RL78/G1D Module

Module Control Software (Including Module Firmware)

R01AN3362EJ0111
Rev.1.11
Jan 31, 2022

Introduction

This application notebook is described about the software which controls the RL78/G1D module (RY7011).

The module software is made based on the modem configuration of Bluetooth® low energy protocol stack V1.11 (BLE Software). The Module firmware for operations checks has been written to the module. And a system consists of a program of the Host MCU which controls that. (The firmware indicates to the program that has been incorporated in the fixed to the microcomputer device)

The Host MCU program that operates in RL78/G1D or PC and the source code of the module firmware are bundled as a sample program. Host MCU program is a program to control in a command line like a sample program attached to the BLE software V1.11. When porting to other MCU, please also refer to sample program application note for Host MCU.

- RL78/G14 Host Sample (R01AN2807)
- RX113 Host Sample (R01AN3155)

The source code of the module firmware corresponding to the BLE software V1.21 and Host MCU program also bundled for reference. The correspondence of the sample program and the BLE software version is shown below.

- Renesas_BLE_Module_V101 : It is BLE Software V1.11 is used. This program is written in the module.
- Renesas_BLE_Module_V111 : It is BLE Software V1.21 is used.

The contents described in this document are shown in the following.

- The outline of the Host MCU program, the module firmware and the profile (Chapter 1)
- Development environment (Chapter 2)
- Added API to the Host MCU program and the serial communication specification with the module (Chapter 3)
- Software specification of the module firmware and the hardware setting (Chapter 4)

Target Device

RL78/G1D Module (RY7011)

Precautions:

(1) A shipping check flag is written in block 254 of a cord flash memory, so please don't rewrite.

(2) Bluetooth Device Address is written in block 255 of a cord flash memory, so please don't rewrite.

(3) Please in the setting that does not erase the unused code flash memory in an integrated development environment. (Refer to "5.1 Code Flash Memory Rewriting Setting")

Rerated Documents

Document Name	Document No.
RL78/G1D Module	
Firmware User's Manual	R01UW0160E
User's Manual : Hardware	R02UH0004E
Bluetooth Low Energy Protocol Stack	
User's Manual	R01UW0095E
Quick Start Guide	R01AN2767E
API Reference Manual : Basics	R01UW0088E
API Reference Manual : FMP (Obsolete)	R01UW0089E
API Reference Manual : PXP (Obsolete)	R01UW0090E
API Reference Manual : HTP (Obsolete)	R01UW0091E
API Reference Manual : BLP (Obsolete)	R01UW0092E
API Reference Manual : HRP (Obsolete)	R01UW0097E
API Reference Manual : GLP (Obsolete)	R01UW0103E
API Reference Manual : TIP (Obsolete)	R01UW0106E
API Reference Manual : RSCP (Obsolete)	R01UW0107E
API Reference Manual : ANP (Obsolete)	R01UW0108E
API Reference Manual : PASP (Obsolete)	R01UW0109E
Application Note : Sample Program	R01AN1375E
Application Note : rBLE Command Specification	R01AN1376E
Application Note : GUI Tool	R01AN2469E
RL78/G14 Host Sample	R01AN2807E
RX113 Host Sample	R01AN3155E

Contents

1. Overview

1.1 System Configuration

A module system is modem composition which consists of the RL78/G1D module (RY7011) and the Host MCU which controls a module. This section describes the configuration of the software in the modem configuration.

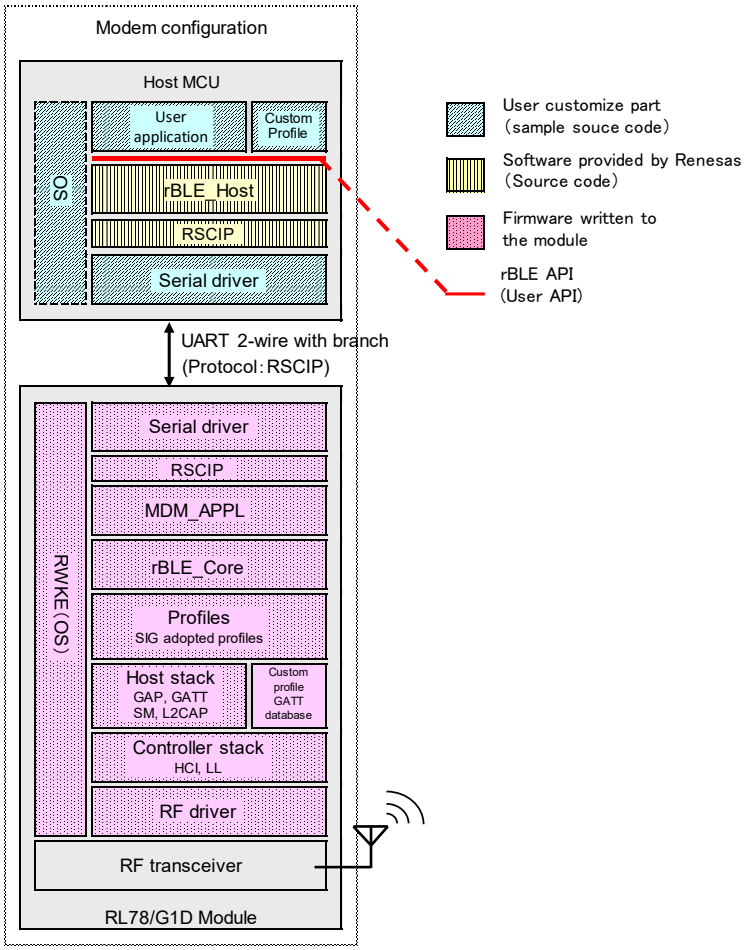

Figure 1-1 Software Structure

1.1.1 Host MCU Program

The Host MCU program is which consists of user application and BLE communication and the programming to control a serial communication with a module. The user application uses rBLE API which is the user API and controls a module. The rBLE API provides API of GAP (Generic Access Profile), GATT (Generic Attribute Profile) and each profile. And it's possible to receive event notice from a module by the call back function.

The rBLE_Host is between the user application and the serial communication part. The rBLE command processing is performed to make the BLE communication with the module. The rBLE_Host provides rBLE API and a notice event to the user application. Change to the rBLE command packet which corresponds to rBLE API and rBLE event packet from a module is changed to event notice to the user application are processed.

The program which controls communication with a module is RSCIP (Renesas Serial Communication Interface Protocol). The RSCIP does the link establishment of Host MCU and module. Change to a RSCIP packet from rBLE command packet and change RSCIP packet from the module to the rBLE event packet are processed. It also has function to recover by retransmitting the error that occurred in the serial communication.

A serial communication driver controls a module and communication of Host MCU by UART. UART 2-wire with branch connection method is expanding 2-wire UART. The TxD line is branched and connected to the WAKEUP of modules from Host MCU, it is a communication system that was utilized to wake up the module from the standby mode.

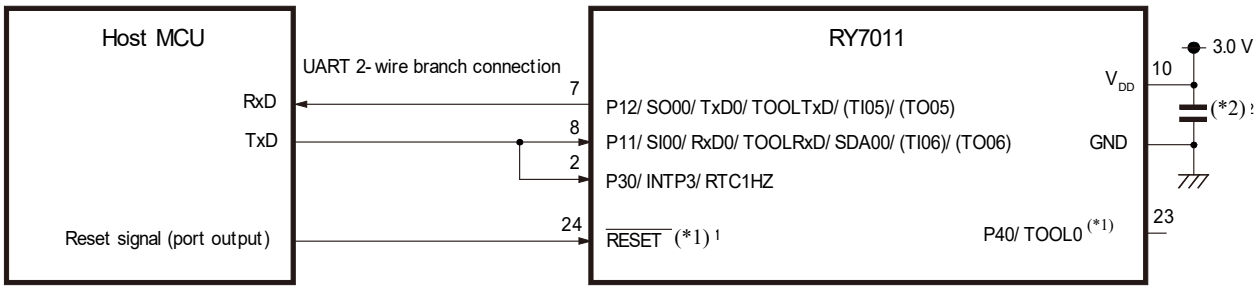

Figure 1-2 UART 2-wire with branch connection

Note *1. /RESET and P40/TOOL0 pins are pulled-up/pulled-down with a resistor in accordance with the system requirement (see RL78/G1D User's Manual: Hardware).

*2. Insert bypass capacitor of several μF between the VDD and GND pins to suit the characteristics the power supply and wiring patterns.

Table 1-1 Host MCU Software Structure

Block Name	Description
User application (*1)	User application which uses rBLE API and does BLE communication.
Custom profile (*1)	The User application part of a custom profile. (Firmware Update, General purpose communication)
OS (*1)	It's possible to include OS in the user application. (It is not used in Host MCU sample program)
rBLE_Host(*2)	Builds command packets for MDM_APPL and analyzes event packets from MDM_APPL, enabling the issuance of rBLE APIs from the application.
RSCIP(*2)	It is a communication protocol used between the Host MCU and module. It provides the recovery function of the serial communication and Link establishment between the Host MCU and module.
Serial driver (*1)	The UART driver which communicates with a module.

Note *1. It is the user customization part. Sample code is provided in the module software.

*2. A file with description of rBLE_Host and RSCIP is the source cord we provide in "5.2.1(3) Module Firmware Folder".

1.1.2 Module Firmware

It has a serial communication drivers and RSCIP to control the serial communication as well as the Host MCU. Serial communication driver of the module, do the WAKEUP process from standby mode by the communication from the Host MCU in addition to the serial communication.

The MDM_APPL analyzes command packets from rBLE_Host and builds event packets for rBLE_Host, enabling the use of BLE stack services via rBLE_Core.

The rBLE_Core provides an interface with the upstream modules for using the services of the main BLE stack.

A profile layer, a Host stack and a Controller stack are the body of the BLE stack which controls BLE communication. It includes profiles that are adopted by the Bluetooth SIG, GAP, GATT, SM(Security Manager)、L2CAP(Logical Link Control and Adaptation Protocol) and LL(Link Layer).

The RF driver controls the RF transceiver of the RL78/G1D.

The RWKE provides the basic functionality used commonly with the other modules and manages the entire BLE MCU. Scheduling and the memory resource of the whole system are managed.

(Later, the module firmware is called the firmware)

Table 1-2 Module Software Structure

Block Name	Description
Serial driver	It is a UART 2-wire with branch connection driver. It consists UART driver to communicate with the Host MCU and WAKEUP driver for receiving a signal to wake up the module from the power saving mode.
RSCIP	Communication protocol between Host MCU and module. It provides the recovery function of the serial communication and Link establishment between the Host MCU and module.
MDM APPL	Analyzes command packets from rBLE_Host and builds event packets for rBLE_Host, enabling the use of BLE stack services via rBLE_Core.
rBLE_Core	Provides an interface with the upstream modules for using the services of the main BLE stack (the part from the profile layer to the RF driver).
Custom profile GATT database	GATT database of custom profile. (Firmware Update, General purpose communication)
Profile layer	BLE stack Profiles : FMP, PXP, HTP, BLP, HRP, GLP, TIP, RSCP, ANP, PASP Host stack : SM, L2CAP, GAP, GATT Controller stack : LL, HCI
Host stack	
Controller stack	
RWKE (Renesas Wireless Kernel Extension)	Provides the basic functionality used commonly with the other modules and manages the entire BLE MCU.
RF driver	It controls the RF transceiver.

1.1.3 Custom Profile

The custom profile of Modem configuration is a structure that has been divided into the module side and the Host MCU side. The module side there is a GATT database section of the custom profile. The Host MCU side there is the custom profile as the user application.

1.2 Profile

In this section, it shows the profile specifications integrated in the firmware.

Due to the deprecation and withdrawal plan of the profile version by Bluetooth SIG, each profile has been obsoleted because product registration using the profile supported by the Bluetooth Low Energy protocol stack is no longer possible.

For product registration, refer to "Bluetooth LE microcomputer/module Bluetooth qualification acquisition application note" (R01AN3177).

1.2.1 Profile List

A list of adopted profiles is indicated below.

Table 1-3 Adopted profiles

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

Table 1-4 Adopted profiles

Profile	Role
General Purpose Communication	Client
	Server
Firmware Update	Sender
	Receiver

1.2.2 Services

The module has the service and UUID is shown below.

