Bluetooth® Low Energy Protocol Stack

RL78/G14 Host Sample

R01AN2807EJ0121
Rev.1.21
Jan 31, 2022

Introduction

This manual describes the device composition, software composition, procedure of checking operation, details of software processing and software sequence about the host sample.

The host sample works on the Renesas Starter Kit for RL78/G14, and controls the Renesas Bluetooth low energy microcontroller RL78/G1D device programmed with Bluetooth Low Energy protocol by serial communication.

Target Device

Renesas Starter Kit for RL78/G14

Related documents

Document Name	Document No.
Bluetooth Low Energy Protocol Stack	
User's Manual	R01UW0095E
API Reference Manual: Basics	R01UW0088E
Application Note: Sample Program	R01AN1375E
Application Note: rBLE Command Specification	R01AN1376E
RL78/G1D	
User's Manual: Hardware	R01UH0515E
RL78/G1D Evaluation Board	
User's Manual	R30UZ0048E
RL78/G14	
User's Manual: Hardware	R01UH0186E
Renesas Starter Kit for RL78/G14	
User's Manual	R20UT0785E
Tutorial Manual	R20UT0786E
Quick Start Guide	R20UT0787E
CPU Board Schematics	R20UT0784E

Contents

1. Overview

This manual describes the device composition, software composition, procedure of checking operation, details of software processing and software sequence about the host sample.

The host sample works on the Renesas Starter Kit for RL78/G14 (RSK), and controls the Renesas Bluetooth low energy microcontroller RL78/G1D device programmed with Bluetooth Low Energy protocol by serial communication. Serial communication between the RSK and the RL78/G1D evaluation board supports the UART 2-wire connection[NOTE1] and UART 2-wire with branch connection[NOTE2].

For details about the BLE protocol stack APIs, refer to Bluetooth Low Energy Protocol Stack API Reference Manual.(R01UW0088)

Notes: 1. UART 2-wire connection is a communication method using TxD and RxD of the UART data signal line.
2. UART 2-wire with branch connection is expanding 2-wire UART. The TxD line is branched and connected to the WAKEUP of modules from Host MCU. About connection between Host MCU and BLE MCU, refer to "4.3.3 Example of Application Circuit".

1.1 Environment

The environment in which the host sample was build and checking operation is shown below.

- Hardware environment
  - Host
    - PC/AT™ compatible computer
    - Processor : At least 1.6GHz
    - Main Memory : At least 1GB
    - Display : 1024 x 768 or higher resolution and 65,536 colors
    - Interface : USB2.0
  - Device
    - Renesas Starter Kit for RL78/G14
    - Renesas BLE Evaluation Board for RL78/G1D
    - Smart Phone (Android device or iOS device)
- Tools
    - Renesas on-chip debugging emulator E1
- Software environment
    - Windows7 Service Pack1
    - e² studio V4.3.1.001 / RL78 Family C Compiler Package V1 V1.03.00
      or Renesas CS+ for CC V4.00.00 / RL78 Family C Compiler Package V1 V1.03.00
      or Renesas CS+ for CA, CX V3.02.00 / Renesas CA78K0R V1.72
    - Renesas Flash Programmer v3.02.00

2. Compositions

2.1 Device Composition

The device composition for checking operation the host sample is shown in Figure 2-1.

Local Device consists of RL78/G14 as a Host MCU and RL78/G1D as a BLE MCU which ware connected by UART. Remote Device is Smart Phone which is Android device or iOS device.

Local Device behaves as a Slave and Remote Device behaves as a Master. RL78/G14 executes BLE communication with Smart Phone by controlling BLE protocol stack on RL78/G1D via interactive UART communication.

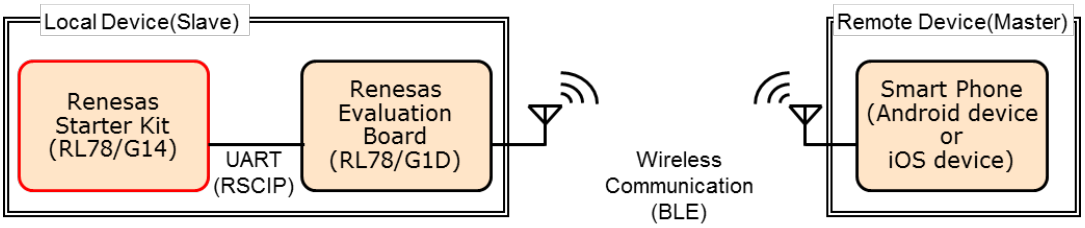

Figure 2-1 Device Composition

The overview of the host sample is shown below.

- ✓ It uses rBLE API and executes below operations.
  - ✧ After power up, it starts broadcasting and establishes a connection automatically.
  - ✧ After establishing a connection, it enables SCP (Sample Custom Profile).
  - ✧ It executes pairing and starting encryption if Remote Device requires.
  - ✧ It sends ADC result value every second if Remote Device permits the notification from Local Device.
- ✓ It is implemented a simple scheduler for management processing sequence.
- ✓ It changes the RL78/G14 state into the STOP mode if there is nothing to execute.
- ✓ It supposes that peer device is a smart phone (Android device or iOS device).

2.2 Software Composition

The software composition of RL78/G14 and RL78/G1D is shown in Figure 2-2.

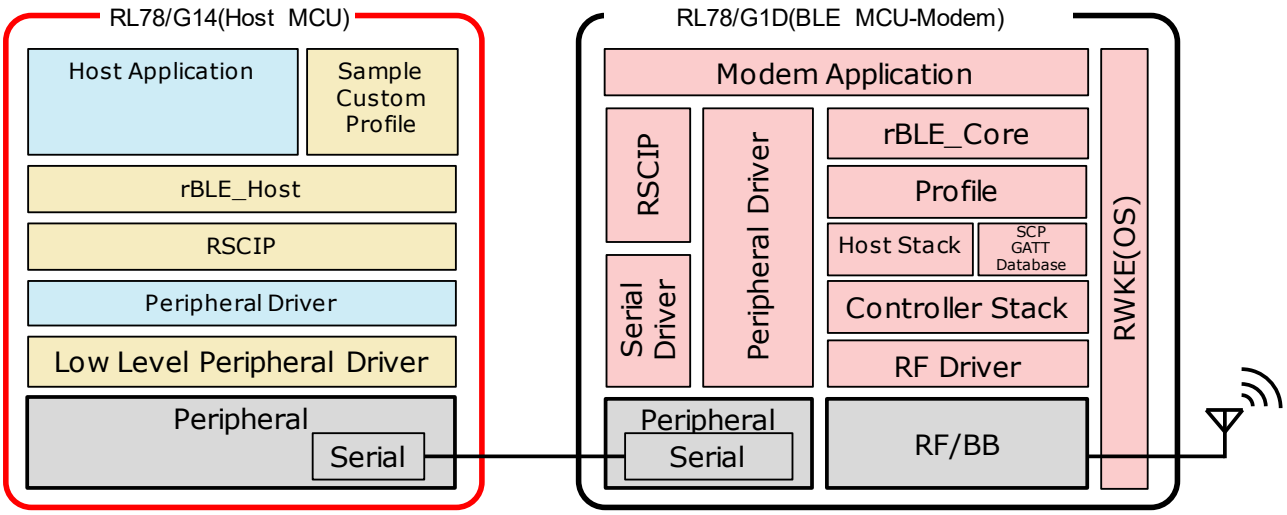

Figure 2-2 Software Composition

The software of Host MCU consists of Low Level Peripheral drivers and Peripheral drivers which controls MCU peripheral hardware, RSCIP (Renesas Serial Communication Interface Protocol), rBLE_Host which provides rBLE APIs, Host Application which controls the system, and Sample Custom Profile using the GATT API.

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

Table 2-1 Host MCU Software Composition

Software	Functions	When developing software
Host Application	Initializing rBLE Scheduling rBLE command execution Registering rBLE event callbacks	Need to be coded
Sample Custom Profile (SCP)	Custom Profile using GATT APIs	No need to be coded (provided by package) Note1
rBLE_Host	Providing rBLE APIs Executing rBLE event callbacks	No need to be coded (provided by package) Note1
RSCIP	Controlling serial communication	No need to be coded (provided by package) Note1
Peripheral Driver	Controlling Host MCU peripheral hardware	Need to be coded
Low Level Peripheral Driver	Controlling Host MCU peripheral hardware primitively	No need to be coded (generated by tool) Note2

Notes: 1. Code files for software development are provided by BLE protocol stack package.
2. Code files for software development are generated by the Code Generator.

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

The build environment and tools and source code and libraries are provided by BLE protocol stack.

Table 2-2 BLE MCU Software Composition

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

2.3 Peripheral Hardware Composition

The peripheral hardware composition of RL78/G14 is shown in Figure 2-3.

The host sample uses the RL78/G14 peripherals of Serial Array Unit, 12bit Interval Timer A/D Converter, and uses the RSK peripheral hardware of Variable Resistor, LCD, LED, and SW. And at least it needs Serial Array Unit and 12bit Interval Timer for using rBLE.

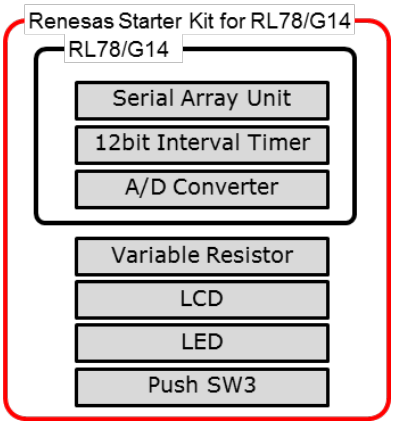

Figure 2-3 Peripheral Hardware Composition

Table 2-3 Peripheral Hardware Composition

Peripheral Hardware	Purpose	Necessity
Serial Array Unit	Communicating with BLE MCU(UART2 or UART0 which can SNOOZE)	Mandatory
12bit Interval Timer	Monitoring UART timeout and Triggering for Profile Notification	Mandatory
A/D Converter	Generating for Notification data	Optional
Variable Resistor(RV1)	Generating for Notification data (connected ADC)	Optional
LCD(LCD)	Displaying BLE command, BLE event and PassKey in pairing	Optional
LED(LED0)	Displaying MCU status (light on: RUN mode, light off: STOP mode)	Optional
SW(SW3)	Triggering for Disconnection	Optional

Note: Peripheral hardware which Host MCU required to use the rBLE is classified “Mandatory”, other peripheral hardware is classified “Optional”.

Notable settings of Code Generator tool in CS+ is shown in Table 2-4.

