host MCU and the BLE MCU is shown below.

(1) **Connection between RSK and Module Evaluation board**

Refer to "Table 8-6" and connect pins RSK board and BLE Evaluation Board by wires.

Table 8-6 Pin connection

RL78/G14 ports (RSK pins)	Module Evaluation Board	Purpose
TXD2(J3-Pin16)	RxD0(TH19)	UART(Host MCU→BLE MCU)
RXD2(J3-Pin15)	TxD0(TH23)	UART(BLE MCU→Host MCU)
Vss(GND1)	GND1 or GND2 or GND3	Ground

Note: Short the jumper (TH18 - TH19) of the module evaluation board.

(2) **Connection between RSK and RL78/G1D Evaluation board**

Refer to "Table 8-7" and connect pins RSK board and BLE Evaluation Board by wires.

Table 8-7 Pin connection

RL78/G14 ports (RSK pins)	RL78/G1D ports (board pins)	Purpose
TXD2(J3-Pin16)	RxD0(CN4-Pin16)	UART(Host MCU→BLE MCU)
RXD2(J3-Pin15)	TxD0(CN4-Pin14)	UART(BLE MCU→Host MCU)
Vss(GND1)	Vss(CN4-Pin26)	Ground

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Apr 14, 2017	-	First edition
1.00	Jan 31, 2022	-	Fixed due to the end of IAR support in Bluetooth Low Energy Protocol Stack.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0