Table 1-5 Adopted services

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

Table 1-6 Custom profiles

Service	UUID (HEX)
General Purpose Communication	D68C0001-A21B-11E5-8CB8-0002A5D5C51B
Firmware Update	01010000-0000-0000-0000-000000000080

1.2.3 GATT Database

The GATT data base integrated in a module is shown below.

Table 1-7 GATT database

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

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

0x006D	Characteristic Declaration	Properties = 0x10(NTF)
0x006E	0x2A45 (Unread Alert Status)	
0x006F	0x2902 (Client Characteristic Configuration)	
0x0070	Characteristic Declaration	Properties = 0x08(WR)
0x0071	0x2A44 (Alert Notification Control Point)	
0x0072	Primary Service Declaration	0x180E(Phone Alert Status Service)
0x0073	Characteristic Declaration	Properties = 0x12(RD, NTF)
0x0074	0x2A3F (Alert Status)	
0x0075	0x2902 (Client Characteristic Configuration)	
0x0076	Characteristic Declaration	Properties = 0x12(RD, NTF)
0x0077	0x2A41 (Ringer Setting)	
0x0078	0x2902 (Client Characteristic Configuration)	
0x0079	Characteristic Declaration	Properties = 0x04(WR_NO_RESP)
0x007A	0x2A40 (Ringer Control Point)	
0x007B	Primary Service Declaration	0x1814(Running Speed and Cadence Service)
0x007C	Characteristic Declaration	Properties = 0x10(NTF)
0x007D	0x2A53 (RSC Measurement)	
0x007E	0x2902 (Client Characteristic Configuration)	
0x007F	Characteristic Declaration	Properties = 0x02(RD)
0x0080	0x2A54 (RSC Feature)	
0x0081	Characteristic Declaration	Properties = 0x02(RD)
0x0082	0x2A5D (Sensor Location)	
0x0083	Characteristic Declaration	Properties = 0x28(WR, IND)
0x0084	0x2A55 (SC Control Point)	
0x0085	0x2902 (Client Characteristic Configuration)	
0x0086	General purpose communication Refer to "Table 1-9 General purpose communication database"	
0x0087		
0x0088		
0x0089		
0x008A		
0x008B		

(1) **Adopted profiles database specification**

About adopted profile database specification, refer to following of "Bluetooth Low Energy protocol stack user's manual" (R01UW0095). And about profile specification of Bluetooth SIG, refer to profile specification of Bluetooth SIG. ("5.4 Referenced Documents")

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

(2) **Custom profiles database specification**

The GATT database specification of the custom profile integrated in a module is shown below.

Table 1-8 Firmware update database

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

Table 1-9 General purpose communication database

Attribute Handle	Attribute Type	Attribute Value
0x0086	Primary Service Declaration (0x2800)	UUID: 0xD68C0001-A21B-11E5-8CB8-0002A5D5C51B
0x0087	Characteristic Declaration (0x2803)	Property: Indicate(0x20) Type: Characteristic Declaration UUID: 0xD68C0002-A21B-11E5-8CB8-0002A5D5C51B
0x0088	Indication Value	By setting characters to this characteristic and send Indication, the characters are sent from the server to the client. Max 20 characters.
0x0089	Client Characteristic Configuration Descriptor (0x2902)	Used for Indication enable / disable of the server from the client. 0x0000: Indications disabled 0x0002: Indications enabled
0x008A	Characteristic Declaration (0x2803)	Property: Write(0x08) Type: Characteristic Declaration UUID: 0xD68C0003-A21B-11E5-8CB8-0002A5D5C51B
0x008B	Write Value	By writing characters to this characteristic with “Write Request”, the characters are sent from the client to the server. Max 20 characters.

1.3 Reference Documents

A related document is indicated below. Also refer to these documents.

- rBLE API
  - API Reference Manual: Basics (R01UW0088)
  - API Reference Manual for each profile

- rBLE Command
  - rBLE Command Specification (R01AN1376)

- Firmware Update
  - Bluetooth Low Energy Protocol Stack User's Manual (R01UW0095)
    "11. Implementation of FW Update Feature"

- UART 2-wire with branch connection
  - Host Sample Application Note (R01AN2807)
  - RX113 Host Sample Application (R01AN3155)

2. Development Environment

Development environment of the module control software is shown below.

2.1 Hardware Environment

- Host
  - PC/ATTM-compatible computer
  - Processor: At least 1.6 GHz
  - Main memory: At least 1 GB
  - Display: 1024 x 768 or higher resolution and 65,536 colors
  - Interface: USB 2.0 (E1 and USB-serial conversion cable)

- Tool
  - Renesas on-chip debugging emulator E1

2.2 Software Environment

- Windows 7 or later
- Microsoft Visual Studio Express 2015 for Windows Desktop
- Microsoft .NET Framework 4＋Language Pack
- e2 studio V5.4.0.015/RL78 compiler CC-RL V1.02.00 and V1.03.00
  CC-RL V1.02.00: It is used by build of module firmware V1.01
  (The firmware written to the module.)
  CC-RL V1.03.00: It is used by build of module firmware V1.11
- Renesas Flash Programmer V3

2.3 Build Preparation

Please copy the software that is bundled with this application note to a path that does not contain blank and multi-byte characters.

In order to build the firmware requires the following library. After having obtained those libraries, copy to a specify folder of "5.2.1(3) Module Firmware Folder".

- Renesas Bluetooth Low Energy protocol stack V1.11 CC-RL Library
- EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family
- Flash Self Programming Library Type01 Package Ver.3.00 for the RL78 Family [for the CA78K0R/CC-RL Compiler]

2.3.1 CC-RL Library of BLE Protocol Stack

CC-RL Library stored folder and files:

\BLE_Software_Ver_1_11\RL78_G1D\Project_Source\renesas\lib

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

Destination folder:

\Renesas_BLE_Module_V101\RL78_G1D\Project_Source\renesas\lib

2.3.2 Installing EEPROM Emulation Library

CC-RL Library stored folder and files:

Files:

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

Destination folder:

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

2.3.3 Installing Flash Self Programming Library

CC-RL Library stored folder and files:

Files:

fsl.h

fsl.lib

fsl_types.h

Destination folder:

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

2.3.4 Folder List

Refer to "5.2 Folder List".

3. Host MCU Program

3.1 Processing Flow

The program of Host MCU consists of the application, the custom profile, the rBLE_Host, the RSCIP, the serial communication driver and the OS block. The processing flow figure between each block is shown below.

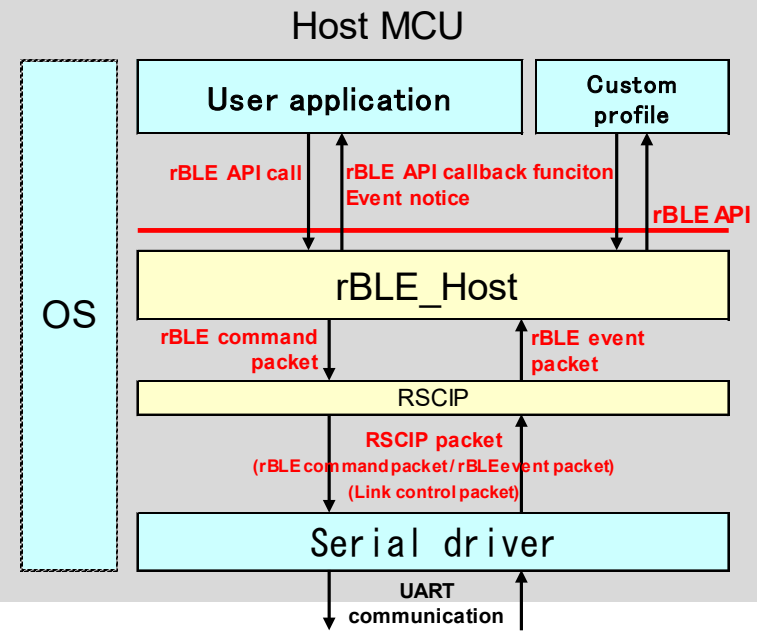

Figure 3-1 Host MCU Processing Flow

A program of Host MCU controls a module by rBLE API the user application calls. Module returns to Host MCU execution result as an event. User application receives as an event notification in rBLE API callback function (The custom profile integrated as the user application also does the same processing).

The block where these rBLE API call and rBLE API call back function event notice are processed is rBLE_Host. The rBLE API called by application is changed to the rBLE command packet. Event notice of the rBLE API call back function is performed by the rBLE event packet from a module.

RSCIP controls communication with a module and does the processing connection establishment of a module (link) does with Host MCU. It performs a process of changing from rBLE command packet to RSCIP packet and changing RSCIP packet form module to rBLE event packet. In addition, it performs a process of conversion. It also has the ability to recover by the retransmission of the error that occurred in the serial communication.

3.2 Communication with the Module

Communication between the Host MCU and the module is performed in UART 2-wire with branch connection. This section describes the procedure of communication with the module.

After establishing a link (connection), communication of Host MCU and a module is performed by a RSCIP packet (include link control packet, rBLE command packet and rBLE event packet).

Link control packet	: Used for link establishment between the Host MCU and the module.
rBLE command packet	: Used for operation instruction to the module from the Host MCU.
rBLE event packet	: Used for notification information to the Host MCU from the module.

About the RSCIP packet, refer to "BLE Command Specification" (R01AN1376).

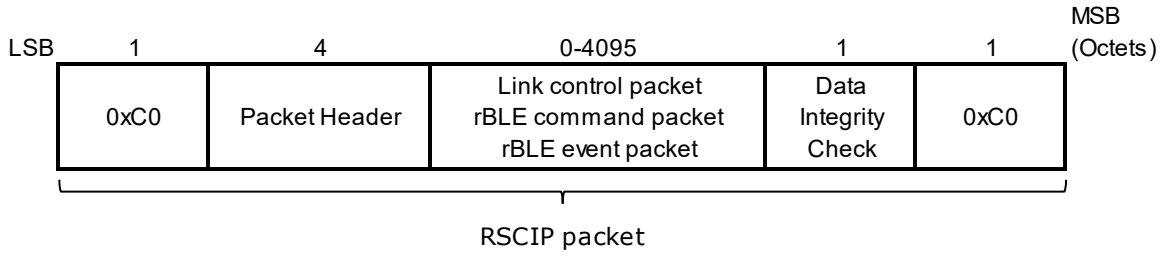

Figure 3-2 RSCIP packet

3.2.1 Link Establishment

Host MCU and modules, in order to communicate using the rBLE command packet and rBLE event packet, it must first establish a link. Link control packet is used by link establishment.

A module and a MCU exchange "SYNC - SYNC RESPONSE" and "CONFIG - CONFIG RESPONSE" of a control packet for link establishment. When sending a RSCIP packet to a module from Host MCU by UART 2 line branch connection method, a handshake is performed. A handshake is the process to confirm that a module has completed reception preparations. Sends the REQ byte (0xC0) from the Host MCU, and receive the ACK byte (0x88) or the RSCIP packet from the module.

About the state transition of the handshake and the UART driver, refer to "3.2.2 Transmission Process". The following shows the RSCIP packet exchange sequence of up to link establishment.