Table 2-4 Peripheral Hardware Configuration

Peripheral Hardware		Setting
Clock Generator └Clock	mode	High-speed main mode 2.7(V) ≦VDD≦ 5.5(V)
	Main system clock(fMAIN)	High-speed on-chip oscillator (fIH)
	Hi-speed on-chip oscillator	24(MHz)
	Sub system clock(fSUB)	used XT1 oscillation(fXT) 32.768(kHz) low power consumption oscillation permit to supply clock in STOP,HALT mode
	RTC, interval timer clock	32.768(kHz)
	CPU/peripheral hardware clock frequency(fCLK)	24000(kHz)
Interrupt functions └External Interrupt	INT10	Use INTP10 detect falling edge low priority
Serial array unit └SAU1 └Channel	Channel 0	UART2 transmitting/receiving
Serial array unit └SAU1 └UART2 └Receiving	Data bit length	8 bit
	Data Direction	LSB
	Parity	no parity
	Stop bit length	1 bit
	Data Phase	standard
	Baudrate	4800(bps)
	Interrupt	reception transfer end interruption(INTSR2) high priority
	Callback function	Received Error
Serial array unit └SAU1 └UART2 └Transmitting	Transfer mode	single mode
	Data bit length	8 bit
	Data direction	LSB
	Parity	no parity
	Stop bit length	1 bit
	Data phase	standard
	Baudrate	4800(bps)
	Interrupt	transmission end interruption(INTST2) low priority
	Callback function	Transmitted
A/D converter	A/D converter	used
	comparator	unused
	Resolution	8 bit
	Trigger mode	software trigger mode
	Conversion mode	one shot select mode
	Interrupt	conversion end interruption(INTAD) low priority
12-bit interval timer	Interval timer	used
	Interval period	10(ms)
	Interrupt	interval period expiration interruption(INTIT) low priority

2.4 File Composition

The file composition of the host sample is shown below.

(R)-mark which is set on the head of file name is shown that file is included in the BLE protocol stack package. When developing software, it is necessary to use the latest code which is provided by BLE protocol stack package.

HostSampleRL78G14		
├──Platform		
│ ├──driver		
│ │ ├──plf		
│ │ │ plf.c		platform driver code file
│ │ │ plf.h		platform driver header file
│ │ ├──serial		
│ │ │ uart.c		uart driver code file
│ │ │ uart.h		uart driver header file
│ │ ├──timer		
│ │ │ timer.c		timer driver code file
│ │ │ timer.h		timer driver header file
│ │ └──lcd		
│ │ lcd.c		LCD driver code file
│ │ lcd.h		LCD driver header file
│ └──include		
│ arch.h	(R)	architecture header file
│ compiler.h	(R)	compiler header file
│ ll.h	(R)	low level macro header file
│ types.h	(R)	type definition header file
│ rscip_api.h	(R)	RSCIP callback header file
│ rskrl78g14def.h		RSK header file
│ iodefine.h		Macro definition for register access header file
├──rBLE		
│ ├──sample_app		
│ │ app.c		host application code file
│ ├──sample_profile		
│ │ └──scp		
│ │ scps.c	(R)	SCP Server API code file
│ ├──host		
│ │ │ rble_host.c	(R)	rBLE_Host code file
│ │ │ rble_if_api_cb.c	(R)	rBLE API callback code file
│ │ ├──gap		
│ │ │ rble_api_gap.c	(R)	GAP API code file
│ │ ├──gatt		
│ │ │ rble_api_gatt.c	(R)	GATT API code file
│ │ ├──sm		
│ │ │ rble_api_sm.c	(R)	SM API code file
│ │ └──vs		
│ │ rble_api_vs.c	(R)	VS API code file
│ ├──rscip		
│ │ rscip.c	(R)	RSCIP code file
│ │ rscip.h	(R)	RSCIP header file
│ │ rscip_cntl.c	(R)	RSCIP control code file
│ │ rscip_cntl.h	(R)	RSCIP control header file
│ │ rscip_ext.h	(R)	RSCIP external callback header file
│ │ rscip_uart.c	(R)	RSCIP serial communication code file
│ │ rscip_uart.h	(R)	RSCIP serial communication header file
│ └──include		
│ │ db_handle.h	(R)	database handle header file
│ │ prf_sel.h	(R)	profile select header file
│ │ rble.h	(R)	rBLE macro header file
│ │ rble_api.h	(R)	rBLE API header file
│ │ rble_api_custom.h	(R)	rBLE SCP API header file
│ │ rble_trans.h	(R)	rBLE communication header file

```
│       │   rble_app.h                    (R)    host application header file
│       └──host
│               rble_host.h                (R)    rBLE_Host header file
└──project
        CS_CCRL                                   Project for CS+ for CC
        CubeSuite                                 Project for CS+ for CA,CX
        e2studio                                  Project for e²studio
```

Each project directory include following low level peripheral drivers. Though the low level drivers for each project have different implementation such as interrupt handler declaration, the low level peripheral drivers have same behavior.

3. Procedure

3.1 Preparation

3.1.1 Host MCU

The procedure about preparing Host MCU of RL78/G14 is shown below. When changing the UART connection method, refer to 3.1.5.

Following shows the procedure to build the firmware for Host MCU. The procedure is different depends on the development environment.

(1) e²studio

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

(2) CS+

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

Table 3-1 Host MCU Project Directory and Firmware Path

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

Next shows the procedure to boot the firmware on Host MCU.

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

Table 3-2 Jumper Setting

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

3.1.2 BLE MCU

The procedure about preparing BLE MCU of RL78/G1D is shown below. When changing the UART connection method, refer to 3.1.5.

Note: The source of supplying power is selectable from AC power adapter or USB port.

Note: It is possible to use "RL78_G1D_CM(SCP).hex" firmware included in the package of BLE protocol stack. If use this firmware, start below procedure from the step 8. But pairing sequence with iOS device is not executed, because this firmware is made without the step2.

First prepare the source code.

1. Download the EEPROM Emulation Library and Code Flash Library from the Renesas web site, and copy the library to following directories. The copy directory is different depends on the compiler.
   - EEPROM Emulation Library
     - CC-RL

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

       RL78_G1D\Project_Source\renesas\src\driver\dataflash\cs
   - Code Flash Library
     - CC-RL

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

       RL78_G1D\Project_Source\renesas\src\driver\codeflash\cs
2. To execute pairing with Remote Device, perform the following procedures. But this alteration is not essential. Open "prf_config.c" file and find "Sample Custom Notify Cfg Value" words. In near the location where found words, change the Attribute Permission configuration from (RBLE_GATT_PERM_RD | RBLE_GATT_PERM_WR) to (RBLE_GATT_PERM_RD | RBLE_GATT_PERM_WR_UNAUTH). This alteration changes Notification configuration of Sample Custom Profile to Write Permission (Unauthentication Required). This configuration will be needed for secure communication with Remote Device.

Then build the source code. The procedure is different depends on the development environment.

(1) e²studio

1. Launch e² studio.
2. Right click on "Project Explorer" and select "Import…" from the dropdown menu.
3. "Import" window is popped up and select "Existing Projects into Workspace" and click "Next >".
4. Fill "Select root directory:" form with the project directory shown in Table 3-3.
5. Make sure that the project you selected is displayed in "Projects:" and click "Finish". Then the windows is closed.
6. Right click on the project just imported on "Project Explorer" and select "Renesas Tool Settings" → "Tool Settings" → "Compiler" → "Source" → "Macro definition" and add "USE_SAMPLE_PROFILE" macro (If "noUSE_SAMPLE_PROFILE" macro is defined, delete "no" word.).
7. Right click on the project on "Project Explorer" and select "Build Project" from the dropdown menu.
8. Refer Table 3-3 for the Firmware file generate path.

(2) CS+

1. Double click the project file shown in Table 3-3.
2. Select [Property] in the right clicked menu of Build Tool in “rBLE_emb” subproject in the project tree.
3. Define “USE_SAMPLE_PROFILE” macro in the location of [Preprocess] → [Defined Macros] in the Compile Option of the dialog displayed. (If “noUSE_SAMPLE_PROFILE” macro is defined, delete “no” word.)
4. Right click on “BLE_Emb” in “Project Tree” and select “Build BLE_Emb” from the dropdown menu.
5. Refer Table 3-3 for the Firmware file generate path.

Table 3-3 BLE MCU Project Directory and Firmware Path

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

Finally write the firmware onto BLE MCU.

1. Connect E1 emulator to BLE Evaluation Board and connect E1 emulator to PC.
2. Connect the power source to BLE Evaluation Board and supply power to BLE Evaluation Board.
3. Start RFP (Renesas Flash Programmer) and create workspace by selecting [File] → [Create New Project]. And then select [RL78] as [Microcontroller] and push [Connect] button.
4. Uncheck Erase and P.V checkbox on the [Block Setting] tab → [Code Flash 1] → [Block255] and all blocks of [Data Flash 1].
5. Select the generated firmware by the menu [User/Data Area] and select [Write after Delete] command.
6. Push [Start] button on [Operation] tab to start writing and confirm that [SUCCESS] is displayed.
7. Remove the power source and E1 emulator from BLE Evaluation Board.

Table 3-4 Switch Setting

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

3.1.3 Host MCU - BLE MCU Connection

The procedure about wired connection between Host MCU and BLE MCU is shown below.

Note: About connection of UART 2-wire with branch connection, refer to "4.3.3 Example of Application Circuit".

1. Refer to Table 3-5and connect pins RSK board and BLE Evaluation Board by wires.
2. Start supplying power to BLE Evaluation Board.
3. Start supplying power to RSK board.

Table 3-5 pin connection

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

3.1.4 Smart Phone

The procedure about preparation Smart Phone is shown below.

Install below application either Android device or iOS device which is used as Remote Device.

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

3.1.5 UART Connection Method Setting

It shows the source files change point for setting the UART connection method.

(1) Host MCU

UART connection method of the host sample is selected by the following macro of uart.h.

Table 3-6 Setting of uart.h

Macro	Description
SERIAL_U_DIV_2WIRE	0 : UART 2-wire connection <default setting> 1 : UART 2-wire with branch connection

(2) BLE MCU

BLE UART connection method of the MCU firmware will be selected in the following macro of serial.h and wakeup.c, which is included in the BLE protocol stack.

Table 3-7 Setting of serial.h

Macro	Description
SERIAL_U_2WIRE (1) SERIAL_U_DIV_2WIRE (0)	UART 2-wire connection : <default setting> Set (1) to SERIAL_U_2WIRE. Set (0) to other macros. UART 2-wire with branch connection : Set (1) to SERIAL_U_DIV_2WIRE. Set (0) to other macros.