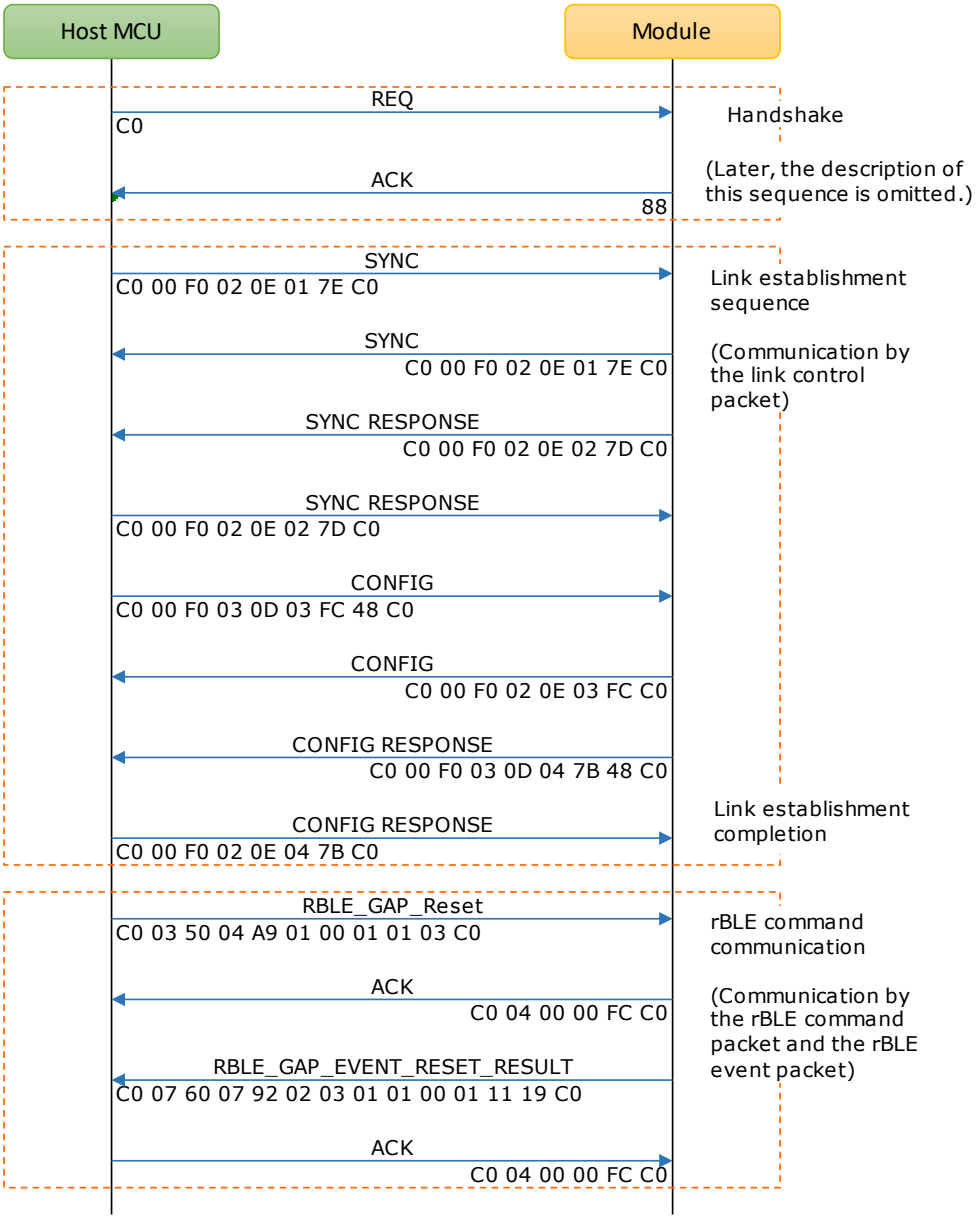

Figure 3-3 Link establishment packet exchange sequence

3.2.2 Transmission Process

A handshake is performed to send the packet to a module from Host MCU. A handshake is performed by send the REQ byte (0xC0) from the Host MCU and send the ACK byte (0x88) or the RSCIP packet from the module. In addition, when performing a handshake performs monitoring by the timer, the timeout occurs and restart the handshake. Host MCU of UART driver for performing a handshake, it has a 5 state by the transmission status.

Table 3-1 UART driver transmission state

STATE	Description
T_IDLE	Initialize UART driver. RSCIP packet transmission completion.
T_REQUESTING	During REQ byte transmission.
T_RCV_BF_REQUESTED	Receive RSCIP packet from the module instead of ACK bytes.
T_REQUESTED	REQ byte transmission completion. (Wait for the ACK byte from the module)
T_ACTIVE	During RSCIP packet transmission.

Transmission from the Host MCU to the module, always start with REQ byte. After sending the REQ byte, Host MCU branches to one of the following operations by the receiving state.

(a) Host MCU has not received RSCIP packet from the module (Figure 3-4)

(b) Host MCU is receiving RSCIP packet from the module (Figure 3-5)

(c) REQ byte transmission time-out (Figure 3-6)

(a) Host the MCU has not received RSCIP packet from the module

This state is RSCIP packet has not been transmitted from the module, after sending the REQ byte from Host the MCU, the Host MCU is waiting to receive an ACK byte. Module sends an ACK byte receive the REQ byte. Host MCU which received ACK byte sends a RSCIP packet to a module.

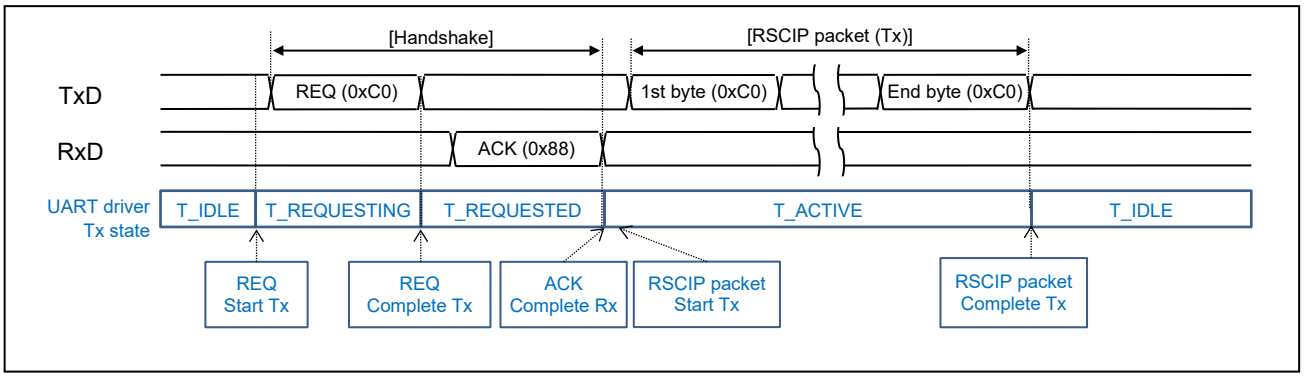

Figure 3-4 Host MCU has not received RSCIP packet from the module

(b) **Host MCU is receiving RSCIP packet from the module**

This state module has to send RSCIP packet, Host MCU is receiving RSCIP packet. Even if a module receives REQ, ACK byte isn't returned. The RSCIP packet which is being sent is made a substitute of ACK byte. A host regards a RSCIP packet from a module as a substitute of ACK byte. And a RSCIP packet is sent to a module.

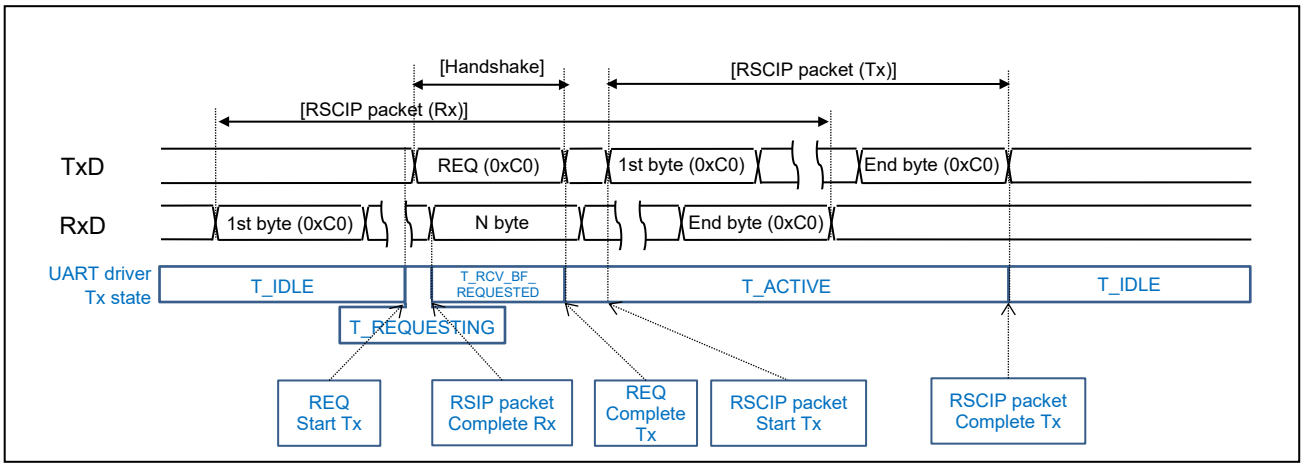

Figure 3-5 Host MCU is receiving RSCIP packet from the module

(c) **REQ byte transmission time-out**

After sending REQ byte, Host MCU starts a timeout timer. If it can not be received ACK bytes for a certain period, and then resend the REQ byte.

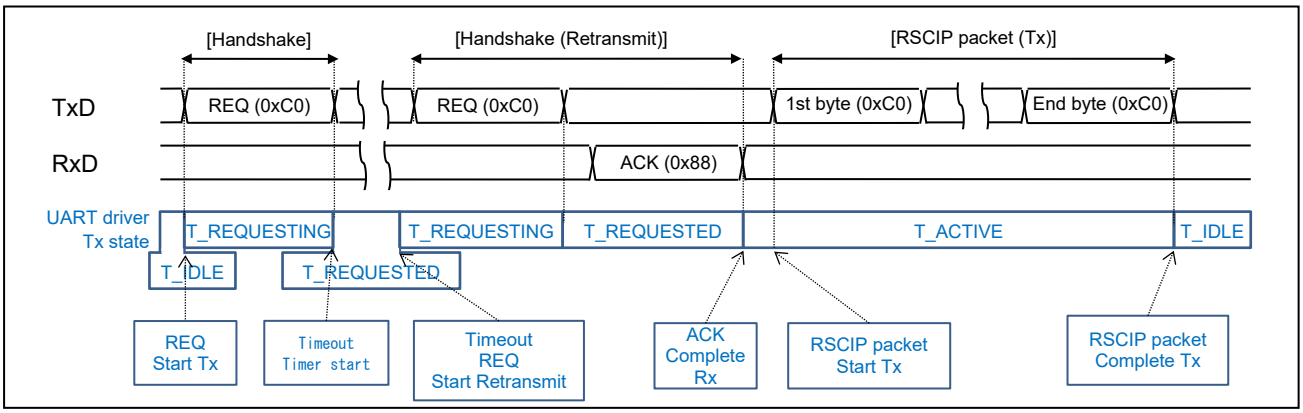

Figure 3-6 REQ byte transmission time-out

3.2.3 Reception Process

There is no state transition of a UART driver at the reception. In order to receive the data from the module, it listens for RSCIP packet from the module in the specified number of bytes from rBLE_Host.

3.3 rBLE API Call and Event Notification

BLE communication is controlled by the calling of rBLE API and notification by the execution results and communication (Event notification). The event notification is received by the callback unction. The call back function prepare by each role (rBLE initialization, GAP, SM, GATT, VS, GATT base profiles). It is possible to receive a event notification by registering the callback function with rBLE_Host. It shows an example of rBLE API calls and rBLE API event notification.

The GAP callback function is registered by the RBLE_GAP_Reset function of rBLE API. Specify callback function of GAP and callback function of SM to parameter of RBLE_GAP_Reset. Thereby, it is registered in rBLE_Host, And make possible to receiving event notification. The BLE_GAP_EVENT_RESET_RESULT event which is an execution result of the RBLE_GAP_Reset function is notified to GAP callback function.

When starting the broadcast, call the RBLE_GAP_Broadcast_Enable function. The RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP event which is an execution result of the RBLE_GAP_Broadcast_Enable function is notified to GAP callback function.

About the rBLE API and the event notification, refer to "API Reference Manual: Basics"(R01UW0088) and each API reference manuals.

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

```
      /* GAP Callback function */
1762:static void RBLE_APP_GAP_CallBack( RBLE_GAP_EVENT *event )
1763:{
1764:  :
1769:    switch( event->type ) {
1770:        case RBLE_GAP_EVENT_RESET_RESULT:
       :
1780:        case RBLE_GAP_EVENT_SET_NAME_COMP:
       :
1783:        case RBLE_GAP_EVENT_OBSERVATION_ENABLE_COMP:
       :
1786:        case RBLE_GAP_EVENT_OBSERVATION_DISABLE_COMP:
       :
1789:        case RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP:
       :



      /* GAP Reset process(Callback function registration) */
2136:static BOOL RBLE_GAP_Reset_Test( void )
2137:{
2138:
2139:    RBLE_STATUS Ret_Status;
2140:
2141:    /* API Call */
2142:    Ret_Status = RBLE_GAP_Reset( &RBLE_APP_GAP_CallBack, &RBLE_APP_SM_CallBack );
       :
2148:}



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

3.4 Hide Services

When service to which firmware corresponds isn't used, as the opposite device does not perform unnecessary processing, do not use the service should be set to private.

Set the service to the private setting from the Host MCU to the module startup. Use the RBLE_GATT_Set_Permission function of rBLE API, set the RBLE_GATT_PERM_HIDE (0x2000) to the permission of the local GATT database.

A setting method set the permission value from start_hdl to end_hdl of parameter. Please note that the original permission value is set to a new value. When the setting is complete, the RBLE_GATT_EVENT_SET_PERM_CMP event of rBLE API event will be notified.

*Do not change permission value of attribute handle from 0x0001 to 0x000B. (Refer to "1.2.3 GATT Database")

The API and the Event is used by permission setting is shown below.

- rBLE API

RBLE_STATUS RBLE_GATT_Set_Permission(RBLE_GATT_SET_PERM *set_perm)			
This function is used to set the permission to the specified range of local GATT database by handles. The result is notified by the set permission completion event RBLE_GATT_EVENT_SET_PERM_CMP.			
Parameters:			
	set_perm	*start_hdl*	Start handle to set permission
		end_hdl	End handle to set permission
		perm	Permission (Refer to “API Reference Manual: Basics - 7.1 Declaration of enumerated type for GATT attribute permission” n)
Return:			
	RBLE_OK (0x00)		Success
	RBLE_STATUS_ERROR (0xF2)		Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

- rBLE API Event

RBLE_GATT_EVENT_SET_PERM_CMP			
This event notifies the results of command execution (set permission of local GATT database).			
Parameters:			
	set_perm_cmp	*status*	Result of command execution (Refer to “API Reference Manual: Basics - 3.2 Declaration of enumerated type for rBLE status”)

If publish the private GATT service, set permission value for each attribute handle using the RBLE_GATT_Set_Permission function. Shows permission initial value to "Table 3-2 Permission initial value", and shows permission definition value "Table 3-3 Permission definition value".

Table 3-2 Permission initial value

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

Table 3-3 Permission definition value

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

3.5 rBLE Command API

The firmware is added the RBLE_VS_Set_Params function of Vendor Specific function and the own rBLE command of RBLE_VS_Flash_Access function as own command.

RBLE_VS_Set_Params function can assign a command to more than 0x80 of the first argument (param_id). The firmware is added following three commands.

- Baud rate setting
- Firmware Update
- Software Reset

RBLE_VS_Flash_Access function is added following command by add to id of parameter structure member.

- Read Firmware Version

rBLE API specification is shown below.

3.5.1 RBLE_VS_Set_Params

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

This function sets the parameters in BLE MCU. The result is reported by using the parameter setting completion

- The Baud rate setting saves baud rate id to data flash. Read the baud rate id when the module is started, and initialize the serial driver. After executing the Baud rate setting, execute the reset. A baud rete will valid after reset.
- After executing the Firmware Update, enter to the Firmware Update mode after one second.
- The Software Reset execute internal reset by executing illegal instruction. After executing the Software Reset, an internal reset occurs after one second.

Parameters:

param_id

Parameter ID

Parameter ID	No.	Brief
RBLE_VS_PARAM_UART_BAUD_ID	0x80	Baud rate setting
RBLE_VS_PARAM_FW_UPDATE	0xD9	Move to Firmware Update mode
RBLE_VS_PARAM_SOFT_RESET	0xFF	Software Reset

param_len

Parameter length

Parameter ID	Parameter length
RBLE_VS_PARAM_UART_BAUD_ID	1
RBLE_VS_PARAM_FW_UPDATE	Unused
RBLE_VS_PARAM_SOFT_RESET	Unused

**param_data*

Pointer to the parameter data(the least significant byte first, left justified)

Parameter ID	Baud rate ID
RBLE_VS_PARAM_UART_BAUD_ID	0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps 6: 250000 bps
RBLE_VS_PARAM_FW_UPDATE	Unused
RBLE_VS_PARAM_SOFT_RESET	Unused

Return:

RBLE_OK (0x00)	Success
RBLE_STATUS_ERROR (0xF2)	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
RBLE_UNSUPPORTED (0x11)	Unsupported param_id
RBLE_PARAM_ERR (0xF3)	Parameter error

RBLE_VS_EVENT_SET_PARAMS_COMP event reports completion of executing the set parameter command. Return execution result is stored to the "status".

RBLE_VS_EVENT_SET_PARAMS_COMP			
This event reports completion of setting up a parameter.			
Parameters:			
	status	*RBLE_OK (0x00)*	Success
		RBLE_STATUS_ERROR (0xF2)	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.
		RBLE_UNSUPPORTED (0x11)	Unsupported param_id
		RBLE_PARAM_ERR (0xF3)	Parameter error

3.5.2 RBLE_VS_Flash_Access

RBLE_STATUS RBLE_VS_Flash_Access (RBLE_VS_FLASH_ACCESS_PARAM *param)

This function writes the data to Data Flash or reads data from Data Flash.
The result is reported by using the Data Flash access command completion event RBLE_VS_EVENT_FLASH_ACCESS_COMP.
* Before calling this function, starts the access to Data Flash by using the RBLE_VS_Flash_Management. In addition, maintain buffer that is specified in the parameter until the data writing or reading is completed.

Parameters:

cmd	Data Flash Access Command RBLE_VS_FLASH_CMD_WRITE : Write data RBLE_VS_FLASH_CMD_READ : Read data
id	Data ID (0x01 – 0xFF)

Setting Parameter ID	No.	Brief
RBLE_VS_PARAM_EEL_ID_MODFWVER	4	Firmware version

size	Data size (1 - 255 bytes)

Setting Parameter ID	Data Size
RBLE_VS_PARAM_EEL_ID_MODFWVER	2

**addr*	Pointer to writing or reading buffer.

Return:

RBLE_OK (0x00)	Success
RBLE_STATUS_ERROR (0xF2)	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

RBLE_VS_EVENT_FLASH_ACCESS_COMP event reports completion of executing the Data Flash access command. Return a pointer of the buffer that the firmware version is stored

RBLE_VS_EVENT_FLASH_ACCESS_COMP	
This event reports completion of executing the Data Flash access command.	
Parameters:	
status	Data Flash command execution result.
cmd	Command
id	Data ID
size	Data size
**addr*	Pointer to data buffer. The firmware version is stored. Format is following. [0] : Minor version [1] : Major version e.g.) V1.00 : [1]=0x01, [0]=0x00

3.6 Host MCU Peripheral

The peripheral function required in the program of Host MCU is shown below.

Table 3-4 Host MCU program peripheral

Function	Purpose / Usage
UART	Used to serial communication. −Baud rate : 4800、9600、19200、38400、57600、115200、250000 (default :115200 bps) −Data length : 8 bit −Parity : none −Stop bit : 1 bit −Flow control : none −Data direction : LSB first
Timer	Monitoring of timeout for RSCIP.

3.7 How to Build Sample Program

The BLE_Sample folder as the sample program of Host MCU stores the program that runs on a PC and RL78/G1D. Stored program folder is below. Or refer to "5.2 Folder List".

PC : \Renesas_BLE_Module_V101\BLE_Sample\project\windows

RL78/G1D : \Renesas_BLE_Module_V101\BLE_Sample\project\e2studio\BLE_Sample\rBLE_sample

3.7.1 rBLE_sample Program

1. Launch the Microsoft Visual Studio Express 2015 for Windows Desktop.
2. Select [File] - [Open project].
3. Select rBLE_Sample.vcxproj of following folder.
   - \Renesas_BLE_Module_V101\BLE_Sample\project\windows
4. Click [Open].
5. Right click on [Solution Explorer] - [rBLE_sample]. And select [Build].
6. The Exe file "rBLE_Sample.exe" is generated in the following path.
   - \Renesas_BLE_Module_V101\BLE_Sample\project\windows\Debug

3.7.2 RL78/G1D Sample Program

1. Launch the Renesas e^2 studio.
2. Select [File] - [Import].
3. Select [General] - [Existing Projects into Workspace] and click [Next>].
4. Set the following path to [Select root directory] and make sure that rBLE_sample is shown in [Project].
   - \Renesas_BLE_Module_V101\BLE_Sample\project\e2studio\BLE_Sample\rBLE_sample
5. Click [Finish].
6. Right click on [Project Explorer] - [rBLE_sample]. And select [Build Project].
7. The firmware update file "rBLE_sample_CCRL.hex" is generated in the following path.
   - \Renesas_BLE_Module_V101\BLE_Sample\project\e2studio\BLE_Sample\rBLE_sample\DefaultBuild