Table 3-8 Setting of wakeup.c

Macro	Description
USE_WAKEUP_SIGNAL_PORT (0) /* Modem Setting */	0 : UART 2-wire connection <default setting> 1 : UART 2-wire with branch connection

3.2 Verification

3.2.1 Android Device

The procedure about verifying host sample with Android device is shown below.

1. Start “GATTBrowser” application.
2. Select a device named “Renesas-BLE” from the nearby devices list to connect with the device. (Figure A1 Arrow-1)
3. Select the Characteristic the UUID is 02000000-0000-0000-0000-000000000080. (Figure A2 Arrow-1)
4. Select [Notification Off]. (Figure A3 Arrow-1)
5. Enter passkey (six digits) to the passkey dialog. Note1 Note2 (Figure A4 Arrow-1). When USE_PAIRING_JUSTWORKS macro is set, the passkey input is not requested. Though GUI indication of BLE Scanner looks like Notification has been enabled, Notification is still disabled. To enable Notification actually, re-selects N button (N button’s color turns to blue) and once again selects N button (N button’s color turns to green).
6. If the passkey entered is correct, notification is started. Bit16:31 of the notification data is incremented in every second. (Figure A5 Arrow-1)
7. Bit0:7 of the notification data is changed depending on the potentiometer’s position on the RSK. (Figure A5 Arrow-1)
8. Select [Notification On]. (Figure A5 Arrow-2)
9. Select [←] to disconnect the connection. (Figure A5 Arrow-3) (Figure A6 Arrow-1)
10. Repeat 2~4 and make sure that notification is re-started without passkey request. Note3
11. Repeat 8~9 to disconnect the connection.

Notes: 1. Sometimes the passkey dialog is not displayed on the foreground. That time you will see the message “Pairing request” in the information pane and click it to open the dialog.
2. To run the pairing again, select [Settings]→[Bluetooth]. In the [Paired devices] list, you will see the device named “Renesas-BLE” and select the device →[FORGET] to delete pairing information.
3. Android 6.0/6.0.1 always fail to connect with bonded devices. To reconnect with bonded devices, use Android 5 series, or follow [Note2] to delete pairing information and re-connect. https://code.google.com/p/android/issues/detail?id=202850&can=1&q=pairing%20bonded&colspec=ID%20Status%20Priority%20Owner%20Summary%20Stars%20Reporter%20Opened

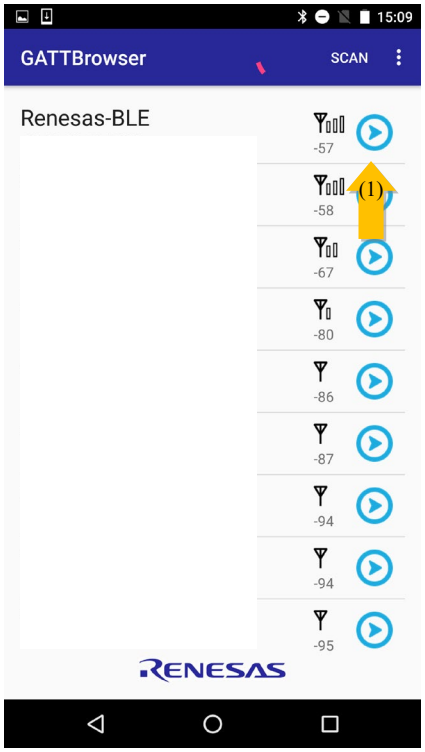

Figure A1

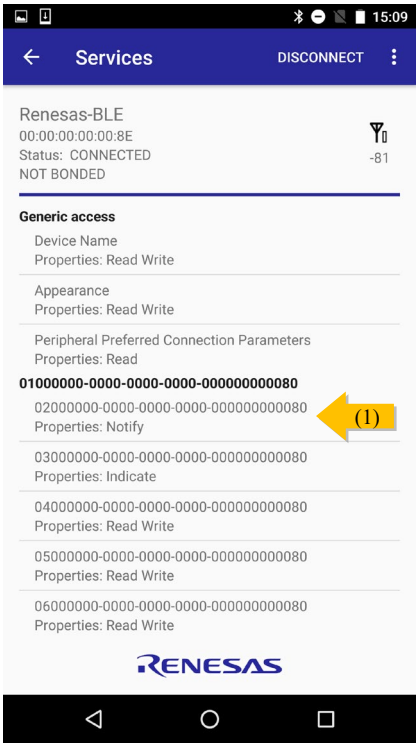

Figure A2

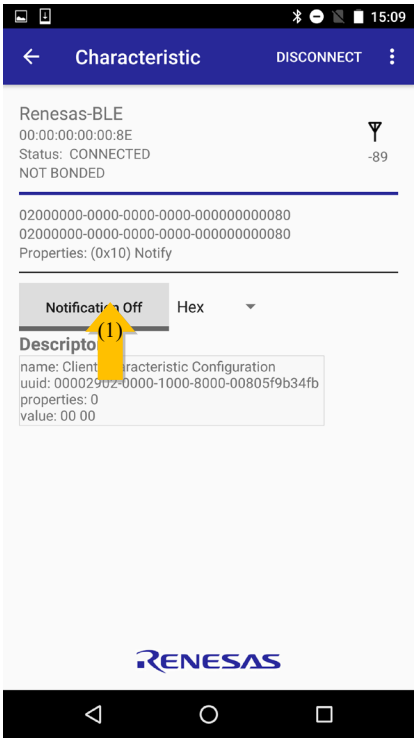

Figure A3

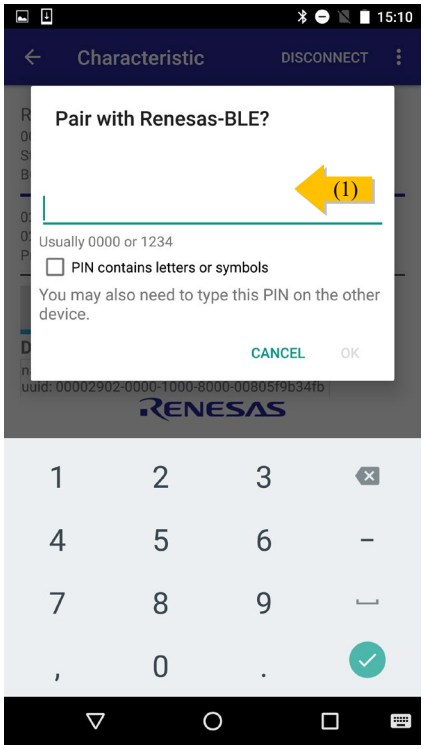

Figure A4

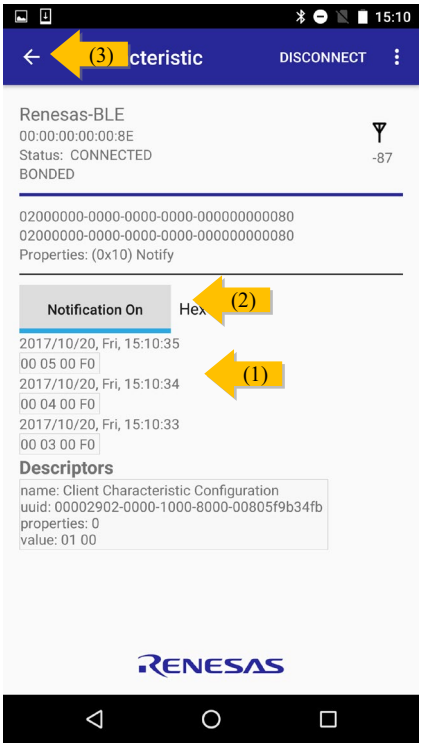

Figure A5

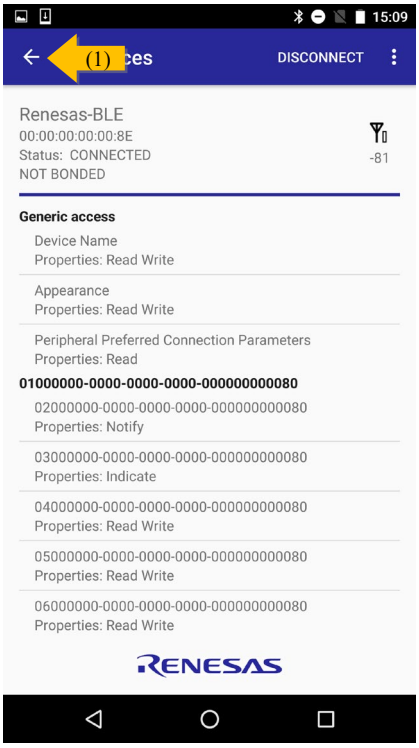

Figure A6

3.2.2 iOS Device

The procedure about verifying host sample with iOS device is shown below.

1. Start “GATTBrowser” application.
2. Select a device named “Renesas-BLE” from the nearby devices list to connect with the device. (Figure i1 Arrow-1)
3. Select the characteristic [UUID: 0x02000000-0000-0000-0000-000000000080]. (Figure i2 Arrow-1)
4. Select [Enable Notification] to enable notification. (Figure i3 Arrow-1)
5. Enter passkey (six digits) to the dialog. Note1 (Figure i4 Arrow-1). When USE_JUSTWORKS_PAIRING macro is set, the dialog showing the receipt of pairing request is pop up. Select “Pair” to accept the request.
6. If the passkey entered is correct, notification is started. Bit16:31 of the notification data is incremented in every second. Note2 (Figure i5 Arrow-1)
7. Bit0:7 of the notification data is changed depending on the potentiometer’s position on the RSK. (Figure i5-1)
8. Select [Disable Notification] to disable notification. (Figure i5 Arrow-2)
9. Select [< Services] → [< Back] to disconnect the connection. (Figure i5 Arrow-3) (Figure i6 Arrow-1)
10. Re-do 2~4 to make sure that notification is re-started without passkey request.
11. Re-do 8~9 to disconnect the connection.

Notes: 1. To re-run the pairing, select [Settings] → [General] → [Bluetooth] → [Device] and select [Forget this Device] to delete pairing information.

Notes: 2. Sometimes notification is not started automatically after entering passkey. In such case, select [Enable Notification] once again.