4. Module Firmware

Firmware that is written to the module has been created based on the Modem configuration of BLE Software V1.11. This section describes about the firmware.

- Firmware specification
- Module peripheral
- Unused pin setting
- Memory map
- Changes of source code from the BLE software

Precautions:

(1) A shipping check flag is written in block 254 of a cord flash memory, so please don't rewrite.

(2) Bluetooth Device Address is written in block 255 of a cord flash memory, so please don't rewrite.

(3) Please in the setting that does not erase the unused code flash memory in an integrated development environment. (Refer to "5.1 Code Flash Memory Rewriting Setting")

4.1 Firmware Specification

Changes from the BLE software V1.11, and setting of the firmware is shown below.

- Changes form the BLE software V1.11
  - Select the UART 2-wire with branch communication to communicate with the Host MCU
  - Add the general purpose communication of custom profile
  - Add the following vendor specific commands to rBLE command
    - Read firmware version
    - Baud rate setting
    - Software Reset
    - Execute the firmware update

- Firmware Settings
  - BLE Software Configuration : Modem Configuration
  - Number of Simultaneous Connections : 6
  - Operating Frequency : 8 MHz
  - On-chip low-speed oscillator in the RF : use
  - DC-DC Converter : use
  - UART Communication Method : 2-wire with branch connection
  - Baud Rate : 4800, 9600, 19200, 38400, 57600, 115200, 250000 (default : 115200 bps)

- UART Settings
  - Data Length : 8 bits
  - Parity : none
  - Stop Bit : 1 bit
  - Flow Control : none
  - Data Direction : LSB first

4.2 Module Peripheral

The peripheral of the RL78/G1D module is used by the firmware is shown below.

Table 4-1 Module Peripheral

Function		Function Name	Purpose / Usage
Data flash		Used	Store the BD address and the baud rate id.
12-bit interval timer		Used	Monitoring of on-chip oscillator for the RF slow clock.
Timer array unit		Timer00	Wait for rBLE command.
Port function		P30/INTP3 (input)	Wakeup pin for UART 2-wire with branch connection. (INTP3)
Serial array unit		UART0	UART 2-wire with branch connection.
Multiplier and divider / Multiply-accumulator		Used	Used by the C Compiler. Set the compiler option to use this hardware.
DMA controller		DMA0, DMA1 DMA2, DMA3	DMA0、DMA1：UART 2-wire with branch connection. DMA2、DMA3：Communication interface between MCU and RF Transceiver.
Interrupt	External pin	INTRF INTP3	INTRF：Interruption from RF transceiver. INTP3：Wakeup signal.
	DMA	INTDMA0 INTDMA1 INTDMA2 INTDMA3	INTDMA0、INTDMA1：UART 2-wire with branch connection. INTDAM2、INTDMA3：Communication interface between MCU and RF Transceiver.
	Serial array unit	INTST0 INTSR0 INTSRE0	UART 2-wire with branch connection.
	12-bit interval timer	INTIT	Monitoring of on-chip oscillator for the RF slow clock.
	Timer array unit	INTTM00	Wait for rBLE command.

4.3 Unused Pin Setting

Unused pin setting of the firmware is shown below.

Table 4-2 Unused Pin Setting

Pin No.	Pin Name	Software Setting		
		Function	I/O	On-chip Pull-up
1	GND	-	-	-
2	P30/INTP3/RTC1HZ	P30	input	not connected
3	P16/TI01/TO01/INTP5	P16	input	connected
4	P15/SCK20/SCL20(TI02)/(TO02)	P15	input	connected
5	P14/SI20/SDA20/(SCLA0)/(TI03)/(TO03)	P14	input	connected
6	P13/SO20/(SDAA0)/(TI04)/(TO04)	P13	input	connected
7	P12/SO00/TxD0/TOOLTxD/(TI05)/(TO05)	TxD0	output	-
8	P11/SI00/RxD0/TOOLRxD/SDA00/(TI06)/(TO06)	RxD0	input	not connected
9	P10/SCK00/SCL00/(TI07)/(TO07)	P10	input	connected
10	VDD	-	-	-
11	GND		-	-
12	P147/ANI18	P147	input	connected
13	P23/ANI3	P23	output	-
14	P22/ANI2	P22	output	-
15	P21/ANI1/AVREFM	P21	output	-
16	P20/ANI0/AVREFP	P20	output	-
17	P03/ANI16/RxD1	P03	output	-
18	P02/ANI17/TxD1	P02	input	connected
19	P01/TO00	P01	input	connected
20	GND	-	-	-
21	P00/TI00	P00	input	connected
22	P120/ANI19	P120	output	-
23	P40/TOOL0	P40	input	not connected
24	/RESET	/RESET	input	not connected
25	P137/INTP0	P137	input	not connected
26	P124/XT2/EXCLKS	P124	input	not connected
27	P123/XT1	P123	input	not connected
28	P60/(SCLA0)	P60	input	not connected
29	P61/(SDAA0)	P61	input	not connected
30	GND	-	-	-
31	IC	-	-	-
32～42	GND	-	-	-

4.4 Memory map

The memory map of the firmware is shown below. The user code section is .

Table 4-3 Memory map (Code flash)

Memory	Block Number	Section Name	Start Address	End Address	Size (byte)	Brief
Code Flash	000-003	.vect	0x00000	0x0007F	128	Vector table
		.callt0	0x00080	0x0009D	30	CALLT table
		.option_byt	0x000C0	0x000C3	4	Option byte
		.security_i	0x000C4	0x000CD	10	Security ID
		.monitor1	0x000CE	0x000D7	10	OCD monitor
		ROM_BOOT0	0x000D8	0x00125	78	Boot code
		.RLIB	0x00126	0x00910	2027	Runtime library
		MAIN_CN0_f	0x00FFE	0x00FFF	2	FW update count management
	004-007	MAIN_CN1_f	0x01FFE	0x01FFF	2	FW update count management
	008	.SLIB	0x02000	0x02011	18	Standard library
	012-016	HDB_CNST_n	0x03000	0x0312B	300	Host database
		HST_CNST_n	0x0312C	0x03B3D	2578	Host stack CONST data
		CNT_CNST_n	0x03B3E	0x0409B	1374	Controller stack CONST data
		RBL_CNST_n	0x0409C	0x042ED	594	RBLE CONST data
		MAINCNST_n	0x042EE	0x04343	86	main CONST data
		DFL_CNST_f	0x04344	0x04361	30	DFL CONST data
		MOD_CNST_n	0x04362	0x0437D	28	Module CONST data
	017-021	.const	0x04400	0x05599	4506	User CONST data
	024-038	RBL_CODE_f	0x06000	0x098F1	14578	RBLE code
		PLF_CODE_f	0x098F2	0x09994	163	Platform code
		UARTCODE_f	0x09995	0x09B1B	391	UART code
	039-040	.monitor2	0x09C00	0x09DFF	512	OCD monitor
		.constf	0x09E00	0x09E00	0	User CONST data
		.sdata	0x09E00	0x09E00	0	User data with initial data
		.data	0x09E00	0x09EC5	198	User data with no initial data
		.text	0x09EC6	0x09EFE	57	User code
		UARTCODE_n	0x09EFF	0x0A065	359	UART code
	042	ACS_TBL_n	0x0A800	0x0A827	40	Access table 0
		CLK_TBL_n	0x0A850	0x0A85B	12	Clock table 0
		TSK_DESC_n	0x0A880	0x0AA7B	508	Task descriptor 0
	043	ACS_TBL_n	0x0AC00	0x0A827	0	Access table 1
		CLK_TBL_n	0x0AC50	0x0A85B	0	Clock table 1
		TSK_DESC_n	0x0AC80	0x0AA7B	0	Task descriptor 1
	044-140	CNT_CODE_n	0x0B000	0x0B3AB	940	Controller stack code
		HST_CODE_n	0x0B3AC	0x0B42C	129	Host stack code
		RBL_CODE_n	0x0B42D	0x0B45C	48	RBLE code
		MOD_CODE_n	0x0B45D	0x0B48C	48	Module code
		FDL_CNST_f	0x0B48E	0x0B497	10	FDL CONST data
		EEL_CNST_f	0x0B498	0x0B49E	7	EEL CONST data
		FDL_CODE	0x0B4A0	0x0B6C3	548	FDL code
		EEL_CODE	0x0B6C4	0x0C138	2677	EEL code
		FSL_FCD	0x0C13A	0x0C1F0	183	FSL initialization code
		FSL_BCD	0x0C1F2	0x0C2F1	256	FSL code
		FSL_BECD	0x0C2F2	0x0C371	128	FSL code
		FSL_RCD	0x0C372	0x0C3D1	96	FSL code
		DFL_CODE_f	0x0C3D2	0x0C747	886	Data Flash code
		CFL_CODE_f	0x0C748	0x0C923	476	Code Flash code
		MAINCODE_f	0x0C924	0x0CFB1	1678	main code
		HST_CODE_f	0x0CFB2	0x17C04	44115	Host stack code
		CNT_CODE_f	0x17C05	0x22F80	45948	Controller stack code
		MOD_CODE_f	0x22F81	0x230AF	303	Module code
	142-184	.textf	0x23800	0x2E0BB	43196	User code/Profile code

Table 4-4 Memory map (RAM)

MEMORY	SECTION NAME	START ADDRESS	END ADDRESS	SIZE (byte)	Brief
RAM	CNT_DATA_n	0xFB310	0xFB93D	1582	Controller stack data
	MAINDATA_n	0xFB93E	0xFB9D9	156	main data
	HDB_DATA_n	0xFB9DA	0xFBA59	128	Host database data
	HST_DATA_n	0xFBA5A	0xFBCAF	598	Host stack data
	RBL_DATA_n	0xFBCB0	0xFC1F7	1352	RBLE data
	CFL_DATA_n	0xFC1F8	0xFC1FF	8	Code Flash data
	DFL_DATA_n	0xFC200	0xFC207	8	Data Flash data
	UARTDATA_n	0xFC208	0xFC226	31	UART data
	MOD_DATA_n	0xFC228	0xFC22A	3	Module data
	.bss	0xFC230	0xFF9F3	14276	User data with no initial data
	.dataR	0xFF9F4	0xFFAB9	198	User data with initial data
	FDL_SDAT	0xFFE30	0xFFE31	2	FDL data
	EEL_SDAT	0xFFE32	0xFFE32	1	EEL data
	.sbss	0xFFE34	0xFFE34	0	User data with no initial data
	.sdataR	0xFFE34	0xFFE34	0	User data with initial data

4.5 Changes of Source code

It shows the source file changes to the firmware from the BLE software.

4.5.1 Profile Selection

The firmware change the profile selection of adopted by the Bluetooth SIG. If profile of firmware is changed, change the defined value. "0" is disable. "1" is enable.

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

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

4.5.2 UART 2-wire with Branch Connection

The firmware choose the UART 2-wire with branch communication for communication with Host MCU. The UART 2-wire with branch communication enable WAKEUP at the same time. "0" is disable. "1" is enable.

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

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

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

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

4.5.3 General Purpose Communication Service

A GATT database is added for the general purpose communication service.

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

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

 :

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

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

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

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