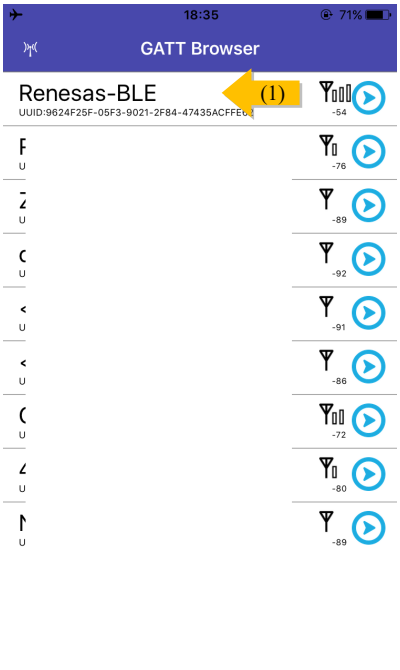

Figure i1

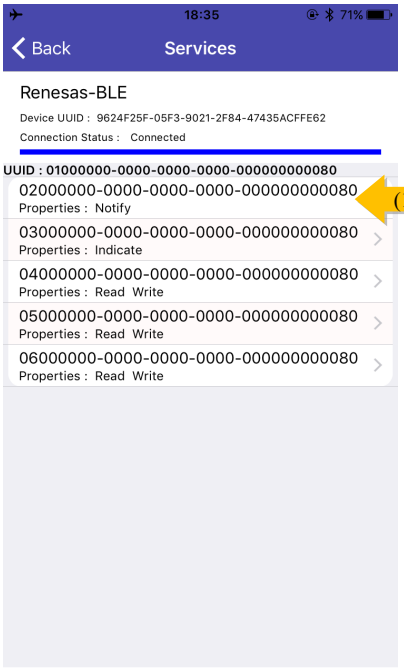

Figure i2

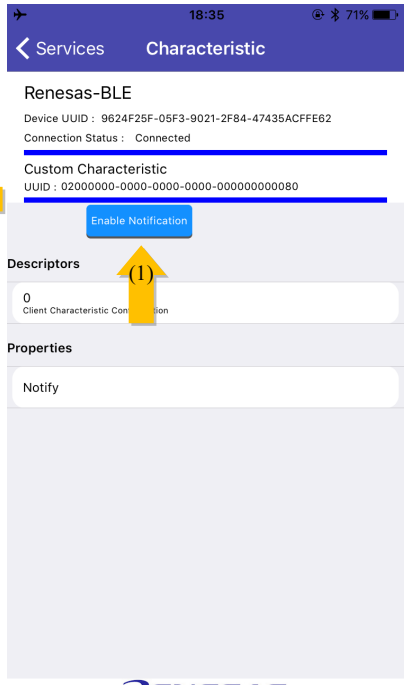

Figure i3

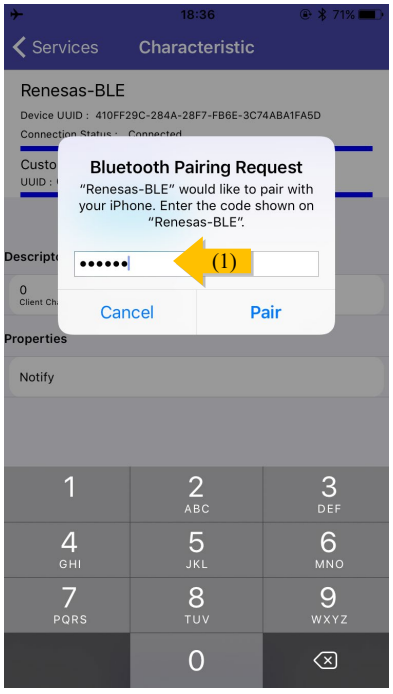

Figure i4

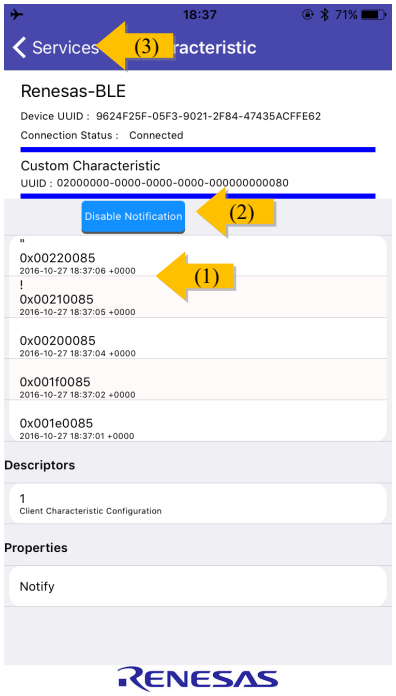

Figure i5

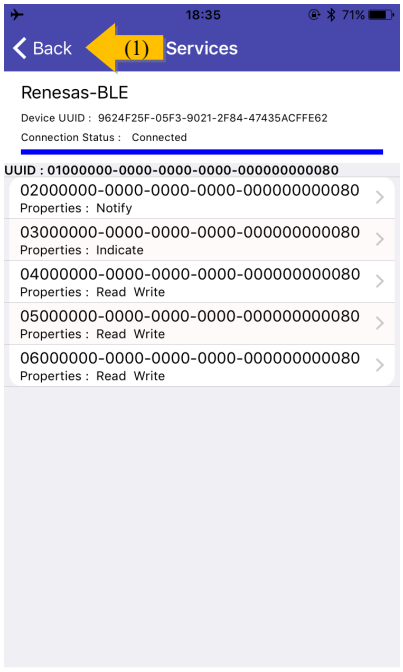

Figure i6

3.3 Configuration

3.3.1 Configuration Macro Settings

The procedure about changing the behavior of host sample is shown below.

The behavior of host sample is able to be changed by changing definition the behavior configuration macros in compile option. The procedure of changing macros is selecting [Property] in the right clicked menu of "HostSample" subproject in the project tree and defined or undefined macros in [preprocess]→[definition macros] on compile option tab in the dialog displayed.

Table 3-9 Behavior Configuration Macros

Macro	when defined	when undefined
USE_PAIRING_JUSTWORKS	execute pairing with JustWorks method	<default setting> execute pairing with PassKeyEntry method
USE_RSK_LCD	<default setting> use LCD on RSK	no use LCD on RSK Note: select JustWorks method in pairing because impossible to display PassKey
USE_RSK_LED	<default setting> use LED on RSK	no use LED on RSK
USE_RSK_SW	<default setting> use SW on RSK	no use SW on RSK Note: impossible to request disconnection from host sample when this setting
USE_RSK_ADC	<default setting> use A/D converter on RSK	no use A/D converter on RSK Note: set A/D converter configuration on Code Generator to [no use A/D converter] when this setting

3.3.2 Low Level Peripheral Driver Update By Code Generation Tool

You can update the low level peripheral drivers by using Code Generation Tool. Following file is updated in this project to avoid the type declaration confliction. You shall not overwrite the file when you use Code Generation Tool.

- r_cg_macrodriver.h

4. Behavior

The software behavior of Host Application (hereafter called APP) and rBLE is shown in this chapter.

4.1 Command and Event

The behavior of command and event which is used by APP and rBLE is shown in Figure 4-1.

1. APP issues the command by calling rBLE API.
2. rBLE executes the command issued by APP.
3. After finishing executing the command, rBLE informs the event by calling the callback function registered by APP.
4. APP decides whether to issue the next command or not in the callback function. (And go back to step1.)

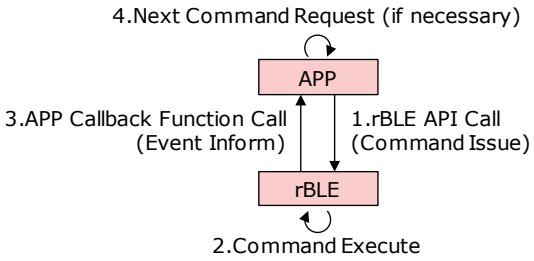

Figure 4-1 Command and Event

4.2 Main Loop

The main loop behavior of the host sample is shown in Figure 4-2.

The main loop of the host sample executes the APP Scheduler, rBLE Scheduler and the MCU Mode Manager repeatedly. The APP Scheduler issues the command. The rBLE Scheduler calls the callback function. The MCU Mode Manager changes the MCU mode to STOP state for reducing power consumption.

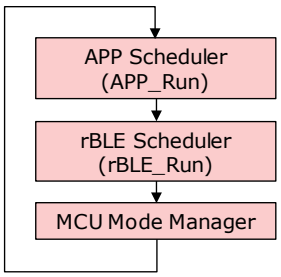

Figure 4-2 Main Loop

The APP Scheduler has the command request queue. If there is a command request in the queue, the scheduler calls rBLE API.

The rBLE Scheduler has the event queue. If there is an event in the queue, the scheduler calls callback function which is registered by APP.

The MCU Mode Manager checks if the command request queue and the event queue is set. If both of the queue is empty, the manager changes the MCU mode to STOP state. After changing to STOP state, MCU is awoke by occurring interruption.

4.3 UART 2-wire with Branch Connection

This section describes the UART driver communication method in the UART 2-wire branch connection.

4.3.1 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 4-1 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 4-3)

(b) Host MCU is receiving RSCIP packet from the module (Figure 4-4)

(c) ACK byte reception time-out (Figure 4-5)

(a) Host 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.

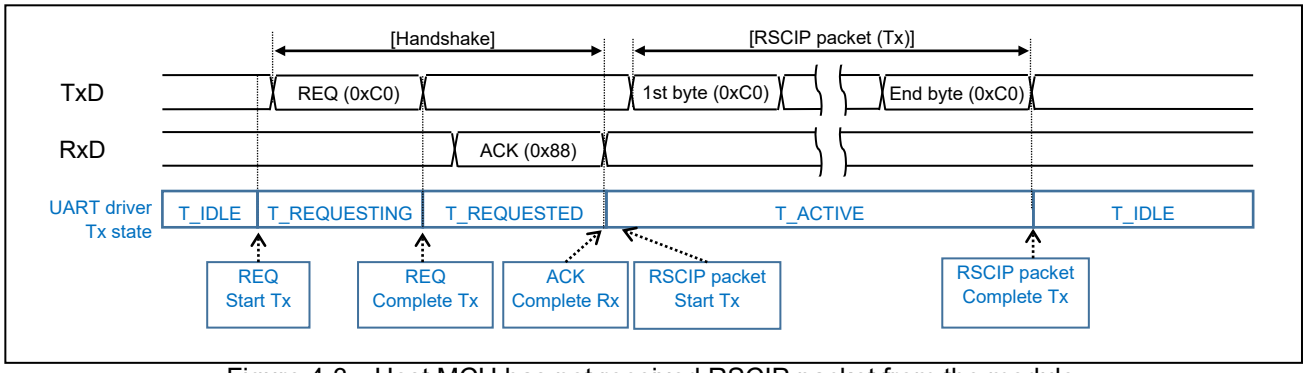

Figure 4-3 Host MCU has not received RSCIP packet from the module

(b) Host MCU is receiving RSCIP packet form 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.