```
406:  VUART_HDL_SVC,
407:  VUART_HDL_INDICATION_CHAR,
408:  VUART_HDL_INDICATION_VAL,
409:  VUART_HDL_INDICATION_CFG,
410:  VUART_HDL_WRITE_CHAR,
411:  VUART_HDL_WRITE_VAL,
```

4.5.4 rBLE Command Process

Describes the functions that perform the command processing added to the Vendor Specific features of rBLE API.

RBLE_VS_Set_Params function, when more than 0x80 is specified by the first argument param_id calls the RBLE_User_Set_Params function of arch_main.c. RBLE_User_Set_Params function performs the following three processes corresponding to the specified param_id.

- Baud rate setting
- Firmware Update
- Software Reset

RBLE_VS_Flash_Access function calls dataflash_rw function of dataflash.c. Information is read from firmware by the specified param_id.

- Read Firmware Version

A description of the RBLE_User_Set_Params function and the dataflash_rw function is shown below.

(1) **RBLE_User_Set_Params**

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

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

This function will be called when more than 0x80 is specified in RBLE_VS_Set_Params function parameters (param_id) of rBLE API. It can add a process for each param.

- The Baud rate setting saves baud rate id to data flash. Read the baud rate id when the module is started, and initialize the serial driver. After executing the Baud rate setting, execute the reset. A baud rete will valid after reset.
- After executing the Firmware Update, enter to the Firmware Update mode after one second.
- The Software Reset execute internal reset by executing illegal instruction. After executing the Software Reset, an internal reset occurs after one second.

Parameters:

param_id

Parameter ID

Parameter ID	Value	Brief
MODFW_SET_PARAM_ID_BAUDID	0x80	Baud rate ID
MODFW_SET_PARAM_ID_FWUP	0xD9	Move to Firmware Update mode
MODFW_SET_PARAM_ID_SOFTRST	0xFF	Software Reset

param_len

Parameter length

Parameter ID	Parameter length
MODFW_SET_PARAM_ID_BAUDID	UARTBAUD_ID_LEN(1)
MODFW_SET_PARAM_ID_BAUDID	unused
MODFW_SET_PARAM_ID_SOFTRST	unused

*param_data

Pointer to the location in which data is stored.
(the least significant byte first, left justified)

Parameter ID	Baud rate ID
MODFW_SET_PARAM_ID_BAUDID	0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps 6: 250000 bps
MODFW_SET_PARAM_ID_BAUDID	unused
MODFW_SET_PARAM_ID_SOFTRST	unused

Return:

RBLE_OK (0x00)	Success
RBLE_ERR (0xF0)	Data flash save error
RBLE_STATUS_ERROR (0xF2)	Not be executed (rBLE mode is not RBLE_MODE_ACTIVE)
RBLE_UNSUPPORTED (0x11)	Unsupported param_id
RBLE_PARAM_ERR (0xF3)	Parameter error

(2) **dataflash_rw**

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

uint8_t dataflash_rw(const uint8_t mode, const df_rw_t rw, const uint8_t id, uint8_t* addr)		
This function write BD address and baud rate ID to data flash, and read baud rate ID and module firmware version.		
Parameters:		
	mode	Data Flash mode DF_MODE_POLLING(0x00) : Polling Mode DF_MODE_ENFORCED(0xFF) : Enforced Mode
	rw	Read or Write DF_READ(0) : read DF_WRITE(1): write
	id	Data ID(1 ～ 255 byte) EEL_ID_BDA(0) : BD address EEL_ID_REMOTE_BDA(1) : Remote BD address EEL_ID_UARTBAUD(2) : Baud rate ID EEL_ID_MODFW_VER(3) : Read the module firmware version
	**addr*	Pointer to the location in which data is stored.
Return:		
	DF_OK (0x00)	Success
	DF_ERR_BUSY (0x01)	Data Flash busy state
	DF_ERR_DISALLOWED (0x03)	Execution disallowed

4.5.5 Firmware Function

Additional functions for the module are shown below.

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

(1) **r_modfw_init**

void r_modfw_init(void)		
Initialize port of the module.		
Parameters:		
none		
Return:		
none		

(2) **r_modfw_flash_write_uart_baudrate_id**

RBLE_STATUS r_modfw_flash_write_uart_baudrate_id(uint8_t param_len, uint8_t *param_data)		
Will be called from RBLE_User_Set_Params function, and write baud rate ID to Data Flash.		
Parameters:		
param_len	Baud rate length	
**param_data*	Baud rate ID	
Return:		
RBLE_OK (0x00)	Normal operation	
RBLE_ERR (0xF0)	Data Flash function Failure	
RBLE_PARAM_ERR (0xF3)	Parameter error	

(3) **r_modfw_flash_read_uart_baudrate_id**

void r_modfw_flash_read_uart_baudrate_id(void)
Will be called startup of the module, and read baud rate ID from Data Flash.
Parameters:
none
Return:
none

(4) **r_modfw_force_reset**

void r_modfw_force_reset(void)
Will be called RBLE_User_Set_Params function. Wait for one second, execute software reset from interrupt handler (r_modfw_inttm00_isr).
Parameters:
none
Return:
none

(5) **r_modfw_update**

void r_modfw_update(void)	
Will be called from RBLE_User_Set_Params function. Wait for one second, execute firmware update from interrupt handler (r_modfw_inttm00_isr).	
Parameters:	
none	
Return:	
none	

(6) **r_modfw_get_ver**

void r_modfw_get_ver(uint8_t* addr)	
Will be called from dataflash_rw function. Return the firmware version.	
Parameters:	
**addr*	Pointer to the location in which the firmware version is stored.
Return:	
none	

(7) **r_modfw_set_timer_1s**

void r_modfw_set_timer_1s(void)
For timer to wait for one second, initialize and start timer00.
Parameters:
none
Return:
none

(8) **r_modfw_stop_timer**

void r_modfw_stop_timer(void)
Stop timer00.
Parameters:
none
Return:
none

(9) **r_modfw_inttm00_isr**

void r_modfw_inttm00_isr(void)
Interrupt handler of timer00. Execute the software reset and the firmware update.
Parameters:
none
Return:
none

4.5.6 RF Slow Clock

The firmware uses the on-chip oscillator for the RF slow clock (32.768 kHz) at initial setting.

If crystal oscillator connected to the XT1 pin and the XT2 pin is used as RF slow clock, delete "no" from following macro, and specified "CLK_SUB_XT1".

Compiler Macro Definition

noCLK_SUB_XT1

1. Launch Renesas e^2 studio.
2. Select [File] - [Import].
3. Select [General] - [Existing Projects into Workspace] and click [Next>].
4. Set the following path to [Select root directory] and make sure that rBLE_Mdm is shown in [Project].
   - \Renesas_BLE_Module_V101\RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem\rBLE_Mdm
5. Click [Finish].
6. Right click on [Project Explorer] - [rBLE_Mdm]. And select [Renesas Tool Settings].
7. Select [Compiler] - [Source]. Display [Macro definition] text box.
   - \Renesas_BLE_Module_V101\RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem\rBLE_Mdm\DefaultBuild

4.6 How to Build

It shows the how to build of the firmware. Refer to "2.3 Build Preparation", and copy libraries to the folders.

- Renesas Bluetooth Low Energy protocol stack V1.11 libraries for CC-RL
- EEPROM Emulation Library Pack02 Ver.1.01 for the CC-RL Compiler for the RL78 Family
- Flash Self Programming Library Type01 for the CC-RL Compiler for RL78 Family Ver.2.21

4.6.1 Build of Firmware

8. Launch Renesas e^2 studio.
9. Select [File] - [Import].
10. Select [General] - [Existing Projects into Workspace] and click [Next>].
11. Set the following path to [Select root directory] and make sure that rBLE_Mdm is shown in [Project].
    - \Renesas_BLE_Module_V101\RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem\rBLE_Mdm
12. Click [Finish].
13. Right click on [Project Explorer] - [rBLE_Mdm]. And select [Build Project].
14. The firmware update file "rBLE_Mdm_CCRL.hex" is generated in the following path.
    - \Renesas_BLE_Module_V101\RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem\rBLE_Mdm\DefaultBuild

4.6.2 Firmware Update File

Binary files used by the firmware update, will be created in the same folder as the firmware "rBLE_Mdm_CCRL.hex". The name of the file that is created is "rBLE_Mdm_CCRL.bin".

When using by an operations check of a firmware update, please change the binary file name to "RL78_G1D_MODULE.bin". About how to use, refer to "RL78/G1D Module Firmware User's Manual"(R01UW0160).

Precautions:

- Firmware updates can change the profile.
- Can't change the arrangement of a section.
- Binary file used in the firmware update, must be the same as the firmware memory arrangement other than the user code section. Be sure to confirm the difference with the binary file of firmware. About user code section arrangement, refer to color of "4.4 Memory map".

Option setting for creating a binary file is shown below.

1. Right click on [Project Explorer] - [rBLE_Mdm]. And select [Renesas Tool Settings].
2. Set the below command of settings menu - [Build Steps] tab - [Post-build steps].
   - rlink -form=binary "rBLE_Mdm_CCRL.abs" -output="rBLE_Mdm_CCRL.bin"=0-3ffff -space=ff
3. Right click on [Project Explorer] - [rBLE_Mdm]. And select [Build Project].
4. The firmware update file "rBLE_Mdm_CCRL.bin" is generated in the following path.
   - \Renesas_BLE_Module_V101\RL78_G1D\Project_Source\renesas\tools\project\e2studio\BLE_Modem\rBLE_Mdm\DefaultBuild

5. Appendix

5.1 Code Flash Memory Rewriting Setting

A shipping check flag and Bluetooth Device Address are written to block 254 and 255 of a cord flash memory. If the firmware is downloaded to the module using a IDE or a E1 emulator, please in the setting that does not erase the memory.

- e^2 studio

1. Right click on [Project Explorer] - [rBLE_Mdm]. And select [Run/Debug Setting].
2. Select [rBLE_Mdm_CCRL.x]. Click [Edit].
3. Select [Debugger] tab - [Connection Settings] tab.
4. Select [Flash] - [Erase Flash ROM When Starting]
5. Select [No]. Click [OK].
6. Click [OK].

- CS+

1. Select [Project tree] - [(Project name)(Sub project)] - [RL78 E1(Serial)(Debug tool)]. Display [Property] tab.
2. Select [Download file setting] - [Download] - [Select download mode]. Select [Data priority]

5.2 Folder List

5.2.1 Module Firmware V1.01

(1) **Root Folder**

Renesas_BLE_Module_V101	
├ BLE_Sample	Sample program folder for Host MCU
├ RL78_G1D	Module firmware folder
└ ROM_File	Hex file of module firmware folder
└ rBLE_Mdm_CCRL_module_fw_v101.hex	Hex file of module firmware v1.01

(2) **Sample Program Folder for Host MCU**

BLE_Sample	
├ project	
│ ├ CS_CCRL	
│ │ └ BLE_Sample	
│ │ ├ rBLE_sample	
│ │ │ └ rBLE_sample.mtsp	CS+ for CC sub project file
│ │ └ BLE_Sample.mtpj	CS+ for CC project file
│ ├ e2studio	Project folder for e^2 studio
│ │ └ BLE_Sample	
│ │ └ rBLE_sample	
│ │ ├ .cproject	Build option file
│ │ ├ .DefaultBuildlinker	Linker file
│ │ ├ .info	Tool chain file
│ │ ├ .project	Project file
│ │ └ rBLE_sample_CCRL.x.launch	Debug information file
│ └ windows	Project folder for Visual Studio Express 2015 (PC)
│ ├ Exe	
│ │ ├ rBLE_Sample.exe	Sample program (executable file for PC)
│ │ ├ run_master.bat	Batch file to start sample program
│ │ └ run_slave.bat	Batch file to start sample program
│ ├ rBLE_Sample.sln	
│ ├ rBLE_Sample.vcxproj	Microsoft Visual Studio Express 2015 project file
│ └ rBLE_Sample.vcxproj.filters	
└ src	
├ Platform	Platform dependent source files
│ ├ G1D_cs_iar	For RL78/G1D
│ │ ├ driver	
│ │ │ ├ console	
│ │ │ │ ├ Console.c	Sample driver (console driver) C source file
│ │ │ │ └ console.h	Sample driver (console driver) header file
│ │ │ ├ led	
│ │ │ │ ├ led.c	Sample driver (LED driver) C source file
│ │ │ │ └ led.h	Sample driver (LED driver) header file
│ │ │ ├ plf	
│ │ │ │ ├ plf.c	Sample driver (platform driver) C source file
│ │ │ │ └ plf.h	Sample driver (platform driver) header file
│ │ │ ├ serial	
│ │ │ │ ├ csi.c	Sample driver (CSI driver) C source file
│ │ │ │ ├ csi.h	Sample driver (CSI driver) header file
│ │ │ │ ├ iic_master.c	Sample driver (serial comm. driver) header file
│ │ │ │ ├ iic_master.h	Sample driver (terminal driver) C source file
│ │ │ │ ├ serial.h	Sample driver (terminal driver) header file
│ │ │ │ ├ term_uart.c	Sample driver (UART driver) C source file
│ │ │ │ ├ term_uart.h	Sample driver (UART driver) header file
│ │ │ │ ├ uart.c	Sample driver (IIC master) C source file
│ │ │ │ └ uart.h	Sample driver (IIC master) C source file

│ │ │ └ timer			
│ │ │ ├ timer.c	Sample driver (timer driver) C source file		
│ │ │ └ timer.h	Sample driver (timer driver) header file		
│ │ ├ include			
│ │ │ ├ arch.h	BLE software basic macro and function declaration file		
│ │ │ ├ compiler.h	Compiler-dependent header file		
│ │ │ ├ iodefine.h	Device header file for CC-RL		
│ │ │ ├ ll.h	Low-level macro definition header file		
│ │ │ ├ port.h	GPIO port driver header file		
│ │ │ ├ rscip_api.h	RSCIP API header file		
│ │ │ └ types.h	Common type definition header file		
│ │ ├ cstart.asm	Startup routine file for CC-RL		
│ │ ├ hdwinit.asm	Initialize hardware file for CC-RL		
│ │ ├ rBLE_Sample.c	Sample file for RL78/G1D		
│ │ └ stkinit.asm	Initialize stack area file for CC-RL		
│ └ win32_vc	For x86 PC		
│ ├ driver			
│ │ ├ console			
│ │ │ ├ Console.c	Sample driver (console input) C source file		
│ │ │ └ console.h	Sample driver (console input) header file		
│ │ ├ serial			
│ │ │ ├ serial_drv.h	Sample driver (serial) interface header file		
│ │ │ ├ Serial_drv_sample.cpp	Sample driver (serial) interface file		
│ │ │ ├ SerialIF.cpp	Sample driver (serial) C++ source file		
│ │ │ └ SerialIF.h	Sample driver (serial) header file		
│ │ └ timer			
│ │ ├ timer.cpp	Sample driver (timer driver) C source file		
│ │ └ timer.h	Sample driver (timer driver) header file		
│ ├ include			
│ │ └ types.h	Header file for computer		
│ ├ rBLE_Sample.cpp	Sample file for computer		
│ ├ StdAfx.cpp			
│ └ StdAfx.h			
└ rBLE	Platform independent source files		
└ src			
├ host			
│ ├ anp			
│ │ ├ rble_api_anpc.c	rBLE Host ANP Client file	rBLE_Host	
│ │ └ rble_api_anps.c	rBLE Host ANP Server file	rBLE_Host	
│ ├ blp			
│ │ ├ rble_api_blpc.c	rBLE Host BLP Collector file	rBLE_Host	
│ │ └ rble_api_blps.c	rBLE Host BLP Sensor file	rBLE_Host	
│ ├ cpp			
│ │ ├ rble_api_cppc.c	rBLE Host CPP Collector file	rBLE_Host	
│ │ └ rble_api_cpps.c	rBLE Host CPP Sensor file	rBLE_Host	
│ ├ cscp			
│ │ ├ rble_api_cscpc.c	rBLE Host CSCP Collector file	rBLE_Host	
│ │ └ rble_api_cscps.c	rBLE Host CSCP Sensor file	rBLE_Host	
│ ├ fmp			
│ │ ├ rble_api_fmpl.c	rBLE Host FMP Locator file	rBLE_Host	
│ │ └ rble_api_fmpt.c	rBLE Host FMP Target file	rBLE_Host	
│ ├ gap			
│ │ └ rble_api_gap.c	rBLE Host GAP file	rBLE_Host	
│ ├ gatt			
│ │ └ rble_api_gatt.c	rBLE Host GATT file	rBLE_Host	
│ ├ glp			
│ │ ├ rble_api_glpc.c	rBLE Host GLP Collector file	rBLE_Host	
│ │ └ rble_api_glps.c	rBLE Host GLP Sensor file	rBLE_Host	
│ ├ hogp			
│ │ ├ rble_api_hgbh.c	rBLE Host HOGP Boot Host file	rBLE_Host	
│ │ ├ rble_api_hghd.c	rBLE Host HOGP HID Device file	rBLE_Host	

│ │ └ rble_api_hgrh.c	rBLE Host HOGP Report Host file	rBLE_Host
│ ├ hrp		
│ │ ├ rble_api_hrpc.c	rBLE Host HRP Collector file	rBLE_Host
│ │ └ rble_api_hrps.c	rBLE Host HRP Sensor file	rBLE_Host
│ ├ htp		
│ │ ├ rble_api_htpc.c	rBLE Host HTP Collector file	rBLE_Host
│ │ └ rble_api_htpt.c	rBLE Host HTP Thermometer file	rBLE_Host
│ ├ lnp		
│ │ ├ rble_api_lnpc.c	rBLE Host LNP Collector file	rBLE_Host
│ │ └ rble_api_lnps.c	rBLE Host LNP Sensor file	rBLE_Host
│ ├ pasp		
│ │ ├ rble_api_paspc.c	rBLE Host PASP Client file	rBLE_Host
│ │ └ rble_api_pasps.c	rBLE Host PASP Server file	rBLE_Host
│ ├ pxp		
│ │ ├ rble_api_pxpm.c	rBLE Host PXP Monitor file	rBLE_Host
│ │ └ rble_api_pxpr.c	rBLE Host PXP Reporter file	rBLE_Host
│ ├ rscp		
│ │ ├ rble_api_rscpc.c	rBLE Host RSCP Collector file	rBLE_Host
│ │ └ rble_api_rscps.c	rBLE Host RSCP Sensor file	rBLE_Host
│ ├ scpp		
│ │ ├ rble_api_sppc.c	rBLE Host ScPP Client file	rBLE_Host
│ │ └ rble_api_spps.c	rBLE Host ScPP Server file	rBLE_Host
│ ├ sm		
│ │ └ rble_api_sm.c	rBLE Host SMP file	rBLE_Host
│ ├ tip		
│ │ ├ rble_api_tipc.c	rBLE Host TIP Client file	rBLE_Host
│ │ └ rble_api_tips.c	rBLE Host TIP Server file	rBLE_Host
│ ├ vs		
│ │ └ rble_api_vs.c	rBLE Host VS file	rBLE_Host
│ ├ rble_host.c	rBLE Host file	rBLE_Host
│ └ rble_if_api_cb.c	rBLE Host Event file	rBLE_Host
├ include		
│ ├ host		
│ │ └ rble_host.h	rBLE Host header file	rBLE_Host
│ ├ prf_sel.h	Profile configuration file	
│ ├ rble.h	rBLE header file	rBLE_Host
│ ├ rble_api.h	rBLE API header file	rBLE_Host
│ ├ rble_api_custom.h	Custom profile additional API header file	
│ ├ rble_api_fwup.h	FW Update profile additional API header file	
│ ├ rble_app.h	Sample program header file	
│ └ rble_trans.h	rBLE packet data header file	rBLE_Host
├ rscip		
│ ├ rscip.c	RSCIP interface file	RSIP
│ ├ rscip.h	RSCIP header file	RSIP
│ ├ rscip_cntl.c	RSCIP control file	RSIP
│ ├ rscip_cntl.h	RSCIP control header file	RSIP
│ ├ rscip_ext.h	RSCIP external request header file	RSIP
│ ├ rscip_uart.c	RSCIP serial file	RSIP
│ └ rscip_uart.h	RSCIP serial header file	RSIP
├ sample_app	Sample application folder	
│ ├ menu_sel.c	Menu selection sample file	
│ ├ menu_sel.h	Menu selection sample header file	
│ ├ r_vuart_app.c	General purpose communication application file	
│ ├ r_vuart_app.h	General purpose communication application header file	
│ ├ r_vuart_app_param.c	General purpose communication application parameter file	
│ ├ r_vuart_console.c	General purpose communication console driver file	
│ ├ r_vuart_console.h	General purpose communication console driver header file	
│ ├ rble_fw_up_sender_app.c	Sample Program file for FW Update	
│ ├ rble_sample_app.c	Sample program file	
│ └ rble_sample_parameter.c	Sample program parameter file for PTS	

└ sample_profile	Sample profile folder
├ fwup	FW Update profile
│ └ fwups.c	FW Update profile sender file
├ scp	Sample custom profile
│ ├ scpc.c	Sample custom profile client file
│ └ scps.c	Sample custom profile server file
├ vuart	General purpose communication
│ ├ vuart.h	General purpose communication common header file
│ ├ vuartc.c	General purpose communication client file
│ ├ vuartc.h	General purpose communication client header file
│ ├ vuarts.c	General purpose communication server file
│ └ vuarts.h	General purpose communication server header file
└ db_handle.h	Attribute database handle header file

(3) Module Firmware Folder

RL78_G1D	
└ Project_Source	
├ bleip	BLE stack folder
│ └ src	
│ ├ common	
│ │ └ co_bt.h	BLE stack common header file
│ └ rwble	
│ ├ rwble.h	BLE stack entry-point header file
│ ├ rwble_config.h	BLE stack configuration header file
│ └ rwble_mem_cont.h	BLE stack configuration header file
├ rBLE	
│ └ src	
│ ├ include	
│ │ ├ rble.h	rBLE header file
│ │ ├ rble_api.h	rBLE API header file
│ │ ├ rble_api_custom.h	Custom profile additional API header file
│ │ ├ rble_api_fwup.h	FW Update profile additional API header file
│ │ ├ rble_api_vuart.h	General purpose communication additional API header file
│ │ ├ rble_app.h	Sample program header file
│ │ ├ rble_rwke.h	RWKE API header file
│ │ ├ rble_trans.h	rBLE packet data header file
│ │ └ rscip_api.h	RSCIP API header file
│ ├ sample_app	
│ │ ├ Console.c	Sample program (console) C source file
│ │ ├ console.h	Sample program (console) header file
│ │ ├ menu_sel.c	Sample program (menu) C source file
│ │ ├ menu_sel.h	Sample program (menu) header file
│ │ ├ rble_fw_up_receiver_app.c	Sample program for FW Update
│ │ ├ rble_sample_app.c	Sample program file
│ │ └ rble_sample_parameter.c	Sample program parameter file for the PTS
│ └ sample_profile	
│ ├ fwup	FW Update profile
│ │ └ fwupr.c	FW Update profile receiver file
│ └ scp	Sample custom profile folder
│ ├ scpc.c	Sample custom profile client file
│ └ scps.c	Sample custom profile server file
└ renesas	Renesas folder
├ config	
│ └ split	
│ └ emb	
│ ├ r_lk.dr	Link directive file for Embedded configuration
│ ├ r_lk_fw_emb.dr	Link directive file for FW Update feature on Embedded configuration
│ ├ r_lk_fw_modem.dr	Link directive file for FW Update feature on Modem configuration
│ └ r_lk_modem.dr	Link directive file for Modem configuration

├ lib	BLE software library folder
├ src	
│ ├ arch	
│ │ └ rl78	
│ │ ├ ll	
│ │ │ └ ll.h	Low-level macro definition header file
│ │ ├ arch.h	BLE software basic macro and function declaration file
│ │ ├ arch_main.c	BLE software main file
│ │ ├ config.h	BLE software configuration header file
│ │ ├ db_handle.h	Attribute database handles header file
│ │ ├ fw_update_count0.c	Firmware update rewrite counter file
│ │ ├ fw_update_count1.c	Firmware update rewrite counter file
│ │ ├ hw_config.h	BLE software H/W configuration header file
│ │ ├ ke_conf.c	RWKE task management file
│ │ ├ main.c	
│ │ ├ prf_config.c	Profile parameter file
│ │ ├ prf_config.h	Profile parameter header file
│ │ ├ prf_config_host.c	Profile parameter file
│ │ ├ prf_sel.h	Profile selection configuration header file
│ │ ├ rble_core_config.c	rBLE_Core parameter file
│ │ ├ rble_core_config.h	rBLE_Core parameter header file
│ │ ├ rble_modem_config.c	MDM_APPL parameter file
│ │ ├ rble_modem_config.h	MDM_APPL parameter header file
│ │ ├ rwble_mem.c	Heap management parameter file
│ │ ├ rwble_mem.h	Heap management parameter header file
│ │ └ rwke_api.h	RWKE API header file
│ ├ compiler	
│ │ ├ ccrl	
│ │ │ ├ cstart.asm	Startup routine file for CC-RL
│ │ │ ├ hdwinit.asm	Initialize hardware file for CC-RL
│ │ │ └ stkinit.asm	Initialize stack area file for CC-RL
│ │ ├ compiler.h	Device header file for CC-RL
│ │ └ iodefine.h	Compiler-dependent header file
│ ├ driver	
│ │ ├ codeflash	
│ │ │ ├ cc_rl	Code flash library folder for CC-RL
│ │ │ ├ codeflash.c	Code flash driver file
│ │ │ └ codeflash.h	Code flash driver header file
│ │ ├ csi	
│ │ │ ├ csi.c	CSI driver file
│ │ │ └ csi.h	CSI driver header file
│ │ ├ dataflash	
│ │ │ ├ cc_rl	Data flash library folder for CC-RL
│ │ │ ├ dataflash.c	Data flash driver file
│ │ │ ├ dataflash.h	Data flash driver header file
│ │ │ ├ eel_descriptor.c	EEPROM descriptor file for EEL Pack01
│ │ │ ├ eel_descriptor.h	EEPROM descriptor header file for EEL Pack01
│ │ │ ├ eel_descriptor_t02.c	EEPROM descriptor file for EEL Pack02
│ │ │ ├ eel_descriptor_t02.h	EEPROM descriptor header file for EEL Pack02
│ │ │ ├ fal_descriptor.c	Flash descriptor file for EEL Pack01
│ │ │ ├ fal_descriptor.h	Flash descriptor header file for EEL Pack01
│ │ │ ├ fdl_descriptor_t02.c	Flash descriptor file for EEL Pack02
│ │ │ └ fdl_descriptor_t02.h	Flash descriptor header file for EEL Pack02
│ │ ├ DTM2Wire	
│ │ │ ├ DTM2Wire.c	2-wire UART Direct Test Mode driver file
│ │ │ └ DTM2Wire.h	2-wire UART Direct Test Mode driver header file
│ │ ├ iic	
│ │ │ ├ iic_slave.c	IIC slave driver file
│ │ │ └ iic_slave.h	IIC slave driver header file
│ │ ├ led	
│ │ │ ├ led.c	LED driver file
│ │ │ └ led.h	LED driver header file

├ module	
├ module.c	Module fw file
└ module.h	Module fw header file
├ peak	
├ peak.h	Peak current consumption notification software interrupt file
└ peak_isr.c	Peak current consumption notification driver header file
├ pktmon	
└ pktmon.h	HCI packet monitor driver header file
├ plf	
├ plf.c	RL78 platform driver file
└ plf.h	RL78 platform driver header file
├ port	
└ port.h	GPIO port driver header file
├ push_sw	
├ push_sw.c	Push switch driver file
└ push_sw.h	Push switch driver header file
├ rf	
└ rf.h	RF driver common header file
├ serial	
└ serial.h	Serial driver header file
├ uart	
├ uart.c	UART driver file
└ uart.h	UART driver header file
└ wakeup	
├ wakeup.c	Wakeup driver file
└ wakeup.h	Wakeup driver header file
└ types.h	Common type definition header file
└ tools	
└ project	
├ CS_CCRL	Project folder for CS+ for CC
└ BLE_Modem	
├ rBLE_Mdm	
├ rBLE_Mdm.mtsp	Sub project file for Modem configuration
└ sect_mdm_modfw.hsi	Section import file for Modem configuration
└ BLE_Modem.mtpj	Project file for Modem configuration
└ e2studio	Project folder for e2studio
└ BLE_Modem	
└ rBLE_Mdm	
├ .cproject	Build option file for module fw
├ .DefaultBuildlinker	Linker file for Modem configuration for module fw