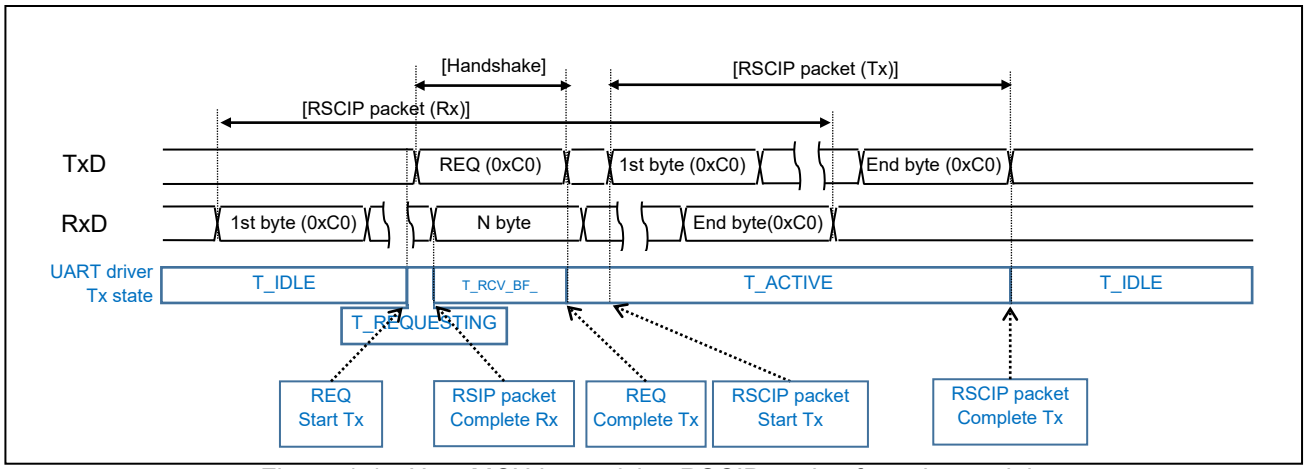

Figure 4-4 Host MCU is receiving RSCIP packet from the module

(c) ACK byte reception 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.

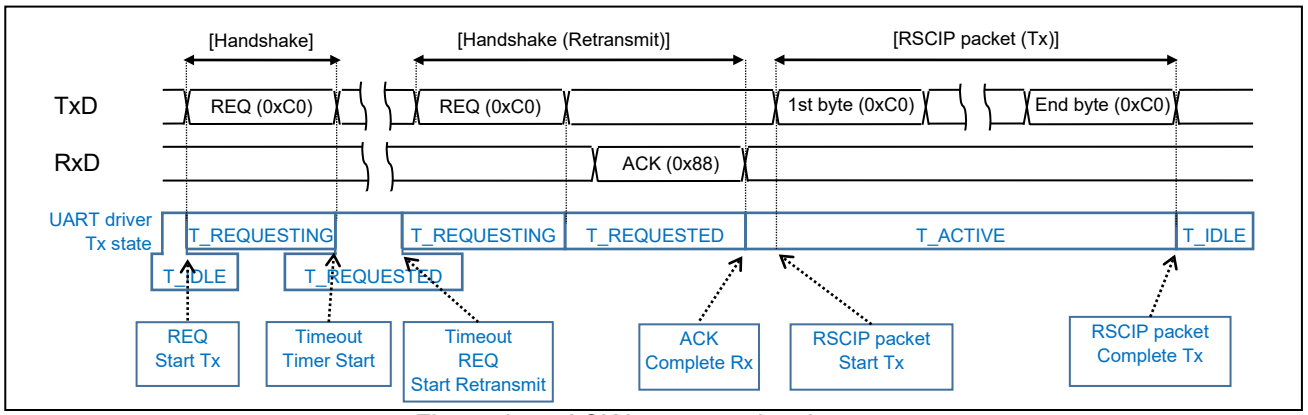

Figure 4-5 ACK byte reception time-out

4.3.2 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.

4.3.3 Example of Application Circuit

UART 2-wire with branch connection example of Host MCU and BLE MCU are shown below.

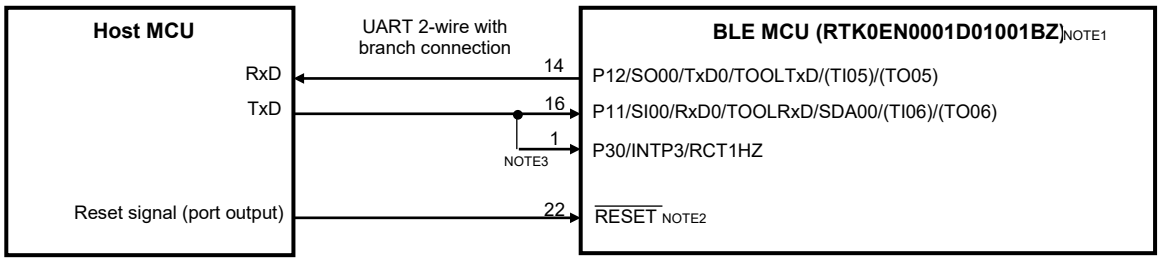

Figure 4-6 UART 2-wire with branch connection example

Notes: 1. Pin number is CN4 external extension interface connector pin number of RL78/G1D evaluation board.
2. /RESET pin are pulled-up/pulled-down with a resistor in accordance with the system requirement (see RL78/G1D User's Manual: Hardware) (R01UH0515).
3. VBUS detection of USB is assigned to P30/INTP3/RCT1HZ (WAKEUP) of the RL78/G1D evaluation board. If it supplies a power to an evaluation board from USB, do not connect TXD line of RSK which diverged to INTP3 of the RL78/G1D evaluation board.

5. Sequence chart

The sequence chart of Local Device and Remote Device is shown in this chapter. Each sequence is consisted of the devices which contains Host MCU, BLE MCU, Smart Phone and the software which contains APP and rBLE.

5.1 Main sequence chart

In the Main Sequence Chart, the processing blocks of 10 steps are shown. The detail of each processing block is shown in following sections.

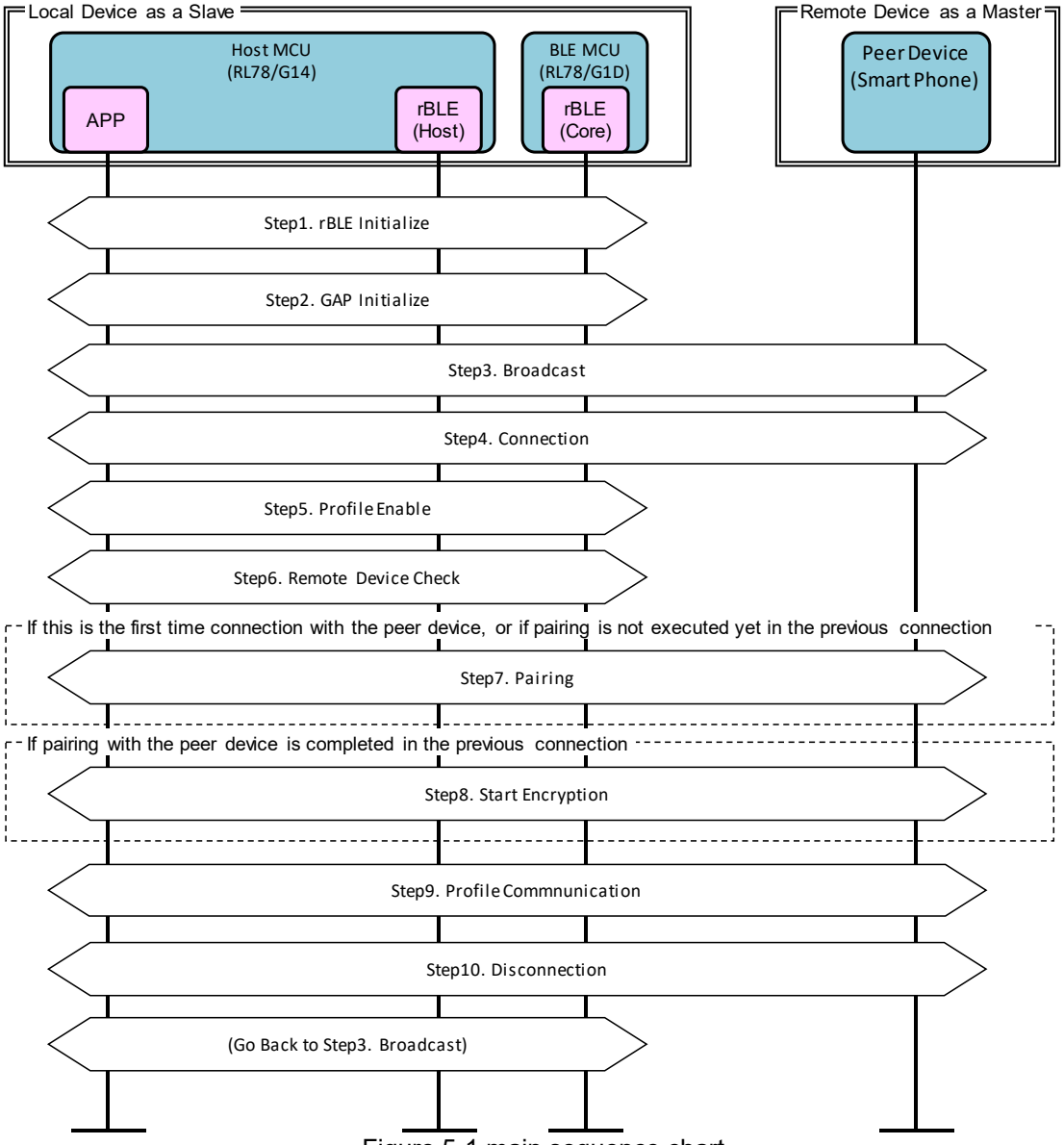

Figure 5-1 main sequence chart

5.2 Step1. rBLE Initialize sequence

APP calls “RBLE_Init” function to initialize rBLE (rBLE_Host and rBLE_Core). After initializing rBLE and establishing communication to BLE MCU, rBLE informs “RBLE_MODE_ACTIVE” event.

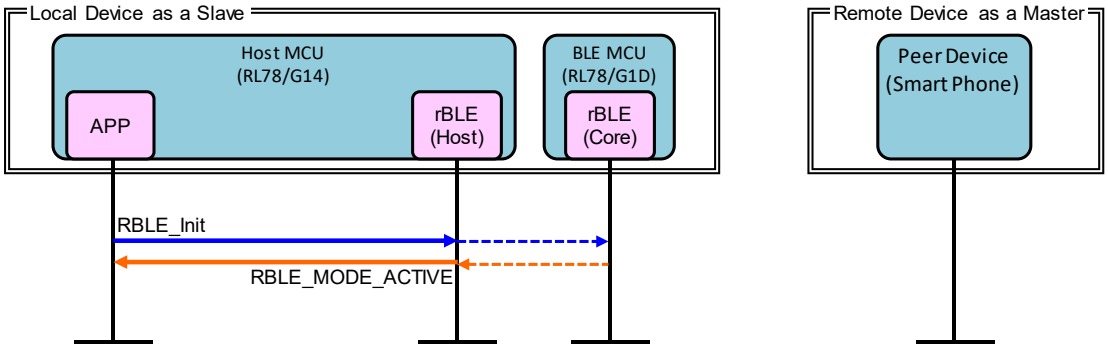

Figure 5-2 rBLE Initialize sequence chart

5.3 Step2. GAP Initialize sequence

APP calls “RBLE_GAP_Reset” function to reset GAP. After resetting rBLE, rBLE informs “RBLE_GAP_EVENT_RESET_RESULT” event.

APP calls “RBLE_GAP_Set_Bonding_Mode” function to permit the bonding with remote device. After setting the permission, rBLE informs “RBLE_GAP_EVENT_SET_BONDING_MODE_COMP” event.

APP calls “RBLE_GAP_Set_Security_Request” function to set security level. After setting security level, rBLE informs “RBLE_GAP_EVENT_SET_SECURITY_REQUESET_COMP” event.

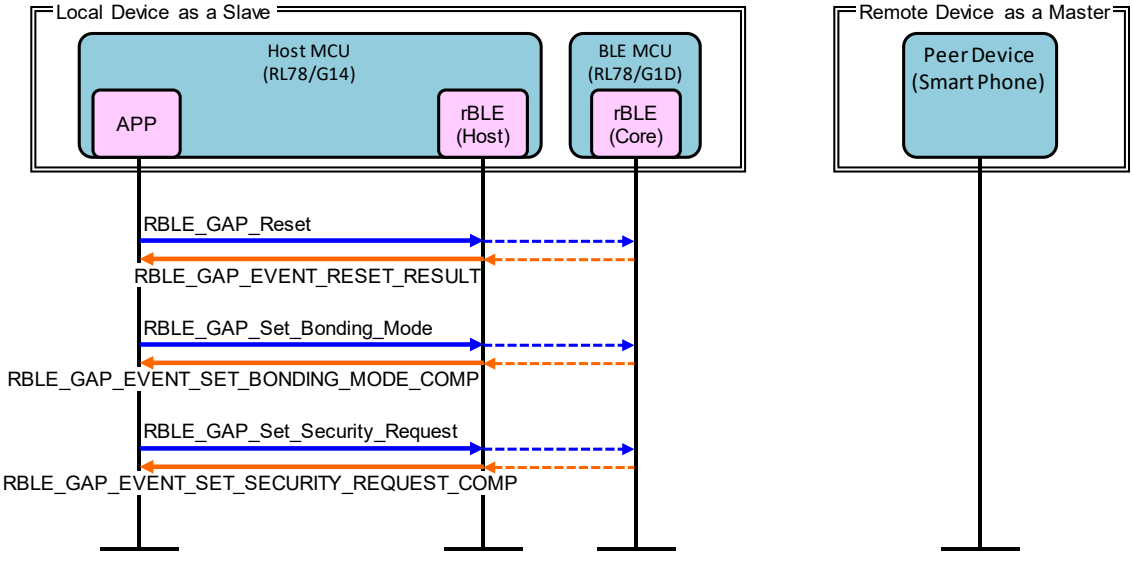

Figure 5-3 GAP Initialize sequence chart

5.4 Step3. Broadcast sequence

Local Device starts broadcasting to establish connection as a slave.

APP calls “RBLE_GAP_Broacast_Enable” function to start broadcasting. After starting the broadcast, rBLE informs “RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP” event.

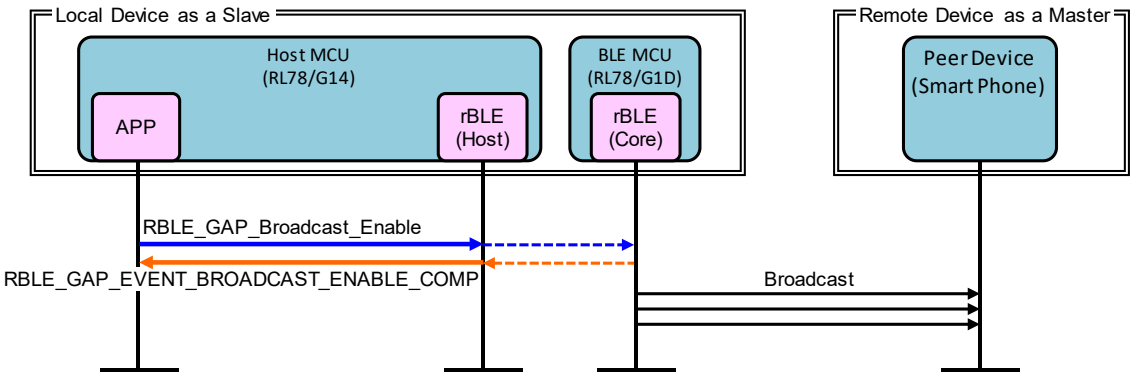

Figure 5-4 Broadcast sequence chart

5.5 Step4. Connection sequence

Remote Device receives the broadcast and requests to establish connection with Local Device.

If the connection between Remote Device and Local Device is established by receiving Connection Request from Remote Device, rBLE informs “RBLE_GAP_EVENT_CONNECTION_COMP” event.

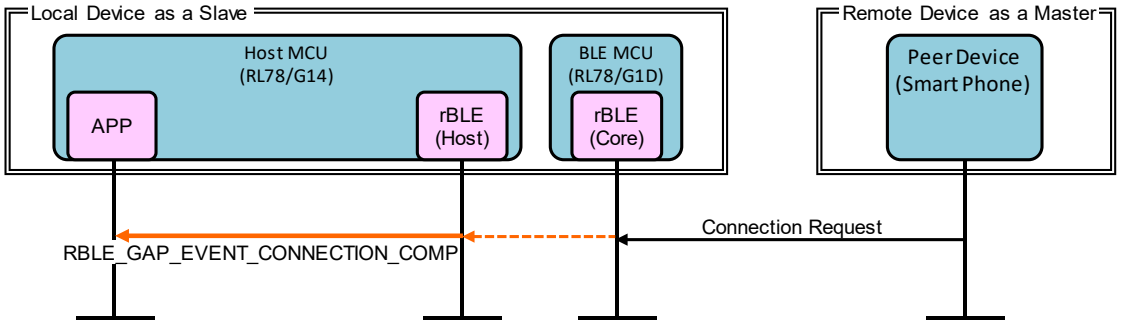

Figure 5-5 Connection sequence chart

5.6 Step5. Profile Enable sequence

Local Device enables SCP (Sample Custom Profile) to send data.

APP calls “RBLE_SCP_Server_Enable” function to enable SCP. After enabling SCP, rBLE informs “RBLE_SCP_EVENT_SERVER_ENABLE_COMP” event.

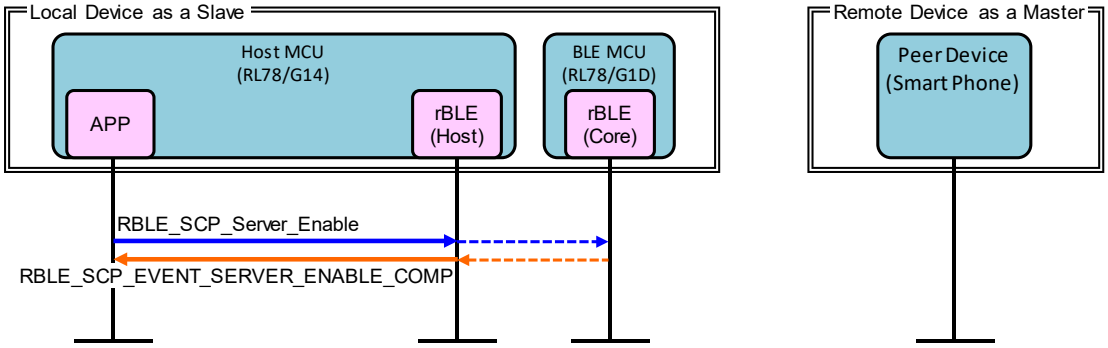

Figure 5-6 Profile Enable sequence chart

5.7 Step6. Remote Device Check sequence

Local Device confirms security information about Remote Device.

If the device address of Remote Device is public address or if it is random address except resolvable private address, rBLE informs “RBLE_SM_CHK_BD_ADDR_REQ” event to acquire security information about Remote Device. APP calls “RBLE_SM_Chk_Bd_Addr_Req_Resp” function to inform security information.

If the device address of Remote Device is resolvable private address, rBLE informs “RBLE_SM_IRK_REQ_IND” event to acquire IRK (Identify Resolving Key) which is used for resolving address. APP calls “RBLE_SM_Irk_Req_Resp” function to informs whether to have IRK or not and informs IRK. If resolving address is success, rBLE informs “RBLE_GAP_EVENT_RPA_RESOLVED” event. If resolving address is failed, rBLE informs “RBLE_SM_IRK_REQ_IND” event repeatedly until it is successful or until all of IRK which APP possess is checked.