├ .info	Tool chain file for module fw
├ .project	Project file for module fw
├ rBLE_Mdm_CCRL.x.launch	Debug information file for module fw
└ sect_mdm_modfw.esi	Section inport file for module fw

5.2.2 Module Firmware V1.11

It will be described only the updated files.

(1) **Root Folder**

Renesas_BLE_Module_V111		
├ BLE_Sample	Sample program folder for Host MCU	
├ RL78_G1D	Module firmware folder	
└ ROM_File	Hex file of module firmware folder	
└ rBLE_Mdm_CCRL_module_fw_v111.hex	Hex file of module firmware v1.11	

(2) **Sample Program Folder for Host MCU**

BLE_Sample		
├ project		
│ ├ CS_CCRL		
│ │ └ BLE_Sample		
│ │ ├ rBLE_sample		
│ │ │ └ rBLE_sample.mtsp	CS+ for CC sub project file	
│ │ └ BLE_Sample.mtpj	CS+ for CC project file	
│ ├ e2studio	Project folder for e^2 studio	
│ │ └ BLE_Sample		
│ │ └ rBLE_sample		
│ │ ├ .cproject	Build option file	
│ │ ├ .DefaultBuildlinker	Linker file	
│ │ ├ .info	Tool chain file	
│ │ ├ .project	Project file	
│ │ └ rBLE_sample_CCRL.x.launch	Debug information file	
│ └ windows	Project folder for Visual Studio Express 2015 (PC)	
│ ├ Exe		
│ │ ├ rBLE_Sample.exe	Sample program (executable file for PC)	
│ │ ├ run_master.bat	Batch file to start sample program	
│ │ └ run_slave.bat	Batch file to start sample program	
│ ├ rBLE_Sample.sln		
│ ├ rBLE_Sample.vcxproj	Visual Studio Express 2015 project file	
│ └ rBLE_Sample.vcxproj.filters		
└ src		
├ Platform	Platform dependent source files	
│ ├ G1D_cs_iar	For RL78/G1D	
│ │ └ driver		
│ │ ├ plf		
│ │ │ └ plf.c	Sample driver (platform driver) C source file	
│ │ └ serial		
│ │ ├ csi.c	Sample driver (CSI driver) C source file	
│ │ ├ csi.h	Sample driver (CSI driver) header file	
│ │ └ uart.c	Sample driver (IIC master) C source file	
│ └ win32_vc	For x86 PC	
│ └ driver		
│ └ console		
│ ├ Console.c	Sample driver (console input) C source file	
│ └ console.h	Sample driver (console input) header file	
└ rBLE	Platform independent source files	
└ src		
├ host		
│ └ rble_if_api_cb.c	rBLE Host Event file	rBLE_Host
├ include		
│ ├ prf_sel.h	Profile configuration file	
│ ├ rble.h	rBLE header file	rBLE_Host
│ ├ rble_api.h	rBLE API header file	rBLE_Host

│ └ rble_app.h	Sample program header file
├ sample_app	Sample application folder
│ ├ r_vuart_app.c	General purpose communication application file
│ ├ r_vuart_app.h	General purpose communication application header file
│ ├ r_vuart_app_param.c	General purpose communication application parameter file
│ ├ r_vuart_console.c	General purpose communication console driver file
│ ├ r_vuart_console.h	General purpose communication console driver header file
│ ├ rble_fw_up_sender_app.c	Sample Program file for FW Update
│ ├ rble_sample_app.c	Sample program file
│ ├ rble_sample_anp.c	"
│ ├ rble_sample_blp.c	"
│ ├ rble_sample_cpp.c	"
│ ├ rble_sample_cscp.c	"
│ ├ rble_sample_custom.c	"
│ ├ rble_sample_fmp.c	"
│ ├ rble_sample_fwup.c	"
│ ├ rble_sample_gap_sm_gatt.c	"
│ ├ rble_sample_glp.c	"
│ ├ rble_sample_hogp.c	"
│ ├ rble_sample_hrp.c	"
│ ├ rble_sample_htp.c	"
│ ├ rble_sample_lnp.c	"
│ ├ rble_sample_pasp.c	"
│ ├ rble_sample_pxp.c	"
│ ├ rble_sample_rscp.c	"
│ ├ rble_sample_scpp.c	"
│ ├ rble_sample_tip.c	"
│ ├ rble_sample_vendor.c	"
│ └ rble_sample_vuart.c	"
└ sample_profile	
├ scp	Sample custom profile
│ └ scpc.c	Sample custom profile client file
├ vuart	General purpose communication
│ ├ vuartc.c	General purpose communication client file
│ ├ vuarts.c	General purpose communication server file
│ └ vuarts.h	General purpose communication server header file
└ db_handle.h	Attribute database handle header file

(3) Module Firmware Folder

RL78_G1D	
└ Project_Source	
├ rBLE	
│ └ src	
│ ├ include	
│ │ ├ rble.h	rBLE header file
│ │ ├ rble_api.h	rBLE API header file
│ │ └ rble_app.h	Sample program header file
│ └ sample_app	
│ └ rble_fw_up_receiver_app.c	Sample program for FW Update
└ renesas	Renesas folder
├ lib	BLE software library folder
├ src	
│ ├ arch	
│ │ └ rl78	
│ │ ├ arch_main.c	BLE software main file
│ │ ├ config.h	BLE software configuration header file
│ │ ├ db_handle.h	Attribute database handles header file
│ │ ├ ke_conf.c	RWKE task management file
│ │ ├ main.c	

│ │ ├ prf_config.c	Profile parameter file
│ │ ├ prf_config.h	Profile parameter header file
│ │ ├ prf_sel.h	Profile selection configuration header file
│ │ └ rwble_mem.c	Heap management parameter file
│ ├ compiler	
│ │ └ compiler.h	Device header file for CC-RL
│ └ driver	
│ ├ dataflash	
│ │ ├ dataflash.c	Data flash driver file
│ │ ├ dataflash.h	Data flash driver header file
│ │ └ fdl_descriptor_t02.h	Flash descriptor header file for EEL Pack02
│ ├ DTM2Wire	
│ │ └ DTM2Wire.c	2-wire UART Direct Test Mode driver file
│ ├ module	
│ │ └ module.c	Module fw file
│ ├ plf	
│ │ └ plf.c	RL78 platform driver file
│ ├ push_sw	
│ │ └ push_sw.c	Push switch driver file
│ ├ uart	
│ │ └ uart.c	UART driver file
│ └ wakeup	
│ └ wakeup.c	Wakeup driver file
└ tools	
└ project	
├ CS_CCRL	Project folder for CS+ for CC
│ └ BLE_Modem	
│ ├ rBLE_Mdm	
│ │ └ rBLE_Mdm.mtsp	Sub project file for Modem configuration
│ └ BLE_Modem.mtpj	Project file for Modem configuration
└ e2studio	Project folder for e2studio
└ BLE_Modem	
└ rBLE_Mdm	
├ .info	Tool chain file for module fw
└.project	Project file for module fw

5.3 Support of BLE Software V1.21

Describes the updates content of the Module firmware which corresponds to BLE software V1.21 and the module firmware. Please also refer to the following document.

- Bluetooth Low Energy Protocol Stack User's Manual (R01UW0095)
- API Reference Manual : Basics (R01UW0088)
- BLE Virtual UART Application (R01AN3130)

5.3.1 Module Firmware

BLE software V1.21:

- Fixed an issue that the pairing or the connection after the pairing can be failed when using RPA (Resolvable Private Address) as a local BD address and pairing with Smartphone.
- Fixed an issue that communication with the peer device becomes unstable when operating as a Slave device under the following operation setting.
  - Set Main System Clock to 4MHz, use On-chip oscillator for RF slow clock (32.768kHz), and set Maximum Number of Simultaneous Connections (CFG_CON) to 1.
  - Set Main System Clock to 16MHz, and use CC-RL compiler.
- Fixed an issue that access to data flash can not be done when external clock is used for CPU clock supply.
- Comply with the RL78/G1D technical update (TN-RL*-A083A), changed unused internal pin settings. See technical update for the detail.

BLE software V1.20:

- Change the event to be notified at the time of encryption start, and added a new event RBLE_SM_LTK_REQ_FOR_ENC_IND.
- Fixed issue: In the case of a serial communication method using the WAKEUP signal in the Modem configuration may not be able to receive data after the WAKEUP request.
- Fixed issue: Cannot be respond to the Service Discovery from a second Slave in multi-connections in the Master Role.
- The module firmware version is changed to V1.11.

5.3.2 Host MCU Program

BLE software V1.21:

- Fix to be able to specify 20 bytes as the second argument of RBLE_VUART_Server_Send_Indication at Server role of the general purpose communication.

BLE software V1.20:

- Fixed the maximum size of long characteristic value define from 80 to 72 (RBLE_GATT_MAX_LONG_VALUE).
- Simple AT Command Mode : Add commands AT-CI=<con_intv>, AT-CI?, ATE0, ATE1.

5.3.3 Build Environment

- e2 studio V5.4.0.015/RL78 Compiler CC-RL V1.03.00

5.3.4 How to Build

To be read the folder name as follows and run a build.

BLE_Software_Ver_1_11 → BLE_Software_Ver_1_21

Renesas_BLE_Module_V101 → Renesas_BLE_Module_V111

(1) **Module Firmware**

Refer to "2.3 Build Preparation" and copy library files to each folders. Refer to "4.6.1 Build of Firmware" and run a build. (Library file name of Bluetooth Low Energy protocol stack (*_CCRL.lib) is the same of V1.11.)

(2) **Host MCU Program**

Refer to "4.6 How to Build" and run a build.

5.4 Referenced Documents

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

5.5 Terminology

Term	Description
Service	A service is provided from a GATT server to a GATT client. The GATT server exposes some characteristics as the interface. The service prescribes how to access the exposed characteristics.
Profile	A profile enables implementation of a use case by using one or more services. The services used are defined in the specifications of each profile.
Characteristic	A characteristic is a value used to identify services. The characteristics to be exposed and their formats are defined by each service.
Role	Each device takes the role prescribed by the profile or service in order to implement the specified use case.
Client Characteristic Configuration Descriptor	A descriptor is used to control notifications or indications of characteristic values that include the client characteristic configuration descriptor sent from the GATT server.
Universally Unique Identifier (UUID)	This is an identifier for uniquely identifying an item. In the BLE standard, a 16-bit UUID is defined for identifying services and their characteristics.
Bluetooth Device Address (BD Address)	This is a 48-bit address for identifying a Bluetooth device. The BLE standard defines both public and random addresses, and at least one or the other must be supported.

5.6 List of Abbreviations and Acronyms

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

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

Abbreviation	Full Form	Remark
RSCIP	Renesas Serial Communication Interface Protocol	
RWKE	Renesas Wireless Kernel Extension	
VS	Vendor Specific	

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	July 08, 2016	-	First Release.
1.10	Oct 21, 2016	-	The source code of the module firmware corresponding to the BLE software V1.20 and Host MCU program were bundled.
		-	Change the folder name of the sample program as follows. • Renesas_BLE_Module_V101 • Renesas_BLE_Module_V110
		57	Add the folder list of the Module firmware V1.10.
		60	Add the update content that corresponds to the BLE software V1.20.
1.11	Nov 01, 2017	-	Update the BLE software version from V1.20 to V1.21.
		-	The source code of the module firmware corresponding to the
		-	BLE software V1.21 and Host MCU program were bundled.
		-	Change the folder name of the sample program as follows. • Renesas_BLE_Module_V111
		-	Delete the module firmware corresponding to BLE software V1.20 and source code of Host MCU program.
		57	Describe the folder list of the module firmware V1.11.
		60	Describe the update content that corresponds to the BLE software V1.21.
1.11	Jan 31, 2022	-	Fixed due to the end of support for the following in Bluetooth Low Energy Protocol Stack. - Adopted profiles by the Bluetooth SIG

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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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 systems; anti-disaster systems; anti-crime systems; and safety 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 (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable 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. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages 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 its 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 be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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. Because the evaluation of microcomputer software alone is very difficult, 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 use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(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.

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, HALII 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 5.0