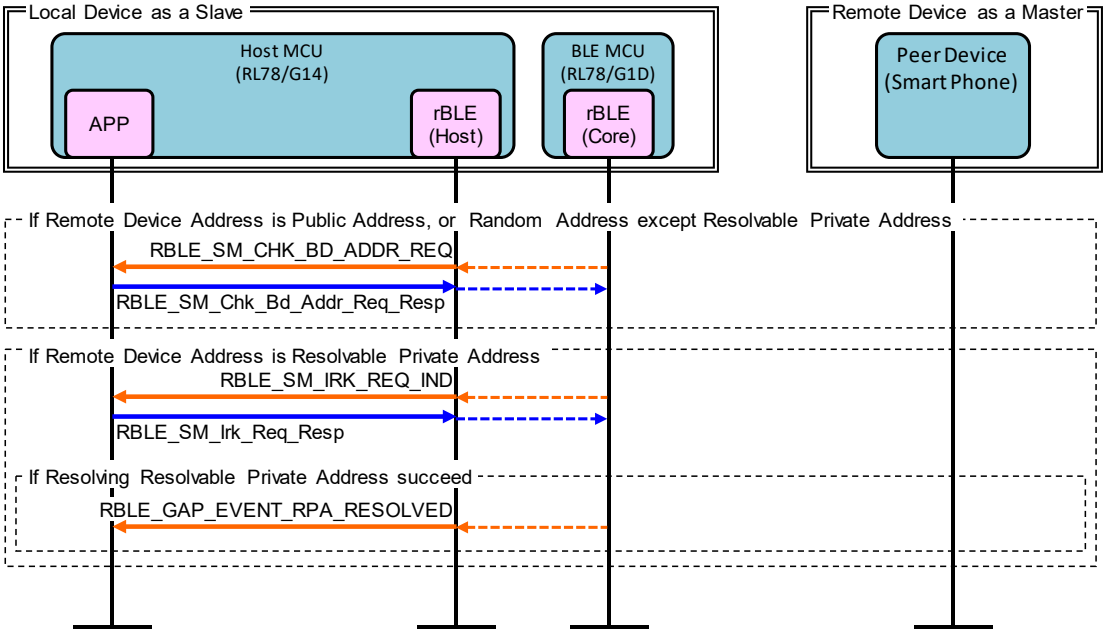

Figure 5-7 Remote Device Check sequence chart

5.8 Step7. Pairing sequence

If the connection with the Remote Device is first time or if pairing is not executed in previous connection, Local Device starts pairing sequence by request from Remote Device. Pairing sequence is consisted of PHASE1, PHASE2, starting encryption and PHASE3.

PHASE1 is for exchanging the pairing features between Local Device and Remote Device.

If Local Device receives Pairing Request from Remote Device, rBLE informs “RBLE_GAP_EVENT_BONDING_

REQ_IND” event. APP calls “RBLE_GAP_Bonding_Response” function to send Pairing Response.

PHASE2 is for generating STK (Short Term Key).

rBLE informs “RBLE_SM_TK_REQ_IND” event to acquire TK (Temporary Key). APP calls “RBLE_SM_Tk_Req_Resp” function to inform TK. After generating STK by BLE_MCU, Local Device and Remote Device start encrypting the contents of communication.

PHASE3 is for distributing encryption keys of Local Device and Remote Device.

rBLE informs “RBLE_SM_LTK_REQ_IND” event to acquire LTK (Long Term Key). APP calls “RBLE_SM_Ltk_Req_Resp” function to inform LTK and send Encryption Information (LTK).

By receiving Encryption Information (LTK) from Remote Device, rBLE informs “RBLE_SM_KEY_IND” event.

By receiving Identity Information (IRK) from Remote Device, rBLE informs “RBLE_SM_KEY_IND” event.

If pairing sequence is success, rBLE informs “RBLE_GAP_EVENT_BONDING_COMP” event.

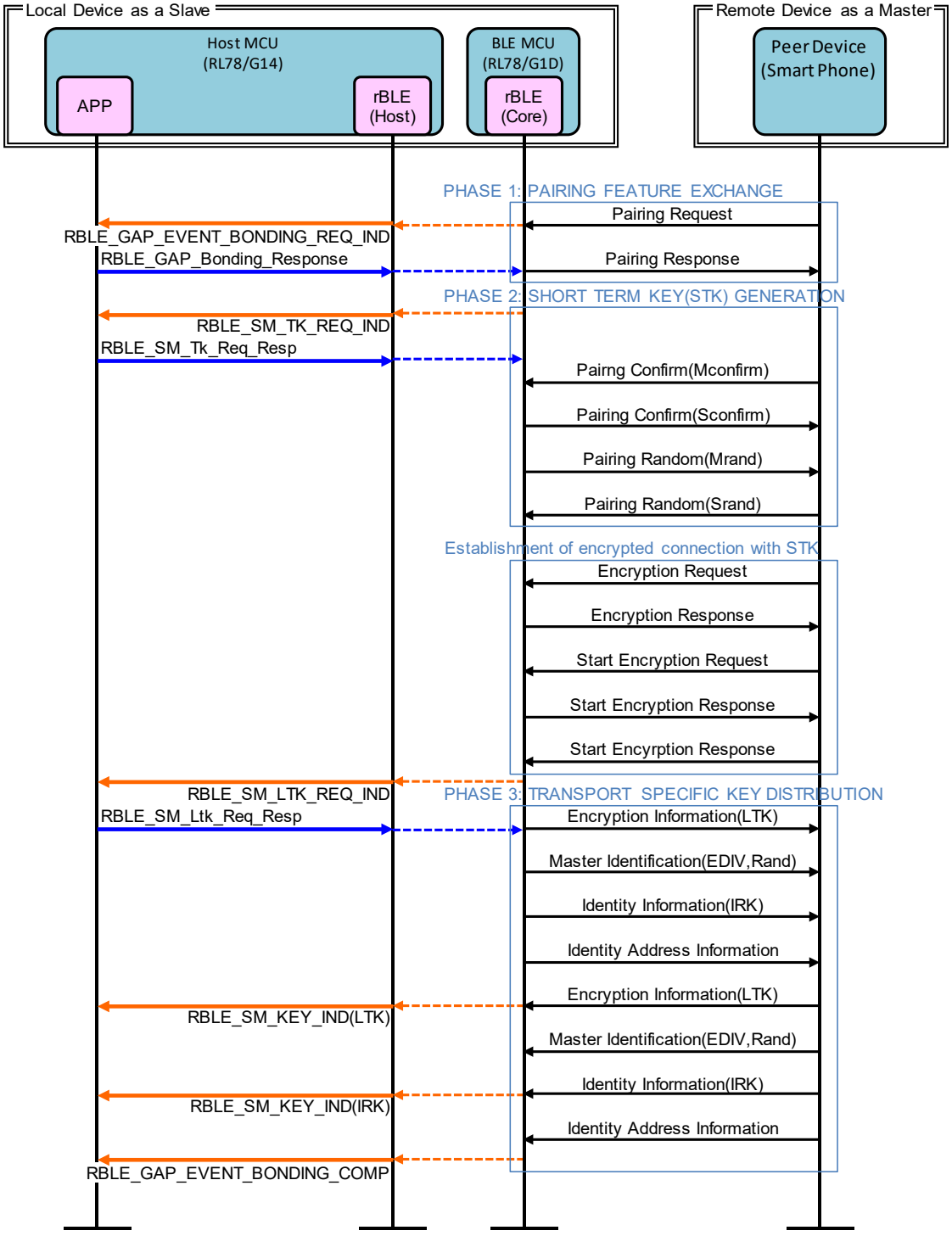

Figure 5-8 Pairing sequence chart

5.9 Step8. Start Encryption sequence

If pairing is success in previous connection, Local Device starts encryption sequence with LTK (Long Term Key) by request from Remote Device.

By receiving Encryption Request from Remote Device, rBLE informs “RBLE_SM_LTK_REQ_FOR_ENC_IND” event. APP calls “RBLE_SM_Ltk_Req_Resp” function to inform LTK and send Encryption Response.

By receiving Start Encryption Request, BLE MCU of Local Device sends Start Encryption Response.

If start encryption sequence is success, rBLE informs “RBLE_SM_ENC_START_IND” event.

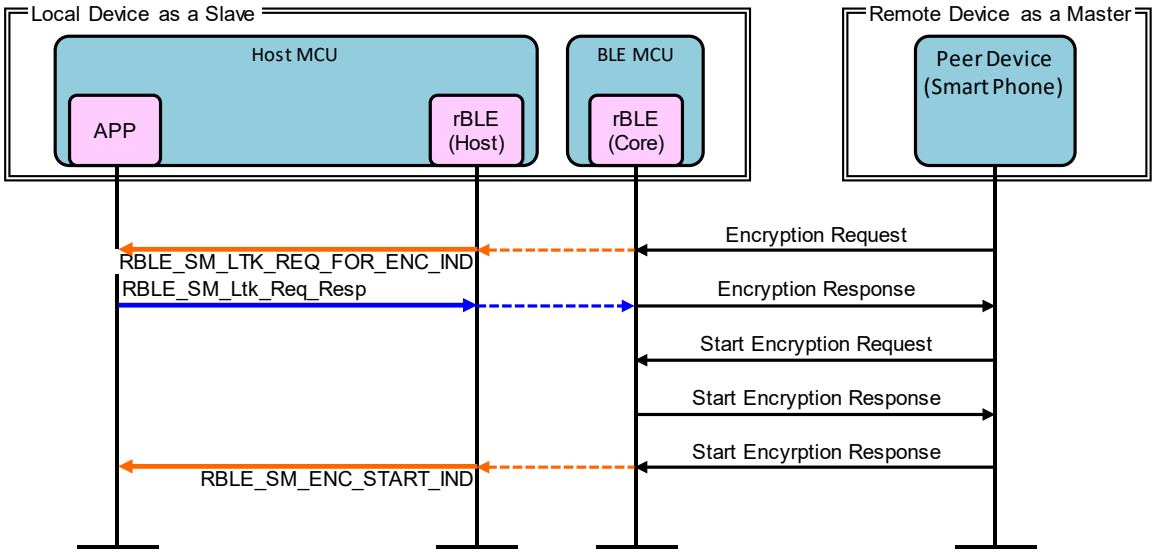

Figure 5-9 Start Encryption sequence chart

5.10 Step9. Profile Communication sequence

Local Device starts sending data to Remote Device with SCP (Sample Custom Profile).

By receiving Write Client Characteristic Configuration for permitting Notification, rBLE informs "RBLE_SCP_EVENT_SERVER_CHG_INDNTF_IND" event.

APP activates interval timer, and the timer generates INTIT interruption periodically. By receiving INTIT interruption, APP activates A/D converter. The converter generates INTAD interruption when finished converting. By receiving INTAD, APP calls "RBLE_SCP_Server_Send_Notify" function to send result value of converting by Notification.

By receiving Write Client Characteristic Configuration for inhibiting Notification, rBLE informs "RBLE_SCP_EVENT_SERVER_CHG_INDNTF_IND" event. APP inactivates interval timer to stop sending data.

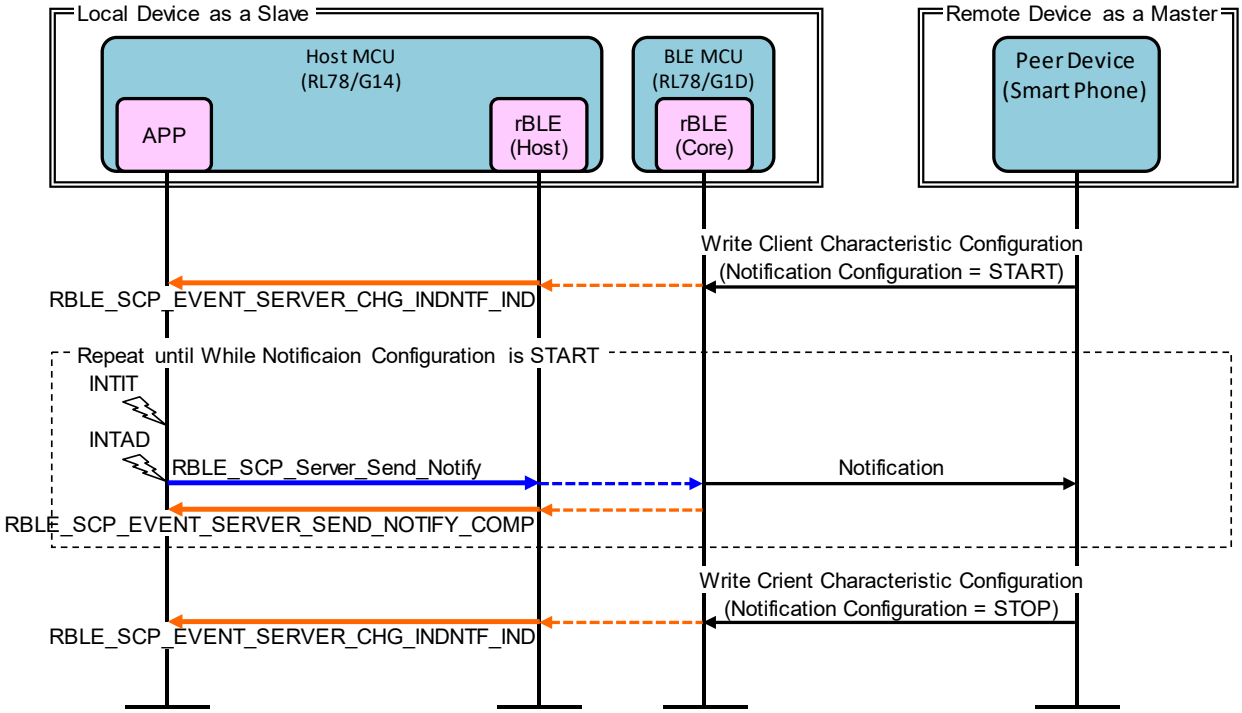

Figure 5-10 Profile Communication sequence chart

5.11 Step10. Disconnection sequence

Remote Device and Local Device are able to request disconnection.

By receiving Disconnect from Remote Device, rBLE disconnects connection and informs "RBLE_GAP_EVENT_DISCONNECT_COMP" event.

If INTP10 interruption occurs, APP calls "RBLE_GAP_Disconnect" function to send Disconnect to Remote Device. After disconnecting, rBLE informs "RBLE_GAP_EVENT_DISCONNECT_COMP" event.

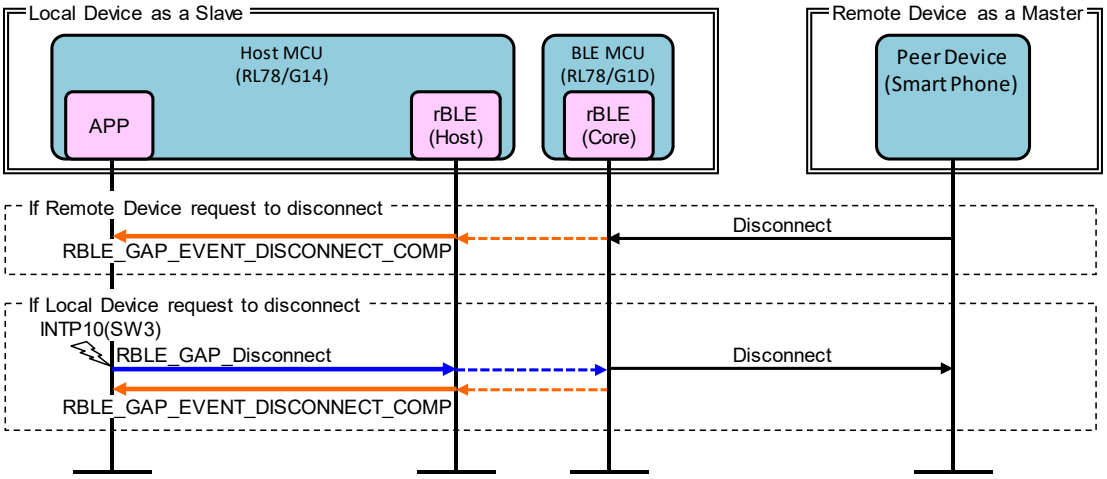

Figure 5-11 Disconnection sequence chart

6. Appendix

6.1 ROM size, RAM size

The ROM size and the RAM size which is used by host sample is shown in Table 6-1.

setting for build : Default setting of host sample released

- Define below macros
  - USE_RSK_LCD
  - USE_RSK_LED
  - USE_RSK_SW
  - USE_RSK_ADC
- Use Passkey as pairing method
- Use UART 2-wire

Table 6-1 ROM size, RAM size

Compiler	ROM (bytes)	RAM (bytes)
CC-RL	20,702	4,524
CA78K0R	29,625	4,736

6.2 References

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

6.3 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.
Connection Handle	This is the handle determined by the controller stack and is used to identify connection with a remote device. The valid handle range is between 0x0000 and 0x0EFF.
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.
Public Address	This is an address that includes an allocated 24-bit OUI (Organizationally Unique Identifier) registered with the IEEE.
Random Address	This is an address that contains a random number and belongs to one of the following three categories : Static Address Non-Resolvable Private Address Resolvable Private Address
Static Address	This is an address whose 2 most significant bits are both 1, and whose remaining 46 bits form a random number other than all 1’s or all 0’s. This static address cannot be changed until the power is switched off.
Non-Resolvable Private Address	This is an address whose 2 most significant bits are both 0, and whose remaining 46 bits form a random number other than all 1’s or all 0’s. Static addresses and public addresses must not be equal. This type of address is used to make tracking by an attacker difficult by changing the address frequently.
Resolvable Private Address	This is an address generated from an IRK and a 24-bit random number. Its 2 most significant bits are 0 and 1, and the remaining higher 22 bits form a random number other than all 1’s or all 0’s. The lower 24 bits are calculated based on an IRK and the higher random number. This type of address is used to make tracking by an attacker difficult by changing the address frequently. By allocating an IRK to the peer device, the peer device can identify the communicating device by using that IRK.
Broadcaster	This is one of the roles of GAP. It is used to transmit advertising data.
Observer	This is one of the roles of GAP. It is used to receive advertising data.
Central	This is one of the roles of GAP. It is used to establish a physical link. In the link layer, it is called Master.

Term	Description
Peripheral	This is one of the roles of GAP. It is used to accept the establishment of a physical link. In the link layer, it is called Slave.
Advertising	Advertising is used to transmit data on a specific channel for the purpose of establishing a connection or performing data transmission.
Scan	Scans are used to receive advertising data. There are two types of scans : Passive scan, in which data is simply received, and active scan, in which additional information is requested by sending SCAN_REQ.
White List	By registering known devices that are connected or bonded to a White List, it is possible to filter devices that can accept advertising data or connection requests.
Device Name	This is a user-friendly name freely assigned to a Bluetooth device to identify it. In the BLE standard, the device name is exposed to the peer device by the GATT server as a GAP characteristic.
Reconnection Address	If a non-resolvable private address is used and the address is changed frequently, not only attackers but also the peer device will have difficulty identifying the device. Therefore, the address to be used at reconnection is reported by setting a new reconnection address as the exposed reconnection address characteristic.
Connection Interval	This is the interval for transmitting and receiving data periodically following connection establishment.
Connection Event	This is the period of time during which data is transmitted and received at the connection interval.
Supervision Timeout	This is the timeout interval after which the link is considered to have been lost when no response is received from the peer device.
Passkey Entry	This is a pairing method whereby a six-digit number is input by each device to the other, or a six-digit number is displayed by one of the devices and that number is input to the other device.
Just Works	This is a pairing method that does not require user action.
OOB	This is a pairing method whereby pairing is performed by using data obtained by a communication method other than Bluetooth.
Identity Resolving Key (IRK)	This is a 128-bit key used to generate and resolve resolvable private addresses.
Connection Signature Resolving Key (CSRK)	This is a 128-bit key used to create data signatures and verify the signature of incoming data.
Long Term Key (LTK)	This is a 128-bit key used for encryption. The key size to be used is the size agreed on during pairing.
Short Term Key (STK)	This is a 128-bit key used for encryption during key exchange. It is generated using TK.
Temporary Key (TK)	This is a 128-bit key required for STK generation. In the case of Just Works, the TK value is 0. In the case of Passkey Entry, it is the 6-digit number that was input, and in the case of OOB, it is the OOB data.

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jun 30, 2015	—	First edition issued
1.01	Oct 30, 2015	10 16	section 2.4 File Composition : File tree modified Table 3-3 pin connection : pin number of RSK modified
1.10	Aug 26, 2016	- 4 6 17 24 36	Change document title 1 Overview : Add the support of UART 2-wire with branch connection. 2.2 Software Composition : Add the Sample Custom Profile. 3.1.5 UART Connection Method Setting : Add the setting method of source program. 4.3 UART 2-wire with Branch Connection : Add the operation explanation. 6.1 ROM size, RAM size : Add the UART 2-wire with branch connection.
1.20	Oct 27, 2016	10 12 20 22 33 36	2.4 File Composition : Update. 3.1 Preparation : Add the procedure for each development environment. 3.2.2 iOS Device : Change application. 3.3.2 Low Level Peripheral Driver Update By Code Generation Tool : Newly Added. 5.9 Step8. Start Encryption sequence : Change the sequence to accommodate BLE Protocol Stack V1.20. 6.1 ROM size, RAM size : Add the size for each development environment.
1.21	Oct 20, 2017	18	3.2.1 Android Device : Use GATTBrowser for the test application.
1.21	Jan 31, 2022	-	Fixed due to the end of IAR support in Bluetooth Low Energy Protocol Stack.

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

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation 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